Course "Softwareprozesse"

Open Source SW (OSS) Development Basics

Lutz Prechelt
Freie Universität Berlin, Institut für Informatik

Part 1:

- What is OSS?
  - Licenses
- Who builds it?
  - "True" OSS
  - Commercial OSS
- Value

Parts 2 & 3:

- Self-organization
- Quality assurance
- Comparison to agile
- Inner Source

Questions

- What is Open Source SW?
- How important is it?
- Who builds it? Why?
- What is 'value'?
  Who is the 'customer'?

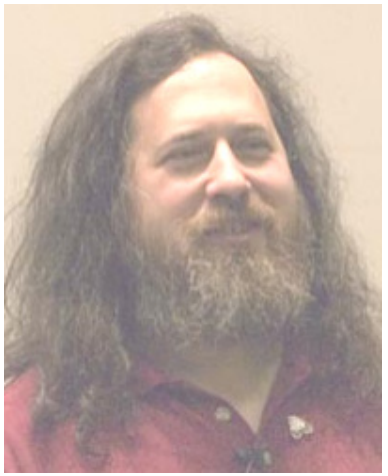
Part 1, today

- How does self-organization work?
  - Basic infrastructure
  - Typical process
  - Leadership
  - Process innovation patterns
- How does quality assurance work?
- Is this agile?
  Is it modern view?
- Is an open process useful *within* companies?
  - Inner Source

At Bloom levels 3 or 4, the ability to discuss these is also the learning objective

Definition "Free Software"

Richard Stallman, Free SW Foundation FSF
http://www.gnu.org/philosophy/free-sw.html


Richard Stallman

- The freedom to **run** the program, for any purpose (freedom 0)
- The freedom to **study how** the program **works**, and adapt it to your needs (freedom 1).
  - This requires access to the source code.
- The freedom to **redistribute** copies so you can help your neighbor (freedom 2).
- The freedom to **modify** the program, and **release your improvements** to the public, so that the whole community benefits (freedom 3).

On Richard Stallman, see

- https://en.wikipedia.org/wiki/Richard_Stallman and http://www.catb.org/~esr/writings/rms-bio.html

Definition "Open Source Software"

- Stallman calls such software *"Free Software"*
  - he promotes it actively since **1985**
  - http://www.fsf.org/ Free SW Foundation
- Today, the more common term is *"Open Source Software" (OSS)*
  - This move was initiated in **1998** by Eric Raymond:
    - because the term *free* "makes a lot of corporate types nervous"

Eric Raymond

- Academically, sometimes also termed "Free/Libre and Open Source Software (F/LOSS)"
  - abbreviated FLOSS or shortened to FOSS or F/OSS
- Free SW now has two "home organizations": FSF and OSI, the Open Source Initiative
  - http://opensource.org/

open source™

The OSS turning point

- [Fitzgerald06] The introduction of the "OSS" term marks a dramatic mainstreaming of F/LOSS development and use:
  - many more OSS developers
    - in particular many more paid OSS developers
  - more pragmatic, less ideological attitude
  - many new business models
    - proliferation of licenses
  - enormous uptake by users
  - enormous uptake by developers as component users
  - appearance of vertical OSS applications
  - some larger-scale OSS projects
  - more explicit, more structured development processes

- Fitzgerald (but nobody else) calls this "OSS 2.0"

Note:
What is an Open Source project?

- The so-called Open Source project is in fact an organization
  - projects are "temporary efforts" (pmi.org)

- Open source organizations are not typical organizations:
  - ad-hoc ("for this"; unlike most companies or associations)
  - mostly without a legal shell, unless large
    - prominent exceptions exist (e.g. Apache SW Foundation)
  - with fuzzy membership

Contrasts: proprietary, shared source, closed source

- Most company software is *proprietary* ("eigen", "geschützt"): The copyright holder reserves the right to use the software
  - either to himself (custom SW)
    - this is the default case in most country's copyright laws
  - or to people who accept restrictions regarding the use of the SW and usually pay a license fee (commercial SW products)

- Usually (but not always) proprietary SW is *closed-source*
  - meaning even the allowed users only get to see a binary version

- If not, this is sometimes called "shared source"
  - e.g. from Microsoft (main purpose: create trust)

OSS licenses: Overview

- There is a rather large number of OSS licenses that define the rights of the public with respect to the software
  - e.g. (as of 2006-10) Academic Free License • Adaptive Public License • Apache Software License • Apache License, 2.0 • Apple Public Source License • Artistic license • Attribution Assurance Licenses • New BSD license • Computer Associates Trusted Open Source License 1.1 • Common Development and Distribution License • Common Public License 1.0 • CUA Office Public License Version 1.0 • EU DataGrid Software License • Eclipse Public License • Educational Community License • Eiffel Forum License • Eiffel Forum License V2.0 • Entessa Public License • Fair License • Frameworx License • GNU General Public License (GPL) • GNU Library or "Lesser" General Public License (LGPL) • Historical Permission Notice and Disclaimer • IBM Public License • Intel Open Source License • Jabber Open Source License • Lucent Public License (Plan9) • Lucent Public License Version 1.02 • MIT license • MITRE Collaborative Virtual Workspace License (CVW License) • Motosoto License • Mozilla Public License 1.0 (MPL) • Mozilla Public License 1.1 (MPL) • NASA Open Source Agreement 1.3 • Naumen Public License • Nethack General Public License • Nokia Open Source License • OCLC Research Public License 2.0 • Open Group Test Suite License • Open Software License • PHP License • Python license (CNRI Python License) • Python Software Foundation License • Qt Public License (QPL) • RealNetworks Public Source License V1.0 • Reciprocal Public License • Ricoh Source Code Public License • Sleepycat License • Sun Industry Standards Source License (SISSL) • Sun Public License • Sybase Open Watcom Public License 1.0 • University of Illinois/NCSA Open Source License • Vovida Software License v. 1.0 • W3C License • wxWindows Library License • X.Net License • Zope Public License • zlib/libpng license
  - for details see http://www.opensource.org/licenses/
  - some concise summaries: http://choosealicense.com/licenses/
- but they all derive from only 2 basic types:

OSS licenses: Essentials

- We have seen Stallman's definition of Free Software
  - which may appear somewhat vague, at least untechnical
- According to the OpenSource Initiative (opensource.org), the defining characteristics are the following:
  1. Right of free redistribution
  2. Source code availability
  3. Derived works (and their distribution) are allowed
  4. Undue restrictions must not be present:
     - no discrimination against persons or groups (e.g. "valid for IBM employees only"),
     - no discrimination against fields of endeavor (e.g. "no military use"),
     - no further steps required (e.g. signing a non-disclosure agreement, making a registration), etc.

- For our purposes, both definitions are equivalent.

OSS licenses: 2 basic types (GPL, BSD)

Freie Universität Berlin

The most crucial difference between licenses is their requirements for derived works:

- The "**copyleft**" licenses require that derived works, if distributed, are also distributed under the same license
  - Prototype representatives: *GNU General Public License* (GPL), GNU Affero General Public License (AGPL), *GNU Lesser General Public License* (LGPL)
    - Different definition of derived work (strong/verystrong/weak copyleft)
  - Private (undistributed) derived works are allowed
    - But even *running* a web app publicly is distribution for AGPL.

- The **liberal** licenses allow that derived works can be published under a different license
  - often including closed source licenses
  - Important representatives: MIT, BSD 2-cl, BSD 3-cl, Apache

Note: Is copyleft "viral"?

- Some people claim, GPL code will "infect" other code with which it is combined
  - any GPL component will make *the entire system* fall under GPL
- This is incorrect:
  - Copyleft pertains to "modified works" (GPL version 3).
- The GPL says:
  - *"To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy."*
- So only modifications to the GPL'd code fall under copyleft
  - not the remainder of a system in which that GPL'd code is used:
  - Modify the GPL'd work, GPL the result, copy it, combine the "exact copy" with your own code.
    - (Where object code is involved, the license wording is complex, but the effect is still the same.)
- Still, the GPL "makes corporate types nervous" to this day.

OSS licenses: Other types (MPL, variants)

- A few licenses can be considered "in between" the copyleft and the liberal licenses

- Sort of a middle ground is defined by the *Mozilla Public License* (MPL):
  - it discriminates deriving from existing parts (which must keep their previous license) from deriving by adding new parts (for which one can choose a license freely)
  - This means that, say, a company can still build proprietary extensions of a work, but has to publish changes in existing parts back to the community.

- Most licenses differ from these 3 types only by minor additional restrictions/permissions (patents, commercialization)
  - (1) perhaps only wording differs, (2) many licenses have multiple versions or variants, (3) small differences can be highly relevant!

Contrast: proprietary licenses

- Most closed-source licences not only
  - require paying a license fee and
  - do not offer seeing the source code
- but also
  - usually prohibit modifying the product
    - often even for fixing bugs!
  - usually prohibit reverse-engineering
  - sometimes prohibit public benchmarking (e.g. Oracle DB)

OSS licenses: Consequences

- When creating derived works from multiple OSS products at once, make sure the respective licenses are compatible
  - You will sometimes need a lawyer to answer that question
  - Example problem: The original Apache Software License was not compatible with the GPL since it contains a patent retaliation clause. (Apache Version 2 has resolved that)
    - https://www.gnu.org/licenses/license-list.html

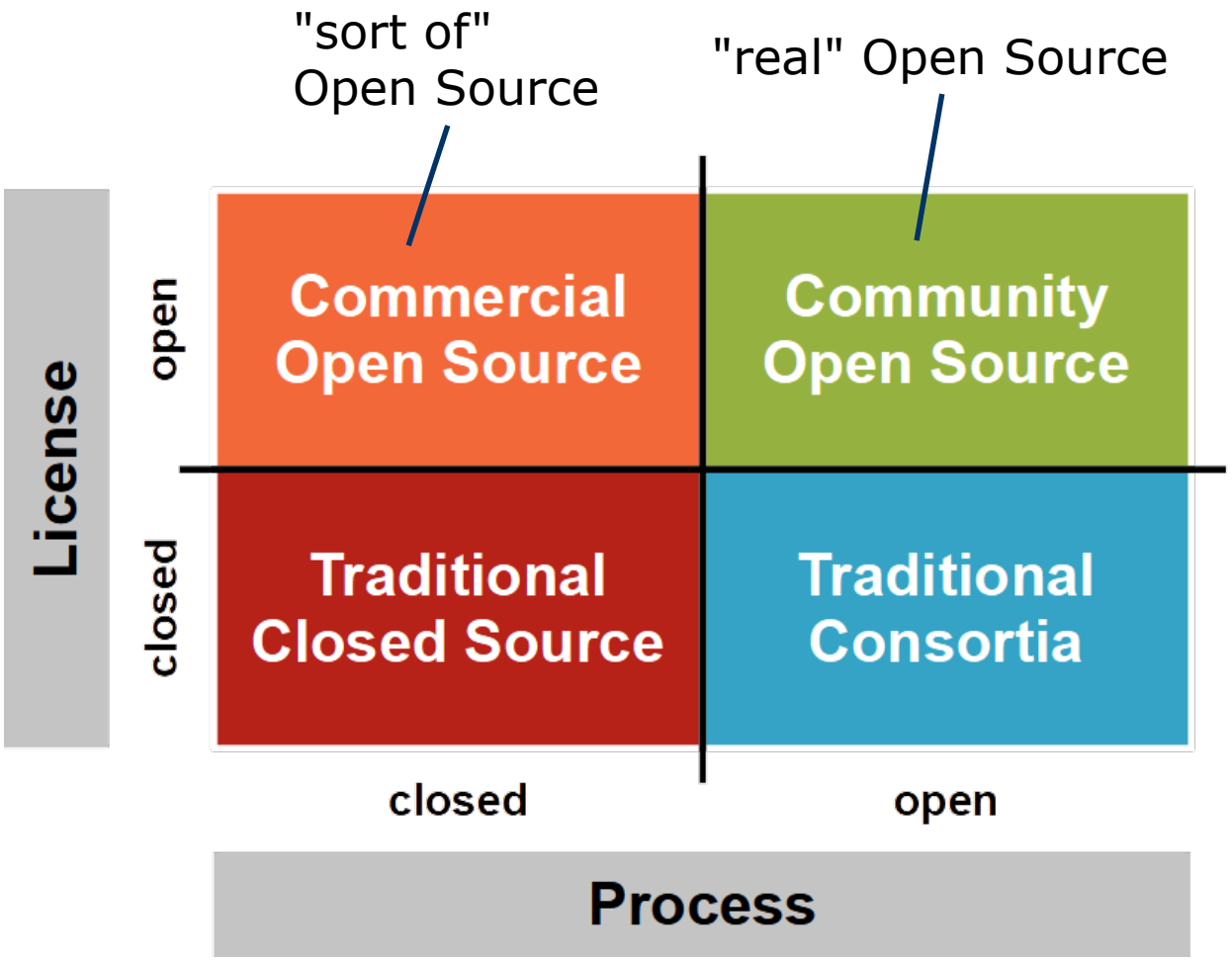
See also: https://opensource.guide/legal/

OSS licenses: Commercial Implications

- OSS licenses don't forbid selling the software
  - But in case of copyleft you still have to provide the source code
  - Liberal licenses are flexible for commercial applications.

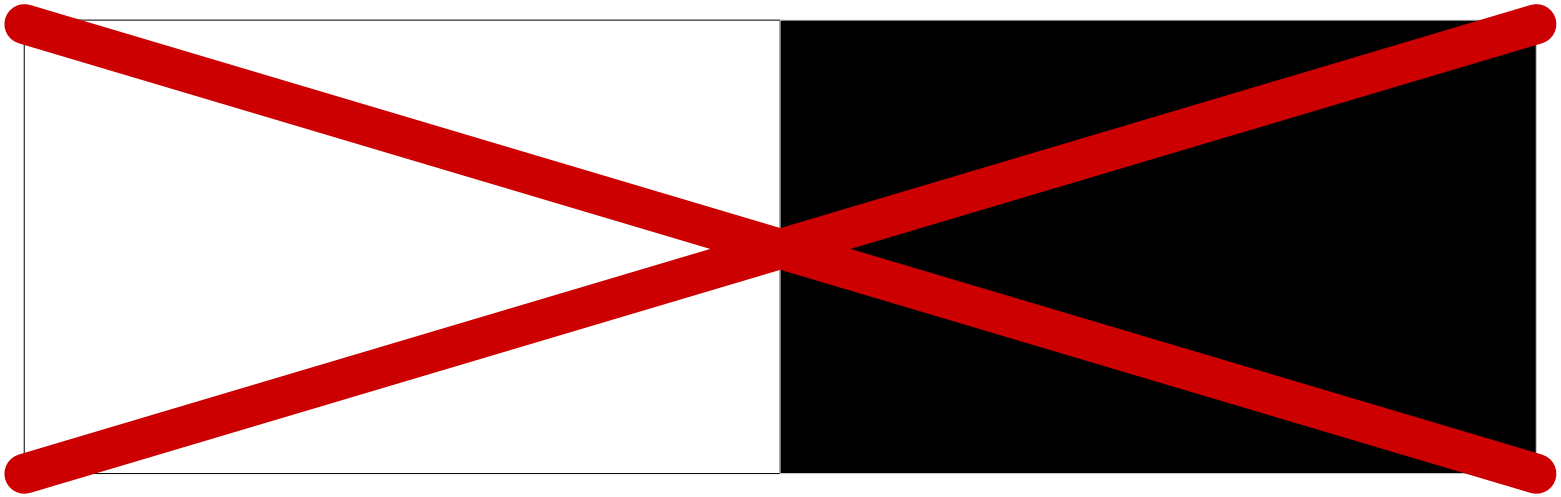
- If the copyright is held by a single entity, a possible move is dual licensing:
  - Companies can either use the free version and have to share their development or they pay and can derive proprietary products.
  - Examples (at some time): MySQL, Qt, Asterisk, Berkeley DB

So Open Source is not a process?



(Abb. Dirk Riehle, FOSS course)

Warning: Processes vary *enormously*. No binary thinking!



Questions

- What is Open Source SW?
- **How important is it?**
- Who builds it? Why?
- What is 'value'?
  Who is the 'customer'?
- How does self-organization work?
  - Basic infrastructure
  - Typical process
  - Leadership
  - Process innovation patterns
- How does quality assurance work?
- Is this agile?
  Is it modern view?
- Is an open process useful *within* companies?
  - Inner Source

How important is it?: Which software is open source?

Freie Universität Berlin

- OSS dominant: Infrastructure software
  - Operating systems
    - Android, *Linux, *BSD, FreeRTOS, etc. etc. (usage statistics)
  - Programming language implementations:
    - C/C++, Java, JavaScript, PHP, Python, R, Ruby, etc.
  - DBMS:
    - MySQL/MariaDB, PostgreSQL, SQlite, most noSQL DBMSs
  - Web servers:
    - Apache httpd, nginx
  - Web browsers:
    - Chrome, Firefox
  - Thousands of libraries and frameworks

etc.

- OSS relevant: Vertical application software
  - https://www.getapp.com
  - CRM systems
  - ERP systems
    - iDempiere, OFBiz, Openbravo, Odoo
  - Finance/accounting
  - Health information systems
  - HR systems

etc.

LOTS!

see also *usage* of: CMS, browsers, languages, …

Questions

- What is Open Source SW?
- How important is it?
- **Who builds it? Why?**
- What is 'value'?
  Who is the 'customer'?
- How does self-organization work?
  - Basic infrastructure
  - Typical process
  - Leadership
  - Process innovation patterns
- How does quality assurance work?
- Is this agile?
  Is it modern view?
- Is an open process useful *within* companies?
  - Inner Source

OSS economical-view success factors: Value of participating in OSS projects

Freie Universität Berlin

- Value **for individuals**:
  - Joy, zealotry
    - [Raymond_HNoo]
  - Solving one's own problem
    - [Raymond_CathBazaar]
  - Increasing one's reputation
    - in hacker's gift culture [Raymond_HNoo]
    - in an exchange culture [CroWeiHow12]
    - in particular freelancers
      - Strong public OSS contributions are *"the ultimate referral"*
  - Earning money
    - Many/most OSS participants are paid by a company [CroWeiHow12]
- A heterogeneous set of motivations
- Consequence for self-organization?
  - Difficult!
    - ➔ part 2
  - Lacking the joint-goal background of a single company

OSS economical-view success factors: Value of participating in OSS projects

Freie Universität Berlin

Is there a catch like this?:

- Value **for companies**:
  - Many sponsor a project almost completely
  - How is this even possible??
    - Where is the catch?
    - [Raymond_MagicCauldron] explains several possible reasons
    - see next few slides:

OSS economical-view success factors: Free Riding

Freie Universität Berlin

- A physical good, when available for free, can easily be overused and hence damaged or destroyed ("free riding")
  - "Tragedy of the Commons" (Lloyd 1833, Hardin 1968)
- However, an intellectual good such as software may even gain from being available for free [HipKro03] :
  - Software/ideas are not damaged when used by more people
  - Market share increases and thus quality improvements become more likely: It is sufficient that any one person sees making an improvement as rewarding (for himself/herself)
  - That all others get the same improvement for free is irrelevant
    - It is impossible to realize the potential market value of small improvements (if that exists at all)
    - Rivals are unlikely to profit from the revealing
    - There are things that the free-rider cannot get without participating (fun of writing code, community, learning)
  - To not openly submit might actually cost something: work for re-integrating the improvement with future versions

OSS economical-view success factors: Sales value vs. use value

Freie Universität Berlin

[Raymond_MagicCauldron]

- Economic reasons for closing source:
  - Protecting sales value
  - Denying others the knowledge embedded in the SW
- ➔ Candidates for opening source:
  - All SW that has no sale value (for you!) and that does not contain crucial knowledge
- Ego-centric benefits from opening source:
  1. Get free help from others for maintaining the SW
  2. Possibly get improvements to the SW you would never make yourself
  3. Reputation
- Opening source reduces sale value, but increases use value
  - There are multiple situations when this is economically advisable:

OSS economical-view success factors: Cost sharing scenario

Freie Universität Berlin

Commercial OSS justification 1:
Cost sharing (The Apache case)

- Assume you need a flexible, reliable, high-performance web server with certain specific features
  - You have three choices:
    1. Buy one (and have vendor risk),
    2. Build your own (and spend a lot of money) or
    3. Join the Apache Group

- Investing into Apache development is actually your cheapest route and hence economically sensible
  - There are now thousands of OSS projects of this type all across the various infrastructure SW domains

(by today, we have almost forgotten that options 1 and 2 even exist!)

OSS economical-view success factors: Maintenance risk reduction scenario

Freie Universität Berlin

Commercial OSS justification 2:
Risk reduction (The Cisco Print Spooler case)

- Assume you have created some useful in-house solution for a problem that is not business-critical
  - e.g. Cisco built a modification of the Unix print spooling service that could re-route "low on toner" print jobs to nearby printers in a global company network, notify administrators, etc.
- You would like to assure you can maintain the solution even if its (few) developers leave your company
  - 2 in Cisco's case
- Your best route is opening source and getting other companies to start using the same solution
  - You may even get further improvements for free
  - Applies to very many projects that once started in just one company

(by today, these typically start as OSS right away!)

OSS economical-view success factors: Market positioning scenario

Freie Universität Berlin

Commercial OSS justification 3:
Loss Leader/Market Positioner (The Mozilla case)

- Loss leader = Lockvogelangebot

- Opening source does not only deny you sale value, but also your competitors (for similar products)
  - This can also help keeping a competitor from achieving quasi-monopoly status
  - or from entering a market in the first place
  - When Netscape opened source of their Mozilla browser it was to deny Microsoft a monopoly with Internet Explorer
    - which would have cut into Netscape's Web Server business via the de-facto control of HTML and HTTP by Microsoft
- A common move for vertical applications
  - *"If you are not the #1 app of a type, open-source it."*
  - Creates chain reactions → several OSS offers appear quickly

OSS economical-view success factors: Widget frosting scenario

Freie Universität Berlin

Commercial OSS justification 4: Widget frosting (The Darwin case)

- If you are building hardware, you need accompanying software but that software does not have sales value itself
  - e.g. device drivers for network/graphics/sound cards, printers
- Opening source brings you the benefits of free help at no loss
  - It is usually impossible anyway to deny your competitors access to any valuable secret in the code
- Example: In 2000, Apple Computers opened the Darwin operating system kernel (the heart of Mac OS X)
  - (Note that OpenDarwin was not successful; later shut down)
- Appears not to be a mainstream behavior

widget = Dingsbums, frosting = Zuckerguss

OSS economical-view success factors: Service reputation scenario

Freie Universität Berlin

Commercial OSS justification 5:
Give away a product to advertise a service

redhat

- Service companies can immensely increase their name recognition and reputation by opening source on internal products
- Examples:
  - Cygnus Solutions — support for GNU tools (1989!)
  - Red Hat, SUSE — Linux support and services
  - Zope Corp. (formerly Digital Creations) — web development
  - Openbravo — ERP software services
- Now a very common model
  - in particular for vertical applications

OSS economical-view success factors: Freemium scenario

Freie Universität Berlin

Commercial OSS justification 6:
Give away a product to
advertise a better product

- Product companies can immensely increase their name recognition and customer trust by opening source on large parts of proprietary products
- Examples:
  - Compiere ERP software
  - Openbravo ERP software
  - Most e-commerce platforms
- Now a common model
  - in particular for vertical applications

OSS sweet spot

High-payoff situations for OSS is SW…

1. …where reliability/stability/scalability are critical
   - → makes a large OSS community particularly helpful
2. …that establishes or enables a common computing infrastructure
   - → highest use of network effects
3. …whose key methods are part of common engineering knowledge
   - → less reason for going closed source; little sales value to be lost
4. or where we want to deny competitors their sales value or simply want to become known as capable people.

In all these cases, OSS is now very common.

OSS economical-view success factors: User risk reduction effects

Freie Universität Berlin

Bonus:

OSS has benefits for the users that reflect back on the supplier:

- Reduced vendor lock-in
- Reduced risk if vendor goes out of business
- Improved transparency of product (quality, security)
- Improved visibility of future developments

So being open source is itself an important feature.

Open-sourcing proprietary SW: How to do it

Freie Universität Berlin

We may want to open-source our software.

- How to decide whether and what?
  - Linåker, Munir, Wnuk, Mols: "*Motivating the contributions: An Open Innovation perspective on what to share as Open Source Software*". Journal of Systems and Software, 2018
    - See also http://linaker.se/2017/11/23/what-to-share-as-open-source/
- How to carry it out?
  - https://opensource.guide/starting-a-project/
  - https://opensource.guide/building-community/

Questions

- What is Open Source SW?
- How important is it?
- Who builds it? Why?
- **What is 'value'?**
  Who is the 'customer'?
- How does self-organization work?
  - Basic infrastructure
  - Typical process
  - Leadership
  - Process innovation patterns
- How does quality assurance work?
- Is this agile?
  Is it modern view?
- Is an open process useful *within* companies?
  - Inner Source

What is 'value'?: 1. "Real" Open Source

- There are no customers, only users.
- [Raymond_CathBazaar] postulates:
  - *"1. Every good work of software starts by scratching a developer's personal itch."*
    - One reason why OSS is so far strongest in infrastructure SW
  - *"2. Good programmers know what to write."*
  - *"5. When you lose interest in a program, your last duty to it is to hand it off to a competent successor."*
- But the itches differ
  - let alone in larger projects:
  - *"the Linux community seemed to resemble a great babbling **bazaar** of differing agendas and approaches […] out of which a coherent and stable system could seemingly emerge only by a succession of miracles."*
    - → **self-organization?**
- Conclusion:
  - An OSS project may not have consensus on 'value'
    - and may somehow also not need it

?

What is 'value'?: 2. Commercial Open Source

There are ordinary customers.

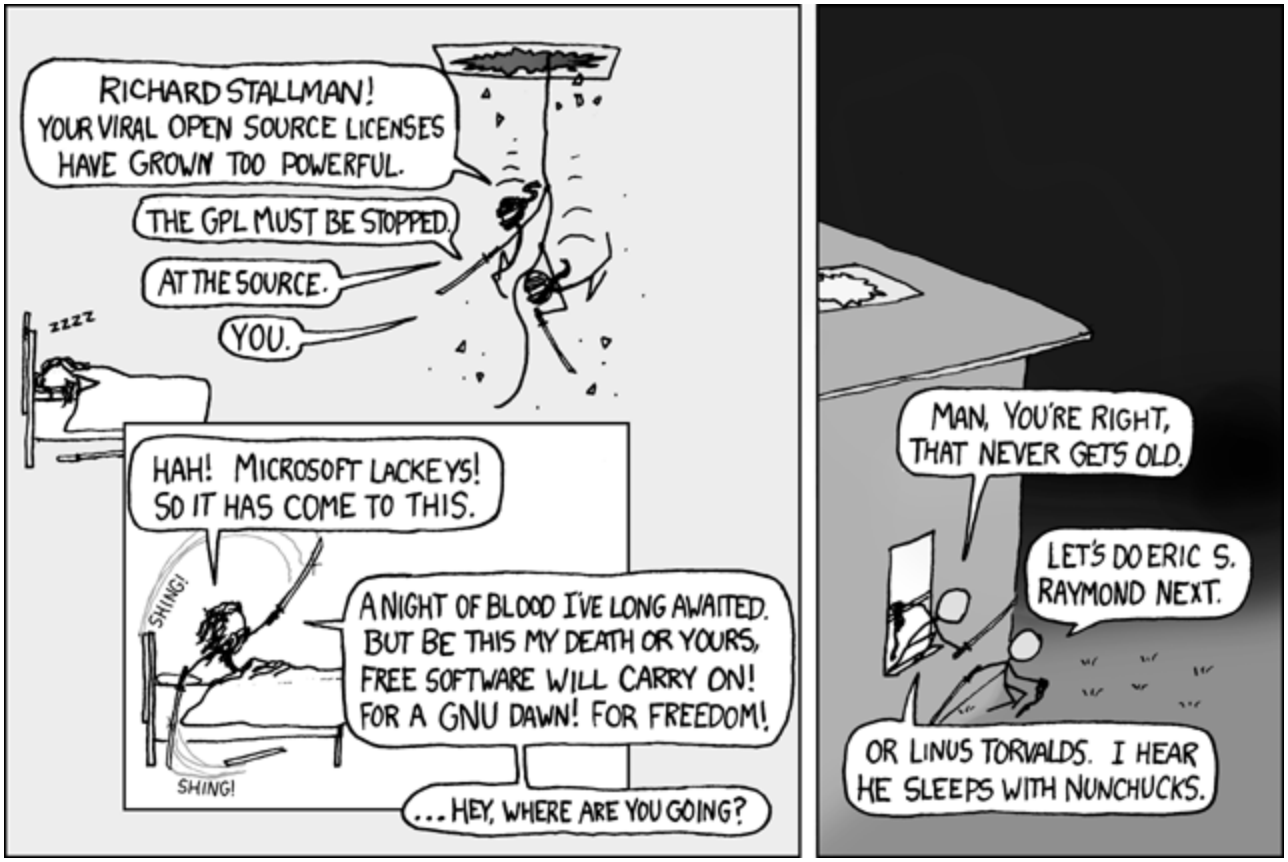
Two sources of contributions:

- The company's own:
  - Service-driven company: Much the same as for normal agile development
  - Product-driven company: Watch out to keep the base product healthy
    - most investment will be in the for-pay parts of the product
- From external participants:
  - usually only bugfixes
    - supplied "as needed"
  - each a slight increase in value

Summary

- Open Source Software is software licensed under an OSS license
  - can be developed openly ("true" OSS)
  - or by (or even only in) a single company (commercial OSS)
    - or "single-vendor OSS"
- OSS is now dominant in many SW domains
  - in particular infrastructure SW
- Individual developers have a range of motivations
- "Value" is often not obvious in true OSS
  - no customers (only users), no product owner
    - clearer for single-vendor OSS
  - leading to the "bazaar" style of development

Thank you!



https://www.explainxkcd.com/wiki/index.php/225:_Open_Source