An Ambient Intelligence Application Integrating Agent and Service-Oriented Technologies

Nikolaos Spanoudakis[a, b] Pavlos Moraitis[b]
[a] Singular Logic S.A.
nspan@singularlogic.eu
www.singularlogic.eu
[b]Paris Descartes University
{nikos, pavlos}@math-info.univ-paris5.fr
www.math-info.univ-paris5.fr

Abstract

This paper presents an agent-based approach into a more general service oriented architecture for addressing the requirements of accessibility content and services in an ambient intelligence context. The developed agent-based information system provides infomobility services for the special requirements of mobility impaired people. Herein, we focus in the task of integrating this multi-agent system in the overall service-oriented architecture. In order to achieve this task we propose a methodology for integrating a FIPA-compliant agent platform with the OSGi service oriented framework.

1. Introduction

Agent technology has been applied to the infomobility services sector in recent years (e.g. the Im@gine IT [6] project). Such services include location-based services like mapping and points of interest search, travel planning and, recently, trip progression monitoring and pushing information and events to the user. A recent research has proposed that elderly and disabled people compose a segment of the population that would profit very much from ambient intelligence (AmI), if the latter was accessible [1]. Furthermore, O'Hare et al. [8] advocate the use of agents as a key enabler in the delivery of ambient intelligence. Thus, in an AmI framework for servicing elderly and disabled (mobility impaired people) the role of agent technology is crucial.

In this paper we present a part of the work proposed in the Integrated Project (IP) "Ambient Intelligence System of Agents for Knowledge-based and Integrated Services for Mobility Impaired users" (ASK-IT, IST-2003-511298), which aims to offer infomobility services to mobility impaired people and support them while on the move. We briefly present the part of the agent-based system that is related to the integration of ambient intelligence in personal travel assistance and focus in showing how we integrate an agent platform compliant to the FIPA standard (Foundation for Intelligent Physical Agents, http://www.fipa.org) with the OSGi

(Open Service Gateway initiative, http://www.osgi.org) service oriented middleware [4].

The rest of the paper is organized as follows: In section 2 we provide a brief presentation of the multi-agent system (MAS) that we conceived for addressing the ASK-IT challenges with regard to ambient intelligence and introduce the need for the integration of the MAS in an overall service oriented architecture (SOA). Following, we firstly present the methodology for integrating the FIPA-compliant agent platform in the OSGi framework in section 3, and then, the integrated architecture for the ASK-IT server and client in section 4. Finally, we conclude in section 5.

2. The Multi-Agent System

The background on relevant agent architectures is the FIPA Personal Travel Assistance [5] standard and the results of the Im@gine IT project [6] that addressed open issues defined by FIPA [5]. The JADE-Leap (Java Agent Development Environment – Lightweight Extensible Agent Platform, http://jade.tilab.com) framework allows for implementing agents on nomad devices like PDAs, laptops and smart phones. Taking all the above into consideration, the proposed ASK-IT architecture is an evolution of the Im@gine IT architecture in two ways:

a. it proposes a server side dynamic coalition formation of specialised agents aiming to serve users with more than one types of impairments (this issue is beyond the scope of this paper and is discussed in [7])
b. it integrates ambient intelligence through the introduction of two new types of agents on the user's device (other than the classical personal assistant type), and,

The family of agents that integrate ambient intelligence includes the following:

- The *Personal Wearable Intelligent Device Agent (PEDA*, acting as a FIPA Mini-Personal Travel Assistant for persons with impairments) that provides the personalized infomobility services to the user
- *Ambient Intelligence Service Agent (AESA)* that configures the environment of the user according to his habits/needs (new type of agent)
- The *Personal Wearable Communication Device Agent (PWDA)* that monitors the user's sensors and provides information either directly to the user or the Personal Wearable Intelligent Device agent in cases of emergency (new type of agent)

In ASK-IT not all cooperating software components are agents, therefore one main challenge of our work was related to the way to connect all the identified modules and agents. Such modules are the *Localization Module* that produces accurate coordinates of the user, the *Navigation Module* that navigates the user to its destination, the *User Interface Module,* the *Domotic Services Module* that allows the user to control and monitor devices in his household (heater, air-condition, etc).

The Open Services Gateway initiative (OSGi) technology provides a service-oriented, component-based environment for developers and offers standardized ways to manage the software lifecycle. It was chosen for integrating all the participating components because of its offering the following possibilities:

a. Firstly, in the same virtual machine, bundles (that is how the OSGi components are named) can be installed, started, stopped, or uninstalled, dynamically on runtime. This feature allows for the best utilization of resources of nomad devices (low computing power).
b. Secondly, a bundle can import but also export java packages when installed. In Java, there is normally a single classpath that contains all the available packages and resources. The OSGi framework caters for controlled linking between different software modules, as they are installed on runtime.
c. Finally, the bundles can locate and invoke services offered by other bundles. This allows for dynamic interoperability between our different components. Services definition is easy and intuitive, as it is based on common Java interfaces definition. Then, these interfaces' implementation classes are started by an OSGi mechanism that undertakes the task of advertising the implemented interfaces in the framework. Then, the OSGi Application Programming Interface (API) can be used for locating and invoking the services.

3. The Integration Methodology

The proposed integration of the FIPA agent platform and OSGi should satisfy the following goals:

- Provide an architecture where agents can simultaneously execute in both the agent platform where they are created and the OSGi architecture integrating the above platform. Any agents that would normally execute in the selected agent platform can also execute normally in the context of the platform's integration in the OSGi architecture.
- Allow the agents to use the OSGi services in an agent instance independent manner. For example, software agents invoke existing web services in a standardized way, having the only prerequisites that the agent has access to the internet and that the service is available. Following this paradigm, the agent should have access to existing OSGi services with the only prerequisites that the agent's platform is integrated with an OSGi platform and the services are available.
- Allow the agents to offer themselves services to the OSGi framework in order to seamlessly integrate with different software modules.

In order to realize the above goals we integrated the agent platform in the OSGi framework and instantiated it as an OSGi service. In Figure 1 the reader can see the operating system on top of which a Java virtual machine executes and the OSGi framework. Different bundles are instantiated, one of which is the agent platform bundle. This approach is more efficient because the OSGi framework offers an environment where any kind of application can execute.

The process for integrating an agent platform into the OSGi SoA framework can be described as follows:

a. Define the ontology that will be common for all bundles and used for defining the services signatures (in the case that all the signatures will use simple Java classes, like Integer and String, then there is no need for an ontology)
b. Define the java interfaces for the services that will be offered by each bundle (all interfaces definitions may import the ontology bundle)
c. Define the different bundles, each implementing the relevant interfaces. All bundles dynamically import the ontology and service descriptions bundles. One of these bundles will be the multi-agent system bundle
d. Select the architecture for directing events from the MAS bundle to the different agents

The final step is relevant to selecting a method for communication between objects and agents. In this case, the multi-agent system bundle instance needs to communicate with the agents. In the JADE-Leap documentation one can find such tools, like in the `SocketProxyAgent` package that defines a JADE agent that can be executed to communicate with remote clients via a plain TCP/IP socket. This choice is especially convenient for nomad device applications, where, due to limited available resources, only basic Java libraries, such as plain TCP/IP sockets, are allowed to execute.

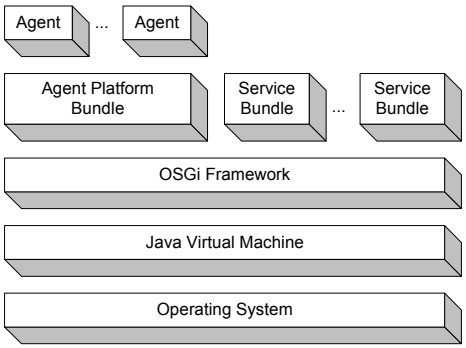

Figure 1: Computing environment structure (five layers)

After a developer has followed this process he can start defining his agents using OSGi services for sensing and acting on his environment, but, also for tracking changes in the environment. Such changes reflect the sudden availability or non-availability of a specific service and OSGi offers the possibility for informing interested bundles about the appearance of specific services.

4. Ambient Intelligence Architecture

In this section we show, as an example, how precisely the OSGi – agent platform integration can provide an efficient solution for the development of ambient intelligence applications, which is our case. Such an application poses a specific limitation on the resources that can be employed on the nomad device but also on

the capability for communication with the outside world. The protocol proposed in [2] and the execution of the JADE-Leap agents in split container mode can address these problems. The split execution mode means that the agent platform on the nomad device needs to depend on another agent platform that uses a static IP address somewhere on the internet.

In Figure 2 the reader can see the architecture for our application. The diagram type used is the UML deployment diagram (Unified Modeling Language, http://www.uml.org). The light grey packages show the different identified components, while the dark grey ones depict the open source libraries used, i.e. the knopflerfish OSGi framework (http://www.knopflerfish.org) and the JADE-Leap framework. The domain ontology (*JADE Ontology Beans* component) has been developed using the JADE ontology beans methodology (see [3]). This approach allows for integrating different component manufacturers' modules, some of which could even be application independent. For example, we used as a S*imple Service Bundle* a localization module that was developed by a third party manufacturer and which was distributed as an OSGi bundle. The *Localization, Navigation, Domotics, User Interface* modules were made available to the platform as OSGi bundles.

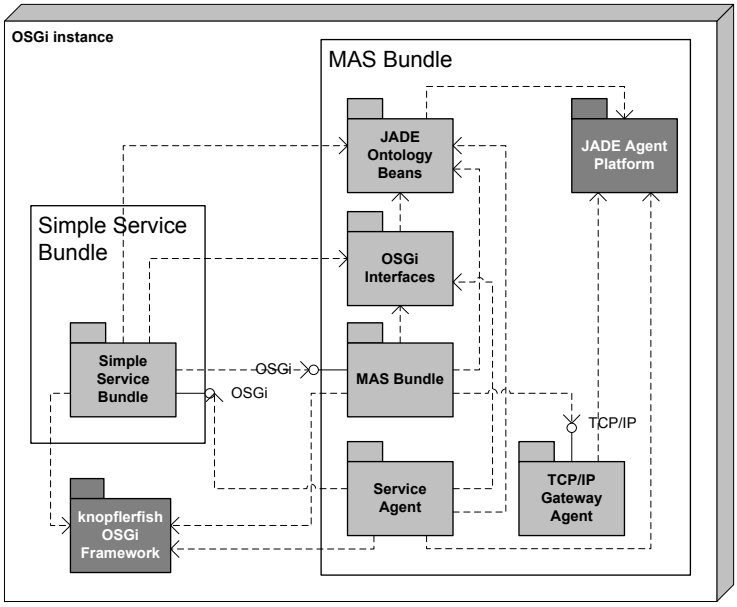

Figure 2: OSGi – agent platform integration architecture on a nomad device

The resulting ambient intelligence system can execute on any nomad device, i.e. it includes the possibility to execute on mobile phones that is the most restricted device type. Figure 2 shows the MAS Bundle and its various components. The *MAS Bundle* component is an implementation class of the MAS Bundle interface (that defines the services that the agents defined in the *Service Agent* component offer) that is included with all other bundles interfaces in the *OSGi Interfaces* component. The *TCP/IP Gateway Agent* gets requests by the *MAS Bundle* instance

using TCP/IP plain sockets, communicates with the interested *Service Agent* (e.g. the PEDA or PWDA agent) using ACL messages and replies through the original socket port to the MAS bundle.

The ontology and the OSGi interfaces are in the same bundle with the MAS because the ontology is dependent on the JADE library and the interfaces on the ontology. Also, the JADE architecture does not permit for the JADE files to be installed in a bundle and the agents to be launched by another. Therefore, this architecture demands that all those components be included in the same bundle.

5. Conclusion

In this paper we presented certain aspects of an agent-based system for the personal travel assistance (PTA) domain. We addressed the issue of integrating personal travel assistance applications with ambient intelligence and showed how to integrate an agent platform in a service oriented architecture. We provided a methodology and architectural guidelines that other developers can follow in order to develop their own ambient intelligence, agent-based applications.

References

1. Abascal, J.. Ambient Intelligence for People with Disabilities and Elderly People. ACM's Special Interest Group on Computer-Human Interaction (SIGCHI), Ambient Intelligence for Scientific Discovery (AISD) Workshop, Vienna, April 25, 2004
2. Caire, G., Lhuillier, N. and Rimassa G., A communication protocol for agents on handheld devices. Proceedings of the first International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2002), Bologna, Italy, July 15-19, 2002
3. Caire, G., Van Aart, C., Bergenti, F., Pels, R.. Creating and Using Ontologies in Agent Communication. Workshop on Ontologies and Agent Systems (OAS'2002) at AAMAS 2002, Bologna, Italy, July 16, 2002
4. Cervantes, H. and Hall, R.S., Autonomous Adaptation to Dynamic Availability Using a Service-Oriented Component Model. Proc. of the 26th Int. Conf. on Software Engineering (ICSE 2004), Scotland, May 2004.
5. FIPA: Personal Travel Assistance Specification. Foundation for Intelligent Physical Agents, XC00080B, http://www.fipa.org, 2001
6. Moraitis, P., Petraki, E. and Spanoudakis, N., An Agent-Based System for Infomobility Services. The third European Workshop on Multi-Agent Systems (EUMAS2005), Brussels, Belgium, December 7 - 8, 2005
7. Moraitis, P., Spanoudakis N.. Argumentation-based Agent Interaction in an Ambient Intelligence Context. IEEE Intelligent Systems, Special Issue on Argumentation Technology, Nov/Dec '07, 2007 (to appear)
8. O'Hare, G. M. P., O'Grady, M. J., Keegan, S., O'Kane, D., Tynan, R. and Marsh, D.. Intelligent Agile Agents: Active Enablers for Ambient Intelligence. ACM's Special Interest Group on Computer-Human Interaction (SIGCHI), Ambient Intelligence for Scientific Discovery (AISD) Workshop, Vienna, April 25, 2004