The Long and Winding Road...

Jim Colson | IBM Fellow | IBM Corporation
The journey begins
The JCP (Java Community Process) was formalized by Sun to allow others to participate in Java platform specification development in 1998.

The Connected Alliance, established in 1998, was an informal name taken by the ad hoc group formed to explore Java software standards for residential gateways. The group included IBM, Ericsson, Oracle, Sun, Motorola, and Nortel among many others.
Technically ...

JCP JSR 8 “Open Services Gateway Specification”
March 1999

Excerpts:

There is need for a well-defined, single Java specification in this area. The HomeAPI organization, www.homeapi.org, is developing a proprietary set of specifications ...

In contrast, the OSG specification will provide a cross-platform solution that will give ISVs and Service Providers a consistent, open environment that they can develop and deploy services to.

The OSG Specification is expected to define APIs for "cradle-to-grave" life cycle management of services, inter-service dependencies, data management, device management, client access, resource management, and security.

...installing, versioning, and configuring the services...

...could conceivably be more complex devices that could implement desktop or even server Java platforms. To handle this potentially wide range of platforms, the OSG Specification will seek to provide a useful, common baseline of functionality...

...dynamic services...

The OSG specification are expected to be based to a large extent on the Java Embedded Server 1.0 specification. Especially relevant is the ServiceSpace specification.

http://www.jcp.org/en/jsr/detail?id=8
The journey begins... and Organizationally...

Open Services Gateway initiative (OSGi for short)
March 1999

OSGi was formally incorporated as a non-profit organization in June, with 11 member companies elected to its Board of Directors.

Board of Directors: Alcatel, Electricité de France, Ericsson, IBM, Motorola, Oracle, Sun, Echelon, National Semiconductor, Nokia, Whirlpool

OSGi is an independent, non-profit industry group working to define and promote open specifications for linking future generations...

http://www.osgi.org
The journey begins with an anticipated merge.

Joint OSGi & Sun Microsystems Press Release
Nov 22, 1999

Excerpts:

The decision to transfer the development of the Gateway specification from JCP to OSGi was made by the OSGi board.

"Sun is a founding member and is strongly committed to OSGi. Sun offered to start the specification process under the JCP to enable OSGi to jump-start the work," said Jim Mitchell, Chief Technology Officer of the Consumer and Embedded Division at Sun Microsystems.

"Now that OSGi has created its own open process, it is natural and fitting that this work be completed in OSGi. We were pleased to help get OSGi a set of existing specifications from Sun in order to save time in the developmental process."
Points of Interest
OSGi releases its first public specifications.
May 3rd, 2000

Created by the OSGi Java Expert Group (JEG)
(now known as the OSGi Core Platform Expert Group (CPEG))

led by Li Gong, director of engineering at Sun

"I am grateful to the past and current members of the Java Embedded Server team, without whom there would not be a Java Embedded Server product. I also appreciate the collaborative spirit and the technical insights offered by members of OSGi’s Java Expert Group. It was my privilege to chair this group until the completion of the Open Service Gateway 1.0 specification...“ – Li Gong, from the book “Programming open service gateways with Java embedded server technology” authored by Kirk Cheng and Li Gong.
OSGi delivers a decade of proven technology, and proven collaborative evolution.

- R1 May 2000: Residential Gateways
- R2 Oct 2001: Residential Gateways
- R3 Mar 2003: Vehicle
- R4 V4.0 Oct 2005: Desktop, Mobile
- R4 V4.1 May 2007: Desktop
- R4 V4.2 Sep 2009: Enterprise

The initial targets for OSGi specifications & technology
- Pockets of market success (MFP, telematics, etc.)
- Generally, too far ahead of their time (e.g. Residential Gateways)
  - 2007 – Residential Expert Group
    - High level of activity and interest
- Starting with Desktop, OSGi market timing “spot on”
Points of Interest

Multiple Implementations

- Initially, all closed source implementations:
  - IBM – Service Management Framework (SMF)
  - Gatespace (now Makewave) – GDSP
  - ProSyst – mBeddedServer (now mBS)
  - OSGi Reference Implementation (seeded from Gatespace)

- 2001
  - Oscar (Richard Hall) – the first open source implementation

- 2003
  - IBM donated SMF to Eclipse which became Equinox.
  - Makewave moved GDSP to the open source Knopflerfish

- 2005
  - Oscar moved to Apache which became Apache Felix
  - Eclipse Equinox became the OSGi Framework RI

*mBS, Knopflerfish, Felix, Equinox all continue today*
Eclipse release 1.0 via Eclipse.org consortium
November, 2001

Why?
Pull together a fragmented Java development eco–system

How?
9 initial members including Borland, Rational, TogetherSoft, WebGain and others

Open Source licensing and operating model

Subsequently 9 more releases (version 2.1.3)
February 2004

During this time
Eclipse invented and evolved a home grown “plug-in” component model. It was touted as simple, and it was focused on Eclipse use cases for enabling different Eclipse based tooling configurations.
Eclipse.org becomes the Eclipse Foundation
EclipseCon 2004

Independent, Not-For-Profit Foundation
Permanent Staff

Why?
Vendor desire for assured independence from IBM
Customer clarity regarding “open” and “free”

How?
Many companies played an active role in forming the Foundation including HP, SAP, Intel, TogetherSoft, QNX, Instantiations, and others.

Eclipse also positioned itself for the future, technically, by...
Eclipse 3.0 Delivers Universal Platform for Application Construction
June 2004

Eclipse plug-in model replaced by the OSGi open standard.

Why?
The Eclipse community saw the value of an Open Standard, which was simultaneously mature and malleable via true open standard processes, as originally outlined in 1999.

How?
The Eclipse community adopted the OSGi standards, and worked together with the OSGi community to evolve the OSGi standard into additional focus areas around desktop and tools.
Dozens of adopters and growing...

**Vendor Product Dev Tools**

**Eclipse IDE**

- Eclipse Tools

**RCP**

- SWT / JFace

**OSGi (Standards)**

**Java™ Runtime (JavaSE™)**
Maturing the Eclipse Foundation

Unleashing the configurable runtime era
June 2005

Eclipse 3.1 continues to evolve and mature the tools. The Rich Client Platform (RCP) is introduced for both desktop and handhelds.

The OSGi standard on which it is based is formalized into a separate deliverable (Equinox).

June 2006
Eclipse 3.2 is delivered. Equinox becomes an Eclipse top level project – the Equinox project.

“And the beat goes on and on…”
Equinox implements the OSGi core component framework specification enabling a symmetric component, service, and deployment programming model spanning servers, clients, and tools.

RCP, adds the ability to blend both new and existing desktop and web applications across operating systems.

IDE, exploits the integration capability of RCP to enable a cross operating system tooling eco-system.
The Enterprise
(The Superhighway)
Superhighway Organizing the grass roots evolution

The Benefits of an Open Service Oriented Architecture in the Enterprise

Craig Hayman
VP, WPLC Development and Technical Support
IBM Software

Keynote

Integrating the OSGi Service Oriented Architecture into the Enterprise

Jim Colson
Distinguished Engineer
Chief Architect – Client Software

OSGi Alliance

2005 Developer Forum & World Congress
Intercontinental Paris Hotel • Paris, France

OSGi Release 4 – Driving Dynamic Services to the Next Level

October 11-14, 2005

Register by September 1 and save over $800 (€665)

An emerging demand for a server side component model
Superhighway

Grass roots demand: Example – WebSphere

IBM WebSphere® Application Server V6.1

Componentization Overview

May 2006

WebSphere begins exposing OSGi in v6.1

V6.1 class loader hierarchy

- Runtime is loaded by a network of OSGi classloaders
- Extensions class loader loads globally shared third party code and also acts as a gateway into the runtime
- Application and shared library classloaders unchanged from V6.0

Summary

- WebSphere Application Server V6.1 is delivered as a componentized set of OSGi bundles
  - Each bundle is loaded by its own class loader
  - Class loading changes have no effect on application class loading
- Runtime JAR files have been moved from the lib directory
  - Client applications should depend on new “thin client” libraries instead
- Application code can be restricted from accessing internal runtime implementation classes
Superhighway

Grass roots demand: Example – Spring

March 2006

SpringSource publishes a “bug” addressing how Spring / OSGi work together. [https://jira.springframework.org/browse/SPR-1802](https://jira.springframework.org/browse/SPR-1802)

Work on enabling Spring Framework using OSGi began soon thereafter. [http://groups.google.com/group/spring-osgi/](http://groups.google.com/group/spring-osgi/)

Interface21, BEA and Oracle were the most active players in the group.

The Spring–OSGi architecture refocused in the Enterprise Expert Group (2007) which resulted in the OSGi Blueprint specification from OSGi as well as the Spring Dynamic Modules project which serves as the OSGi Blueprint specification reference implementation.
Superhighway

Grass roots demand: Example – Glassfish

April 2008 – Glassfish V3 switching to run on OSGi

- From Jerome Dochez’s (GlassFish architect and a Principal Engineer at Sun) web blog http://blogs.sun.com/dochez/entry/glassfish_v3_runs_on osgi “GlassFish V3 runs on OSGi”, 2008–04–11:
  - “Now the interesting question that everyone will be asking soon, are we switching to OSGi as our underlying module subsystem? Today I can say yes, we will. Some people might say that we changed our mind about OSGi, we didn't. From the beginning I always said we wanted to be friendly to OSGi, we just realized that vision... It is pretty clear that there is a big industry support for OSGi and it is important that GlassFish can be part of that excitement.”
  - “We are still capable of running in both HK2 mode and in OSGi mode, I am not sure how long we will maintain the HK2 mode...”

- From Sanjeeb Kumar Sahoo’s (“Sahoo”) web blog http://weblogs.java.net/blog/2008/04/15/glassfish-v3-osgi-part-i, Engineer, Sun Microsystems, 2008–04–15:
  - “We have implemented HK2' module layer APIs on OSGi by delegating to OSGi module layer... ...What are the drawbacks of this approach? By limiting ourselves to HK2 APIs, we are not able to take advantage of rich module management APIs that OSGi provides. Will we continue to support both the modes? Not very long...”

  - “GlassFish v3 is now based on an OSGi R4 compliant runtime (Apache Felix by default) and it is delivered as a set of OSGi bundles. Modularity is important for the application server, since it allows quick consumption of libraries and reusable components developed by the community.”
  - “We wanted to have OSGi support, because it's clearly the leading solution for modules, and it is important for GlassFish to be able to consume OSGi services directly.”
OSGi Enterprise Expert Group (EEG)

March 2010

OSGi Service Platform Enterprise Specification Release 4, Version 4.2 is published
Advancing the state-of-the-art for constructing robust, modular Enterprise applications

The path to delivery:

September 11th, 2006

Enterprise Workshop
http://www.osgi.org/EnterpriseWorkshop/HomePage
Open invitation to anyone.

January 2007
OSGi Enterprise Expert Group (EEG) formed
FOR IMMEDIATE RELEASE

WORLD-LEADING ENTERPRISE APPLICATION SERVER PROVIDERS PROPEL OSGi TECHNOLOGY TO AN INDUSTRY-WIDE STANDARD

IBM, Oracle, Paremus, ProSyst, Red Hat, SAP, SpringSource and Sun Microsystems

Use OSGi technology in current and next-generation Java technology-based servers

SAN RAMON, Calif. – Sept. 16, 2008 – Market-wide adoption of the OSGi™ Service Platform has made it a de facto industry standard for modularization in enterprise application servers, the OSGi Alliance announced today. The leading providers in the market have demonstrated their support of the platform through the adoption or planned deployment of OSGi technology in their enterprise application servers.

“With the lion’s share of the enterprise application server market deploying OSGi technology, the alliance has created the dynamic module system for Java™ technology,” said Stan Moyer, president of the OSGi Alliance. “The OSGi Service Platform delivers universal middleware for Java to providers and their customers, modularizing and componentizing the Java platform and allowing applications to be adapted remotely and in real time.”

Leading vendors using OSGi technology include IBM’s WebSphere, Oracle’s WebLogic, Paremus’ Infiniflow Service Fabric, ProSyst’s ModuleFusion, Red Hat’s JBoss, SpringSource’s SpringSource Application Platform and Sun Microsystems’ GlassFish Enterprise Server. Both Oracle and SAP AG have announced that they will use OSGi technology as the foundation for their next-generation application servers.
The foundation of the road (The underlying runtime)
Special Guest

Tim Ellison
VP, Apache Harmony & Java Strategist, IBM UK
Apache Harmony

- Top level project at the Apache Software Foundation

- Charged with:
  “the creation and maintenance of open-source software implementing Java virtual machines, related class libraries, and other software that is commonly associated with Java platforms, for distribution at no charge to the public”

- Has grown significantly over the last 5 years and is now directly managing over 1.5 million lines of code implementing better than 99% of Java 5 SE APIs and 96% of Java 6 SE APIs
Apache Harmony – everything is pluggable
Modularity in Harmony – benefits to software engineering

- New code base allowed us to build in modularity from the start across VM, JIT, GC, Class library, tools, etc.

- Class library comprises ~30 modules

- A module...
  - is related functionality scoped by Java packages
  - defined by dependencies in the Java specification
  - ‘exports’ user-API and internal-API, hides private internal implementation
  - minimizes coupling by explicit internal APIs
  - is delivered as a JAR file, a real OSGi bundle
Modularity in Harmony – unexpected social benefits

- Modularity is “fundamentally good software engineering”, but also brought other somewhat unexpected community benefits...

- **Vocabulary**: gave us precise and concise names for functional areas
  - Everyone knows what we mean by “beans”
  - Created areas where people can take natural ownership
  - Can switch in alternative implementations, safely!

- **Minimize coupling**: explicit modularity reduces spaghetti code
  - Updating the Manifest is a deliberate act, and scrutinised by the community
  - Ability for users to selectively *consume* modules
  - Clear boundaries between Harmony code and dependencies

- **Innovation**: replacement and composition combinations
Harmony Select – fit for purpose runtimes

- Detailed package list at [http://s.apache.org/uD](http://s.apache.org/uD)
Demo

• ...of a modular, composable runtime
Apache Harmony – future work

- Many ways in which we can build on the current capabilities...

- **Runtime support** *(a.k.a. “OSGi right down to the metal”)*
  - Enhance the Harmony JVM with OSGi management support
  - Dynamic download / install for modules
  - Version control for side-by-side runtimes

- **Componentized runtimes**
  - Ability to compose custom runtimes from modules with given characteristics
    - Realtime, qualities of service, debug enhancements, ...
  - Cloud-aware (multi-tenant) implementations of runtime modules

- **OSGi evolution**
  - Work with the OSGi Alliance on runtime execution environments
Summary
A pattern has emerged

Bespoke modularity design has been ... ...repeatedly abandoned in favor of OSGi participation, evolution, and adoption.

- HomeAPI -> OSGi
- Eclipse “plugins” -> OSGi++
- Enterprise servers – many examples -> (OSGi++)++
Summary

• Observe the pattern
  - OSGi continues a decade of participation, evolution, and maturity/adoptio
  - A track record of providing modular, service-oriented, cross-platform architec
  
• Contributions and leadership from some of the greatest computing pioneers
  - Multiple implementations demonstrate the robustness of the specification

• Responsive and evolutionary with use cases spanning embedded, mobile, desktop, and servers
  - Established in the industry and adapted over the years to address modern problems

• Has become the "go to" convergence point in light of many attempts for "something new"
  - Modularity, versioning, life-cycle management, deployment, security, ...
  - It’s hard to get all this right => innovate on top of the established science.
The Road Ahead
Today’s runtime modularity landscape

- Today’s Harmony landscape:
  - OSGi. Metadata done. Runtime enforcement – future work.
  - Adapting OSGi to a new domain

- Today’s JavaSE™ landscape:
  - JSR 277 “Java Module System”; started June 2005
  - JSR 294 “Improved Modularity Support in the Java Programming Language”; started April 2006
  - Jigsaw described as replacing JSR 277 in December 2008
  - Modularity deferral to Java 8 (late 2012) announced Sept. 2010
  - Documentation of the timeline (through 2008) at:
    http://www.osgi.org/blog/2008/12/project-jigsaw.html
• Learn from and follow the pattern of successful pioneers
  - Explore and drive new use-cases for today’s OSGi.
  - Drive the OSGi specification’s evolution.
  - Adopt the result.

• Platform modularity has the focus of engineers
  - Go with it and accelerate adoption.

• The world continues with one module system.
  - Fragmentation and lack of interoperability is not a competitive advantage for modularity.
The Road Ahead  Next …

- “The Sky is the Limit”

- Explore and experiment
  - Where are there similar problems on a large scale today?
  - Push the boundaries!

- “As ubiquitous as the air you breath”
  - Cloud? Virtual worlds? Unknown horizons?
  - OSGi has a history of inclusiveness, malleability, and uncharted waters.