POSSE NOTES

Compiling-turning source code into machine(object code).

You can pay someone to patch an open source software?

Open source means that an end user can turn to any other provider because the code is “open.” This avoids vendor “lock-in.”

Open source companies begin to appear in the late 1980’s with the capabilities of fixing existing software on the fly.

“Productively Lost” – Throwing you in the deep end and letting you try to swim. Requires a tolerance of one’s own ignorance.

Packaging – the act of putting software (lots of raw code) into a format (one nicely bundled file\_ that is easy to install.

Upstream-producers

Downstream-consumers

Sebastian Dziallas—(Germany) talking about Sugar on a Stick. He is usually in Teachingopensource.org/sugar

Should get folks in Open Source community to talk about the community live in the classroom.

Deployment – roll out or use of a product

RSS feeds

Planets—fed to by RSS feeds

Aggregator—the software that runs a planet.

Pirate Pad

Create accounts on Fedora, Teaching Opensource,

To promote an open source project wiki, blog, IRC, mailing lists.

Git clone—download files from repository

Git diff – made a record of just the changes that you made to the files managed by git

Hat on—changing role

Commarch—community architecture

Spins

Remixes

<http://bugs.sugarlabs.org/ticket/581> Putting in a Sugar bug ticket.

<http://www.theopendisc.com/education/> educational open source software

<http://fedoraproject.org/wiki/FWN>

[#link](irc://irc.freenode.net/#link) <http://quadpoint.org/articles/irssi>[#link](irc://irc.freenode.net/#link) <http://quadpoint.org/articles/irssi>

<http://www.irssi.org/documentation> IRC client for me to check out

Red Hat Bugzilla is being used to track Fedora bugs and issues.

How to file a good bug:

<http://www.softwaretestinghelp.com/how-to-write-good-bug-report/>

1. Include steps to reproduce the bug
2. Make a case for why the bug is important
3. Be responsive to questions that appear
4. If you want to bring people’s attention back to the problem is to post another comment on the bug; this will generate update posts to everyone who is watching the bug. OR go into IRC and prod the person who is working on it.

New => confirmed => assigned => closed or fixed. “Post” means that a patch has been created but has not been approved/included.

PROCESS

Bug Report

Need Info

Closed Duplicate

Proposed Fix

Patch

New Release

Test

Feedback (can go back to Proposed Fix)

TRACK is the bug reporter/tracker for Sugar, is a lighter version of Bugzilla

<http://bugs.sugarlabs.org/>

Version Tracking system

<https://fedorahosted.org/web/>

<https://fedorahosted.org/fossrit/>

Another Version tracking system

<http://sourceforge.net/>

i18n = internationali(s/z)ation

Ohloh and OpenHatch collect information on open source projects.

Have students look at community and deduce how the community is organized

Who approves patches?

Who has commit access?

Who is involved in the history of the project?

Who are the maintainers?

Who is on the front and back end?

What have been some of the major bugs/problems/issues that have arisen during development?

[CDOT](http://cdot.senecac.on.ca/)

<http://chris.tylers.info/ols2008/>

DIGEST

Presentations for Karli

1. Participants in OSS community must implement some information overload strategies

2. There are people out there who are needlessly nasty and who are compulsively “Troll-ish” More on mail lists, less apparently in IRC (?)

3. a list of paradigms/habits that one needs to address/abandon in the open source environment would be useful as an introductory tool.

Vocabulary Powerpoint

Vendor lock-in: software vendors enclose software in a black box by selling compiled software written in machine code. The source code is protected as a copyrighted trade secret and is not available to the end user. Open source means that an end user can turn to any other provider because the code is “open.” This avoids vendor “lock-in.”

How much productivity is lost due to software being enclosed in a black box. Imagine if you bought a car or a motorcycle with a locked hood that only dealers could open. Imagine if it were considered a crime for you to even look inside the engine compartment of a car. God forbid you showed/told people what you saw inside.

Participants in OSS community must implement some information overload strategies such as?

a good info overload strategy in intensive meetings my memory of the best way to remember things in meetings is to immediately teach them back.  I had a day-long seminar once where we spent 20 minutes after each session going around the room (small group) and each person was required to talk briefly about two things they learned, and to come up with something they had concluded from each of those.  That was the hard part - it's easy to parrot, but you have to actually grasp the concepts don't know if something like that would work for your circumstance, but thought I would throw it out there. Also depends on what type of info overload and what you're trying to accomplish, but in general, when I'm overloaded the \*only\* thing I try to learn is how to search for and ask about things later. I'll let the actual answers pass in and out of my brain - write them down if I can (on a wiki or somewhere findable later), teach them back if I can, but... sort of let it go, knowing that I can find it again at some point. in a way, it's basically allowing yourself to trust your own ability to recover what you do need later - and only concentrate on what's immediately relevant to the short-term goal you're trying to accomplish. if that makes any sense - it sounds a bit zen to me as I re-read what I wrote. But basically, "remember search strategies and understand that you \*can\* always ask again."

Commarch—community architecture

Open Source Community Characterization Assignment

Look at community/project and deduce how the community is organized

Use [Ohloh](ohloh.net) and [OpenHatch](http://openhatch.org/) collect general information on open source projects.

You can use the projects IRC logs, git repo, mail list history, and open IRC channels to get the answers to the following questions:

Describe software project.

Give brief history of the project.

Look at the git repository and the devand ask the following questions:

Who approves patches? How many people?

Who has commit access? & how many?

Who is involved in the history of the project?

Have the principals changed over time?

Who are the maintainers?

Who is on the front and back end?

What have been some of the major bugs/problems/issues that have arisen during development?

How is the project’s participation trending and why?

"to eat one's own dogfood" - which means that developers are using the software product they're developing.