Business Leadership

written by Manoj Khanna | October 9, 2005
Business Leadership is a very important subject. Through this blog I plan to create a thread of my thoughts about businesses and their leaderships. Not necessarily in the order they are in different industry domains. Through different readings in books, mags, and on-line I intend to capture the gist and present it here.

This is an important subject from academic stand-point as well. Though, biz school adopt a new approach almost every year to tackle this subject but the practical walkthrough of any industrial domain says and talks a lot on itself. Probaly, it would be wise to say that this cannot be taught in text. But rather, be experienced.

On the oher hand, the good thing is, that the busines today have started realizing the importance of it organization’s demands and priorities. There are more things to look into than profit and maximization.

More to follow…

Business Leadership

written by Manoj Khanna | October 9, 2005
Business Leadership is a very important subject. Through this blog I plan to create a thread of my thoughts about businesses and their leaderships. Not necessarily in the order they are in different industry domains. Through different readings in books, mags, and on-line I intend to capture the gist and

present it here.

This is an important subject from academic stand-point as well. Though, biz school adopt a new approach almost every year to tackle this subject but the practical walkthrough of any industrial domain says and talks a lot on itself. Probaly, it would be wise to say that this cannot be taught in text. But rather, be experienced.

On the oher hand, the good thing is, that the busines today have started realizing the importance of it organization's demands and priorities. There are more things to look into than profit and maximization.

More to follow…

Management Platform, Sum of Technologies (part 1).

written by Manoj Khanna | October 9, 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, …)

Management Platform, Sum of Technologies (part 1).

written by Manoj Khanna | October 9, 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, …)

Management platform, Open source or bust.

written by Manoj Khanna | October 9, 2005

When you start to think about Enterprise Management Platform (EMP), few questions pop into the head:

- Do we really need one?
- What is wrong with “legacy” platforms?
- Why no one came up with “new thing” yet?
- Does open source really has an answer?

Do we really need one?

I guess the answer to this question depends on what is it we are talking about when we are saying “Enterprise Management Platform”. If we are talking about piece of software which can “discover” all computers on your network, show them on nice and colorful console and then ping them once in a while to make sure that they still plug-ed in – than the answer is most definitely NO. On the other hand, if we are talking about piece of software which can easily integrate “best of the breed” discovery tool with “best of the breed” reporting tool and than provide you with ability to define what exactly you mean when you are saying that Managed Object XYZ is “in groovy state”, then the answer could be quite different.

Basically Enterprise IT Universe consists of so many different things and needs to be managed from so many different “angles”, that it is impossible to put all needed knowledge into single product. So Enterprise Management Platform should not even try to do any “management” but instead should

transform into “Enterprise Management Tools Collaboration Platform”.

What is wrong with “legacy” platforms?

But wait, “That Nice and Great” platform has endless list of options. Surely they can cooperate with any “best of breed” tool you care to throw into it. And “Highly Popular Old Working” platform can do web services. Should not it be enough? What about “Best Mechanical Calculator” platform? It was built around “script everything” concept. If you want to deal with “groovy state” or XYZ object, all you need is to write another script.
Unfortunately, it is not that simple. None of the “legacy” platforms were built with much emphasis on “Ease of Tool Collaboration”. There were no reasons for it back then. Companies which developed these platforms truly believed that they can “boil the ocean” and “cure common cold” of Enterprise Management all by themselves (maybe with few acquisitions). Best of them, did have some sort of APIs to let 3rd parties to do some relatively minor additions which were too “small of a fish” to go after. None of them were too eager to let customers define what he/she wants to manage and how. They had very good reason not to make this kind of customization too simple. All of these companies have huge Customers Support teams and they make as much money (and sometimes even more) by “adapting” their software to customer real needs as they make by selling it. If you add to all this the fact that most of the code in these “legacy” platforms was written 5 or even 10 years ago, it will become increasingly clear that “legacy” platforms is a wrong place to look for “The Answer”.

Why no one came up with “new thing” yet?

The answer is very simple – money. Big companies which can afford R&D expenses associated with development of a new shiny platform, do not have any incentive to do it. They are mildly interested in making their own platforms a little bit more

robust/reliable/scalable, but they have all that "compatibility skeletons in the closet" to deal with so they can not do anything drastic. At the same time, they have no interest at all in making platform easily adaptable to customer needs (what all these folks from Customer Support will do?). Small innovative companies just can not cope with this kind of problems. Let's face it, there is no money in the plumbing. No one in their right mind will spend millions of dollars to develop something they can not sell.

Does open source really has an answer?

I guess you will never know for sure until you try. But Open Source have some unique features which can make this kind of project a possibility. We can start with most obvious things, like bunch of highly skilled developers which could join such project for all the different reasons people do join Open Source Projects. Then we can talk about almost endless plethora of existing Open Source Projects which could provide building blocks for such project. It could be anything from something huge like JXTA to something small like Apache Commons. Open Source at the moment reached the "critical mass" when you can build almost anything just by combining existing pieces. The last thing in favor of Open Source is the fact that such project will be vendor neutral. Hopefully other companies will not consider it as a threat. Maybe they even will support it (it did become cool lately to support Open Source Projects).

Management platform, Open

source or bust.

written by Manoj Khanna | October 9, 2005

When you start to think about Enterprise Management Platform (EMP), few questions pop into the head:

- Do we really need one?
- What is wrong with “legacy” platforms?
- Why no one came up with “new thing” yet?
- Does open source really has an answer?

Do we really need one?

I guess the answer to this question depends on what is it we are talking about when we are saying “Enterprise Management Platform”. If we are talking about piece of software which can “discover” all computers on your network, show them on nice and colorful console and then ping them once in a while to make sure that they still plug-ed in – than the answer is most definitely NO. On the other hand, if we are talking about piece of software which can easily integrate “best of the breed” discovery tool with “best of the breed” reporting tool and than provide you with ability to define what exactly you mean when you are saying that Managed Object XYZ is “in groovy state”, then the answer could be quite different.

Basically Enterprise IT Universe consists of so many different things and needs to be managed from so many different “angles”, that it is impossible to put all needed knowledge into single product. So Enterprise Management Platform should not even try to do any “management” but instead should transform into “Enterprise Management Tools Collaboration Platform”.

What is wrong with “legacy” platforms?

But wait, “**T**hat **N**ice and **G**reat” platform has endless list of options. Surely they can cooperate with any “best of breed” tool you care to throw into it. And “**H**ighly **P**opular **O**ld **W**orking” platform can do web services. Should not it be enough? What about “**B**est **M**echanical **C**alculator” platform? It

was built around “script everything” concept. If you want to deal with “groovy state” or XYZ object, all you need is to write another script.
Unfortunately, it is not that simple. None of the “legacy” platforms were built with much emphasis on “Ease of Tool Collaboration”. There were no reasons for it back then. Companies which developed these platforms truly believed that they can “boil the ocean” and “cure common cold” of Enterprise Management all by themselves (maybe with few acquisitions). Best of them, did have some sort of APIs to let 3rd parties to do some relatively minor additions which were too “small of a fish” to go after. None of them were too eager to let customers define what he/she wants to manage and how. They had very good reason not to make this kind of customization too simple. All of these companies have huge Customers Support teams and they make as much money (and sometimes even more) by “adapting” their software to customer real needs as they make by selling it. If you add to all this the fact that most of the code in these “legacy” platforms was written 5 or even 10 years ago, it will become increasingly clear that “legacy” platforms is a wrong place to look for “The Answer”.

Why no one came up with “new thing” yet?

The answer is very simple – money. Big companies which can afford R&D expenses associated with development of a new shiny platform, do not have any incentive to do it. They are mildly interested in making their own platforms a little bit more robust/reliable/scalable, but they have all that “compatibility skeletons in the closet” to deal with so they can not do anything drastic. At the same time, they have no interest at all in making platform easily adaptable to customer needs (what all these folks from Customer Support will do?). Small innovative companies just can not cope with this kind of problems. Let’s face it, there is no money in the plumbing. No one in their right mind will spend millions of dollars to develop something they can not sell.

Does open source really has an answer?

I guess you will never know for sure until you try. But Open Source have some unique features which can make this kind of

project a possibility. We can start with most obvious things, like bunch of highly skilled developers which could join such project for all the different reasons people do join Open Source Projects. Then we can talk about almost endless plethora of existing Open Source Projects which could provide building blocks for such project. It could be anything from something huge like JXTA to something small like Apache Commons. Open Source at the moment reached the “critical mass” when you can build almost anything just by combining existing pieces. The last thing in favor of Open Source is the fact that such project will be vendor neutral. Hopefully other companies will not consider it as a threat. Maybe they even will support it (it did become cool lately to support Open Source Projects).

New Destination – Mars! Are we spaced out?

written by Manoj Khanna | October 9, 2005

This article published at WIRED definitely brings out a good question. Are we spaced out? Also, it brings out where the new information age may lie. Development of applications and their IT innovation is not limited but still a whole new scope lies beyond the realm of the usual IT world. The exciting and new challenges might be seen there. May be in next 30 years or so we have a new platoon of IT outsourcers specializing off-planet-shoring!! You never know. Nevertheless its interesting to read whats surrounding us. Thanks to WIRED!

Check the article here or click above.

New Destination – Mars! Are we spaced out?

written by Manoj Khanna | October 9, 2005
This article published at WIRED definitely brings out a good question. Are we spaced out? Also, it brings out where the new information age may lie. Development of applications and their IT innovation is not limited but still a whole new scope lies beyond the realm of the usual IT world. The exciting and new challenges might be seen there. May be in next 30 years or so we have a new platoon of IT outsourcers specializing off-planet-shoring!! You never know. Nevertheless its interesting to read whats surrounding us. Thanks to WIRED!

Check the article here or click above.

Open Source – today and tomorrow

written by Manoj Khanna | October 9, 2005
Apache Group started a small revolution. A simple **http server** – which has been the quite simplest ever. And most of all "it works". I'm not surprised by the success of **Mozilla's FireFox**. Or for that matter **Linux**. That small revolution started by Apache and followed by others has today given a new direction

to the enterprise computing community.

These events were waiting to happen. In a world of technological chaos someone has to invent simplicity and ease of use. The data structures, stacks and linked lists all makes a perfect sense when they are made to do exactly what they are made to do. A system which works seamlessly might be like wanting a perfection but its quite not happening yet. But we are still getting there. Although in the plain IT terms – we're getting there.

Likewise, when the thought process for the IT thinkers was taking different shapes at just about the same time some different thinking persuaded a different set of IT herd that brought about just the change which was wanted. In the economical sense – the need and the want. And it was the want that became the need later on looking at the wider picture on what all was happening around in the IT world.

Open Source today is the de facto for innovation. Tech experts and the high-rise tech corporations are not leaving a single strand to be missing on this opportunity.

What lies ahead? Open Source as a service provider. The bottom line for a innovation survival is funding. If the innovation continues to take place with no revenue then it might not last for long. Investors and VCs alike understand the potential for a service industry which is slowly taking its own shape in the form of open-source. The future right now is bright for Open-Source. And this ain't no dot-com-bubble. Noway.

Open Source – today and tomorrow

written by Manoj Khanna | October 9, 2005

Apache Group started a small revolution. A simple **http server** – which has been the quite simplest ever. And most of all "it works". I'm not surprised by the success of **Mozilla's FireFox**. Or for that matter **Linux**. That small revolution started by Apache and followed by others has today given a new direction to the enterprise computing community.

These events were waiting to happen. In a world of technological chaos someone has to invent simplicity and ease of use. The data structures, stacks and linked lists all makes a perfect sense when they are made to do exactly what they are made to do. A system which works seamlessly might be like wanting a perfection but its quite not happening yet. But we are still getting there. Although in the plain IT terms – we're getting there.

Likewise, when the thought process for the IT thinkers was taking different shapes at just about the same time some different thinking persuaded a different set of IT herd that brought about just the change which was wanted. In the economical sense – the need and the want. And it was the want that became the need later on looking at the wider picture on what all was happening around in the IT world.

Open Source today is the de facto for innovation. Tech experts and the high-rise tech corporations are not leaving a single strand to be missing on this opportunity.

What lies ahead? Open Source as a service provider. The bottom line for a innovation survival is funding. If the innovation continues to take place with no revenue then it might not last for long. Investors and VCs alike understand the potential for a service industry which is slowly taking its own shape in the form of open-source. The future right now is bright for Open-Source. And this ain't no dot-com-bubble. Noway.