FOSS and Software Engineering

Tony Wasserman
Carnegie Mellon Silicon Valley

FOSS 2010 Workshop, UC Irvine
February, 2010

Carnegie Mellon
SILICON VALLEY

What *is* “Software Engineering”?

A process by which an individual or team organizes and manages the creation of a software-intensive system, from concept through one or more formal releases

Carnegie Mellon
SILICON VALLEY

40 Years of Software Engineering R&D

- Good understanding of the basic principles of software design and development
  - Abstraction
  - Modularity and openness
  - Coupling and dependencies
  - Internationalization and localization
- Tremendous increase in complexity of systems that we are able to design and build
  - “Standard” architectures and frameworks
  - Extensive component and subsystem libraries
  - Powerful open source tools

Carnegie Mellon
SILICON VALLEY

FOSS and software engineering

- Vast number of free and open source tools to support software engineering processes
- Wide variety of software engineering practices in FOSS development
- Comparing quality of FOSS software vs. SaaS and proprietary software

FOSS development tools

- Have completely transformed the market for software development tools
  - Effectively driven tools cost to zero
- Are widely used by developers of both proprietary and open source software projects
- Are available for a broad range of platforms: Windows, Mac, Linux
- Are available for almost every aspect of the software development process

Sampling of FOSS development tools

- IDEs: Eclipse, NetBeans, RadRails
- Requirements Mgmt: OSRMT
- Visual modeling: ArgoUML, OpenAMEOS
- Issue tracking: Bugzilla, Trac, Mantis
- Version control: Subversion, CVS
- Build tool: Ant
- Code repositories: SourceForge, GForge, Tigris
- Java Unit Testing: Junit
- Test Management: TestLink
- Scripting languages: Perl, Python, PHP, Ruby
- Web GUI builders and toolkits: YUI, GWT, Qt Creator, Dojo/Wavemaker
- Project and process management: Redmine, OpenProj, IceScrum

Carnegie Mellon SILICON VALLEY

FOSS Tools build on FOSS components

Example: IceScrum

- Java EE application
- Five layer architecture
- FOSS components
  - ICEfaces Ajax library
  - Spring
  - Hibernate
  - Tomcat
  - MySQL
  - Ant
  - JUnit

Carnegie Mellon SILICON VALLEY

“Instant” FOSS application development

- Prebuilt open source stacks for
  - Content management systems: Drupal, Joomla, Alfresco, Plone
  - Customer relationship management: SugarCRM
  - Wikis: DocuWiki, MediaWiki
  - Project management: Redmine
- No new code means no new bugs

Carnegie Mellon
SILICON VALLEY

FOSS and software engineering

- Vast number of free and open source tools to support software engineering processes
- Wide variety of software engineering practices in FOSS development
- Comparing quality of FOSS software vs. SaaS and proprietary software

Carnegie Mellon SILICON VALLEY

Huge variety of FOSS projects

- Mirrors variety of proprietary project teams
  - Individual contributors
  - Small co-located teams
  - Small distributed teams
  - Large distributed teams
  - Commercial teams
- Community, foundation, and commercial open source
  - Membership
  - Project cohesion
  - Management and governance

Differences among FOSS projects

- Leadership, management, and governance
- Project size (people, code base)
- Project team (experience, technical background)
- Technology base (platforms, languages)
- Individual reasons and goals for participation
- Geographical separation
- Cultural differences
- Commercial pressures (release schedules, roadmaps)

These differences strongly affect SE processes.

Carnegie Mellon
SILICON VALLEY

Four examples

- Tiny community project
  - Unfunded project with 1-2 junior people, no schedule, no plans for wide use by others
- Large community project
  - Unfunded project with core leadership team, numerous volunteers, no business model
- Large foundation-based project
  - Managed project with core team, many volunteers, overall governance, and large audience for testing and use
- Commercial open source project
  - Managed and controlled contributions, roadmap, business plan

Carnegie Mellon SILICON VALLEY

Project Characteristics (1)

- Commercial open source projects behave similarly to commercial proprietary projects
  - Employees and paid contractors write all core code
  - Management hierarchy for project management, staffing, etc.
  - Business and marketing decisions influence features, user interfaces, and release dates
  - Controlled releases, not nightly builds
  - Community participates in testing
  - End users expect high quality software
  - Commercial customers will pay for support and professional services

Carnegie Mellon
SILICON VALLEY

Project Characteristics (2)

- Large foundation-based open source projects
  - Foundation leadership approves project and influences releases
  - Core project management team oversees committers and screens contributions from volunteers
  - Many contributors paid by their employers to work on project
  - Open discussions through IRC channels, Wikis, or similar mechanisms
  - High degree of transparency and participation
  - Community participates in testing
  - No direct commercial support

Project Characteristics (3)

- Community-based open source projects
  - Little or no formal governance except for commitment rules
  - Lower cohesion with higher contributor turnover
  - Few schedules: “It's done when it's done”
  - High degree of transparency and participation
  - Volunteers may not have appropriate technical skills
  - No built-in audience for project; hard to build a “community”
  - No commercial support

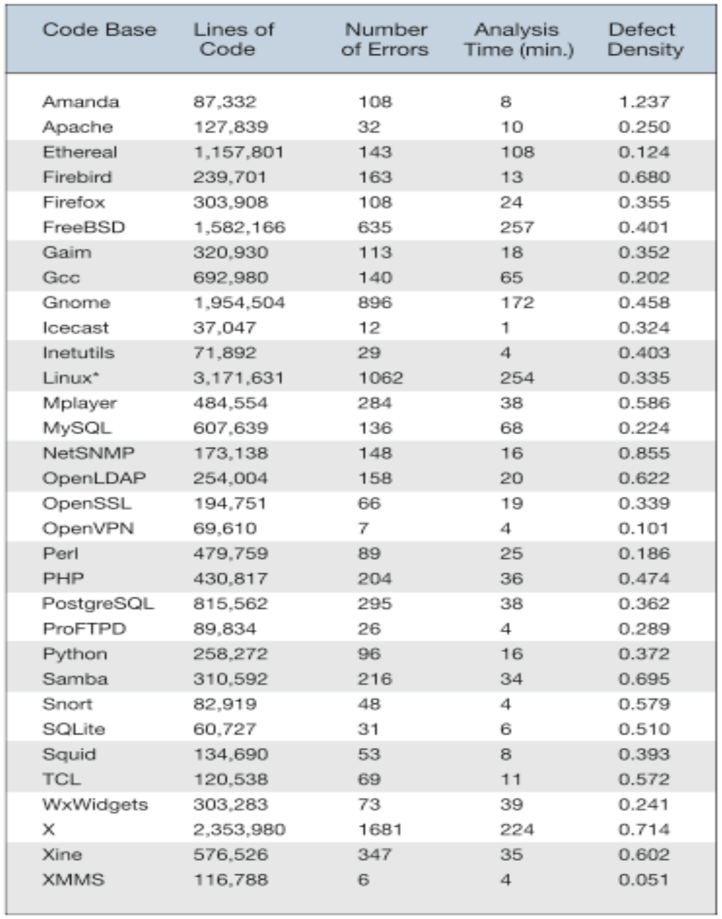
FOSS and software engineering

- Vast number of free and open source tools to support software engineering processes
- Wide variety of software engineering practices in FOSS development
- Comparing quality of FOSS software vs. SaaS and proprietary software

Carnegie Mellon
SILICON VALLEY

Coverity Scan project on open source quality http://scan.coverity.com

Table 1



Code Base	Lines of Code	Number of Errors	Analysis Time (min.)	Defect Density
Amanda	87,332	108	8	1.237
Apache	127,839	32	10	0.250
Ethereal	1,157,801	143	108	0.124
Firebird	239,701	163	13	0.680
Firefox	303,908	108	24	0.355
FreeBSD	1,582,166	635	257	0.401
Gaim	320,930	113	18	0.352
Gcc	692,980	140	65	0.202
Gnome	1,954,504	896	172	0.458
Icecast	37,047	12	1	0.324
Inetutils	71,892	29	4	0.403
Linux*	3,171,631	1062	254	0.335
Mplayer	484,554	284	38	0.586
MySQL	607,639	136	68	0.224
NetSNMP	173,138	148	16	0.855
OpenLDAP	254,004	158	20	0.622
OpenSSL	194,751	66	19	0.339
OpenVPN	69,610	7	4	0.101
Perl	479,759	89	25	0.186
PHP	430,817	204	36	0.474
PostgreSQL	815,562	295	38	0.362
ProFTPD	89,834	26	4	0.289
Python	258,272	96	16	0.372
Samba	310,592	216	34	0.695
Snort	82,919	48	4	0.579
SQLite	60,727	31	6	0.510
Squid	134,690	53	8	0.393
TCL	120,538	69	11	0.572
WxWidgets	303,283	73	39	0.241
X	2,353,980	1681	224	0.714
Xine	576,526	347	35	0.602
XMMS	116,788	6	4	0.051

http://www.coverity.com/library/pdf/open_source_quality_report.pdf

Qualipso Project (qualipso.org)

Qualipso

Quality Platform for Open Source Software

Trust and quality on **Free** and **Open Source** systems

- Competence Centers
- Business Models
- Next Generation Forge
- Trustworthy results and process
- Information management
- Interoperability
- Legal Issues

Carnegie Mellon
SILICON VALLEY

Qualipso results on trustworthiness

- Studied 96 projects against 11 dimensions, including
  - Repository
  - Standalone vs. part of larger project
  - Application type
  - Developer organization
  - Size of project team
  - User community size
  - Programming language
  - Tool support

- Report available at http://www.qualipso.org/node/84

Carnegie Mellon
SILICON VALLEY

Some key SE research questions

- Is open source software of “higher quality” than traditional commercial software?
- Is open source software more secure than traditional commercial software?
- Is the community-based open source development model more effective than other approaches to software development? (Cathedral vs. the Bazaar)
- What are the most effective approaches to open source leadership and project governance?

Carnegie Mellon
SILICON VALLEY

Tony's Hypotheses

- The more polished the release, the more it costs to build
  - Scalability, user interface, installation, documentation
- No significant differences in quality between FOSS and commercial software created by professional software developers
- No significant differences in processes between commercial open source and other commercial software
- The larger the company, the more likely it is to use commercial software development tools

Carnegie Mellon SILICON VALLEY

Contact information

Anthony I. (Tony) Wasserman

post:	Carnegie Mellon SV Moffett Field, CA 94035 USA
tel:	+1.415.641.1180 (ofc) +1.415.612.0600 (m)
email:	tonyw@sv.cmu.edu
Skype:	tony.wasserman
Twitter:	twasserman
Googletalk:	tony.wasserman

Carnegie Mellon
SILICON VALLEY