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

商業利用されるオープンソースソフトウェア（OSS）が近年多く存在する．それは，OSS の品質が高く利用料を必要としないからであるが，欠点として品質の保証やサポートの不備が挙げられる．もし，OSS に問題があった場合，企業は OSS の担当者に問い合わせをする可能性があるが，OSS プロジェクトの性質上特定の人物を探すのは困難であると考えられる．そこで，本研究では版管理システムのコミットログからトピックを抽出し，開発者に着目した可視化を行った．また，可視化を行うためのツールを実装し評価を行った．

Recently, commercial software products often incorporate Open Source Software (OSS). Industrial developers often need to know plans of enhancement and bug fix for a specific feature/component 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/component 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/component. 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 は商業利用する傾向が大きくなっている [5]．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．

開発履歴の可視化は今までも行われている．Hindle らはソフトウェア開発履歴から Word-bugs 解析や LDA によるトピック解析を用いて統一プロセスにおける各フェーズのプロセスに関連付け，RUP Hump チャートを模倣し可視化を行っている [2]．Thomas らはソースコードの識別子とコメントからトピックを抽出し，可視化を行うことでソフトウェア開発の変化を表している [4]．しかし，これらの既存研究は開発全体にしか着目されていない．

そこで，本研究では開発履歴であるバージョン管理システム（VCS）の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．コミットログは開発者が行った活動内容を簡潔に記述するため開発者の活動を知ることが期待でき，開発者に着

A Topic-based Visualization Tool for Archives of Developer Activity

Yuta Yamada, Kenji Fujiwara, Norihiro Yoshida, 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.

目した可視化により開発者ごとの活動の流れを容易に知ることが可能になる．また，コミットログのみを用いることで特別なデータ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．可視化はツールを通して行いインタラクティブに情報を提示することを可能とし，ツール利用者が可視化から自由に情報を得られるようにする．可視化を行うことで開発履歴から開発の流れを知る必要が無くなり，誰にでもそれが理解できるようになる．

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測』『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．

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*

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 [3] 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され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

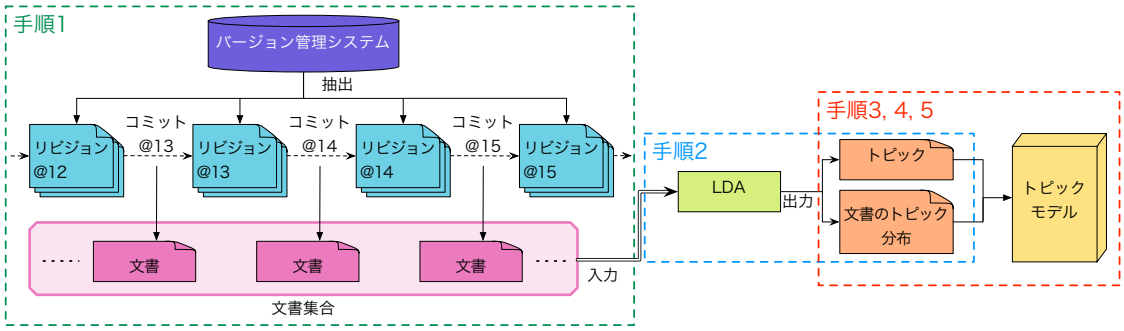

図 1　トピックモデルの構築手順

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

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

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測

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．

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．

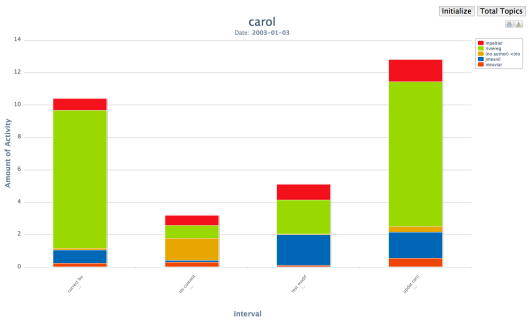

図 3　トピック可視化ツールの画面（カラムスタイル）

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．[†1]実装は HTML+JavaScript で行われているため Web ブラウザが動作する環境で動作し，チャート作成ライブラリ Highcharts を利用している．

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．開発プロジェクト全体に着目したものと開発者に着目したものの 2 種類があり，前者は従来の可視化のように開発全体のトピックの変化を表しており，後者は開発者個人のトピックの変化を表すため開発者が行った活動を読み取ることができる．

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．トピックグループに着目した場合は類似のトピックが行われていた時期のみを知ることが可能であり，また，そのトピックの活動に関わる開発者の変化を知ることが可能である．区間に着目した場合も同様に時間経過により，関係する開発者の変化を読み取ることが可能である．

2.2.2　可視化ツールの機能

トピック可視化ツールには 5 つの機能がある（図 2 を参照）．

ツールチップ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，副作用としてキャプションの表示することにより，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．もし，ドットが密集して視認性が悪くても，密集箇所を拡大すること解決することができる．

†1　http://www.highcharts.com/

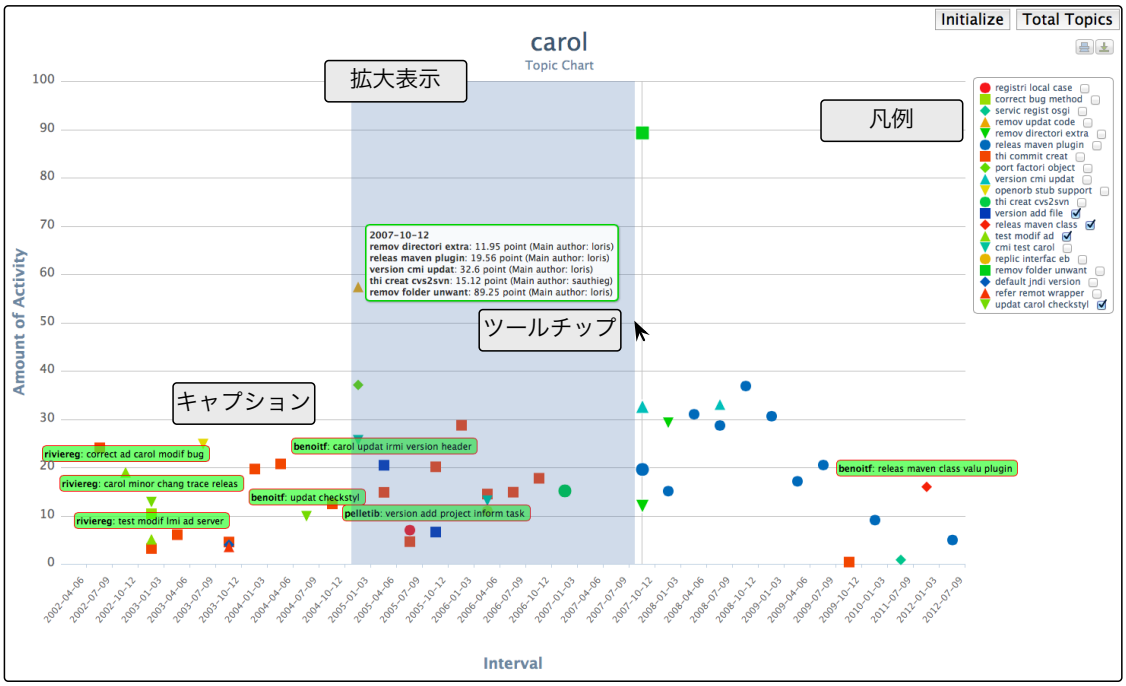

図 2　トピック可視化ツールの画面（ドットスタイル）

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, CAROL, jEdit, WALA 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 人の大学院生であった．

3.1　実験手順

実験はまず事前講義として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 コマンド，grep コマンド，wc コマンド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．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以外のコマンドでも被験者自身が普段から利用しているコマンドや知らないコマンドをネットで調べることで利用可とした．また，大問には複数の小問があり解答時間の計測は小問ごとに行った．小問 1 問ずつ解いてもらい，解けない限り次の小問へ進めないようにし，また，原則として 1 問 5 分で解答し，5 分を過ぎた場合は解答を無視して次の小問へ進んでもらった．

大問が全て解き終わったら，被験者のコマンドやリ

表 4　小問あたりの平均解答時間と正解率

	全体	ツール	コマンド
平均解答時間	2:25.1	1:37.3	3:31.1
正解率	68%	83%	52%
正解率※	78%	83%	78%

※時間切れの問題を無視する場合

ポジトリマイニングに関する知識を問うアンケートを行った．

3.2　実験結果と考察

実験結果は表 4 のようになった．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，時間切れの問題を無視し，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 5 ポイント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．

最後に行ったアンケートにより，被験者全員が今まで UNIX コマンドの利用経験があり，また Git や Subversion の知識やリポジトリマイニングに関する知識を有していることが分かっている．そのため，被験者は初めて利用する可視化ツールよりコマンドを用いてリポジトリから情報を検索することに慣れているはずである．しかし，実験結果からは初めて利用する可視化ツールの方がコマンドを用いた場合よりも良い結果を示している．つまり，可視化ツールを利用方法を短時間学ぶだけで使いこなすことができ，また，可視化により詳細なデータを直ぐに読み取ることが可能になる．

4　まとめと課題

本研究では，開発者に着目したソフトウェア開発履歴の可視化を目的として OSS プロジェクトを対象にトピックモデルを構築し，それをツールを可視化することで開発履歴を分かりやすく知ることができることの有用性評価と考察を行った．その結果，可視化ツールは UNIX コマンドを利用した方法よりも短時間で情報を求めることが可能であり，短時間の学習時間で利用可能であることが分かった．また，トピックモデルが開発者活動を表していることも確認できた．

しかし．被験者からトピック可視化ツールに関して機能不足や使い方が分かりづらいなどの指摘された．そのため，有用性が確認できたものの，更なる改良が必要であることが考えられる．

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

[4] Thomas, S. W., Adams, B., Blostein, D., and Hassan, A. E.: Studying Software Evolution Using Topic Models, *Science of Computer Programming*, (2013), pp. 1–23. in press.

[5] IPA（独立行政法人情報処理推進機構）: 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）.