Quality Improvement and Release Management in FOSS projects

Martin Michlmayr <tbm@cyrius.com>
University of Cambridge

This work has been funded in part by Intel and Google.

Objectives for today

- What is quality?
- What is quality assurance (QA)?
- FOSS and QA
- The problems of release management
- ... and a possible solution

Quality

Everyone knows, but ...

- Hard to define
- Hard to measure

Definitions of quality:

- Fitness for purpose
- Attributes of quality: efficiency, reliability, usability, extendability, portability, reusability, maintainability

Different aspects:

- User perception
- Developer perception

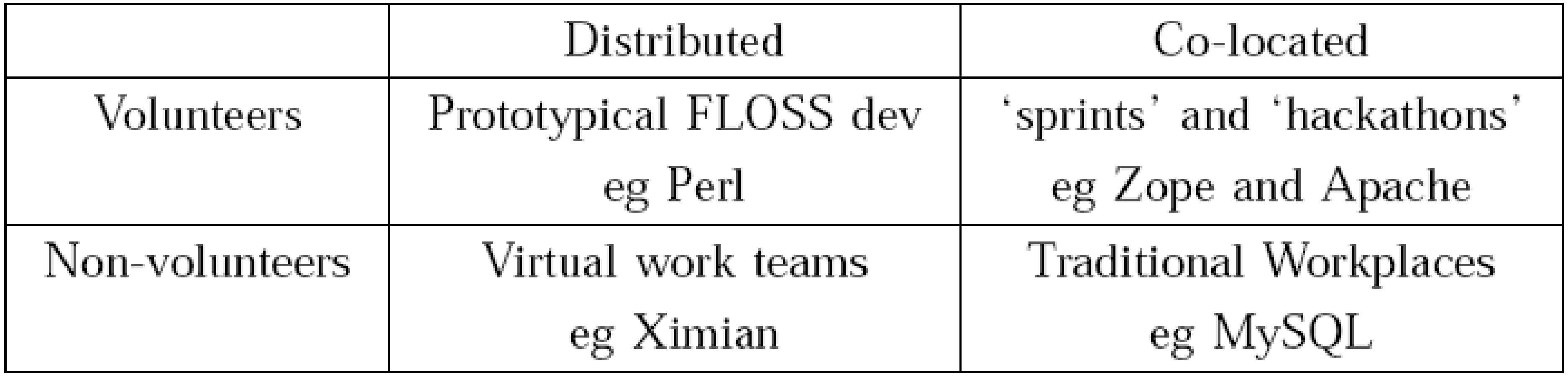
What is Quality Assurance?

Traditional Quality Assurance (QA)

- Does what it should do (meets the specification)
- Does what others do as good or better as others (meets the "Industrial Standard")
- QA begins before the implementation!
  - You cannot "add" quality later
- QA is not (just) testing
- ISO defines QA as all "planned and systematic activities" (to ensure quality)

Quality and FOSS

A four-square depiction of FLOSS organization



	Distributed	Co-located
Volunteers	Prototypical FLOSS dev eg Perl	'sprints' and 'hackathons' eg Zope and Apache
Non-volunteers	Virtual work teams eg Ximian	Traditional Workplaces eg MySQL

(Figure by James
How

We will focus on "typical" (traditional) FLOSS projects:

- Distributed development
- Done by volunteers

Eric S. Raymond (1999): The Cathedral and the Bazaar
Linus' law

Quality and FOSS

- Quality is often high
  - peer review (Cathedral and the Bazaar)
  - World domination

- ... but not always
  - Many small, unsuccessful projects
  - example: SourceForge has over 100,000 projects
  - Big projects have problems too
  - Contrast to QA as "planned and systematic activities"

Interviews: identifying quality issues

Interviews with members of FOSS projects

3 main areas:

- leadership: benevolent dictator, team
- release cycle:
  - "release when it’s ready", time based
  - fast vs slow, development vs user release
  - beta cycle, release candidates
- company involvement

Interviews: underlying topics

Processes and infrastructure

- Communication
- Bug tracking systems
- Contributing to the project

Success

- What is success?
- Relation of success and quality?
  - Success: more volunteers
  - Contribute, Improve
  - More quality
- Cathedral to bazaar

Release Management

- Scope: small vs big projects
- Small projects: often don’t know much about release management and user requirements.
- Large projects: Coordination is hard.

Problems of Release Management

Examples:

- Debian: "we release when it’s ready" as a way of saying "never"
- The Linux kernel: from the "Linux model" (odd/even) to... chaos(?)
- Mutt: stable versions severely out of date (until recently)

Feature-based Releases

- In large projects: there are always more features; you can always improve something.
- Planning of features is hard (cf. volunteer nature)
- Freezes announced out of the blue -> "thundering herd of patches" problem (Ted Ts’o)
- Project is late, people think they have time to cram in their features: project is even later (repeat)

Time-based Releases

- Relatively novel concept
- GNOME as the successful example (1.x vs 2.x cycle)
- The idea: don’t talk about features, talk about time.
- Give a detailed plan (time line), give people deadlines.
- Review and possible revert functionality that is not ready!

Reasons for the Time-Based Model

- In large project, there is always some development (bug fixes, minor features)
- FOSS projects don’t need to justify new releases as much as companies
- You get your features/fixes out quicker; get quicker/more (useful) feedback.
- You can still talk about features; just not commit to anything.

Possible Incentives

- End-users: get fixes periodically, each version is a gradual increase.
- Companies: predictable releases.
- Developers: development speed and motivation increases because of feedback, coordination is easier

Lessons learned

- Quality in FOSS in an important area
- Some projects have realized this
    - Debian: http://qa.debian.org/
    - KDE: http://quality.kde.org/
    - GNOME: http://developer.gnome.org/projects/bugsquad/
- It is important to think about quality and to
    - Find ways to measure quality
    - Find ways to improve quality
    - Find ways to automate quality
    - Document quality practices
- Researchers and FOSS developers can work together