Ready for Distribution? *

Turning Modular into Distributed Applications with the R-OSGi Deployment Tool

Jan S. Rellermeyer Gustavo Alonso Timothy Roscoe
ETH Zurich, Department of Computer Science, 8092 Zurich, Switzerland
{rellermeyer, alonso, troscoe}@inf.ethz.ch

Abstract

In this demonstration we show drag-and-drop distribution of centralized, modular Java applications.
Our system is based on OSGi, an industry standard for building Java applications out of modular units loosely connected through services. Since OSGi is a centralized system, we have elaborated a solution to seamlessly distribute OSGi applications along the boundaries of services and thereby turning arbitrary OSGi applications into distributed applications. In this demonstration, we present an Eclipse based tool that takes the source code of an OSGi application as input, produces a graph of its modules and module dependencies, and allows the user to deploy the application across a distributed system by dragging-and-dropping its constituent modules on different machines. By defining constraints on the distribution, the tool can also support advanced features like load-balancing or redundancy of modules.

Categories and Subject Descriptors D.3.3 [*Programming Languages*]: Language Constructs and Features—Modules, Packages; C.2.4 [*Computer-Communication Networks*]: Distributed Systems—Distributed Applications; K.6.m [*Management of Computing and Information Systems*]: Miscellaneous

General Terms Design, Management

Keywords R-OSGi, OSGi, Deployment, Eclipse, Concierge

* The work presented in this paper was supported (in part) by the National Competence Center in Research on Mobile Information and Communication Systems NCCR-MICS, a center supported by the Swiss National Science Foundation under grant number 5005-67322.

OOPSLA'07, October 21–25, 2007, Montréal, Québec, Canada.

1. Introduction

Designing and developing distributed applications is still a difficult task that involves in-depth knowledge of both application semantics and networking issues. In contrast, building modular applications is well understood and such applications are easier to design. As we argue in [3], it is possible to distribute an application by locating its constituent modules on different machines. For this purpose, we have developed R-OSGi, a middleware platform that implements the OSGi standard in a distributed setting. The key to R-OSGi is that it treats distribution transparently and uses the natural boundaries between software modules to establish the potential boundaries for distribution of the application. R-OSGi masks all failures as connectivity failures across modules, thereby effectively making distribution transparent to the developer. In this demonstration, we present an Eclipse based deployment tool that allows to distribute an application by simple drag-and-drop of its modules. We further demonstrate the potential of the tool by incorporating rich configuration options addressing requirements commonly encountered in distributed scenarios such as redundancy for load balancing or failure-resilience.

2. Background

2.1 OSGi

OSGi [1] recently became popular because it addresses a major issue with Java: The lack of built-in support for describing and handling modules and their dependencies. In OSGi, the modular units are called *Bundles* and are JAR files with additional information in the manifest. Most importantly, the manifest can explicitly declare which Java packages of the Bundle are offered for export and which packages provided by other bundles are required as imports, thereby making dependencies between modules expicit. The runtime infrastructure of OSGi (the *Framework*) can thereby delegate between the classloaders of Bundles in a well-defined manner.

OSGi allows inter-module dependencies to be specified declaratively. Bundles define what to import but typically not from where. While dependencies potentially limit the flexibility of a modular design (since they pose constraints),

OSGi provides loose coupling of Bundles through services. Bundles can register their implementation of a service interface with a central service registry and other bundles can retrieve the service implementation from the registry. Since access to the service is restricted to its interface, clients of the service only have to know the interface type and have no knowledge of the service implementation. The OSGi framework handles the consistency of package exports.

2.2 R-OSGi

Two features of OSGi have so far prevented it from being used as middleware for distributed applications: Firstly, the central service registries of different peers are isolated from each other. Secondly, since packages are allowed to share code through package imports, a module import on one machine must be resolvable even though the local Java VM might only hold a subset of the application's classes and packages.

To address these limitations, we have developed R-OSGi which runs on top of an arbitrary OSGi framework implementation as an ordinary service, but transparently handles all issues related to distribution. Through a service discovery protocol, R-OSGi extends the idea of the central service registry to whole networks. By creating dynamic proxies for services on-the-fly and registering these under the interfaces of the original service, R-OSGi transparently hides the distributed nature of the deployment. With R-OSGi, we can turn arbitrary service-oriented OSGi applications into distributed applications without changing them. Finally, R-OSGi turns network and node failures into *module unload* events, which OSGi applications already expect and can gracefully handle.

3. Demonstration

We will demo a deployment tool (Figure 1) implemented as an Eclipse plugin that takes existing OSGi applications as input. Since service dependencies cannot be statically determined from the bundles, the tool performs code analysis to reason about which services are provided or consumed by each bundle. The result is a visualization of the application structure when it is executed as a conventional OSGi application running on a single Java VM. Attendees of the demonstration will then see how we turn this application into a distributed one by simply dragging and dropping individual OSGi services to different machines. We then deploy the distributed application to the corresponding machines and start it with a single click. Using our own OSGi R3 implementation *Concierge* [2], even small devices like PDAs and Smartphones can participate in an R-OSGi deployment. To demonstrate the advanced capabilities of the tool, during the demo we will modify the configuration of the application with just a few clicks and introduce load balancing for a service.

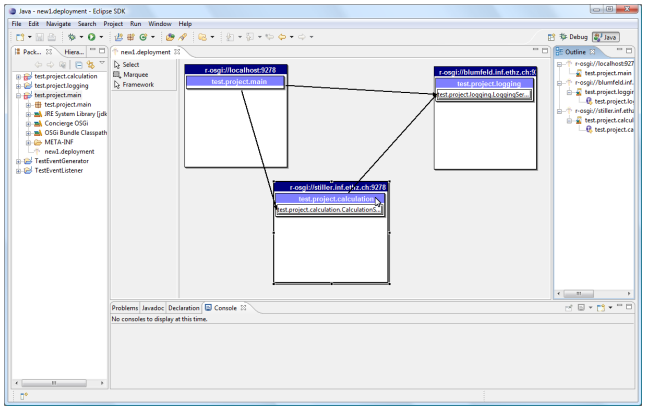

Figure 1. R-OSGi Deployment Tool

The innovation in our approach is not only the capabilities of the deployment tool, but also its ease of use. In order to give attendees an impression of what is going on in the middleware, we will visualize and monitor the structure of the deployment and all the messages exchanged in the depths of the R-OSGi system.

4. System Setup

The demonstration setup consists of one notebook with the application and the Eclipse-based deployment tool, plus several other machines (Notebooks or PDAs) that will take part in the deployment. In order to make the whole process and the resulting distributed application more visible, the notebook running the tool is connected to a projector.

5. Presenter

Jan S. Rellermeyer has received his MSc in Computer Science from ETH Zurich in 2006. He is currently doing a PhD in the Information and Communication Systems Research Group (IKS) at ETH. His fields of research are OSGi in distributed environments and fluid computing. The latter deals with building collaborative, highly flexible applications for future mobile and ubiquitous applications. Jan has worked with OSGi for several years and pioneered the introduction of support for cross-VM services in OSGi. The OSGi Alliance has shown interest in using his experience in upcoming standards. Jan presented the core approach of R-OSGi in a long-format talk at EclipseCon in 2007.

References

[1] Open Service Gateway Initiative. `http://www.osgi.org`.

[2] J. S. Rellermeyer and G. Alonso. Concierge: A Service Platform for Resource-Constrained Devices. In *Proceedings of the 2007 ACM EuroSys Conference*, 2007.

[3] J. S. Rellermeyer, G. Alonso, and T. Roscoe. R-OSGi: Distributed Applications Through Software Modularization. In *Proceedings of the ACM/IFIP/USENIX 8th International Middleware Conference*, 2007.