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

Q

OSS quality assurance: Feedback, co-testing

Freie Universität Berlin

- 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)
  - → 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

Freie Universität Berlin

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

Freie Universität Berlin

- *"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

Freie Universität Berlin

- 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:

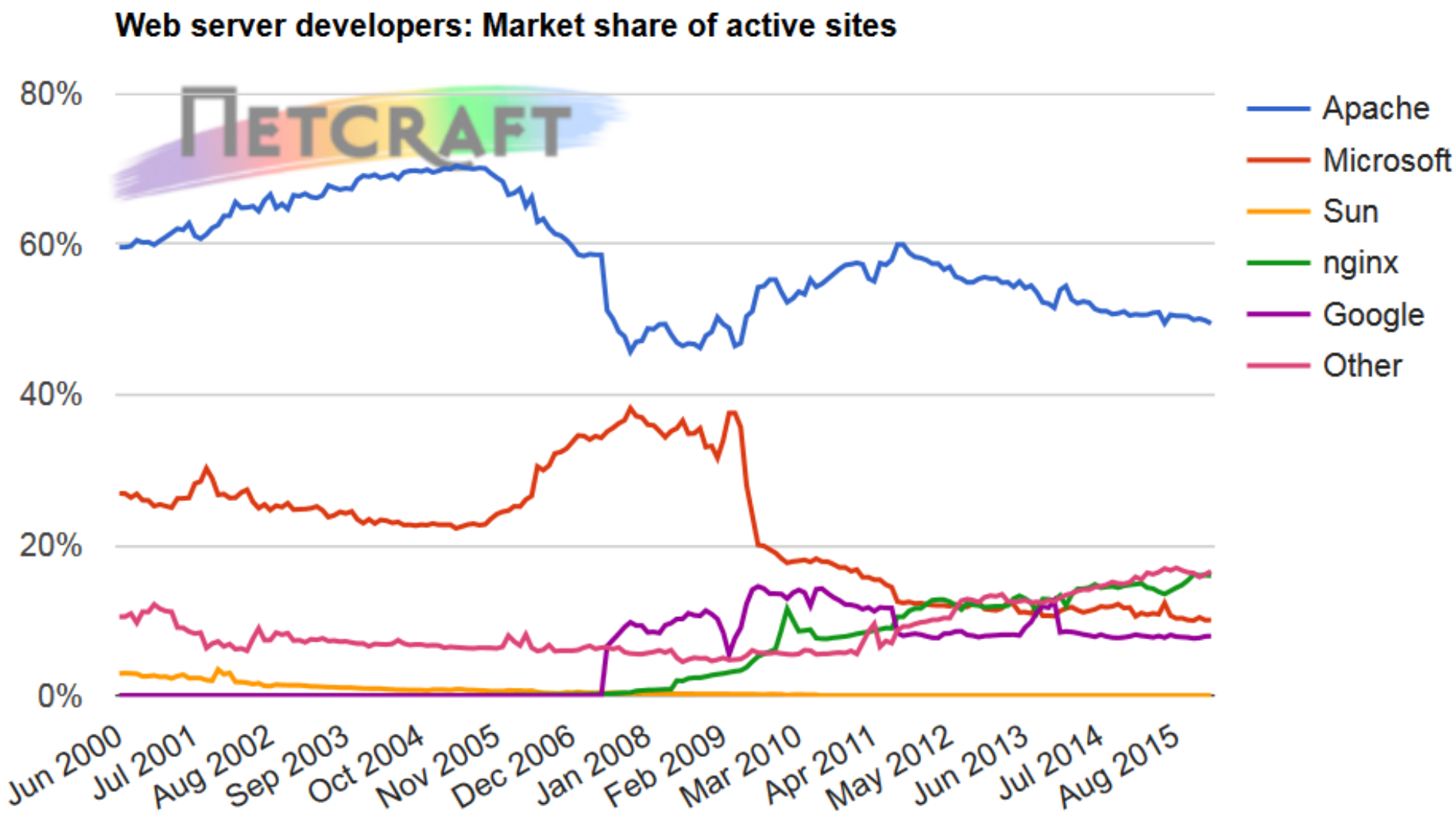
OSS quality assurance: Case study Apache httpd

Freie Universität Berlin

- Audris Mockus, Roy Fielding, James Herbsleb: "*A Case Study of Open Source SW Development: The Apache Server*", ICSE, 2000
  - 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

Case study: Apache httpd
Market share (Netcraft web server survey)



Case study: Apache httpd
Team size & work distribution

Freie Universität Berlin

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 led to one or more changes

Magnitude hypothesis for successful OSS projects:

  - if core developers := 1 then developers=10, bugreporters=100

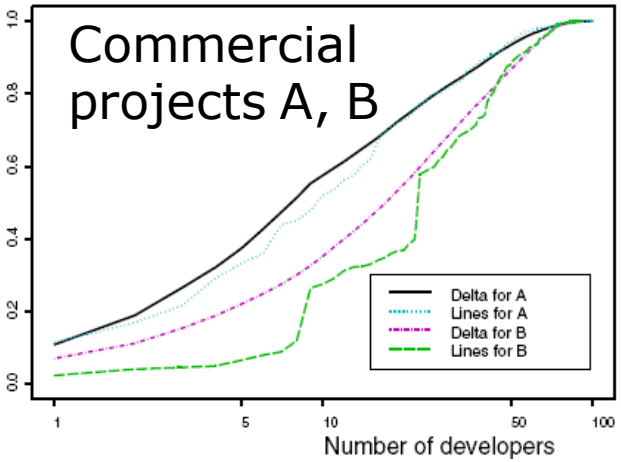
How widely was work distributed among people?

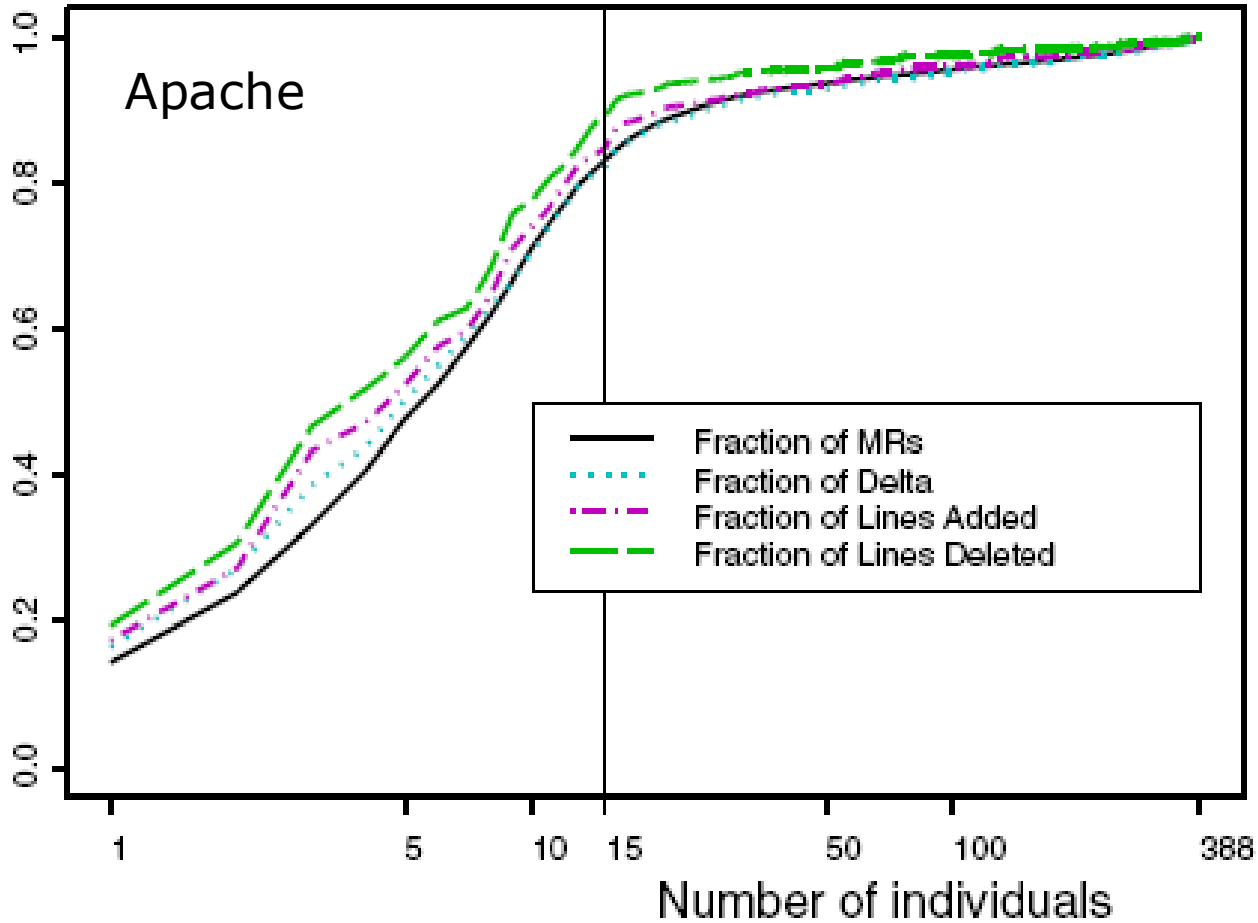
- The top 15 developers (out of 388!) contributed 83% of the change transactions, 88% of the added lines, and 91% of the deleted lines
  - (see graph on next slide)
  - i.e. by far most people make few and small changes only

Case study: Apache httpd
Team size & work distribution (2)

Freie Universität Berlin

- 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





Case study: Apache httpd
Team size & work distribution (3)

Freie Universität Berlin

- **MRs:** number of changes (**m**odific. **r**equest)
- **Delta:** number of files changed

Table 1. *Statistics on Apache and five commercial projects.*

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

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

Measure	Apache	A	C	D	E
Post-release Defects/KLOCA	2.64	0.11	0.1	0.7	0.1
Post-release Defects/KDelta	40.8	4.3	14	28	10
Post-feature test Defects/KLOCA	2.64	*	5.7	6.0	6.9
Post-feature test Defects/KDelta	40.8	*	164	196	256

No system-testing is common in OSS

Avoids favoring bloated code

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

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

Q

Is OSS agile?
Is it modern-view?

Freie Universität Berlin

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

Freie Universität Berlin

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 quality is meeting the specs	humanist preferences: reacting, iterating, strive to produce high value
Central ideas:	
specification	collaboration & iteration
View on people:	
exchangeable resources	drivers and purpose of process

OSS's view is much less explicit!

Is OSS agile?
Is it modern-view?

Freie Universität Berlin

Conclusion for "true" OSS:

- Classical view?
  - Absolutely not.
- Modern view?
  - Yes, in it's own way.
    - But not very explicit
- 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"

Freie Universität Berlin

- Method 1: Join an OSS organization
  - Mozilla, Wikimedia?, ??
- Method 2: Join an organization that has some almost-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 veryvery many. No clear separation from method 3.
- Method 5: Free-lance and do OSS for reputation
  - Nice route for high-skill freedom lovers.

Using OSS processes for closed-source SW: "Inner Source"

Freie Universität Berlin

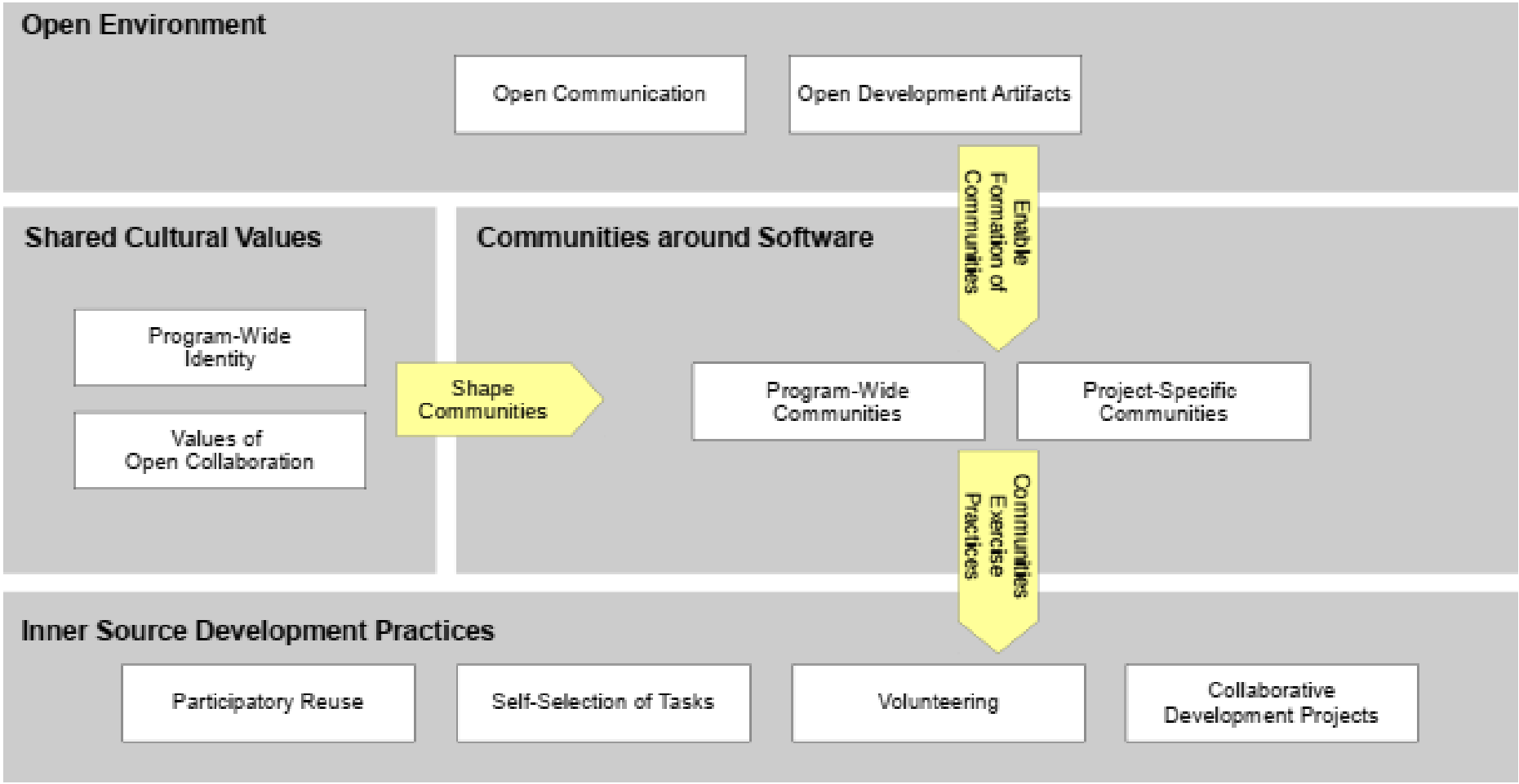
[StoFit15]

- 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

Elements of successful Inner Source development

Freie Universität Berlin

- 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.

Dimensions of IS **Programs**		
Prevalence	Degree of Self-Organization	Internal Economics
Universal	Free Task Choice & Free Component Choice	Local-Library
Selective	Assigned Tasks & Free Component Choice	Private-Market ?
Project-Specific	Assigned Tasks & Assigned Components	

Inner Source: Where? What?

- Can have very different scope and purpose

Dimensions of IS **Projects**		
Governance	Objective	
Single Organizational Unit	Exploration-Oriented	(for innovation)
Multiple Organizational Units	Utility-Oriented	(often small-scale)
All Organizational Units	Service-Oriented	(business-critical)

Inner Source: Benefits

Freie Universität Berlin

Product:

- Effectiveness
  - costs, time-to-market
- Better reuse
  - of competence
  - decoupling providers from reusers
  - relief for providers
- Better quality
  - code quality
  - better ideas

People:

- 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

Freie Universität Berlin

The challenges are often HUGE!

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

Inner Source: Discussion

Freie Universität Berlin

- Would Inner Source work in your company?

- Tendency:
  - only large companies need Inner Source
  - but large companies also have the biggest difficulty getting it to work:
    - cost centers and other incentive structures
    - culture issues

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

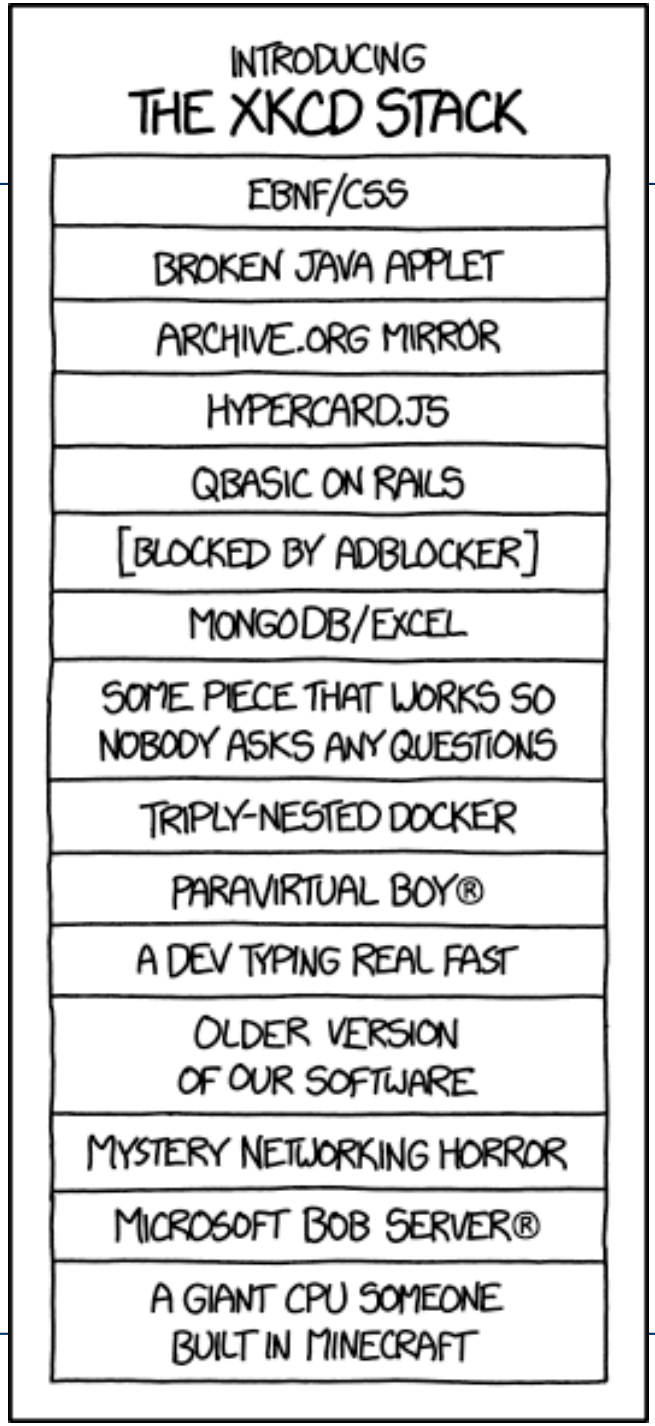
References (for all 3 parts)

- [CroWeiHow12] Kevin Crowston, Kangning Wei, James Howison, Andrea Wiggins: "Free/Libre Open-Source Software Development: What We Know and What We Do Not Know", ACM Computing Surveys 2012
  - Covers nearly all FLOSS research until 2008.
  - Summarizes knowledge per topic, for 31 topics.
- [DenDun06] Denning, Dunham: *"Innovation as language action"*, Communications of the ACM 2006
- [Fitzgerald06] Brian Fitzgerald: "The Transformation of Open Source Software", MIS Quarterly 2006
- [Johnsson01] Kim Johnsson: *"A descriptive process model for OSS development"*, M.Sc. thesis, Univ. of Calgary, 2001
- [StoFit15] K.-J. Stol, B. Fitzgerald: "Inner Source -- Adopting Open Source Development Practices in Organizations: A Tutorial", IEEE Software 2015
- [HowCro14] Howison, Crowston: *"Collaboration Through Open Superposition: A Theory of the Open Source Way"*, *MIS Quarterly 2014*.

References (2)

- [HipKro03] von Hippel, von Krogh: "Open Source Software and the "Private-Collective" Innovation Model: Issues for Organization Science", Organizat'n Science 2003
- [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