Quality Issues in Free Software Projects

1 Findings of the Interviews

Semi-structured interviews with seven open source developers have been carried out at open source conferences. These developers were from a random selection of projects of different nature. Due to the limited number of interviews, the following findings are not necessarily generalizable to the whole open source community. However, they give a good indication of general themes related to quality practices and issues.

1.1 Interview Questions

- name of project
- size of project
- nature of project

1. Are there any quality issues in your project? If so, what are they and how do you deal with them?
2. What techniques can be applied in open source projects to ensure quality?
3. Are there any unresolved quality issues? If so, what are they and why are they unresolved?
4. Which development model and life cycle do you follow?
5. What kind of coordination and management are there in your project, and what kind of management style is used?
6. What kind of facilities do you have to ensure quality?
7. Would you say that your project is successful? How do you measure or perceive success?
8. How would you compare the quality of open source and proprietary software? When might the quality in one be higher than in the other?

1.2 Current Practices and Techniques

The following practices and techniques have been observed, but few projects make use of all of these practices.

1.2.1 Infrastructure

- Bug tracking systems (e.g. Bugzilla): they are used to capture feedback from users. They are used to store both actual bug reports as well as feature requests.
- Version control systems (CVS, SVN, Arch): they are used to allow multiple people to work on the same code basis and keep track of who makes which changes.
- Automatic builds (e.g. tinderbox): they make sure that the newest code in the version control system still builds. Those builds can be done on a number of different hardware or software environments.

1.2.2 Processes

- Joining: projects require prospective members to follow specific, mostly undocumented procedures in order to join a project.
- Release: different processed are employed in release management, such as freezes (feature freeze, string freeze).
- Branches: they are used to differentiate between versions of a program (for example, a stable and a development branch).
- Peer review: used to obtain feedback from peers and find bugs. Some projects have very rigorous peer review procedures while most projects assume that someone will look at their code.
- Testing (checklists): there is not much automatic testing but some projects have checklists which they use before a release to make sure the core functionality works.

1.2.3 Documentation

- Coding styles: documentation aimed at developers which describes the style which should be used to code.
- Code commit: documentation which describes when and who can make changes in a project's version control system.

1.3 Outstanding Quality Issues

- Ports: some projects have problems keeping different ports of their software to obscure hardware or operating systems in sync. This is because a volunteer contributes a port and then vanishes but the project maintainers do not have access to that environment and therefore cannot test changes.
- Configuration management: it is impossible to testing all possible combinations and it is often hard to find testers.
- Users do not know how to report bugs: there are many duplicates and bad bug reports. The problem is that documentation about writing bug reports will not help since users are not interested in reading it.
- Automatic tests: most projects do not have any automatic tests because they are apparently boring to write. Furthermore, people who write the code should not write their own test suites but there are few volunteers who test other people's code and write test suites.
- Security updates: in many cases they are made in a timely manner but often they are not.
- Attracting volunteers: current developers are busy and often cannot focus on new development because they have to deal with users and bug reports. Projects try to attract new volunteer to help with bugs, coding and testing. However, this does not seem to be easy. Furthermore, it is hard to ensure that they are competent. There is almost no time to mentor new members.

- Lack of documentation: there is little documentation, regarding both user and developer documentation.