ソフトウェアオープン化時代に向けたソースコード自動検証技術の開発

Automated code review system for open source software development

伊原　彰紀（Akinori IHARA, Ph. D.）
和歌山大学　システム工学部　講師
(Lecturer, Faculty of Systems Engineering, Wakayama University)
IEEE, 電子情報通信学会，情報処理学会，ソフトウェア科学会
受賞：情報処理学会 SIGSE 卓越研究賞 (2019 年). IEEE 関西支部 Young Professionals 賞 (2017 年) . International Conference on Open Source Systems (OSS) Best Paper Award (2017 年)　他
研究専門分野：ソフトウェア工学　リポジトリマイニング　プログラム解析

あらまし

今日のソフトウェアは，商用ソフトウェア開発企業のような閉じられた世界で開発するものではなくなりつつある．オープンソースソフトウェア（OSS）をはじめ，商用ソフトウェア開発企業でさえも，Web を介してソフトウェア部品（ソースコード，等）を広く公開し，誰でもソフトウェアの新機能を提案，実装することができる．このようにクラウド上で他者と共同で進めるソフトウェア開発は，不特定多数の知的協調作業が織りなすオープンイノベーションによってソフトウェアの技術やサービスの発展を実現している．しかし，ソフトウェア開発への参加間口の拡大によってソフトウェアの開発者不足の課題が解消に向かう一方で技術力の未熟な開発者によって実装されたプログラムの提出が増加し，その検証作業（コードレビュー）コストが増大するジレンマを引き起こしている．本研究は，このようなジレンマの解決に向けて，ソースコード品質を自動評価するための技術開発を目指し，ソースコード改善システム DevReplay を紹介する．

1．研究の目的

本研究では，プログラムの記法はもちろん，ソフトウェア開発組織の方針，技術進化の動向に合わせてソースコードの保守を効率化するため自動検証技術の開発を目指した．具体的には，ソフトウェア開発において機能追加，不具合修正のために提案されたソースコードに対して，検証者の指摘に基づき修正されたソースコードを解析し，修正内容を開発者の属人性を排除した形式に変換するための検討を行った．

2．研究の背景

金融，流通などのミッションクリティカルなシステムにも使用されるLinuxは，1973年にC言語で実装されてから47年余りが経過した．Linuxの開発が長期間継続できる背景には，ソースコードをオープンソースソフトウェア（OSS）として広く公開することで，不特定多数の開発者が入れ替わりながらも停止することなく新たな機能の実装，不具合修正を継続していることが挙げられる．クラウド上で他者と共同で新機能を実装，提案する開発形態（ソーシャルコーディング）は，OSSをはじめ，商用ソフトウェア開発企業にも定着しつつある．ソーシャルコーディングは，ソフトウェアの開発データを管理する共有WebサービスGitHub[1]が想定する開発形態であり，ソフトウェア開発のデファクトスタンダードとして確立されつつある．GitHubのSNS機能は，組織のソフトウェア部品を構成管理するデータサーバ（リポジトリ）を広く共有，伝搬することを実現している．ソフトウェアを管理するリポジトリを誰でも取得できるようになったことで，ソフトウェア開発への参加間口が拡大し，他の開発者が実装した機能を容易に利用・拡張できるようになった．しかし，ソーシャルコーディングは開発に貢献する人的リソースの確保を可能にする一方で，実装スタイルの異なる個々の提案を保守するためのコストが増大するジレンマを引き起こしている．このジレンマを解決するために，ソフトウェアの実装方法を共通化し，開発者間で相互に期待する保守性の高いソフトウェアの実装方法を共有することが必要である．

[1] GitHub: https://github.com/

表１　コード改善とみなす変更内容［1］

改善名	コード改善内容
** ADD_REMOVE_NEWLINE	改行の追加削除
** ADD_REMOVE_SPACE	空白やインデントの追加削除
** CHANGE_VALUE_STYLE	変数名の大文字小文字，またはハイフン記号の変更
ABSTRACT_VALUE	変数の抽象化
ADD_REMOVE_ARRAY_ELEMENTS	配列要素の追加削除
ADD_REMOVE_DECORATOR	デコレータの追加削除
ADD_REMOVE_DICT_ELEMENTS	辞書要素の追加削除
ADD_REMOVE_RETURN	文の追加削除
CHANGE_COMENT	コメント文の変更
CHANGE_DEFINE_ORDER	変数を定義する順序の変更
CHANGE_FUNC_OBJECT	同一名の関数を呼び出すオブジェクトの変更
CHANGE_IMPOR	文で呼び出すライブラリの変更
CHANGE_STRING	文字列の変更
CHANGE_VALUE_NAME	“CHANGE VALUE STYLE”以外の変数名の変更
UPDATE_VERSION	ライブラリのバージョン更新
ADD_REMOVE_NEWLINE	改行の追加削除

** Pylint または Flake8 で自動検出可能な改善

３．研究の方法

GitHub が多くの組織で利用されるようになって 10 年余りが経過し，現在公開されているリポジトリは 4,200 万件[2]にも達し，GitHub を使って多数のソースコード，また，その改善履歴が蓄積されている．本研究では，これら蓄積データを対象に，コードレビューによってソースコード改善された内容を調査した上で，自動検証システム DevReplay の開発を行った．

３．１．ソースコード改善内容の調査

GitHub を用いて管理される大規模ソフトウェア OpenStack, Google, Microsoft, Facebook を対象に，コードレビューを通して改善されるソースコードの変更内容を調査した．開発者が作成し，プロジェクトに投稿したソースコード初版を $Patch_1$，検証者による指摘に基づき開発者が修正を繰り返して採用されたソースコード最終版を $Patch_n$ とし，$Patch_1$ と $Patch_n$ の変更対 384 件（許容誤差は 5%，信頼度は 95%）を対象に，既存研究［1］が定義した不具合修正パターン（表 1 参照）に基づき変更内容がコード改善であるか否かを目視で検出した．

本研究が分析対象としたソフトウェアは，開発者が統一的な実装方法を実現するために，ソフトウェア開発組織でコーディング規約を取り決めている．コーディング規約とは，開発者の知識や経験から不具合となりやすい実装方法を防止するためにまとめたガイドラインであり，具体的にはインデントやコメント量の基準，変数等の命名規則，禁止事項などがある．規約に違反しているソースコードを自動検出するために静的解析ツールが使用される．分析対象とする組織では Python 言語を主に使用しており，Python 言語標準のコーディング規約 PEP8 に基づいた静的解析ツール Pylint や Flake8 を使用している．プロジェクト内のソースコード記述方式に一貫性をもたせることを目的に静的解析ツールの使用を推奨しているが，コードレ

[2] https://github.com/search?q=is:public（2020 年 7 月現在）

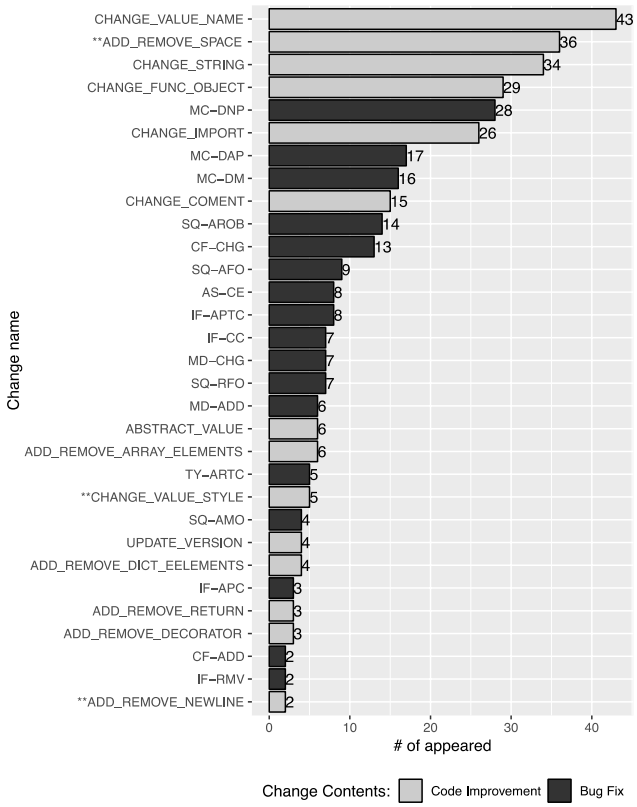

図 1 コードレビューを通して実施されたソースコード改善と不具合修正の出現回数
（総変更数: 384 件中ソースコード改善は 211 件）

ビューにおいて検証者は，コーディング規約以外についても確認している．しかし，具体的な指摘，および，改善内容は従来研究で明らかにされていなかった．

図1は，本研究での調査により，コードレビューを通して実施されたソースコード改善と不具合修正の出現回数を示す．目視で384件のソースコード改善履歴を分類した結果，ソースコードの改善内容384件中211件（56.0%）がコード改善であることを確認した．特に多く出現したのは変数名の変更（CHANGE_VALUE_NAME）であった．また，空白やインデントの追加削除（ADD REMOVE SPACE，CHANGE VALUE STYLE）のように既存の静的解析ツール自動的に検出可能であるにもかかわらずコードレビューにおいて改善されたのは384件中46件（11.5%）含まれていることを明らかにした．既存の静的解析ツールでは検出が困難なプロジェクトの固有名詞や出力文字列の記法に関する修正は，各組織が独自にコーディングガイドラインを定義し，開発者間で共有することでこれらの問題を予防することが可能である．

３．２．DevReplay：ソースコード自動検証システム

本研究は，コードレビューを通して多数のコード改善が実施されていることを明らかにした [2] [3]．また，掲載の都合上，詳細は割愛するが，ソフトウェアのオープン化に伴い，不特定多数の開発者が入れ替わりながら開発を進める中で，コードレビューにおいて改善されるソースコードの変更パターンは時間経過とともに変化していることも明らかにした [4]．これら調査結果を受けて，本研究では，過去に開発者が行ったソースコードの検証，および，修正提案を行うソースコード自動検証システム DevReplay を開発した．DevReplay は，GitHub で公開されているソフトウェアから，ソースコードの開発履歴を収集し，他の開発者によって実装提案されたソースコードの修正方法を自分のソースコードに適用する開発環境である．DevReplay により，開発者が過去に直面したことのない問題も，関連プロジェクトから修正方法を知ることが可能となる．また，実行速度に優れたソフトウェアの開発履歴を利用することで，自身のソースコードをパフォーマンスに優れたものに改善することも可能とる．

図 1 は，DevReplay において Git を用いた場合のプログラム修正パターンの生成プロセスを示す．まず対象となるリポジトリから変更前変更後のコミットを得る．次に変更前，変更後のソースコードを抽象構文木に展開する．このとき変更前と変更後で共通している識別子や数字は$1 や${2:num}の形式に抽象化する．最後に抽象化し，識別子も含めた TextMate Snippet を修正パターンとして出力する．修正パターンに基づき，新たなソースコードの提案に対し，改善するための実装方法を提示することで自動検証を実現した．

現在 DevReplay は C, C++, Java, Dart, JavaScript, Python, Go, TypeScript, COBOL, Ruby, PHP, R の 12 言語に対応している．

DevReplay: https://devreplay.github.io/

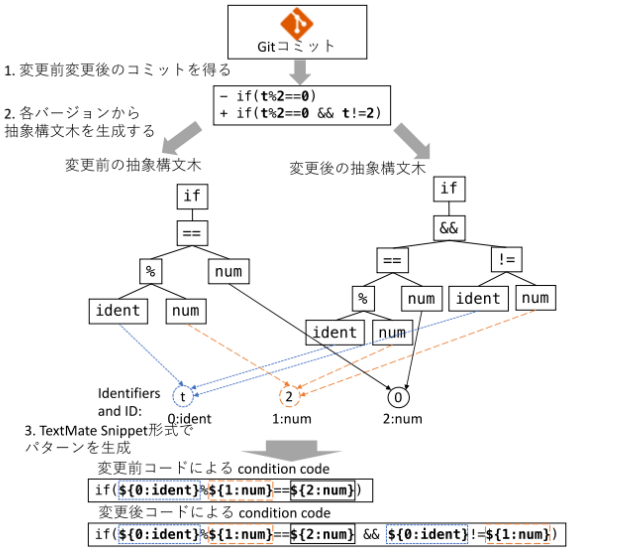

図 2　Git を用いた修正パターン生成プロセス

４．将来展望

Linux のようにソフトウェアの長期的な開発保守の実現には, 継続的な機能追加, 不具合修正はもちろん, それら根底を支える実装方法の統一化も不可欠な要素である．本研究が開発する DevReplay は，開発者が暗黙知として持つ実装方法に基づき，高品質なソースコードを提案するシステムである．しかし，DevReplay が提案する実装方法も技術進化に伴い開発者にとって良い提案でなくなることが考えられる．今後は，DevReplay が持つ修正パターンの更新，及び，実装方法の自律的な更新が課題の一つである．

さらには，DevReplay は，プログラミング上級者の書いたソースコードの実装方法を蓄積することで，その上級者の実装に類似したソースコードに矯正することも可能である．ソフトウェア開発企業はもちろん，プログラミング初学者の実装矯正にも効果を発揮すると考えており，今後検討を進める．

おわりに

本研究は，ソフトウェア開発において開発者が多くの工数を費やしているコードレビュー作業を対象に，機能追加，不具合修正のためのソースコード提案に対する改善内容の調査を行った．本調査を通して，各組織が独自にコーディングガイドラインを定義することで，開発者の検証作業のコスト削減が期待できることを明らかにした．本分析を受けて，ソースコード自動検証システムDevReplayを開発した．DevReplayは，実装提案されたソースコードの修正方法をパターン化し, 組織独自のガイドラインを容易に作成可能である．

今日，国内のIT人材は17万人不足していると言われる［5］．このような状況を解決する一つの手段としてソフトウェアをオープン化することは開発人材確保するための有効な方法として期待している．今後，ソーシャルコーディングにおいて不規則に入れ替わる開発者間でソースコードの「良い実装方法」が共有され，DevReplayを活用した実装方法の知識移転が成功することを願う．

参考文献

[1] Kai Pan, Sunghun Kim, and E. James Whitehead, “Toward an understanding of bug x patterns,” Empirical Software Engineering, Vol.14, No.3, pp.286-315, 2009.

[2] Yuki Ueda, Akinori Ihara, Takashi Ishio, Toshiki Hirao, Kenichi Matsumoto, “How are IF-Conditional Statements Fixed Through Peer CodeReview?”, IEICE Transactions, pp.2720-2729, November 2018.

[3] 上田裕己,石尾隆,伊原彰紀,松本 健一, “コードレビューにおいて検出されるソースコード改善内容の分析,” Vol.37, No.2, pp.76-85, コンピュータソフトウェア, 2020.

[4] Yuki Ueda, Takashi Ishio, Akinori Ihara, and Kenichi Matsumoto, “Mining Source Code Improvement Patterns from Similar Code Review Works”, In Proc. 13th International Workshop on Software Clones (IWSC’19), pp. 13-19, 2019.

[5] “平成 30 年度我が国におけるデータ駆動型社会にかかる基盤整備-IT 人材需要に関する調査-,” 経済産業省調査報告書.

この研究は，平成２８年度ＳＣＡＴ研究助成の対象として採用され，平成２９～令和元年度に実施されたものです．