社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

OSS開発における品質に関する議論の可視化に向けて

中西　駿太† 　崔　　恩瀞† 　飯田　　元†

† 奈良先端技術大学院大学情報科学研究科　〒 630–0192 奈良県生駒市高山町 8916–5
E-mail: †{nakanishi.shunta.nl7,choi}@is.naist.jp, ††iida@itc.naist.jp

あらまし　ソフトウェアの開発コストを削減するため，企業のソフトウェア開発現場においてOSS(オープンソースソフトウェア) の導入が盛んである．しかし，品質の低いOSSを導入すると開発コストの増加を招く恐れがある．これを防ぐため，導入の対象となるOSSの品質を事前に把握する必要があるが，OSSの品質を公開情報から把握することは困難である．本研究ではOSS開発における意思決定プロセスやレビュープロセスで行われた，OSS開発者による品質に関する議論を可視化するシステムを開発する．このシステムによって企業でOSS導入する際に品質情報の把握に役立つことが期待できる．

キーワード　ソフトウェア品質，可視化，トピックモデル，オープンソースソフトウェア

Toward Visualization of Discussions about Software Quality in OSS Development

Shunta NAKANISHI†, Eunjong CHOI†, and Hajimu IIDA†

† Graduate School of Information Science, Nara Institute of Science and Technology　8916–5, Takayama, Ikoma, Nara, 630–0192,Japan
E-mail: †{nakanishi.shunta.nl7,choi}@is.naist.jp, ††iida@itc.naist.jp

Abstract　To reduce software development cost, Open Source software (OSS) is frequently used in software development companies. However, using OSS with low software quality increase development cost. To prevent this problems, it is necessary to grasp the quality of target OSS in advance, however, it is difficult to grasp the quality of the OSS from available information. In this pape, we present a system that that visualizes OSS developers' discussions on software quality during decision-making and review processes. We hope that this system will be helpful for grasping quality information of OSS in software development companies

Key words　software quality, visualization, Topic Model, Open Source Software

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 [1] [2].

- 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

しかし，品質の低いOSSを導入してしまった場合，その修正に想定外のコストがかかってしまう可能性がある．一般的にOSSは企業が定める品質基準に達していないことが多く，特に異常系やテストコードなどで作り込みが必要となることが多い [2]．そのため，企業がOSSを開発現場に導入するためには，事前に，そのOSSの品質や追加でどのような作り込みが必要かを確認する必要がある．しかし，その確認には網羅的な検証が必要とするためコストが高い．そのため，企業がOSSの品質を容易に把握するための支援が求められている．

企業におけるソフトウェア開発とOSSの開発とでは，品質に対する方針が大きく異なる．企業におけるソフトウェア開発では，明確な開発方法論によって品質を作り込み，構造化された品質保証の方法論によって出荷前に検査をすることで品質を確保している．それに対し，OSSの開発では，開発者だけでなくユーザまで含めたコミュニティを形成することで品質の向上

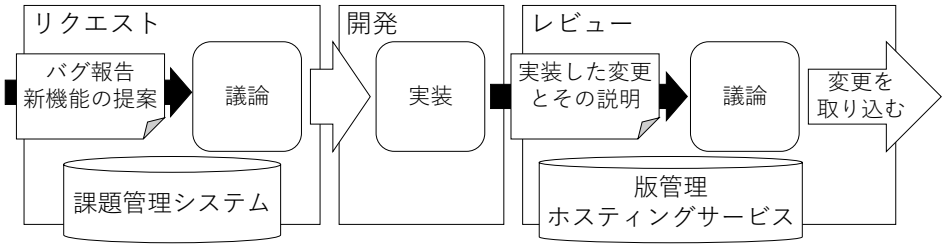

図 1　OSS の開発プロセス

を図っており，品質管理はバグ報告やピアレビューによってなされている [3]．Raymond の「目玉の数さえ十分にあればどんなバグも深刻ではない」[4] という言葉が有名である．このことから，OSS の開発においても品質向上の取り組みがなされているにも関わらず，それが構造的でないために OSS 導入時に企業においても同様の議論を改めて行う必要が生じていると考えられる．

そこで，本研究では OSS 開発における品質に関する議論をそのトピックごとに整理して可視化することを提案する．これにより，OSS 開発時と OSS 導入時における品質保証活動の重複が減少し，OSS 導入時における効率化することが期待できる．

以降，2. 章では関連研究について述べる．3. 章では提案する可視化の手順について述べる．4. 章では予備実験とその結果について述べる．5. 章ではまとめと今後の課題について述べる．

2. 関 連 研 究

2.1 OSS の品質に関する研究

OSS の品質を評価する手法は Open Business Readiness Rating(OpenBRR)，OpenSource Maturity Model(OMM)，Qualification and Selection of Open Source software(QSOS) 等，数多く存在する．これらの手法はいずれも，バグ数など様々な OSS の評価指標を取得し，それらをスコアリングルールに基いてスコアを算出した上で重み付けして得られた値を用いて OSS を評価する．中野らは，OSS の事前品質評価を行うために，OSS の品質に対して企業の開発者にアンケートやヒアリングを実施し，その結果を重み付けに反映させることで系統的な重み付けによる OSS の品質評価手法を提案した．[5] この研究は，企業による OSS の評価を想定しているため，企業の開発者に対しアンケートやヒアリングを実施し，開発者からの意見を重要視する反面，OSS の開発プロセスに注目していない．

2.2 開発活動の可視化に関する研究

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 [6]．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．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. 可視化手法

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 つのプロセスによって行われる．

- **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．
- **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．

1. 章で述べたように，OSS の開発ではバグ報告やピアレビューによって品質管理されている．上述のプロセスでは，リクエストとレビューが品質管理プロセスで対応すると考えられる．そのため，これらのプロセスで発生する議論の中には，品質に対する指摘および懸念，方針などが多く含まれると考えられる．従って，リクエストとレビューのプロセスで行われたける議論を可視化することで，企業が OSS の品質を容易に把握するための支援することができると考えた．以降，そのツールの概要を説明する．

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．これによって，どのような議論をどれだけ行われたかが示される．OSS を導入する企業はこれを基に，OSS 開発時に議論が漏れていた点を重点的に検証することが可能となる．

具体的な可視化の手順を図 2 に示す．以降，各手順の詳細について述べる．

3.1 議論データの収集

リクエストに関する議論として課題管理システム上の Issue を，レビュー時の議論として版管理ホスティングサービス上

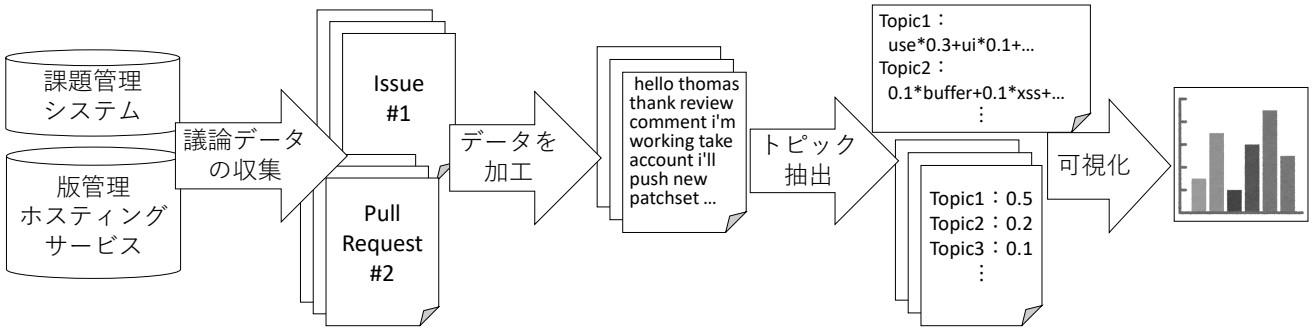

図 2　可視化の手順

の Pull Request を収集する．1 つの Issue，Pull Request をスレッドと呼ぶ．特定のプロジェクトに依存しない，汎用性の高いトピックを求めるため，多くのプロジェクトから収集する必要がある．また，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対象とする．1 つのスレッドは複数の人の投稿からなり，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が，議論のトピックを抽出する上でノイズとなるので分析の対象から除外する．これら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．

3.2　議論データの前処理

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表記に揺れが存在する．そのため，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，小文字への統一とステミングを行う．ステミングとは，品詞の違いや活用による語形の変化に対し，語幹のみを取り出すことで正規化を図る手法である．例えば，「security」，「secure」，などは語幹「secur」で正規化する．ステミングのアルゴリズムは複数存在するが，本研究では Poter のステミングアルゴリズム [8] を利用する．これは，「lying」のような不規則変化する単語も正しく処理できるという特徴がある．

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，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) (注1) 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

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．[9] 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．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) [10] がある．本研究では LDA を階層ディリクレ過程 (HDP: Hierarchical Dirichlet Processes) [11] でノンパラメトリックベイズ拡張した，HDP-LDA を利用する．これにより，LDA では事前に指定するる必要があったトピック数を推定できる．パラメータによる結果の差異が生じないというメリットがある．

3.4　可　視　化

得られたトピックを吟味し，品質に関連するものを抽出して可視化する．可視化の方法として時系列，議論の参加者別，機能別，ファイル別などの内，関連の強い要素で整理し，議論データをグラフ化する方法が考えられる．

4.　予 備 実 験

可視化ツール開発に向けて，OSS の開発プロセスにおいて議論されるトピックはどんなものがあるかを調べるため，予備実験を実施した．この予備実験では，実験対象のデータに対し，3. 章で説明したように，実験データセットの前処理とトピック抽出を行った．

4.1　実験データセット

本実験では，課題管理システム及び版管理ホスティングサービスの両方の機能を備えたサービスである，GitHub 上で開発されている OSS を対象とした．以下の条件を満たすリポジトリのうち，人気度を表す Star 数の上位 11 個のプロジェクトを実験の対象として選択した．また，これらのプロジェクトから Issue や Pull Request を取得し，データセットとして構築した．対象となった OSS のリポジトリとそれぞれの Issue 及び Pull Request の数を表 1 に示す．

- Issue が 100 件以上ある
- Pull Request の数が 100 件以上ある
- GitHub がプログラミング言語を検出している

4.2　結果と考察

実験の結果 11 種類のトピックを抽出した．各トピックの生起確率上位 10 個のキーワードを表 2 に示す．

抽出されたトピックのうち，トピック 2 とトピック 9 の「tensorflow」，トピック 5 の「react」，トピック 8 の「electron」は，対象とした OSS の名前がそのトピックで生起確率が最も高い

(注1) Natural Language Toolkit : http://www.nltk.org/

表 1　実験対象のリポジトリ

URL	Issue(件)	Pull Reqest(件)
https://github.com/freeCodeCamp/freeCodeCamp	11735	4535
https://github.com/twbs/bootstrap	16417	8520
https://github.com/tensorflow/tensorflow	9128	6409
https://github.com/facebook/react	5266	6471
https://github.com/vuejs/vue	6014	1009
https://github.com/d3/d3	1973	1065
https://github.com/airbnb/javascript	661	947
https://github.com/robbyrussell/oh-my-zsh	1723	3499
https://github.com/facebook/react-native	10723	6438
https://github.com/angular/angular.js	8234	7540
https://github.com/electron/electron	7106	3889
合計	78980	50322

表 2　抽出されたトピックのキーワード

トピック	キーワード
トピック 1	use work issu like would function chang react compon class
トピック 2	tensorflow tf python file use op x lib c py
トピック 3	node react modul js use issu fix npm chang test
トピック 4	issu code class challeng use pleas thank div text like
トピック 5	react issu nativ android use app java work view com
トピック 6	x fff framework c b dylib f e com version
トピック 7	sign cla thank request pleas pull test commit pr contributor
トピック 8	electron window app use issu work version thank get js
トピック 9	tensorflow gpu cc core cuda runtim common devic alloc chunk
トピック 10	chang request updat pull pr thank fix doc commit featur
トピック 11	div class tab use li id issu item ng thank

キーワードとなっている．

本研究では，議論に対しトピック抽出を行うと議論中の品質に関する指摘などがトピックとして得られると想定していた．しかし，今回の結果では品質に関するトピックではなく，実験対象ソフトウェアに関するトピックが多くあった．理由として，議論する際にその対象を指し示すため，議論の本題とともにそれらが述べられることが多いということが考えられる．従って，品質に関するトピックを得るためには，形態素解析を導入するなど文章の構造に配慮することが必要と考えられる．

5. おわりに

企業が OSS の品質を容易に把握するためのに，OSS 開発プロセスにおいて品質に関する議論データからトピックを抽出し，可視化するツールを提案した．予備実験として，提案したて手順でトピック抽出を行った．しかし，期待した品質に関するトピックは得られず，手順の改良が必要であることがわかった．今後は，この手順の改良及び効果的な可視化の方法について検討していく．

文　献

[1] 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.

[2] 野村佳秀，木村功作，福寄雅洋，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，vol.33，no.3，pp.3_50–3_65，2016.

[3] M. Aberdour, “Achieving quality in open-source software,” IEEE Software, vol.24, no.1, pp.58–64, Jan. 2007.

[4] E.S. Raymond, The Cathedral and the Bazaar, T. O’Reilly, ed., 1st edition, O’Reilly & Associates, Inc., Sebastopol, CA, USA, 1999.

[5] 中野大扉，松本卓大，山下一寛，亀井靖高，鵜林尚靖，高山修一，岩永裕史，岩崎孝司，“開発形態を考慮した企業内 oss 事前品質評価手法,” 情報処理学会研究報告，vol.2017-SE-195，no.21，pp.1–8，March 2017.

[6] A. Hindle, M.W. Godfrey, and R.C. Holt, “Software process recovery using recovered unified process views,” Proc. of ICSM, pp.1–10, 2010.

[7] 山田悠太，吉田則裕，藤原賢二，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，vol.31，no.2，pp.2_144–2_150，2014.

[8] M.F. Porter, “An algorithm for suffix stripping,” Program, vol.14, no.3, pp.130–137, 1980.

[9] 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. http://ci.nii.ac.jp/ncid/BB18367469

[10] D.M. Blei, A.Y. Ng, and M.I. Jordan, “Latent dirichlet allocation,” Journal of machine Learning research, vol.3, no.Jan, pp.993–1022, 2003.

[11] Y.W. Teh, M.I. Jordan, M.J. Beal, and D.M. Blei, “Hierarchical dirichlet processes,” Journal of the American Statistical Association, vol.101, no.476, pp.1566–1581, 2006.