Lesson 2: The Engineering Economics of Open Source Software (or why we do what we do)

Stephen R. Walli

Objectives

- Understand the engineering economics of consuming OSI-licensed projects into one's own software projects.
- Understand the imperative of engineering contributing back when one consumes OSI-licensed projects.
- Understand something of the dynamic nature of software.

Notes:

https://github.com/jhu-ospo-courses/JHU-EN.601.210/tree/main/lessons/2#notes

Build versus Buy

- The cost of build from scratch vs the cost of buying/licensing
- Add in the cost of maintenance
- Add OSI-licensed projects and you add borrow-and-share to decision

Lines-of-Code & Boehm COCOMO

- Lines-of-code (LoC) is a horrible measure
- LoC is also an interesting relative measure
- Boehm's Constructive Cost Model (COCOMO) turned LoC and the team and aspects of the project into a (rough) value equation

The Interix Compiler Example

- Interix was the UNIX front-end on Windows NT (1995)
- ~300 software packages covered by ~30 licenses (pre-OSD)
- Running on Windows NT across Intel x86, DEC Alpha, MIPS CPUs.
- Needed to replace the Microsoft compiler base …

https://www.usenix.org/legacy/publications/library/proceedings/usenix-nt97/full_papers/walli/walli.pdf
https://medium.com/@stephenrwalli/running-linux-apps-on-windows-and-other-stupid-human-tricks-part-i-acbf5a474532

Build vs Buy for a compiler

- Venture funded startup with limited capital
- Build would involve turning the team to building a compiler on three computer architectures
- Buy would involve licensing+support from Intel and DEC costing hundreds of thousands (USD) and doesn’t solve for MIPS
- Borrow-and-share and the GNU Compiler Suite (gcc)

gcc costing (in 1997)

- An experienced compiler developer for 6-8 months ~$100K
- gcc runs on all architectures
- ~750K LoC == ~$10M-$20M (COCOMO) depending on cost/head
- Robust, hardened compiler including F77 and C/C++ front-ends
- This is two orders of magnitude of value capture
- But … now we're on a fork

gcc fork integration

- Every new major gcc release will cost approx. 6 months of integration
- If we get our changes merged upstream it will be ~1 month
- This is bringing integration costs into the $10K space
- The Interix engineering team hired gcc specialists (gcc maintainers) for $40K to merge to head rev
- The gcc project was evolving as well – by the time the Interix fork was added the gcc code base had grown to 1.2M LoC

Engineering Economics

- This wasn’t altruism – it was contributing back to the same hardening from which we benefited while managing engineering costs
- Economics works in the small – consider Apache httpd from Lab #1
- For a few lines of changes, a developer gets an entire webserver
- Economics works in a research community long-term as well
- The Booking Agents using Red Hat Linux while paying for JBoss

Side Notes

- It isn’t about altruism – it’s about economics
- It is economically an easier decision to use-and-share than build or buy
- You always get more than you give

Personal Economics

- Having one's name on key contribution streams in an open source community world is some of the best resumé content one can have:
  - to get work done,
  - to work in a collaborative engineering setting,
  - to demonstrate your understanding of a technology base.
- The connections you make in a community expands your network
- Reading software is a good way to improve one's skills, so reading known good software projects is an opportunity to improve skills.

All of this assumes that we are talking about healthy OSI-licensed projects …