

Bambi Meets Godzilla: They Elope

Open Source Meets the Commercial World

Eric Allman
EuroBSDcon 2012
Warsaw, Poland
October 20, 2012

Where I Was When I Got The Email (Last Week)



My Background

- Long time open source developer (started ~1975)
 - INGRES RDBMS (early days)
 - syslog, -me (troff) macros, trek, other BSD utilities
 - sendmail
 - a few defunct drivers (punch cards! 9-track tape!)
 - the guy who got Berkeley to start using SCCS
- Jobs in academia, commercial, and research
- Started Sendmail, Inc. in 1998
 - One of the early Open Source “hybrid” companies
 - Survived the tech crash (but that’s another story)

Sendmail's Background

- Sendmail started as one of the first open source projects (as part of BSD), a “classic example”
- Like most Open Source of the era, it went through some growth spurts
 - Built to solve a single, local problem
 - Generalized due to community need
 - Got caught up in the Internet explosion
 - Remained community-supported, usually with the assistance of a small group of people (sendmail used the benevolent dictator model with trusted henchmen, same as Linux)

The Onset of Success Disaster

- At some point, scaling collapsed
 - I no longer had time to do coding due to support requirements
 - Some projects used the RTFM approach (i.e., “you’re on your own”), but that only works with dedicated audiences (and a FMTR)
 - Assertion: all successful large Open Source projects get outside money at some point
- I wanted to get time to do coding again, which meant ditching the day job
- So I started a company

Models for Monetization

- Start a foundation, get donations (e.g., Mozilla, Eclipse, Apache, FreeBSD)
- Find an angel who will shower you with money
 - Hard to do
- Sell yourself to someone with deep pockets
 - Note: they may not have your best interests in mind; may just want to shut you down
 - Leverage limited if you are the only asset
- Start your own company (e.g., Sendmail, Red Hat)

Open Source Needs Commercial Input

- Developers seldom are also the customers
 - Open Source's traditional base
 - Also true of most software-based research
- Developer-designed consumer software usually “unimpressive” to “outright bad”
 - Developers don't think like normal humans (or communicate well with them on software design)
- Examples of other benefits
 - “Soft” items such as user documentation
 - Front line support (unburden developers)

Commercial Markets for Open Source

- Who's going to pay for product?
 - Folks who just want it free? Good luck with that
 - Businesses? What size? They buy trust, not just code
 - Consumers? Fickle, need polished product
- Most customers won't care about open source
 - Think like a customer. What are they buying?
- Open source tends to commoditize a market
 - Brings down the unit price
 - Suppliers have to move up the food chain

Commercial Models for Open Source

- Completely free, sell something else
 - Support, services, documentation, stability, etc.
 - Limited economies of scale
- Free, sell bundles (distribution or appliance)
- Free basic technology, commercial non-open-source add-ons
 - Works best when you have a clean extension model or can “wrap” OSS in commercial software
 - Generally supersedes “sell something else”
- Technology grab (close the software base)

Starting a Company

- Starting a company is not about technology

- It is about:

- Finance (starting with Investors)



- Sales



- Marketing



- Support



- Services



- oh yeah, and some Engineering



- I didn't know what most of these functions did.
Now I do. Poor me.

Deep Tension Between Open Source & Commercial

- Open source is about building, sharing, flexibility
 - Make the world a better place (give back)
 - Solve an interesting problem
 - Personal development (and perhaps fame?)
- Commercial is about making money
 - Sales guys do not understand how to make money by giving the product away (“you’re a communist”)
 - Immense pressure toward feature creep to keep a revenue stream going (e.g., Quicken, iTunes)
 - If you miss payroll, you’re dead

Some Corporate Open Source Justifications

- Leverage other people's work, speed development, thus reduce engineering costs
 - Note that licenses can be show-stoppers here
- Can be useful as recruiting incentive
 - Sometimes unanticipated result: culture shift
- Disrupt the market
 - By giving it away you make it harder for competitors to make money
- Can make customers more comfortable
 - If you drop the product they aren't stuck

Corporate Culture

- Engineering driven or Sales/Marketing driven?
 - Almost no large company is engineering driven (Google comes the closest, and it is an anomaly)
 - Investors prefer S/M driven, and they run the board
- Purely Sales/Marketing driven leads to aberrations, but it is very hard to avoid this
 - Sales always wins in a fiscal crisis
 - A fiscal crisis always comes along sooner or later
 - Possible exception: when you are sitting on a ton of cash (e.g., Apple, Google)

A Note About Foundations

- Foundations insulate you from the day to day pressures of corporations
- Foundations *do not* prevent you from being pressured
- You might lose some of the good things (e.g., good marketing input)

Livelifecycles: Open Source, Research, Companies

- A brief (and woefully imprecise) comparison of the lifecycle of an Open Source Project, a Research Project, and a Company
 - Open (non-proprietary, non-military) research
- Note the similarities — and the differences

The Initial Inspiration

| | |
|------------------|---------------------------|
| Open Source | “Scratch an itch” |
| Research Project | Ask a question |
| Company | See a revenue opportunity |

Making It Possible

| | |
|------------------|--|
| Open Source | See if it's already been done (optional) Do an architectural design (optional) Choose language/tools Start writing code |
| Research Project | Research the literature Get a grant Line up grad students |
| Company | Write a business plan Line up investors Figure out corporate culture (optional) Hire a team |

Birthing the Baby

| | |
|------------------|--|
| Open Source | Do early (0.x) releases Start building community |
| Research Project | Start writing code/researching Start writing “teasers” |
| Company | Start building product Line up early customers Start trade shows |

Making it Real

| | |
|------------------|--|
| Open Source | Release 1.0 Address support problem Got docs? Oops.... |
| Research Project | Publish or Perish |
| Company | First release Scale out support & services |

Growing It

| | |
|------------------|---|
| Open Source | No community? Hang it up Write the O'Reilly book Avoid Second System effect (optional) Release 2 |
| Research Project | Thesis time Slaves Students graduate |
| Company | Second release Push to profitability First (second?) round of layoffs Second (3rd, 4th) investment round |

Next Steps

| | |
|------------------|--|
| Open Source | Throw it to the winds? Hand over to larger organization? Commercialize it? Just keep going? |
| Research Project | Ask a question (often suggested by previous cycle) |
| Company | “Liquidity Event” and continue “Liquidity Event” and assimilation Bankruptcy and die |

Lifecycle of BSD (1)

- Discover fascinating new technology written by a few people “under the radar” which mostly kinda works
- Grow frustrated by its limitations, while still admiring the underlying concepts
- Recognize all the other cool things it could do “if only”

Lifecycle of BSD (2)

- Find a charismatic leader who can convince already underpaid grad students to do even more work for free
- Share it, evangelize it, convince the original creators that you get it
- Be in the right place at the right time (i.e., when DARPA is adopting UNIX as their preferred platform). Win the contract
- Eventually do a (nearly) complete rewrite of the system

Lifecycle of BSD (3)

- Get sued by the original owner, who has finally woken up and noticed this thing that they own has real value
- Win the court case (eventually)
- Spin off from the university, while somehow miraculously retaining the original underlying principles (mostly)
- Set up sustainable models for running the projects
- Award maturity rather than profligacy

Two Mistakes Founders Make

- Assuming they know everything. They don't, and making other's life miserable is a good way to get forced out early
- Assuming everyone else is more knowledgeable and has no hidden agendas
 - Beware of people who tell you that their field is so arcane that you can't possibly understand it. Sometimes it's true, but not very often.
- Obviously, a happy medium is needed

Some Conclusions

- Without a doubt, commercial input to open source has permitted it to take on far larger problems
- Similarly, good marketing input permits open source take on different kinds of problems
- Conversely, open source has lost its innocence
- Corporations emphasize survival (i.e., money) over technological beauty

Bonus Third Mistake Founders Make

- Believing that they will be able to get back to coding by starting a company

Thank You

Eric Allman

euobsdcon-2012 (at) eric.allman.name

Bambi Meets Godzilla Video:

<https://www.youtube.com/v/n-wUdetAAIY>
