RT Symposium at Eclipse Summit Europe 2008

organized by

Jeff McAffer (Code9)

Heiko Seeberger (WeigleWilczek)

Martin Lippert (it-agile)

Outline

- Walkthrough the answers
- Open-Space-kind of topic collection and selection
- Discuss each topic area for 30min each (3x)
- Fishbowl at the end

- Have a coffee break in between

Personal Introductions

Keep it short!!! (~30sec)

- My name is …
- I work for …
- I am here because …
- My goal for this symposium is …

Maturity of runtime technologies

- Adoption of OSGi as base technology
- Programmability
- Robustness
- Completeness
- Cross platform / single sourcing (e.g. RAP, eRCP, RCP)

Putting systems into production usage

- How to develop
- How to build
- Provisioning

Synergies

- Do not reinvent the wheel
- Uniformity of different runtime technologies?

The answers to our questions

What is missing?

- The biggest thing missing is a good set of **realistic reference implementations**. There is a bunch of cool technology out there, but folks are more or less left to hack their own path or beg for advice from the message board. There reference implementations need to be addressing a set of realistic scenarios, not just toy examples. It's hard to make these, and even harder to maintain them, but I think it's one of the biggest barriers to adoption.

What is missing?

- OSGi pretty much is feature complete from a Java perspective. Though it might be desirable to easy **integration of non Java components**. This would allow reuse in a world full of legacy applications and with gazillions lines of non Java code. They keyword here is probably "unified OSGi" first read on Peter Kriens' blog.

What is missing?

- Component model (OSGi R4.2) like Spring DM
- Distributed runtimes (OSGi R4.2) to solve part of OSGi Application Grouping needs
- True modular "per bundle" persistency support: every bundle should expose its own data model to other bundles. A persistency manager bundle using the extender model should match/resolve bundle data models constraints (PK, FK, etc)
- Strict mode from packages resolution: in the long run, package exported without versions boundaries specified hurts manageability
- JMX and OSGi (though solutions seems to be coming up)

What is missing?

- I, for one, do not consider the different Eclipse run-times just one technology. From a technical point-of-view there are many commonalities, but from users point-of-view, the technologies are very different. Consequently, I don't think we have talk about "What is missing from the technology?"

What is missing?

- This is not really technology related but when Eclipse started some years ago, it provided a nice tooling that allows new developers to start developing new Eclipse plug-ins with minimal effort. Compared to this it is really hard to get something running from most of (not all!) the new RT projects. That being said: The RT projects deliver great technology, but in some cases the tooling could be improved and the bar could be lowered.

The three most important problems?

1. Lack of reference implementations
2. Difficulty of building and deploying an application
3. Biased toward building tooling, not runtimes. Biased toward the desktop, not headless embedded or servers.

The three most important problems?

1. Too few OSGified libraries so far (evangelism)
2. Lack of tooling for querying bundle imported/exported services
3. Dependencies management: finding correctly OSGified bundles, managing transitive dependencies, managing the number of dependencies on large scale projects

The three most important problems?

1. Focusing on the RCP run-time, the primary problem currently seems to be in the deployment. Everybody agrees that the old update functionality did not work properly, but to get the new p2 stuff to work is difficult...

The three most important problems?

1. Complexity
2. Manageability
3. convergence of different existing tools

The three most important problems?

1. Working with Target Platforms should be much easier: I would like to use one workspace and decide "these projects belong to Target Platform x and those to y".

Biggest implementation challenges

- Turning non OSGi Java applications into good citizens in the OSGi world. While this is certainly nearly impossible to automate, at least excellent tooling like PDE would lower the cost for reuse.

Biggest implementation challenges

- Design proper bundles granularity
- Ensure cohesive platform while maximizing loose coupling principles
- Proper balance between unit (mock) and integration testing

Biggest implementation challenges

- 10.000 foot view: All RT projects seem to be projects of its own but the question is: How do they interact, how to get the best value from them, what's the best way of integrating them from the RT user point of view?

Biggest implementation challenges

- there should be an easy way to generate a server application directly from an OSGI Framework Launch Configuration

Vision for the future

- I would like to see better adoption of OSGi and services in particular inside the implementation of Eclipse. It would be wonderful to see an RCP-like platform based purely on OSGi services.

Vision for the future

- Projects listed under RT should required to not just work on Equinox, but also on other (all) OSGi runtimes. Do not use Equinonx specifics. Interoperability is a strong selling point and excels broader adoption. Why lock ourself out of possible use cases/markets? This certainly requires constantly efforts during development. E.g. a build and test suite that is runtime implementation independent is a must.

Vision for the future

- Closer integration with dependencies management tools like Ivy or Maven
- Merge of Project Models (PDE, Maven, Ivy) specs
- On-demand deployment of bundle sets to Cloud Computing services

Vision for the future

- I think many of the goals of the e4 project is correct and sound, but I fear they will come up with a revolutional approach rather than a evolutional approach! Whatever we do, the current RCP based applications are so many that we cannot leave them behind. On the shorter term, I hope to see scripting in the platform! But how to do this across the run-times, I don't know.

Vision for the future

- The Eclipse RT provides a reliable OSGi RT implementation which enables the seamless dynamic execution of components on a big variety of devices (from embedded via handheld via desktop via serve to cloud). The Eclipse RT contains the tools for dynamic management of distributed systems including the enforcement of SLA and QoS.

Vision for the future

- an eclipse server project like RCP for RichClient and RAP for Web

What should projects do?

- Common build environment (see Bug #XXXXX)
- Common testing (check if everybody behaves like a good citizen) See Bug about unified testing, monthly release train that doesn't get published.

What should projects do?

- Orbit project to move to a repository like SpringSource repository
- EclipseLink to work on modular persistency support

What should projects do?

- g-Eclipse together with Cloud Computing technologies could be hosting exemplary applications of the RT projects and give an overview about how those technologies can be used together.

What should projects do?

- It should be easy to combine different eclipse server technologies, to have an eclipse.server.core and then to add what you need: Riena Remoting, ECF - XMPP etc.

What confuses people?

- The Eclipse programming model of extension points is different from OSGi's programming model of services. This must confuse people quite a bit.

What confuses people?

- All the redundancy between OSGi and the early Eclipse component model: Log, EP vs. Services...
- Missing examples in the OSGi compendium.

What confuses people?

- Too many ways of doing the same thing (old APIs, new APIs, etc.)

What confuses people?

- Currently there seems to be many approaches from the different "incubation" projects, which makes it hard to decide which one would last.

General topics

- App-server-models for OSGi
- Deliverables of RT projects for Galileo
  - More than just a few zips?

Position Paper from Gunnar Wagenknecht

- Slides inlined

RT Symposium – Position Paper

November 3, 2008

Blog:	http://wagenknecht.org/blog/
Mail:	gunnar@wagenknecht.org

Dynamic Elements Today

- OSGi Services
  - Very dynamic, coupled to bundle start and stop
- Equinox Extension Registry
  - Somewhat dynamic, coupled to bundle resolve and uninstall
  - Forces use of singleton
- Equinox Adapter Manager
  - Somewhat dynamic, coupled to bundle start and stop and bundle resolve and uninstall

Contextual Runtime in CloudFree

- Runtime execution differs based on a specified context
  - Possible today: a bundle contributed by A is not allowed to access service B
  - Not possible: a bundle contributed by A is not allowed to see extensions contributed by B to extension point C
    - Need to filter extensions
  - Not possible: adapters should be different based on configured permissions (eg., myObject.getAdapter(aClass.class, aContext))
    - Need to filter adapters

Discussion Topics

- Is this interesting for a broader community (SaaS in general)?
- What is the best way to implement it?
- How can we support Equinox in providing those enhancements?

And now: Something like Open Space