

Reefmerge

Grzegorz Konefal

September 10, 2017

Abstract

This document describes an idea for Master's thesis which is focused on merging problem in Python's git repositories. Conflicts are more often especially for big and old projects. A lot of conflicts are simply solvable and it is possible to automatically determine what solution should be applied. My work is focused on such cases in Python's repositories.

Problem

I have provided an investigation how big the problem is in open source Python repositories, which are known as big and mature. First I have checked how many merge commits in those repositories had conflicts that were resolved manually by developers. It turns out that averagely 6% of merge commits had parents that were conflicting (this number differ from 2% to 12% for different repositories). Of course the number doesn't show how many developer work was put on resolving conflicts when rebasing or applying patches - unfortunately it is impossible to measure basing on git history, because rebasing or applying patches leaves no mark in repository's history.

Solution

The main idea is to write a tool that automatically resolves git conflicts in Python code, no matter if conflict occurred while merging, rebasing, or applying a patch. I expect the tool usage will be most possibly easy, so this functionality will be injected to git as a plugin - a merge driver.

Lets imagine that programmer A added a function `foo_a()` at the end of file `code.py`. At the same time programmer B implemented a function `foo_b()` at the end of the same file, and pushed his code onto master branch. Then the programmer A wants to merge branch master onto his developing branch and encounter a conflict. In fact, there are implementations of two different functions with different names, so what is the problem? Git does not recognize Python syntax and semantics and doesn't understand that this conflict can be simply resolved just by placing `foo_b()` implementation just after `foo_a()`. We need a tool that understands Python and can resolve such conflicts automatically.

The best option that we could imagine is to somehow tell the git how to resolve conflicts in Python code. Fortunately, git allows to select custom merging driver. Reefmerge will be a merge driver that understands Python and resolves conflicts in Python's code automatically.

Of course, there are conflicts that cannot be resolved automatically, for example when someone introduces syntax errors to code. But I expect that this tool will work in most of cases.

Alternatives

There are already available some tools that can help programmer to fix conflicts. Part of them really attempts to resolve conflicts, the other part deliver some features that makes fixing conflicts simpler, but doesn't fix conflicts at all. Reefmerge will attempt to take place of programmer in resolving conflicts in Python code. But now let's have a look what programmers already have.

git-rerere

git-rerere is a git tool that is able to remember already fixed conflicts and its resolutions. If git-rerere remembers some resolution and identical conflict occurred, git-rerere can automatically resolve this conflict in the same way as it was done previously.

Python conflicts also may reoccur, but if it is a problem like in the previous chapter, git-rerere will be useless while functions names or implementations differ. git-rerere is prepared for fixing the same conflict many times rather than smartly resolving new conflicts.

mergetools and difftools

Conflicts are hard to resolve quickly manually, hence to increase speed of manual conflict resolution many mergetools were created. These are tools that during manual conflict resolving allow to see e.g. two or three pretty windows and differences between them, and gives the ability to quickly resolve these conflicts e.g. by selecting by mouse which version will be applied.

Mergetools help to increase efficiency of manual conflicts resolving, but they do not attempt to fix anything by their own. Difftools are similar to mergetools - but they just help to see diffs more clearly.