Is the Future Open Source?

And what can I do to help

James Bottomley
Distinguished Engineer

Novell®

Introduction

- So, obviously, the answer to “Is the Future Open Source?” is “Yes”
- However, to realise this dream fully, Open Source has to be both a Commercial and an Ecosystem Success.
- This means that as a developer you have to be able to explain to a manager why Open Source makes Business Sense
- And as a manager you have to understand how Open Source can work for you Commercially
  - And how to motivate developers into contributing to it

Open Source and Economics
(For Managers)

- There's a lot of money in Open Source
  - Up to 2008, total investment in Linux: $2,000,000,000
  - Up to 2007, total revenue from Linux: $2,000,000,000
- The open nature of the platform makes it very easy to mold it to the latest commercial need.
- The licence permits any form of use
  - Provided you follow the rules of the GPL
  - Implies no deployment problems
- The open nature of the development ecosystem makes it easy to identify people who can help you
  - Either by hiring or paying them directly.
  - Or by collaborating with them through your own developers.

Open Source and Economics
(For Developers)

- There's a lot of money in Open Source
  - Up to 2008, total investment in Linux: $2,000,000,000
  - Up to 2007, total revenue from Linux: $2,000,000,000
- Problem is that not much of it trickles into the developer ecosystem.
- However, you can help your company make money in open source
  - Which means they let you spend time on it.
  - To do this, you need to understand the business justifications
- If you don't work for a Company yet, open source is a great way to showcase your skills
  - All the code is available and (some) recruiters know this.

Employers and Open Source

- Employer understanding of Open Source is very variable
  - Even in companies with stellar reputations in the area
- An employee who can relate an understanding of open source to an employer's business need is very valuable
- Once an employer is on the open source bandwagon, an employee who can participate in the ecosystem and train others to do so is valuable.
- Learning why open source makes Business Sense can be very useful to you (a developer!)

Justifying Open Source
(to your Employer)

- Corporations produce software as unit commodities
- The corporation (or IP lawyer) thinks
  - To control the commodity entirely, you need to control the code absolutely.
  - This is the intellectual property view: You need to own as much property as you can
- But
  - Code only has business value if it's differentiating
    - If everyone does it, customers won't pay extra because you can do it
- Explaining the difference between IP value and Business Value is where Open Source wins.
  - The pieces which have IP value but no Business Value might as well be Open Source

Common Fears

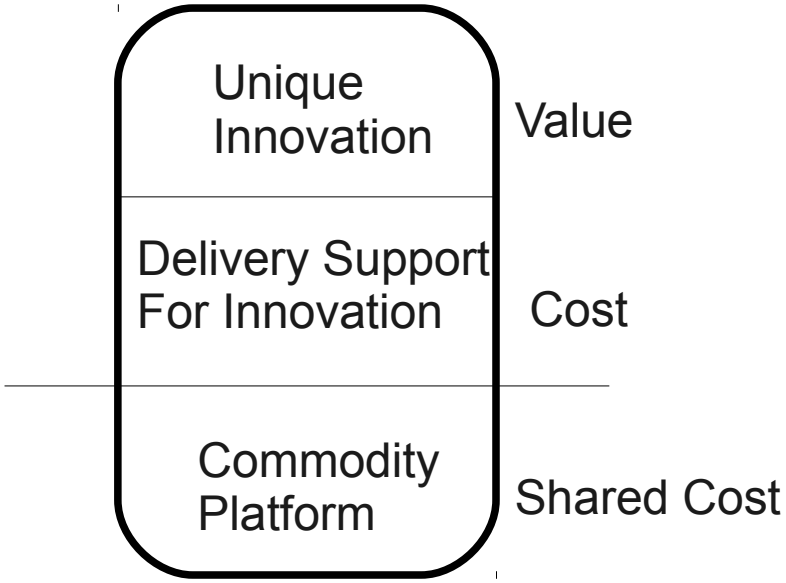
- Sharing code with IP value but no Business value is still releasing valuable IP and Giving up Control
- “value” argument easy to shoot down
  - If the code has no business value to the company, who else could you sell it to?
  - Sharing code actually adds value by sharing support
- “Control” is harder
  - However, very few commercial entities control the stack from top to bottom
    - Usually runs on hardware chosen by the customer
    - Probably runs on a customer chosen OS
    - OS tends to come with JVM, C library etc.
  - Thus “Control” is largely a myth anyway

The Code Value Equation

- The **True** value of an application is
  - What the customer is willing to pay for the function performed by the application
  - Less what it cost you to build (the Sunk costs)
  - Less the cost to you of supporting the code and fixing bugs in it (the Support costs)
  - Less the cost of forward porting to newer interfaces and platforms (the Maintenance costs)
- For old, proprietary applications, this value can be negative
- Open source allows the reduction of this negative value burden by sharing it.

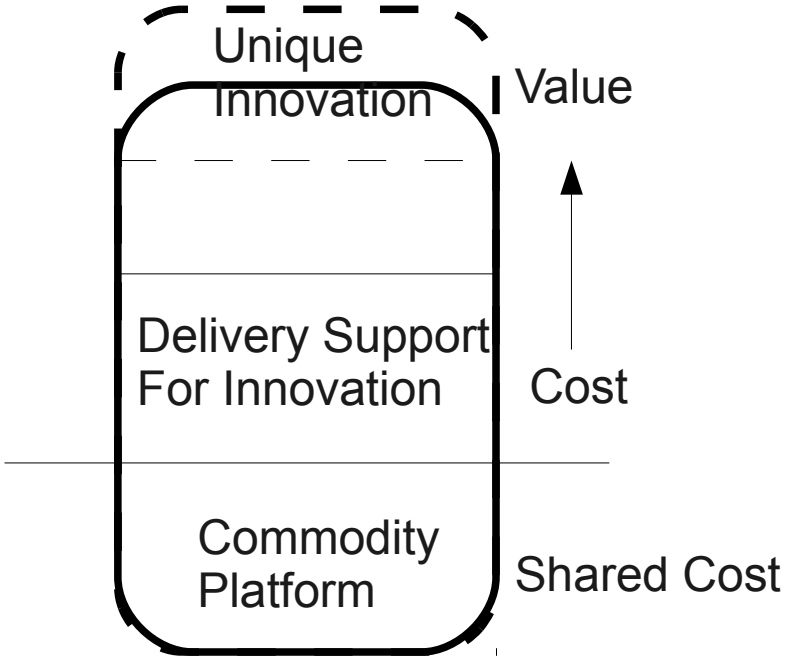
Project Innovation Diagrams

All projects start like this:



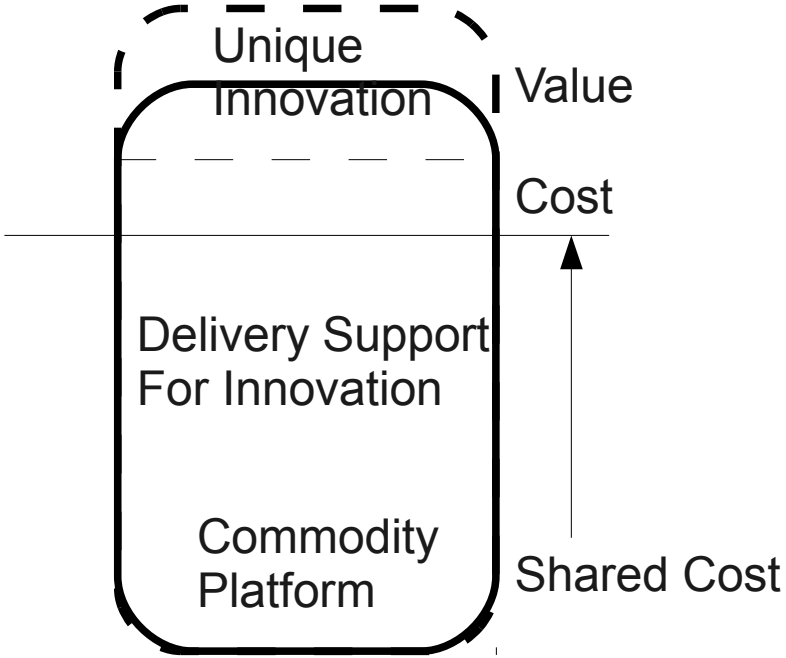
Project Innovation Diagrams

But as they get older, this happens:



Project Innovation Diagrams

If you move your Deliver Support into Open Source:



Innovation Diagram Takeaways

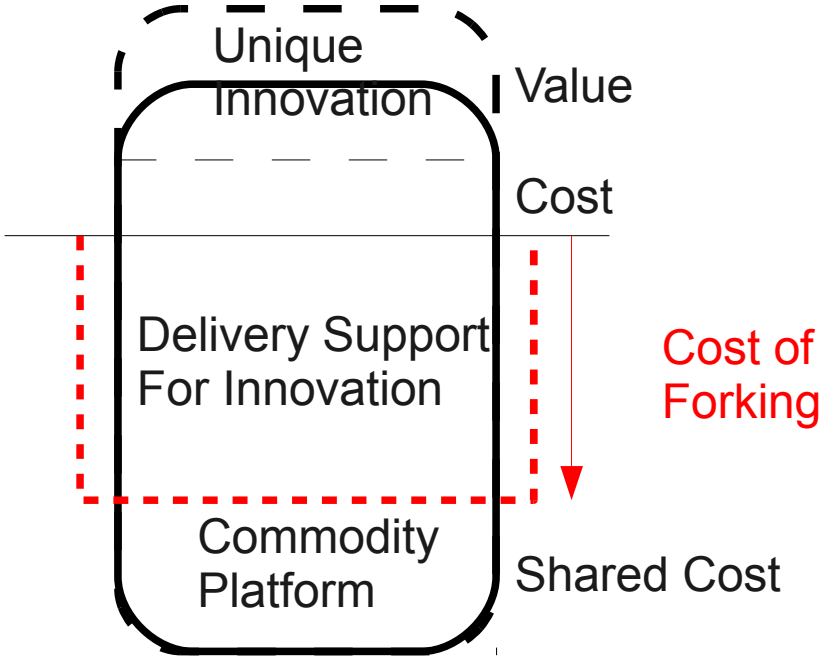
- Open Sourcing decreases cost by sharing
  - Move non-innovative, non-value providing components into open source to share maintenance costs.
  - Shared cost doesn't mean zero cost … how much you save depends on how well you share
- Ongoing projects can reduce or eliminate usual cost accretion by judicious use of open source
- The process of open sourcing as you go has some hidden dangers
  - Must be careful about licence contamination
  - Not just the GPL: You can't open source something that doesn't belong to you.

Licensing Problems

- Every Corporation's nightmare
  - Drives a huge amount of unfounded fear of the GPL
- GPL is easy to handle, provided you're careful
  - Firstly, design a “Bright Line” system
    - Make sure ABI between components is fully laid out
    - All components interact over the ABI (no code contamination)
  - If you open source components you link proprietary code to, make sure you use the LGPL
  - When you ship a system based on GPL code, make sure you publish all the necessary source code
- Finally, beware of shipping someone else's component
  - If they have a GPL violation, you may become liable

The Cost of forking

When you fork a project, you move the shared cost
Directly to become your cost



The Benefits of Forking

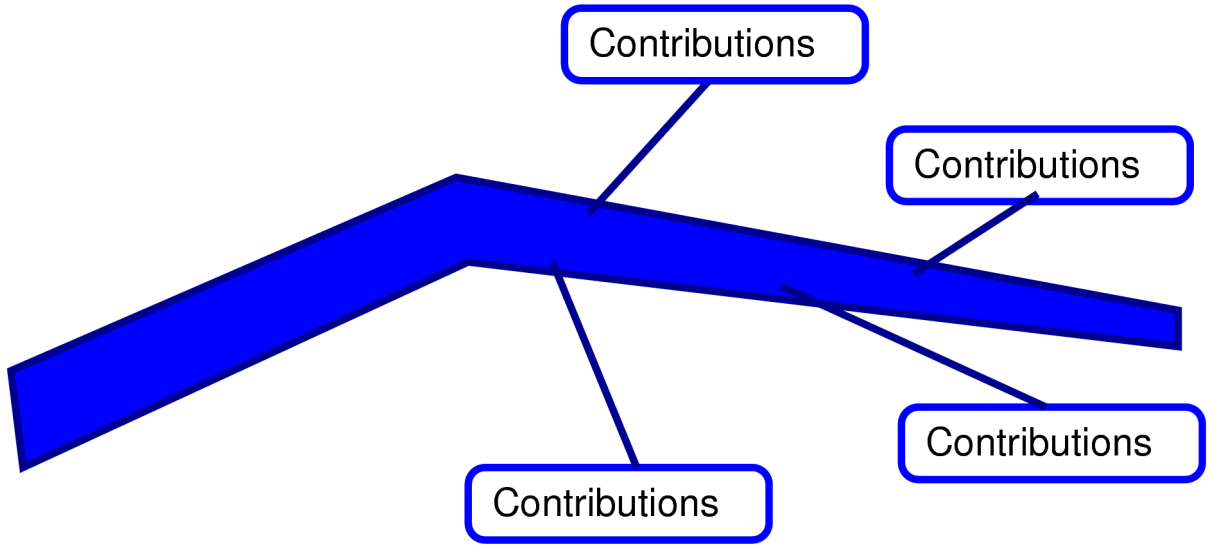
- Forking is a natural process that allows for experimentation within the ecosystem
- Thus Forking itself is not wrong
- The problems come if the fork lives a long time
  - Because whoever created the fork has to bear its costs for a long time
  - Plus a long lived fork separates the innovation stream
    - Developers usually either work on the fork or the mainline
- The moral is Fork often but Merge Early

Basic Economics
(for Developers)

- Corporations have effectively two economic benchmarks
  - Expense
  - Revenue
- Requirement is for Revenue minus Expense to be positive (in the black)
  - Corporations are in trouble when it's negative (in the red)
- One way to get into the black is to reduce expense
  - Previous slides about shared cost show how to do this
- Another is to increase Revenue
  - So lets see how Open Source can do this

The Innovation Stream

- Can think of Open Source contributions as tributaries of a stream
- The greater the number contributions, the bigger the stream



Broadening the Innovation Stream

- If a Corporation develops in Isolation, its innovation stream is only as broad as it's own personnel
  - Works well if you hire the best (Bell Labs in 70s-90s)
  - But this becomes very expensive, very fast
- Developmen in isolation is often being repeated all over the industry
  - Implies a huge waste of resources.
- Open Source facilitates sharing the innovation stream
  - Combined stream becomes as broad as all contributors
  - From every company which contributes
  - Much greater than a single Corporation can sustain

Selling the Shared Innovation Stream

- Concept is much more frightening
  - You're no longer simply commoditising components which no longer have any Business value
  - You're collaborating with your competitors on technology advances
  - You don't control anything that's created this way and your competitors have immediate access.
- However, you're not paying for all the innovation either
  - Shared innovation is often the only way of running a cost effective project delivering complex technology
- Plus the pace of development is far faster
  - In marketing terms, this develops Buzz (excitement) around the platform

“Controlling” the Innovation Stream

- In a truly open project, no-one has control
- However, like any stream, it responds to currents and flows to cut channels
- Therefore, Corporations making important contributions can influence the direction in which the stream flows.
  - Thus developers become essential advocates for features and direction needed by corporation
  - They also become Corporations ambassadors to the community
  - Even have to learn diplomacy
    - That's what negotiating with your competitors developers over how best to advance the system entails.

Know your Business Model

- So for Developers, this means you have to understand what a Business Model is.
- Traditional Open Source Business Model is selling support not software
- Newer Models include selling services on the side
  - To other companies (Google Ad Words)
  - Or to the consumer (Cloud based applications)
- Also using Open Source as an enabler for something else
  - Hardware (Intel, IBM, HP)
  - Services (see also Cloud Based Applications)

Japan Specific Problems

- Personnel structures in Japan are much more rigid
  - Open Source requires the developer (quite low in the hierarchy) to be the company representative to the community
  - Quite a few corporations see this as a threat to personnel control.
- Language Barriers
  - Open source development is all done in English
  - Code is written in C which is based on Western script
- Cultural Barriers
  - Linux is an extreme proponent of dialectic
    - Establishing correctness by Argument (often heated)
  - To participate, you can't be shy about your code.

Personnel structures can be co-opted

- A manager can be judged by the productivity and utility of his employees
  - For a good employee, visibility and peer judgment can substitute for this (small change from today).
- The trick is to manage the relationship pro-actively looking at the quid pro quo
  - Explain to your manager what you can do for him
  - Show how you can increase exposure for your company and thus the stature of your manager within the company.
- In Western parlance, this is called “employee enabling”
  - US companies stimulate innovation by encouraging their employees to play with technology

Linux is no longer Ignorable

- Most companies in software today need some contact with Linux in some form
- In order to participate in the communites, you have to have credibility
  - This means someone on staff contributing code that the community as a whole values
- The utility of an employee who interfaces with open source is therefore measured by their credibility in the community.
- Therefore, the stature of a manager in Open Source is measured by the community credibility of all their employees

So where's the Quid Pro Quo?

- The ability to manage Open Source developers Successfully is a very sought after skill
- It's also a very easily recognised skill because your employees will be visible in the community
- Greatest Managerial skill is using the credibility and visibility of your employees to move the project in the direction you want
  - Requires a slight change in management culture in Japan
  - So you have to convince your manager of this
  - And they have to convince their managers …
- So this is a collaboration with your manager to make a slight change to the Company culture

Language Barriers can Be Overcome

- Unfortunately, only the hard way
  - Developers really have to learn english
- Written English (email) is easier than spoken
  - Don't have to speak proficiently, just write it
  - Email isn't instantaneous, so get time to think and formulate replies
    - i.e. don't have to reply immediately; can consider response
- English used in email exchanges is much simpler than formal written english
  - Mistakes are tolerated (even expected)
  - Don't need conversational modes and idioms

Cultural Interfacing

- Rule number one: Don't send your first post to the Linux Kernel Mailing List
  - It is famous for having the least signal to noise (i.e. flame war) ratio of any list dedicated to Linux
  - It also has the biggest contingent of non contributing criticisers
  - Therefore find a subject specific lower traffic list
    - And if there isn't one ask yourself if one should be created
    - David Miller (controller of vger) is sympathetic to new mailing list requests
- Stick to being technical
  - There are no real cultural pitfalls in the exchange of technical information.
  - If someone else wants to flame and insult, don't engage.

Create your own Cultural Interface

- The key to being comfortable with interactions on the mailing list is not to let someone else control them
  - Therefore, don't join existing flame wars if you're not comfortable doing so
  - Start your own new thread pointing out the technical issues
- What you're trying to do is create a culture for interaction where you're comfortable
- i.e. make them interact in your culture rather than you interacting in theirs.

To Repeat

- To take full advantage of Open Source, you need to both increase revenue and reduce costs
- Thus you need to build a contributor community with your competition
  - This is partly why Foundations (LF, Gnome Foundation …) proliferate in Open Source
  - They form neutral environments for natural enemies to co-operate
- If your company does't participate in Shared Innovation, It's missing out on revenue increasing opportunities

Conclusions

- There are two excellent business reasons for participating in Open Source
  - **Cost Reduction** by releasing code
  - **Revenue increase** by sharing innovations
- Developers who understand both the business processes behind using Open Source as well as the Development models are very valuable
- The only person who can Influence your Company towards better open source participation is **YOU**
- So the true answer to the question “Is the Future Open Source” is “Yes, provided you help”.

Thank you for listening

Questions?


Novell®