SEAGENT: A Platform for Developing Semantic Web Based Multi Agent Systems

Oguz Dikenelli
Ege University
Computer Engineering
Department
35100, Bornova, Izmir, Turkey
oguzd@staff.ege.edu.tr

Riza Cenk Erdur
Ege University
Computer Engineering
Department
35100, Bornova, Izmir, Turkey
erdur@staff.ege.edu.tr

Ozgur Gumus
Ege University
Computer Engineering
Department
35100, Bornova, Izmir, Turkey
gumus@staff.ege.edu.tr

ABSTRACT

In this paper, a new agent development platform, which includes built-in features for semantic web based multi agent system development, is introduced. All agents and services in the platform use semantic web standards to represent their internal knowledge and semantic web query languages are used to query them. Semantic web services can be discovered and dynamically invoked. The directory service is implemented in a way to support semantic matching of agent capabilities. The ontology service allows ontology translation based on defined mappings between platform ontologies and external ontologies. With these features already integrated, the agent development platform that is developed simplifies the semantic web based multi-agent system development.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Design

1. INTRODUCTION

Semantic web and agent research are evolving together. Semantic web research aims to transform the World Wide Web into a knowledge representation system in which the information provided by web pages is interpreted using ontologies. This gives the opportunity for autonomous computational entities - agents - to collect and interpret semantic content on the behalf of their users.

In this paper, we introduce SEAGENT, which is a new agent development platform that is specialized for semantic web based multi agent system development. The communication and plan execution infrastructure of SEAGENT looks like other existing agent development frameworks such as DECAF [3] , JADE [1] . To support and ease semantic web based multi agent system development, SEAGENT includes the following built-in features that the existing agent frameworks and platforms do not have:

i) Agents created using SEAGENT handle their internal knowledge base using semantic web standards and the platform provides specifically designed interfaces to manage and query the internal knowledge without being dependent on a particular application programming interface.

ii) The directory service of SEAGENT is implemented in a way that the directory knowledge is held in semantic web standards and the directory service supports semantic matching of the agent capabilities to find the semantically similar agents.

iii) FIPA-RDF content language has been used to transfer semantic content in the agent communication language messages and OWL-QL [2] is integrated to the FIPA-RDF content language to query the agents and services.

iv) SEAGENT introduces a new service for managing and translating ontologies. It provides a means to define mappings between platform ontologies and external ontologies. The translation process is based on these defined mappings.

v) SEAGENT supports discovery and dynamic invocation of semantic web services by introducing a new platform service for semantic service discovery and a reusable agent behavior for dynamic invocation of the discovered services.

These built-in features will be explained in the following sections. Section 2 discusses the overall architecture of SEAGENT.

2. PLATFORM ARCHITECTURE

The layered software architecture of SEAGENT is shown in Figure 1. In the following, we briefly discuss each layer with an emphasis on the semantic support given by that layer.

The bottom layer is responsible of abstracting platform's communication infrastructure implementation. SEAGENT implements FIPA's Agent Communication and Agent Message Transport specifications to handle agent messaging. Although Communication Infrastructure Layer can transfer any content using FIPA ACL and transport infrastructure, SEAGENT platform only supports FIPA RDF content language since it is very suitable to transfer semantic web enabled content.

The second layer includes packages, which provide the

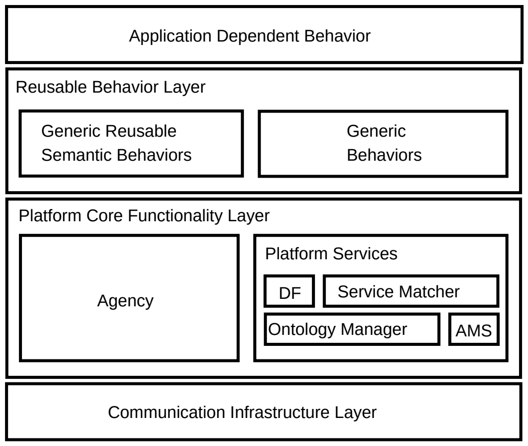
Application Dependent Behavior
Reusable Behavior Layer
Generic Reusable Semantic Behaviors
Generic Behaviors
Platform Core Functionality Layer
Agency
Platform Services
DF
Service Matcher
Ontology Manager
AMS
Communication Infrastructure Layer

Figure 1: SEAGENT Platform Overall Architecture

core functionality of the platform. The first package, called as Agency, handles the internal functionality of an agent. Agency package provides a built-in 'agent operating system' that matches the goal(s) to defined plan(s), which are defined using HTN planning formalism [4] . It then schedules, executes and monitors the plan(s). From semantic web based development perspective, an agent's internal architecture must support semantic web ontology standards for messaging and internal knowledge handling to simplify semantic based development. For this purpose, Agency package provides a build-in support to parse and interpret FIPA RDF content language to handle semantic web based messaging.

The second package of the Core Functionality Layer includes service sub-packages, one for each service of the platform. These services follow the FIPA standards but they are implemented differently using the capabilities of a semantic web infrastructure. In the SEAGENT implementation, DF uses an OWL ontology to hold agent capabilities and includes a semantic matching engine to be able to return agent(s) with semantically similar capabilities to the requested ones. Similarly, AMS stores agents' descriptions in OWL using FIPA Agent Management Ontology and can be queried semantically to learn descriptions of any agent that is currently resident on the platform.

Besides implementing standard services in a semantic way, SEAGENT platform provides two new services to simplify semantic web based MAS development. The first one is called as Semantic Service Matcher (SSM). SSM is responsible for connecting the platform to the semantic web services hosted in the outside of the platform. SSM uses 'service profile' construct of the Web Ontology Language for Semantic Web Services (OWL-S) standard for service advertisement and this knowledge is also used by the internal semantic matching engine for discovery of the service(s) upon a request. The second unique service is the Ontology Manager Service (OMS). The most critical support of the OMS is its translation support between the ontologies. Through the usage of the ontology translation support, any agent of the platform may communicate with MAS and/or services outside the platform even if they use different ontologies.

Third layer of the overall architecture includes pre-prepared generic agent plans. We have divided these generic plans into two packages. Generic Behavior package collects domain independent reusable behaviors that may be used by any MAS such as well known auction protocols (English, Dutch etc.). On the other hand, Generic Semantic Behaviors package includes only the semantic web related behaviors. In the current version, the most important generic semantic behavior is the one that executes dynamic discovery and invocation of the external services. This plan is defined as a pre-prepared HTN structure and during its execution, it uses SSM service to discover the desired service and then using OWL-S 'service grounding' construct it dynamically invokes the found atomic web service(s). Hence, developers may include dynamic external service discovery and invocation capability to their plan(s) by simply inserting this reusable behavior as an ordinary complex task to their HTN based plan definition(s).

3. CONCLUSION

The platform's overall architecture has been implemented and it is operational. To evaluate the platform, we have developed an experimental but a realistic application in tourism domain. The semantic features are tested in different scenarios that make up the application. The main scenario aims of finding the right hotel agent with the cheapest price. In this scenario, the agents use different domain ontologies. The ontology handling capability of the platform makes the usage of ontologies easy for the developer. In the application, hotel agents that use different ontologies are modeled to evaluate the ontology translation capability of the platform. We observed that built-in ontology translation service simplifies the development of MAS where multiple ontologies exist.

4. ACKNOWLEDGMENTS

This work is supported in part by the Scientific and Technical Research Council of Turkey, Project Number: 102E022.

5. ADDITIONAL AUTHORS

Additional authors: Erdem Eser Ekinci (Ege University, email: `erdemeserekinci@hotmail.com`), Önder Gürcan (Ege University, email: `gurcan@staff.ege.edu.tr`), Geylani Kardaş (Ege University, email: `geylani@bornova.ege.edu.tr`), İnanç Seylan (Ege University, email: `seylan@staff.ege.edu.tr`) and Ali Murat Tiryaki (Ege University, email: `tiryaki@staff.ege.edu.tr`).

6. REFERENCES

[1] F. Bellifemine and G. Rimassa. Developing multi-agent systems with a FIPA-compliant agent framework. *Softw. Pract. Exper.*, 31(2):103–128, 2001.

[2] R. Fikes, P. Hayes, and I. Horrocks. OWL-QL: A language for deductive query answering on the semantic web. Technical Report KSL 03-14, Stanford University, Stanford, CA, 2003.

[3] J. R. Graham, K. Decker, and M. Mersic. DECAF - A flexible multi agent system architecture. *Autonomous Agents and Multi-Agent Systems*, 7(1-2):7–27, 2003.

[4] M. Paolucci, O. Shehory, K. P. Sycara, D. Kalp, and A. Pannu. A planning component for RETSINA agents. In *ATAL '99: 6th International Workshop on Intelligent Agents VI, Agent Theories, Architectures, and Languages (ATAL),*, pages 147–161, London, UK, 2000. Springer-Verlag.