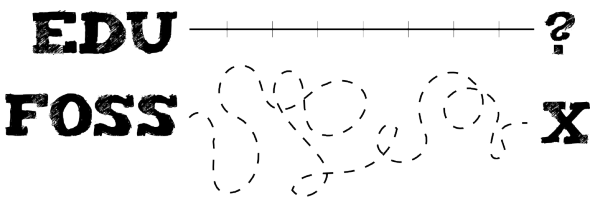
The difference between FOSS and Academia



The Dreyfus Model

1. Novice
2. Advanced beginner
3. Competent
4. Proficient
5. Expert

See the Wikipedia article: “Dreyfus model of skill acquisition” or http://bit.ly/docs-fable for more information.

Your first 5 minutes...

1. Where are the people? What are they talking about, and how? What's the project's mission/goals?
2. What's the user base like? Are they being supported? By whom? Do developers interact with them?
3. Where's the code? What's it written in? What does dev activity look like? Release cycle? Commit process?
4. How is this licensed? What businesses and/or other projects are or could be part of the ecosystem?
5. How do I set up an instance? Is this something I'd like to use myself?

A FOSS project is like...

- An ocean – an ecosystem that can be nurtured and harvested, fished-from, given-back-to, harmed and healed, shaped and researched, played-in and dived-into on many different levels.
- A farmer's market – a bazaar of many roles – farmer, chef, butcher, baker, procurer, consumer, coordinator.
- A lab lounge or project incubator – a place of shared discovery driven by an overarching rhythm and common practice, with fascinating collisions between structured projects and accidental learning amidst the coming-and-going of individuals.
- What else?

Want to learn more?

- Open source cultural principles: http://theopensourceway.org
- Applying open source to the world: http://opensource.com
- Our workshops for faculty: http://communityleadershipteam.org/posse (Next: July 23-24, 2011 in Raleigh, NC)
- Our open content curriculum: http://bit.ly/posse-curriculum
- Our community of practice: http://teachingopensource.org
- Everything from this talk: http://bit.ly/ccscne-2011-fieldtrip

Learning objectives

1. Be comfortable lurking in a FOSS community, and get a high enough signal-to-noise ratio to be useful for your teaching in 1-3 years. When you see people do FOSS stuff or hear them talking about it, you'll be able to follow the conversation.
2. We're not going to get you to the point where you could contribute or guide students – that takes more time than we have today. But you'll know where to ask about diving in, when & why you'd want to, and how to let us overhear your thoughts.
3. Be able to pitch a FOSS community on why *your* class of newcomers will help *them*.

Phrases you'll hear often

- Productively lost.
- Fail faster.
- “Oh!”

Who are we, anyway?

Mel Chua - Community Leadership Team, Red Hat (mel@redhat.com)

Sebastian Dziallas - Release Engineering, Fedora Project (sdz@fedoraproject.org)

a Red Hat community service

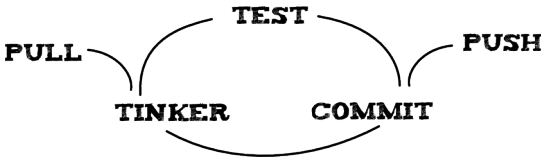
FOSS cultural principles

1. Default to open. Always ask how you can be more radically transparent – what exactly are you afraid of? Could you explain the situation to a friend? Then it's probably *not that bad*.
2. It's not what you know, it's what you want to learn. (Start now – you can.)
3. Pay it forward; document in exchange for lessons. This is a great way to start talking to people, especially if you're shy (we are.)
4. Release early, release often.
5. Show me the code. (Open source is a do-ocracy; those who do a task decide how it gets done.)
6. Given enough eyeballs, all bugs are shallow. (Also known as Linus's Law)
7. If it's not public and reproducible, it doesn't count.
8. Begin with the finishing touches. Don't reinvent the wheel – find something that's 85% of the way there and finish it. (Allow people to use your work for their finishing touches – this is where open licensing and open formats come in handy.)
9. Plan to improvise. Life is a series of pleasant surprises.
10. In general, it's better to communicate the undone than to do the uncommunicated.
11. Push to upstream.
12. It takes one yes to win. (Keep going.)

For more, see http://theopensourceway.org.

Open source practices

1. Your first 15 minutes in a project
2. Release cycles
   - Development begins
   - Feature acceptance & freeze
   - Alpha freeze & release
   - Feature completion
   - Beta freeze & release
   - Final freeze & release
   - Celebrate!
3. Proposing a feature
4. Version control and upstreaming



5. Tickets and ticket trackers
6. Marketing & buzz-building
7. Licensing & Business
8. Working with other project teams
   - Infrastructure/Sysadmin
   - Documentation
   - Design/Art
   - Testing/QA
   - Marketing/News
   - Ambassadors/Events
   - Translation/Internationalization
9. Packaging and distribution

Commonly used tools

1. Realtime communication: IRC
2. Longer-term documentation: Wiki (we will show Mediawiki)
3. Asynchronous updates: Mailing lists (we will show Mailman)
4. Shared thoughtstreams: Blogs and Planet aggregators
5. To-do lists: Ticket trackers (we will show Trac; Bugzilla is common too)
6. Workspaces: Version control (we will discuss git; svn is common too)

Uncommonly used tools

1. Realtime text-editing: Etherpad
2. Custom Linux distributions: Remixes
3. Thumbdrives: liveusb-creator
4. Look awesome: Design Suite
5. Be a polyglot: Transifex
6. Automatic note-taking: meetbot
7. Get local: LUGs and meetups
8. Get together: conferences, hackfests
9. Find beginner bugs: OpenHatch
10. Ask for help: people

We wish we'd known...

Mel: That I was good enough. I spent 6 years trying to “get ready” to contribute.

Sebastian: That talk is cheap and doing stuff is what really matters.