Open Source Software Development Process

Yongsen MA
Shanghai Jiao Tong University
E-mail: mayongsen@gmail.com

I. Introduction

Aaron Koblin: Artfully visualizing our humanity

Why free software has poor usability, and how to improve it

A. motivating, joining, participating and contributing

1) acquire: knowledge, experience, opportunities; backup, platform
2) participate: happiness, communication
3) contribute: freedom, trustworthy

1) developer
2) user(evaluation)

1) public
2) private

B. modeling, examination, investigation

1) individuals
2) groups
3) organizations

1) operate systems
2) web
3) application
4) network

1) contribute:

2) process: stable, scalable
3) acquire: software, individuals, groups

1) graph theory
2) multiproject
3) interdependent

II. Open Source Project

A. Components

1) Home Page
2) Code Repository
3) Mailing List
4) Bug Tracking System
5) Wiki

B. Participating

1) Starting
2) Discussion
   - Subscribe Mailing List
   - Take part in News Group
   - Participate in Conference
3) **Programming and Debugging**
   - Consummate documents
   - Running test codes
   - Report Bugs
   - Submit patch
4) Improving

C. Developing

1) Creating a Repository
2) Making Changes

  - Adding Files
  - Committing Changes
  - Files Status and Differences
  - Managing Files

3) **Managing Branches**
  - Creating Branches
  - Merging Branches
  - Handling Conflicts
  - Deleting and renaming branches

4) Handling Releases

III. Debugging

For example, how are crash reports handled? How are bug reports handled? How are bugs classified and confirmed?

A. Basic Debugging

B. Functional Debugging

IV. Branching

How are the assignments to individual developers made? How to merge code changes in Git? How are code inconsistency handled? In each step of the process, have you identified any software engineering issues which have rooms for improvements?

A. Branching

- Creating Branching

  ```
  git branch new
  ```

  1) Test Changing
  2) Add new functionality
  3) Fix bugs
- Merging Branching
  1) straight merge

2) squashed commits
3) cherry picking

- Handling Conflicts

1) Manual
2) Tools

```
git mergetool
```

- Deleting and renaming branches

Local Use Cases

- Pulling Updates
- Making Patches

1) Test Changing
2) Add new functionality
3) Fix bugs

- Merging Patches

1) straight merge
2) squashed commits
3) cherry picking

- Finding a Commit
- Cherry Picking
- Reverting a Commit
- Resolving Merges
- Rebasing Local Changes

B. Sending Changes Upstream

- Generate and send patches via email
  - Most developers send patches to a maintainer or list
  - Highly visible public review of patches on mail list
- Maintainer pulls updates from a downstream developer
  - Maintainer can directly pull from your published repository

  - Initiated by upstream maintainer
- Developer pushes updates to an upstream maintainer
  - Some developers have write permissions on an upstream repository
  - Initiated by downstream developer

C. Merging

Merge: Combine directory and file contents from separate sources to yield one combined result.

- Sources for merges are local branches
- Merges always occur in the current, checked-out branch
- A complete merge ends with a new commit

Git uses several merge heuristics:

- Several merge strategies: resolve, recursive, octopus, ours
- Techniques: fastforward, threeway

References

[1] A. Bonaccorsi and C. Rossi. Why open source software can succeed. *Research Policy*, 32(7):1243 – 1258, 2003.

[2] S. Chacon, J. Hamano, and S. Pearce. *Pro Git*, volume 288. Apress, 2009.

[3] G. Hertel, S. Niedner, and S. Herrmann. Motivation of software developers in open source projects: an internet-based survey of contributors to the linux kernel. *Research Policy*, 32(7):1159 – 1177, 2003.

[4] B. Kernighan and R. Pike. *The practice of programming*. Addison-Wesley Professional, 1999.

[5] B. Kogut and A. Metiu. Opensource software development and distributed innovation. *Oxford Review of Economic Policy*, 17(2):248–264, 2001.

[6] W. Scacchi, J. Feller, B. Fitzgerald, S. Hissam, and K. Lakhani. Understanding free/open source software development processes. *Software Process: Improvement and Practice*, 11(2):95–105, 2006.

[7] G. von Krogh and E. von Hippel. Special issue on open source software development. *Research Policy*, 32(7):1149 – 1157, 2003.

[8] C. Yilmaz, A. M. Memon, A. Porter, A. S. Krishna, D. C. Schmidt, and A. Gokhale. Techniques and processes for improving the quality and performance of open-source software. In *Software Process: Improvement and Practice*. John Wiley & Sons, 2006.