Vol.2012-SE-178 No.16
2012/11/2

# トピック抽出に基づく 開発者の活動に着目したリポジトリ可視化手法

山田 悠太[1,a)]　藤原 賢二[1,b)]　吉田 則裕[1,c)]　飯田 元[1,d)]

概要：ソフトウェア開発プロジェクトでは，個々の開発者の活動がソフトウェアの品質に影響を与えると考えられており，開発プロセスの改善にはプロジェクト全体ではなく開発者単位でも行われるべきと考えられている．しかし，開発者の活動を逐一記録するのは困難である．通常ソフトウェア開発では構成管理システムが利用されており，開発で作成・編集されるソースコードや開発における不具合の修正履歴などを記録している．そこで，構成管理システムの 1 つであるバージョン管理システムに記録されたデータから活動を可視化する手法を提案する．まず，ソースコードのコメントや識別子名，システムのコミットログからコミット単位でドキュメントを作成し，それを LDA（latent Dirichlet allocation）を用いてトピックの抽出を行う．次にドキュメントの基になったコミットを行った開発者とドキュメントのトピック分布を用いてトピックの変化を可視化する．最後に抽出されたトピックと可視化されたトピックの変化から開発者の活動を推定する．本論文ではオープンソースソフトウェアプロジェクトの Columba に提案手法を適用し得られた一部の特徴的なトピックを中心に結果と考察を述べる．

キーワード：リポジトリマイニング，トピック解析，可視化

# Repository Visualization Technique Based on Developer Activity Using Topic Analysis

Yuta Yamada[1,a)]　Kenji Fujiwara[1,b)]　Norihiro Yoshida[1,c)]　Hajimu Iida[1,d)]

***Abstract:*** During a software development, the activity of an individual developer is considered to affect software quality. Process improvement should be performed on not only a project but also an individual developer. However, it is difficult to keep the record of the activity of a developer continuously. Recent software development uses configuration management systems that keep the record of source file creation and modifications as well as also bug fixes. In this paper, we propose an approach to visualize the activity of an individual developer using the record in a version control system. In the approach, at first, documents for the input of the Latent Dirichlet Allocation are generated from comments and identifiers involved in source code as well as commit logs, and then topics are derived from generated documents. Finally, the evolutions of derived topics are visualized by linking the information on each developer who contributed to the source code and each topic. A user of the proposed approach is expected to understand the activity of a developer from the visualization of the derived topics and their distributions. The case study mainly presents the topics derived from an open source project Columba and the discussion on them.

***Keywords:*** Mining software repositories, Topic analysis, Visualization

[1] 奈良先端科学技術大学院大学
Nara Institute of Science and Technology
[a)] yuta-y@is.naist.jp
[b)] kenji-f@is.naist.jp
[c)] yoshida@is.naist.jp
[d)] iida@itc.naist.jp

## 1. はじめに

ソフトウェア開発プロセスの分析や改善は，開発組織やプロジェクト単位で行われることが多いが，開発者単位で

も行うべきとされている [1]．開発者自身が自分自身のプロセスを分析すると，自分自身の生産性や頻繁に行う誤りを認識することできる．それにより，開発者自身が開発プロセスや計画の変更を提案しやすくなったり，作業効率の改善を検討しやすくなる [2] ．しかし，開発者単位の開発プロセスを分析するためには，個々の開発者が自身のプロセスの分析に時間をかける必要があり，開発組織やプロジェクトに定着させることは難しい [3] ．個々の開発者のプロセスの分析のためには，行った作業の内容と時期を手作業で記録する必要がある．

我々の研究グループでは，開発プロセスの可視化分析環境としてプロジェクトリプレイヤを開発した [4] [5] ．この環境は，個々の開発者の編集作業に関しては，編集されたファイルと編集者を時系列に沿って表示するのみであり，個々の開発者が作業内容を理解し，プロセスの分析や改善を行うことは難しい．

本研究では，バージョン管理システムに記録された開発履歴を用いて，開発者の作業内容の可視化を行う手法を提案する．本手法は，ソースコードの編集履歴に対して，LDA（latent Dirichlet allocation） [6] を用いたトピック（テキストに含まれるキーワードの集合）抽出を適用し，個々の開発者のトピックの変化を時系列に沿って可視化する．ケーススタディでは，本手法をオープンソースソフトウェアの開発履歴に適用した．以降，2 章では本研究に関連する研究の説明を行う．3 章では提案手法であるトピック抽出に基いて開発者の作業内容を可視化する手法について述べ，4 章ではオープンソースソフトウェアに提案手法を適用した結果，5 章ではその考察について述べる．最後に 6 章ではまとめと今後の課題について述べる．

## 2. 関連研究

我々の研究グループではソフトウェア開発プロジェクトを可視化し，プロジェクトの振り返りを支援する環境としてプロジェクトリプレイヤを開発した [4] [5]．プロジェクトリプレイヤは，構成管理システム（CVS，Subversion など）に記録されたソースコード変更履歴，バグ管理システム（Bugzilla，GNATS など）に記録されたバグに関する情報，プロジェクトにおいて利用されていたメーリングリストの情報を時系列に沿って可視化する．既存のソフトウェア開発履歴とプロジェクトリプレイヤを用いることで，プロジェクト分析者は容易にプロジェクトを時系列に沿って俯瞰することができる．Hindle らはソフトウェア開発履歴から開発プロセスを半自動的に推定し，可視化する手法を提案している [7]．彼らの手法では，Word-bugs 解析や LDA によるトピック解析を用いてコミットやバグ修正などを統一プロセスにおける各プロセスに関連付け，RUP Hump チャートを模倣した図として可視化する．これにより，プロジェクトのどういった時期にどのようなプロセスが重点的に行われていたかを分析することができる．これらの手法はプロジェクト全体を俯瞰することを主目的としており，開発者個々人の開発プロセスを分析するための情報提示は十分でない．プロジェクトリプレイヤは構成管理システムへのコミット，メーリングリストへの投稿，バグ票の投稿及び対応といった開発者の活動を開発者ごとに時系列に沿って表示するのみであり，Hindle らの手法は開発者に関する情報は可視化していない．

LDA は文書集合からトピック（話題）の集合を抽出し，各文書におけるトピックの確率分布を推定する手法である．Bernardi らは LDA を用いて Firefox と Chrome のバグ管理システムからトピックを抽出し，両プロジェクト間の違い分析を行っている [8]．また，Chen らはソースコードから抽出したトピックとソフトウェアの欠陥を用いてトピックメトリクスを提案している [9]．彼らの手法では，ソースコードの識別子とコメントを文書として LDA を適用する．本研究の提案手法でも同様の方法でソースコードからトピックを抽出する．

## 3. バージョン管理システムからの活動の可視化

今日，バージョン管理システムは多くのソフトウェア開発プロジェクトにおいて利用されており，開発者がファイルの作成・編集する度にそのファイルをバージョン管理システムに記録するコミットが行われる．このバージョン管理システムには開発者の活動に関係するデータがあると考えられる．

そこで，バージョン管理システムのデータから単語を抽出し，それを対象にトピック解析し開発者の活動を可視化する．そして，可視化したトピックの変化とトピックのキーワードから開発者の活動を推定する．

トピック解析を行う利点として，単語の集合であれば自然言語以外にも適用が可能であるため，ソースコードへの適用が可能である点が挙げられる．また，意味を持ったキーワードの集合が得られるため，活動内容を推定する際の意味付けが容易になる点も挙げられる．

### 3.1 提案手法

バージョン管理システムのデータを用いて可視化を行う手順は以下の通りである．

**手順 1**　コミットごとにドキュメントを作成する．
**手順 2**　ドキュメントを入力とし，LDA を用いてトピック解析を行う．
**手順 3**　ドキュメントのトピック分布を用いてグラフを描く．
**手順 4**　グラフから開発者の活動を推定する．

図 **1** は提案手法の概略図である．以降，各手順の詳細について記述する．

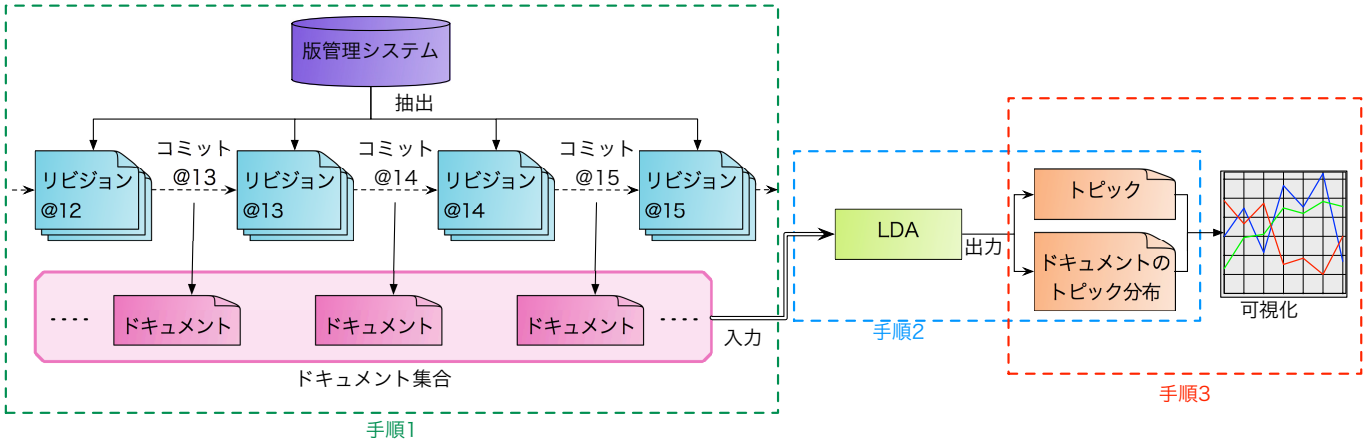



図 1　提案手法の概要
**Fig. 1**　An overview of proposed technique

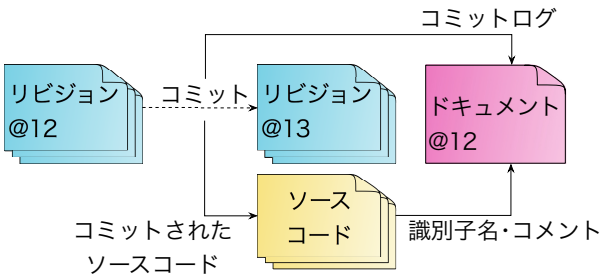



図 2　ドキュメントの作成
**Fig. 2**　Document generation

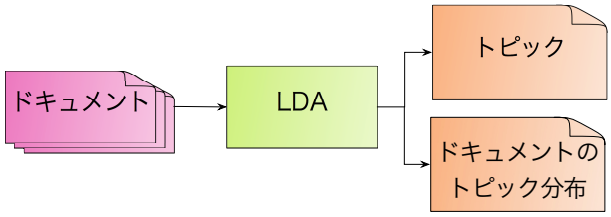



図 3　トピック解析
**Fig. 3**　Topic analysis

### 3.1.1　ドキュメントの作成

トピック解析を行うためには最初にドキュメントを用意する必要がある．本研究では，バージョン管理システムの開発履歴からトピック解析で利用可能な単語を抽出し，その集合をドキュメントとする．抽出する対象はソースコード中の識別子名・コメント，バージョン管理システムへのコミットする際のログである．

識別子名は一般的にそのプログラムの動作に関係した用語が使われる．例えば GUI に関する処理を行うプログラムのコードであった場合，*width, height, window* などが含まれた識別子が頻出すると考えられる．また，コメントにはソースコード中のクラスやメソッドに関する説明が書かれる．識別子名やコメントの一般的な使われ方を考えると，識別子名とコメントからは活動の対象に関するトピックを得られることが期待できる．次にコミットログであるが，これは行われた活動に関する内容を簡潔に記述したものであるため，コミットログからは活動内容に関するトピックを得られることが期待できる．

本研究ではドキュメントを作成する単位をコミット単位とした．つまり，コミットに含まれる作成・編集されたソースコードとコミットログから 1 つのドキュメントを作成する．図 **2** はドキュメントの作成例を示しており，リビジョン@12 からリビジョン@13 へコミットした時，ドキュメント@12 が作られる．また，編集されたソースコードの場合でも単語を抽出するのはソースコード全体からであり，編集された前回からの差分箇所のみを対象に抽出は行わない．

### 3.1.2　トピックの解析

3.1.1 で作成されたドキュメントからトピックを抽出する．本研究では，全ドキュメントを 1 度だけ LDA の入力とする Hall モデル [10] を用いる．また，トピック抽出には LDA が実装された自然言語処理ツール MALLET [11] を用いる．LDA を適用すると，出力としてトピックの集合と入力したドキュメントのトピック分布が得られる．

トピックとは順位付けされた単語の集合である．順位が上位のキーワードがそのトピックを表す単語になる．例えば上位の単語が *os, cpu, memory* であった場合，その集合はソフトウェアの基幹部分に関するトピックである可能性が高い．

LDA を適用すると，各ドキュメントについて，トピック分布が得られる．ドキュメントのトピック分布は，全て

表 1　ドキュメントごとのトピック分布
**Table 1**　Distribution of topics for each document

| | トピック A | トピック B | トピック C |
|---|---|---|---|
| ドキュメント A | 0.2 | 0.1 | 0.7 |
| ドキュメント B | 0.1 | 0.8 | 0.1 |
| ドキュメント C | 0.4 | 0.0 | 0.6 |

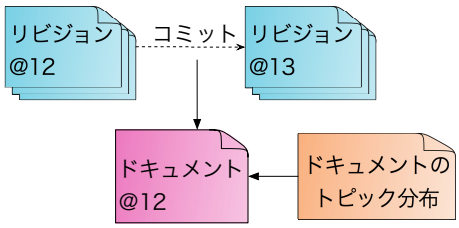



図 4　コミットとドキュメントのトピック分布の関連付け
**Fig. 4**　Linking commit and topic distribution

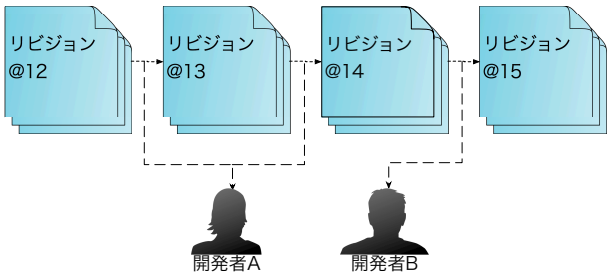



図 5　コミットを開発者ごとに分類
**Fig. 5**　Categorization of commits based on a developer

のトピックに対して，そのドキュメントがどのトピックに属しているのかを生起確率で表したものである．表 **1** を例とすると，ドキュメント A はトピック A について 0.2，トピック B について 0.1，トピック C について 0.7 の生起確率を持っている．この時，分布で最大の生起確率を持つトピック C がドキュメント A を表すトピックであると言える．

#### 3.1.3　可視化

可視化を行うには，最初にコミットと 3.1.2 で求めたドキュメントのトピック分布を関連付ける必要がある．これは 図 **4** のように両者に関連するドキュメントを通して容易に求めることができる．

次に開発者単位で可視化を行うために，コミットを開発者ごとに分類する必要がある．本研究ではファイルの作成・編集をした開発者とそれをリポジトリにコミットした開発者を同一人物であると仮定している．

開発者ごとにコミットを分類したら，更にそれをコミットの時期ごとに 1 つのまとまりを作り，トピックのポイントを計算する．これはトピックの変化を表すために必要である．ポイントはコミットに関連付けられたドキュメントのトピック分布の生起確率の値をそのまま用いる．例えば，表 **2** はある時期における開発者のトピック分布である．これらの値を同じトピックごとに合算すると 表 **3** のようになる．この合算値がその時期のトピックの活発具合を表すポイントになる．表 3 の場合，この時期にはトピック C に関する活動が大きく行われていたことが分かる．

最後に横軸をコミットの時期，縦軸を合算したポイントとしてトピックの変化を可視化する．

表 **2**　開発者のトピック分布
**Table 2**　Distribution of topics for a developer

| | トピック A | トピック B | トピック C |
|---|---|---|---|
| ドキュメント@12 | 0.1 | 0.2 | 0.7 |
| ドキュメント@13 | 0.0 | 0.4 | 0.6 |

表 **3**　トピックの分布の合算値
**Table 3**　Total amount of distributions of topics

| トピック A | トピック B | トピック C |
|---|---|---|
| 0.1 | 0.6 | 1.3 |

#### 3.1.4　活動の推定

活動の推定には抽出されたトピックが何を意味するのかを知ることと，トピックの変化のグラフが何を表しているかを知ることが必要になる．

トピックの意味を知ることはトピックに名前を付けることに等しい．この作業を簡潔にしたい場合はトピックに含まれる上位のキーワードを並べるだけでトピックに名前を付けることができる．しかし，それだけではただの単語の並びになってしまい，理解するのは困難である．そこで，上位キーワードからトピックが何を表しているのかを考えなければいけない．ただし，この作業にはプロジェクトに関わる知識を持った人が行うことが望ましい．抽出されたトピックのキーワードは専門用語が含まれる可能性が高く，それらを理解するにはある程度の知識が必要だからである．

次にグラフから活動を推定する．ポイントが大きいトピックは，そのトピックに関する活動が活発であったことを示す．これは本研究ではコミットを単位でドキュメントを作成しているため，コミットが頻繁であると合算する際のドキュメントのトピック分布が多くなり，全体的にポイントが大きくなるからである．また，ポイントを割合で見たとき，割合の大きいトピックがその時期に大きく関係するトピックであることを示す．

## 4.　ケーススタディ

本研究の提案手法をオープンソースソフトウェアの Columba [*1] に適用する実験を行った．本研究で用いた Columba のデータを 表 **4** に示す．また，本実験ではトピック解析で抽出するトピック数を 10 個とした．

表 **4**　Columba の統計情報
**Table 4**　Statistics of the Columba

| 種別 | メールクライアントソフト |
|---|---|
| 言語 | Java |
| 期間 | 2006/07 - 2012/05 |
| コミット数 | 327 回 |
| 開発者数 | 9 人 |

[*1] http://sourceforge.net/projects/columba/

### 4.1 適用結果

表 **5** に抽出された 10 個のトピックのキーワードを示す．本研究では，トピックの意味を上位のキーワードから第一著者が推定した．例えば，トピック B の上位キーワードは *calendar, event, range, date* である．Columba がメールクライアントソフトであることを考えると，このトピック B はカレンダーの処理に関するトピックであると考えられる．また，トピック F のキーワードを見ると *folder, list, message, filter, mail* などが見られるため，トピック F はメッセージフィルタの処理に関するトピックであると考えられる．

図 **6(a)** はプロジェクト全体のトピックの変化を表したグラフである．時期のまとまりは 10 日間隔にした．グラフより，2006 年から 2007 年においてプロジェクトで様々な活動が行われていることがトピックの変化から分かる．例えば，2006 年 8 月中旬から下旬まではトピック J に関する活動が活発である．また，2006 年 9 月下旬から 10 月上旬で再びポイントが大きくなっていることが分かる．表 5 より，トピック J はプラグインに関するトピックであると考えられる．つまり，2006 年 8 月中旬ごろからプラグインに関する活動が集中して行われ，9 月下旬ではそれらの不具合を修正する作業が行われたと推定することができる．また，図 6(a) からは分かりづらいが，図 6(b) から 2006 年 12 月中旬から 1 ヶ月のトピックの割合を見ると，トピック J が同様の変化をしているのが見られる．これはポイントが小さいことから小規模の活動ではあるが，追加で新しいプラグインが追加されたと推定することができる．

図 **7** は開発者 eschman に関するドキュメントのトピック分布を用いて表したグラフである．グラフからこの開発者が 2007 年 4 月下旬から Columba プロジェクトに参加していることが分かる．図 **7(a)** を見ると 2009 年 11 月中旬でトピック B，2010 年 3 月中旬以降にトピック F でポイントが大きくなっていることが分かる．先述の通り，トピック B はカレンダー，トピック F はメッセージフィルタに関するトピックであると考えられるため，この開発者はそれらに関する活動を行っていたことが分かる．しかし，このグラフからは開発者 eschman がこのトピックに関するどのような活動を行っていたのかを知ることはできない．抽出されたトピックからは活動の内容を表すキーワードは無く，それを知ることができない．ただし，図 6(a), 7(a) より，2008 年以降はそれ以前よりも活動量が少なく，また，開発者 eschman の活動しか見られない．そのため，Columba プロジェクトはソフトウェアの保守作業の段階であると考えられる．つまり，ポイントの大きなトピック B や F は，そのトピックに関する修正作業であると推定することができる．

表 **5** 抽出トピックデータ
**Table 5** List of extracted topics

| | 上位トピックキーワード |
|---|---|
| トピック A | search, border, criteria, result, provider, icon |
| トピック B | calendar, event, range, date, util, start, end |
| トピック C | item, string, mail, message, selected, imap, box |
| トピック D | view, table, event, id, tag, folder, list |
| トピック E | dietz, stich, frederik, timo, list, id, param |
| トピック F | folder, uid, list, message, filter, header, mail |
| トピック G | button, action, loader, timo, frederik, dietz, stich |
| トピック H | message, header, html, list, body, assert, equals |
| トピック I | model, panel, field, label, controller, form, editor |
| トピック J | instance, manager, path, plugin, line, system |

## 5. 考察

### 5.1 抽出されたトピックの解釈

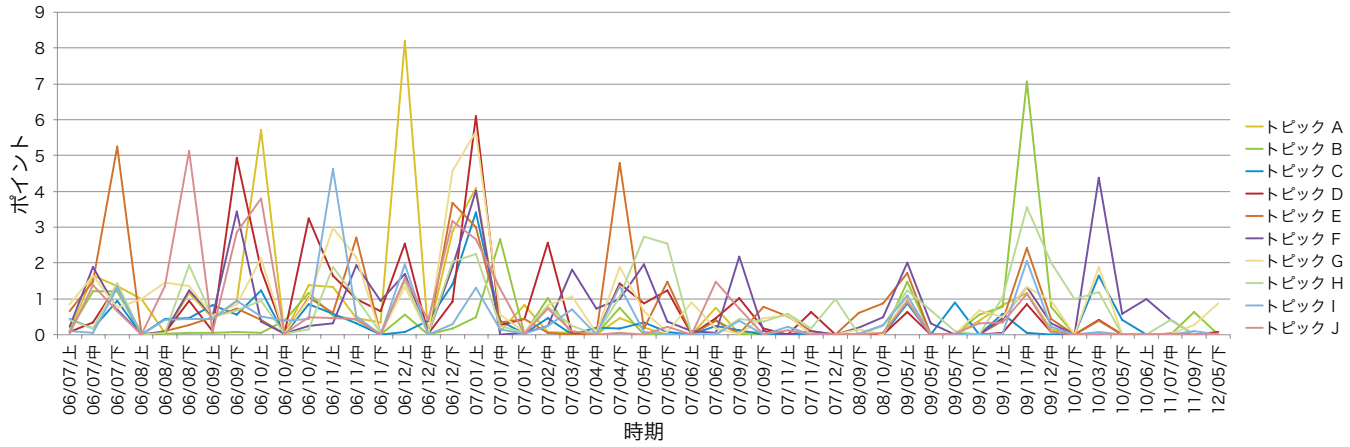
本研究ではトピックが開発者の活動を表していると仮定していた．しかし，今回抽出されたトピックは “活動” ではなくソフトウェアの “機能” が抽出されたと見られる．これはバージョン管理システムからソースコードの識別子を用いたことが原因であると考えられる．ソースコードの識別子は通常ソフトウェアの機能に関する用語が使われることが多い．そのため，多くの機能に関するトピックが抽出された．

それでは，活動のトピックを得るにはどうすれば可能かを考えると，ドキュメント作成の対象を拡大することが 1 つの解決になると考えている．バグ管理システムのコメントからやメーリングリストのデータには開発者間のやり取りが記録されている．それを使用することで，より開発者の活動に関するトピックが抽出できると期待される．また，機能のトピックが期待できるドキュメントと活動のトピックが期待できるドキュメントを分けて解析し可視化することで，機能と活動に関する繋がりを知ることが期待できる．

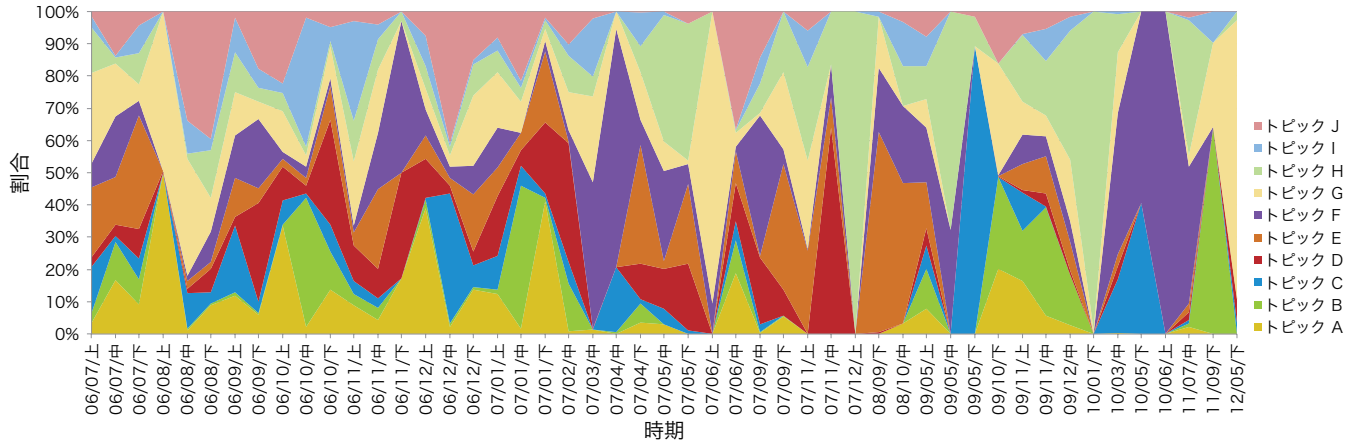
### 5.2 LDA に設定するトピック数の影響

今回は抽出するトピック数を 10 個としたが，これが妥当な数であるかは分からない．LDA は自由に抽出するトピック数を決めることが可能だが，ドキュメントのサイズやそこに含まれる単語数で妥当なトピック数を決めることができない．抽出するトピック数を変化させて結果を比較することは可能であるが，数が多くなるほど多くの工数を割くことになってしまう．また，トピック数が多くなると類似のトピックが表れることも考えられる．

そこで対策としてあらかじめソフトウェア開発で考えられる活動を複数決めておき，抽出されたトピックがどのような活動であるかマッピングする方法が考えられる．この方法であればトピック数を制限する必要がなくなる．




(a) ポイントを縦軸とした折れ線グラフ




(b) 100%積み上げ面グラフ

図 6　Columba のトピックの変化
**Fig. 6**　Evolution of topics in the project

## 6. おわりに

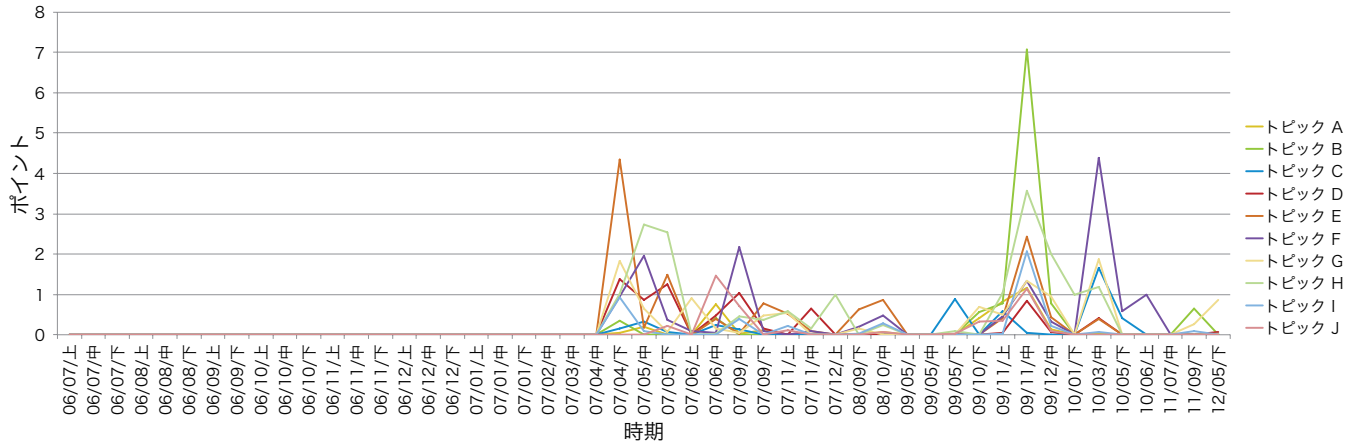
本研究では，バージョン管理システムに記録された開発履歴を用いて，開発者の作業内容の可視化を行う手法を提案した．バージョン管理システムのデータから抽出されたトピックには開発者の作業を行った対象に関するトピックが抽出することができた．しかし，作業内容を表すようなトピックを抽出することはできなかった．

今後の課題はバグ管理システムやメーリングリストのデータやその組み合わせからドキュメントを作成し，それらを用いて開発者の作業内容を表すトピックを抽出が可能かを確認することである．また，抽出するトピック数を決定する方法の検討や可視化の評価を行う必要があると考えている．

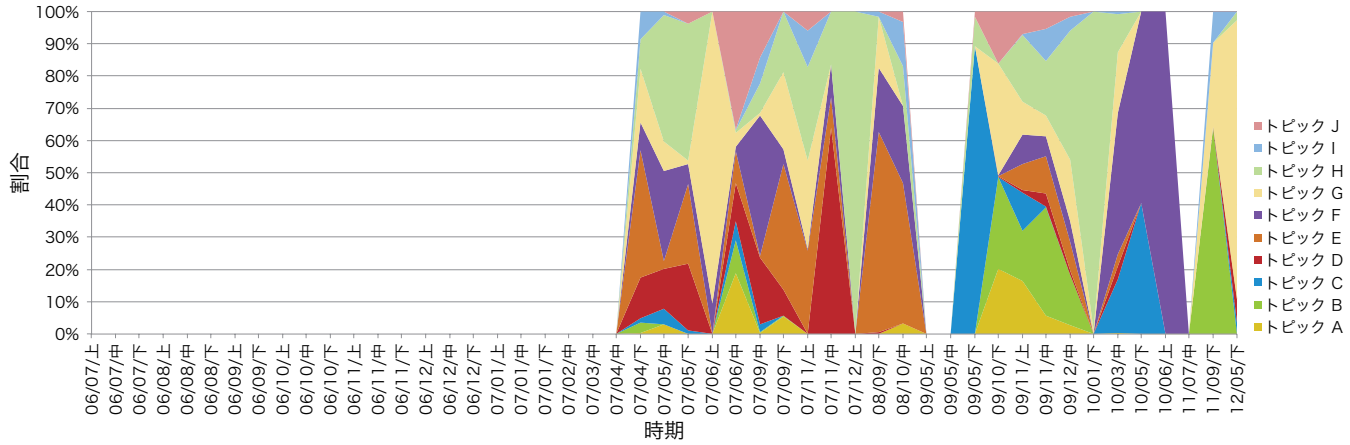
謝辞　本研究は，日本学術振興会 科学研究費補助金 基盤研究 (C) （課題番号:22500027）の助成を得た．

参考文献

[1] Humphrey, W. S.: Using A Defined and Measured Personal Software Process, *IEEE Software*, Vol. 13, No. 3, pp. 77–88 (1996).
[2] 池田　寛，有賀　淳，藤原克則，渡辺洋司：ゼロからはじめる PSP — Personal Software Process，エンジニアマインド，No. 4 (2008). http://gihyo.jp/magazine/em/archive/no004.
[3] 山口雅史：Personal Software Process（PSP）の実施の定着化，ソフトウェアプロセス改善カンファレンス 2009, (SPI Japan '09) (2009). http://www.jaspic.org/event/2009/SPIJapan/session2C/2C2.pdf.
[4] Ohkura, K., Goto, K., Hanakawa, N., Kawaguchi, S. and Iida, H.: Project Replayer with email analysis – revealing contexts in software development, *Proceedings of the 13th Asia Pacific Software Engineering Conference (APSEC '06)*, pp. 453–460 (2006).
[5] 大蔵君治，後藤慶多，川口真司，花川典子，飯田　元：ソフトウェア開発における知識還元のためのプロジェクト再現ツール，ソフトウェアエンジニアリング最前線 2006, pp. 75–78 (2006).
[6] Blei, D. M., Ng, A. Y. and Jordan, M. I.: Latent dirichlet allocation, *J. Mach. Learn. Res.*, Vol. 3, pp. 993–1022 (2003).
[7] Hindle, A., Godfrey, M. W. and Holt, R. C.: Soft-




(a) ポイントを縦軸とした折れ線グラフ




(b) 100%積み上げ面グラフ

図 7　開発者 eschman のトピックの変化

**Fig. 7**　Evolution of topics related with a developer eschman

ware Process Recovery using Recovered Unified Process Views, *Proceedings of the 26th IEEE International Conference on Software Maintenance*, (ICSM '10) (2010).

[8] Bernardi, M. L., Sementa, C., Zagarese, Q., Distante, D. and Di Penta, M.: What Topics do Firefox and Chrome Contributors Discuss?, *Proceedings of the 8th Working Conference on Mining Software Repositories*, (MSR '11), pp. 234–237 (2011).

[9] Chen, T.-H., Thomas, S., Nagappan, M. and Hassan, A.: Explaining software defects using topic models, *Proceedings of the 9th Working Conference on Mining Software Repositories (MSR '12)*, pp. 189–198 (2012).

[10] Hall, D., Jurafsky, D. and Manning, C. D.: Studying the history of ideas using topic models, *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, (EMNLP '08), pp. 363–371 (2008).

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