

Development Good Practise using Open Source Software

Neil Blakey-Milner
nbm@mithrandr.moria.org

<http://mithrandr.moria.org/>

<http://mithrandr.moria.org/writings/development-good-practise-using-oss/>

Industry Best Practise?

- A misnomer
- Tools are less important than principles
- Best practise depends on your exact situation
- Open Source is providing best practise for some

My context

- Independent Online
 - Our tools built exclusively with Open Source Software
 - Interfacing with proprietary software in the group
 - Replacing some proprietary software
- 170k lines of code, 4 developers
- Nightmare to manage
- These principles helped us turn things around

Version control

- Most important!
- Maintain history of versions of files
 - For comparison
 - Reverting
- Process:
 - Check in new files
 - Check out any version of the file

Advanced Version Control

- Display differences
 - Between your changes and the original version
 - Between two versions in version control
- Branches (Lines of Development)
- Automated actions on check-in

Version Control options

- CVS (old faithful)
- Arch
- Subversion

Automated builds

- One command
 - to build binaries
 - to install your software
 - to build packages
 - anything else you need to do
- One less excuse not to build before check-in

When to build

- Time-based
 - daily
 - hourly
 - at lunch-time
- Action-based
 - At check-in

How to build

- Build binaries, packages, &c.
 - for each of your different platforms
 - with different build options
- Tinderbox (time-based builds)
- Buildbot (build-on-checkin)
- Both offer “Blamelists” to discover who changed stuff since last build.

Tests

- Heresy: rather test and bugtrack!
- Why you should test is another presentation!
- Types of tests:
 - functional, component, regression
 - performance, load, stress
 - integration, manual
- Schools of thought:
 - Write tests when you fix bugs (regression)
 - Write tests after you write code
 - Write tests before you write code

Testing best practise

- What your team can afford to invest into testing, based on their experience.
- At least do regression tests!
- Examine payoffs:
 - Low initial payoff with non-bug-driven testing
 - Increased payoff as experience grows
 - Realise when returns are diminishing

How to test

- Developers can test before check-in
 - Not a clean environment
- Pristine environment from CVS
- Time-based or action-based

Bug tracking

- Manual process doesn't scale.
- Problems:
 - Problems forgotten by developers
 - Problems aren't easily prioritised
 - Problems that won't be solved recur
 - Solutions to problems are forgotten
 - Non-system problems recur and waste debugging time

Qualities of good bug tracking

- Low barriers to use
 - Good interfaces for users and developers
 - Capture only essential information
- Effective
 - Capture all essential information
 - Provide feedback/statistics on problem areas

Essential information

- Requestor
- Description of bug
- Category of bug:
 - Product
 - Service
 - Business area
- Priority
- State
- Owner