Extended Methodology and Data Sources for Review early, Review often: A case study of Apache peer review

Peter C. Rigby
University of Victoria
BC, Canada
pcr@uvic.ca

Daniel M. German
University of Victoria
BC, Canada
dmg@uvic.ca

Margaret-Anne Storey
University of Victoria
BC, Canada
mstorey@uvic.ca

1. METHODOLOGY AND DATA SOURCES

This section contains an elaboration of Section 4.1 in in our FSE 2007 submission [1]. Section 1.4 provides a detailed discussion of our threading techniques.

1.1 Developers' mailing list

Apache developers are usually volunteers with other time commitments who rarely meet in a synchronous, transitory manner, so almost all project communication is recorded. The Apache community fosters a public style of discussion, where anyone subscribed to the mailing list can comment. All contributions that require discussion must be sent to the developers mailing list. There was 104,000 email messages on the developer mailing list between February 1995 and October 2005. From 1997 onwards, all contributions that are reviewed before being committed (RTC) have an email subject containing the keyword "[Patch]". This original message becomes the contribution, and all replies to this message become reviews and discussion pertaining to the contribution. There were 2603 contributions and 9216 replies to these contributions. All messages beginning "re: cvs [or svn] commit:" [2] are replies to commits with which a reviewer found an issue (CTR). In this case, the original message is a commit recorded in the version control mailing list. Responses can be classified as reviews, discussion, and occasionally confirmation that the "fix" worked for another developer. There were 2647 commits that received at least one reply and 9833 replies to these commits. Post commit review did not become an official Apache policy until early 1997.

1.2 Version Control Data

Apache policy requires developers to describe the changes made in each transaction to the version control system. This description, the commit log, includes information such as, who submitted and who reviewed the contribution. The log and the difference between the old and new version of modified files is sent to the version control mailing list. According to Fielding [1], a founding member of the Apache project, developers create a complete commit log more than 90% of the time and core project members review all commits. There were 2373 commits that had at least one reviewer named in the commit log. We cannot compare the number of accepted contributions to the number of submitted contributions on the mailing list because, according to Apache policy, minor contributions that do not require discussion only need to be submitted to the bug repository and fixes for post-commit reviews will have the name of the reviewer in the commit log. In future we would like to consider contributions sent to Bugzilla as well as linking mailing list reviews to actual commits.

1.3 Review Type Data Limitations

The data was not created for future analysis of review efficacy. We discuss the limitations of our data for both review types. **CTR** is never explicitly recorded, but we can deduce when it occurs by examining the mailing list for replies to committed contributions. Since a reply to a commit usually represents someone finding a problem with the commit, we know how many CTR contributions contained issues. There is no record, explicit or implicit, when a contribution is reviewed and no issues are found. Without a record, ignored contributions and reviewed contributions that do not have an issue are indistinguishable; we do not know what percentage of reviewed contributions do not contain issues. There are two sources of information about **RTC**. The first is the commit log which has an explicit record of who reviewed the contribution. This information is only available for review-then-commit approved (RTCA) contributions and only record which voting members approved the contribution and not how many people were actually involved in the discussion or how long the review took. To round our analysis, we examine contributions on the mailing list. This technique does not allow us to distinguish between accepted (approved) and rejected contributions. We ignore all contributions that did not receive a reply because no reply indicates that the contribution was either ignored or it was committed without review. This technique eliminated 841 emails that did not receive a response, reducing the number of RTC contributions to 1762. We are working on techniques to compare version numbers and source to link accepted contributions in the mailing list with their

[1] See `helium.cs.uvic.ca/thread.html` for a copy of our submission

[2] The start of this subject is automatically inserted by the version control system.

commits. Whenever possible we provide results from all representations of the data. With these limitations in mind, we often add qualitative understanding, through developer quotes and manual examination, to our quantitative results.

1.4 Extraction Tools and Techniques

We created scripts to extract the mailing list data into a database. A mail script extracted the mail headers including sender, in-reply-to, and date headers. The date header was normalized to GMT time. Once in the database we threaded messages by following the references and in-reply-to headers[3].

Since the references and in-reply-to headers are not required, 15% of the messages (replies to commits are left to later discussion) did not have a valid reply header. These messages were re-threaded using the date and subject of the invalid message. Valid messages with a similar subject ("re:" was striped off) that where within a seven day period prior of the invalid message were selected as potential parents. If this process left more than one potential parent, we selected the one closest in date to the invalid message. Some threading algorithms [4] use only subject matching. This technique is dangerous when subjects are common. For example, the subject "patch" was frequently used in the early days of Apache development, so threading on this subject alone would result in messages being incorrectly associated with each other. However, 4% of message subjects still started with "re:" but did not have a parent message. Examining some messages further, indicated that their parent message was not in the mailing list. One of the authors is an active member of an OSS project and stated that sometimes messages begin in private, but when it becomes obvious that the group would benefit from the discussion, the mailing list is included in the reply. However, this explanation is one of many. We removed these messages from the analysis. For contributions submitted to the mailing list (RTC) the percentage of broken threads was 1.8%. This percentage is likely lower than general discussion (4%) because most discussions of contributions are public, while some general discussion happens in private.

Since replies to a commit (CTR) will appear on the mailing list as a thread without a parent, we threaded these messages separately. We re-threaded messages associated with a commit as follows. First, all messages with a reply header that corresponded to a message (a commit) on the commit mailing list were threaded; this left 18.3% of the messages without a parent. Second, since the subject is the names of first few files involved in the commit and developers often made a series of commits to similar set of files within a short period of time, using subjects alone was not initially useful. Instead we choose a 3 day range and only re-threaded messages with one parent that had a matching subject. Finally, we used the algorithm mentioned above to re-thread siblings; in this case, the first reply to the commit is considered the parent. These techniques combined reduced the number of messages without a valid parent to 7.7%. Since the subject is automatically created and is a subset of the files modified, it is not indicative of the actual problem or even the files under discussion. We used a more conservative approach when re-threading these messages, and this is likely why a greater percentage of messages could not be re-threaded.

[3]For more information see RFC 2822 – Internet Message Format

[4]See `http://www.jwz.org/doc/threading.html` November 2006

2. REFERENCES

[1] A. Mockus, R. T. Fielding, and J. Herbsleb. Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology*, 11(3):1–38, July 2002.