Blocking Bugの早期修正支援に向けた特徴分析

金城 清史　　大平 雅雄
和歌山大学システム工学部
E-mail: {161017, masao}@sys.wakayama-u.ac.jp

概要

Blocking Bug とは，他の不具合修正を妨害する不具合である．*Blocking Bug* が修正されない限り関連する他の不具合 *(Blocked Bug)* も修正することができないため，*Blocking Bug* は優先的に修正されるべき不具合である．しかし，*Blocking Bug* の修正にはより多くの時間を要することが知られている．本稿では，不具合報告の依存関係，欠陥を有するモジュールの依存関係，モジュール開発者の依存関係といった *Blocking Bug* の原因となる重畳的依存関係を分析する．*Blocking Bug* の早期修正を支援するための知見を得ることを目的としている．

1　はじめに

ソフトウェア開発では，ソフトウェア保守に要するコストがソフトウェアライフサイクル全体に要するコストの大半を占めることが知られている [2, 9]．そのため，ソフトウェア保守の改善支援を目的とする研究が盛んに行われている．例えば，不具合修正コストを見積もるための不具合修正時間予測手法 [10] や不具合修正タスクの割当て支援手法 [1] などがある．

先行研究の多くは不具合修正の効率化を目的とするものの，個々の不具合の特徴や影響範囲を考慮していなかった．しかし，特に最近では，報告された不具合がユーザの満足度や開発スケジュールの変更など開発プロセスへ与える影響の大きさを考慮した研究が増えてきている．保守プロセスに充てられるコストやリソースには限りがあり，全ての不具合を同等に扱うのではなく，High Impact Bug[5] と呼ばれる影響度の大きな不具合 [3, 6, 8] を優先的に修正することの必要性が認識されてきたためと思われる．

High Impact Bug の内，本研究では特に，Blocking Bug [4] を対象にする．Blocking Bug とは，他の不具合修正を妨害する不具合である．Blocking Bug が修正されない限り関連する他の不具合 (Blocked Bug) も修正することができないため，Blocking Bug は優先的に修正されるべき不具合である．しかし，Blocking Bug の修正にはより多くの時間を要することが知られている．

Garcia らの [4] 研究では，新たに報告された不具合が Blocking Bug かどうかを予測する予測モデルが提案されているが，Blocking Bug が生じる原因や予防する方法については述べられていない．そのため本稿では，不具合報告の依存関係，欠陥を有するモジュールの依存関係，モジュール開発者の依存関係といった Blocking Bug の原因となる重畳的依存関係を分析し，Blocking Bug の早期修正と予防を支援するための知見を得ることを目指している．

2　Blocking Bugとその問題点

本研究では，他の不具合の修正を妨害する不具合のことを Blocking Bug と呼ぶ．図 1 を用いて Blocking Bug の特徴と問題点を説明する．

報告された不具合 A は，不具合 B によって修正を妨害されており，不具合 B が先に修正されない限り不具合 A を修正することができない状態にある．図のように，不具合 B は不具合 A を修正しようとする中で Blocking Bug であることが判明する場合もあれば，不具合 B が先に検出された後に不具合 B の修正内容と依存関係のあるコード中に不具合 A を検出する場合もあり得る．いずれにしても，たとえ不具合 A そのものの修正は軽微であった

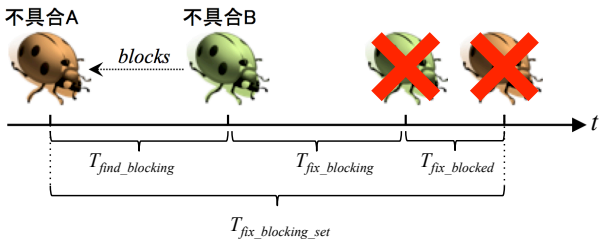

図 1. 不具合修正を阻害する Blocking Bug

としても，不具合 B が修正されない限り不具合 A を修正することができない．したがって，不具合 A を修正するには，本来必要な時間 ($T_{fix\_blocked}$) よりも多くの時間 ($T_{fix\_blocking\_set}$) が必要となる．不具合修正の遅延は顧客満足度の低下に繋がるため，できるだけ早く Blocking Bug を検出する必要がある．

しかしながら，不具合 A の担当者は不具合 B を修正するためのコードやモジュールについて熟知しているとは限らず，不具合 B，すなわち，Blocking Bug の検出自体に時間 ($T_{find\_blocking}$) がかかってしまう場合もある．また，Blocking Bug である不具合 B を検出できたとしても，不具合 A と不具合 B の担当者が異なる場合はお互いに情報交換しながら修正作業を行う必要があるため，Blocking Bug の修正は通常の不具合よりも多くの時間 ($T_{fix\_blocking}$) を必要とすることがある．実際，6 つのオープンソースプロジェクトを対象とした Garcia らの研究 [4] では，Blocking Bug は，Non-Blocking Bug (Blocking Bug 以外の不具合）に比べ 15–40 日程度の修正時間が長くなるとされている．なお，Blocking Bug が検出されるまでの時間 ($T_{find\_blocking}$) は短い場合もあれば長い場合もあり (3–18 日)，プロジェクト毎に傾向は異なる．プロジェクトで開発されるプロダクトの複雑度やコンポーネント間の結合度に関連があるように思われる．Garcia らの分析結果を受けて，Blocking Bug の発生原因を理解することが Blocking Bug を早期に検出したり予防したりするのにより重要であると考えるに至った．

3 Blocking Bug の発生原因の支援に向けて

Garcia ら [4] の研究では，前述した分析結果を踏まえて，プロジェクトに新たに報告される不具合が Blocking Bug として扱うべきかを予測するためのモデルを構築している．不具合修正の優先度や重要度を予測する既存研究 [7] と類似するものであるが，既存研究は優先度（優先度 P1–P5）や重要度（Blocking, Critical, Major, Normal, Minor, Trivial など）のすべてのカテゴリを扱うため，個々のカテゴリの予測精度はさほど高くなく，いくつかのカテゴリを統合する（P1–P2 を高，P3 を中，P4–P5 低，と再カテゴリ化する）ことで予測精度をある程度確保していた．一方，[4] では，不具合管理プロセスにおいて最も重要度が高いものとして通常設定される Blocking Bug のみを対象とすることで比較的高精度に Blocking Bug 予測できるため,（優先度の高くない不具合まで分類する必要がないという文脈では）より実用的であると言える．

また，Garcia ら [4] は，対象とした 6 つのプロジェクトのうち 5 つのプロジェクト (Chromium, Eclipse, FreeDesktop, NetBeans, OpenOffice) において，不具合報告に記述されるコメントの内容 (comment text) が構築した予測モデルの予測精度に最も寄与することを明らかにしている（Mozilla プロジェクのみ，同報リスト (CC list) に含まれる開発者の人数）．だたし，Blocking Bug が発生する原因や Blocking Bug を予防するための方法については十分な分析が行われていないため不明なままである．

本研究の最終目的は，Blocking Bug の早期修正を支援する（あるいは Blocking Bug の発生を予防する）ことである．そのための第一歩として，Blocking Bug の発生原因を理解する必要があり，今後は以下の分析に取り組む予定である．

- (1) Blocking Bug（不具合票）の関係の分析（図 2 上段）
- (2) 修正対象ファイル（コンポーネント）の関係の分析（図 2 中段）
- (3) 修正担当者間の関係の分析（図 2 下段）
- (4) (1) と (2)，(2) と (3)，(1) と (3)，(1)~(3) の横断的分析

4 おわりに

本稿では，Blocking Bug が不具合修正プロセスに与える影響について議論した．また，先行研究の問題点を指摘し，Blocking Bug の発生原因を理解する必要性について

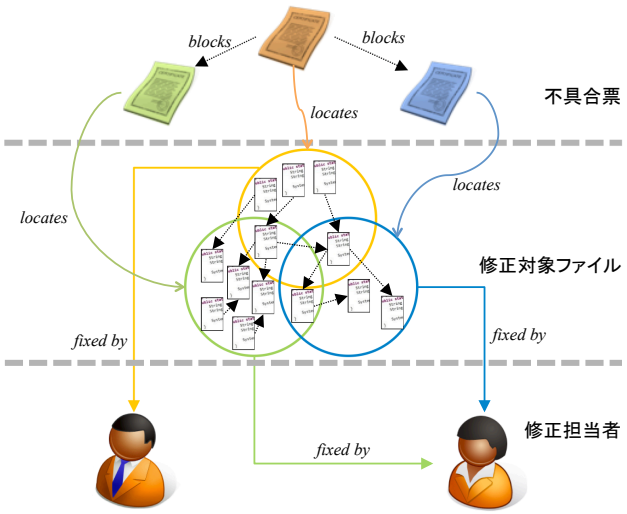

図 2. Blocking Bug の発生原因を理解するための分析フレームワーク

述べた．今後は，Blocking Bug の発生原因を特定すべく，Blocking Bug に関連する不具合票，ファイル（コンポーネント），修正担当者らの関係を調査する予定である．

謝辞

本研究の一部は，独立行政法人情報処理推進機構が実施した「2013 年度ソフトウェア工学分野の先導的研究支援事業」の支援および文部科学省科学研究補助金（基盤 (C): 24500041）による助成を受けた．

参考文献

[1] John Anvik and Gail C. Murphy. Reducing the effort of bug report triage: Recommenders for development-oriented decisions. *ACM Trans. Softw. Eng. Methodol.*, 20(3):10:1–10:35, August 2011.

[2] Alain April and Alain Abran. *Software Maintenance Management: Evaluation and Continuous Improvement*. Wiley-IEEE Computer Society Press, 2008.

[3] Tse-Hsun Chen, Meiyappan Nagappan, Emad Shihab, and Ahmed E. Hassan. An empirical study of dormant bugs. In *Proceedings of the 11th Working Conference on Mining Software Repositories (MSR '14)*, pages 82–91, 2014.

[4] Harold Valdivia Garcia and Emad Shihab. Characterizing and predicting blocking bugs in open source projects. In *Proceedings of the 11th Working Conference on Mining Software Repositories (MSR '14)*, MSR 2014, pages 72–81, New York, NY, USA, 2014. ACM.

[5] Yutaro Kashiwa, Hayato Yoshiyuki, Yusuke Kukita, and Masao Ohira. A pilot study of diversity in high impact bugs. In *Proceedings of 30th International Conference on Software Maintenance and Evolution (ICSME2014)*, pages 536–540, 0 2014.

[6] Adrian Nistor, Tian Jiang, and Lin Tan. Discovering, reporting, and fixing performance bugs. In *Proceedings of the 10th Working Conference on Mining Software Repositories (MSR '13)*, pages 237–246, 2013.

[7] M. Sharma, P. Bedi, K.K. Chaturvedi, and V.B. Singh. Predicting the priority of a reported bug using machine learning techniques and cross project validation. In *12th International Conference on Intelligent Systems Design and Applications (ISDA) 2012*, pages 539–545, Nov 2012.

[8] Emad Shihab, Audris Mockus, Yasutaka Kamei, Bram Adams, and Ahmed E. Hassan. High-impact defects: A study of breakage and surprise defects. In *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering (ESEC/FSE '11)*, pages 300–310, 2011.

[9] Gregory Tassey. The economic impacts of inadequate infrastructure for software testing. Technical report, National Institute of Standards and technology, 2002.

[10] 正木 仁, 大平 雅雄, 伊原 彰紀, and 松本 健一. Oss 開発における不具合割当パターンに着目した不具合修正時間の予測. **情報処理学会論文誌**, 52(2):933–944, 2 2013.