Management Platform, Sum of Technologies (part 1).

written by Manoj Khanna | September 28, 2005

Since no one attempted to disprove statement I made in my last post (Management platform, Open source or bust. 09/15/2005) I will assume that every one think that Open Source Management Platform is a brilliant idea (just for the records, I can not claim ownership of this particular idea, but it does not make it any less brilliant). Today I will try to create “shopping list” of other Open Source projects which could be used as a building blocks for such Platform.

But of course, Java

This is not per-say open source project, but none of the less choice of “the language” will play major role in selection of Open Source projects available for consideration. In my mind, there is only one answer to this – Java. Despite relentless attempts by Mr. Gates, enterprise networking landscape is as diverse as ever. And the only way to avoid porting nightmare is to do Java. It might not be 100% “write once, run anywhere”, but for none-GUI applications it is damn close. Last thing to consider in regards to “the language” is what version of Java to use? It is very tempting to go along with Java 5.0 but I think we need to stick with 1.4. Though Java 5.0 provides bunch of new functions/features none of them are “must have” kind and I have a feeling that its proliferation in the “enterprise world” is not that wide yet (just a couple years ago I run into customer who still used 1.1.8)

Built to be customizable

There are at least two level of customization which need to be addressed:

ability to add new functional modules (plugins),

ability to implement new functions by combining basic functions provided by plugins.

Many Open Source (and closed source) applications implement

support for plugins in one way or another. I think JBoss does it best. JBossMX is clean room implementation of Sun JMX API. It is available as a part of JBoss server source distribution. One additional benefit of using JBossMX as a “container” is ability to do “cross-plugging”. It should be possible to plug JBoss modules into the Platform and Platform modules into JBoss!
With such a flexible container as a JBossMX it will be a shame to require to do any Java codding to implement “new functions”. Also if we are trying to build Platform which could be “customized” by end user it is not reasonable to expect such an end user to know how to program in Java. What we need is nice scripting language. It should be:
well known (we do not wont to invent our own and then expect people to learn it)

- reasonably powerful
- easily extendable
- with available Java binding.

Choosing the scripting language is always rather “emotional” affair. Every one has its own favorite. I have my personal favorite too – JavaScript. And to be more exact – Rhino JavaScript engine from Mozilla folks. Let see if it satisfy requirements we set for the scripting language. No one will object that JavaScript is “well known”. What about “reasonably powerful”? To my taste it strike nice balance between been OO language (you can “simulate” such OO features as class definitions, instance and class members, class inheritance, …) and been typeless forgiving scripting language. Rhino provides us with two level of extensibility: Host Objects and direct to Java interface. Between these two it is possible and very easy to add any kind of functionality to it. Some people will say that Java itself is not exactly speed daemon and Rhino (been scripting language implemented in Java) must be even slower. There are two answers to this concern:

- Java is plenty fast if used properly and Rhino in most

cases at least as fast – it actually capable of producing Java byte code,
- We are talking about “scripting” language. The glue to assemble new functions from basic functionality provided by plugins. In most cases all “heavy lifting” should be done in plugin code, so actual performance of scripting engine is more or less irrelevant.

Networking

It goes without saying that our Platform will have to be distributed. Many components of it will be spreaded across enterprise network. These components will have to learn about each other existence and they will have to be able to communicate with each other. To make things even more “interesting”, most likely these components will be deployed into many different sub-nets with many network devices (like routers and firewalls) between them. It is almost guarantied that many of them will not have a luxury of “direct communications” to each other.

I might be biased, but as far as I know the only Open Source project which fit the bill is JXTA. It has Java binding, it provides excellent set of “discovery” functions, it provides ways to communicate in “request-response” and “fire-and-forget” manner and it does all these while completely hiding physical network topology. You can have 10 firewalls and 20 routers between two of your JXTA applications and still from programming point of view it will not look any different from them been running on the same sub-net. As an added bonus, JXTA has Java ME binding (for those “itsy bitsy” devices which can not support full Java) and C++ binding.

Conclusion (of part 1)

So far our platform looks like JMX container (courtesy of JBossMX project) with two plugins in it:

- Rhino Java Script Engine (with collection of Host Objects which provide access to any other plugins loaded into the local or remote JMX container)

- JXTA plugin (providing platform components discovery and communicate functionality)

Next time we are going to discuss Storage requirements and ways to interface with outside world entities (like humans, report writers, …)