修士論文

GitHub における Issue と Pull Request を対象としたトピック抽出とその傾向の考察

中西 駿太

2019 年 1 月 31 日

奈良先端科学技術大学院大学
情報科学研究科

本論文は奈良先端科学技術大学院大学情報科学研究科に
修士 (工学) 授与の要件として提出した修士論文である。

中西 駿太

審査委員：

飯田 元 教授	（主指導教員）
松本 健一 教授	（副指導教員）
市川 昊平 准教授	（副指導教員）
崔 恩瀞 助教	（副指導教員）

GitHubにおけるIssueとPull Requestを対象としたトピック抽出とその傾向の考察*

中西 駿太

内容梗概

近年，企業の開発現場においてOSS(オープンソースソフトウェア)の活用が盛んになってきている．しかし，企業における開発とは異なるOSSの特徴的な開発形態は長期に渡って健全なプロジェクト運営を行うことを困難にしている．これは企業がOSSを活用する上での大きなデメリットとなっている．厳格な指揮系統が存在しないOSSの開発において，健全なプロジェクト運営が行われているかを判断することは困難である．そこで，本研究ではOSSの開発プロセスにおいて，開発者間で行われるコミュニケーションに着目した．IssueとPull Requestの間で議題が一貫しており，遅延が少ない場合，そのプロジェクトは比較的健全な運営がなされていると考えられる．しかし，IssueやPullRequesの量は多く，どのような議論を行っているか，すべてを人の手で確認するのは困難である．そこで本研究ではIssueやPull Request上の議論からトピックを抽出し，時系列のグラフとして可視化した．その結果，一部のトピックでプロセス間の連携を確認できた．また，得られたトピックについて分析を行った結果，本手法が有効であることがわかった．

キーワード

リポジトリマイニング, オープンソースソフトウェア, トピックモデル, 可視化

*奈良先端科学技術大学院大学 情報科学研究科 修士論文, 2019年1月31日.

Topic extraction for Issue and Pull Request in GitHub and Consideration of its Tendency*

Shunta Nakanishi

Abstract

In recent years, utilization of OSS (open source software) is becoming popular at companies' development sites. However, the distinctive development form of the OSS different from the development in the company makes it difficult to manage a sound project for a long time. This is a major disadvantage for companies to utilize OSS. It is difficult to judge whether sound project management is being carried out in the development of OSS without strict command system. In this research, therefore, we focused on communication between developers in OSS development process. If the agenda is consistent between Issue and Pull Request and the delay is small, the project is considered to be relatively healthy. However, the amount of Issue and PullReques are many, and it is difficult to confirm all the discussions with people by hand. In this study, topics were extracted from discussions on Issue and Pull Request and visualized as a time series graph. As a result, we were able to confirm the cooperation among processes on some topics. As a result of analyzing the obtained topic, we found that this method is effective.

Keywords:

Mining software repositories, Open Source Software, Topic Model, Visualzation

*Master's Thesis, Graduate School of Information Science, Nara Institute of Science and Technology, January 31, 2019.

目 次

図 目 次

表 目 次

1　はじめに

近年，企業の開発現場においてOSS(オープンソースソフトウェア) の活用が盛んになってきている．また，利用されるOSSの形態もオペレーティングシステム(OS) やプログラミング言語，開発環境，ライブラリなど様々である．例えば，複数の企業がスマートフォン用のOSで有名なOSSであるAndroidを自社のスマートフォンの開発に活用している．　企業がOSSを活用するメリットとして，以下が挙げられる [8][4].

- 開発コストを低減できる
- 特定のベンダーに依存しない
- 最新技術を早期に取り込むことができる
- 実装が公開されているため，独自に拡張できる

その一方で，デメリットとして以下の回答が多くなっている．

- 緊急時の技術的サポートが得にくい
- バグの回収や顧客からの要請対応に手間がかかること
- 将来のバージョンアップの計画が把握しにくいこと
- セキュリティホールに対するコミュニティの対応に不安がある

このようなデメリットが挙げられる要因としてOSSの特徴的な開発形態が考えられる．OSSの開発は企業による開発とは異なり，以下のような特徴を有している [6].

- 開発者が自由に参加離脱することが可能
- ボランティアでの参加が基本
- 厳格な指揮系統が存在しない

- ネットワークを介した分散開発環境

このため，長期に渡って健全なプロジェクト運営を行うことは困難と言える．健全な運営がなされていないプロジェクトでは，ユーザーからのフィードバックが成果物に取り入れられにくく，また，場合によってはプロジェクトが分裂や消滅する恐れがある．

そこで，本研究では健全性の目安として開発プロセスで行われる開発者間のコミュニケーションに着目した．OSS 開発の各プロセスで行われるコミュニケーションに対しトピック抽出を行い，その結果を可視化した．また，その考察を行った．

2　背景

2.1　OSS の開発プロセス

OSS の開発は一般に，図 1 のようにリクエスト，開発，レビューの 3 つのプロセスからなる．リクエストのプロセスは課題管理システム上で，レビューのプロセスは版管理ホスティングサービス上でそれぞれ行われる．GitHub[1]はリクエストに関する機能は Issue として，レビューに関するの機能は Pull Request として備えており，多くの OSS プロジェクトで利用されている．

[1]http://github.com

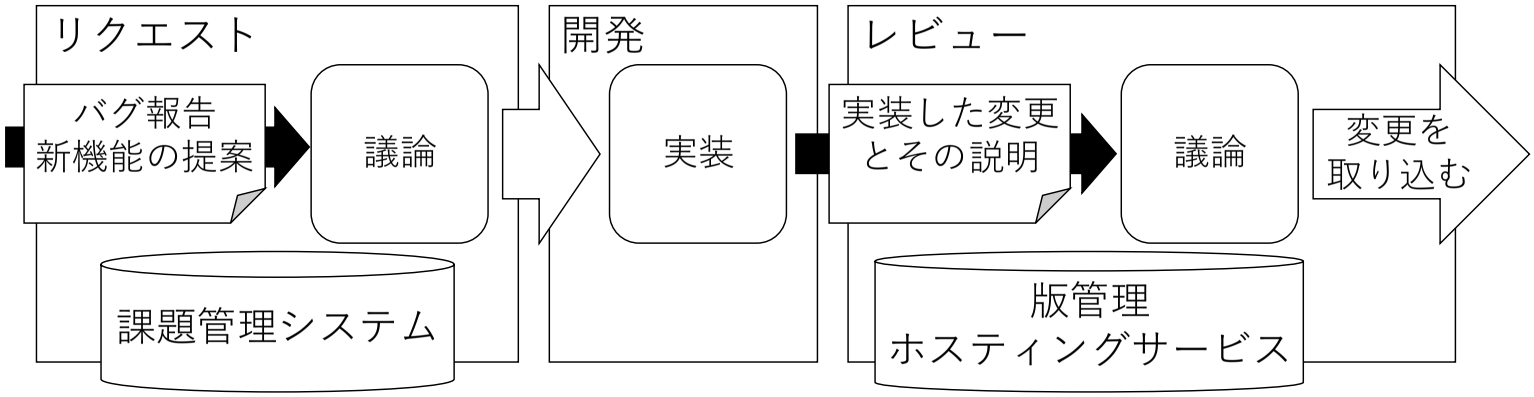

図 1　OSS の開発プロセス

リクエスト ユーザ及びOSS開発者からバグ報告や新機能の提案などが課題管理システム上にIssueとして投稿される．このIssueをOSSへ取り入れるかどうかをは，そのIssueに対してOSS開発者がコメントを投稿することで議論し，決定する．OSSへ該当Issueが取り込まれる場合には，実装の優先順位や実装方法，担当者などについても議論することがある．図2にGitHubにおけるIssueの例を示す．最初にユーザ及びOSS開発者がバグ報告や新機能の提案を添えてIssueをたて，それに対して他のOSS開発者が返信する形で議論が行われている．

開発 上述の議論に基いて実装する．小規模な変更であれば，リクエストプロセスを経ずに実装されることもある．

レビュー 開発者が実装した変更とその変更に対する説明を版管理ホスティングサービスにPull Reqestとして投稿する．その後，レビュアーがその変更をレビューする．レビューの結果，変更が承認されなかった場合，開発者はレビューコメントに基いて修正を繰り返す．また，変更が承認された場合は，その変更はOSSに反映される．このようにレビュープロセスを通して開発者の変更をOSSへ反映しても問題無いかが議論され，問題なければその変更がOSSへ取り込まれる．図3にGitHubにおけるPull Requestの例を示す．最初に開発者が実装した変更を版管理システムGitのコミットの形でつけて，その変更に対する説明を添えてPull Requestをたてる．開発者はそれに返信する形で，該当する適宜ソースコードを参照しながらレビューコメントを投稿する形で議論が行われている．

2.2 トピック分析技術

トピックモデルを用いて各文書のトピックを同じトピックを持つ文書に現れやすい単語とその生起確率の集合として求める．トピックモデルとは，1つの文書が複数のトピックを持つと仮定した言語モデルである． [5]

図 2　GitHub における Issue の例

図 3　GitHub における Pull Request の例

2.3 開発活動の可視化に関する研究

2.3.1 Hindle らの研究

Hindle らはソフトウェアの成果物およびリポジトリから開発プロセスを半自動化に測定し，可視化する手法を提案している [2]．彼らの手法は，Word-bags 解析およびトピック抽出を用いてコミットの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，可視化する．

2.3.2 山田らの研究

山田らは，版管理システムのコミットログからトピック抽出行い割当値を算出し，トピックを可視化するツールを開発した [7]．また，開発した可視化ツールと git や grep などの UNIX コマンドを比較する被験者実験を実施し，トッピング可視化ツールの有用性を確認した．

3　可視化手順

本研究で提案するツールは，リクエストやレビュープロセスで行われた議論からそれぞれのトピックを抽出し，そのトピックごとに整理し可視化する．

具体的な可視化の手順を図 4 に示す．以降，各手順の詳細について述べる．

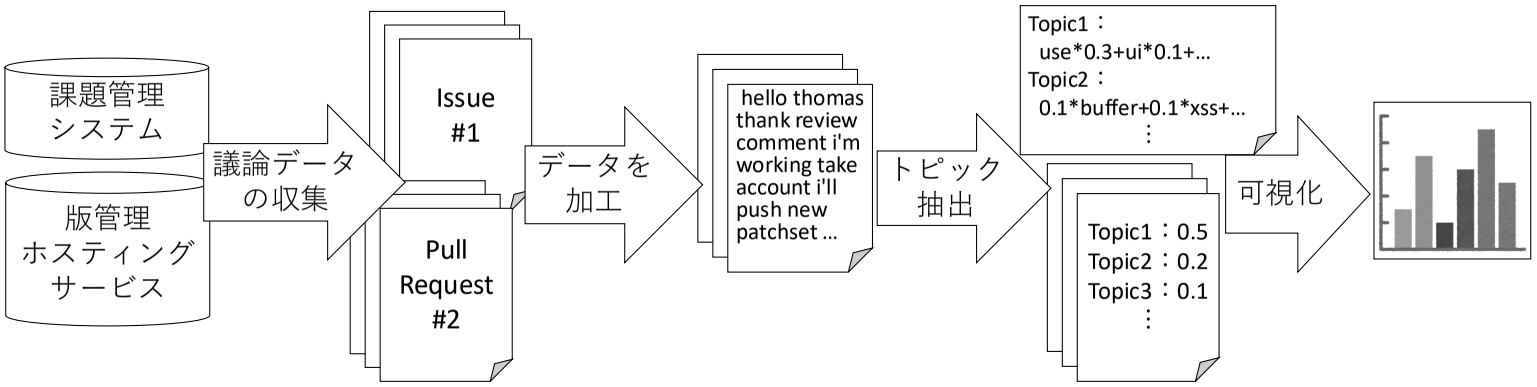
課題管理
システム
版管理
ホスティング
サービス
議論データ
の収集
Issue
#1
Pull
Request
#2
データを
加工
hello thomas
thank review
comment i'm
working take
account i'll
push new
patchset ...
トピック
抽出
Topic1：
use*0.3+ui*0.1+...
Topic2：
0.1*buffer+0.1*xss+...
Topic1：0.5
Topic2：0.2
Topic3：0.1
可視化

図 4 可視化の手順

3.1 議論データの収集

課題管理システム上のIssue及び，版管理ホスティングサービス上のPull Requestを収集する．1つのIssue，Pull Requestをスレッドと呼ぶ．精度向上のため，議論が終わったことを示すClosed状態のスレッドまたは，Pull Requestにおいて変更が取り込まれたことを示すMerged状態のスレッドのみを収集の対象とする．1つのスレッドは複数の人から投稿されるメッセージでなる．議論のデータはメッセージ単位でHTML形式で収集する．

3.2 議論データの前処理

3.2.1 不要な要素の除去

投稿の中にはURLやファイルパス，ソースコード片などが含まれる．しかし，これらは議論のトピックを抽出する上でノイズとなるので除去する必要がある．これらをHTMLのタグを頼りに削除する．

3.2.2 語の正規化

3.1節で収集した議論データは，自然言語で記述されたものであるため，大文字小文字のや動詞の活用，複数形など，意味上の大きな違いが無いにもかかわらず，異なる表記がなされている．これらを正規化し同一に扱うことでトピック抽出の精度の向上が期待できる．ここでは，小文字への統一と見出し語化を行う．見出し語化では，Python用の自然言語処理ライブラリであるNLTK(Natural Language Toolkit) [2] で提供されているLemmatizerを利用する．

3.2.3 ストップワードの除去

次にストップワードの除去による効率化を行う．ストップワードとは，「a」，「the」，「to」などの頻繁に出現するが文書を特徴づけるような意味を持たない，

[2]Natural Language Toolkit : http://www.nltk.org/

分析する上で役に立たない単語のことである．本研究では，辞書による除去と出現頻度による除去の 2 通りの方法で除去する．辞書による除去では，Python 用の自然言語処理ライブラリである NLTK(Natural Language Toolkit) で提供されている辞書を利用し，一致した単語を文書から取り除く．出現頻度による除去では，全文書のうち 50%以上の文書に含まれている単語を文書から取り除く．

3.3 トピック抽出

メッセージをスレッドごとに結合したテキストを文書と呼ぶ．トピックはこの文書ごとに求める．

トピックモデルとして代表的なものに，潜在的ディリクレ配分法 (LDA: Latent Dirichlet Allocation)[1] がある．これを用いることで，1 つのスレッドで複数の議論が行われていたとしても対応できる．

3.4 可視化

3.4.1 LDAvis

得られたトピックについて理解しやすくするため，LDAvis[3][3] を利用した．

3.4.2 kibana による可視化

得られたトピックについて時系列で可視化を行う．ここでは Kibana[4] という可視化ツールを利用して図 5 のような全トピックについて積み上げたグラフと，図 6 のような個々のトピックについて，Issue と Pull Request を比較できるグラフを作成した．

[3]https://github.com/cpsievert/LDAvis
[4]https://www.elastic.co/jp/products/kibana

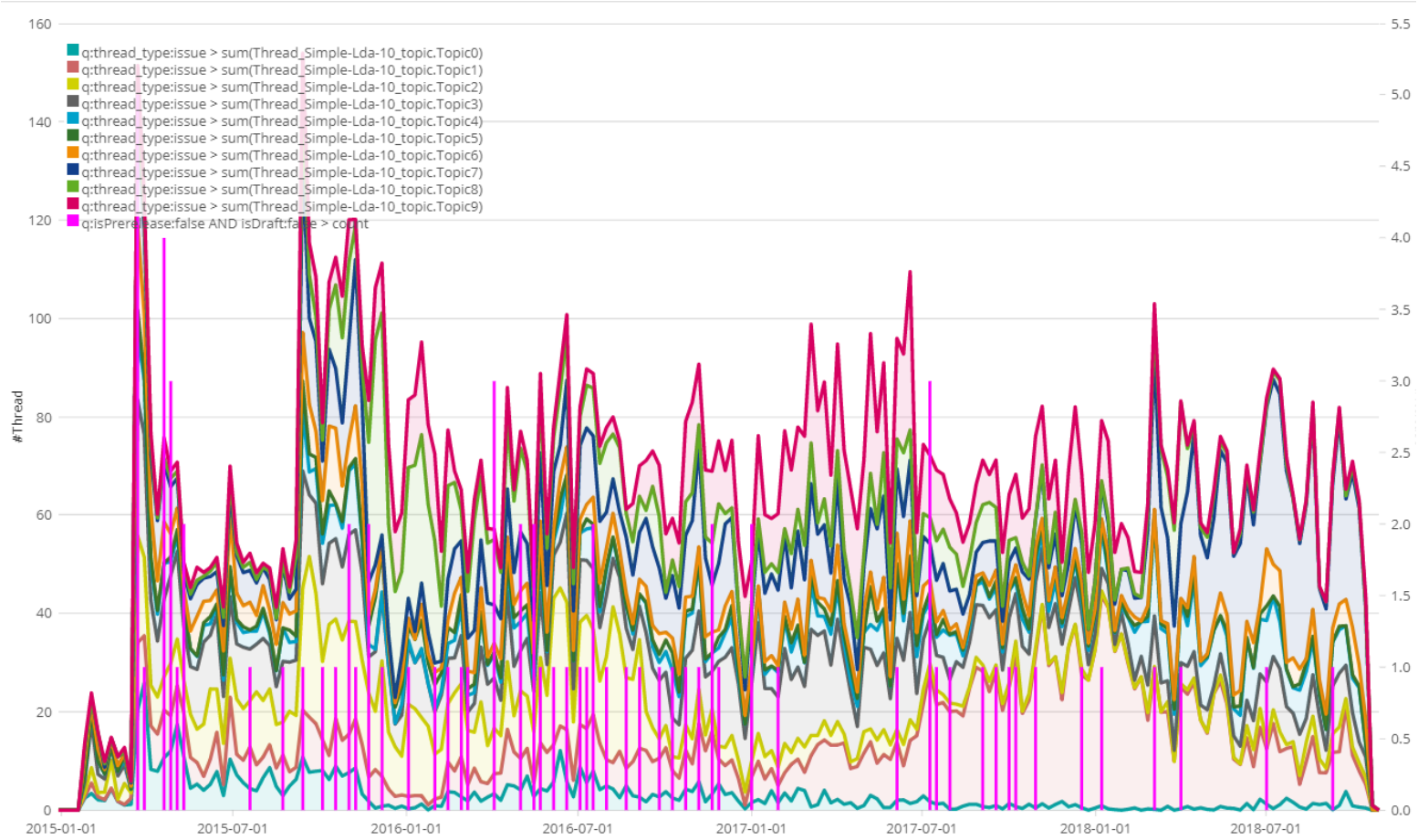

図 5 全トピックのグラフ

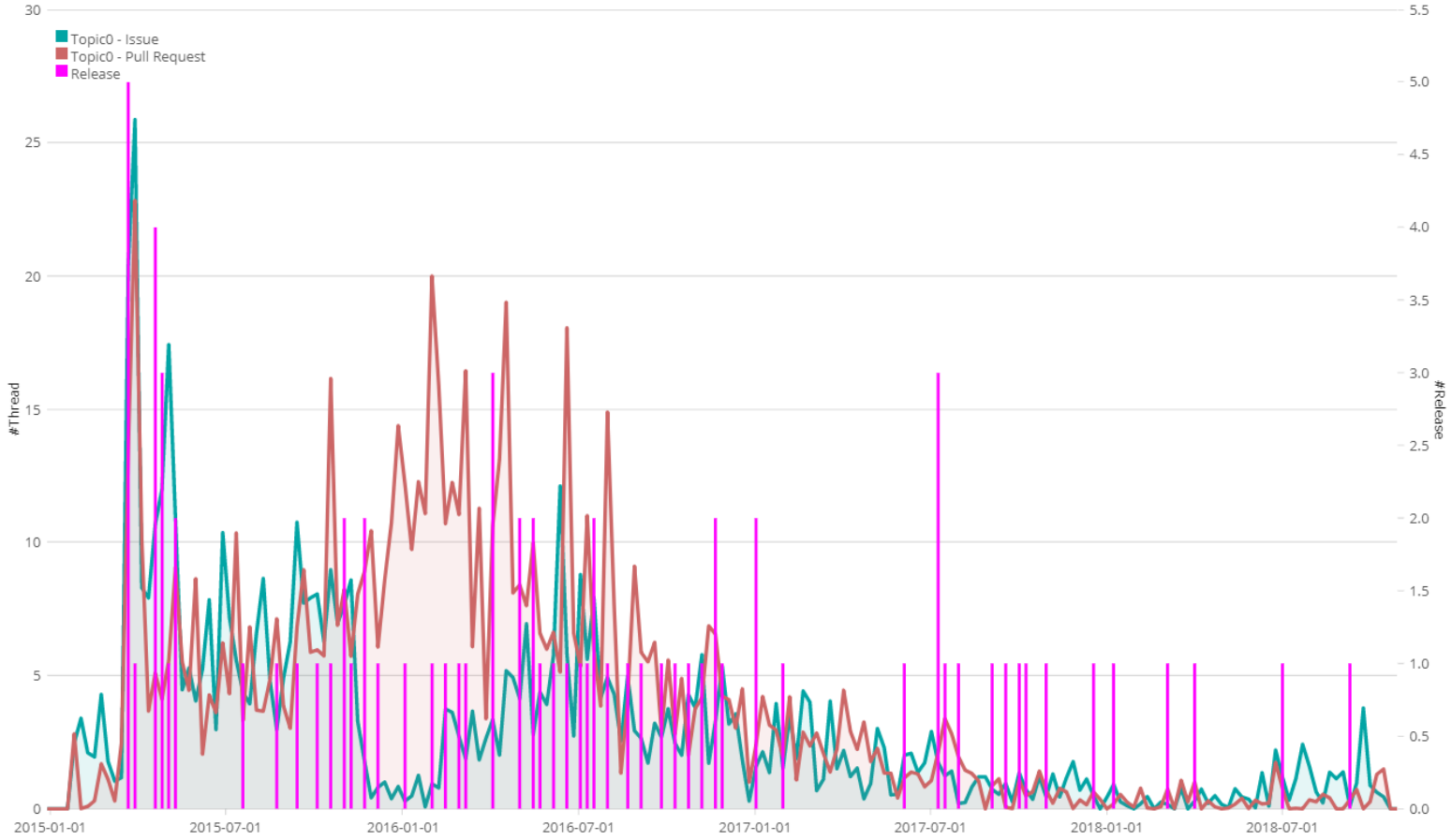
Topic0 - Issue
Topic0 - Pull Request
Release
#Thread
#Release
30
25
20
15
10
5
0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
2015-01-01
2015-07-01
2016-01-01
2016-07-01
2017-01-01
2017-07-01
2018-01-01
2018-07-01

図 6 個々のトピック

4　ケーススタディ

4.1　対象プロジェクト

GitHub を活用している OSS のうち，React Native[5]を対象に 3 章で述べた手順で可視化及び分析を行った．React Native は Facebook 社が開発している，クロスプラットフォームなモバイルネイティブアプリを開発するための JavaScript フレームワークである．

調査は，GitHub 上のリポジトリに対して最初の投稿がされた 2015 年 01 月 28 日から 2018 年 10 月 21 日までの間に投稿されたものを対象に行った．表 1 に調査対象となった議論データの件数を示す．なお，この期間内に成果物のリリースが 76 回行われていた．

表 1　調査対象となった議論データ (単位:件)

	スレッド数	メッセージ数
Issue	13,595	95,260
Pull Request	7,210	68,817
合計	20,805	164,077

4.2　結果と考察

以下のようなトピックが得られた．

トピック 1 facebook 社員にでないと merge できない Pull Request

トピック 2 他のコンポーネントが絡む修正

トピック 3 ビルド時に必要な外部ライブラリに関するエラー

[5]公式サイト：https://facebook.github.io/react-native/
GitHub リポジトリ：https://github.com/facebook/react-native

トピック 4 UI に関する機能の要望

トピック 5 外部ライブラリの import に関するエラー

トピック 6 実装方法の比較

トピック 7　android に依存するもの

トピック 8　他のライブラリとの連携

トピック 9 バージョンが変わったあとのビルドエラー

トピック 10 使い方の質問など

図 7 から得られた 10 個のトピックは，大きく 4 つ分けることができる．トピック 2，3，5，9 はビルドに関連するスレッドが見られた．トピック 4，6，7．8 では主に機能面に関するスレッドが見られた．

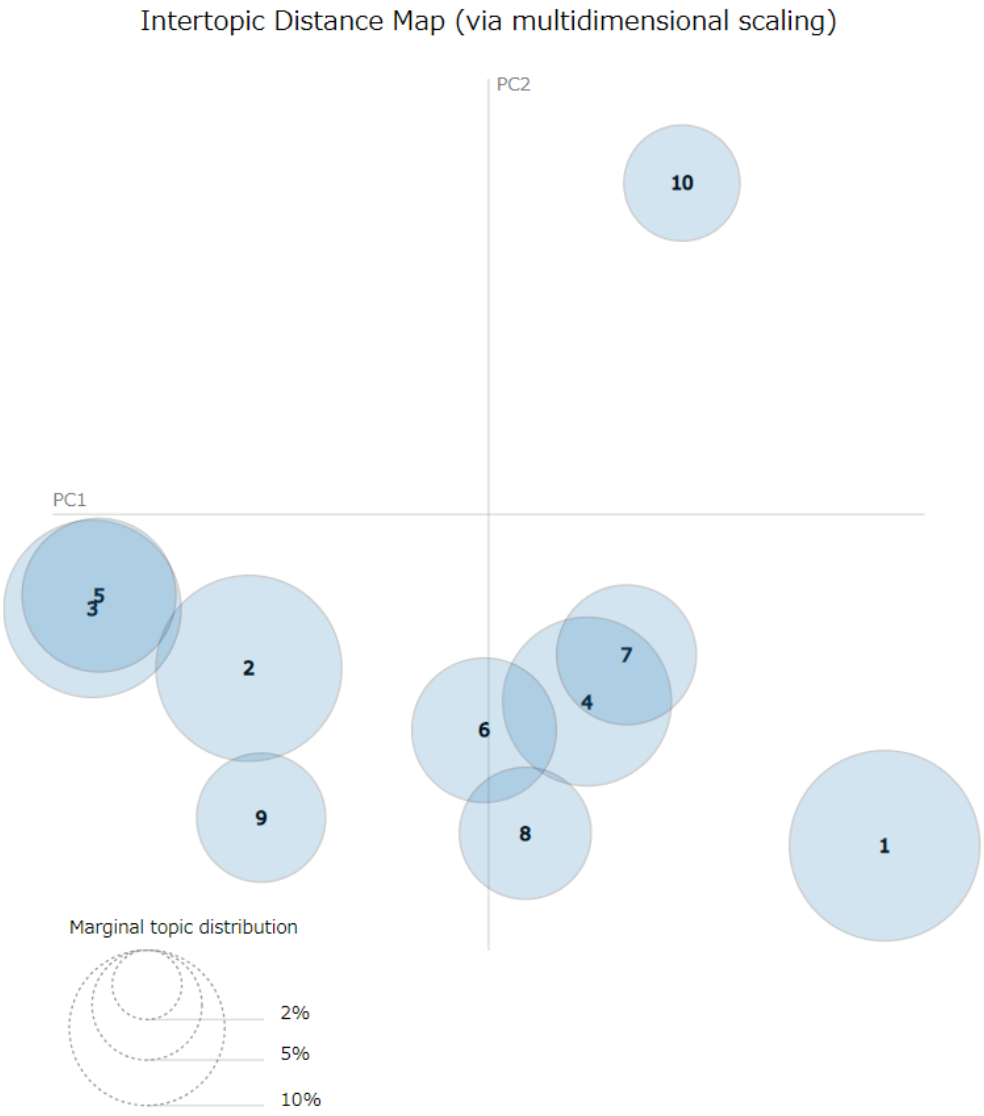

図 7　トピックの分布

また，トピック 4，トピック 6，トピック 8 では，それぞれ図 8，図 9，図 10 に示すように，Issue の推移と Pull Request の推移で重なりが見られた．これらのトピックでは，推移が重なっている時期において Issue と Pull Request の連携が図られていると考えられる．

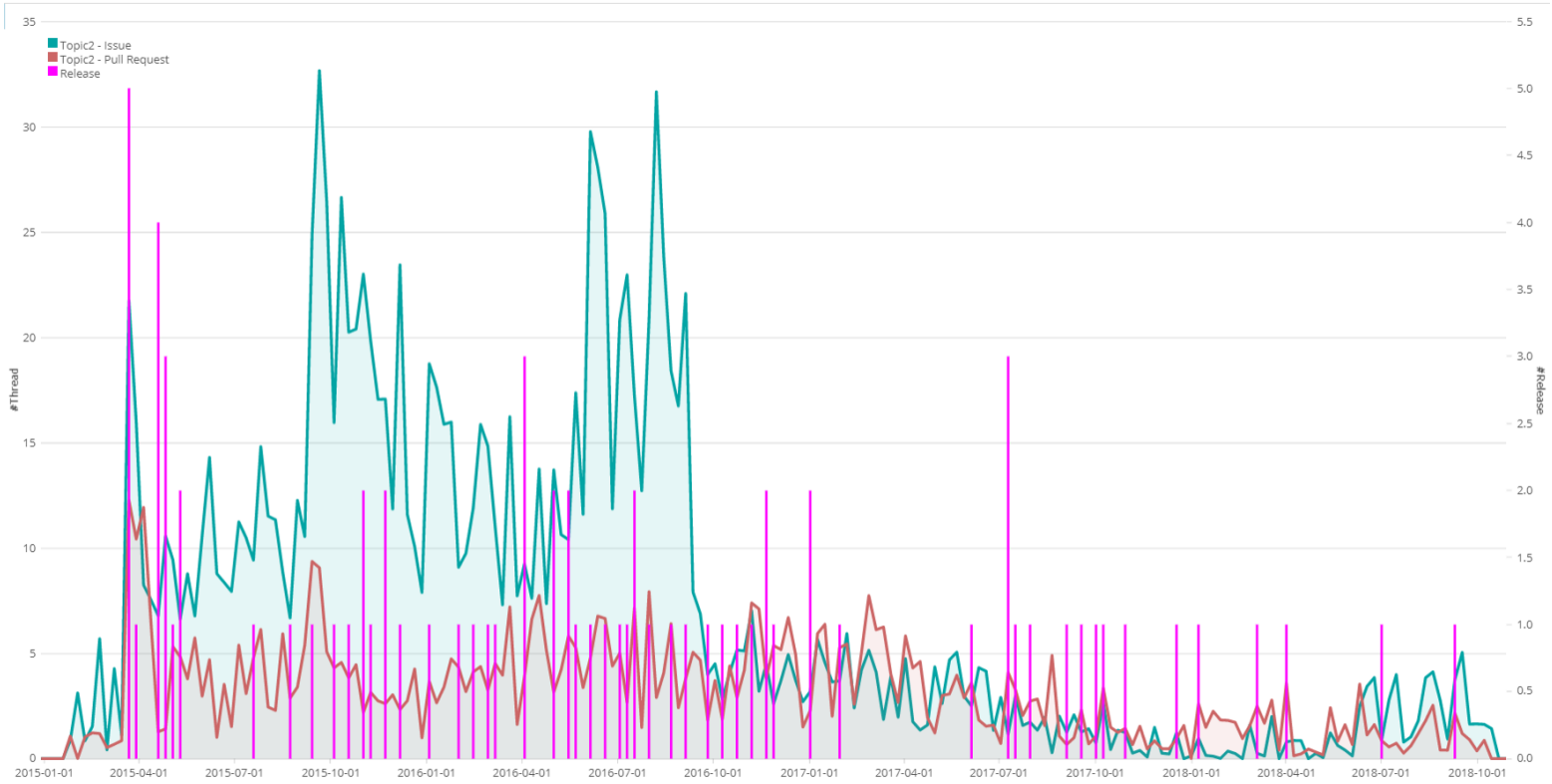
Topic2 - Issue
Topic2 - Pull Request
Release
#Thread
#Release
35
30
25
20
15
10
5
0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
2015-01-01
2015-04-01
2015-07-01
2015-10-01
2016-01-01
2016-04-01
2016-07-01
2016-10-01
2017-01-01
2017-04-01
2017-07-01
2017-10-01
2018-01-01
2018-04-01
2018-07-01
2018-10-01

図 8　トピック 4

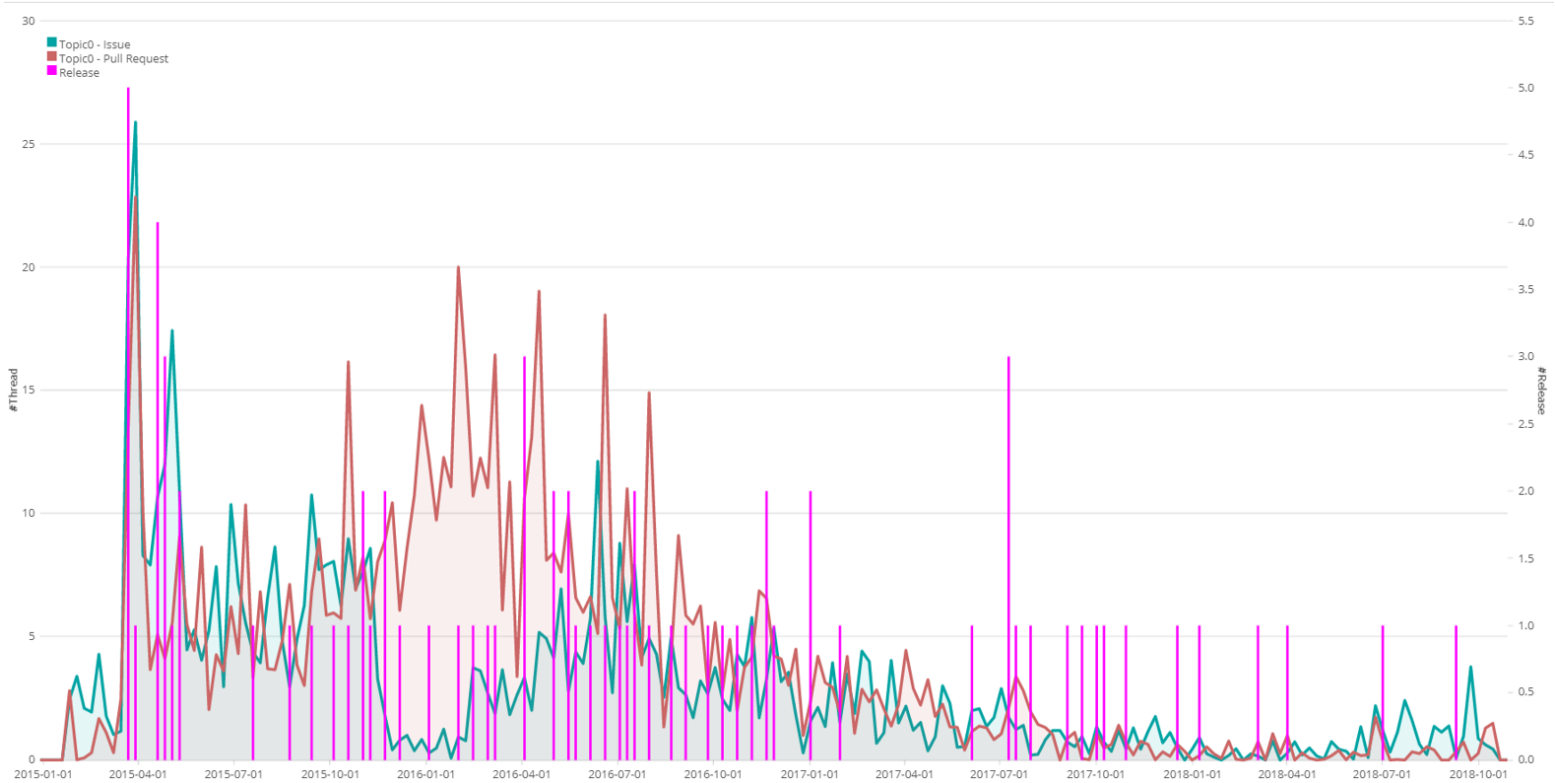
Topic0 - Issue
Topic0 - Pull Request
Release
#Thread
#Release
30
25
20
15
10
5
0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
2015-01-01
2015-04-01
2015-07-01
2015-10-01
2016-01-01
2016-04-01
2016-07-01
2016-10-01
2017-01-01
2017-04-01
2017-07-01
2017-10-01
2018-01-01
2018-04-01
2018-07-01
2018-10-01

図 9　トピック 6

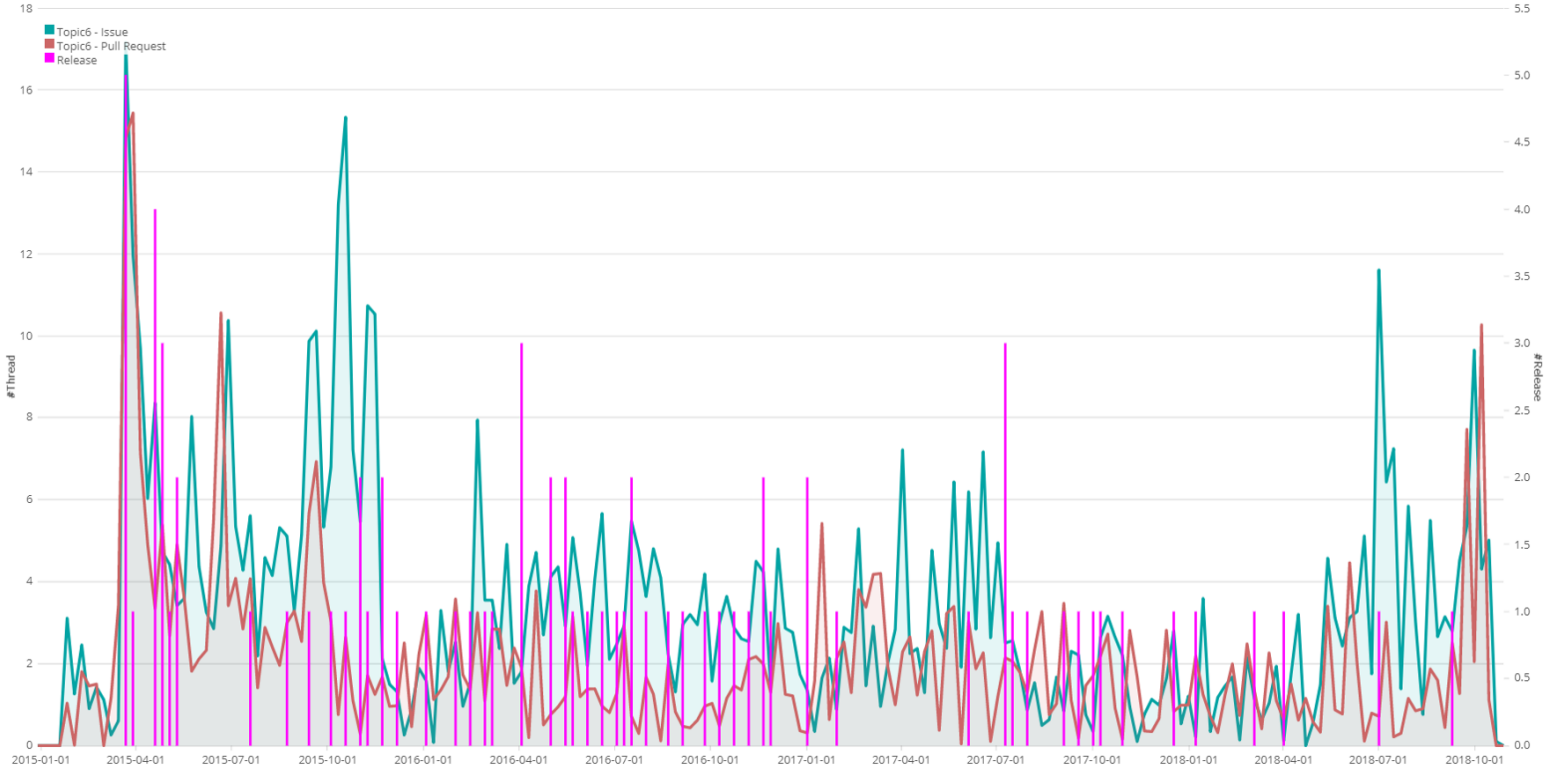
Topic6 - Issue
Topic6 - Pull Request
Release
#Thread
#Release
0
2
4
6
8
10
12
14
16
18
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
2015-01-01
2015-04-01
2015-07-01
2015-10-01
2016-01-01
2016-04-01
2016-07-01
2016-10-01
2017-01-01
2017-04-01
2017-07-01
2017-10-01
2018-01-01
2018-04-01
2018-07-01
2018-10-01

図 10　トピック 8

5　終わりに

5.1　まとめ

厳格な指揮系統が存在しない OSS の開発において，健全なプロジェクト運営が行われているかを判断することは困難である．そこで，本研究では OSS の開発プロセスにおいて，開発者間で行われるコミュニケーションに着目した．開発プロセスの間で一貫性のあるコミュニケーションが少ない遅延で取られている場合，そのプロジェクトは比較的健全な運営がなされていると考えられる．日々かわされる膨大かつ複雑なコミュニケーションに対しトピックを抽出し，時系列でグラフにすることで可視化した．その結果，一部のトピックでプロセス間の連携を確認できた．また，得られたトピックについて分析を行った結果，本手法が有効であることがわかった．

5.2　今後の課題

今後はケーススタディの追加による検証が必要となる．また，より定量的な評価を行い，本研究の手法で OSS プロジェクトの健全性を示すメトリクスとして提案することを目指す．

謝辞

本研究を進めるにあたり，多くの方々に御指導，御協力，御支援を頂きました．深く感謝いたします．

奈良先端科学技術大学院大学 情報科学研究科 飯田 元 教授には，本研究において熱心な御指導を賜りました．また常日頃から研究活動や学生生活に関しても多くの御助言を頂きました．心より厚く御礼を申し上げます．

奈良先端科学技術大学院大学 情報科学研究科 松本 健一 教授には，本研究に対し，大変貴重な御意見を賜りました．心より厚く御礼を申し上げます．

奈良先端科学技術大学院大学 情報科学研究科 市川 昊平 准教授には，本研究に対し貴重な御指導を頂きました．また常日頃から研究活動や学生生活に関しても多くの御助言を頂きました．心より感謝申し上げます．

奈良先端科学技術大学院大学 情報科学研究科 崔 恩瀞 助教には，本研究を設定する段階から熱心な御指導を頂きました．また常日頃から研究活動や学生生活に関しても多くの御助言を頂きました．心より感謝申し上げます．

奈良先端科学技術大学院大学 小川 暁子 様，仲田 綾奈 様には研究の遂行に必要な事務処理など，多岐にわたりご助力いただきました．心より感謝申し上げます．

奈良先端科学技術大学院大学 情報科学研究科 ソフトウェア設計学研究室の皆様には日頃より多くの御協力を頂きました．日常の議論を通して多くの御助言や示唆を頂き，多くのことを学ばさせていただきました．また，研究だけでなく，日頃の生活において支えていただきました．特に，博士後期課程 3 年の上村 恭平 氏には，非常に楽しい時間を過ごさせていただきました．心より感謝申し上げます．

香川大学大学院 工学研究科 信頼性情報システム工学専攻 博士前期課程 2 年の河村 宗一郎 氏には，本研究を進めるにあたり，自然言語処理に関して知識が乏しかった私に，多くの助言をいただきました．深く感謝いたします．

参考文献

[1] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[2] A. Hindle, M. W. Godfrey, and R. C. Holt. Software process recovery using recovered unified process views. In *Proc. of ICSM*, pages 1–10, 2010.

[3] Carson Sievert and Kenneth Shirley. LDAvis: A method for visualizing and interpreting topics. In *Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces*, pages 63–70, Stroudsburg, PA, USA, 2014. Association for Computational Linguistics.

[4] 野村 佳秀, 木村 功作, 福寄 雅洋, and 谷田 英生. 『オープンソースソフトウェア工学』シリーズ企業における OSS 活用の実際. **コンピュータ ソフトウェア**, 33(3):3_50–3_65, 2016.

[5] 岩田 具治. **トピックモデル** = *Topic models*. MLP 機械学習プロフェッショナルシリーズ. 講談社, 2015.

[6] 伊原 彰紀 and 大平 雅雄. 『オープンソースソフトウェア工学』シリーズオープンソースソフトウェア工学. **コンピュータ ソフトウェア**, 33(1):1_28–1_40, 2016.

[7] 山田 悠太, 吉田 則裕, 藤原 賢二, and 飯田 元. 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. **コンピュータ ソフトウェア**, 31(2):2_144–2_150, 2014.

[8] 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ェア活用ビジネス実態調査, 2010.