I-AGENT : A System For Multi-agent Interoperability

Salah EL. Mohammed , Ahmed M Elmisery
Computer Science Department
Technical College ,Tanta University ,Egypt

Abstract

This paper presents I-agent system (Intelligible agent system) to improve multi-agent system (MAS) interoperability. It employs ontology management system and its inference ability to assist semantic translation of message content. Also, it uses Java management extensions to manage and integrate heterogeneous multi-agent systems. Besides, the intelligible service description is provided to support service matching. The I-agent system can:

- deal with different content languages, and apply semantic translation of content. This achieves semantic compatibility.
- support non-FIPA compliant MASs, and makes them interoperate with FIPA compliant MASs.
- supply reliable service identification.

Keywords ontology, multi-agent system, interoperability.

1 Introduction

Since established in 1996, the foundation for intelligent physical agents (FIPA) has released standards aiming for the interoperability of heterogeneous multi-agent systems (MASs). However, with implementation considerations (such as easy to develop or specific features in agent communication language (ACL)), and further, agent is an active research area, the standards usually cannot catch up the rapid advance of agent technology [1]. Thus, it is not possible for all MASs to follow FIPA standards. This makes MAS interoperability become a difficult issue. [2] categorizes the MAS communication compatibility into three layers:

1. technical compatibility :- we have to ensure that two systems are able to establish communication (at the lowest level, e.g. through network protocols) and exchange messages.

2. syntactic compatibility:- we have to make sure that both systems use the same syntax of messages, i.e. that both systems are able to parse messages from each other and to recognize syntactical elements.

3. semantic compatibility:- In this layer we want systems actually understand the content of messages, i.e. to understand the meaning of messages.

FIPA Message Transport specifications [3] enable FIPA compliant MASs to establish connection and exchange messages in FIPA format. [4] uses the CORBA bridge to make heterogeneous MASs interoperable. Both achieve technical compatibility. On the other hand, systems, such as [16], deal with message format only achieve syntactic compatibility. However, none of the above deals with message content (that is, semantic incompatibility), which is apt to cause misunderstanding making MASs unable to interoperate correctly. Another key point of interoperability is to know which MASs a MAS can interoperate with. That is, it considers whether or not to interoperate after knowing other MASs’ locations and their services. The possible solutions can be: 1) use SuperAgent [5] to represent all other MASs’ services or 2) use directory facilitator to federate distributed MASs.To improve MASs interoperability, the followings need to be addressed:

1. Semantic translation of content:

1.1. Content language specifies how message content is encoded. Several possible languages can be used, such as semantic language (SL), constraint choice language, knowledge interchange format, resource description framework (RDF) by FIPA [6], DAML+OIL, Prolog [7], ebXML [8] and OWL [9].

1.2. Ontology defines vocabulary used by content and relationship among terms in vocabulary. It can be distinguished into non-web based (such as [10]) and web based ontology. Ontology may results in mismatch. Granularity mismatch is the level of detail to which domain is modeled [11].Synonym terms mismatch on the other hand, refers to using different names to represent the same concept [11]. Significantly, the mismatch can be solved by ontology mapping[11].

Therefore, in order to achieve above-mentioned semantic compatibility, 1) various content languages and 2) mismatched ontology must be considered.
2. non-FIPA compliant MASs support:

Non-FIPA compliant MASs cannot interoperate with FIPA compliant ones, as pointed out earlier. Nevertheless, if semantic compatibility can be assured as depicted in "1" above, then resolving the differences in message transport and in ACL will solve the interoperability problem.

3. Service registry and searching among MASs:

MAS must know some information about other MASs, such as locations and states of agents or services they offer before they can interoperate with each other. However, registry content and searching policy are quite different among heterogeneous MASs. Those differences thus must be eliminated.

Therefore, this paper presents I-agent system (Intelligible agent system) to facilitate MAS interoperation. It employs ontology management system (OMS) and its inference ability to assist semantic translation of message content. Also, it uses Java management extensions [12] to manage and integrate heterogeneous MASs. Besides, the Intelligible service description is provided to support service matching. The remainder of this paper is organized as follows: Section 2 discusses related work. Section 3 depicts the architecture and the implementation. Finally, Section 4 draws conclusions.

2. Related Work

This section compares I-agent system with Ontology-Service Agent [13], [14] (OSA) and [16], [17] from two perspectives: 1) communication capability and 2) desired service registry and searching, respectively.

1) Communication capability:

OSA implements an ontology agent that offers ontology service [15] in a MAS. The agent monitors only those agents which are in conversation, and offers them assistance when necessary. Thus, it is not required to "map" ontologies of all the agents in the MAS. Although semantic translation of content is achieved, the ontology agent only assists the agents within the same MAS, but not those outside.

[16] and [17] integrate (interoperate) FIPA compliant MASs, and map all the ontologies before runtime to provide semantic translation. However, the system can only integrate FIPA compliant MASs. In other words, it assumes that all MASs use the same ACL format, and all follow FIPA message transport mechanism.

Conversely, our I-agent system does not presume all MASs use the same ACL format and the same message transport mechanism. Through 1) semantic translation of content by the OMS and 2) MAS interface, I-agent system achieves semantic compatibility among heterogeneous MASs.

2) Desired services registry and searching:

OSA is used in single MAS only. Therefore, it does not have such service registry and searching problems. Consequently, it cannot make use of other MASs’ services.

[16] and [17] define an ontology to describe MASs’ capabilities and location information. The matching engine can match request to advertisement based on that ontology. Thus service matching is provided. Whenever directory facilitator (DF) could not find desired services, it will ask matching engine to help. This achieves transparent interoperability. However, the prerequisite is: all MASs must be FIPA compliant, and their DFs must be implemented in accordance with [16] and [17] definition.

On the other hand, our I-agent system use:

1) intelligible service description (ISD) to describe the MASs’ service information to provide service matching

2) MAS interface which allows heterogeneous MASs to implement their service searching according to their requirements, such as adopting SuperAgent [5] to achieve transparent interoperability. Therefore, I-agent system can register distributed, heterogeneous MASs’ agents and services information, and can provide service matching to make service searching more reliable.

3. Intelligible-agent System

Sections 3.1 and 3.2 respectively describe the
architecture and implementation of Intelligible-agent System (I-agent system)

3.1 Architecture
Figure 1 illustrates the architecture of I-agent system, dividing into three parts: 1) external interfaces, 2) core components and 3) integration tier.

1) **External interfaces:** The interfaces include SOAP, RMI and HTTP connectors so that MASs or application programs can interact with the I-agent.
2) **Core components:** The core part of I-agent, including component loader (CL), ontology analyzer (OA), content language builder (CLB) and agent platform mediator (APM).

   1. CL loads core components, such as APM and OA, during I-agent initialization time. After I-agent started, CL will load other components which are to be deployed, such as MASs or OMSs. Furthermore, CL can dynamically deploy new components at runtime.
   2. OA processes message content by the following steps:
      - Extracting action, actor, service name, arguments and/or other information from message content.
      - Matching extracted information with the ISD.
      - If there are matched services or agents, but the ontologies used by the matched and those by initiator mismatch, OA will use OMS to assist semantic translation of content.
      - Finally, give the translated result to CLB.
   3. After receiving the translated terms from OA, CLB builds the appropriate message content in accordance with the content language format described by meta ontology (MO), and then give the built result to APM.
   4. APM monitors, registers and mediates deployed MASs. It also delivers built content from CLB to MAS interfaces containing the matched agents. And it will send the service results back to external interfaces.
3) **Integration tier:** Including OMS interface and MAS interface.

   - OMS interface integrates OMSs into I-agent and handles all requests about OMS accesses.
   - MAS interface integrates heterogeneous MA$S$s into I-agent. It encodes the content from APM into specific ACL, and uses specific message transport mechanism to interoperate with MASs. It also receives the MASs’ replies, and gives them back to APM.

Additionally, all MASs integrated into I-agent must provide: 1) a deployment file, 2) an Intelligible service description (ISD) and 3) a MO defining by I-agent. 1) CL deploys a MAS into I-agent in accordance with its deployment file. 2) ISD extends FIPA’s directory facilitator agent description by adding extra profiles. For example, ISD includes agent states, agent services, service profiles and ontologies used by agents. I-agent stores ISD in OMS, and then performs service matching. 3) MO describes the structure of content language, helps OA to extract information from content and assists CLB to build content.

3.2 Implementation:
At present, I-agent integrates Java agent development framework (JADE) [18], open agent architecture (OAA) [19] and JATLite [20], making them interoperate with each other. The I-agent implementation is described below:

   1) I-agent follows JMX [12] to define: 1) MAS interface and 2) OMS interface.

   - MAS interface declares methods which may be called among agents’ interoperation, such as request or query (naming after FIPA communication act [21]). Every implementation of MAS interface must be able to generate the ACL corresponding to called method name, excluding content part. For example, if APM calls the “request” method of FIPA compliant MAS interface, this interface must be able to generate an ACL message using “REQUEST” as performative.
   - OMS interface wraps Sesame [22] into I-agent. The OMS stores MO, ISD and the ontologies used by MASs.
2) *I-agent* uses OWL to define: 1) MO and 2) ISD.
   - Owing to not all content languages can be described by MO, such as non-XML based content languages. Therefore, MO can define a downloadable URL containing a Java bytecode file which can process those content languages, and then *I-agent* uses Java class loader to load that file to assist semantic translation of content. Furthermore, that file can make use of content language translation service [23] to assist translation.
   - Since some MASs does not provide their ontologies usage information. They even extremely use ontologies implicitly encoded with the implementation of the agents, such as OAA. Therefore, ISD must be able to describe the ontologies used by those kinds of MASs. And then *I-agent* can load those ontologies into OMS.

4. Conclusions
This work presents the *I-agent* system which can facilitate heterogeneous MASs to interoperate and ensure semantic correctly. *I-agent* has three benefits:

1. Semantic translation of content:
   - The meta ontology describes different structures of content languages, which allows *I-agent* to: 1) extract service name and other information, such as terms which needs translation, from contents encoded in different content languages, 2) build content encoded in different content language formats.
   - The OMSs store the ontologies used by MASs. *I-agent* uses OMS to do semantic translation of content.
   Therefore, *I-agent* can deal with different content languages, and can apply semantic translation of content. This achieves semantic compatibility.

2. non-FIPA compliant MASs support:
   - MAS interface builds heterogeneous MASs’ ACL, and uses specific message transport mechanism to interoperate with other MASs. That is, *I-agent* integrates MASs.
   - Through SOAP, non Java-based MAS still can make use of *I-agent*. Therefore, *I-agent* supports non-FIPA compliant MASs, and makes them interoperable with FIPA compliant MASs.

3. Reliable service identification:
All MASs deployed on *I-agent* use Intelligible service description to declare their services. *I-agent* stores these information into OMS, and then provides service matching. Therefore, *I-agent* supplies more reliable service identification.

References
[8] Luis Botelho and Steven Willmott, Review of Content Languages Suitable for Agent-Agent Communication.


[11] Ying Ding, Dieter Fensel, Michel Klein and Borys Omelayenko: *Ontology management: survey, requirements and directions*


[22] Sesame RDF database, Web site. [https://www.openrdf.org](https://www.openrdf.org)

[23] Content Language Translation Service ,Web site. [https://clts.we-b-mind.org](https://clts.we-b-mind.org)