

Ryan Milligan
September 4th, 2007
Research Report #1

“Understanding Open Source Software Development” by J. Feller & B. Fitzgerald

Overview

open source software – software under distributed terms that comply with the OSD (12)
the open source definition is a document maintained by the open source initiative (12)
OSD: “open source” doesn’t just mean access to source code – it must meet criteria (13):

1. Free Redistribution – no restrictions from selling or giving away software
2. Source Code – must have code (or means of getting it) and compiled form
3. Derived Works – must allow modifications that can be distributed w/ terms
4. Integrity – must explicitly permit distribution of software built from code
5. No Discrimination Against Persons or Groups
6. No Discrimination Against Fields of Endeavor
7. Distribution of License – any rights must apply to all; no more licenses needed
8. License Must Not Be Specific to a Product
9. License Must Not Contaminate Other Software

two public licenses created by Richard Stallman’s Free Software Foundation (FSF) (18):

GPL: General Public License – users may modify software in any way they see fit

LGPL: Lesser General Public License – does not apply if only libraries are linked

various examples of open source software products (21):

Linux: fastest-growing, most high-level functionality is based on GNU software

FreeBSD: most successful of Berkeley Software Distribution OS’s, robust/secure

Languages and Editors, including Perl, Python, PHP, Tcl/Tk, and Emacs

What kinds of people make decisions that affect the Open Source Initiative?

Stallman heavily favors GPL, so why is LGPL still around?

History

BSD was established in 1977 at Cal-Berkeley by Bill Joy, future founder of Sun (29)

BSD group modified Unix, linked together research nodes – later became Internet (29)

Donald Knuth developed TeX, a typesetting system used in math and physics books (31)

Stallman founded FSF in 1979 at the AI Labs at MIT after a Xerox printer problem (32)

Estimates suggest that worldwide development on Linux exceeds 40,000 developers (34)

Since so many people work on modifying Linux, how many different versions are there?

Are some versions meant solely for internal use? If so, is that considered a violation?

If the FSF didn’t intend “free” to mean “free of cost”, then what did they mean by it?

Landscape

The two major advocacy groups want the same thing, but have very different views (41):

FSF has taken an ethical position focused on freedoms associated with software
OSI has taken a pragmatic position focused on the superiority of OSS
Stallman claimed that replacing “Free” with “Open” both solved and caused problems
Eric Raymond replied with “Shut up and show them the code” – favors the OSI view
Stallman’s stance involves principles while Raymond’s involves more tactics (44)
Today, it is sufficient to divide OSS companies into two categories (47):
Pure-play – purely OSS business model (Red Hat, VA Linux, Sleepycat, etc)
Hybrid – mix proprietary and OSS models (IBM, Apple, Dell, Gateway, etc)

*Wouldn't it just be easier to have one group oversee all of Perl instead of three (46)?
What, exactly, is an “open source software business model”?*

Analysis

Framework is based on two models for understanding software systems/development:

Zachman’s IS Architecture Framework (52)

material-oriented description: “what the thing is made of”

function-oriented description: “how the thing works”

location-oriented description: “where the connections exist”

people-oriented description: “who is doing what”

time-oriented description: “when events take place”

purpose-oriented description: “why choices are made”

Checkland’s CATWOE Framework (54)

clients may be internal or external and represent the beneficiaries

actors are the agents within the system who carry out the activities

transformation refers to the system’s core (input to output)

weltanschauung refers to the assumptions that underpin the process

owner refers to the agency that has prime concern for the system

environment represents the wider systems of which the system is a part

Based on ISA and CATWOE, there are five categories for OSS analysis (58):

Qualification (what), Transformation (how), Stakeholders (who)

Environment (where/when), World-view (why)

Does Zachman’s framework have completely set criteria, or is some of it subjective?

Qualification

An OSI “Certified” mark is not a defining characteristic of all OSS projects (66).

What role does the OSS development process play in defining a product as OS?

Is the software Open Source? (OSD serves as a useful tool for getting the answer)

Cosource Applications – majority deal with internet, drivers, systems, utilities (68)

Freshmeat Applications – majority deal with internet, software development, games (69)

Red Hat Linux 6.0 Applications – majority deal with applications, development (72)

Programming Languages – C, OO (C++, Java, Ruby), Scripting (Perl, Python, Tcl) (75)

If the “Certified” mark isn’t set in stone, then what purpose does it serve?

Transformation

Characteristics of the generic OSS development process (84):

- parallel development (85) – developers work on different aspects at the same time
- involves large communities (86) – OSS projects are often international in scope
- truly independent peer review (87) – not guaranteed in traditional environments
- prompt feedback (88) – developers work asynchronously in different time zones
- developer participation (90) – attracting skilled developers is a worldwide issue
- increased levels of involvement (92) – users also play an important role
- rapid release schedules (93) – focus is on the product, so time-to-market is big

OSS Development Life Cycle (101): planning, analysis, design, and implementation

primary is implementation: code, review, pre-commit test, release, debug, release

In parallel development, how difficult is it to stay on pace with other developers?

How is the shortage of developers being handled since there are so many OSS projects?

Stakeholders

Developer Communities (108)

- Linux studies show a lot of diversity within the developer group (gender, ethnic)
- most participants are new to the community, but are specialists/professionals

User Communities (112)

- demographic stats are scarce, but there is enough to detect a few trends

Commercial Organizations (115)

- IBM – builds and sells complete Linux systems as an ind. software vendor (ISV)

- Red Hat – can be characterized as value added resellers (VARs)

- Cosource – different in that it made a model out of the process, not the product

Non-commercial Organizations (121)

- Regular clients of OSS products (such as Apache, GNOME, Python)

*Review the table on page 124 for a brief overview of categories

Do companies like Red Hat lose any money in outsourcing for software from vendors?

Ryan Milligan
September 11th, 2007
Research Report #2

“The Cathedral and the Bazaar” by Eric Raymond

A Brief History of Hackerdom

Beginning of “hacker” culture can be traced back to 1961 – MIT acquires first PDP-1
ITS (Incompatible Time-Sharing System) was written in assembler/LISP (EMACS)
XEROX PARC (Palo Alto Research Center) was home to many innovations: (7)
 modern mice, windows, icons style, laser printer, local area networks, etc
Dennis Ritchie developed C for Ken Thompson’s Unix (portable, KISS principle) (9)
The first PC was marketed in 1975, and Apple was founded in 1977 (10)
In 1982 Stallman began the GNU project, a clone of Unix (in C) which was free (11)
The term “hackers” started to refer to computer vandals in 1984 (Unix / AT&T) (12)
Big rivalry of the eighties was between Berkeley Unix and AT&T Unix (13)
Those contributing to Unix were so bad in marketing, Microsoft took over (15)
Linus Torvalds developed Linux, a full-featured Unix with free/redistributable sources
 He was in competition with the BSD Unix (William and Lynne Jolitz) (15)
 BSD Unix was more technologically sound, but Linux had more contributors
 Along with the Internet explosion in the nineties, Berkeley’s Unix shut down

*Why didn’t Unix contributors hire marketing experts when Microsoft made a push?
Why didn’t the Jolitz’s think to take Torvald’s approach since it was so successful?*

The Cathedral and the Bazaar

Raymond describes the gradual shift from “cathedral”-style to “bazaar”-style: (21)
 Cathedral: “...crafted by wizards or small bands...isolation...no beta...”
 Bazaar: “...release early and often...delegate everything you can...be open...”
It was a shock to him that the bazaar-style development worked that well
Raymond’s “lessons” (using his personal Fetchmail experience as an example):
-Every good work of software starts by scratching a developer’s personal itch (23)
-Good programmers know what to write, great ones know what to rewrite (24)
 Torvalds didn’t start from scratch, reused code from Minix (for clone PCs)
-Plan to throw one away; you will anyhow (25)
 You don’t really understand the problem until you implement a solution
-If you have the right attitude, interesting problems will find you (26)
-When you lose interest in a program, your last duty is to hand it off (26)
-Treating your users as co-developers is your best route to rapid coding (27)
 Examples include both the Emacs editor and MATLAB
-Release early, release often, and listen to your customers (29)
-Given a large enough tester/developer base, every problem will be obvious (30)
 In Linux development, contributors for any project are self-selected
 More users find more bugs – there are more ways to see the problem

- Smart data structures and dumb code works better than the other way around (37)
- If you treat beta-testers as your most valuable resource, they will become so (38)
- Next best thing to having good ideas is recognizing good ideas from users (40)
- The most striking and innovative solutions come when you realize that your concept of the problem was entirely wrong (40)
- Perfection is achieved when there is nothing more to take away (41)
 - You know it's right when the code is getting both better and simpler
- Good tools are useful, great tools are useful in unexpected ways (44)
- When writing gateway software, disturb the data stream as little as possible (44)
- When your language isn't near Turing-completion, use syntactic sugar (46)
- A security system is only as secure as its secret; beware of pseudo-secrets (46)
- To solve an interesting problem, find one that is interesting to you (49)
 - The best hacks start as personal solutions and spread because it applies
- Many heads are inevitably better than one (54)

Software project management has five functions (57):

- To define goals and keep everyone pointed in the same direction
- To monitor and make sure crucial details don't get skipped
- To motivate people to do boring but necessary drudgework
- To organize the deployment of people for best productivity
- To marshal resources needed to sustain the project

Homesteading the Noosphere

Internet open source culture: everyone agrees that open source is a good thing (67)

Mentions two degrees of variation, zealotry and hostility to commercial software (68)

Each degree can be great, moderate, or little – nine different combinations

All nine combinations are represented in the open source culture

The most visible part of the hacker culture is greatly zealous and hostile

A number of communities budded from the Unix/Internet root (71)

Perl (Larry Wall), Tcl (John Ousterhout), Python (Guido van Rossum)

In 1997, common definition elements became the Open Source Definition (71)

Forking: duplicate sources, evolve them, but then claim to be the originator (72)

Open source licenses do nothing to prevent forking or pseudo-forking

There is a strong social pressure against it – requires renaming

Distributing changes without cooperation of moderators is frowned upon

Removing names from a project history is not done without consent

Three ways to acquire ownership of an open-source project (74):

Found the project yourself

Have project handed to you by previous owner

Owner loses interest, but project needs work (as long as no objections arise)

Noosphere: the territory of all ideas and the space of all possible thoughts (78)

Where our society is an “exchange” culture, the OSS society is a “gift” culture (81)

There are consistent rules to prevent ego and humility in hackers (94):

-If it doesn't work as well as I've been led to expect, it's no good (94)

-Work that extends the noosphere is better than work that duplicates (95)

-Work that makes it into a major distribution is better than work that doesn't (95)

- Utilization is the sincerest form of flattery (95)
- Devotion to hard, boring work is more praiseworthy than fun, easy hacks (96)
- Nontrivial extensions of function are better than patches and debugging (96)

In conflicts over open source software, we can identify four major issues (99):

- Who gets to make the binding decisions?
- Who gets credit or blame for what?
- How do you prevent duplication of effort?
- What is the “right thing”, technically?

*Has the zealotry and hostility towards commercial software decreased over the years?
 What else stemmed from the Unix/Internet root besides the popular scripting languages?
 Why don't OSS licenses do anything about forking? Doesn't it steal originator's credit?*

The Magic Cauldron

The following discriminators push towards OSS and away from closed source (146)

- Reliability, stability, and scalability and critical
- Design/implementation cannot readily be verified by means other than peers
- Software is critical to the user's control of his/her business
- Software establishes and enables a common computing infrastructure
- Key methods are part of common engineering knowledge

Are tools such as Klocwork an attempt at trying to eliminate open source peer review?

Revenge of the Hackers

Key themes to capture CEO/CIO/CTO types (177):

- Forget bottom-up approaches and go top-down
- Linux is the best demonstration case
- Capture the Fortune 500
- Co-opt the prestige media that serve the Fortune 500
- Educate hackers in guerrilla marketing tactics
- Use the open source certification mark to keep things pure

“JavaScript: A Beginner's Guide” by John Pollock

Read and Completed Mastery Checks:

- Module 1: Introduction to JavaScript (14/15 correct)
- Module 2: Placing JavaScript in an HTML File (14/15 correct)
- Module 3: Using Variables (15/15 correct)

Ryan Milligan
September 18th, 2007
Research Report #3

“OpenSources: Voices from the Open Source Revolution”

“The GNU Operating System and the Free Software Movement” – Richard Stallman (53-70)

Notes

- Started working at MIT AI Lab in 1971, which used ITS (OS written in assembler for PDP-10)
- “Free Software” had not been coined yet, but that’s what they were doing at the time with ITS
- In early 80’s, PDP-10 series was discontinued and all of the Lab hackers had been hired away
- The more modern computers (VAX, 68020) were not “free”, needed a disclosure agreement
- Stallman says that there are assumptions that software publishers make:
 - They think they have the natural right to own software and thus have power over users
 - They think the only important thing about software is what jobs it allows you to do
 - They think there wouldn’t be usable software if a company wasn’t offered power
- Stallman could have joined proprietary software (misuse skills) or given up computers (waste)
- He decided to go his own route, starting off with an operating system (GNU – Unix compatible)
- “FREE”: run program, modify it, redistribute it (for \$ or free), distribute modifications
- GNU system is NOT the same as GNU software (which was developed by others – still free)
- Worked on GNU... Developed GCC (w/o Pascal compiler)... Developed Emacs text editor
- “COPYLEFT”: give permission to run, copy, modify, and distribute, CAN’T add restrictions
- Free Software Foundation (1985): tax-exempt charity, most income comes from sales/services
- GNU Kernel: HURD (collection of servers, do various jobs) on top of Mach (microkernel, CM)
- Currently, Linux (developed in 1991) is combined with GNU (1992) = GNU/Linux

Questions

- How close are Linus Torvalds and Richard Stallman (just out of curiosity)?
- Are there laws against developing free software for use with proprietary software (like if someone wanted to start developing applications for Windows and make them free)?
- (<http://www.gnu.org/software/hurd/hurd.html>) – When the HURD kernel is finally completed – latest release is 0.2, will it have that big of an impact since Linux has been so successful?
- If something is becoming obsolete, why are licenses (although free), still in effect and somewhat hard to get a hold of? (<http://www.cs.cmu.edu/afs/cs/project/mach/public/FAQ/license.info>):

Subject: How to get a Mach license
Author: Mary R. Thompson
Date: Oct 1994

A Mach 3.0 license is required in order to get the complete BSD 4.3 compatible Mach 2.5/3.0 release. This is a rather obsolete system dating from 1992 which is only available for i386/486 platforms and supports the Tahoe level release of BSD 4.3.

CMU does not charge for this license, but with the cessation of Mach Project support, it may take quite a long time to process the paper work.

Before you can get a Mach license, you must have a Fourth Berkeley Software Distribution (4.3BSD) license. That license in turn requires that you have a Unix source license. Information about the BSD license can be obtained from <bsd-dist@cs.berkeley.edu>. These licenses were reasonably priced for educational institutions but not for individuals or small companies.

For example, what is the point of this? And why are so many licenses needed?

Thoughts

I'm starting to get over the idea that "free" has to mean "free of cost", but still getting used to the idea that the FSF sells its software for funding. Stallman comes across as bitter but seems to have good intentions with everything he has done (did he have a problem with the MIT group dissolving?). I first thought that the focus on extending the functionality of already-existing GNU system components (instead of integrating them to complete a bare-bones system) wasn't a very intelligent choice, but in the long run, it did build up two big things: money and enthusiasm.

Also Read "Why Software Should Not Have Owners" by Richard Stallman

"JavaScript: A Beginner's Guide" by John Pollock

Read and Completed Mastery Checks:

Module 4: Using Functions (15/15 correct)

Module 5: JavaScript Operators (15/15 correct)

Module 6: Conditional Statements and Loops (14/15 correct)

Ryan Milligan
September 25th, 2007
Research Report #4

“OpenSources: Voices from the Open Source Revolution”

“The Linux Edge” – Linus Torvalds (101-111)

Notes

Linux was not initially intended to be widely portable; the focus was having a solid design. (101)

-It was originally intended for just the Intel 386 architecture, which Linus had. (101)

Linux is a Unix-like operating system, but it is NOT a version of Unix. (102)

-FreeBSD: Started with the source code to Berkeley Unix, building off of the kernel.

-Linux: The kernel was written from scratch, without reference to Unix source code.

First Linux kernel “port”: Motorola 68K chip in an Amiga computer (102)

-Took same approach: wrote code from scratch to support an interface (considered a fork)

-First serious attempt was porting Linux to DEC’s Alpha (again, code from scratch)

-This inspired Linus to rewrite the kernel – one compatible with multiple architectures

The difference between microkernel-style and monolithic-style: (103)

-Monolithic: Memory is divided into kernel space (for kernel-level ops) and user space

-Microkernel: Much smaller set of ops, limited, less hardware-specific, more abstractive

-Linus took the monolithic approach – thought microkernels were complex and slow (claims microkernel approach was a “dishonest approach at getting research dollars”)

Linux kernel is NOT written to be portable to any architecture. (104)

-However, many architectures have enough core similarities. (Sparc, Alpha, PowerPC)

-Making a few assumptions can help simplify the process. (i.e. is there paging?)

With a monolithic kernel, it’s risky allowing new code and features into the kernel. (105)

-Avoid interfaces, or you’ll be stuck using the same one after you start coding with it.

(example: Microsoft only allowed 11-character filenames, had to make major changes)

The Unix kernel heavily relies on C to give it most of its portability; same with Linux. (107)

-Portability of Linux is tied to the fact that GCC is ported to many architectures.

2.0 Linux kernel introduced loadable modules, which created an explicit structure. (108)

Linux has achieved many of the design goals most thought only a microkernel could do. (109)

Questions

Where is Linux now? How many processors can it handle?

What kind of structure does Windows use? Is that the better choice for that type of OS?

Why is he so negative on Stallman’s Emacs? Is there something besides size that’s an issue?

Thoughts

At least for reading something of his for the first time, Linus Torvalds comes across as fairly arrogant, talking down on other operating systems and those who invest money in microkernels. I printed out the exchange of posts between him and Tenenbaum to get some more insight.

Also read the “Linux is Obsolete” post by Andy Tenenbaum

“JavaScript: A Beginner’s Guide” by John Pollock

Read and Completed Mastery Checks:

Module 7: Event Handlers (14/15 correct)

Module 8: Objects (13/15 correct)

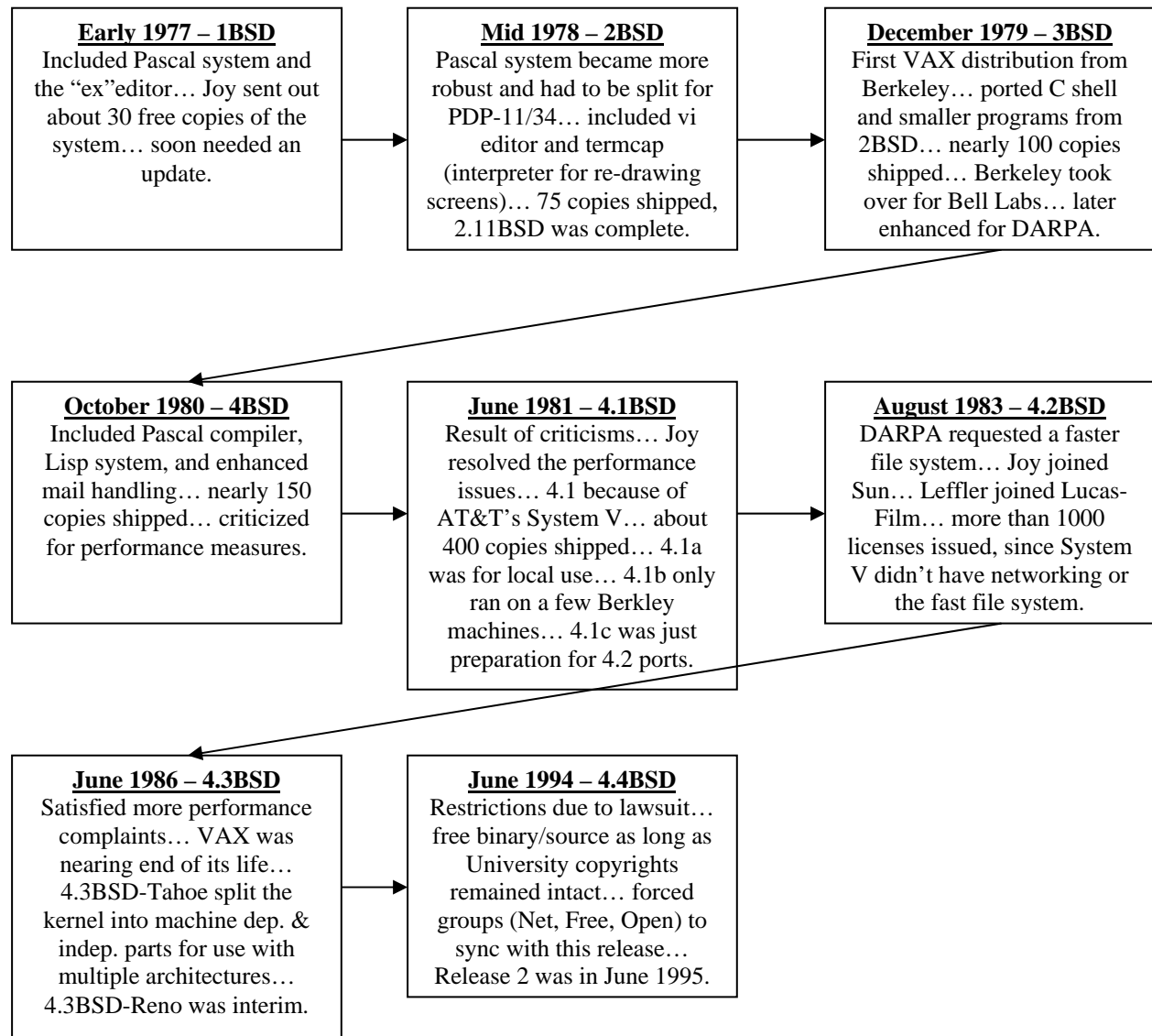
Module 9: The Window Object (13/15 correct)

“OpenSources: Voices from the Open Source Revolution”

“Twenty Years of Berkeley Unix” – Kirk McKusick (31-46)

Notes

A flowchart seems more appropriate for outlining this excerpt (BSD history and progression):



NetBSD was the doing of Bill Jolitz – complete system for 386 architecture, done almost entirely on the net.

FreeBSD formed a few months later – targeted less technically advanced users – shipped on low-cost CD-ROM.

OpenBSD spun off NetBSD – technical focus was improved the security – used FreeBSD ease-of-installation ideas.

Questions

- It can be seen that it took longer periods of time to release a new version of BSD as it matured throughout the years. Is this typical of large-scale open source projects?
- How closely linked are the Net/Free/OpenBSD groups? Do people work of two or all of them?
- What other major examples are there of a big commercial company (such as AT&T Bell Labs) sharing their work with a smaller group (such as Berkeley)?
- Are changes made to open source typically at the demand of the developers or the “consumers”?
(Question influenced by reading about how DARPA made requests concerning 3BSD)

DomainKeys Identified Mail (from Eric Allman’s talk on 9/24)

Printed out slides (created by Allman) from www.dkim.org, the official website of DKIM.

Concurrent Versioning System (CVS) on SourceForge.net

Here is a table of a few projects that I found involving CVS:

Name	# Devs	Description	License	Prog. Lang(s)
wxCvs	2	wxCvs is a cross-platform GUI frontend to the GNU Concurrent Versioning System (CVS). It is written in python and uses bindings to the wxWindows C++ cross-platform GUI toolkit. The python bindings to wxWindows is known as wxPython	GPL	C++, Python
Lomag Cvs	1	Lomag-Cvs is a graphical cvs client written in java. It uses the cvslib library from netbeans to handle the actual cvs calls. Lomag-Cvs focuses on providing a simple to use interface for beginning users of cvs.	Sun PL	Java
Cvs-Summary	1	CVS-Summary is a program that generates an HTML summary of CVS activity, very similar to that provided by the popular ViewCVS.	GPL	Python
DART	3	Digital Archive and Retrieval Tool (DART) presents engineers with an easy to use alternative to typical SVN or other CVS type systems. It allows engineers to easily archive data and provides an interface for managing and accessing store information.	Other	C#, Java, JavaScript

“JavaScript: A Beginner’s Guide” by John Pollock

Read and Completed Mastery Checks:

Module 10: Window Object (14/15 correct)

Module 11: JavaScript Arrays (14/15 correct)

Module 12: Math, Number, and Date Objects (15/15 correct)

JavaScript Tools Progress (for use with zerlot)...

Checkbox Input Linked to Predefined Text Fields:

So far, I have managed to create a list of checkboxes, each with a text value next to it, and a “submit” button below them. Underneath the submit button is an initially-empty text field that fills up with the checked values once the button is pressed. It’s very primitive, and only exists on a single page right now – I need to find a way to have the page with the checkboxes (the “schema browser”) open up a new page (the “query page”) with the text field.

Unix Epoch Time Converter:

The tool is almost complete – the only tweak that needs to be made is one that accounts for leap years (which I forgot to take into consideration initially). Basically, the user enters an integer (seconds since 1/1/1970) into a text field and clicks a “convert” button to convert that number to a date/time format (month/day/year @ hours/minutes/seconds). Matt mentioned the possibility of utilizing cookies to save international time formats for the appropriate visitors.

Ryan Milligan
October 9th, 2007
Research Report #6

“OpenSources: Voices from the Open Source Revolution”

“Future of Cygnus Solutions” – Michael Tiemann (71-90)

Notes

Cygnus: Founded in 1989, largest OS business as of 1998, primary product = GNUProDK (71)
GNUPro Developer’s Kit: leading compiler/debugger product in embedded tools market (71)
Tiemann read Stallman’s “GNU Emacs Manual” (self-published, readers could distribute copies)
Emacs: text editor, read/write email, newsgroups, shells, compile/debug, more features
Stallman also developed the GNU debugger and the GNU C Compiler (for VAX, Sun3)
Tiemann ported GCC to 32032, took 4 weeks to make it 40% faster than proprietary compiler
Tiemann’s inspiration: GCC + GDB + Emacs worked profoundly better than their alternatives
How much \$ is there in replacing proprietary technology with something better, faster?
1989: Proprietary software was dismal... primitive tools, built-in limitations, horrible support
Printing software copies = Printing money... COGS is negligible, margin is nearly perfect (76)
Advice: expensive to create standards, but it’s more expensive to work without any (77)
Cygnus created its own business model and concepts of finance, accounting, marketing, etc.
Basic business premise: proven technical support (2-4 times the quality for 1/2-1/4 of the cost)
By the end of the first year, over \$725K in support/development contracts written (78)
FOCUS was on the compiler/debugger... plans to support shell tools, etc. were thrown out (79)
GCC: Got it within 5% of Sun’s compiler performance, but CISC to RISC transition...
G++: Tiemann wrote GNU C++ in 1987, but had no time to keep it current with updates
GDB: “all over the map”... suffered fragmentation (hundreds made their own version)
Other Problems: no complete toolchain, no C library, competitors had complete products
Skeptics: if Cygnus came out with a successful shrink-wrapped product, there would be no need for the support business, and they’d be out of business within months.
Founders: Tiemann (GCC/G++), David Henkel-Wallace (library/support), John Gilmore (GDB)
Occasionally lost customers, but annually renewal rate remained around 90% consistently (85)

Questions

- How common is it for large-scale OS products to give extensive technical support to users?
- Current state of GCC/G++/GDB... how often is each one being updated these days?

“Open Source as a Business Strategy” – Brian Behlendorf (149-170)

Notes

PLATFORMS

Definition: defines a piece of software, enables people to use one piece on top of another
According to Behlendorf, business models based on platforms have long-term problems:

Platforms evolve so quickly that those who provide it have trouble keeping up.

Customers can eventually become dependant on a platform, spending lots of \$\$\$.

Win32: collection of routines defined by Microsoft for 95/NT application developers

CGI: allows web developers to write programs/scripts that run behind a web server (150)

GOALS/MARKETS

Based on a revenue model, doesn't seem like giving away software for free is profitable...

...but lower costs can result in motivated customers (who can help with big fixes)

Development costs can go down if customers contribute their own code. (155)

IMPORTANT: Before open-sourcing a project, you need to prove significant evidence that what you're offering is better than what people are currently using.

ROLEPLAYERS

Infrastructure support: Maintain mailing list, web server, CVS, bug database, etc.

Code captain: Monitor all commits and code quality, integrate patches, fixing bugs

Bug database maintenance: first line of support, forwards real issues to developers

Documentation/web: position that is often neglected and sometimes never touched

The "cheerleader": build momentum, find developers, push potential customers

LICENSES

BSD-Style Copyright (165):

Used by Apache and BSD-based OS (Free, Open, Net)

Basically, do what you like with the code, just give credit if you try to sell it

Best type of license for jumping into an existing project – no restrictions

The openness has risks – no incentive exists to contribute the enhancements

Mozilla Public License (166):

Developed by Netscape Mozilla team for use on their project

Changes must be released to same copyright, so it's available back to the project

Solid license – best to use to develop an end-user app where patents are an issue

GNU Public License (167):

There are certain aspects which are attractive for commercial purposes:

Mandates that GPL'd code is also released as code under the GPL

TOOLS

CVS, discussion forums, bug tracking (example: GNATS), etc.

“JavaScript: A Beginner’s Guide” by John Pollock

Read and Completed Mastery Checks:

Module 13: Handling Strings (14/15 correct)

Module 14: JavaScript and Forms (13/15 correct)

Module 15: JavaScript and Frames (11/15 correct)

*Are using frames advisable?

JavaScript Tools Progress (for use with zerlot)...

Checkbox Input Linked to Predefined Text Fields:

Demo...?

Unix Epoch Time Converter:

Demo...?

Other Design/Project Ideas...

A single window with all table descriptions (since it doesn't seem like any of them have been edited on individual pages... they can be listed alphabetically by table name (example: user_group)... just insert a link underneath "Other Information About the Data" that brings the user to the new page and right to the row that displays details about the requested table ([http://zerlot.cse.nd.edu/mywiki/index.php?_____#\(field\).html](http://zerlot.cse.nd.edu/mywiki/index.php?_____#(field).html)). This way, if a user knows which tables he/she wants to look up details about (especially multiple ones similar in name, they're all in one place.

Ryan Milligan
October 16th, 2007
Research Report #7

“OpenSources: Voices from the Open Source Revolution”

“Giving It Away: How Red Hat Software Stumbled Across a New Economic Model and Helped Improve an Industry” – Robert Young (113-126)

Notes

Early days – 1993 – Linux OS was a small software distribution company. (113)

-Offered Unix apps and books, but Linux CD-ROMs were becoming bestsellers for them.

Red Hat Software was created in January 1995 by Young and Marc Ewing. (114)

-Worked with dev. teams to assemble 400 software packages into an operating system

-Tests finished product and offer support and services for the Red Hat users

How to make money in free software? Build a great product with services, market it well. (114)

-OS development model produces flexible, customizable software – advantageous

-Patience is required for the market size and, ultimately, market share to grow.

Key to success: Establish a brand that Linux OS customers simply prefer over others. (116)

-Offer convenience and quality to customers, and define what an OS can potentially be.

-Customers should have the mentality that they’re purchasing Red Hat, not an oper. sys.

Example: Fermilab recognized Red Hat as a popular choice so they chose them for open source.

-Red Hat never responded to their calls, but still ended up doing business with them (119)

Licensing is practical, especially in this case (essentially shipping 435 separate packages).

-Open source chooses licenses that provide control over the software that is used. (121)

Questions

-What other companies with open-source models have done well in the market?

Thoughts

Robert Young’s mentality parallels that of Motorola’s CEO Ed Zander, who stressed brand management. Motorola does not lead in the market share, but still owns a decent chunk because of its ability to market new phone models and make its services known to the public.

“Diligence, Patience, and Humility” – Larry Wall (127-148)

Notes

“There is more than one way to do it” is true of both Perl and the open source community (127)

Three great virtues of programming are laziness, impatience, and hubris – opposite of the title.

Things are complex because they have to deal with reality (Perl, WWW, English language...)

-Often times, when we try to make things simpler, we end up oversimplifying.

-Things that are a “mess” fit well into the problem space, which is also a mess aka reality.

WWW is the perfect example of the concept, “there’s more than one way to do it.” (134)

People like things to be visually distinct from their surroundings. (143)

-Just like various classes of operators and variables in Perl are visually distinct...

It’s all a matter of perspective – Wall himself can’t predict what Perl will be like down the road.

-Language designer needs to see things from different perspectives so all can benefit.

“The Open Source Definition” – Bruce Perens (171-188)

Notes

OSD defines certain rights that a software license must grant in order to be certified as OS. (171)

Many programmers feel comfortable contributing to Open Source because they are assured:

-The right to make copies of a program, and distribute those copies.

-The right to access source code, a necessity before it can be changed.

-The right to make improvements to the program.

OSD started as a policy document of the Debian GNU/Linux distribution. (173)

-Debian was built entirely of free software, but there was a problem defining “free”.

-Free Software Guidelines made it possible to distinguish “free” and “non-free”.

Makes reference to Eric Raymond, the “Cathedral”, and the “Bazaar” – agrees with him. (173)

-Perens/Raymond worked together to edit guidelines and remove Debian-specific stuff.

-This formed the Open Source Definition, they later started the Open Source Initiative

The OSD is not itself a software license... see Raymond notes for the ten criteria. (176)

Covers licenses... refer to last week’s notes for overview of GNU, BSD, Apache, etc...

License	Can be mixed with non-free software?	Modifications don’t have to be returned?	Can be re-licensed by anyone?	Original copy-right holder privileges?
GPL				
LPGL	X			
BSD	X	X		
NPL	X	X		X
MPL	X	X		
Public Domain	X	X	X	

Questions

-What is the process one takes in writing a new license intended for open source software?

Also see other materials on OSDL/SCO/Mercurial...

Possible agenda for Fall Break: (?)

Continue developing JS tools and thinking of design proposals

Listen to some of the FLOSS web casts available on the research website

Start research paper (as least get a title/format/outline together)

Ryan Milligan
October 30th, 2007
Research Report #8

“OpenSources: Voices from the Open Source Revolution”

“The Internet Engineering Task Force” – Scott Bradner (47-52)

Notes

- Overview: All basic Internet technology (except TCP/IP) has been developed in the IETF. IETF standards are developed in an open process – anyone can participate. (47)
- History: Started in January 1986 – meetings were opened to the public later that year. (48)
It has never been a legal entity – \$\$ comes from gov’t grants and meeting fees.
Internet Society was formed in 1992 to provide a “legal umbrella” for the IETF.
- Structure: No specific criteria for membership (people, not companies, are members). (48)
Working groups are organized into areas, each of which has an Area Director.
Area Directors form the IESG (Steering Group) – 12-person Internet Arch. Board
- Groups: Almost all groups are formed when individuals get together and make proposals.
Working group charters are used to list deliverables, liaisons, limits, etc. (49)
The IESG must approve the charter before a working group can be created. (49)
- Documents: All IETF documents are freely available over the Internet – limited copyright.
Basic publication series in RFC – has both standard and non-standard tracks.
- Processes: There is no formal voting process; show of hands is used to get the consensus.
After being a Draft Standard for 4 months, proposal can be an Internet standard.

Questions

Looking at more recent material, it doesn’t seem like the IETF has changed. Is this a surprise?

“OpenSources 2.0: Continuing the Evolution”

“Open Source and Security” – Ben Laurie (57-70)

Notes

“Many Eyes” Theory

Once security vulnerabilities are tracked down, they are relatively easy to fix. (60)

Non-security bugs have a significant qualitative difference from security bugs.

Open source helps because those who want to look at the source code for errors can. (60)

Open v. Closed Source

There’s a growing interest in the claim that open source is more secure than closed. (61)

What does “more secure” mean? Fewer bugs? Less severe bugs? Purely subjective?

“Time to Fix (TTF)”

TTF = Time between spotting a vulnerability and developing a fix for it. (62)

Much take into consideration private (vendor) vs. public (open source) disclosure. (62)
If a closed source security problem is found and the author doesn't fix it... too bad.
With open source, users aren't at the mercy of the maintainer, they can do it themselves.

Projects

OpenSSL	Apache 2	Mozilla	
GnuPG	Enigmail	CVE	TOR

Check attached sheets for more information on these projects...

Questions

Is it a concern that someone may make modifications to OSS with intentional security flaws?

“Open Source and the Small Entrepreneur” – Russ Nelson (137-148)

Notes

Freemacs (Open Source)

Started writing a programmable editor for MS-DOS, based off Stallman's Emacs. (138)
Out of politeness, he asked Stallman if he could sell the Emacs memo with the editor.
Stallman later called him and convinced him to give the editor away for free.
“Freemacs” was ultimately distributed from an FTP site with useful MS-DOS software.

Freemacs (Business)

As the editor gained users, the users wanted updates so Nelson charged them a copy fee.
FreeDOS has adopted Freemacs as its standard editor – no commercial users. (141)
Freemacs only gave Nelson enough money so that he could buy a home computer...

Packet Drivers (Open Source)

Packet drivers allow the sharing of an Ethernet card between two protocol stacks. (141)
3Com's 3C501 was once the market leader, but it was very insufficient – one buffer.
Nelson wrote packet drivers himself and published them as open source software. (143)
Left Clarkson University and started Crynwr Software – income = packet drivers.

Packet Drivers (Business)

Most profitable type of service is “contract programming”, for bugs or new features.
-Charges per hour with a set maximum and minimum price range. (143)
Nelson has also sold proprietary packet drivers and dual-licensed packet drivers.

Questions

Is it safe to say that most OSS entrepreneurs do it for the sense of accomplishment, and not \$\$\$?

“FLOSS Weekly” Podcasts (<http://nd.edu/~oss/FLOSS/floss.html>)

Chris DiBona – Google – 4/7/2006

Gaming industry is difficult – expensive with art, marketing, keep up w/ state-of-the-art.
Google's servers run on the Linux kernel, which has some customized modifications.
-Actual number of servers belonging to Google is disclosed information.

Google uses OSS – Linux and languages (C++, Python) – Van Rossum works @ Google.
Linux World – Conference in Boston, used to be in San Jose – mostly for business now.
Anyone using a Mac is using BSD... DiBona is a big fan of OpenBSD.

OpenBSD has a lot of cryptography, they're not comfortable developing in USA.
“FLOSS” explanation – “Libre” is more accurate of the intention than “Gratis” (Spanish).
Open source – you can always make something better rather than start from scratch.

Ben Goodger – Mozilla Firefox – 4/14/2006

Ben helps guide the engineering work that gets done for Firefox releases (interface code).
User interface is built with XML (tags, buttons, etc) – makes it easy for web developers.
Mozilla used to be a department within Netscape – still uses pieces from old management
Management felt it would be worthwhile to get more development help... OSS.
Extensions, themes, plug-ins, etc. in Firefox are written in XUL (XML User Language).
Mozilla launched an add-on site that allows users to install/enable/disable these.
Back/forward page cache has been said to cause “complex memory leaks” in V1.5.
Browser downloads the page's HTML... the cache keeps it for instantaneous use.
Mozilla 2.0 – New bookmarks/history system that uses SQL Lite, which is open source.
Tab navigation has also been modified for better usability based on feedback.
Firefox is a very profitable OSS company... \$17M a year on the Google search bar.
In the open source world, as long as there is a demand for a product, it'll get developed.
Thunderbird is also built with XUL (lot of code common w/ Firefox), own user interface.
Perfect examples of the power of open source and how it can be modified.

JavaScript Tools – Progress and Ideas

Field description window and textfield/checkbox code are both in progress, nearing completion.

Any other ideas...?

Ryan Milligan
November 7th, 2007
Research Report #9

“OpenSources 2.0: Continuing the Evolution”

“Open Source and Proprietary Software Development” – Chris DiBona (21-36)

Notes

One common misconception is that proprietary developers are different than OSS developers.

It is rare to find someone who just works on proprietary or open source software. (21)

The two groups often learn from each other – OSS developers are the “crazy cousins”.
The difference lies in the programs themselves, which are built using similar tools. (22)

It isn't just OSS that benefits from code reuse – proprietary software has libraries too.

Even companies who claim to not use code from other projects use these libraries.

Barriers when using existing code: interfacing issues, standards for security and style, etc. (23)

Software can be difficult to add to code if it was developed in a different language or platform.

There are very good reasons to have a consistent code style – it aids in debugging. (25)

Given the time, programmers often prefer to learn from other people's code without using it.

According to DiBona, proprietary development has been changed by OSS ideals. (26)

If there hadn't been tools like the GNU compilers, industry would have to make them.

Mentions CVS/SVN/Btkpr, and SourceSafe, which is more of a local version controller. (29)

Communication is the key to maintaining OSS development. Best example: SourceForge

Distribution: OSS developers have created very smart packaging/installation systems.

Most updating is now done online instead of CDs – Linux installation is a WIP. (31)

Proprietary software has changed OSS: bugs/security, testing, project scaling, and control. (33)

Questions

Do proprietary software companies ever make libraries open source if they go under?

...or is that a licensing issue?

What are some other significant things that proprietary developers have learned from OSS?

“Patterns of Governance in Open Source” – Steven Weber (361-372)

Notes

The rise of the Internet made it much easier to discover potential collaborators/subcontractors.

Complex software is hard to build – demands multiple dimensions, division of labor. (363)

One way to manage this is to enclose the process within a proprietary organization.

The OSS community comprises a very different organizational model.

A successful method for development must allow the community to change and share dialog.

“With more eyeballs all bugs become shallow.” – it depends on how eyes are organized. (366)

Observing the OSS community suggests that sometimes, it's not worth trying to get it right.

Sensible alternative: parse uncertainties so that the systems become more robust.

Seven Design Principles for a Referee Function: (368)

Weighting of Contributions: Not everyone is equally knowledgeable about a problem.
Contributor v. Contribution: It is often easier to pre-qualify info based on the person.
Status Quo v. Change Bias: The process of learning is in large part one of forgetting.
Timing: Ask yourself... How urgently should information be used, refereed, updated?
Granularity of Knowledge: Where do you draw the boundaries around a module?
System Failure Mode: How failures present themselves are critical design challenges.
Security: Becomes a greater consideration as the value of the system rises over time.

Questions

Are most OSS projects started by group of people who mutually know each other?
...or do random people tend to hop in and start contributing?

“FLOSS Weekly” Podcasts (<http://nd.edu/~oss/FLOSS/floss.html>)

Rob Malda – Slashdot – 4/22/2006

Slashdot interface used to consist of terrible code – now has better HTML standards.

Now has dynamic forums... integrating a new tagging system for users.

Bookmarks are a good generic way to define a data type (in this case, a URL to a site).

Slashdot launched in the fall of 1997, and it was an inspiration for Wiki software.

Wiki was initially a mess and took a few years to get all of the bugs fixed.

Slashdot posts about one story for every twenty-five submissions, trying to increase it.

Often times, story are approved twice because one person forgot he posted it.

The Internet is different from TV in that podcasts and news streams don't show emotion.

JavaScript Tools – Progress and Ideas

Demo the generic and zerlot-specific JavaScript file that fills in a text area based on selection.

Also starting to format/outline final paper.

Ryan Milligan
November 16th, 2007
Research Report #10

“OpenSources 2.0: Continuing the Evolution”

“The Mozilla Project: Past and Future” – Mitchell Baker (3-20)

Notes

Mozilla project was launched in March 1998 – Netscape Communicator source available. (3)
All code that Netscape didn't have the right to license under an OSL was removed.
Mozilla Public License was written and approved by the Open Source Initiative.
Basics of the OSS model were known at the time, but acting on them was “new territory”. (3)
Differed from Cygnus, Sendmail, and Apache... leadership interacted with commercial teams.
Netscape wanted to see a broad development effort to produce browser products as shared. (4)
Initial problem: only people who knew the code well enough were Netscape employees.
Also, no volunteer communities yet, which needed to happen for future success.
Creation of mozilla.org was important because it meant Netscape management gave up control.
1998 launch of the Mozilla Organization was mostly Netscape employees with a vision.
Gradually changed development styles from a proprietary to an open-source model. (6)
A public bug- and issue-tracking system was set up as OSS and named Bugzilla. (6)
Early 21st century: people thought Mozilla would die out, but 1.0 debuted in June 2002.
Cross-platform applications were created with an XUL toolkit, which became award-winning.
A very disciplined methodology was used for code checked in to Mozilla products: (8)
Who checked it in? History of bugs? Who reviewed it? Broken on any platforms?
Mozilla codebase is very large, so there was reason to always worry about code quality.
Another issue was determining who has “write-access” to the source code repository. (10)
Netscape remained the largest distributor of Mozilla-based products – lots of talent. (11)
Mozilla Foundation was launched on July 14, 2003 as an indy nonprofit organization. (13)
Donators included AOL (\$2M), IBM, Sun, and Mitch Kapor – 10 employees initially.
Ben Goodger and Scott McGregor were the lead Firefox and Thunderbird developers. (15)
Security issues with the Internet in 2004 somewhat set back development schedules.
1.0 versions were finally released in November 2004 – 2 million downloads first 2 days.

Questions

Is limiting write-access to open source code a contradiction in this case, or a security measure?
Why would companies like AOL fund a potential competitor?
Do some people shy away from open source products because they fear a lack of security?

“A Tale of Two Standards” – Jeremy Allison (37-56)

Notes

Samba developers understand both Windows/Unix environments, created “glue” between them.

Samba is one of the most complex OSS projects out there, and one of the hardest to join. (37)

The POSIX standard (Portable OS Interface-X) was developed in the late 1980's. (38)

Much more detailed than operating system APIs – system commands, scripting, etc.

The document costs money (IEEE) and reads more like a legal document – very detailed.

Not static – Super Unix Specification is a superset of POSIX developed in 1998. (44)

Linux changed everything, and is now considered the dominant version of Unix.

The Win32 standard was Microsoft's attempt to move beyond DOS and compete with Unix.

When released, the Win32 subsystem call interface was entirely undocumented. (47)

This allowed Microsoft to modify the call layer and “cheat” where others couldn't.

There exist applications that allow snooping on NT kernel calls made by applications.

While the POSIX standard has better documentation, the Win32 API has good security. (48)

However, the Win32 security mechanism is simply too difficult to use.

It's important that businesses/governments selecting products pay attention to open standards.

Questions

Are there any rules that state a piece of OS/proprietary software must have documentation?

“The Open Source Paradigm Shift” – Tim O'Reilly (253-272)

Notes

“Paradigm shift” has been generalized to describe a profound change in our frame of reference.

O'Reilly's premise is that both open source and proprietary developers are doing things wrong:

Open source developers are failing to realize the consequences of change – others benefit.

Proprietary developers are still playing by the old rules as new ones come into play. (255)

Most widely-used applications on the Internet are a result of either Linux or FreeBSD.

The operating system is only a component – the true platform is the Internet.

According to O'Reilly, open source is an expression of three deep, long term trends: (255)

Commoditization Software has been driven by standards due to Internet-oriented systems.

Software that starts as proprietary will eventually become standardized.

Google and Amazon use the restrictive GPL but still aren't constrained.

Custom distributions will be a key differentiator among Linux vendors.

Commoditization of software drives value to the services it enables.

Collaboration Heart of the phenomenon was the use of networking to connect interests.

The cultural shift first hit software, but is tied to licenses/philosophies.

Looking at OSS licensing as a means of encouraging collaboration helps.

Customizability Dynamically typed languages like Perl, Python, and PHP are important.

Perl has been referred to as the “duct tape” of the Internet.

Sites like Google/Amazon that use them are processes, not products.

It's better to think of open source as a field of scientific and economic inquiry, not just software.

“FLOSS Weekly” Podcasts (<http://nd.edu/~oss/FLOSS/floss.html>)

Larry Augustin – VA Software – 5/19/2006

Linux became a viable OS around 1992-1993... three times the performance as SunOS.
VA = James Vera and Larry Augustin, the two founders of the software company.
They were the first Linux system integrators – very few were able to set it up.
Customers looked at it as a cheaper, better version of working with a Sun system.
There were a lot of people who wanted a great Sun system but just couldn't afford it.
Comparisons to Dell... Dell is a packager, and an effective distributor of open source.
Open source: There are enough people out there who want to fix bugs, other problems.
Augustin says that the open source software is better as a result of this.
Linux was unpopular for a very long time, took years before they got consistent funding.
No capital venture funding was received by Linux until 1998.
Early on, no one knew whether they should pick Linux or go with a version of BSD.
Linux didn't have any shared libraries in the beginning, BSD had more features.
Community marketing = user group is incredibly important, even when they don't buy.
VA Software had engineers as their marketers, since they understood the products.
Software used to be really expensive to write – lots of R&D dollars were invested into it.
Open source alleviated some of this, helped a lot of businesses start up (training).
Some companies don't immediately go open source because they need to clean up code.

Randal Schwartz – Perl – 7/15/2006

Schwartz... Perl developer since 1980s when he downloaded v1.0 of the language.
Perl 3 (1989) is where people started to realize just how useful it could be.
One of the more common criticisms of Perl is there are multiple ways to do everything.
Wall doesn't just add features to add them, will only do so if they're effective.
One of the main goals of Perl is to eliminate as many keystrokes as possible.
Perl was not developed for someone who would be using it an hour or so per week.
Most powerful developers use it often and know what it's capable of doing.
Schwartz... never got schooling in computer science, but has taught grad-level.
Sites running Perl: TicketMaster, City Search, Match.com, Amazon, Yahoo!, IMDB
Perl is still working on creating more standardized classes, objects, threading issues...

JavaScript Tools – Progress and Ideas

Ryan Milligan
December 4th, 2007
Research Report #11

“OpenSources 2.0: Continuing the Evolution”

“Libre Software in Europe” – Gregorio Robles (161-188)

Notes

The spread of the Internet made a libre software community possible in Europe in the late 80's.

The evolution of libre software in Europe paralleled the Internet's penetration. (162)

Most European countries (Germany, France, Spain) have grown communities in isolation.

Torvalds (Linux) = Finnish, van Rossum (Python) = Dutch, Widenius (MySQL) = Swedish
Lerdorf (PHP) = Danish... a lot of foundations and companies also started up in Europe. (163)

In a FLOSS-US study, about 900 of 1500 developers declared themselves to be living in Europe.

Highest # of European developers: France, Germany, Italy, UK, Netherlands, Spain.

These numbers show a correlation with the countries with the greater populations.

Although hosting facilities/infrastructures exist in Europe, the U.S. is well ahead (SourceForge).

BerlioOS (Germany) = 2,400 projects, Software-libre.org (Spain) = 66, Gna! (France) = 360

Events: LinuxTag (Germany), Solutions Linux (France), Linux Forum (Denmark), and more...

The ITEA Report on OSS (January 2004) is aimed at uncovering business opportunities. (168)

Companies: SuSE (Germany), Mandriva (France), Open CASCADE (France)

Trolltech AS (Norway), MySQL AB (Sweden)... Nokia and Ericsson also use OSS.

France, UK, Germany, Italy, Finland, Denmark, etc. all have their own national initiatives. (178)

In 2004, the French police saved EUR 2 million by switching to OpenOffice.org.

In 2003, Munich started to migrate to GNU/Linux on its desktops – political decision.

Most OSS licenses were formed in the United States in a jurisdiction alien to Europe. (182)

Validity was a concern; there have been efforts to translate/localize those licenses.

One of the fields in which open source has made the biggest impact in Europe is education. (183)

“Libre” software provides real advantages in terms of innovation and social benefits in Europe.

Questions

Has the United States at all taken into consideration how European countries benefit from OSS?

“Open Source Biology” – Andrew Hessel (281-296)

Notes

OSB = Idea that products such as drugs/vaccines can be developed w/ open intell. prop. models.

Academic science supports the belief that knowledge evolves quicker with shared ideas.

With biology facing a paradigm shift, genetic and software engineering are converging. (281)

Late 1960's... biological science was tied to academia, chemistry/engineering were industrial.

Early 1970's... DNA technology became big, so a new generation of companies appeared. (282)

Since it dealt with biology, academic scientists became executives and businessmen. One of the questions during this time was how biotech intellectual property would be managed. Today, the biotech industry is expanding rapidly due to pharmaceutical research/development. The problem: a lot of research never gets to the development stage due to IP issues. (286) Both big corporations and small companies struggle with IP and high R&D costs. Another problem: no incentive for companies to improve drugs or develop new technologies. The success of OSS has shown evidence that open biotech development may be viable. Internet has dropped the direct cost of scientific research while encouraging IP freedom. (288) Two efforts: OneWorld Health and The Biological Innovation for Open Source initiative. Both of these somewhat resemble an OSS model, since DNA is “biological source code”. “Synthetic biology” is a platform of software tools used to design and test DNA molecules. (290) Dozens of bioinformatic tools have already been released under OS licenses. It would be a great advantage if OSB can start up developer communities for biological software. RISK: This may raise the chance of hackers or terrorists developing malicious designs. Gaining access to synthetic DNA is not particularly difficult, even for individuals. (292) Open source biology would save researchers time... no more writing patent applications. (294) Open source biology would save researchers money... not nearly as many legal/IP costs. Finally, OSB fits in with trends in medicine/science... individual variation plays a large role.

Questions

In this case, could someone write open source code then give it to a proprietary company?

“Community Many to Many” – Jeff Bates (373-396)

Notes

Goal: find networking conditions such that more participants improves communication quality. Slashdot defies today’s trends... the more people who post, the more valuable discussion. Earlier blogs resembled hand-built directories complete with links, guidance, comments. (374) Rob Malda’s (“Chips and Dips”) was pure HTML, people submitted by sending e-mail. By the spring of 1998, Slashdot had dynamic content through CGI and Apache modules. Malda challenged Netscape to open source its browser code, and Mozilla was later released. It did not take long for Slashdot’s popularity to boom... by March 1998, 100000 hits per day. “Slashdot Effect”: the sequence of posting a story, rushing to the site, failure to load. Controversial modification came when Slashdot decided to moderate comments by type/quality. Slashdot achieved comment moderation through principles used in the roots of open source. At any given time, there are roughly 1,850 users moderating comments! (381)

Questions

Do open source project leaders sometimes turn volunteers down because it may slow things up?

“FLOSS Weekly” Podcasts (<http://nd.edu/~oss/FLOSS/floss.html>)

Guido Van Rossum – Python – 8/4/2006

Originally, “ABC” was designed for educational purposes (learning how to program). At the end of the 1980’s, things were borrowed from the ABC language to create Python. One of Python’s biggest advantages is that it has strict formatting & a simple interpreter.

This results in very clean source code and indentation that is easy to follow.

Whereas Perl is “there’s more than one way to do it”, Python is “there’s only one way”. From the surface, it kind of resembles C, although it’s designed differently under the hood. Python developers have decided to wait instead of coming out with new versions often. Why Python? Lots of third-party libraries that can be downloaded for immediate use.

Developers are trying to make it so that Python can run in a browser just like JS.

Many members of the Python development community are close acquaintances.

A lot of very large websites run purely on Python, Google uses it in their infrastructure.

In Python, everything is an object, but not everything is a class (easy overloading).

(Went to python.org – briefly skimmed through pages and a tutorial to get idea of syntax)

Rasmus Lerdorf – PHP – 8/11/2006

PHP originally started as a set of tools used to manage Personal Home Page tracking. As people started using the tools, they wanted more, so PHP started referring to the code. PHP was written inline in the HTML, and it executed in C... it was just a tag interpreter.

Now, it really doesn’t stand for anything... it’s a hypertext pre-processor.

Is “burning out” an issue in open source communities? Some do leave for other jobs.

There are now over 1100 contributors and developers to the PHP project (CVS account).

PHP is not particularly known for its marketing... PHP6 will have Unicode support.

A lot of users are actually still running version 4, don’t really need version 5.

There are various warning levels, but it isn’t as strict in terms of formatting as Perl.

PHP5 “filters” all data coming in – a lot gets stripped out, leaving just raw text.

PHP Projects: “Moodle” – learning management system for univ. (grades, schedules).

“Sahana” – Crisis/Disaster management system by developers in Sri Lanka

Advice: best way to learn code is by taking an existing program and changing it.

(Went to php.net – briefly skimmed through pages and a tutorial to get idea of syntax)

Jeremy Allison – Samba – 10/13/2006

Samba is used everywhere as a method of networking systems – Window/Unix “glue”.

What motivates people to put up with so much grief just to develop free software?

Allison enjoys doing it, makes developing more of a “town council” atmosphere.

People take it into their own hands to “scratch their own itch”, improve skills.

Few people do it with the intention of getting back at proprietary companies.

Samba is not a file system, it’s a file server. Standalone file servers are a commodity.

It enables one to take a Linux desktop and put it into a Windows active directory.

Another reason for open source... Windows = “black box”, people know what’s in OSS.

Ryan Milligan
December 11th, 2007
Research Report #12

“OpenSources 2.0: Continuing the Evolution”

“Open Source and the Commoditization of Software” – Ian Murdock (91-102)

Notes

Success attracts attention, and competitors offer low prices to compensate for average quality.

Low-end providers often employ strategies of imitation, let big names “fight it out”. (91)

The open source movement represents another disruptive shift in the software industry.

In the first two decades of the software industry, every layer was proprietary to the vendor. (93)

Every computer spoke a different language due to lack of standards, hard to interoperate.

Firms such as Apple, Apollo, and Sun began to create products that targeted the individual user.

The strategy and new “personal computing” products were received with much success.

For hardware, IBM used off-the-shelf products instead of developing proprietary parts in-house.

For software, Sun based its workstations on the Unix OS, which was popular in labs/academia.

As the Unix market grew, competition became fierce and the market fragmented. (94)

“Divide and conquer”: Intel and Microsoft were given an entire market base to sell to.

As Microsoft grew, so did competitors, namely Linux in the 1990’s – goal = out-commoditize.

Dell is a perfect example of supply chain... negotiated with multitude of suppliers for parts. (96)

Dell shows that commodity markets call for an entirely different business model.

Linux is not a single system, it’s hundreds of subsystems built by thousands of individuals. (97)

A Linux distribution is a collection of software with the kernel to form a complete OS.

While Linux companies kept “value add” proprietary, RedHat was purely open source.

RedHat Enterprise Linux- certified “high-end” platform, no longer freely redistributable.

Questions

To what extent is “open source hardware” implemented and circulated out in industry?

Has anyone such as Linus Torvalds openly disagreed with the way RedHat conducts business?

“Open Source and the Commodity Urge” – Andrew Hessel (103-120)

Notes

When the industry started, software was just something used by vendors to help sell hardware.

Customers knew they were paying mostly for hardware – true today with embedded.

No single company did more to promote applications than Microsoft (Visual Basic, Office, etc)

Customers benefit as vendors focus on solving business problems and forming models. (105)

Vendors also benefit from increased use of OSS because it removes the “IP safety blanket”.

Code is open, so vendors need to find innovative ways to lock in customers. (105)

As software prices drop, more buyers will enter the market, so everyone wins.

In copyleft and open source IP, the code matters less than the coder, unlike proprietary software.

The code can be replicated, but the influence on the community takes effort. (106)

Not all vendors need an open source strategy, but they need to compete with cost/distribution.

Open source enables a vendor to maximize market penetration at minimal cost, the main goal.

Hessel uses the phrase “Porsche technology at Pinto pricing” to describe the OSS model.

Source code extends benefits beyond those who exercise the right to modify it. (111)

Access to source code offers choice, and choice almost always ensures lower prices.

“Both Source” model: offers a way to fill open source’s gaps and charge a “premium” service.

Other models: Dual-License (MySQL), ASP (software is delivered to customers via Internet)

Questions

Are the various Linux distributions (RedHat, Debian, Ubuntu) competitors or collaborators?

“Under the Hood: Open Source and Open Standards Business Models in Context” – Stephen Walli (121-136)

Notes

Common examples of voluntary IT standards include SQL, HTML, TCP/IP, C/C++, C#. (123)

Standards act as a “yardstick” so that competitors can be judged in the marketplace.

While standards exist to encourage multiple implementations, patents protect implementations.

An interesting dividing line in licensing schemes is whether a license is considered “viral”. (126)

A reciprocal license such as GPL attaches itself to new software, same rules apply.

Many companies participate in OSS projects to help them deliver both products and services.

There is a competitive edge to OSS community developers (e.g. two vendors w/ similar product)

Questions

Why would someone create an open source license that isn’t viral? Does that leave it vulnerable?

“FLOSS Weekly” Podcasts (<http://nd.edu/~oss/FLOSS/floss.html>)

Eben Moglen – Free Software Foundation / GPL3.0 – 9/27/2006

Eben = The “Legal Godfather” of OSS... ex-professor who worked underneath Stallman.

There needed to be strong encryption for e-commerce, but the US had rules with exportation.

IBM, HP, and Google all have the insight that there is a lot of value in open source software.

If companies aren’t careful as to how they develop products, there can be legal issues.

At the time of the podcast, GPL3.0 was in its second draft – eventually released on 6/19/2007.

Last time GPL was updated was in 1991... didn’t make many important provisions.

As the Foundation gets more and more specific in terms of problems, versions narrow.

One issue deals with mixed code from different licenses, such as GPL, LGPL, and BSD.

Some licenses over time pick up their own attributes, which can be a potential problem.

The complexity that comes with license conflicts is one of the biggest concerns in GPL3.0.

Companies like Google have come out with some important code management tools.

Jon Hall – Linux International – 3/24/2007

A little bit of hardware used to go a long way, it was expensive software that was the problem.

For example, it was common to spend \$100,000 on a single copy of a compiler.

“Early open source” - DECUS (Digital Equipment Corporation User Society) and IBM’s Share

These companies had code libraries that were submitted by scientists and engineers.

They had no time to market or make profit off their code, so they made it available.

1972 – A “catalog” of code cost \$5 (the fee was mostly to account for the code’s disc).

Jon Hall went to countries such as Fiji where people heard of Linux but couldn’t access it.

Brought a CD and told them to download it – used mostly for academic purposes.

They no longer had to spend millions of dollars on proprietary software packages.