# History of Open Source

Mukkai Krishnamoorthy (moorthy) Lally 305

# Licensing

- Our Lecture Slides have greatly benefited from the previous classes taught by Dr. William Shroeder and Dr. Luis Ibanez at RPI
- Also thanks to Dr.TV Raman for helping me understand the critical issues.
- Creative Commons (CC BY 3.0)
  http://creativecommons.org/licenses/by/3.0/

### Reading Material

 The Cathedral and the Bazaar by Eric Raymond

http://www.unterstein.net/su/docs/CathBaz.pdf Sections 1,2,3, 4 and 9

#### Open Source Issues

- Why Open Source?
- Why not Open Source?
- Difference between Open Source and Free Software

#### Why Open Source

- Fun (Form Communities, Engage in a hobby, Learning Experience)
- Profitable and Successful Business Models (Red Hat, inc, Cygnus Solutions, Service Oriented Business Model)
- Altruism serve the planet If the world improves, you improve

# Why Open Source (contd)

- Open Science and Engineering (Open Access to Data, Open – Access to Source Code, Collaboration oriented)
- Intellectual Freedom (idea is a property if and when that gets patented, you no longer have the freedom)
- Open Medicine (data, Chemical Compound, Effectiveness of Treatment)

# Why Open Source? (contd)

Scalable Software Development

Eric Raymond's The Cathedral and The Bazaar
 "Open source peer review is the only scalable method for achieving high reliability and quality"

#### The Cathedral and the Bazaar

- Cathedral Model (commercial world) done by a single person or by a chosen committee
- Bazaar Model (linux world) contribution by people – but used in alpha, pre alpha stage by a lot of people – Release early and release often

- Every Good Work of Software starts by scratching a developer's itch. - Most students projects tend to be on games!
- Good Programmers know what to write; Great ones know what to rewrite and reuse!
- Plan to throw one away; you will anyhow (Fred Brooks, "The Mythical Man Month")

- If you have the right attitude, interesting problems will find you (be part of a community)
- When you lose interest in a program, your last duty is to hand off to a competent successor.
- Treating your users as co-developers as your least-hassle route to rapid code improvement and effective debugging.

- Release early, Release often. And listen to your customers.
- When you lose interest in a program, your last duty is to hand off to a competent successor.
- Treating your users as co-developers as your least-hassle route to rapid code improvement and effective debugging.

- Getting a large enough beta-tester and codeveloper base, almost every problem will be characterized quickly and the fix obvious to someone
- Smart data structures and dumb code works a lot better than the other way around.
- If you treat your beta testers as if they are your most valuable resource, they will respond by becoming your most valuable resource.

- Any tool should be useful in the expected way, but a truly great tool tends itself to users you never expected.
- When your language is no where near Turingcomplete, syntactic sugar can be your friend.
- To solve an interesting problem, start by finding a problem that interests you.

- Many heads are better than one.
- When your language is no where near Turingcomplete, syntactic sugar can be your friend.
- To solve an interesting problem, start by finding a problem that interests you.

# Software Management Functions

Software Management has five functions.

- 1. Define goals and keep every one pointed in the same direction.
- 2. To Monitor and make sure critical details do not get skipped.
- 3. To motivate people to do boring and drudgery work.

# Software Management Functions (contd)

- 4. To organize the deployment of people for best productivity.
- 5. To marshal resources needed to sustain the project.

### Why Not Open Source

- Intellectual property concerns
- Chaotic development environment (volunteer based, distributed, no clear authority)
- Hard to change code (Public API visible, Internal structure visible)

### Why Not Open Source

- Benefits are a function of community size
- Proprietary business model (Better understood, Greater potential for \$\$\$)
- Hard to change code (Public API visible, Internal structure visible)

# Establishing An Open Source Project

- Create a clear vision (requirements doc) –
  Technical domain, Software Stack/Tools
- Involve team oriented people (big egos are big problems)
- Identify leadership/management structure (Methods to break conflicts)
- Establish an effective software process

# Establishing An Open Source Project (contd)

- Define Communication protocol chat room, developer mailing list, periodic face-to-face meetings.
- Pitfalls (Establish Core architecture early, start development with a few key people, Don't start testing soon, Using version Control, Lock up language)

#### Use external Tools

- Use external open source tools and libraries.
- Redevelopment is a waste of time (most of the time)

# Licensing

- Understand licensing and use one
- Select the software libraries that use similar licenses.
- Commercialization strategy (pure support, open toolkits but closed applications, open standards closed implementations, Open platforms, closed plug-ins)

### Reading Work

- The Cathedral and the Bazaar
- Free Culture (Introduction and Chapter 4) Find out where RPI is mentioned!

http://www.free-culture.cc/freeculture.pdf

# **Questions and Discussions**