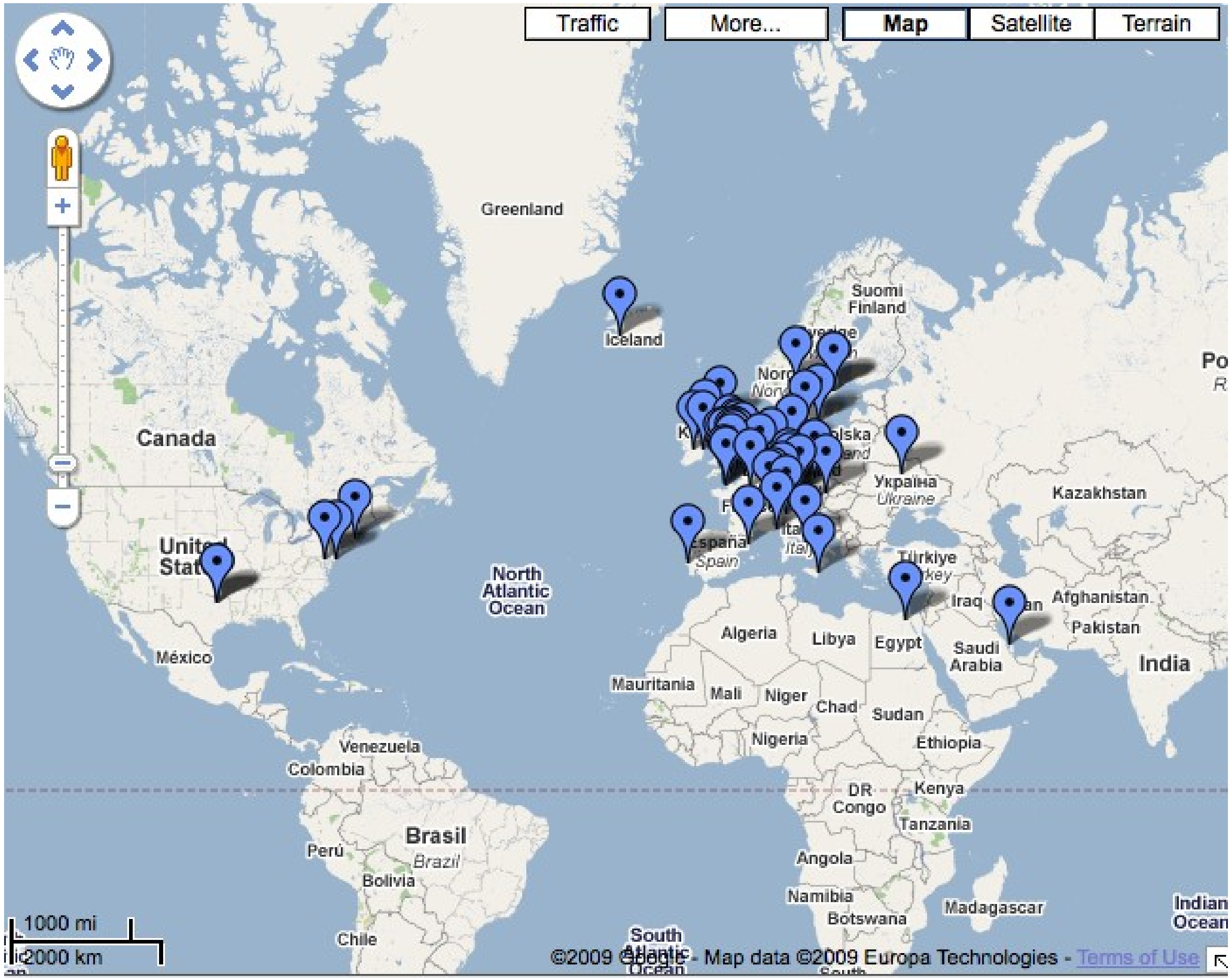
Adopting Open Source in the Enterprise

ApacheCon Europe 2009

Track: Business
Level: Overview

Adrian Trenaman
Distinguished Consultant
http://trenaman.blogspot.com

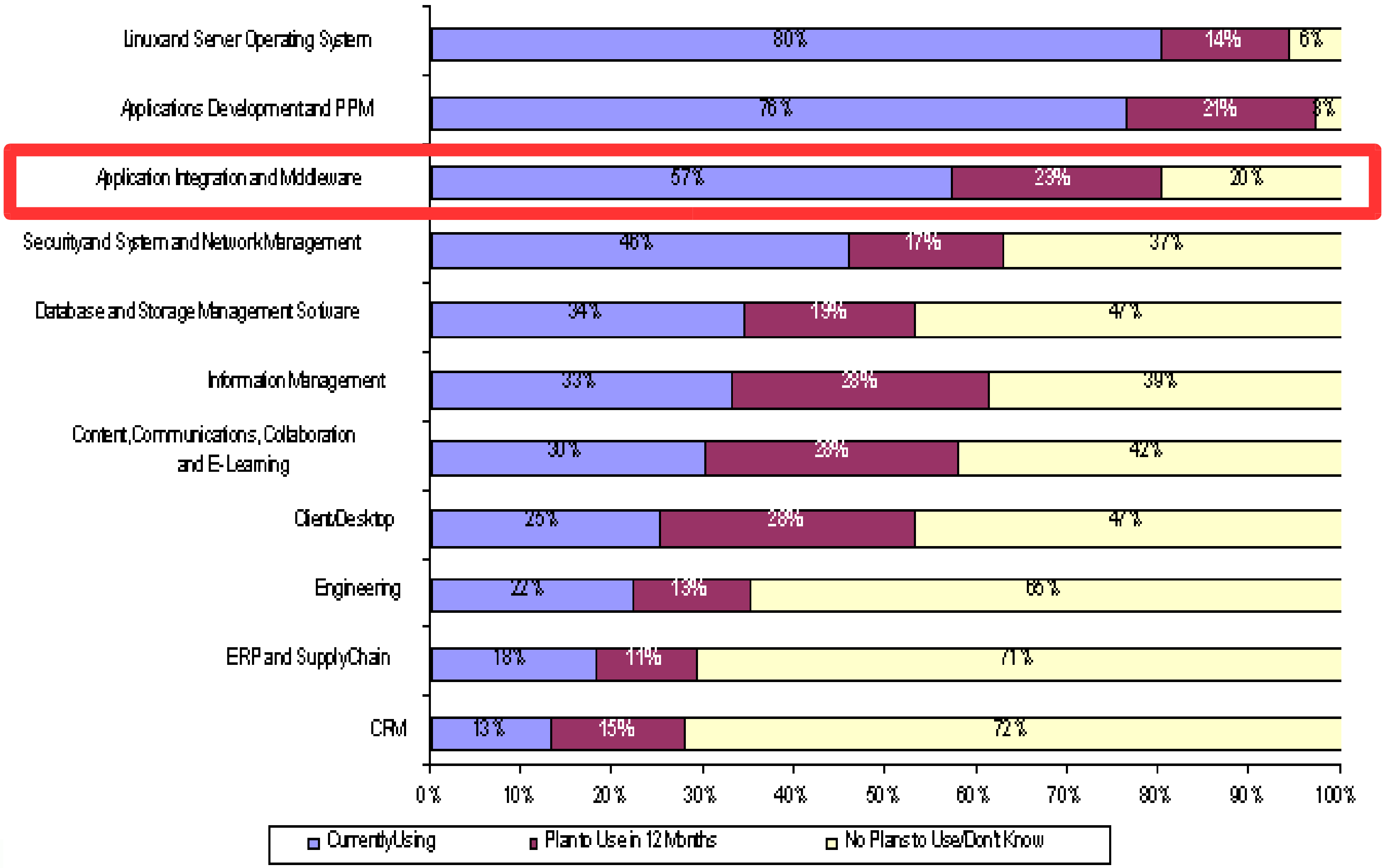
Ade's Consultancy Map



Agenda

- Open Source is a very big word...
  - ... so is 'Enterprise'
- Let's focus on adoption of Apache-based Open Source for Middleware and Integration.
  - Focus on in ISVs, SIs, and large enterprises.
  - Focus on ServiceMix, ActiveMQ, CXF, Camel
- Discussion:
  - Why adopt open source?
  - Who is driving the adoption?
  - How is open source being adopted? What works? *What doesn't*? The role of the OS vendor.
  - What are the implications?

Some context...



Source: Gartner

Aside...

- “Opinion is the lowest form of fact”
  - And yet, strangely, we value respected opinion greater that facts themselves.
  - The opinions and observations in this presentation are based on years of experience in open source 'from the trenches'
    - Thanks to Wolfgang Schulze, Roland Tritsch, Rich Bonneau, Rich Newcomb, Martin Murphy, Andreas Gies, Ashwin Karpe, ... and others at Progress.
- “Flattery gets you nowhere”
  - You are a fabulously intelligent audience...
  - ... probably in the top 10% of coders / hackers / architects / business-people in the world!
  - Remember: it is a mistake to believe everyone else will be as passionate / excellent / brilliant / committed as you.

Aside (cont')

commons | changing | challenging | rewarding

“Proof by analogy is fraud” ... and yet, analogy is very useful in helping us discuss and flesh out ideas.

Open Source Code ≈ Mountains

Open Source Vendor ≈ Mountain Guide

Why are enterprises adopting open source?

Motivations for adopting open source

- Price *is* a deciding factor.
  - ... price is not *the* deciding factor
  - Any investment [time or money] requires an investigation of risk and ROI.
  - Price (or rather, price scalability) is *very* important for SIs, ISVs, and enterprises with large-scale or geography-wide deployment.
    - Some closed-source vendors *haven't figured this out*.
- Agility
  - Faster detection and resolution of issues cuts development time and increases time-to-market
- Control
  - Avoid vendor lock-in (only applies to permissive licenses)

Motivations for adopting open source (cont')

- *Quality*.
  - Sometimes the open source alternative is simply *better*.
  - Better = wider adoption, easier to use, multi-platform, standards-based.

Who is driving adoption?

Adoption

- Top-down
  - Sabre: CTO initiative to adopt standards-based, open-source container
    - Adopted ServiceMix / ActiveMQ in their Supplier Side Gateway project.
    - 1.5m transactions per day; 14 months zero down-time.
  - US Federal Aviation Authority – `http://www.swim.gov`
    - System Wide Information Management
    - Towards NGATS (Next Generation Air Transportation System)
- Bottom-up
  - Retail-pharmacy: application manager sketched solution with gregorgrams, and implemented using EIPs in ServiceMix

Driving adoption top-down from the CT(I)O or program level

- Make a strategic plan around open source
  - Vision. Goals. Milestones. Resources.
- Involve technology leaders in your organization.
  - You won't succeed without their buy-in.
- Create a centre of competence around chosen open-source technologies
  - We'll discuss this in more detail later on.
- Execute the plan.
  - “The plan rarely survives contact with the enemy”
  - You've opened the door: make sure there's someone to walk through it.

Aside: Open Source Maturity Model (TM)

- OSMM from `http://www.navicasoft.com` provides a framework to assess 'maturity' of an open source product.
  - Maturity: a number based on weighted assessment of different areas
    - Functionality
    - Training
    - Documentation
    - Support
    - Integration
  - Threshold of acceptance is then based on the your organization
    - Innovator, *or*
    - Pragmatist

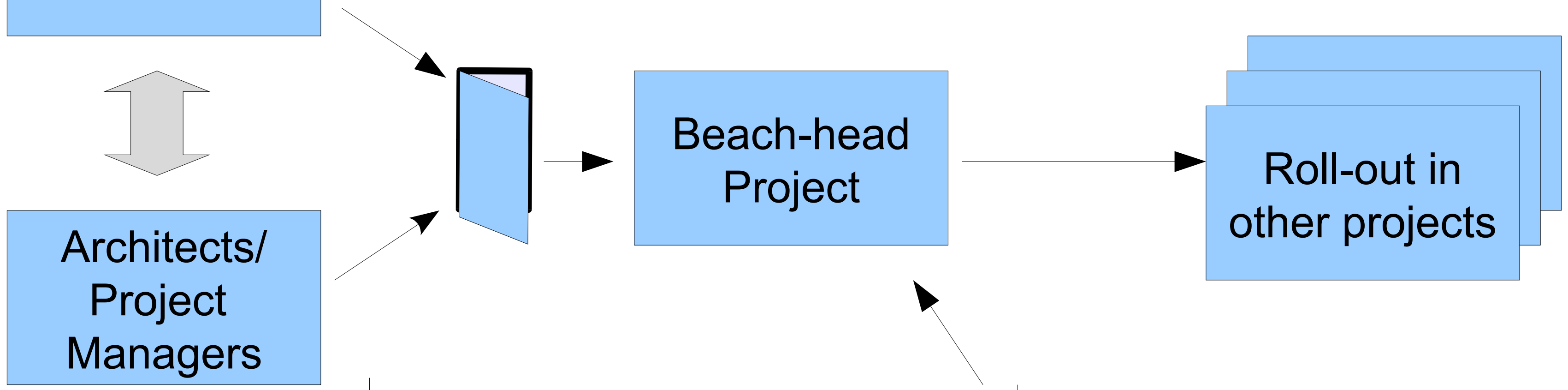
Bottom-up adoption

- Driven at a project level by architects and senior engineers
  - Drivers: code quality, standards, ease-of-access, cost, ...
- Sometimes skunk-works projects bubble up to the service.
  - e.g. replacement for JEE stack at a major financial services company.
  - e.g. Integration backbone for another major FS company.
- Tends to emerge in organizations who pride themselves in their engineering expertise.
  - "Hang on a minute: we can do this *better/cheaper/faster* with open-source"

Open Stealth

- Avoid big-bang, boil-the-ocean approaches
  - Many will resist.
  - Particularly, and ironically, your IT department, the bastion of conservatism.
- Select a 'beach-head' project.
  - With clear, strategic value and potential for 'poster-child' success
  - Make it successful...
    - ... and use it as a platform for organizational learning.
  - Successful innovation attracts followers (think of Apple!)
    - ... build a constituency; gather support.
- Plan wider roll-out.

Adoption in the enterprise



"Open the door..."
Technology recommendations
Vendor selection
Evangelism
Centre of competence

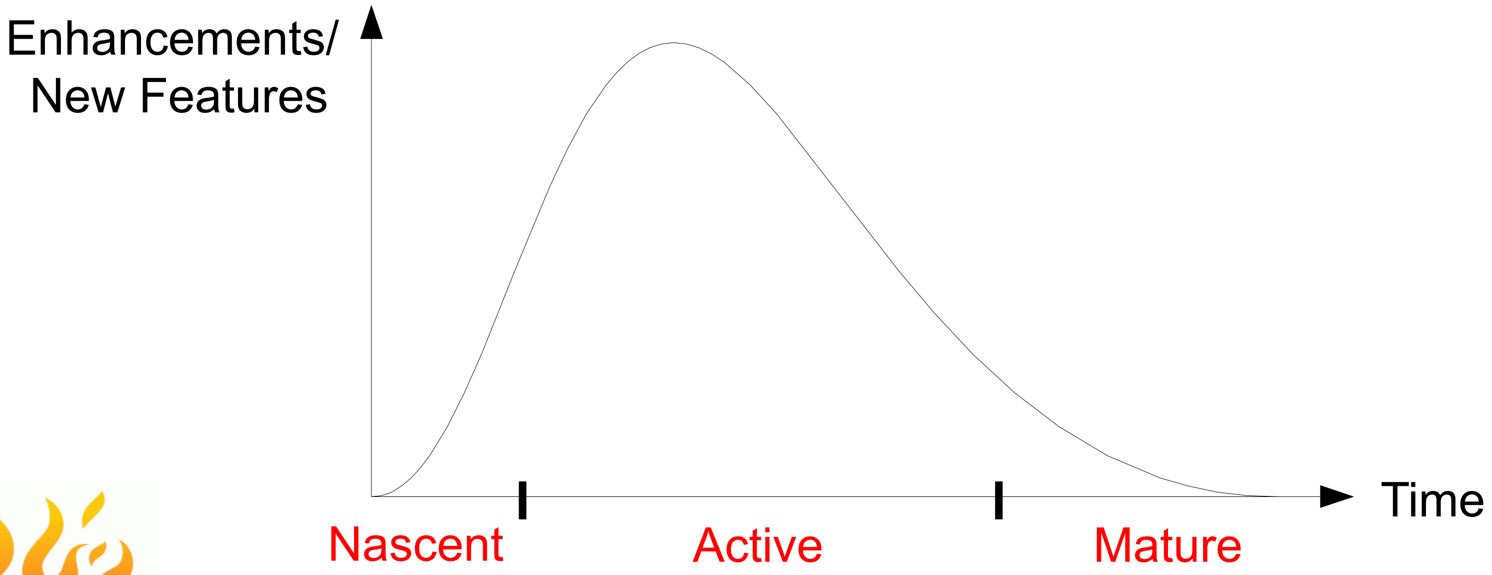
"...Walk through it"
Business case
Requirements
Compelling event

"... Learn what works"
Use your best people
Get outside help if necessary
Prepare to learn

Community Involvement

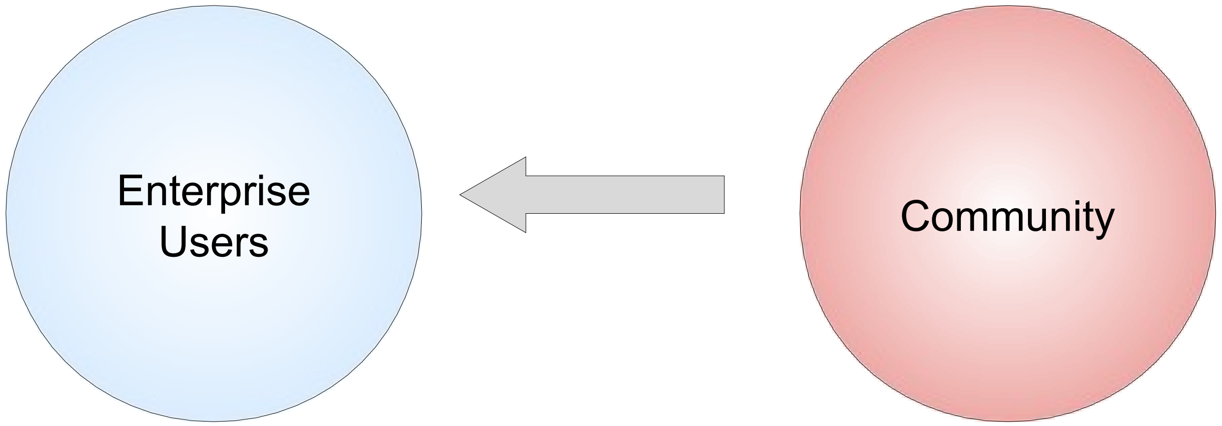
Download 'n' Go!

- Can you just freeload?
- How engaged with the community/source must you be?
  - Depends on how mature the source is. Here's one way of looking at it: projects are either *nascent*, *active* or *mature*.
  - Where there is *innovation*, there will be *issues*.



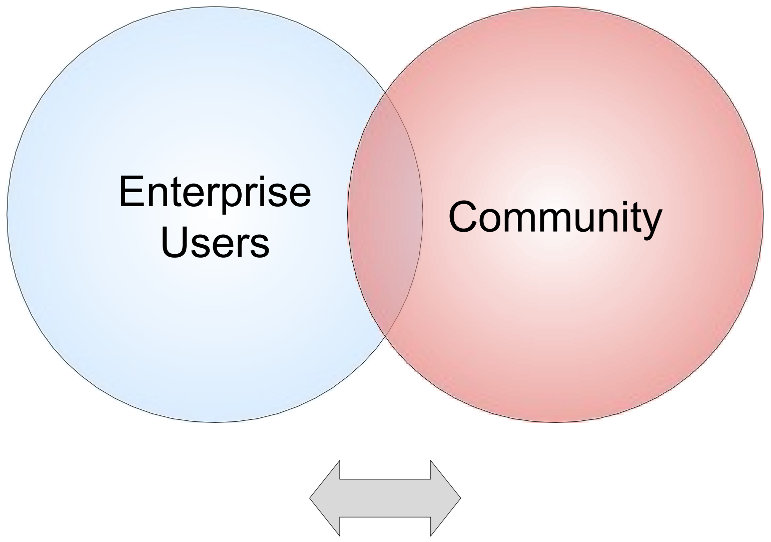
Model 1: No interaction with community

- Treat the *project* as a *product*
  - No need to download the source, just the binaries please!
  - Suited to *mature* open source projects; e.g. Apache Server, PostgreSQL, Open Office, ...
  - Suited to 'product' rather than 'framework' style projects.
    - Product: finished article; does what is says on the tin.
    - Framework: tools or building blocks with which to build solutions.



Model 2: Direct interaction with community

- Hmmm: this project is great, but needs more work... we're happy to help!
  - Ideal for nascent projects, and early adopters
  - Engineers can become committers, and drive adoption.
  - But does it *scale*?
    - Must all engineers have intimate knowledge of the code?
    - What if I use *n* open source projects?
    - ... ?



The dark side of the source

- Good fences make good neighbors
  - Clear boundaries tend to be a good thing!
- Opening up the code can “increase the problem space”
  - Abstractions make things easier; detail makes things more complex.
  - 7 ± 2 concepts at a time, please.
- Cross fertilization of code can be mind boggling.
  - “I once found myself debugging jetty continuations...”
- Not all developers have time for (or are up to) the challenge.
  - This is not a criticism; just a fact of life.

Enterprise vs. Community: culture clash?

- "I value the finished product."
  - => "I can't stand incomplete product."
- "I'm focussed on my work"
  - => "I do not want to help you with yours"
- "I want my team to be actively contributing to achieving its goals"
  - => "I do not want them 'distracted' by community work"
- "I should be able to use this without knowing the nuts and bolts"
  - => "You can use it best by understanding the nuts and bolts"

The successful project team

- In any project team, there are:
  - Achievers (top 20%): motivated, talented, engaged
  - Adequates (top 75%): need direction, effective when given a cookie cutter.
  - Wasters (the rest): Useless. Move them on if you can. Contain them if you can't.
- When it comes to projects adopting open-source, *attitude* is the most important thing.
  - Open-Source Positive.
    - Focus on solutions through the source, not problems due to the source.
  - Hire for attitude and ability, train for skill.
    - Consider training as necessary but not sufficient.
    - Need training + practice + coaching.

Scaling open-source knowledge

The problem is *not* the achievers.
They will always adopt the 'right attitude'.

The problem is the *adequates*.
Or rather, how to make/keep them effective.

The Law of Comparative Advantage

- Entities should specialize in areas where they have competitive advantage.
- E.g.: I am very good at DIY. On my weekends, should I:
  - Put in a patio? *or*
  - Provide $$$ consultancy services?
- I may have *absolute advantage*, however, LoCA says I should specialize.
  - I win, as does the landscaper.

David Riccardo (source Wikipedia)

Scaling open-source knowledge (cont')

So, how do we apply LoCA to teams where only a few players have absolute advantage in open source?

In a team of, say, ten...

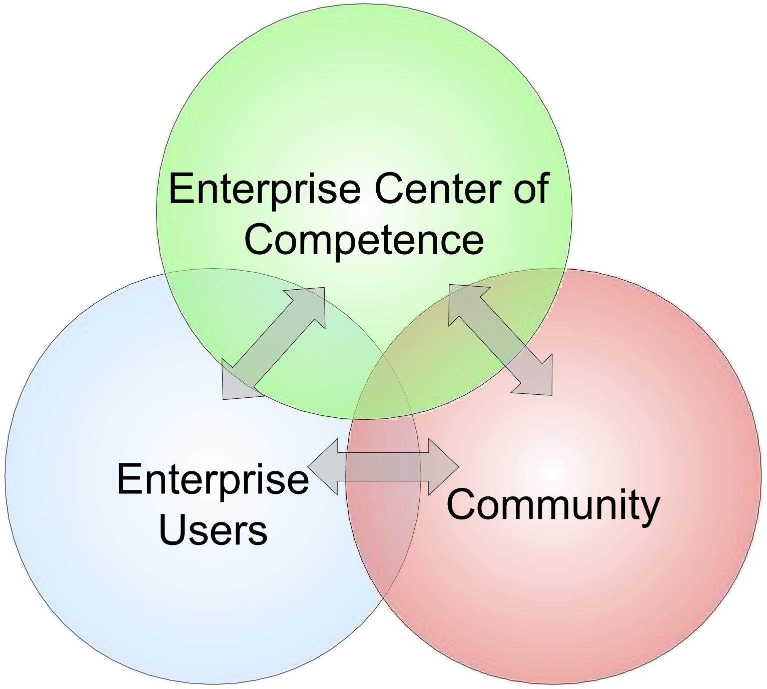
- In the 80-20 model, two things can happen.
  - 'Hero' model: two guys do all the work, eight guys watch by in amazement, shock and awe.
    - The eight step back and take on peripheral tasks.
    - Very like the 'Mythical Man Month' surgical-team model.
      - Except in *that* model, everyone had a assertive, positive role.
    - Drawbacks: high-dependency, fatigue, fracture, prima-donnas.
  - 'Lever' model: two guys work out the architecture, the patterns, the archetypes.
    - Their role is to lead by example.
    - Their focus: remove blocks for the eight.
    - Drawbacks: need the right kind of hero.

LoCA in action – lever model

- We know the achievers have *absolute* advantage.
  - But they should focus on architecture, patterns, technology expertise, mentoring.
  - The challenge is to stop them doing everything and get them to act as *enablers* rather than *doers*.
- The adequates have comparative advantage on some aspects.
  - Local domain knowledge. Implementation (based on patterns). Testing. Documentation.
  - The challenge is to make sure that blocks are removed.

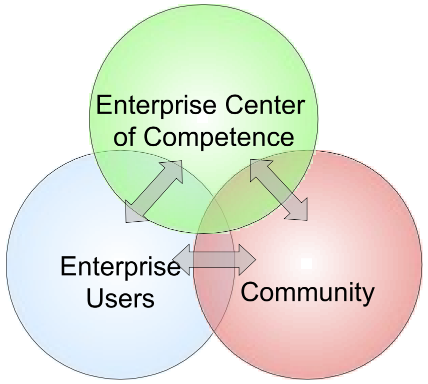
... Model 3: Centre of competence

- Larger enterprises can use a CoC to leverage *specialization*.
  - Create a dedicated technology/architecture group to own the relationship with the community.
  - Let rest of organization become 'users', focussed on the core business.
    - Projects can pull-in skills and resources from the CoC.



Role of an open source centre of competence

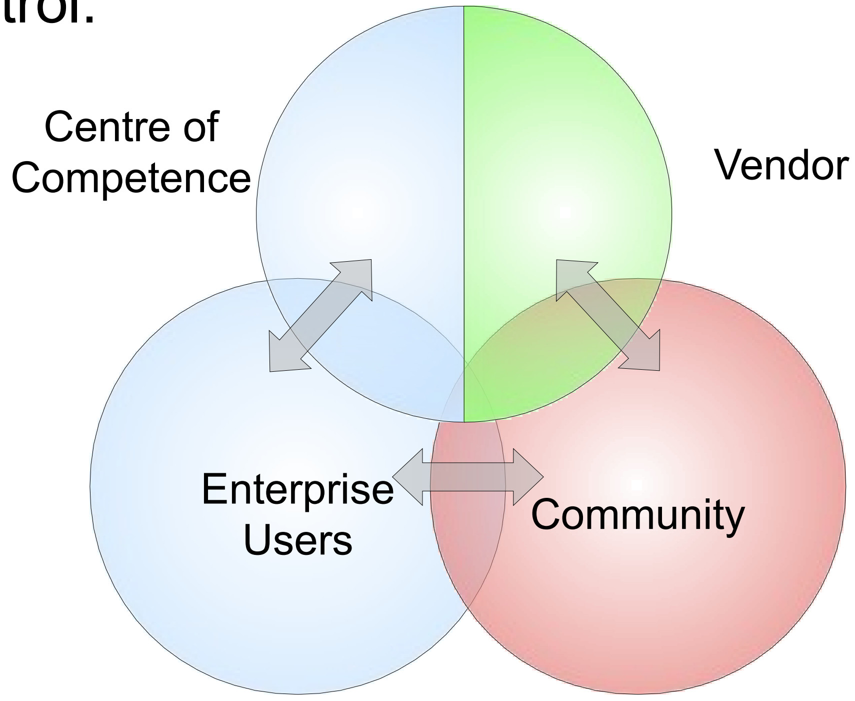
- Provide regular stable releases of project(s)
  - Potentially with in-house fixes
  - Track issues and merge fixes to community
- Maintain a 'forge'
  - For internal releases and internal projects / plugins
  - SCM, Issue-management, Wiki, Forums, IRC, Maven ...
- Support developers
  - Training, documentation, how-to, use-cases, patterns ...
- Enforce licensing compliance
- Evangelize open source technology & philosophy



Enterprises will support themselves unless the cost associated with that support exceeds the cost of outsourcing it.

Out-sourcing the centre of competence

- Vendors can play a part as an *out-sourced* centre of competence.
  - “We can resolve your issues faster and cheaper than you can”
  - Stay agile, reduce cost.
- Prefer 'part-sourced' rather than 'out-sourced'
  - Stay in control.



Getting it *wrong*.

- Remember the project phases?
  - Nascent, active, mature
- The worst mistake is to misjudge an open-source project.
  - “Hmmm. I'll use a commodity open source framework...
  - ... with adequate developers ...
  - ... and I'll save massive amount of money!”
- If the project is active, there *will* be issues
  - .. which will require a team of committed, engaged, quality developers.
  - ... and, perhaps, a culture change.
    - Pro-active, code-hunting, engaging.

Aside: Progress OSCoC

- Team of consultants dedicated to Open Source
  - Contributors, committers
- Goals:
  - Make users successful
  - Drive adoption (writing, blogging, contributing, forums, initiatives, ...)
  - “Scale out” skills throughout the larger PS organization

Back to the hills!

Is a trusted guide going to get you there quicker, safer and easier?

Advice: enterprises should simplify rules around licensing

Understand Open Source Licensing

- Some licenses such as GPL can be restrictive.
  - GPL: if you use this GPL software in your solution, *and then redistribute*, then your software must also be GPL.
  - Dual-licensers tend to use GPL: any competitor who attempts to improve on the code must release these improvements to the community for free!
- LGPL (Library/Lesser GPL): you can link LGPL software with your own commercial non-LGPL software, so long as it is not considered a "derivative work".
  - Definition of "derivative work" is ambiguous and untested.
- Apache License: simply provide an acknowledgement, disclaimer and copyright notice. Very Friendly!

Keep it simple, keep it safe.

- Example policy for a software vendor using open-source internally
  - Legal department vets all licenses used in products to ensure compliance with license T&C's

✓	*Apache (v1.1, v2.0), BSD, MIT, X, OpenSSL, OpenSSLeay*	
?	*CPL v1.0, EPL, LGPL, MPL*	*No: unless certain conditions are met*
✗	*GPL*	

What can we do to increase adoption?

From a community perspective...

- Make *heroes* out of technical writers.
  - Why are they less important than engineering committers?
  - Their input can drive adoption.
  - They can impact on the perception of risk
    - Use cases, patterns, ...
- Reduce source-code 'barrier to entry'.
  - Surely there must be a way to mark 'trails' in the code?
  - Automatically discover well-trod execution / browse paths.
- Make it easy for adopters to submit success stories
  - Templates? Gentle nudges on the forums?

From a user's perspective

- Contribute to the source!
  - Raise issues, even when you find workarounds.
  - “Poor usability is a bug” - raise issues when something annoys you.
  - Submit demonstrations
- If your project has been successful, tell the world!
  - And if it's not, don't grumble in silence. Tell the world!

Thing's *we're* doing beyond the Source

- Progress Knowledge Services
  - Major documentation drive, impacting Apache & FUSE materials.
    - Reference Material
    - User Guides
    - Deployment Guides
- Progress Professional Services
  - Phase 0 initiative: the first two hours after download.
    - Getting Started Screen-casts
    - Webinars
  - Usability on common use-cases.
  - Technology white-papers
  - Masterclass Webinars

Summary

- Adoption is driven from many areas: top-down and bottom up
  - Nothing builds success like success
  - Thing big, start small.
- OS project maturity plays a big part in how you adopt
  - Design your team to facilitate specialization
- Vendors play a part in reducing costs through specialization
  - Knowing the territory is key – the “mountain guide”
- Vendors play a part in 'rounding out' the project.
  - Documentation, ease-of-use, education, etc.