An Open Source Platform for the Integration of Distributed Services

Félix Cuadrado*, Juan C. Dueñas*, José L. Ruiz*, Jesús Bermejo**, Miguel García***
*UPM, **Telvent, ***Telefonica I+D
{fcuadrado,jcduenas,jlruiz}@dit.upm.es, jesus.bermejo@telvent.abengoa.com, mgl@tid.es

Abstract

Service-orientation is not only a means for systems integration, but also an architecture paradigm for building systems. Flexibility, adaptability and interoperability are some of the benefits derived from the adoption of SOA. These principles are often applied to build distributed systems. But whether SOA is applied or not, building a distributed system is still complex, what increases the demand on high-quality middleware to achieve success. This paper presents a runtime service integration architecture that enables distributed service interactions regardless the capabilities of the devices involved, i.e. to deal with services distribution over a heterogeneous network. The solution presented is built upon the OSGi service platform which has been leveraged with OSS middleware components to provide remote communications (with connectors and adaptors), service discovery, service binding and service publication. The development of this architecture is carried out within the ITEA OSIRIS Consortium and is being applied to industrial case studies; including a tax administration applications, service provisioning to residential environments and a mobile CRM.

1. Introduction

Current generation networks enable the provision of ubiquitous services, delivered by a mesh of smart devices surrounding end users. However, the technical challenges are numerous, because of the heterogeneity of the interconnected devices and protocols. The European project ITEA OSIRIS[1] [1] aims at delivering a complete infrastructure for the dynamic integration of services over distributed environments. This line of work advances the results obtained in a previous project, ITEA OSMOSE [2], which developed middleware components for service provisioning in the home domain [3].

[1] The work described here has been performed in the context of the European project ITEA-OSIRIS (ip.04040),under grant by Spanish Ministerio de Industria,Turismo y Comercio in the PROFIT program.

OSIRIS aims at providing a generic architecture, suitable for several heterogeneous domains (Public administration services, End to end service provisioning, financial services, and mobile CRM – Customer Relationship Management – services). These industrial demonstrators are used to validate that the solution enables dynamic service interaction among a wide range of devices (from low-capacity smart-phones to high-end backbone servers). Some examples of these scenarios are: submitting e-administration forms, either through a personal computer's browser, a mobile phone or an interactive TV. Another example would be the integration of Internet multimedia services such as Flickr or Youtube with the infrastructure available at a smart home, abstracting end users from any configuration and enabling at the same time the remote management of the domains by the service providers.

2. State of the Art on SOA

The SOA (Service-Oriented-Architecture) paradigm [4] is the foundation for the OSIRIS architecture. Service-Oriented Architectures are the natural evolution of the distributed computing paradigm. A service is a stateless software component, which performs a specific task, defined through a communication interface.

The SOA interaction protocol is executed by three different actors; the service provider, the service consumer and the service broker (or registry). This third element is the main deviation from traditional client-server communications. The registry reduces the coupling between the other two entities, reduces vendor-dependency and enables service discovery

The SOA principles have been adopted in different contexts (from business-level organization to design level pattern). We will see in the following paragraphs an overview of these different approaches to the same idea.

2.1. Web Services

The popularity of Web Services (WS) [5] has fostered the massive adoption of the SOA paradigm. However, despite its huge success, it is important to clarify that WS are only one implementation (and not the first one)

of the base principles behind it. Interoperability is the main success factor, thanks to an impartial specification, supported by large companies (IBM and Microsoft) and standardized by neutral organisms. The WS stack is composed of several platform-independent protocols, such as HTTP (as a transport protocol), XML (message encoding), WSDL [6], *Web Services Description Language,* (service interface description), SOAP [7], *Simple Object Access Protocol*, (service invocation protocol) or UDDI [8], *Universal Description, Discovery, and Integration* (service registry).

Web Services have transformed enterprise communications, both inside and outside their domain, through the form of the ESB (Enterprise Service Bus), WS have become one of the most relevant standards for Internet communications. Because of that, the WS stack has been completed with additional specifications for improving non-functional aspects of service invocation, such as security, service choreography or transactionality.

2.2 Device-level SOA

Web Services focus on coarse-grained, high-level business interactions. However, in the recent years the interest of adopting SOA for smaller scale activities and devices has boosted. That flavor of SOA has been called device-centric [10] SOA or device-level services [11]. Restricting the interactions to a more controlled environment reduces computing complexity and enables dynamic service discovery and binding on a local environment. Jini was the first attempt on that model, although recently it has fallen out in favor of device-centric technologies such as UPnP (Universal Plug and Play) [9] and DPWS [11] (Device Profile for Web Services)

2.3. OSGi

The OSGi service platform [12] brings service-oriented computing principles to single nodes. OSGi has evolved from its initial conception as a residential gateway to becoming the 'Java Operating System'. OSGi leverages the Java Virtual Machine with a dynamic component model based on services, suitable for a wide range of device profiles (from mobile phones to high-end servers).

Deployable OSGi components are called bundles. A bundle is composed by a jar archive with additional metadata for clearly defining the requirements for the component to work and the additional resources offered to the platform (expressed as Java packages), which are automatically processed by the framework. At runtime, each bundle has an independent lifecycle, allowing hot-deploy and update operations in a

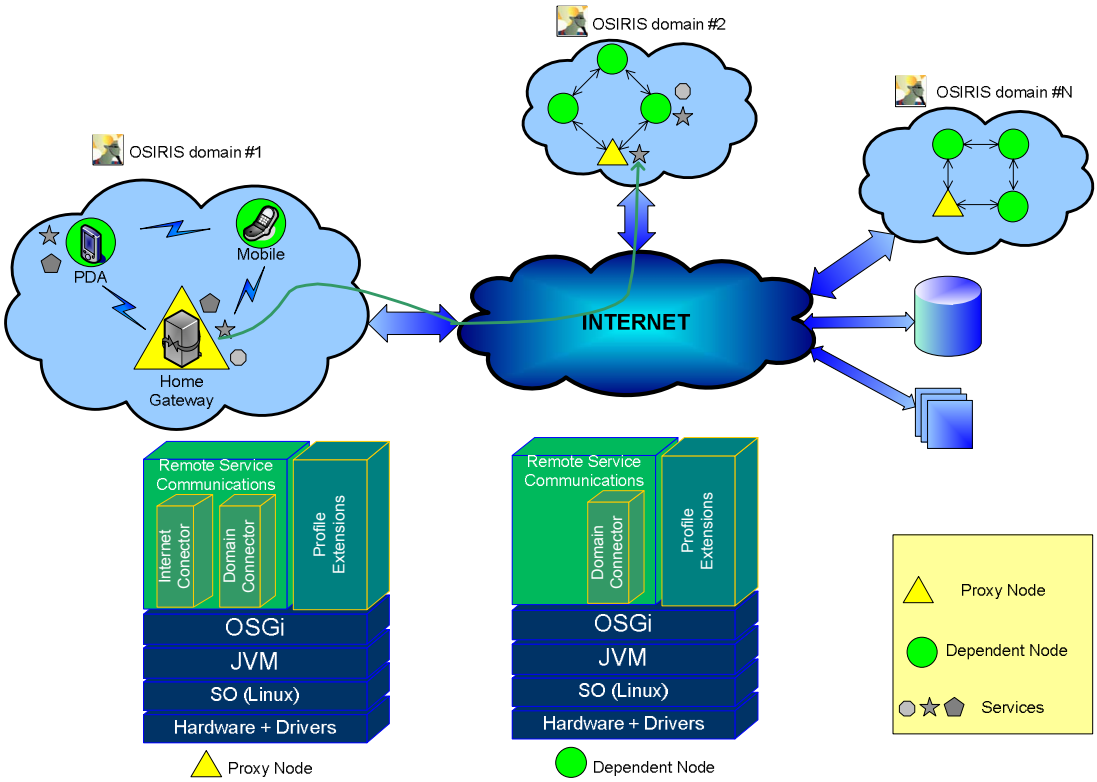

Figure 1 The OSIRIS Platform

platform without the need of rebooting the whole system.
Dynamic bundle interactions are performed through services. The OSGi platform defines a local service registry where bundles can register services (runtime objects registered with a contract specified by a Java interface and a set of properties) and query for implementations. This way, the SOA paradigm can be adopted also inside the applications. The basic OSGi specification is further enriched by a wide range of services defined by the standard. They provide capabilities such as platform management (*FrameworkAdmin, StartLevel, ConfigurationAdmin, EventAdmin*) or remote communications (*JiniService, UPnPService, HttpService*).

3. The OSIRIS Service Platform

The OSIRIS architecture aims at supporting dynamic service-oriented distributed applications. Different application domains (E2E services, CRM, financial services, public administration) have contributed their specific requirements in order to define a generic architecture. Because of that, devices with a wide range of computing power can be part of a system as long as they are connected.
Figure 1 depicts a high-level view of an OSIRIS system. An OSIRIS system is distributed over one or more domains, interconnected through Internet. Services are the foundation of the architecture. As we have seen in the state of the art analysis, there are different approaches to the SOA paradigm, depending on the abstraction level (from business to network devices to design patterns). OSIRIS aims at unifying those three views into a common model, as shown in Figure 2. The OSIRIS middleware mediates between those levels and opens up service-based communications among heterogeneous devices, regardless of the protocol details and the physical location.

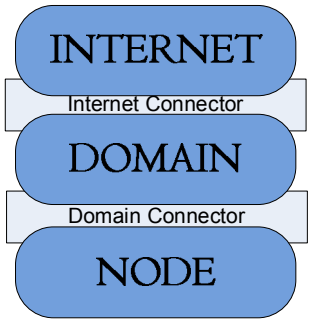

Figure 2 OSIRIS Service view

3.1. OSIRIS Nodes

OSIRIS nodes are hardware devices leveraged with additional middleware. Some examples of candidate nodes are PDAs, personal computers or blade servers. More limited devices can also be part of OSIRIS systems as 'light-weight nodes'. Focusing on the software elements, the OSGi service platform has been selected as the cornerstone of the OSIRIS middleware. OSGi provides an efficient service programming model for each node and a robust component specification enabling a high-level of modularity for both middleware and applications.
OSIRIS extends the base OSGi specification with seamless remote service communications support. As mentioned before, nodes aggregate into domains, and communicate through service invocation, both at domain level and over Internet. As mentioned before, the intent of OSIRIS is closing the bridge among the existing three levels of SOA enabling communications between heterogeneous devices. The Domain Connector and the Internet Connector modules support these scenarios. Regarding communications, nodes can play two different roles inside a domain: dependent nodes and proxy nodes.
Proxy node. Every domain connected to the Internet needs one node to act as the proxy node for the domain. The mission of the proxy is to transparently mediate between services invocation from the domain and Internet. Remote service operations from any node inside the domain are directed and translated to the appropriate protocol by the proxy.
Dependent node. Remaining nodes play the dependant role. They are only domain-aware although they can export / import services through the use of the proxy.

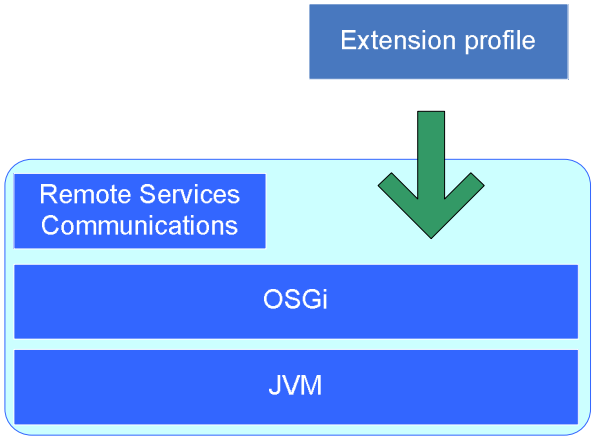

Figure 3 OSIRIS Node Basic profile

Figure 3 shows the OSIRIS node basic profile. It defines the OSIRIS component model and provides the remoting functionality. The intent is to reduce the mandatory components to the minimum, in order to

support a wide range of devices. However, the basic profile can be enriched by the installation of extension profiles. An extension profile is a set of components providing a specific functionality or capability to the platform. Inside the project several extensions have been developed such as remote management, distributed deployment, security or context adaptation.

3.2. Domain Connector

The Domain Connector (DC) is the mandatory component for each OSIRIS node, as it manages domain-wide services communications. Its purpose is twofold; discovery and service invocation. Discovery is made automatic through the use of a multicast protocol for announcing a node and its associated services to the rest of the network. With that information, DCs build a distributed service registry.

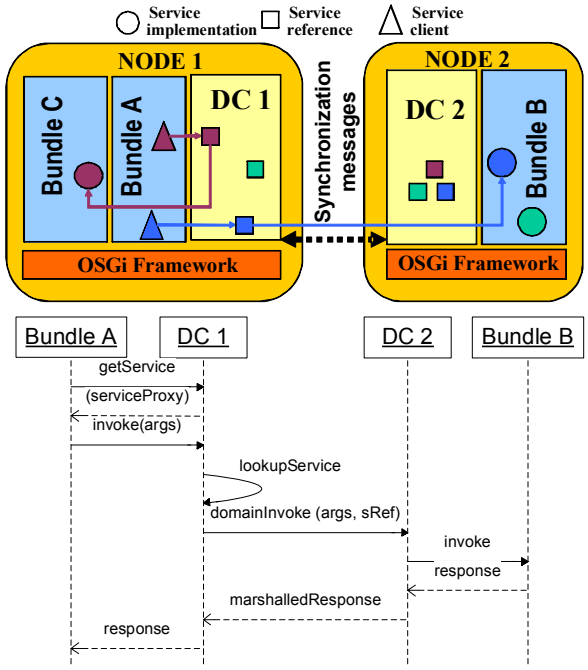

Figure 4 Domain Connector Sequence Diagram

The second function of the DC is allowing service invocation inside the domain transparently of the physical location of the service. Figure 4 depicts how this mechanism works. In this scenario there are two nodes, each one with several components. Registered services in either node appear in each other node DC registry. Whenever a component wants to invoke a service, it is unaware of its locality. For services located outside the consumer node, the DC translates the call to the domain-wide protocol and transmits it to the DC of the provider node, which performs the actual operation with the service. Stepping back the same steps the response is transported to the consumer node and delivered to the component.

3.3. Internet Connector

The Internet Connector (IC) extends service communications beyond the domain, enabling inter-domain or interoperability with non-OSIRIS systems. With these requirements in mind, we have selected Web Services as the service model for Internet communications.

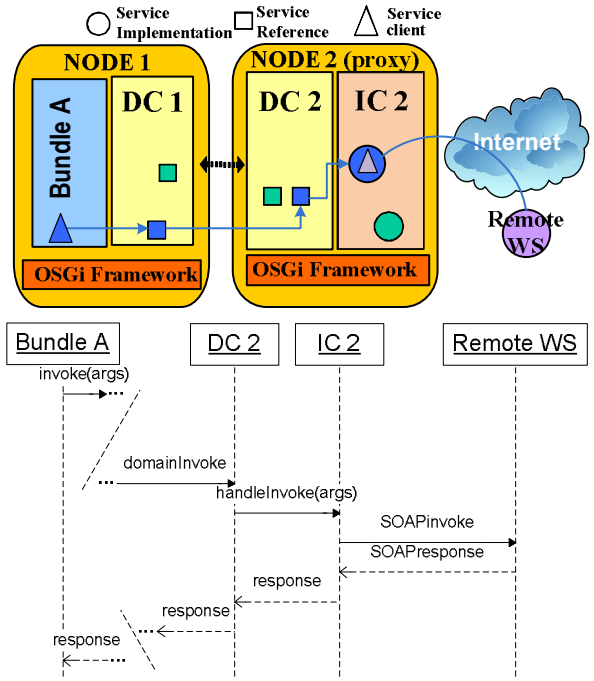

Figure 5 Internet Connector Sequence Diagram

Unlike the DC, the IC does not provide a solution for service discovery, and focuses instead on service invocation. Figure 5 illustrates how this component enables, in collaboration with the DC, remote service invocation for limited nodes.

Existing WS are made available to an OSIRIS domain by registering a delegate service in the DC. Whenever a node requests one of those services the petition will reach the IC, which will construct a SOAP message and perform the invocation on behalf of the local node. The service response will be delivered to the consumer component in its local format by the collaboration of both connectors.

The IC also allows exporting any domain service to the outside. Services can be tagged for Internet exporting, triggering the IC to generate and publish a WSDL descriptor for the service. Incoming requests will be handled by the IC and redirected to the provider in a similar way as in the previous case

4. Architecture implementation

In order to deliver a complete solution for distributed services the OSIRIS architecture has been supported by additional technical developments aligned with some of the most important open source communities. The objective is to provide a complete stack for developing OSIRIS applications while avoiding at the same time an overlap with existing work in the communities.

The technical work can be classified into three different categories: core middleware implementations, profile extensions and supporting tools. On the first group, the development effort has been focused on the remote service connectors, as the OSGi specification already has three different open source implementations. The DC is not bound to any specific protocol. A reference implementation (the Osiris Domain Connector) has been developed to showcase the communication mechanisms, based on RMI (Remote Method Invocation) for the domain-wide services. There are several device-centric SOA specifications which could also perform this role, such as the aforementioned UPnP and DPWS. In particular, the ITEA SIRENA[15] project has provided a Java-based lightweight implementation of the DPWS stack, which is being refined and adopted to the OSGi model under the project ITEA ANSO [16]. This solution could be adopted in future versions to unify the service model among domains and outside communications. Other alternatives include JMDNS, based on the DNS-SD (DNS-Service Discovery) protocol, or the R-OSGi project [13], which uses an efficient SLP (Service Location Protocol) implementation [14] for discovery and local service invocation.

Regarding the IC, a specific implementation is necessary in order to integrate with the DC. With that objective in mind, two existing Java-based Web Service implementations have been adapted to this role: the Morfeo Application Server (delivering both IIOP and WS Java support) and the Apache Axis WS engine, under the OS4OS community.

The second category covers the extension profiles developed at the project. The required profiles have been identified by the domain-specific input received from the project demonstrators. Whereas the architecture definition only covers the basic communication services, extensions define platform-level services and non-functional extensions, which can be independently added. The management profile defines a remote management architecture with OSGi instrumentation agents (JMood) and remote management APIs (OSGi Access). The distributed deployment defines a complete architecture composed by a deployment repository and a set of agents for the deployment domain, provided by the JBones project. Security profile extensions leverage the Internet Connector with the federated authentication and authorization mechanisms of OpenLiberty and OpenPMI [17], integrated by the PAPI developers

Finally, additional developments have taken place for proving an adequate tools support for the architecture. Component-based software and SOA impose an extra layer of complexity which can hamper the success of that kind of technologies, and tools can bridge that dangerous gap. Some partners of the OSIRIS project are active contributors of the leading service tooling projects, such as Eclipse PDE (OSGi tooling), Eclipse WTP (Web-Services tooling) and Apache Lomboz (SCA tooling), developed in parallel with more more specific tools. Table 1 summarizes some of these initiatives carried out by OSIRIS project members.

Table 1 – OSIRIS OSS initiatives

Name	Description	Community
ODC	DC Reference impl	OS4OS[2]
Axis Bundle	IC Reference Impl.	OS4OS
MAS Internet Connector	IC Reference Impl.	Morfeo[3]
Apache Felix JMood	Ext. Profile Management agent	Apache[4]
JBones	Ext. Profile Deployment arch.	OS4OS
PAPI AA Extension	Ext. Profile Security extensions	IRIS-LIBRE[5]
ISIS	Ext. Profile Data&Interface services	OS4OS
SOFA 2	Ext. Profile Services lifecycle	ObjectWeb[6]
MADAM	Ext. Profile Context adaptation	MADAM[7]
eLuzien	OSGi tool support	OS4OS
Trinity	Remote services creation tool support	Morfeo
ObjectWeb Lomboz	Remote services creation tool support	ObjectWeb
Eclipse WTP	Remote services creation tool support	Eclipse[8]
Eclipse PDE	OSGi tool support	Eclipse

[2] OS4OS, http://www.os4os.org/
[3] Morfeo Project, http://morfeo-project.org
[4] Apache Software Foundation, http://www.apache.org/
[5] IRIS-Libre, http://www.rediris.es/gt/iris-libre/
[6] Object Web, http://www.objectweb.org/
[7] MADAM Project, http://www.ist-madam.org
[8] Eclipse Foundation, http://www.eclipse.org/

5. Conclusions

The SOA paradigm provides a solution for distributed systems made of low-coupled elements. However, the diversity of service-oriented specifications and the increased technical complexity hamper a wider adoption of these techniques. The OSIRIS platform defines a generic architecture which manages this complexity by unifying the different flavors of SOA.
The minimum requirements of the architecture have been kept as low as possible to enable the participation of embedded devices as first-class nodes. The low footprint does not limit the middleware functionality, as the specification defines mechanisms to extend its capabilities. This way, extension profiles have been defined for providing management capabilities, securing services, and so on.
The architecture specification is only a part of the OSIRIS objective. In collaboration with several international OSS communities, the concepts behind the platform have been made available as software components. In addition to those, development of supporting tools also enables the adoption of OSIRIS architecture. This way, both the architecture and the implementation will receive additional input for further polish, enabling a better support and broader use scenarios.

6. References

[1] ITEA-OSIRIS (Open Source Infrastructure for Run-time Integration of Services) project, website at: http://itea-osiris.org

[2] ITEA-OSMOSE (Open Source Middleware for Open Systems in Europe) project, website at: http://itea-osmose.org

[3] J.C. Dueñas, J.L. Ruiz, M. Santillan "An end-to-end service provisioning scenario for the residential environment"Communications Magazine, IEEE, 2005

[4] T. Erl, "Service-Oriented Architecture. Concepts, technology and design." Ed. Prentice Hall., 2005

[5] E Newcomer , "Understanding Web Services: XML, WSDL, SOAP and UDDI", Ed. Addison-Wesley, 2002

[6] D. Booth, C. K. Liu, "Web Services Description Language (WSDL) Version 2.0" , W3C Recommendation, 2007, available at w3c.org

[7] N. Mitra, Y. Lafon, "Simple Object Access Protocol (SOAP) " Version 1.2 (Second Edition), W3C Recommendation, 2007, available at w3c.org

[8] OASIS, "Universal Description, Discovery and Integration (UDDI) specification", v3.0, disponible en http://www.oasis-open.org

[9] UPnP Device Architecture v1.0.1, UPnP Forum, 2006. Available at http://www.upnp.org/specs

[10] D. Barisic, M Krogmann, G. Stromberg, P. Schramm, "Making Embedded Software Development More Efficient with SOA", Proceedings of the 2nd International IEEE Workshop on Service Oriented Architectures in Converging Networked Environments. Canada, May 2007

[11] F. Jammes, A. Mensch, H. Smit, "Service-Oriented Device Communications using the Devices Profile for Web Services", Proceedings of the 2nd International IEEE Workshop on Service Oriented Architectures in Converging Networked Environments. Canada, May 2007

[12] Open Services Gateway Initiative. "OSGi Service Platform," Specification Release 4.0, August (2006).

[13] J. S. Rellermeyer, G. Alonso, "Services Everywhere: OSGi in Distributed Environments", EclipseCon 2007, Santa Clara, USA.

[14] E. Guttman, "Service location protocol: automatic discovery of IP network services", IEEE Internet Computing, Jul/Aug 1999

[15] ITEA SIRENA(Service Infrastructure for Real-time Embedded Networked Applications), website at http://sirena-itea.org

[16] ITEA ANSO (Autonomic Networks for SOHO users)

[17] J.López, I. Agudo, J. Montenegro, "On the deployment of a real scalable delegation service", Information Security Technical Report Vol.12, Issue 13, pp: 139-146