Exposing agents as web services in JADE

Orso Negroni[1], Anthony Othmani[1], Arthur Casals[2], and Amal El Fallah Seghrouchni[1]

[1] Sorbonne Université, LIP6- CNRS UMR 7606
4 place Jussieu, 75005 Paris - France
amal.elfallah@lip6.fr, orso.negroni@etu.upmc.fr,
anthony.othmani@gmail.com
[2] Escola Politécnica - Universidade de São Paulo
Av. Prof. Luciano Gualberto, 158 - trav. 3 - 05508-900 - São Paulo - SP - Brazil
arthur.casals@usp.br

Abstract. The objective of this demonstration is to show how intelligent agents using a BDI architecture can be exposed as web services and integrated with existing cloud services. We will use JADE and BDI4JADE to expose an intelligent agent as a service that can be integrated with different cloud-based services, such as Amazon AWS services and Google Home.

Keywords: Multi-Agents Systems · Web Service Agents · JADE/WSIG.

1 Introduction

Using agents in conjunction with the Semantic Web is not a novel idea [3, 7]. Additionally, model-driven development (MDD) techniques for developing agent models such as GAIA [9] and Tropos [1] can be used in the Semantic Web context [5] in order to allow the modelling of web-capable agents. Despite the extensive research already done in these fields, the advent of the Web of Things (WoT) [4] has encouraged a re-visitation on the use of agents in the context of the Semantic Web as a viable approach for deploying autonomous systems within the WoT [2].

Our research resides in the Agent-Oriented Software Engineering domain. During the study of existing agent modeling methodologies and their relationship with Web architectures, we decided to implement web-capable agents (exposed as web services) to serve as an evolving proof-of-concept, to be used during the course of our work.

With that in mind, in this demo session we will present a Smart Agenda multiagent system (MAS), which is built to function as an agent-based personal assistant. For example, if the user updates his agenda with a meeting in Brussels and he is in Paris, the Smart Agenda system is supposed to detect that the user needs a train to go from Paris to Brussels, and use a text-to-speech service in order to ask Google Home to book a train ticket. We will detail our implementation in the next paragraphs.

2 Framework

In this first phase of our project, our objective was to implement a MAS able to handle all the tasks associated with a specific event. While we identified existing work on integrating JADE agents and web services [6], we felt it was necessary to have a new implementation in face of the current state of all related technology. The main technologies and related concepts used in this implementation are described below:

- **Jade and WSIG:** Java Agent DEvelopment Framework (JADE)[1] is a FIPA[2]-compliant software framework implemented using the Java programming language. It also possess an add-on aimed at providing support for exposing agents as web services called WSIG[3]. This add-on acts as a relaying gateway, handling all requests coming from the Web and sending them to the agent-based system.
- **BDI4JADE [4]:** A Java package that implements the BDI (belief-desire-intention) agent architecture [8] on top of JADE. It allows the creation of BDI agents through the extension of provided classes. Since it is built on top of JADE, all BDI agents created through the use of this library are also JADE agents.
- **AWS Polly [5]:** An Amazon web service that provides text-to-speech features. It is used to turn text into speech that sounds like a human voice.

3 Smart Agenda

In this section we present the Smart Agenda, a MAS aimed at providing smart event management for its users. With the aid of intelligent agents, events are scheduled and modified according to personal preferences and contextual conditions.

3.1 Features

The Smart Agenda possesses an web page used as an interface between the user and the MAS. After creating an account, the user can set personal preferences regarding transportation, hotel stars, and hours. The web page is used to schedule new events and to modify or view existing events. The system also allows the user to create events involving other users, or to join existing events created by other users. Events are classified either as ”individual events” or ”group events”.

Individual events are scheduled solely according to the user’s preferences. They possess two distinct properties, named ”Automatic” and ”Movable”. An automatic event can be scheduled by the agent arbitrarily along the day, according to the preset user preferences. A movable event can be rescheduled without

[1] http://jade.tilab.com/
[2] http://www.fipa.org/
[3] http://jade.tilab.com/doc/tutorials/WSIG_Guide.pdf
[4] http://www.inf.ufrgs.br/prosoft/bdi4jade/
[5] https://aws.amazon.com/polly/

the user's confirmation if it's necessary. Thus, if an automatic event is created, the agent will try to find an available time slot in the agenda according to the user's preferences. If there is not a free slot long enough to accommodate it, the agent will try to reorganize other movable events to optimize the day.

Group events are events shared by two or more users. They are non-movable by default. When a user creates an event, he can assign it to an existing users group. Group events can be marked as "Optional", meaning that the attendance for all users is not mandatory. If at least two users within the group are available at the scheduled time, the event will be created and added to these users' agenda. In the case the event is not marked as "Optional", the attendance is considered mandatory for all participants - meaning that if at least one user is not available at the scheduled time the event will not be created.

3.2 Architecture

The MAS is composed by four different agents: Coordinator, Manager, Agenda, and Assistant. The Coordinator is responsible for handling all requests from the Web. When a new user account is created, the request is forwarded to an available Manager agent, which creates two new agents linked to this user: the Agenda agent, and the Assistant agent. In order to provide scalability to the system each Manager agent is responsible for a limited number of users. The Agenda agent is responsible for processing all future event operations for the new user, and the Assistant is the BDI agent responsible for processing complementary actions related to an event (such as booking a ticket) and generating the final sentence with the help of AWS Polly. Consequently, there are two agents (Assistant and Agenda) for each user registered in the system.

When a new event is created in the user's calendar, the Coordinator forwards it to the corresponding Manager, which then sends it to user's Agenda agent. This event is added to the agenda and, if this is an individual event, the Assistant agent creates its related Goals. A plan will be selected from an existing plan library according to the user's preferences and context-dependent feasibility. If it fails, the next one will be tried. If the event contains the keywords "rdv", "rendez vous" or "meeting", and "@SomeTown", the resulting plan will be a sentence containing all the event's elements: the date, the time, the starting and arrival towns. This sentence is then translated into audio (speech) through the use of the AWS Polly service, and finally played to Google Home with the help of speakers. This architecture is shown in Figure 1, and its demonstration can be found at: http://bit.ly/Emas18Demo

4 Discussion

As previously stated, this project is an evolving proof-of-concept. In this first phase, we found a few difficulties related to the implementation (mostly related to WSIG), but the MAS architecture was simple enough to be modelled using a traditional BDI architecture. At the moment, the BDI model is limited to

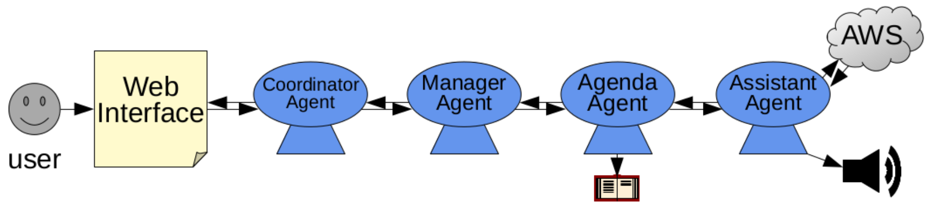

Fig. 1. Implementation architecture for the Smart Agenda agents

individual events and sequential goals (context dependency and AWS integration). The next steps reside in (*i*) expanding the BDI model (taking geographical constraints and group preferences into consideration), (*ii*) the original related research (architectural patterns for web-capable agents and related interaction protocols, as well as their relationship with existing agent modelling methods) and (*ii*) the implementation itself (comparing non-agent and agent-based approaches, automating the ticket booking process, improve shared events).

References

1. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An agent-oriented software development methodology. Autonomous Agents and Multi-Agent Systems **8**(3), 203–236 (2004)
2. Ciortea, A., Boissier, O., Ricci, A.: Beyond physical mashups: Autonomous systems for the web of things. In: Proceedings of the Eighth International Workshop on the Web of Things. pp. 16–20. ACM (2017)
3. Dickinson, I., Wooldridge, M.: Agents are not (just) web services: considering bdi agents and web services. In: Proceedings of the 2005 Workshop on Service-Oriented Computing and Agent-Based Engineering (SOCABE'2005), Utrecht, The Netherlands (2005)
4. Guinard, D., Trifa, V., Wilde, E.: A resource oriented architecture for the web of things. In: Internet of Things (IOT), 2010. pp. 1–8. IEEE (2010)
5. Kardas, G., Goknil, A., Dikenelli, O., Topaloglu, N.Y.: Model driven development of semantic web enabled multi-agent systems. International Journal of Cooperative Information Systems **18**(02), 261–308 (2009)
6. Nguyen, X.T., Kowalczyk, R.: Ws2jade: Integrating web service with jade agents. In: Huang, J., Kowalczyk, R., Maamar, Z., Martin, D., Müller, I., Stoutenburg, S., Sycara, K.P. (eds.) Service-Oriented Computing: Agents, Semantics, and Engineering. pp. 147–159. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)
7. Ricci, A., Denti, E., Piunti, M.: A platform for developing soa/ws applications as open and heterogeneous multi-agent systems. Multiagent and Grid Systems **6**(2), 105–132 (2010)
8. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach. Pearson Education, 2 edn. (2003)
9. Wooldridge, M., Jennings, N.R., Kinny, D.: The gaia methodology for agent-oriented analysis and design. Autonomous Agents and multi-agent systems **3**(3), 285–312 (2000)