様　式　Ｃ－１９、Ｆ－１９、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

平成 ２７ 年　 ６ 月　 ９ 日現在

機関番号：１４６０３
研究種目：若手研究(B)
研究期間：2013～2014
課題番号：２５７３００４５

研究課題名（和文）ソフトウェア欠陥部品の自動特定に関する研究

研究課題名（英文）A Study of Localizing Buggy Module in Large-scale Software Development

研究代表者
伊原　彰紀（Akinori, Ihara）

奈良先端科学技術大学院大学・情報科学研究科・助教

研究者番号：４０６３８３９２

交付決定額（研究期間全体）：（直接経費）　　2,500,000円

研究成果の概要（和文）：本研究はソフトウェアの障害を引き起こした原因となる欠陥部品の自動特定技術を開発した．ソフトウェア開発者は，障害報告書の内容から欠陥部品を特定し，修正する．複数のシステムが協調的に機能するサービスに混入する欠陥を修正するためには，システムの幅広い知識や経験を要する．本研究は，膨大なソフトウェア開発データのマイニング技術を用いて欠陥部品，及び，複数システムに波及する同時修正（co-change）箇所の自動特定技術を開発し，保守作業の効率化，及び，人手による修正箇所の見落としを減らすことを目指す研究である．

研究成果の概要（英文）：The goal of this study proposed a method to identify buggy modules that cause problems in software. In general, software developers often localize buggy modules based on issue reports posted by users, and they fix the modules. However, if developers do not have extensive knowledge and experience of the software, they might take time to localize buggy files, and are likely to miss buggy files that should be changed to fix the problems. In order to understand the reports and localize correct buggy files, this study built a model to identify buggy modules with machine-learning approach.

研究分野：ソフトウェア工学

キーワード：ソフトウェア開発　リポジトリマイニング　欠陥箇所特定技術

１．研究開始当初の背景

大規模ソフトウェア開発では，プロダクトに発見される障害を解決するために，開発者はユーザからの障害報告書に基づいてソフトウェアを構成する１万以上のソースコードファイルの中から欠陥を含むファイル群を突き止める．特に，大規模オープンソースソフトウェア(OSS)プロジェクトでは，膨大な障害報告を日々受けており（Mozilla プロジェクトでは年間 6 万件以上）[Guo_ICSE10]，全ての障害原因を突き止め修正するには膨大なコストがかかる．その理由として，欠陥部品を特定することの難しさが，障害報告書の品質（情報量，読みやすさ等）に依存していることが挙げられる．従来研究では障害報告書の書き方に関する研究が進められている[Betternburg_FSE08]が，品質の高い障害報告書が完成したとしても，欠陥部品を特定するためには豊富な開発経験を要し，OSS 開発やオフショア開発のような開発拠点が分散している開発環境では，障害を解決するための知識を持っている開発者を見つけ出すことは容易ではない．

また，昨今では複数のシステムが協調的に機能するサービスが増加し，欠陥が及ぼす障害波及範囲も拡大しているため，複数人による欠陥修正は必要不可欠となっている．つまり，単独の開発者の知識や経験のみに頼った欠陥修正は現実的に不可能になっている．複数人の開発者による協調作業は，欠陥混入箇所が明らかにならない限り，修正担当者の決定が困難であり，欠陥の修正時間の長期化に影響している[伊原_Infosocio12]．

障害解決を迅速に対応するためには，OSS 開発やオフショア開発のような分散開発を行っている場合，障害の原因を理解できる開発者が早急に対応することが必要であり[Ihara_SATA12]，欠陥部品の特定はもとより，当該部品の修正と同時に，変更すべき部品(co-change)部品を特定することも困難となる．今後もソフトウェアの大規模化，複雑化が進み，分散開発が主流になる昨今，保守作業効化のためには，ソフトウェア欠陥部品の自動特定が重要である．

２．研究の目的

本研究課題はソフトウェアに発見された障害を引き起こす原因となる欠陥部品の特定を自動化することで，ソフトウェア保守作業を効率化することである．

ソフトウェアの欠陥修正では，1 件の欠陥修正のために変更する部品は 1 つ（1 ファイル）とは限らず，欠陥が混入している部品と依存関係のある部品も同時修正せざるを得ないことが多い．本研究では，欠陥部品の特定と，同時に修正を要する co-change 部品を特定する技術開発を行い，評価実験を行った．研究代表者のこれまでの研究から，OSS 開発

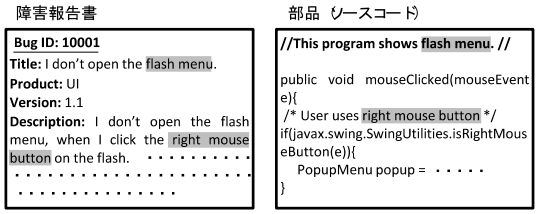

図1　障害報告書と部品の特徴語の一致

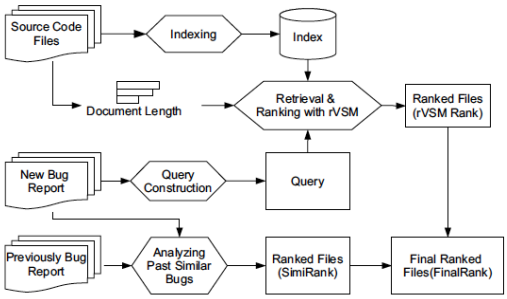

図2　欠陥部品特定手法の概略図

者は，欠陥の修正に 1 か月（約 30 日）かかっていた中で，ソフトウェアを構成する大量の部品の中から欠陥部品を特定するために約 10 日かけていたことが分かっている．つまり，本研究課題のソフトウェア保守領域を自動特定することにより，欠陥部品を特定するために要していた最大約 10 日(修正時間 1 カ月の約 30%）の削減を期待できる．

３．研究の方法

本研究開始当初には(1)欠陥部品の特定，及び，(2) 修正を要する co-change 部品群の特定について提案していた．本研究では(1)(2)に加え，提案手法を実用的な技術の確立に向けて，(3) 変更時期に基づく欠陥部品の推薦，(4) 推薦する欠陥部品の粒度による効果についての研究を行った．本章では，それぞれの手順について述べる．

(1) 欠陥部品の特定

障害報告書，及び，ソースコード中のコメント欄には，開発者間で障害の内容やソースコードの動作を共有するために，図１のように文章中に特徴的な単語（以下，特徴語）が埋め込まれていることが多い．本研究課題では，障害報告書と部品中の特徴的な単語の一致を基に修正を要する欠陥部品を推薦する．具体的な手順①-⑤を図 2 に示す．

① 障害報告書，部品から特徴語の抽出

障害報告書，部品に頻出する単語，且つ，全てのドキュメントに頻出しない単語を特徴語とし，テキストマイニング技術 tf-idf 技術を用いて抽出する．

② ソースコード規模による重み付け

規模の大きい部品には欠陥が混入されやすいことが従来研究より明らかにされていることから，ロジスティック回帰を用いて，規模の大きい部品に対して重み付けを行う．

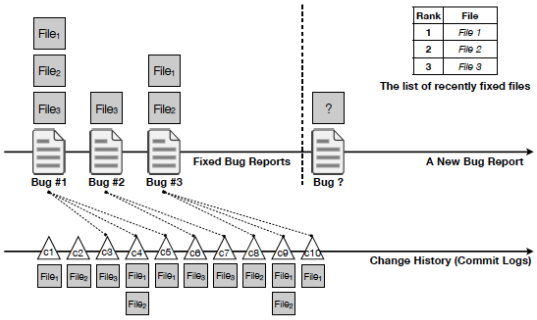

図3 変更時期に基づく欠陥部品推薦

③ 障害報告書と部品の類似度順位の計測

手順②において，障害報告書に記載されているキーワードが，数多く発見された部品を探索し，部品規模を考慮した順位値（rVSM Rank）を計測する．

④ 欠陥混入記録を持つ部品の特定

過去に修正された部品は，再び欠陥が混入する可能性が高いため，欠陥混入可能性の高い部品を特定し，順位値（SimRank）を計測する．

⑤ 欠陥混入部品の順位付け

手順③④でそれぞれ計測した順位値を統合し，最終的な順位を決定する．

⑥ 修正を要する co-change 部品群の特定

手順⑤で推薦した部品の変更記録を追跡し，当該部品と同時に変更した部品を特定する．同時に変更した回数が多い部品を当該部品に加えて推薦する．

(2) 変更時期に基づく欠陥部品の推薦

部品の変更直後，欠陥が発見される可能性が高いという従来研究の知見から，本研究では，部品の変更時期に基づく欠陥混入の推薦に取り組んだ．(1)で推薦した部品の変更記録を追跡し，直近に変更された部品に重み付けを行い，再度順位付けを行う．図 3 の例で示すように，予測時点から時間をさかのぼった場合に File1，File2，File3 の順で変更され，欠陥部品として推薦される．

(3) 推薦する欠陥部品の粒度による効果

(1)(2)では，欠陥部品の粒度をファイルレベルで推薦していた．最終的に，開発者は推薦された欠陥部品（ファイル）を手動で確認する必要があるが，推薦された部品が大規模なファイルである場合，ファイル中の修正すべき箇所を特定するのに時間がかかる．従って，限られた開発工数の中で効率的に修正すべき箇所を特定するためには，より細粒度な推薦技術が求められる．図 4 は推薦粒度による効果の違いを示している．限られた開発工数（図中の LOC threshold）の中で，図中の右側（Class Level）では推薦技術により Class1 と Class2 が推薦され，それぞれには Function A, D と Function B, E が存在する．実際修正すべき箇所を黒の楕円で示しており，Function A, B が修正すべき箇所である．図中の左側のケース（Function Level）は推薦粒度を Function レベルで行った場合，Class Level で推薦した場合に発見できなかった Function C が推薦できている．図 4 のように細粒度で欠陥部品を推薦することで，効率的に欠陥部品を推薦することが実現できるため，本研究では，障害報告書に基づく欠陥部品推薦技術におけるメソッドレベルで推薦する場合の効果について実験を行った．

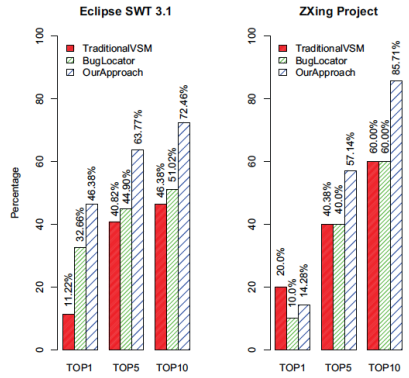

図5 co-change部品の変更記録を用いた欠陥部品推薦モデル予測精度の比較

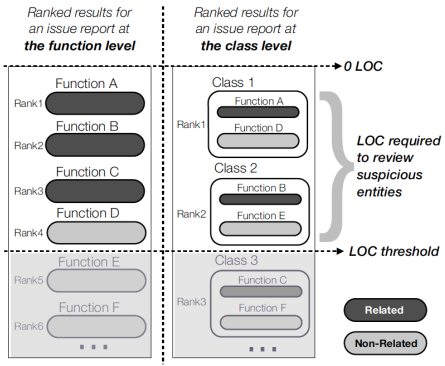

図4 推薦する欠陥部品の粒度による効果

4．研究成果

3 章で述べた(1)欠陥部品の特定は，国際会議 SNPD[Tantithamthavorn_SNPD13]，(2) 変更時期に基づく欠陥部品の推薦の成果を ISSRE[Tantithamthavorn_ISSRE13]，(3) 推薦する欠陥部品の粒度による効果の成果は APRES[Tantithamthavorn_APRES14] においてそれぞれ発表した．本章ではそれぞれの成果を述べる．

(1) 欠陥部品の特定

実験では Eclipse SWT 3.1，Android ZXing が管理する障害報告書の中からランダムに抽出した 98 件，20 件を対象とし，従来手法 VSM（Vector Space Model），及び，BugLocator[Zhou_ICSE12] の予測精度（順位上位 1 ファイル，5 ファイル，10 ファイルを推薦した場合の予測精度）と比較実験を行った．図 5 は予測精度の結果を示す．ZXing プロジェクトの順位上位 1 ファイルを推薦した場合を除き，従来手法に比べ 1.4 倍の精度で予測できることがわかった．

(2) 変更時期に基づく欠陥部品の推薦

Eclipse Platform，Eclipse JDT の開発記録データセットを用いて従来手法との予測精度の比較を行った．データセットの概要は表 1 に示す．表 2 は予測精度の結果の示す．従来手法である VSM モデルに比べて，1.18 倍～1.94 倍の精度向上が見られた．

表1 データセットの概要

Project	Study Period	# Bugs	# Files
Eclipse Platform	Apr 2002 - Jan 2013	744	1,758
Eclipse JDT	Jun 2002 - Mar 2013	468	4,222

表2 変更時期に基づく欠陥部品予測モデルの精度

Project	Approach	Top 1	Top 5	Top 10	Top 20	Top 30	MRR	MAP
Eclipse Platform	**Our Approach**	**14.25%**	**40.46%**	**53.36%**	**64.92%**	**71.77%**	**0.2692**	**0.2394**
	Vector Space Model Approach	12.10%	32.39%	43.15%	55.38%	63.58%	0.2247	0.1890
	Cache-based Approach	11.56%	23.66%	31.18%	44.62%	53.36%	0.1810	0.1762
Eclipse JDT	**Our Approach**	**11.97%**	**32.91%**	**44.66%**	**53.42%**	**58.76%**	**0.2208**	**0.1611**
	Vector Space Model Approach	6.17%	19.23%	26.70%	35.90%	41.45%	0.1327	0.0940
	Cache-based Approach	9.19%	25.00%	35.04%	45.73%	54.49%	0.1680	0.1573

(3) 推薦する欠陥部品の粒度による効果

Eclipse Platform, Eclipse PDE, Eclipse JDT の開発記録データセットを用いて，開発工数による各推薦粒度（ファイルレベル，メソッドレベル）の予測精度を比較実験を行った．データセットの概要は表 3 に示す．図 5 は予測精度の違いを示す．図 5 の横軸はプロジェクト，または，開発者の開発工数を示し，縦軸は，開発工数が与えられた時に発見できる欠陥部品の予測精度を示す．例えば，開発者が欠陥部品を調査するために 1000 行を読むだけの工数を持つ場合，本研究で提案したモデルがメソッドレベルで推薦した場合とファイルレベルで推薦した場合で，どちらが数多くの欠陥を発見することができるかを比較している．比較を行った結果，いずれのプロジェクトにおいても 1000 行の開発工数ではメソッドレベルで推薦した場合の方が数多く欠陥部品を発見することができることが明らかになった．Eclipse Platform, Eclipse PDE では，開発工数が増えるほど，欠陥部品を発見する確率に違いがなくなることもわかった．

＜引用文献＞

[Guo_ICSE10] Philip J. Guo, Thomas Zimmermann, Nachiappan Nagappan, and Brendan Murphy. Characterizing and predicting which bugs get fixed: an empirical study of Microsoft Windows. In Proceedings of the 32nd International Conference on Software Engineering (ICSE'10), pp.495-504, 2010.

[Betternburg_FSE08] Nicolas Bettenburg, Sascha Just, Adrian Schroter, Cathrin Weiss, Rahul Premraj, and Thomas Zimmermann. What makes a good bug report?. Proceedings of the 16th International Symposium on Foundations of software engineering (FSE'08), pp.308-318, 2008.

[伊原_Infosocio12]伊原彰紀，大平雅雄，松本健一，``OSS 開発における不具合修正プロセスの現状と課題 ：不具合修正時間の短縮化へ向けた分析 ’ ’，情報社会学会誌，Vol.6, No.2, 2012.

[Ihara_SATA12]Akinori Ihara, Yasutaka Kamei, Akito Monden, Masao Ohira, Jacky Wai Keung, Naoyasu Ubayashi, and Ken-ichi Matsumoto. An Investigation on Software Bug Fix Prediction for Open Source Software Projects -A Case Study on the Eclipse Project -. Proceedings of the

表3 データセットの概要

Project Name	Study Period	Issues	# of classes	# of functions	# LOC
Eclipse Platform	May 2, 2001 - Dec 31, 2012	744	1,758	7,121	165,404
Eclipse PDE	June 5, 2001 - Dec 31, 2012	756	4,979	26,339	271,002
Eclipse JDT	May 24, 2001 - Dec 31, 2012	468	4,222	49,486	1,076,985
	Total	1,968	10,959	82,946	1,513,391

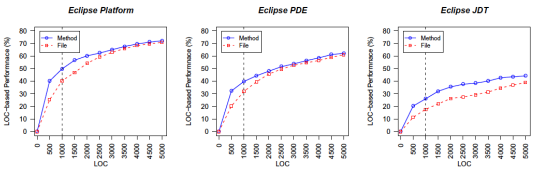

図5 開発工数による予測精度の違い

International Workshop on Software Analysis, Testing and Applications (SATA '12), pages.112-119, 2012.

[Zhou_ICSE12] Jian Zhou, Hongyu Zhang, and David Lo, ``Where should the bugs be fixed? - more accurate information retrieval-based bug localization based on bug reports, ’ ’ Proceedings of the 34th International Conference on Software Engineering (ICSE ’ 12), pages.14-24, 2012.

[Tantithamthavorn_SNPD13]Chakkrit Tantithamthavorn, Akinori Ihara, and Ken-ichi Matsumoto, ``Using Co-Change Histories to Improve Bug Localization Performance, ’ ’ Proceedings of 14th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD ’ 13), pages.543-548, 2013.

[Tantithamthavorn_ISSRE13]Chakkrit Tantithamthavorn, Rattamont Teekavanich, Akinori Ihara, and Ken-ichi Matsumoto, ``Mining a Change History to Quickly Identify Bug Locations : a Case Study of the Eclipse Project," Proceedings of the IEEE 24th International Symposium on Software Reliability Engineering (ISSRE ’ 13), pages.108-113, November 2013.

[Tantithamthavorn_APRES14]Chakkrit Tantithamthavorn, Akinori Ihara, Hideaki Hata, and Kenichi Matsumoto, ``Impact Analysis of Granularity Levels on Feature Location Technique, ’ ’ Proceedings of the 1st Asia Pacific Requirements Engineering Symposium (APRES2014), pages.135-149, 2014.

５．主な発表論文等
（研究代表者、研究分担者及び連携研究者には下線）

〔学会発表〕（計３件）
① Chakkrit Tantithamthavorn, Akinori Ihara, Hideaki Hata, and Kenichi Matsumoto, ``Impact Analysis of Granularity Levels on Feature Location Technique,'' Proceedings of the 1st Asia Pacific Requirements Engineering Symposium (APRES'14), pages.135-149, 2014, Auckland (New Zealand).
② Chakkrit Tantithamthavorn, Rattamont Teekavanich, Akinori Ihara, and Ken-ichi Matsumoto, ``Mining a Change History to Quickly Identify Bug Locations : a Case Study of the Eclipse Project," Proceedings of the IEEE 24th International Symposium on Software Reliability Engineering (ISSRE'13), pages.108-113, November 2013, Pasadena, California (USA).
③ Chakkrit Tantithamthavorn, Akinori Ihara, and Ken-ichi Matsumoto, ``Using Co-Change Histories to Improve Bug Localization Performance,'' Proceedings of 14th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD '13), pages.543-548, 2013, Honolulu, Hawaii (USA).

６．研究組織
(1)研究代表者
伊原　彰紀（IHARA, Akinori）
奈良先端科学技術大学院大学・情報科学研究科・助教
研究者番号：40638392