CVSLogMonitor: オープンソース開発のためのモニタリングツール

藤田 充典　藤枝 和宏　落水 浩一郎
北陸先端科学技術大学院大学 情報科学研究科

1　はじめに

近年, レイモンドの報告 [1] や, Linux などの普及からソースコードを公開して開発を進めるオープンソース開発が注目されている. 公開されたソースコードはネットワークを介していつでも自由に入手することができる. また, ソースコードが明らかであるため, 利用者が不具合を見つけた場合や機能に不満を感じた場合に, 利用者自身が修正を加えたパッチを開発元に送ることができる. 開発元に送られたパッチは開発元の方針に従って処理され, 短期間でソフトウェアの修正や改善を促進することができる.

しかし, 綿密な打合せやスケジュール管理の下で進められる開発スタイルに比べ, 各開発者の自由な方針で進められるオープンソース開発では開発状況がわかりにくいという問題がある. 多くのプロジェクトではこの問題に対して, メーリングリストなどの連絡手段によってプロジェクトに関する議論を交わし, 開発状況に関する情報を補っている. しかし, 大規模なプロジェクトではその情報量が膨大となるために, 情報を記録, 要約する作業が大変な負荷となってしまう.

そこでこれらの問題を解決する一つの手段として, メールやドキュメントの内容を把握していなくても, プロジェクトの状況把握を実現できるツールの開発が必要だと認識した.

2　目的・手段

本報告では, オープンソース開発において, 全体の進捗状況や他の開発者達の作業内容の把握が困難になるという問題に対し, 開発環境に残されたログ情報を利用して状況把握をサポートする手法を提案する.

具体的には, 多くのオープンソース開発で利用されている版管理システム CVS[2] に残された膨大なログ情報を分析することで, 次の様な情報を抽出し, それをもとに分析を進めることで状況把握を実現する.

- 開発のベースラインから派生したブランチの数からブランチの存在が持つ意味の大きさを推測し, 各ブランチの大きさを調べることで, 影響力の大きなブランチを特定する.
- 開発者毎に, 更新を行っているファイルの種類や数, 更新量などを調べることで積極的な開発者を特定する.
- リリース間に行われた更新の頻度や, 更新量の変化に着目することで, 開発工程の波を抽出し, 更新の集中など, 突出した箇所を特定する.

以上のような情報を獲得するために開発したツールの紹介と, 実際にこれらの情報がどの程度推測可能かを確認することで, ツールの有効性を確かめるために行った実験について以下に述べる.

3　CVSLogMonitor

オープンソース開発を進めるために, 多くのプロジェクトが CVS を利用している [3] ことが知られている. そこで CVS のリポジトリから得られるログ情報を利用することで, 分析に必要なデータが得やすいと判断した. このため, ログ情報の獲得と分析作業をサポートするログ解析ツール CVSLogMonitor を設計, 実装を行っている (図 1).

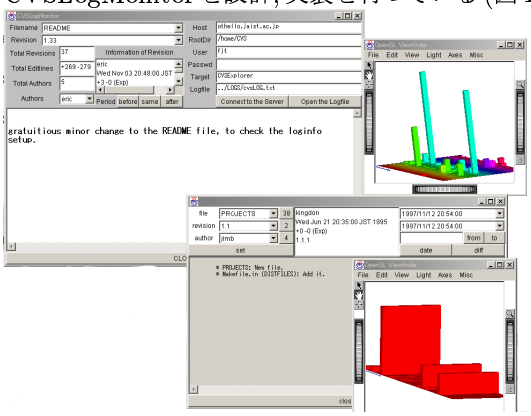

図 1: CVSLogMonitor の実行画面

このツールはネットワークを介して CVS のサーバに直接接続するか, CVS の log コマンドの出力を保存したファイルを指定することでログ情報を読み込むことができる. また, このツールはリポジトリから直にログ情報を獲得することができるため, ソースコードをチェックアウトする手間を省くことができる. 獲得されたログ情報はファイルの更新単位であるリビジョン毎にパースされ, 後から参照できるようにファイルにも保存される. CVSLogMonitor はこのログ情報からファイル名や登録された日時, 開発者の名前などで絞り込みを行い, それを出力する. 表示可能なデータには以下のものがある.

- 全てのファイル名とその情報
- ファイルの各リビジョンごとの情報
- リビジョンが更新された日時
- 開発者の名前 (アカウント名)
- 追加された行数／削除された行数
- リビジョンから派生しているブランチの有無
- 開発者が更新時に書き残したコメント

現実装では, ファイル毎のリビジョンツリーに残されたログ情報から得た数値データを視覚化することで直観的な分析をサポートする. CVSLogMonitor の主となる機能は, ファイルとそのリビジョンごとの変更情報を出力することである. また、その出力フォーマットとして, ファイル名や更新者, 更新日時による条件選択やソートといった, ログ情報の分析に有用と考えられる機能を備えている. 例えば, 現在のバージョン (1.11) の CVS の log コマンドではタグを指定して絞り込みをかける機能が提供されていないが, CVSLogMonitor ではタグのリストを出力し, 利用することができる。

3.1 プロジェクトの状況把握実験

ログ情報を利用するアプローチと CVSLogMonitor の有効性を検討するため, 開発者が更新を行なっているファイルの種類とその更新量に着目し, 開発者の役割や関わり具合を調べる実験を行った. 今回は, オープンソースで開発が進められている GIMP プロジェクト [4] から, モジュールの 1 つである glib の開発ログを獲得してその分析を行なった.

分析の結果からは, 更新を行なった開発者が 35 人, モジュール全体での更新行数が 45562 行, そのうち C プログラムに関する更新が 22072 行であることが分かった. まず, 開発者毎の全ての更新行数をグラフ化したものと, ドキュメントを除いた, ソースコードの更新行数だけをグラフ化したものを図 2 に示す.

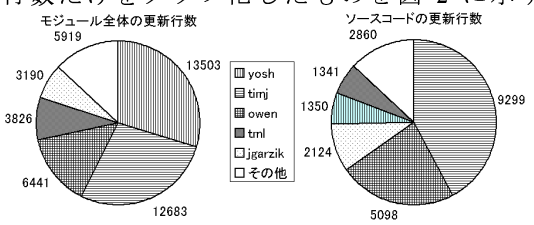

図 2: 開発者ごとの更新量の違い

このグラフからは, モジュール全体で行われた更新行数の半数以上が yosh と timj という 2 人の開発者によって行われていることが分かる. また,2 つのグラフを比較して見ると, 全体では 2 番目に更新量が多かった timj がソースコードに関しては最も多くの更新を行なっていること, 全体では最も多くの更新を行なっていた yosh がソースコードに関してはそれほど更新を行なっていないなど, 開発者毎の更新対象の違いを見ることができる.

3.2 実験結果の考察

実験の結果からは, プロジェクトにおいて積極的に開発を行っている開発者とその作業対象を眺めることができた. このような分析結果にさらにいくつかの情報を付加していくことで, プロジェクトの状況把握の助けになると考えられる.

4 まとめと今後の課題

本報告では, CVS リポジトリから得られるログ情報を利用してプロジェクトの状況把握を行なう手法を提案し, 実際にツールを利用して作業履歴を獲得し, その分析実験を行った. そして, その分析結果からは開発者の開発者対象の違いを見ることができた. しかし, 作業履歴だけでは具体的な作業内容に関する情報を獲得できないため, 足りない情報については, メーリングリストなどで行われている議論の内容や, ソースコードを解析することで得られる情報を利用して補う必要がある.

まだ実験が十分でないため, 確証的なデータが得られていないが, 今後, 分析作業の洗練によって獲得できる情報の精度が上がることで, このログ情報の分析によるアプローチはプロジェクトの状況把握の有効な手段になると考えている.

参考文献

[1] Eric Steven Raymond. The cathedral and the bazaar. availed at `http://www.tuxedo.org/~esr/writings/cathedral-bazaar/`, May 1997.

[2] B.Berliner. Cvs ii: Parallelizing software development. In *1990 Winter USENIX Conference, Washington D.C.*, 1990.

[3] Karl Fogel. *Open Source Development With CVS*. CoriolisOpenPress, 1999.

[4] Gimp project. availed at `http://www.gimp.org/`.