バージョンに導入される変更提案の特徴分析

上中 瑞稀[1,a)]　伊原 彰紀[1,b)]　柏 祐太郎[†1,c)]

概要： [SS のまま] 大規模なプロジェクトには，ソースコードの変更提案が日々膨大に提出される．プロジェクトの開発者は，ソースコードの品質や保守性の向上のために変更提案を順に検証するが，検証が遅延する変更提案が少なくない．本研究では，直近のバージョンで導入される変更提案を優先的に選択するために，リリースまでの期間に導入される変更提案同士の特徴の違いを明らかにした後，直近のバージョンで導入される変更提案の選択手法を提案する．具体的には，個々の変更提案の特徴を機械学習アルゴリズムに学習したモデルにより，リリースまでの期間に応じた変更提案の予測精度，および予測に寄与する特徴を比較した．分析の結果，リリースまでの期間に応じて変更提案の選択に重要となる特徴が変化することを確かめた．

How to Prepare Your Paper for IPSJ SIG Technical Report (version 2018/10/29)

1. はじめに

オープンソースソフトウェア（OSS：Open Source Software）は，ソフトウェアを変更・配布するための権利を著作権者の定義するライセンスに基づいてソースコードが公開されているソフトウェアであり，OSS のソースコードは，ライセンスに準じて誰でも自由に利用や修正，拡張や再配布が可能である [?].

ソフトウェア開発においてプロジェクトの開発者は，不具合修正や機能追加のためにソースコードを変更する．とりわけオープンソースソフトウェア (OSS) 開発では，不特定多数の開発者が開発プロジェクトのソースコードを変更する場合，変更したソースコードの内容を変更提案としてプロジェクトに提出する．プロジェクトの中心的な開発者は，提出された変更提案に対してコードレビューを行うことで，ソースコードの可読性や欠陥の有無を評価する．また，開発者はコードレビューの内容に従って変更提案のソースコードを変更することで，変更提案のソースコード品質の向上を図る [?]．このように，OSS 開発では，コードレビューを行うことで，ソースコード品質を一定以上に保つ [?]．大規模な OSS 開発プロジェクトでは，プロジェクトに提出される変更提案が膨大であるため，全ての変更提案に対してコードレビューを行うことは困難である．このような課題を解決するために，従来研究では，個々の変更提案の特徴を学習させた機械学習アルゴリズムによって，変更提案が翌日までにコードレビューされる確率を予測し，それをもとに変更提案に優先順位付けする手法を提案している [?]．しかし，個々の変更提案の特徴を学習させるだけでは，予測時点における未対応の変更提案数，対応可能な開発者数などの開発状況を考慮できず，結果としてプロジェクトの開発状況に応じて変更提案を選択することが容易でない．

本研究では，直近のバージョンで導入される変更提案を優先的に選択するために，開発状況を考慮した変更提案の選択手法を提案する．開発が進むにつれて変化する未対応の変更提案数や対応可能な開発者数などを開発状況として捉える．具体的には，直近のバージョンで導入される変更提案を優先的に選択するために，直近のバージョンで導入される変更提案の予測モデルを構築し，予測の際に重要となる特徴量を分析することで，直近のバージョンで導入される変更提案の特徴を明らかにする．また，本研究では変

[1] 和歌山大学
Faculty of Systems Engineering, Wakayama University, 30 Sakaedani, Wakayama, 640–8441 Japan
[†1] 現在，奈良先端科学技術大学院大学
Presently with Nara Institute of Science and Technology, 8916–5 Takayama-cho, Ikoma, Nara, 630–0192 Japan
[a)] s246035@wakayama-u.ac.jp
[b)] ihara@wakayama-u.ac.jp
[c)] yutaro.kashiwa@is.naist.jp

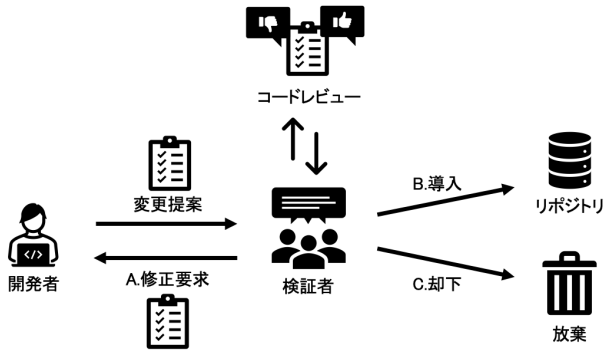

図 1　コードレビュープロセス

更提案の推薦手法の有効性を評価するために 1 つの RQ を設定する．

- RQ:リリースまでの期間に応じて優先的に導入される変更提案の特徴は異なるか？

RQ では，リリースまでの期間に応じた優先的に導入される変更提案の特徴の違いを明らかにするために，リリースまでの期間に応じた変更提案の導入に重要な特徴を調査する．以降，本論文では，**??**章で OSS 開発におけるコードレビュープロセスと従来研究について述べる．その後，**??**章で本論文で対象とするデータセットについて述べ，**??**章で RQ の分析手法と結果を述べる．そして，**??**章で結果の考察および妥当性について述べ，**??**章でまとめる．

2. OSS 開発におけるコードレビュープロセス

2.1 コードレビュープロセス

図**??**はコードレビュープロセスの概要を示す．レビュアーはコードレビューに基づく判断によって，変更提案に対して，A. 修正要求・B. 導入・C. 却下のうちいずれかの対応を行う．A. 修正要求の場合，変更提案を提出した開発者は，コードレビューの内容に従ってソースコードの見直しを行い，再度修正を行ったものをプロジェクトに提出する．このような作業を各変更提案ごとに繰り返し行うことで，プロジェクトのソースコード品質を維持しつつ，不具合修正やさらなる機能拡張を行うことが可能となっている．

コードレビューはソフトウェア開発に多大な貢献をもたらす一方で，多大なコストがかかる．1 つの変更提案に対するコードレビューに数日から数ヶ月の期間を要することもあり [**?**]，1 週間で平均 6 時間程度をコードレビューに費やしている [**?**]．また，OSS 開発では，不特定多数の開発者が変更提案を提出可能であるという特徴から，プロジェクトに膨大な変更提案が提出される．膨大な変更提案全てに対してコードレビューを行うことは困難であるため，コードレビュー対象とする変更提案を選択する必要があるが，コードレビューする変更提案の選択もまた，変更提案が膨大であるため容易ではない．

表 1　従来研究 [**?**] で用いる 14 種類の説明変数

説明変数	説明
経過時間	変更提案が提出されてからの経過分数
貢献率	変更提案作成者のプロジェクトでのコミット率
受入率	変更提案作成者が過去に提出した変更提案の導入率
追加行数	変更提案で追加されている変更行数
削除行数	変更提案で削除されている変更行数
リビジョン数	変更提案のリビジョン数
ファイル数	変更提案の変更ファイル数
コメント数	変更提案のコメント数
レビュー数	変更提案に対してのコードレビュー数
コアメンバー	変更提案の作者はプロジェクトメンバーか
ブランチ	変更コードとマージ先のリポジトリが同一か
issue 含有	変更提案に紐づいている issue があるか
最終コメント	最終コメントでユーザがメンションされているか
テストコード含有	変更ファイルにテストコードが含まれているか

2.2 従来研究

2.2.1 変更提案の優先順位付け

従来研究では，OSS 開発における，コードレビューを行う変更提案の選択が困難であるという課題を解決するために，変更提案の優先順位付けの手法を提案している．従来研究では，説明変数として表**??**に示す変更内容や作成者の特徴などの変更提案の 14 種類の特徴と，目的変数として翌日までにコメントやコードレビューされるか否かを機械学習アルゴリズム（ランダムフォレスト）に学習させ，学習させた機械学習モデルを用いて，各変更提案が翌日にレビューされる確率に基づいて変更提案の優先順位を予測している [**?**]．機械学習モデルの性能評価のために，475 プロジェクトに対して 10 分割交差検証を 10 回ずつ行った結果，適合率 0.64，再現率 0.85 を達成している．

2.2.2 変更提案の導入と開発者間の関係

Bosu[**?**] らは，OSS 開発における開発者の地位が変更提案の導入に影響するのか否かを明らかにするために，OSS 開発を活発に行う開発者と不活発な開発者での変更提案の導入プロセスの違いに関する調査を行なった．8 つの OSS プロジェクトから導入もしくは却下と判断された変更提案のコードレビューデータを調査した結果，OSS 開発を活発に行う開発者の方が，不活発な開発者より変更提案の導入率が高いことが明らかとなった．また，OSS 開発を活発に行う開発者の方が，変更提案に対して初回のコードレビューが行われる時間が短いことや，変更提案に対して初回のコードレビューが行われてから，導入もしくは却下の判断が為されるまでの時間が短いことも明らかとなった．そのため本研究では，開発で導入される変更提案の予測モデル構築の際に，初回のコードレビューが行われるまでの時間や，初回のコードレビューが行われてからの時間を特徴量として学習させることで，予測精度の向上を図る．

表 2　プロジェクトごとの対象リリースバージョン

プロジェクト	リリースバージョン（変更提案数）
Neutron (24,467)	2013.1.g3(317)，2013.2.b2(580)，2013.2.rc1(423)，2014.2.b1(320)，2015.1.0b1(391)，2015.1.0b2(306)，7.0.0.0b1(371)，7.0.0.0b3(396)，8.0.0.0b1(408)，8.0.0.0b2(326)，9.0.0.0b3(363)，11.0.0.0b3(388)，15.0.0.0b1(312)，19.0.0.0rc1(420)，20.0.0.0rc1(380)
Keystone (10,764)	2011.3(195)，essex-4(357)，2013.2.b1(186)，2013.2.b3(261)，2014.1.b3(252)，2014.2.b1(185)，2015.1.0rc1(199)，8.0.0a0(187)，9.0.0.0b2(212)，9.0.0.0b3(214)，10.0.0.0b1(179)，10.0.0.0b2(220)，11.0.0.0b1(273)，15.0.0.0rc1(408)，16.0.0.0rc1(225)
Horizon (12,475)	folsom-2(391)，2014.2.b2(190)，2014.2.b3(279)，2015.1.0b2(198)，2015.1.0b3(237)，8.0.0.0b2(402)，8.0.0.0b3(384)，9.0.0.0b1(230)，9.0.0.0b3(237)，10.0.0.0b2(193)，10.0.0.0b3(171)，11.0.0.0b2(193)，13.0.0.0b1(310)，14.0.0.0b1(194)，15.0.0.0b2(178)
Nova (39,870)	folsom-1(1,550)，2013.1.rc1(2723)，2013.2.b3(1,517)，2013.2.rc1(555)，2014.1.b3(517)，2014.2.b2(554)，2015.1.0b2(838)，2015.1.0b3(625)，13.0.0.0b3(529)，14.0.0.0b2(607)，16.0.0.0b2(524)，17.0.0.0b1(516)，19.0.0.0rc1(865)，21.0.0.0rc1(808)，24.0.0.0rc1(771)

2.3　動機

従来研究 [?] では，ソフトウェア開発において優先的にコードレビューを行う変更提案の選択を容易にするために，変更提案に対して優先順位付けを行なっているが，変更提案の特徴量のみを用いて予測を行うため，直近のバージョンリリースまでの期間やその時点での対応可能な開発者数などの開発状況は考慮されていない．しかし，ソフトウェア開発において，バージョンリリースが近い場合は新たに提出された大規模な変更を導入するためにコードレビューのリソースを割く時間が無いというように，開発状況によって対応する変更提案は異なると考える．RQ では，リリースまでの期間に応じた変更提案の導入に重要な特徴を調査する．

3.　分析対象データセット

本章では，RQ に共通して用いるデータセットを説明する．本論文では，OpenStack におけるプロジェクトの中から，導入もしくは却下と判断された変更提案が多い Neutron，Keystone，Horizon，Nova プロジェクトを分析対象プロジェクトとし，各プロジェクトの変更提案の中から，収集時点で導入もしくは却下されることが決定している変更提案を，各リビジョンごとに収集する．具体的には，コードレビュー管理システムである Gerrit[*1]から，それぞれの変更提案のリビジョンごとの特徴量や，コードレビュー履歴を収集し，リポジトリホスティングサービスである GitHub[*2]から，プロジェクトのコミット履歴や，リリースバージョンのリリース日やコミットハッシュを収集した．4 プロジェクトそれぞれの変更提案数は表**??**のプロジェクト名の下に示す．また，表**??**には，各プロジェクトにおいてリリースされたバージョンと，導入された変更提案数の上位 15 バージョンを示す．正解ラベルのついた変更提案数を一定以上確保するためにこれら 15 バージョンを分析対象とする．

[*1] https://review.opendev.org
[*2] https://github.co.jp/

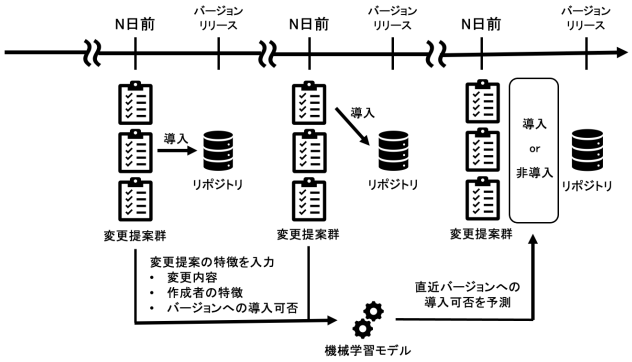

図 2　RQ の分析概略図（リリース N 日前を対象とした場合）

4.　RQ:リリースまでの期間に応じて優先的に導入される変更提案の特徴は異なるか？

4.1　概要

本章では，RQ を検証するために，分析対象データセットから変更提案の特徴や変更提案作成者に関する 16 種類の説明変数と目的変数を計測し，ランダムフォレストを用いて分析対象とするバージョンリリースまでに導入される確率を予測するモデルを構築する．構築した予測モデルを用いて，リリースまでの異なる時点ごとに分析対象とするバージョンリリースまでに導入される変更提案を予測する．異なる時点ごとでの予測精度とそれぞれの説明変数の重要度を比較することで，リリースまでの期間に応じて優先的に導入される変更提案の特徴の違いを明らかにする．

図**??**は分析方法の概略図を示す．本分析では，リリースまでの期間の違いによって優先的に導入される変更提案の特徴が異なるか否かを分析するために，対象リリースバージョンまでの異なる時点に存在する変更提案の特徴量を算出する．また，予測モデルの構築の際は，予測するバージョンより前にリリースされたバージョンのリリース N 日前の時点で存在する変更提案を学習データ，予測するバージョンのリリース N 日前に存在する変更提案をテストデータとして用いる．

4.2　変更提案の導入確率予測モデルの構築

変更提案の導入確率の予測モデルを構築するために，本研究では対象データセットとするプロジェクトの予測時点での変更提案のリビジョンから 16 種類の説明変数を計測する．表**??**は計測する説明変数を示す．また，変更提案のコミット履歴を追跡し，変更提案が導入されたバージョンを分析することで，変更提案が目的のリリースに導入されるか否かの 2 値を目的変数として計測する．

また，本論文では，予測モデルの構築のための機械学習アルゴリズムとして，決定木を複数組み合わせてモデル構築を行う教師あり機械学習アルゴリズムであるランダムフォレスト [?] を用いる．ランダムフォレストを用いる利

表 3　予測モデル構築のための説明変数

説明変数	説明
経過時間	変更提案が提出されてからの経過分数
貢献率	変更提案作成者のプロジェクトでのコミット率
受入率	変更提案作成者が過去に提出した変更提案の導入率
追加行数	変更提案で追加されている変更行数
削除行数	変更提案で削除されている変更行数
リビジョン数	変更提案のリビジョン数
ファイル数	変更提案の変更ファイル数
レビュー数	変更提案に対してのコードレビュー数
プラスレビュー数	変更提案に対してのポジティブなコードレビュー数
マイナスレビュー数	変更提案に対してのネガティブなコードレビュー数
放置時間	変更提案が提出されてから最初にレビューされるまでの経過分数
レビュー経過時間	変更提案が最初にレビューされてからの経過分数
作成済み変更提案数	変更提案作成者が過去に提出した変更提案数
ブランチ変更提案数	ブランチに提出されている変更提案数
テストコード含有	変更ファイルにテストコードが含まれているか否か
issue 含有	変更提案に紐づいている issue があるか否か

点として，アンサンブル学習によって過学習を防ぐことができる点や，特徴量の正規化や標準化の必要がない点が挙げられる．本論文では，ランダムフォレストを Python の sklearn.ensemble.RandomForestRegressor を用いて実装する．また，実装の際，パラメータは全てデフォルト値を用いる．

4.3　評価方法

4.3.1　予測モデルの評価

本分析では，予測モデルの性能を評価するために，予測モデルの汎化性能の調査によく用いられる交差検証を用いる．交差検証とは，モデル構築の際にデータセットを K 個に分割し，分割したデータセットの中から K-1 個を訓練データとしてモデルに学習させ，残りの 1 個をテストデータとしてモデルの評価を行うという工程を K 回繰り返し行うことで，より正確な予測モデルの評価を行うための手法である．単純な交差検証の欠点として，データセットに時系列データを用いた場合，未来のデータを学習したモデルを用いて過去のデータを予測することがあるため，予測モデルの汎化性能を正しく評価できないことが挙げられる．そのため，本論文では予測モデルの汎化性能の調査のために，時系列交差検証 (Time Series Split) を用いる．時系列交差検証とは，モデル構築の際にデータセットを K+1 個に分割し，K 番目の予測では分割したデータセットの中から K 番目までのデータを訓練データとしてモデルに学習させ，K+1 番目のデータをテストデータとしてモデルの評価を行うという工程を K 回繰り返し行うことで，過去のデータを訓練データ，未来のデータをテストデータとして用いた予測モデルの評価を行うための手法である．なお，時系列交差検証ではデータセットを時系列順で並べておく必要があるため，本論文では 4 プロジェクトの対象リリースバージョンを交互にリリースの古いバージョンから順にデータセットとする．

また，予測モデルの評価指標として，回帰モデルの評価指標としてよく用いられる平均絶対値誤差（MAE），二乗平均平方根誤差（RMSE），決定係数 (R^2) を用いる．それぞれの本論文における分類精度の説明を以下に示す．

平均絶対値誤差（MAE）とは，全データに対して，直近のバージョンリリースに導入されるか否か（1,0 の 2 値）と直近のバージョンリリースに導入される確率の差の絶対値の平均値を示す．また，MAE は，0 に近いほど精度が高いことを表す．

MAE は，データの総数を n，実際の値を y_i，予測する確率を $\hat{y_i}$ とすると，(**??**) 式で表される．

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y_i}| \tag{1}$$

二乗平均平方根誤差（RMSE）とは，全データに対して，直近のバージョンリリースに導入されるか否か（1,0 の 2 値）と直近のバージョンリリースに導入される確率の差の 2 乗の平均値に平方根をとった値を示す．また，RMSE は，0 に近いほど精度が高いことを表す．

RMSE は，データの総数を n，実際の値を y_i，予測する確率を $\hat{y_i}$ とすると，(**??**) 式で表される．

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2} \tag{2}$$

決定係数 (R^2) とは，データに対する推定された回帰式の当てはまりの良さを示す．また，決定係数は 0 から 1 の値をとり，1 に近いほど精度が高いことを表す．

決定係数は，データの総数を n，実際の値を y_i，予測する確率を $\hat{y_i}$，実際の値の平均を $\bar{y}$ とすると，(**??**) 式で表される．

$$\text{決定係数} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y_i})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{3}$$

4.3.2　説明変数の重要度の評価

本論文では，どの説明変数が予測に寄与したのかを明らかにするために，説明変数の重要度を算出する．説明変数の重要度は，ランダムフォレストによって予測を行う際に，各説明変数が予測精度に寄与した割合であり，全説明変数の重要度の和は 1 となる．また，各説明変数の重要度は，1 に近いほど重要度が高いことを表す．

説明変数 f_i の重要度 $importance_{f_i}$ は，説明変数 f_i で分割したノードの集合を $node_{f_i}$，ノード m の改善度を $\triangle I_m$ とすると，(**??**) 式で表される．

$$importance_{f_i} = \sum_{m \in node_{f_i}} \triangle I_m \tag{4}$$

4.4　分析結果

本章では，異なる時点として，1 週間前，2 週間前，1 ヶ月前，2 ヶ月前の 4 時点を対象として分析した結果を示す．また，本論文では時系列交差検証を 10 分割で用いている

表 4　モデルの予測精度

予測時点	MAE	RMSE	決定係数
1 週間前	0.03	0.12	0.11
2 週間前	0.04	0.14	0.22
1 ヶ月前	0.06	0.17	0.27
2 ヶ月前	0.06	0.17	0.21

ため，各評価指標，説明変数の重要度は，10 回分の数値の平均値を用いる．評価指標については小数第 3 位で四捨五入した値を示す．

4.4.1　予測精度

予測精度を表??に示す．リリース 1 週間前時点は MAE 0.03，RMSE 0.12，決定係数 0.11 であり，リリース 2 週間前時点は MAE 0.04，RMSE 0.14，決定係数 0.22 であり，リリース 1 ヶ月前時点は MAE 0.06，RMSE 0.17，決定係数 0.27 であり，リリース 2 ヶ月前時点は MAE 0.06，RMSE 0.17，決定係数 0.21 であった．評価指標の中でも MAE と RMSE は全ての予測時点において高い精度を示しているため，直近のバージョンリリースに導入されるか否かと予測確率の差はほとんどないと考えられる．特に，RMSE は MAE と比べて外れ値に敏感であるという特徴を持つため，MAE と比べ精度が下がっていると考えられる．また，決定係数は全ての予測時点において低い精度を示しているため，予測結果は目的変数の分散を予測できていないと考えられるが，本研究では目的変数は連続値ではないため，値が分散していないために低い精度を示したと考えられる．

4.4.2　説明変数の重要度

予測に用いた説明変数の重要度の平均を図??から図??に示す．比較的バージョンリリースが遠いリリース 1,2 ヶ月前の時点においては，レビュー経過時間が重要度の高い説明変数である．一方で，バージョンリリースが近くなるにつれて，経過時間の重要度が高くなる．また，全予測時点において比較的重要度の高い説明変数として，貢献率や作成済み変更提案数などの変更提案作成者の特徴や，追加行数などの変更提案の特徴が挙げられる．

また，各時点において特に高い重要度であった経過時間とレビュー経過時間の 2 種類の説明変数に関して，目的変数別のデータの分布を調査するために，箱ひげ図を用いる．それぞれのデータの分布図を図??，図??に示す．図より，直近のバージョンリリースに導入される変更提案は導入されない変更提案と比べて経過時間，レビュー経過時間ともに短いことがわかった．また，図??，図??からは導入された変更提案の細かい値の変化を読み取ることができないため，導入される変更提案の大半の経過時間，レビュー経過時間が 100,000 以下の値を示していることを考慮し，100,000 を閾値とし，それぞれの値が閾値以下の変更提案に絞って箱ひげ図を作成する．それぞれのデータの分布図

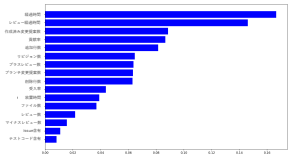

図 3　説明変数の平均重要度（1 週間前）

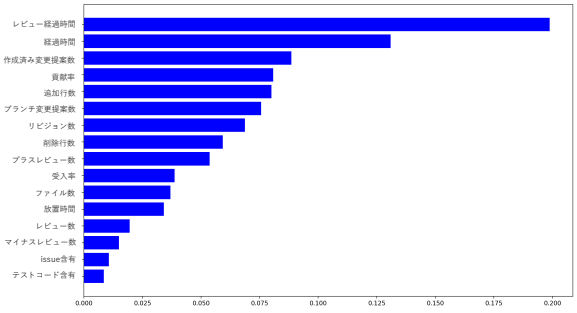

図 4　説明変数の平均重要度（2 週間前）

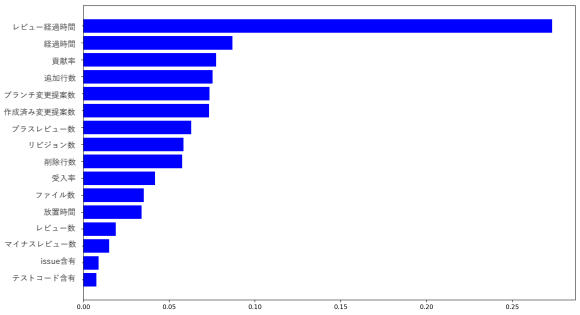

図 5　説明変数の平均重要度（1 ヶ月前）

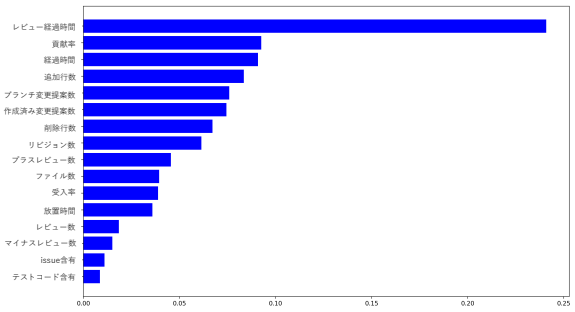

図 6　説明変数の平均重要度（2 ヶ月前）

を図??，図??に示す．図より，直近のバージョンリリースに導入される変更提案はバージョンリリースが近くなるにつれて，経過時間，レビュー経過時間ともに短くなることを確認した．

5.　考察

RQ の結果から，直近のバージョンリリースに導入される変更提案の予測において，経過時間とレビュー経過時間が変更提案の特徴として重要な説明変数であることがわかり，直近のバージョンリリースに導入される変更提案

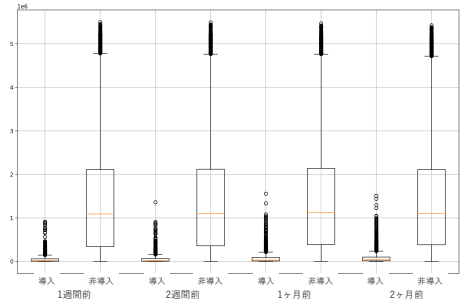

図 7　経過時間

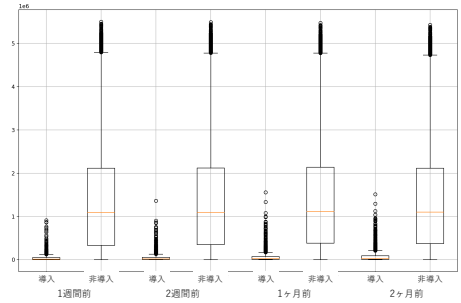

図 8　レビュー経過時間

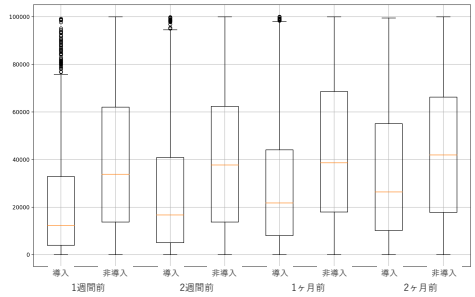

図 9　経過時間 (閾値以下)

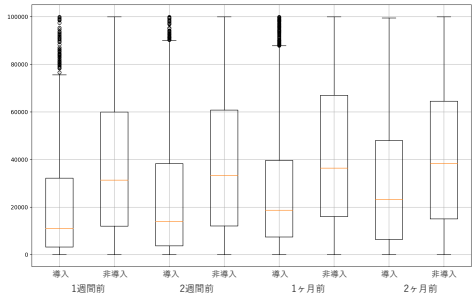

図 10　レビュー経過時間 (閾値以下)

はバージョンリリースが近くなるにつれて，経過時間，レビュー経過時間ともに短くなることがわかった．また，経過時間，レビュー経過時間の重要度の平均は，リリース 2 週間前までの時点においてはレビュー経過時間の方が高く，リリース 1 週間前の時点においては経過時間の方が高くなることがわかった．これらの結果から，リリースまで時間がある場合はコードレビューされてから時間の経っていない変更提案が導入され，リリースまで時間がない場合はコードレビューの有無に関わらず提出されてから時間の経っていない変更提案が導入されると考えられる．

5.1　妥当性の脅威

5.1.1　内的妥当性

本研究では，変更提案の予測モデルを構築するために，変更提案から 16 種類の特徴量を学習させた．RQ の結果として決定係数の精度が低かったため，予測のための特徴量の妥当性が危ぶまれる．しかし，本研究では予測モデルによって予測する目的変数の正例と負例の割合は，1:30〜150 程度とかなり負例の多いデータセットとなっている．そこで，学習させるデータセットとテストを行うデータセット両方の正例と負例の割合を 1:1 にして予測したところ，ほとんどの予測タイミングにおいて，決定係数が 0.5 以上という精度の高い結果が得られた．この結果をもって，予測のための特徴量の妥当性の脅威を削減する．

5.1.2　外的妥当性

本研究では，4 つの OSS プロジェクトを対象として分析を行なったため，対象とするプロジェクトを変更することで，異なる結果となる可能性がある．本研究では，多くの変更提案が提出されており，かつ現在も開発が続けられているプロジェクトを選択することで，脅威を削減する．

6.　おわりに

本論文では，直近のバージョンリリースに導入される変更提案の選択を容易にすることを目的として，開発状況を考慮した変更提案の選択手法を提案し，設定した RQ を検証した．対象データセットとして，OpenStack におけるプロジェクトの中から，膨大な変更提案が提出されている Neutron，Keystone，Horizon，Nova プロジェクトの変更提案を収集し，変更提案の特徴量を学習させることで，直近のバージョンリリースに導入される変更提案の予測モデルを構築した．モデルの予測精度を評価した結果，MAE や RMSE が高い精度であった一方，決定係数の値は低くなるという結果が得られた．また，予測の際に重要となった特徴を確認した結果，リリースが遠い時点では初回のレビューからの経過時間が重要となるが，リリースが近づくにつれて作成からの経過時間が重要となった．

謝辞　ほげ