Course "Softwareprozesse"

Open Source SW (OSS) Development, part 3

Lutz Prechelt
Freie Universität Berlin, Institut für Informatik

Part 1:
• Licenses
• "True" OSS (also with companies)
• Commercial OSS

Part 2:
• Self-organization
  • Process elements, leadership, process innovation

Part 3:
• 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'?
• 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 quality assurance: Feedback, co-testing

- Using *Linux* and *fetchmail* as case studies, [Raymond CathBazaar](#) formulates success factors for OSS dev:

  - "6. Treating your users as co-developers is your least-hassle route to rapid code improvement and effective debugging"
    - Requires sufficiently technical users (see next slide)
    - \(\rightarrow\) OSS is easier for infrastructure SW than for vertical apps
  
  - "7. Release early. Release often. And listen to your customers."

  - "8. Given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix obvious to someone."
    - "Or, less formally, 'Given enough eyeballs, all bugs are shallow.' I dub this: *Linus's Law*". The Linux kernel is indeed proof that this principle can work.

(sort of a hacker's version of the agile manifesto)
OSS quality assurance: Better defect reports

Why finding and fixing defects is easier with OSS:

- In closed source cases, users and developers use different mental models of the system
  - users: surface phenomena
  - developers: code structure, program state and control flow
- But defect reports stated in terms of surface phenomena are often useless
  - because the failure can often not be reproduced
    - e.g. because the user did not report some important condition
- In contrast, Open Source gives users the chance to report defects directly in terms of problematic program elements
  - For difficult-to-locate defects with multiple symptoms or multiple different paths from symptom to defect, it is useful if many people attempt to find a path:
    One will stumble over a simple path even if most will fail.
OSS quality assurance: Collective design

• "It is not only debugging that is parallelizable; development and (to a perhaps surprising extent) exploration of design space is, too."

Preconditions for founding a successful OSS project:

• "It is absolutely critical that the coordinator be able to recognize good design ideas from others"
  • But you need not have those ideas yourself (Linux is an example)
• "A bazaar project coordinator or leader must have good people and communications skills."

• But then, "many heads are inevitably better than one."
OSS quality assurance: Summary

- One big advantage of OSS over closed source is the large number of contributors it makes possible
  - This helps in many dimensions:
    - Development speed (time-to-maturation)
    - Requirements and usefulness
    - Correctness, design quality

- A second important factor is developer self-selection combined with meritocratic developer selection
  - developers are motivated; only competent ones will be accepted

- A third is release planning without deadlines
  - or alternatively sometimes planning with variable feature sets

- Does this work well? Yes:
OSS quality assurance: Case study Apache httpd

  - study compares to several closed-source industrial SW projects

- Apache httpd is among the top 3 webservers since 20 years
  - Evolved from NCSA httpd server's maintenance collaboration
    - an early OSS 2.0 project
  - Highly stable and function-rich Web Server
    - Plug-in architecture with hundreds of extensions ("modules")
  - Core team size about 60 people, democratic process
    - There were 8/12/12/25 members in 1995/1996/1998/2000
    - Current members vote on acceptance of new members after about 6 months of contributions
  - Founding project of the Apache SW foundation
    - http://www.apache.org now >350 different projects (2021-11)
Case study: Apache httpd
Market share (Netcraft web server survey)
Case study: Apache httpd
Team size & work distribution

The size of the Apache development community 1996-1998:

- Apache core team had **12** members during this time
- Overall, almost **400** people contributed code
- **3060** people submitted the 3975 problem reports
  - 458 of them submitted the 591 that lead to one or more changes

Magnitude hypothesis for successful OSS projects:
- if core developers := 1 then developers=10, bugreporters=100

How widely was work distributed among people?

- The top 15 developers (out of 388!) contributed 83% of the change transactions, 88% of the added lines, and 91% of the deleted lines
  - (see graph on next slide)
  - i.e. by far most people make few and small changes only
Case study: Apache httpd
Team size & work distribution (2)

- Distribution of number and size of contributions over people
  - most pronounced for new code: there are 4 developers per 100 non-PR changes, but 26 per 100 PR changes
    - PR: problem report

Apache

Commercial projects A, B
Case study: Apache httpd  
Team size & work distribution (3)

Table 1. Statistics on Apache and five commercial projects.

<table>
<thead>
<tr>
<th></th>
<th>MRs (K)</th>
<th>Delta (K)</th>
<th>Lines added (K)</th>
<th>Years</th>
<th>Developers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache</td>
<td>6</td>
<td>18</td>
<td>220</td>
<td>3</td>
<td>388</td>
</tr>
<tr>
<td>A</td>
<td>3.3</td>
<td>129</td>
<td>5,000</td>
<td>3</td>
<td>101</td>
</tr>
<tr>
<td>B</td>
<td>2.5</td>
<td>18</td>
<td>1,000</td>
<td>1.5</td>
<td>91</td>
</tr>
<tr>
<td>C</td>
<td>1.1</td>
<td>2.8</td>
<td>81</td>
<td>1.3</td>
<td>17</td>
</tr>
<tr>
<td>D</td>
<td>0.2</td>
<td>0.7</td>
<td>21</td>
<td>1.7</td>
<td>8</td>
</tr>
<tr>
<td>E</td>
<td>0.7</td>
<td>2.4</td>
<td>90</td>
<td>1.5</td>
<td>16</td>
</tr>
</tbody>
</table>

domain. Project A is code for a wireless base station, project B is a port of legacy code for an optical network element, and projects C, D, and E represent various applications for operations, administration, and
Case study: Apache httpd
Resulting software quality

<table>
<thead>
<tr>
<th>Measure</th>
<th>Apache</th>
<th>A</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Post-release Defects/KLOCA</td>
<td>2.64</td>
<td>0.11</td>
<td>0.1</td>
<td>0.7</td>
<td>0.1</td>
</tr>
<tr>
<td>Post-release Defects/KDelta</td>
<td>40.8</td>
<td>4.3</td>
<td>14</td>
<td>28</td>
<td>10</td>
</tr>
<tr>
<td>Post-feature test Defects/KLOCA</td>
<td>2.64</td>
<td>*</td>
<td>5.7</td>
<td>6.0</td>
<td>6.9</td>
</tr>
<tr>
<td>Post-feature test Defects/KDelta</td>
<td>40.8</td>
<td>*</td>
<td>164</td>
<td>196</td>
<td>256</td>
</tr>
</tbody>
</table>

- Note that Apache is much higher-used than A, C, D, E
  - so the numbers will represent a higher fraction of all defects

No system-testing is common in OSS

Avoids favoring bloated code
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

Lutz Prechelt, prechelt@inf.fu-berlin.de
Is OSS agile?
Is it modern-view?

Is "true" OSS agile?
- Yes:
  - "Working software over comprehensive documentation"
  - "Customer collaboration over contract negotiation"
  - Highly iterative
  - Very little planning
- No:
  - Less mention of people as people
    - "Individuals & interactions over processes and tools"?

Commercial OSS will be more "normally" agile with respect to customers and planning.

Is it modern view?
- Let's consider each attribute of
  - classical view
  - modern view
Summary of classical view vs. modern view

**Epistem. stance:**
- rationalism, positivism
- empiricism, interpretivism

**Means of description:**
- activities, artifacts, roles
- fewer ditto, practices, principles

**Ideals:**
- engineering ideals:
  - planning,
  - getting it right at once
- humanist preferences:
  - reacting,
  - iterating,
  - strive to produce high value

**Central ideas:**
- specification
- collaboration & iteration

**View on people:**
- exchangeable resources
- drivers and purpose of process
Is OSS agile?
Is it modern-view?

Conclusion for "true" OSS:

- Classical view?
  - Absolutely not.
- Modern view?
  - Yes, in it's own way.
- Agile?
  - No paying customer in "true" OSS
  - Even less planning
  - Much less emphasis on people
    - Where agile folks are people-centric, OSS folks tend to be techno-centric

Commercial OSS may be more akin to agile development
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
"I want to get paid to make OSS"

- Method 1: Join an OSS organization
  - Mozilla, Wikimedia?

- Method 2: Join an organization that has some full-time OSS participants
  - There are many, large and small.

- Method 3: Join an organization that has part-time OSS participants
  - There are very many.

- Method 4: Join an organization that tolerates some OSS work
  - There are very very many. No clear separation from method 3.

- Method 5: Free-lance and do OSS for reputation
  - Nice route for high-skill freedom lovers.

Lutz Prechelt, prechelt@inf.fu-berlin.de
Using OSS processes for closed-source SW: "Inner Source"

[StoFit15]

- Companies struggle with distributed work and with SW reuse
  - but OSS is very successful at both.
- But companies cannot open-source all their SW.
- Thus, companies now attempt to establish OSS-ish work modes internally, sometimes with success.
- Requirements:
  - advocacy from top management
  - suitable infrastructure, common tools
  - suitable seed products
    - with sufficient modularity!
  - creating enough transparency
  - successful self-organization
  - adopting incremental OSS development and QA styles

Lutz Prechelt, prechelt@inf.fu-berlin.de
Elements of successful Inner Source development

- Capraro, Riehle: "*Inner Source Definition, Benefits, and Challenges*, 2016 (a literature survey)
Inner Source: Where? What?

- Literature reports Inner Source use at (in order of frequency)
  - HP (7x), Philips (7x), Global Soft, Lucent, Nokia (3x), IBM, DoD, DTE Energy, Google, Microsoft, SAP, DLR, Ericsson, Kitware, NeoPost, Rolls Royce.

<table>
<thead>
<tr>
<th>Dimensions of IS Programs</th>
<th>Prevalence</th>
<th>Degree of Self-Organization</th>
<th>Internal Economics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prevalence</td>
<td>Universal</td>
<td>Free Task Choice &amp;</td>
<td>Local-Library</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Free Component Choice</td>
<td></td>
</tr>
<tr>
<td>Selective</td>
<td>Selective</td>
<td>Assigned Tasks &amp;</td>
<td>Private-Market</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Assigned Components</td>
<td>?</td>
</tr>
<tr>
<td>Project-Specific</td>
<td>Project-Specific</td>
<td>Assigned Tasks &amp; Assigned Components</td>
<td></td>
</tr>
</tbody>
</table>
**Inner Source: Where? What?**

- Can have very different scope and purpose

<table>
<thead>
<tr>
<th>Dimensions of IS Projects</th>
<th>Governance</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Single Organizational Unit</td>
<td>Exploration-Oriented</td>
</tr>
<tr>
<td></td>
<td>Multiple Organizational Units</td>
<td>Utility-Oriented</td>
</tr>
<tr>
<td></td>
<td>All Organizational Units</td>
<td>Service-Oriented</td>
</tr>
</tbody>
</table>

(for innovation)

(often small-scale)

(business-critical)
Inner Source: Benefits

- Effectiveness
  - costs, time-to-market
- Better reuse
  - of competence
  - decoupling providers from reusers
  - relief for providers
- Better quality
  - code quality
  - better ideas
- More flexible use of devs
  - esp. detached ones
- Employee motivation

- Knowledge management
  - community-based learning
  - availability of knowledge
- Overcoming unit boundaries
  - collaboration
  - cost sharing, risk sharing
Inner Source: Challenges

Mismatch with organization:
- Resistance
  - Change in work style
  - Lack of cultural fit
- Selfish interests
  - Fear of resource loss
  - Code ownership mismatch
  - Fear of maintenance effort
  - Individual disadvantages
- Heterogeneity
  - Diverse existing processes
  - Diverse existing tools

Inner source adoption:
- What to inner-source?
- Utilizing openness
  - Too much data, awareness
  - Navigating code of others
- Resenting transparency
  - Security, Intell. Property
  - Being scrutinized
- Control and steering
  - Leadership, incentives
- Contribution process
  - Quality issues
  - Uncontrolled forking

Lutz Prechelt, prechelt@inf.fu-berlin.de
Summary

- OSS quality assurance is based on
  - frequent releases, crowd-ish testing with open source code, meritocratic self-selection and leadership

- OSS development is a modern-view paradigm
  - with less humanist emphasis compared to agile
  - and perhaps no product owner

- Intra-company OSS ("Inner Source") can be useful
  - but may be difficult to introduce
References (for all 3 parts)

  - Covers nearly all FLOSS research until 2008.
  - Summarizes knowledge per topic, for 31 topics.
References (2)

- [FLOSS02] FLOSS project: "Free/Libre and Open Source Software: Survey and Study", 2002
- Michi Henning: "The rise and fall of CORBA", ACM Queue 2006
  - A story how design-by-committee can fail that teaches how and why OSS processes are often successful. Interesting!
Thank you!

https://www.explainxkcd.com/wiki/index.php/1636:_XKCD_Stack