Understanding Enterprise Java Open Source

Rod Johnson

CEO, Interface21

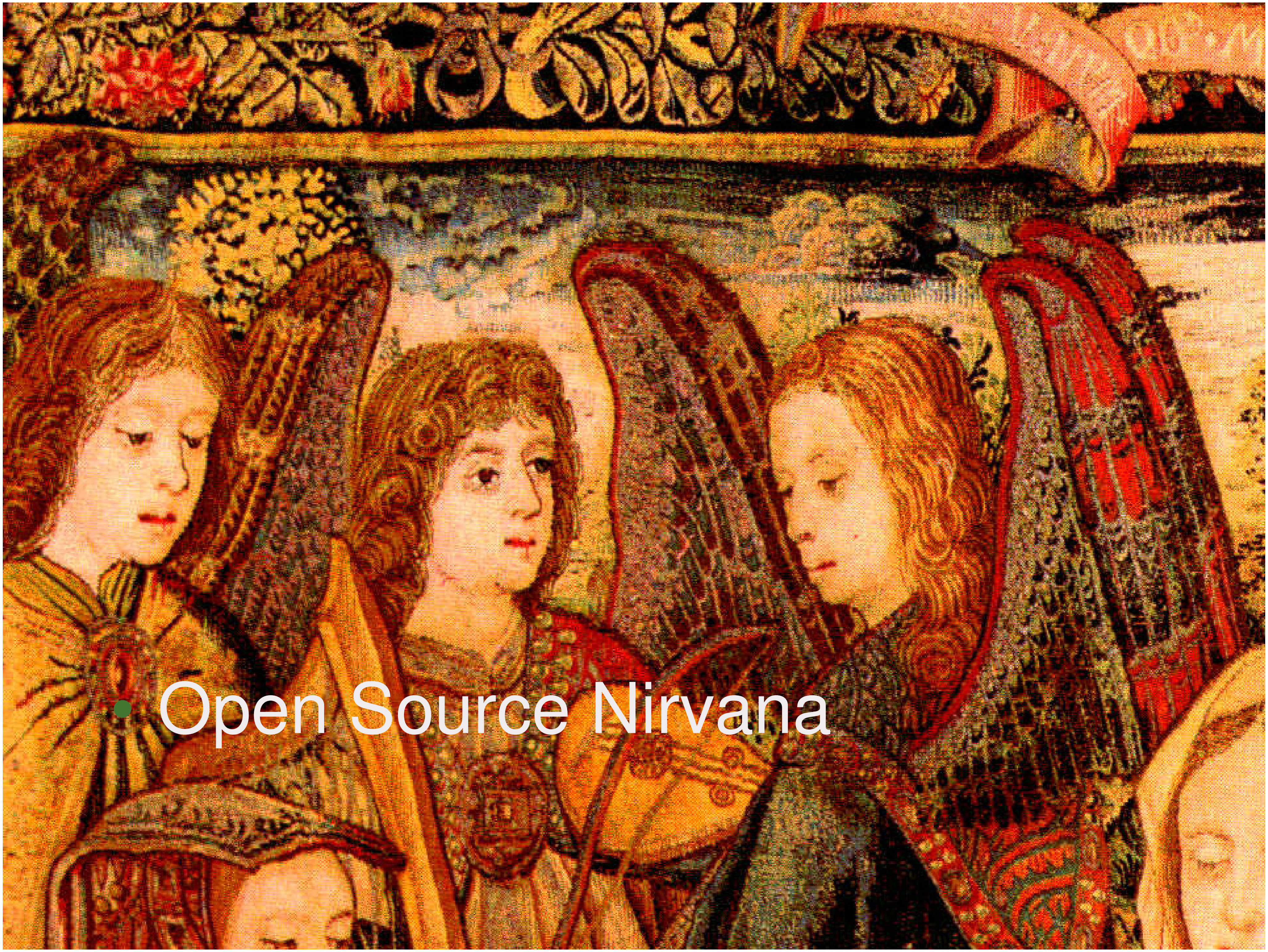
Spring

Open source is here to stay

- Who does *not* use Java open source solutions?
- Open source has unstoppable momentum in most areas
- Will gradually spread

The question is not whether open source continues to grow, but how we interact with open source

Two Visions of the Future



- Open Source Nirvana

Open Source Nirvana

- **Enterprise software is free**
- Enterprise software is reliable
- It’s easy to learn because of high quality documentation is available
- Software becomes easier and easier to use
- Errors never occur in production
- Innovation continues to flow

Open source means free software infrastructure for the enterprise

- The Horror of the Open Source Zombies

Open Source Hell

- Open source industry does not generate enough revenue and profit to progress beyond being funded by VCs and hobbyists to customers
- Software degrades in quality
- Lack of further investment results in lack of competition and hence lack of innovation

Open source ends up destroying value

How could this happen?

- The JBoss business unit at Red Hat does not grow beyond $70m
  - Red Hat disinvests in it
  - Morale collapses
    - More key JBoss staff leave
  - Leadership of projects fail
  - Bugs are not fixed, quality falls
- Geronimo development stalls because IBM ceases to invest in open source
- Perception that Java server software should be free causes BEA to end of life WebLogic and focus on SOA software
- Only WebSphere remains as a fully supported alternative to low quality open source

The zombies take over

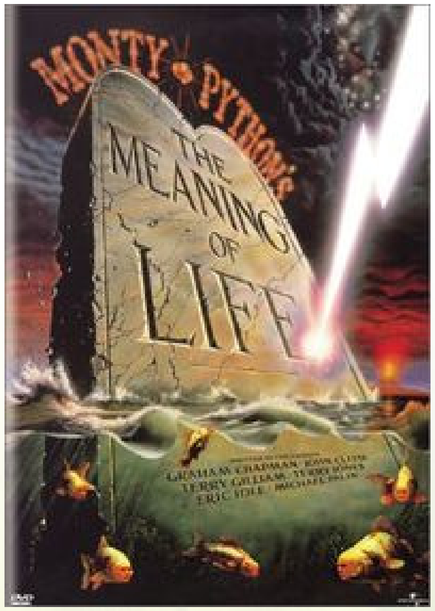
- Projects gradually fall apart
- No energizing force besides personal interest and hobbies
- Little innovation in open source due to lack of competition
- Little motivation for developers to solve hard or unpleasant problems
- Documentation falls by the wayside

A few survivors

- Apache Commons
  - Indestructible because not related to economics
  - …at least as long as corporations keep funding Apache Foundation
- Log4j
  - Simple, doesn't require significant ongoing investment

…well it would probably be good for .NET

How Did we Get Here? - The Ages of Enterprise Java Open Source


MONTY PYTHON'S
THE MEANING OF LIFE
GRAHAM CHAPMAN · JOHN CLEESE
TERRY GILLIAM · TERRY JONES
ERIC IDLE · MICHAEL PALIN

Part I: The Miracle of Birth

In the beginning it was small…and simple

- Log4j
- Struts

- Simple, reliable
- Open source sustained by hobbyists or individuals raising their own value
  – Could be well worth investing a few man months work to build a reputation

Useful, but not exciting: Not yet likely to change the world

Part II: Growth and Learning

Spring

JBoss, Hibernate, Jakarta, Spring

- JBoss
  - Ambitious project, if only commoditizing what was already available in closed source
- Tomcat starts to be used to run a proportion of enterprise applications
- Spring, Hibernate emerge as preferred programming model

Enterprise Java open source becomes widespread
We see the start of innovation in open source
Winners emerge from the thousands of projects

Part III: The Teenage Years

The Teenage Years

- All about getting attention
- About breaking through old barriers and assumptions
- Everyone needs to go through this phase

- …but it can be painful

The Teenage Years

- The early teens

```
From: "marc fleury" <marc.fleury@jb...>
 |To: "Trawick, James" <James_Trawick@tv...>;
 |"Jboss-Development@Lists. Sourceforge. Net"
 |<jboss-development@li...>; "Jboss-User@Lists.
  Sourceforge.
 |Net" <jboss-user@li...>
 |Sent: Thursday, March 14, 2002 10:22 PM
 |Subject: RE: [JBoss-dev] RE: [JBoss-user] JBOSS
  3.x FINAL

 |> mr trawick james,
 |>
 |> suck my dick
 |>
 |> marcf
```

The Teenage Years (2)

- The mid teens

- The late teens

Part IV: Settling into a Job

After the partying…Time to find a job

- No longer a transitional phase
- About the next 5, 10, 20 years…
- Need to find a way of interacting with enterprise class customers that is comfortable and natural for them

Long term sustainability is the real question

Let's be clear about the potential problem

- **Right now, VCs, hobbyists and large corporations fund the majority of open source development**
  - All suffer from the buffalo problem
  - All may prove fickle
- For an industry to succeed, it must be funded by *customers*

Remember, at this point, open source *must succeed* for users and developers alike

Let’s understand the model behind open source *in the enterprise*

Misconceptions about enterprise open source

1. Open source is about commoditization
2. Open source is magically produced by a community
3. Open source developers code for fun
4. Open source software is free

Myth #1: *Open source is about commodization*

- Open source products are inferior versions of closed source software
  - Maybe a few years late, once the market has become uninteresting

- May have been true once
- Is no longer the case

- Was *never* the case with Spring, AspectJ, Tapestry and many other open source products

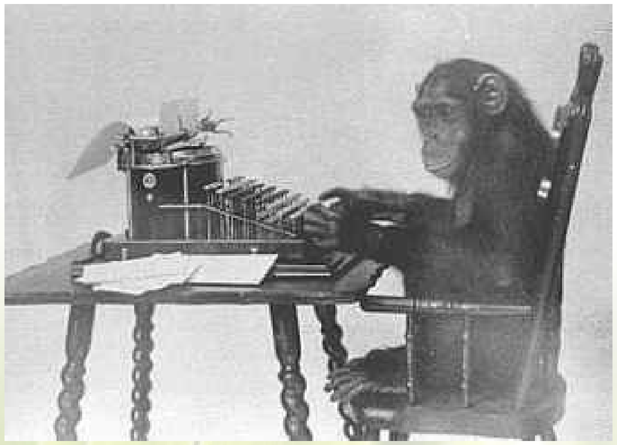
Case study: Spring Framework

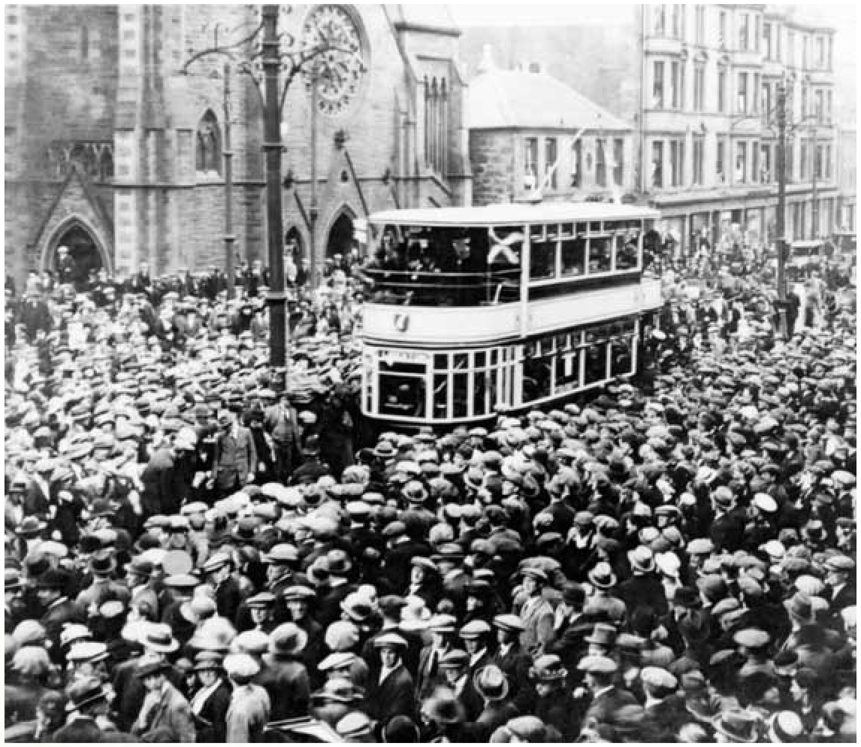
- Spring created a new product category
- Brought Dependency Injection and AOP for enterprise services to the mainstream
- Spring created a new style of component model that could run in any environment

Aimed to be the best in everything it does,
compared to alternatives, whether
open or closed source

Myth #2: *Community magically generates open source software*

- All we need is lots of developers
- With all those fingers on all those keyboards, great software just happens





Bottom up or top down?

- Reality is that projects need leadership and vision
- Contributions and patches from a broader community are important
  - But developing enterprise products is a long-term proposition

Bottom up or top down? (2)

- Essentials
  - Organisation
  - Dedication
  - Endurance
- Much of the work is *not* exciting
  - Support and maintenance
  - Documentation

Case Study: Linux

- *Dispelling the perception that Linux is cobbled together by a large cadre of lone hackers working in isolation, the individual in charge of managing the Linux kernel said that* ***most Linux improvements now come from corporations****.*

  "People's stereotype [of the typical Linux developer] is of a male computer geek working in his basement writing code in his spare time, purely for the love of his craft. Such people were a significant force up until about five years ago," said Andrew Morton, whose role is maintaining the Linux kernel in its stable form.

  Morton said contributions from such enthusiasts, "is waning." Instead, ***most code is generated by programmers punching the corporate time clock****.*

  About 1,000 developers contribute changes to Linux on a regular basis, Morton said. Of those 1,000 developers, about 100 are paid to work on Linux by their employers. And those 100 have contributed about 37,000 of the last 38,000 changes made to the operating system.

- http://www.gcn.com/online/vol1_no1/26641-1.html

Not everyone wants to play

- *"The myth of open source software is the aura of freedom that surrounds it. Download the source code and play with it if you wish. And best of all, you won't have to pay the freight.*

  Although that's the public image of open source, the reason why open source software is growing popular within enterprises has nothing to do with open source itself. Few enterprises care about whether they can monkey around with source code because the relative minority that still have active internal software development staffs have more important things to do."

Tony Baer, Datamonitor Computerwire

Most enterprise open source projects come from *companies*, not individuals

Myth #3: *Open source developers work for fun*

- Developers want to code for fun
- They do this because their day jobs are boring

Enterprise software development is not a part time job

- Around 1 million lines of codes in the Spring Portfolio

- Integrations with a wide range of products that require testing
  - All major
    - Application servers
    - Database drivers
    - O/R mapping tools
    - …

Reality

- Successful enterprise developers typically have lives
- Developers have jobs, families
  - Even *hobbies*

Well, at least they have *families*

The best people can get interesting jobs

- Building enterprise infrastructure for mission critical applications requires great people
  - Those people can pick and choose their jobs
  - Those people *will* have an impressive track record apart from open source

No one commits *indefinitely* for satisfaction

- Even developers who are willing to contribute to open source for fun or resume enhancement right now, cannot be relied on to do so in the long term
- They are less likely to contribute as prominent individuals cash out of open source
  - People have made large fortunes out of open source
  - If it's producing massive rewards for some people, why should developers sacrifice salary or time?

Myth #4: *Open source software is free*

- No enterprise software is free
  - Far too complex
  - Far too much work to develop and maintain
  - Needs to be maintained for many years
  - Needs to have a migration path
  - …

Open source and TCO

- License cost is just one element of TCO
- Many other factors
  - Performance
    - Bang per box
  - Training
  - Maintenance
  - Support
  - Availability of skills

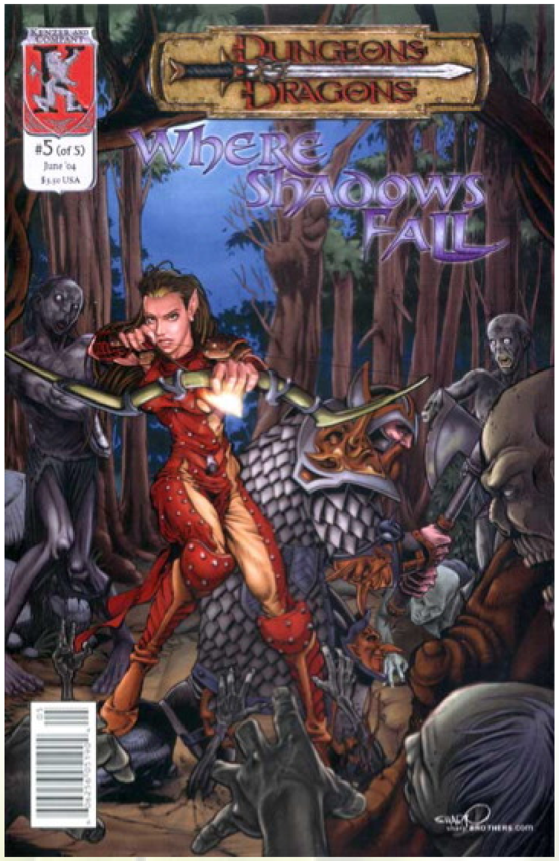
Don’t sell it short…

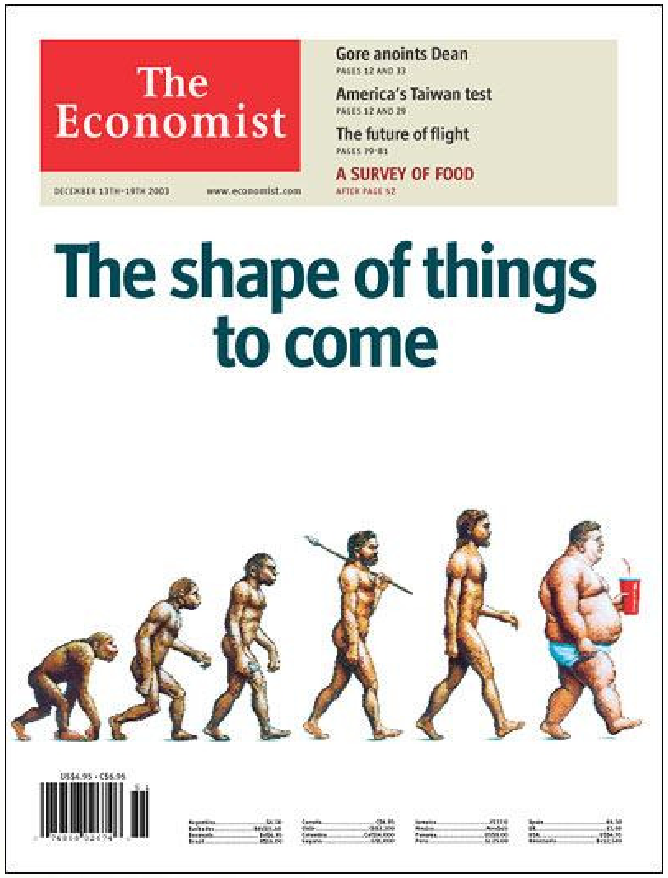
- The appeal of open source is not that it’s free
- It depends on quality and fitness for purpose

Market Forces still Apply

- *The Mythical Man Month* still applies
  - Do not get high quality software by throwing people at the problem
- Viable projects need viable businesses behind them
- Open and closed source enterprise software are not so different
  - Both are usually commercial

The Shape of Things to Come





The Key: Rewarding intellectual property creation

- Open source does not suspend the rules of capitalism
  - Bubbles come and go, but market forces soon reassert themselves
- Viable projects rest on a viable economic model
- Development is expensive
- **The key essential of the economic model is rewarding IP creation**

What if IP creation is not rewarded?

- Leads to the horror of the open source zombies
- Projects become leaderless, lack strategic vision
- Only support available is of poor quality
  - Based on hacks that degrade product quality

Why would IP creation *not* be rewarded?

- Ability to separate monetization from IP creation is both a blessing and a curse
- Pros
  - Keeps vendors honest: Customers can choose other suppliers
- Cons
  - No *direct* return on creation of IP as produced by traditional product licenses
    - Lack of effective return can cause long-term problems

Why would IP creation *not* be rewarded?

- #1: Self service
  - Customers who feel that they should support themselves make two errors
    - They get inferior support and waste money
    - They are hurting the long-term potential of open source

Why would IP creation *not* be rewarded? (2)

- #2: Exploitation
  - Companies that bundle open source into projects without giving anything back
  - Sometimes even compete with the creators of the IP they consume
  - MyEclipse / Spring IDE
- Copyleft licensing can help with this problem, but has other disadvantages

Why would IP creation *not* be rewarded? (3)

- #3: Aggregators, who “support” multiple projects but don’t lead or sustain any
  - Do not contribute to IP creation
    - Degrade projects in the long term
  - Can only offer poor quality support
    - Not involved with leadership or roadmap of any product
    - Destroy user confidence in the long term
- Short-sighted aggregators make no contribution to forums, development, documentation, and conferences of most open source products…
- Good aggregators do form relationships with companies that create IP
  - Valid broker model

Why the Aggregation Model doesn't work

- Based on flawed assumptions:
  - That open source is generated by hobbyists who don't much care about their income
  - The "it's all too complex" myth: that the problem is one of integration rather than IP creation
  - The "it's not so hard assumption"
    - Enterprise open source products are complex
    - Serious work on them requires the core team
    - It's not realistic to provide support for a complex product you don't contribute heavily to

Aggregation may work for commodity open source, but not innovative open source

What can you do to help long term viability?

- Interactions with open source fall into *consuming* or *sustaining* activities
- *Consuming* activities do not advance open source, and may even weaken it
  - Consuming activities do not contribute to creation of IP
  - Consuming activities do not contribute to thought leadership
- *Sustaining* activities advance open source and contribute to sustaining it at high quality

What can you do?

- Analyze your interactions with open source in terms of consuming and sustaining activities
- **Contribute**
  - Contribute documentation
  - Contribute patches, maybe advance to committer status
    - If you do this, be prepared for a long term commitment
      - Do not think you can dump some code and move on to another interest
  - Help strengthen the community through helping other users in forums
  - Report issues
  - Evangelize – The more users use a project, the stronger it is
    - Help promote your favoured products
- **Pay**
  - Seek out companies or individuals who are driving your preferred products forward and purchase support or training

Actually, Marc Fleury had a point: “customers pay”

- *Contribute* or *pay*, but don’t think you can consume forever without giving back
- It’s not hard
  - There are solid companies you can engage with
- Remember:
  - Open source is increasingly the only game in town
  - If it doesn’t work out, everyone loses

If we get it right, everyone wins

- Open source *does* lower total cost
  - Low cost of distribution lowers cost of sales
  - Closed source vendors enjoyed excessive profit margins prior to open source challenges
- Freedom of information and access to source helps during development
- Ease of collaboration benefits everyone
  - Can add to open source software with incremental cost

- Sustaining relationships can deliver outstanding value to customers
  - High quality support
  - Expertise around the product, built up through engagement with a large user and customer community

The Open Source Business Model has worked in other Industries

- Red Hat and other Linux vendors

- Still maturing in enterprise Java

- Dangerous disconnect between size and revenue of open source product vendors and the importance of their products to customers

Take aways

- Open source is here to stay in enterprise Java
  - The question is how to make it work best for customers and vendors
- Understand the model
  - It’s not about hackers
- There is no such thing as a free lunch
  - Nothing is free, including open source software

Interacting with Open Source

- Understand the difference between *consuming* and *sustaining* interactions with open source
  - If you want to be enjoying high quality enterprise software in 5-10 years, choose sustaining interactions

Q & A