トピック抽出を用いたソフトウェア開発履歴の可視化ツール

山田 悠太　吉田 則裕　藤原 賢二　飯田 元

近年，企業が開発するソフトウェアプロダクト中に，OSS が組み込まれることが多くなった．企業の開発者は，OSS を採用するかどうか判断する際に，関心のある機能に貢献している開発者に連絡をとり，その機能に関するバグ修正や拡張の保守計画を知りたいと考える．しかし，個々の機能に貢献している開発者を特定することは，OSS がボランティア活動で成り立っていることから難しい場合が多い．本研究では，開発履歴に対してトピック抽出を行うことで，個々の OSS 開発者の活動を可視化するツールを開発する．本ツールを用いることで，個々の機能に貢献している OSS 開発者を特定する作業を支援できると期待される．

Recently, commercial software products often incorporate OSS. Industrial developers often need to know plans of enhancement and bug fix for a specific feature of OSS when they should determine whether or not to incorporate it. However, it is difficult for outsiders to retrieve a person familiar with a specific feature in OSS due to the voluntary nature of the contributions. In this paper, we present a tool for visualizing version archives to support industrial developers who find OSS developers familiar with a specific feature. This tool applies topic analysis to version archives for characterizing activities of individual developer.

1　はじめに

オープンソースソフトウェア（OSS）は SourceForge.net や GitHub のソフトウェア開発プロジェクトを共有するための Web サービスが近年台頭してきたことにより開発が活発である．そして，OSS は利用料を必要としないため，品質の高い OSS は商業利用される傾向が大きくなっている [3]．しかし，OSS を利用する問題点としてソフトウェアの品質保証や OSS プロジェクトのサポート体制の不備が挙げられる．もし，企業のプロジェクトにおいて OSS 利用箇所で問題が発生した場合，企業は OSS プロジェクトに問題について問い合わせることが考えられるが，OSS プロジェクトの体制によっては回答までに時間が掛かってしまう．そのため，直接問題箇所を担当した開発者に問い合わせることが効率的であるが，OSS プロジェクトでは開発箇所やその担当を管理することは少なく，またプロジェクトへの参加は開発者の自由意志に基づくため時間経過によりプロジェクトの参加者は刻々と変化してしまうため，開発者を探すにはプロジェクトの開発履歴から探すことになる．

これまでに，ソフトウェア開発履歴の可視化手法は数多く提案されてきた [2] [5]．しかし，膨大な開発履歴を対象として「どの活動を行った開発者は誰であるか」を可視化する手法は，著者らの知る限り存在しない．

そこで本研究では，バージョン管理システムに蓄積されたコミットログからトピックを抽出し，開発者の活動に着目した可視化を行う．本研究におけるトピックは，コミットログ中に含まれる単語の確率分布であり，上位に属する単語を提案する開発者活動の可視化に用いる．コミットログのみを用いることで特別なデータを収集する必要が無く，開発言語に依存せず本手法を適用することが可能である．本研究において可視化の対象とする情報は，コミットログに記載され

A Topic-based Visualization Tool for Archives of Developer Activity

Yuta Yamada, Norihiro Yoshida, Kenji Fujiwara, Hajimu Iida, 奈良先端科学技術大学院大学 情報科学研究科, Graduate School of Information Science, Nara Institute of Science and Technology.

ているプロセスの種類（リファクタリング，バグ修正など）およびそのプロセスの実施者や対象となったプロダクト名（ソースファイル名，ディレクトリなど）である．

2 可視化手法

本章では提案する手法であるトピック解析を用いたモデル構築手法とトピック可視化ツールについて述べる．トピック解析はプロジェクトにおける活動履歴に関するトピックを求めるために，VCS のコミットログコメントを利用する．可視化はプロジェクトの活動履歴及びその活動を行った開発者を提示することを目的としている．また，可視化ツールは開発者単位の視点やトピックに着目した視点など複数の視点を切り替えることができ，ツール利用者がインタラクティブに必要な情報を見つけ出すことを可能としている．

2.1 トピックモデルの構築

トピックモデルを構築するには 5 つのプロセスが必要であり，それは順に『文書の作成』『トピック解析』『トピック割当値の計算』『トピックの連結』『トピックの命名』である．図 1 はモデルの構築手順の概要である．

2.1.1 文書の作成

文書の作成では，VCS のコミットログコメントからトピック解析を行うための文書を作成する．文書とは単語の集合のことであり，文書はコミットごとに作成する．また，コミットログコメントを単語に分解後にステミングを行う．ステミングとは語幹以外を取り除く処理であり，例えば *removed, removing* → *remov* のように，語幹のみにすることでトピック解析時の精度を上げることができる．コミットログコメントを用いる理由は，開発で行われたことが要約されたものであり開発の活動内容を含んだ単語が含まれていることが期待できるからであり，コミットごとに活動を行った開発者が一意に定まるためである．

2.1.2 トピック解析

トピック解析では，手順 1 で作成した文書からトピックを抽出する．トピック解析は潜在的ディリクレ配分法（LDA: Latent Dirichlet Allocation）[1] が実装された機械学習ツール MALLET [4] を用いる．LDA は自然言語処理で使われるトピックモデルを作成する手法であり，入力に用いる文書集合からその文書に含まれる単語の出現確率を利用してトピックを解析する．解析で抽出するトピック数は自由に設定することが可能であり，解析によりトピックの単語分布と入力に用いた文書ごとのトピック分布を求められる．

トピックは単語の確率分布であり，分布の上位のキーワードからトピックの意味を知ることが可能である．表 1 はトピックの単語分布の例であり，トピック A は上位のキーワードから不具合の修正のトピックと分かる．同様に，トピック B はメソッドやクラスの変更，トピック C は新規メソッドの追加と分かる．

文書のトピック分布とは，文書がどのトピックに属するかを示した確率分布である．例えば，表 2 では文書 A はトピック A について 0.7，トピック B について 0.2，トピック C については 0.1 の生起確率を持っているため，文書 A はトピック A の意味を持った文書である確率が高いことを示している．LDA の利点として，直接的な繋がりを持たない単語でも同一のトピックのキーワードとして抽出することが期待できることが挙げられる．ソフトウェア開発は複数人で行われているため，同じ活動を行ったとしても人によりコミットログコメントの表現揺れが起きることが考えられるが，LDA であれば表現揺れを抑えて同一のトピックとして抽出することが可能である．

また，本手法では区間ごとにトピック解析を行う．例えば，区間を 1ヶ月単位とする場合はコミットログを 1ヶ月単位で区切り，区間ごとにそこに含まれるコ

表 1 トピックの単語分布

	上位キーワード
トピック A	*fix, bug, error*
トピック B	*change, method, class*
トピック C	*add, method, new*

表 2 文書のトピック分布

	トピック A	トピック B	トピック C
文書 A	0.7	0.2	0.1
文書 B	0.2	0.8	0.0
文書 C	0.1	0.0	0.9

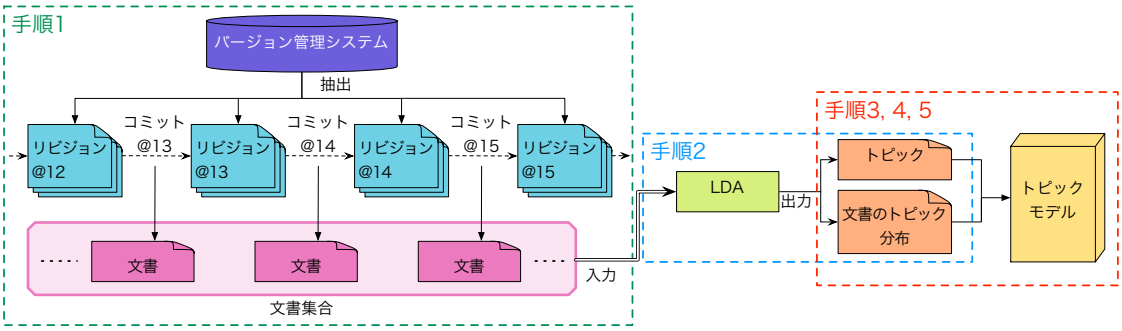

図 1　トピックモデルの構築手順

ミットログのみを用いてトピック解析を行う．

2.1.3　トピック割当値の計算

トピック割当値とは，トピックの大きさ尺度であり，同一区間に含まれる文書のトピック分布の生起確率を合算したものである．また，トピック割当値は開発者ごとに求める．例えば，表 2 がある区間のトピック分布であり，開発者 α が文書 A および B，開発者 β が文書 C に属するとき，各開発者のトピック割当値は表 3 のように求まる．トピック割当値から，この区間において開発者 α はトピック A とトピック B に関する開発を行っていたと考えられ，同様に開発者 β はトピック C に関する開発を行っていたと考えられる．

2.1.4　トピックの連結

本手法では区間ごとにトピック解析を行っているため，区間ごとに異なるトピックが抽出されてしまう．そのため，トピックの変化を求めるためには異なるトピック間で類似するトピックを結びつけ 1 つのグループにする必要がある．トピックは単語分布であり，生起確率の大きい上位のキーワードがトピックの意味を表している．つまり，上位キーワードが一致したとき，類似のトピックと考えることが可能である．しかし，上位キーワードであっても生起確率の大きさに差があるため，生起確率の値も考慮して類似度を計算し，閾値を超えたときに 2 つのトピックを同一のトピックグループとする．トピックは過去から順に比較し上位キーワードを記録する．もし 2 つのトピックが類似トピックだった場合，過去の上位キーワードを新規のトピックの上位のキーワードで上書きし記録する．

表 3　開発者ごとのトピック割当値

	トピック A	トピック B	トピック C
開発者 α	0.9	1.0	0.1
開発者 β	0.1	0.0	0.9

2.1.5　トピックの命名

トピックの名前は 2 種類あり，トピックグループの総称とトピック単体ごとの名称である．トピックグループの総称は，グループに属するトピックの単語分布を利用し，トピック割当値同様に同一キーワードごとの生起確率の値を合算し，合算値の上位であるキーワードの 3 つを順に並べて命名する．また，トピック単体の名称は，トピックの単語分布の上位 5 つのキーワードを順に並べて命名する．もし，トピックグループに属するトピックが 1 つである場合，グループの総称とトピックの名称の上位 3 つは一致することになる．

2.2　トピック可視化ツール

トピック可視化ツールは前節で構築したトピックモデルを用いて可視化を行う．

2.2.1　プロットスタイル

可視化を行う際のスタイルには 2 つあり，図 2 のドットスタイルと図 3 のカラムスタイルである．

ドットスタイルは主に横軸が時系列を表すときに用いられる．ドットはトピックを表しており，色と形状

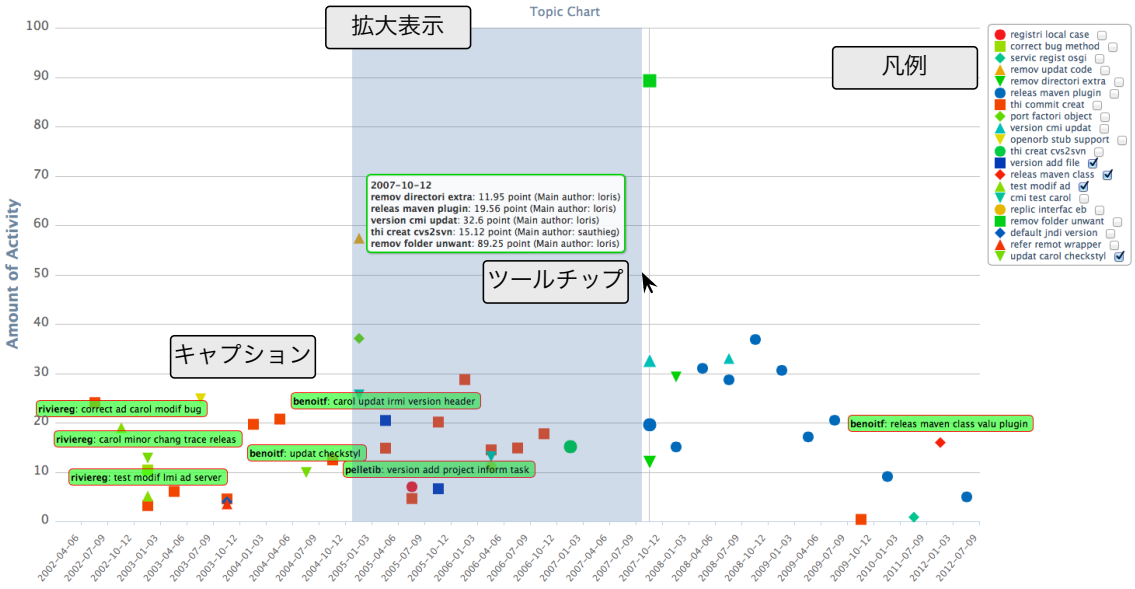

図 2　トピック可視化ツールの画面（ドットスタイル）

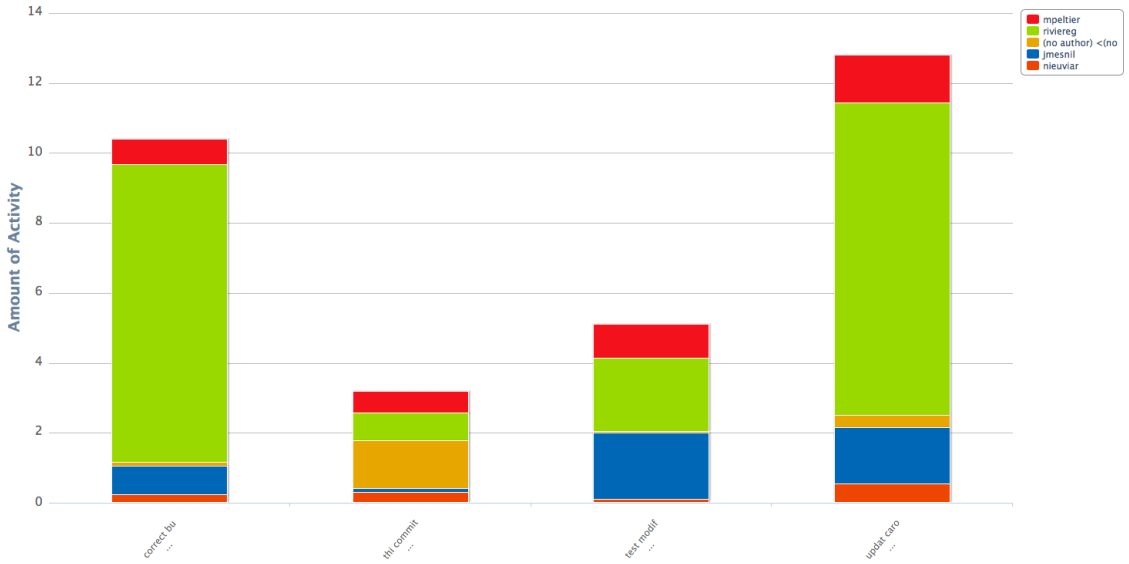

図 3　トピック可視化ツールの画面（カラムスタイル）

が一致するものは同じトピックグループに属しているトピックである．縦軸はトピック割当値の大きさを表しており，時間経過によるトピックの変化を知ることが可能である．例えば図 2 から，2007 年 10 月 12 日を含む区間において，トピック割当値が最大のトピックは 89.25 ポイントの (remove, folder, unwant) であり，主要な貢献者は loris という ID を持つ開発者であることがわかる．このことから，2007 年 10 月 12 日頃，不要なフォルダの削除に関する作業が主に行われており，loris という ID の開発者がその作業に大きく貢献していたと推測される．

カラムスタイルは主にトピックグループや区間に着

目したときに用いられる．横軸はトピックグループに着目した場合はグループに属するトピックが抽出された各区間，区間に着目した場合は区間に含まれる各トピックが表示される．縦軸はドットスタイルと同じくトピック割当値の大きさを表しており，そのトピックに関係する開発者のトピック割当値を積み上げたものになっている．積み上げられたトピック割当値の大きさを比較することで，開発者がそのトピックに関する活動にどれだけ寄与していたのかを知ることができる．例えば図 3 において，最も左側の棒グラフはバグ修正に関するトピックグループ (correct, bug, method) を表しており，開発者毎のトピック割当値を積み上げられている．この棒グラフからは，上から 2 番目の領域に対応する riviereg という ID の開発者が最も大きく貢献していることが推測できる．

2.2.2 可視化ツールの機能

ツールチップ（図 2）は，プロットエリア上にマウスポインタを載せると表示され，その軸上にあるドットごとのトピック割当値やカラムなら各開発者のトピック割当値の詳細を知ることが可能である．また，開発プロジェクト全体をドットでプロットしてる場合は，その区間で最も開発を行っていた開発者の名前も表示するため，容易に主要開発者のみを知ることが可能である．

凡例（図 2）には，トピックグループ名が載っている．凡例の名前をクリックする事により，対応したドットなどを非表示したり表示することが可能である．そのため，トピックグループ名を読み取り必要の無いものを非表示にして，プロットされているグラフの可読性を上げることが可能である．また，凡例のトピックグループ名の横にあるチェックボックスをオンにすることで，そのトピックグループが属するトピックのドット上にキャプションを表示することが可能である．キャプションにより表示される情報はツールチップとほぼ同じであるが，キャプションは常に表示させることが可能であり，トピックごとの名前を知ることも可能である．さらに，キャプションとして表示することの副作用として，どの位置にドットがプロットされているかを容易に知ることができる．

プロットエリア上でマウスをドラッグすると，ドラッグして指定した範囲を拡大することが可能である（図 2）．そのため，ドットが密集している場合は，密集部分を拡大することができる．

3 評価実験

本章ではトピック可視化ツールの有用性を評価するために行った評価実験について記述する．実験では，OSS プロジェクトの Apache Ant(12789 コミット，約 3 年間)，CAROL(2236 コミット，約 10 年間)，jEdit(7151 コミット，約 11 年間)，WALA(3611 コミット，約 6 年間）のコミットログからそれぞれトピックモデルを構築し可視化を行った．そして，それぞれのコミットログから開発者の活動に関する 4 つの大問を作成し，被験者には可視化ツールのグラフから情報を読み取る方法とリポジトリから直接 UNIX コマンドを用いて解答を導く方法の 2 種類で解答してもらい，その解答に要した時間と解答の正解率で評価を行った．また，被験者はソフトウェア工学を学ぶ 6 人の大学院生であり，UNIX コマンドやリポジトリマイニングに関する知識を有している．

3.1 実験手順

実験では，まずトピック可視化ツールと UNIX コマンドの事前講義を行った．可視化ツールはデモ用のデータを用いて被験者にツールを実際に使ってもらいながら，各機能の説明を行った．また，UNIX コマンドである git log，grep，wc を利用したリポジトリマイニングの方法について説明を行った．

事前講義後，被験者を 2 つのグループに分け問題を解答してもらった．問題はプロジェクトごとに複数問を用意した．問題は，全てリファクタリングやバグ修正等のプロセスの種類，およびそのプロセスを実施した開発者や対象となったソースファイル名などのプロダクト名を問う内容である†1．一方のグループがツールを利用する場合はもう一方はコマンドを利用して問題を解答し，大問が終わると次の大問では解答方法をグループで入れ換え実験を行った．可視化ツールを用いる場合は，ツールから得られる情報のみで解

†1 問題に関する詳細は [6] の 4.1.3 節を参照されたい．

答してもらった．UNIX コマンドを用いる場合は，事前講義で教えたコマンドに加え，被験者にとって既知のコマンドや実験中にインターネットで利用方法を調べたコマンドの使用も許可した．また，大問は複数の小問から構成され，小問ごとに解答時間を計測した．小問 1 問毎に 5 分を割り当て，5 分経過した場合は次の小問へ進んでもらった．

3.2 実験結果と考察

実験結果を表 4 に示す．小問あたりの平均時間は可視化ツールの方がコマンドを利用した場合と比較して 2 分ほど早く解答されていることが分かる．また，問題の正解率に関しても，ツールの方がコマンドに比べて正しく解答されている．しかし，時間切れの問題を除去し，解答された問題のみを対象とする場合は，ツールとコマンドでで は 4%の差があるだけである．

それぞれの解答時間と正解率の平均に統計的有意差があるかを有意水準を 5%とし，T 検定と F 検定を用いて確認した．まず，解答時間の場合，制限時間により計測できなかった問題があるためデータに対応関係がないため，F 検定を行い分散が不等分散であるかを確認し，その結果，解答時間の分散は不等分散であることが分かった．次に不等分散を考慮した T 検定を行い，両側確率が 0.000757 となり有意水準を下回った．よって，可視化ツールとコマンドの解答時間には統計的有意差があることが分かった．また，同様に正解率についても両側確率が 0.000517 であったため，それぞれの正解率にも有意差があることが確認された．しかし．時間切れの問題を無視する場合は両側確率が 0.167 となり，有意水準を上回るため有意差が確認できなかった．

これらの結果から，可視化ツールは，コマンドと比べて短時間で同程度の正確さの分析を行うことができると考えられる．ただし，数千回のコミットからなる開発履歴が前提となると考えられる．また，被験者全員がコマンドやリポジトリマイニングに関する知識を有しているにもかかわらず，初めて利用した可視化ツールの方が優れた結果であったことから，可視化ツールの使用方法に対する学習コストは低いと考えられる．

表 4　小問あたりの平均解答時間と正解率

	ツール	コマンド
平均解答時間	1:37.3	3:31.1
時間切れ率	0%	27%
正解率	83%	53%
正解率※	83%	79%

※時間切れの問題を除去する場合

4 まとめと課題

本研究では，トピック抽出を用いてソフトウェア開発履歴を可視化するツールを開発した．また，被験者実験を通した有用性の評価を行ったところ，本ツールは UNIX コマンドを利用した方法よりも効率的に開発履歴の理解支援を行うことができることがわかった．今後の課題として，より多くの被験者を対象とした実験の実施や，コミットログ以外の開発履歴への適用が考えられる．コミットログ以外の開発履歴として，メーリングリストのアーカイブやソースコードの変更履歴が挙げられる．また，これら開発履歴の可視化と本研究で行ったコミットログを利用した可視化を組み合わせることで，より効率的な可視化を目指すことも今後の課題である．

参 考 文 献

[1] Blei, D. M., Ng, A. Y., and Jordan, M. I.: Latent dirichlet allocation, *Machine Learning Research*, Vol. 3(2003), pp. 993–1022.

[2] Hindle, A., Godfrey, M. W., and Holt, R. C.: Software Process Recovery using Recovered Unified Process Views, *Proc. of ICSM '10*, 2010.

[3] IPA: 第 3 回オープンソースソフトウエア活用ビジネス実態調査 (2009 年度調査）. `http://www.ipa.go.jp/software/open/ossc/seika_1004.html`.

[4] McCallum, A. K.: MALLET: A Machine Learning for Language Toolkit, 2002. `http://mallet.cs.umass.edu`.

[5] Thomas, S. W., Adams, B., Blostein, D., and Hassan, A. E.: Studying Software Evolution Using Topic Models, *Science of Computer Programming*, Vol. 80(2014), pp. 457–479.

[6] 山田悠太: 開発履歴からのトピック抽出に基づく開発者活動の可視化ツール, 修士論文, 奈良先端科学技術大学院大学, 2013. `http://sdlab.naist.jp/pman3/pman3.cgi?DOWNLOAD=78`.