様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

令和　元 年　 ６ 月 ２０ 日現在

機関番号：１４３０１
研究種目：若手研究(B)
研究期間：2015～2018
課題番号：１５Ｋ１５９７３

研究課題名（和文）ソフトウェア進化分析に基づくソフトウェア保守支援環境

研究課題名（英文）Software Maintenance Support Environment based on Software Evolution Analysis

研究代表者
渥美　紀寿（Atsumi, Noritoshi）

京都大学・学術情報メディアセンター・助教

研究者番号：７０３９７４４６

交付決定額（研究期間全体）：（直接経費）　　3,000,000円

研究成果の概要（和文）：本研究では，各バージョンにおける静的検査結果，メトリクス測定結果，利用ライブラリの情報を蓄積する環境を構築し，ソフトウェアの進化に基づく保守支援ツールを開発した．
品質保持や保守性向上のために用いられる静的解析ツールにおいて，警告箇所を管理する機能を実現し，開発者の負担を軽減できることを示した．
また，利用ライブラリの進化情報を収集し，OSS におけるライブラリ進化への対応方法を提示する手法によって，利用ライブラリの更新を促進する環境を構築した．

研究成果の学術的意義や社会的意義
ソフトウェアの不具合がもたらす社会的影響は非常に大きく，不具合を削減することが強く求められている．
我々はソフトウェアの不具合を削減するために，静的検査ツールの改善，ライブラリ進化に伴なう修正支援を実現した．
静的検査ツールは品質維持や保守性向上のために有益なツールであるが，問題点があり，それらを改善することで不具合の削減に貢献することが可能である．
また，ソフトウェアの開発では多数のライブラリが用いられるが，ライブラリの脆弱性も数多くあるにも関わらず，ライブラリの進化に伴なうソフトウェアの修正が行われていない．これらの対応により，潜在的な不具合を

研究成果の概要（英文）：We built an enviroment to store static analysis results, software metrics values and information of the used libraries, and implemented software maintenance tools based on software evolution.We impletemted a management tool of the alerts reported by static analysis checker.
This tool reduce the maintenance cost of developer. Moreover, we built an enviromnet to promote to update the version of using libraries based on the update information of other OSS.

研究分野：ソフトウェア工学

キーワード：ソフトウェア工学　ソフトウェア保守　ソフトウェア進化　プログラム解析

様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９、ＣＫ－１９（共通）

１．研究開始当初の背景

ソースコードの品質の保持や保守性の向上を目的として， GNU コーディング規約，MISRA-C，セキュアコーディング規約が提案されており，コーディング規約違反をチェックするツール（QA-C，ECLAIR, LDRA tool suite）や不具合につながると考えられる箇所をチェックするツール（Splint，CppCheck）などの静的検査ツールが実現されている．

これらの静的検査ツールを用いて開発中のソースコードを検査することは，ソースコードの品質保持や可読性向上に有用であるが，これらのツールによって検出される問題箇所は false-positive が非常に多いことや，問題箇所で指摘された内容が複雑でわかり難いなどの理由によって，多くの開発者あるいは開発プロジェクトでは利用されていない．ソフトウェアは進化し続けているため，ソースコードが変化するたびに静的検査を行い，その品質を維持する必要があるが，静的検査ツールではソースコードの進化に対応しておらず，毎回すべての警告箇所を確認する必要があり，多大な労力が必要となっている．

また，ソフトウェアは多くのライブラリを用いて構成されているが，ライブラリは開発中のソフトウェアとは独立して開発されており，それぞれが独自に進化していく．品質を保持するためにはライブラリの進化に追随する必要があるが，多くの開発者はライブラリの進化にはあまり関心がなく，ライブラリで発見された脆弱性の対応が遅れがちである．

２．研究の目的

本研究ではソフトウェアの進化において保守性や品質を維持するために，ソフトウェアの進化分析を行い，その情報を用いた保守支援環境を構築する．既存の静的検査ツールは個々の版を対象とした不具合の可能性がある箇所を検出するツールであり，過去の情報を利用した警告箇所の検出は実現していない．また，利用しているライブラリの更新の適用を支援する技術はあまり進歩していない．

本研究では進化情報を用いることによって，膨大な確認項目を減らし，修正支援を可能にする．さらにライブラリの更新に伴う修正支援を実現する．具体的には下記項目を実現し，提案手法の有効性を示す．図 1 に提案する保守支援環境の概要を示す．

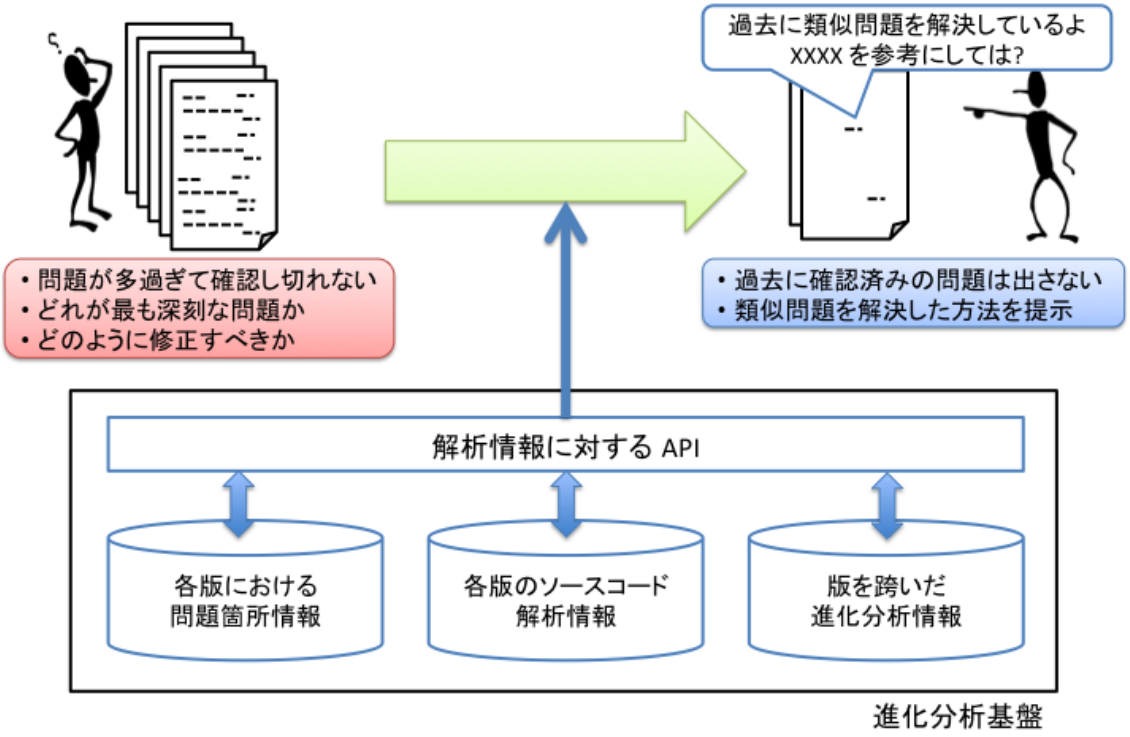

図 1 提案する保守支援環境の概要

① 各バージョンのソースコードに対する細粒度の解析結果を蓄積し，それらにアクセスするための進化分析基盤を構築
② 版間における構文要素間の対応関係を取得し，進化情報を蓄積
③ 進化情報に基づき，静的検査ツールにおける問題を解決するためのツールを構築
④ 利用しているライブラリに関する進化情報の蓄積
⑤ OSS におけるライブラリの進化に伴う修正情報の収集
⑥ 開発中のソフトウェアに対するライブラリの進化に伴う修正支援環境の構築

ソースコードの細粒度な情報として，構文解析・字句解析・意味解析の結果，依存関係の情報，コー ディング検査結果，メトリクスの測定結果，利用部品の情報を対応する構文要素の情報を付与して蓄積する．また，版間の差分情報を利用し，版間の対応関係を取得するツールを構築し，各情報のソフトウェアの進化に伴う変化 (進化情報)を容易に追跡できるようにする．
さらに，これらの情報を基 に保守性向上に有用な静的検査ツール等において，確認すべき警告箇所の削減および確認や修正を支援するためのツールを構築する．

３．研究の方法

(1) ソースコード解析基盤の構築
研究代表者が所属する研究グループにおいて，C，Java，JSP，JavaScript を対象とした構文解析器を開発している．版管理システムのリポジトリ URI を指定することによって初期バージョンから最新バージョンまでのソースコードをチェックアウトし，解析結果を XML 形式で格納する．
(2) 進化分析用の情報抽出ツール開発
進化分析用の情報として (a) LOC，サイクロマチック数などのメトリクス，(b) ライブラリ利用状況，(c) コーディング規約違反に関する情報を抽出する．それぞれの情報をバージョンごとに DB に記録し，バージョン間の変化を分析可能にする．
(3) バージョン間追跡基盤の構築
バージョン間の関係を分析するためには，バージョン間でソースコードがどのように変更されたかを把握する必要がある．これを実現するために git の blame 機能を用いて行ごとの変化を追跡し，プログラム要素間の関係を取得する．
(4) 進化分析フレームワークの構築
(1) 〜 (3) の各ツール群を連携し，品質保持や保守支援を行うための環境を構築する．

４．研究成果

(1) 静的検査ツールの利用促進するために,静的検査ツールが生成する警告に false-positive な警告がどの程度含まれているかを調査するとともに，ソースコードの変更に伴う検査コスト削減手法を提案した．
4 つのオープンソースソフトウェアを対象に FindBugs による警告が,バージョンの進化によってどのように変化するか，false-positive な警告がどの程度含まれているか調査を行った．その結果，ソースコード行数の増減に伴って警告数が増減しているが，警告をなくすためのコード改変は行われていないこと，false-positive な警告はそれほど多くないことがわかった．
また，静的検査における警告箇所の確認作業を効率化するために，バージョンごとに静的検査ツールが生成した警告箇所の記録と，それに対して開発者が false-positive および無視して良い警告を登録することを可能とし，それらの警告を再度検査実行時に表示しないツールを eclipse plugin として実装した．これらの研究により，静的検査ツールの警告情報を版間の対応関係とともに蓄積することが可能となった．

(2) C によるバリエーションソフトウェアを対象とした前処理による条件命令の構成方法の違いによる保守性に関する調査を行った．
C によるバリエーションソフトウェアでは，ソフトウェアを構成するフィーチャをマクロで表現し，マクロの組み合わせを前処理条件の条件式で用いることによって，コードを分割して実現する．各バリアントのためのコードを生成するための前処理条件の構成方法には様々な方法があり，その構成方法の違いと保守性にどのような関係があるかを調査した．本調査では保守性が低いコードは頻繁に改変されると仮定し，コードの改変と前処理条件の構成方法に関して調査した．
OSS を対象にバリアントコード(前処理条件によって制御されるコード)の改変頻度と，前処理条件，バリアントコードの行数，前処理条件の分岐数，前処理条件で使用されるマクロ数の

関係について，版管理リポジトリの開発履歴から情報を抽出し，分析した．その結果，バリアントコードの改変頻度が高い前処理条件が存在することは明らかとなったが，バリアンコードの行数，分岐命令数，マクロ数と改変頻度との間に関係はないことがわかった．
本調査において前処理条件について改版履歴を遡って追跡する方法と，その条件によって制御されるコードに関して分析するための基盤技術を実現することができた．

(3)ソフトウェアが利用している外部ライブラリの更新に伴う変更を支援するための手法を提案した．
外部ライブラリの更新では，API の引数の増減や型の変更などによりビルドに失敗することや，API の処理の変更などにより動作に変わることがある．そのため，外部ライブラリの更新に伴って，それを利用している箇所を修正する必要がある．
外部ライブラリの更新に伴なる変更を支援するためには,外部ライブラリの更新で API の変更にどのようなパターンが存在するか，調査する必要がある．これを調査するため，ライブラリが提供する API のシグネチャを解析するツールを開発した.本解析ツールを用いて既存のライブラリに対して各バージョン間の違いを分析することで，変更パターンを収集する．それぞれの変更パターンに対してどのように修正すべきかを開発者に提示することで，外部ライブラリの更新に伴う変更支援が可能となる．
Java では API を変更する場合，古い API には @deprecated アノテーションを付けることで,それを利用しているソフトウェアのビルド時に変更があることがわかるようになっている．多くの場合，@deprecated アノテーションが付けられた API は，新たな API を用いて実装されるため，そのコードを示すことで支援が可能である．
また,OSS の外部ライブラリおよびそれを利用した OSS を対象にそれぞれの外部ライブラリの更新に伴って，それを利用したソフトウェアの変更がドのように行われているか変更履歴から抽出するための手法を提案した．

(4) (3)の手法を実現するための環境を構築した．
本環境では、(a) OSS ライブラリを対象にリリースバージョンの更新履歴としてバージョン番号と個々のバージョンにおける公開 API の情報を蓄積，(b) OSS ソフトウェアを対象にそれぞれのソフトウェアの変更履歴から外部ライブラリの更新履歴としてソフトウェアのバージョンとライブラリ名およびそのバー ジョンを蓄積,(c) 開発中のソフトウェアで利用している外部ライブラリとそのバージョンから(a)，(b) の情報を利用し，自動更新可能なライブラリとそのバージョンの検出および手動更新するために必要な情報の提示を可能にした．
我々の調査によって外部ライブラリの自動更新可能なソフトウェアが多数存在することがわかった．一方で，既存のソフトウェアでは外部ライブラリの更新頻度が低く，外部ライブラリの更新に伴うソフトウェアの修正支援を十分に行うことができないことが明らかとなった．外部ライブラリの不具合によるソフトウェアの不具合を防ぐためには，外部ライブラリを更新することは重要である．我々が構築した外部ライブラリの更新に伴う修正支援環境を利用することによって，外部ライブラリの更新を促進することが可能となる．

５．主な発表論文等

〔雑誌論文〕（計　２件）

① Kunihiro Noda, Takashi Kobayashi, Noritoshi Atsumi, Identifying Core Objects for Trace Summarization by Analyzing Reference Relations and Dynamic Properties, IEICE Transactions on Information and Systems, Vol.E101-D, No.7,2018, pp.1751-1765.
DOI:10.1587/transinf.2017KBP0018

② 渥美 紀寿, 桑原 寛明, MAFP: ソースコードに対する静的検査における警告の管理ツール, コンピュータソフトウェア, Vol.33, No.4, 2016, pp.50-66.
DOI:10.11309/jssst.33.4_50

〔学会発表〕（計　９件）

① 渥美 紀寿, 桑原 寛明, ライブラリ進化への追随のためのソフトウェア修正の共有手法の提案, 電子情報通信学会 ソフトウェアサイエンス研究会, 2019.

② 夏目雅槻, 相澤遥也, 渥美紀寿, 小林 隆志, ソースコードの XML 表現のための選択例を用いた対話的 XPath 生成支援, 電子情報通信学会 ソフトウェアサイエンス研究会, 2018.

③ Kunihiro Noda, Tatsuya Toda, Takashi Kobayashi Noritoshi Atsumi, Identifying Core Objects for Trace Summarization Using Reference Relations and Access Analysis, The 41st Annual IEEE Computer Software and Applications Conference, 2017.

④ 野田 訓広, 小林 隆志, 渥美 紀寿, 実行トレース抽象化を目的とした参照 関係・アクセス解析によるコアオブジェクト特定, 情報処理学会 ソフトウェア工学研究会, 2017.

⑤ 森 達也, 小林 隆志, 林 晋平, 渥美 紀寿, 前処理命令による可変点を考慮した共変更ルール抽出, 電子情報通信学会 ソフトウェアサイエンス研究会, 2017.

⑥ 渥美 紀寿, バリアントソフトウェアの開発・保守支援に関する研究, 情報処理学会 ウィンターワークショップ 2017・イン・飛騨, 2017.

⑦ 今西 洋二, 渥美 紀寿, 森崎 修司, 山本 修一郎, 阿草 清滋, バリアントコードの改変履歴に基づく前処理条件の構造に関する特徴調査, 日本ソフトウェア科学会 ソフトウェア工学の基礎ワークショップ, 2016.

⑧ 今西 洋二, 渥美 紀寿, 森崎 修司, 山本 修一郎, 阿草 清滋, 前処理命令の制御構造とその構造内のコード改変に関する調査, 情報処理学会 ソフトウェア工学研究会, 2016.

⑨ 桑原 寛明, 渥美 紀寿, ソースコードの静的検査における警告の版間追跡ツール, 情報処理学会 ソフトウェアエンジニアリングシンポジウム, 2015.

※科研費による研究は、研究者の自覚と責任において実施するものです。そのため、研究の実施や研究成果の公表等については、国の要請等に基づくものではなく、その研究成果に関する見解や責任は、研究者個人に帰属されます。