2003 International Symposium on Advanced Intelligent Systems
September 25-28, 2003, Jeju, Korea

A Study for FIPA-OS Multi-Agent Framework in OSGi Service Platform

*Hyung-Jik Lee, Kyu-Chang Kang, Jeun-Woo Lee

Electronics and Telecommunications Research Institute, 161 Gajeong-dong, Yuseong-gu, Daejeon, 305-350, Korea
(Tel : 82-42-860-1597, Fax : 82-42-860-6671, e-mail : leehj@etri.re.kr)

Abstract: In this paper, we implemented a FIPA-OS multi-agent framework bundle in OSGi Service Platform. FIPA-OS is an open agent platform for constructing FIPA compliant agent using mandatory components that required by all FIPA-OS agents to execution and optional components that FIPA-OS agent can optionally use. The platform supports communication between multiple agents and communication language which conforms to the FIPA standards. FIPA-OS framework bundle is composed of DF(Directory Facilitator), AMS(Agent Management System), ACC(Agent Communication Channel) and MTS(Message Transport System) bundle. These bundles installed in the OSGi service platform and their life cycle can be managed by the framework.

Keywords : OSGi, FIPA, FIPA-OS

I. Introduction

Today, OSGi is the leading standard for the next-generation Internet services to homes, cars, small offices and other environments [1, 2, 3, 7]. It enables a new category of smart devices due to its flexible and managed deployment of services. To dynamically and automatically manage the services in OSGi, it is necessary for an agent service, such as remote management and SNMP agent service. Therefore, the agent framework and its technology should be considered in OSGi. FIPA is a non-profit organization aimed at producing standards for the interoperation of heterogeneous agents, and FIPA-OS is the first open source implementation of the FIPA standard. FIPA-OS 2 is a components-based agent framework implemented Java [4, 5]. In this paper, we developed the FIPA-OS multi-agent framework bundle in the OSGi service platform. The FIPA-OS multi-agent bundle is comprised of two types of bundle. One is a MTS(Message Transport Service) bundle such as RMI(Remote Method Invocation), IIOP(Internet-Inter ORB Protocol), and ACC(Agent Communication Channel). Another is an agent loader bundle.

II. FIPA-OS(Open Source)

The Foundation for Intelligent Physical Agents, FIPA, was formed in 1996 to produce software standards for heterogeneous and interacting agents and agent-based systems [4]. The purpose of FIPA is to promote the development of specifications of generic agent technologies that maximize interoperability within and cross agent based applications. In the production of these standards, FIPA requires input and collaboration from its membership and from the agent's field in general to build specifications that can be used to achieve interoperability between agent-based systems developed by different companies and organizations. This is encapsulated in FIPA's mission statement.

According to the FIPA specifications, Emorphia developed the FIPA-OS that is the first open source implemented in Java. The FIPA-OS now supports most of the FIPA experimental specifications currently under development. The FIPA97, the first specification of the FIPA, is produced in 1997, and FIPA-OS 2 is now available.

The FIPA-OS 2 is a component-based toolkit implemented in 100% pure Java. One of the most significant contributions received is a small-footprint version of the FIPA-OS, aimed at PDA's and smart mobile phones, which has been developed by the University of Helsinki as part of the IST project Crumpet [5]. The FIPA-OS 2 is an open source implementation of the FIPA standards. The message transfer among agents is possible through the message transport system, and message transfer protocol is HTTP, Remote Method Invocation (RMI), Internet-Inter ORB Protocol (IIOP) and WAP. The message format is implemented in XML, Agent Communication Language (ACL) and KQML. The software stack of FIPA-OS is shown in Figure 1.

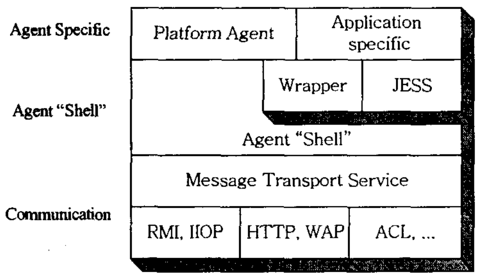


Fig. 1. Software stack of FIPA-OS

The FIPA-OS is composed of three parts: communication layer, agent shell and Application specific service agent layer [6].

Agents can be built using agent shell. These are implemented as Java base class with pre-defined hooks into the platform to use platform service. The FIPAOSAgent class provides a shell for agent implementation to use by simply extending this class. The FIPAOSAgent shell is responsible for loading an agent's profile, and initializing the other components of which the agent is composed.

The MTS(Message Transport Service) provides the ability to send and receive message to an agent implementation. The MTS within FIPA-OS is logically split such that incoming and outgoing messages path through a number of services within a service stack. This service stack is shown to Fig.2.

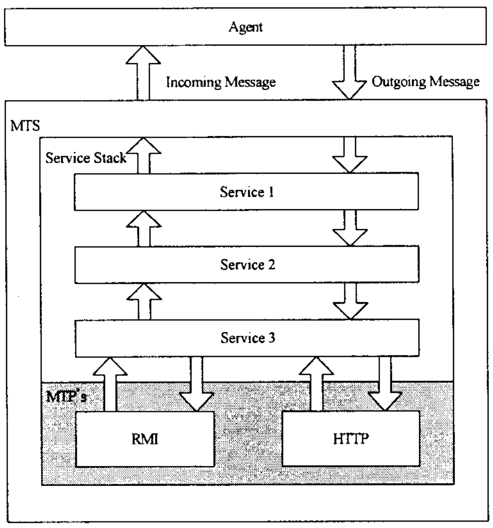


Fig. 2. Logical composition of the MTS

Each service is a stand-alone component that performs some transformation on outgoing messages, and the inverse transformation on incoming messages.

FIPA-OS currently comes bundled with MTP implementations that are specialization of the MTPBase class. The RMI transport is based upon Sun's RMI implementation that is part of the core Java 1.1 and Java 2 Standard Edition API. Due to the fact that this transport relies upon the use of Java serialization to encode messages, it is not interoperable with agents written in languages other than Java. However, this also means that it is much more efficient for communications between agents written in Java. Thus this transport is an internal MTP. The IIOP transport is based upon Sun's CORBA implementation that is part of the Java 2 Standard Edition API. It is compliant with FIPA IIOP MTP specification, and hence is potentially interoperable with an agent written in any language that supports CORBA, and this specification. Hence, this transport is an External MTP.

III. The OSGi Service Platform

The OSGi service platform is composed of services, bundles and entities [8]. Service is a Java object or interface, and the bundle is a form of packaging for service. It is also a functional unit with life cycle operations and class loading capability. It contains well-defined hooks that allow it to be plugged into the framework.

The release 1.0 of the OSGi service platform was developed in 1999, and now release 3 is available. The components of the OSGi service platform are listed in Table 1, and its software stack is shown in Figure 3.

Table 1. The package of the OSGi service platform.

Package	Description
org.osgi.framework	Framework
org.osgi.service.cm	Configuration admin
org.osgi.service.device	Device access
org.osgi.service.http	Http service
org.osgi.service.io	IO connector
org.osgi.service.jini	Jini service
org.osgi.service.log	Log service
org.osgi.service.metatype	Metatype
org.osgi.service.packageadmin	Package admin
org.osgi.service.permissionadmin	Permission admin
org.osgi.service.prefs	Preference service
osg.osgi.service.provisioning	Initial provisioning
org.osgi.service.startlevel	Bundle start levels
org.osgi.service.upnp	UPnP service
org.osgi.service.url	URL stream & content
org.osgi.service.useradmin	User admin
org.osgi.service.wireadmin	Wire admin
org.osgi.util.measurement	Measurement utility
org.osgi.util.position	Position utility
org.osgi.util.tracker	Service tracker
org.osgi.util.xml	XML parsers

The OSGi service platform release 3 includes support for mobile service platforms and application where data access is handled by a variety of secure service. Release 3 has reference architecture and remote management reference architecture.

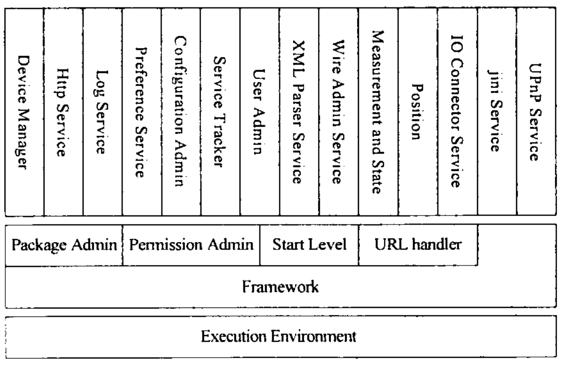


Fig. 3. The software stack of the OSGi service platform Release 3.

The Framework forms the core of OSGi service platform. It provides a general-purpose, secure and managed Java framework that supports the deployment of extensible and downloadable bundles. The framework manages the installation and update of bundles in an OSGi environment in a dynamic and scalable fashion, and manages the dependencies between bundles and service.

In the OSGi service platform, bundles are the only entities for deploying Java-based applications. A bundle is comprised of Java classes and other resources which together provide functions to end users and provide components called services to other bundles, called services. A bundle is deployed as a JAR files. JAR files are used to store applications and their resources in a standard ZIP-based file format.

In the OSGi service platform, bundles are built around a set of cooperating services available from a shared service registry. Such an OSGi service is defined semantically by its service interface and implemented as a service object. The service object is owned by, and runs within, a bundle. This bundle must register the service object with the framework service registry so that the service's functionality is available to other bundles under control of the framework. Dependencies between the bundles using it are managed by the framework.

IV. FIPA-OS Multi-Agent Framework Bundle

The FIPA-OS is an agent-based open source implemented in Java. It is comprised of message transport service, content language parser, agent management system and directory facilitator. So it is necessary for agent communication and agent loader bundle to develop the FIPA-OS framework bundle. The message transport service bundle is deployed of RMI, IIOP and ACC bundle. RMI and IIOP bundle act as agent communication among included in the same and different platforms. The ACC bundle is communication language bundle that can tell FIPA communication languages. And agent loader bundle start and shutdown a service agent using a convenient loader GUI.

In this paper, we implemented the FIPA-OS framework bundle and service agent bundle over the OSGi service platform. The FIPA-OS bundle is composed of two bundles. One is a MTS bundle such as RMI, IIOP, HTTP and ACC. Another is an agent loader bundle.

The initialization of the OSGi service platform is shown in Figure 4. The MTS bundles and agent loader bundle are installed and started in the OSGi service platform.

In Figure 6, the result of the RMI bundle execution is shown. The RMI naming service and the lookup service can be enabled as registering of "RMINS" to the RMIRegistry. The role of the RMI bundle is to support a communication between agents in the same platform.

Fig. 4. The initialization of the OSGi service platform.

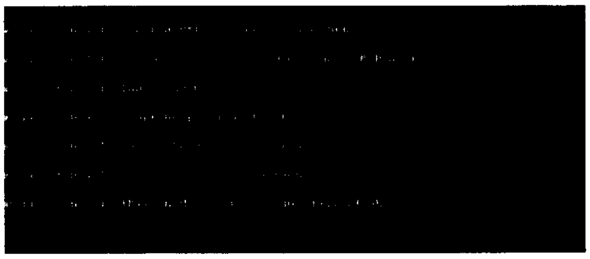


Fig. 5. RMI bundle .

It has been designed to be more efficient than the Internet-Inter ORB Protocol transport for inter-agent communications, and can be used by any agent or the Agent Communication Channel. Naming services provide the mechanism for agents on a platform to locate one another using a name resolution service. This provides a mapping between the names of an agent and there physical location, such other agents can interact with them. In order for an agent to be located on a platform, it must register with at least one naming service that is used by the platform. In Figure 5, RMI port number is 3000, and lookup service is enabled as binding it to the agent communication channel, agent management system and directory facilitator.

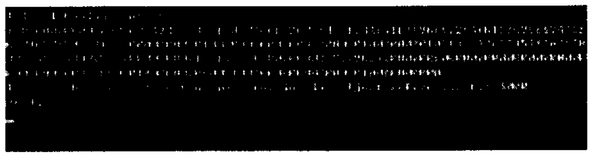
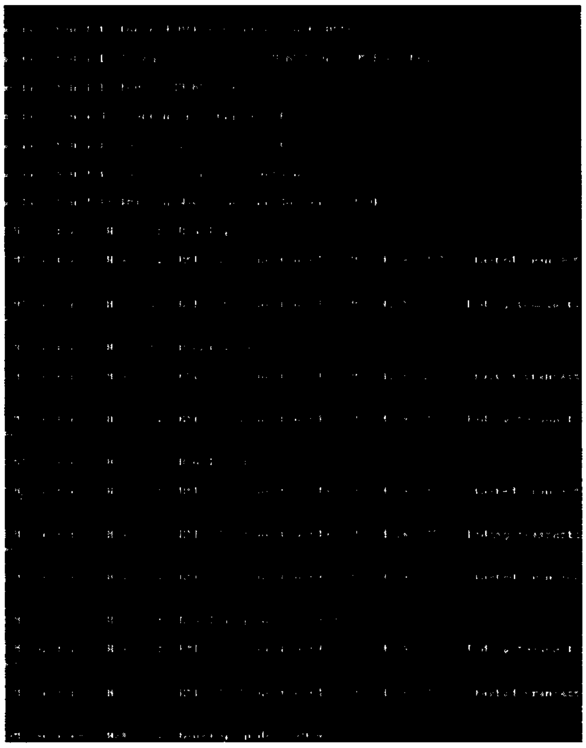


Fig. 6. The IIOP bundle.

The result of the IIOP bundle execution is shown in Figure 6. The port number of the IIOP is 3000, and the role of the IIOP bundle is to support a communication between agents in a different platform. This is an implementation of the FIPA IIOP specification, and can presently only be used by the Agent Communication Channel.

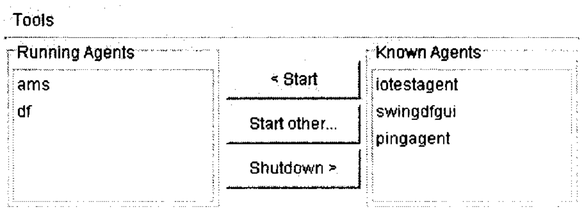
The execution result of the ACC bundle is shown in Figure 7. The ACC bundle provides the ability for agents on a platform to interact with agents on other platforms by providing a message passing service.

In Figure 8, the execution result of the agent loader bundle is shown. The running agents are agent management system agent and directory facilitator. The Known agents are iotestagent, swingdfgui and so on.



```
[illegible]
```

Fig. 7. The ACC bundle.



Tools

```
Running Agents                        Known Agents
ams                  < Start          iotestagent
df                   Start other...   swingdfgui
                     Shutdown >       pingagent
```

Fig. 8. The gent loader bundle.

V. Conclusion

In this paper, we implemented the FIPA-OS multi-agent framework bundle in the OSGi service platform. The FIPA-OS multi-agent framework bundle is comprised of two bundles. One is a message transport service bundle such as RMI, IIOP and ACC. Another is an agent loader bundle. Experiments are conducted on the service agent communication and loading by the agent loader bundle, and these bundles installed in the OSGi service platform and their life cycle can be managed by the framework.

References

[1] Condry M, Gall U, Delisle P, pen Service Gateway architecture overview," *IEEE Industrial Electronic Society*, vol. 2, pp. 735-742, 1999

[2] Jordan D, "Java in the home: OSGi regidential gateways," *Java Report*, vol.5, no. 9, pp. 38-42, 2000

[3] K. Chen, L. Gong, Programming Open Service Gateways with Java Embedded Server^{TM} Technology, *ADDISON-WESLEY, New York*, 2001

[4] FIPA homepage. http://www.fipa.org

[5] FIPA-OS homepage. http://www.emorphia.com/EPL

[6] FIPA-OS V2.1.0 Distribution Notes. http://flow.dl.sourceforge.net/sourceforge/fipa-os/FIPA_OSv2_1_0.pdf

[7] OSGi homepage. http://www.osgi.org

[8] OSGi service platform, release 3. http://www.osgi.org/resources/sp-r3/r3.book.pdf