High Impact Bug が与える影響の分析に向けて

柏　祐太郎[†1]　　吉行　勇人[†1]　　大平　雅雄[†1]

Mining Software Repositories (MSR) 分野では，報告された不具合を適任の開発者に割り当てるためのバグトリアージ手法が盛んに研究されてきた．しかし，個々の不具合がプロダクトや開発（スケジュール）に与える影響については考慮しておらず，不具合割り当てにおけるプロジェクトマネージャの意思決定を支援するためには十分であるとは言い難い．報告される不具合には，タイプミスといった軽微な不具合からセキュリティに関する緊急度の高い不具合まで多岐に渡るため，細粒度の不具合分類と高精度のコスト見積もりに基づくバグトリアージ手法が求められている．本ワークショップでは，不具合の分類方法や不具合が開発者やユーザに与える影響を計測する方法について議論したい．

Toward Analyzing Effects of High Impact Bug

Yutaro Kashiwa,[†1] Hayato Yoshiyuki[†1] and Masao Ohira[†1]

Since increasing complexity and scale of modern software products imposes tight scheduling and resource allocations on software development projects, a project manager must carefully triage bugs to determine which bug should be necessarily fixed before shipping. In order to detect high impact bugs, in this early stage of the study, we analyze bugs to understand which bug highly impacts on users or developers.

1. はじめに

大規模・複雑化する近年のソフトウェア開発では，製品出荷後にも多くの不具合が報告される．一般に，プロジェクトの人的リソースやスケジュールには限りがあるため，すべての不具合を迅速に修正することは困難である．プロジェクトマネージャは，報告された多くの不具合を，すぐに修正すべき緊急度の高い不具合，次のリリースまでは修正すべき不具合，次のリリースでも修正しない不具合などに分類し，不具合が報告される度に動的に人的リソースを割り当てスケジュールを調整する必要がある．

Mining Software Repositories (MSR) 分野では，報告された不具合を適任の開発者に割り当てるためのバグトリアージ手法 [1,2] が盛んに研究されてきたものの，個々の不具合がプロダクトや開発（スケジュール）に与える影響については考慮しておらず，不具合割り当てにおけるプロジェクトマネージャの意思決定を支援するためには十分であるとは言い難い．報告される不具合には，タイプミスといった軽微な不具合からセキュリティに関する緊急度の高い不具合（他の不具合よりも優先して対処すべき不具合）まで多岐に渡るため，細粒度の不具合分類と高精度のコスト見積もりに基づくバグトリアージ手法が求められている．

本研究は，日々報告される多くの不具合の中から，バグトリアージの効果に大きな影響を与える不具合を自動分類するための手法の構築を目標としている．その初期段階として，我々は現在，開発者やユーザーに大きな影響を与える不具合を文献調査に基づいて分類し，実際の不具合がどのような分布で存在しているのかを確かめるためのケーススタディを実施している．

次章では，開発者やユーザーに大きく影響を与える不具合 (High Impact Bug) に関する文献を紹介する．

2. High Impact Bug

近年では，開発者やユーザーに大きな影響を与える不具合 (High Impact Bug) を対象とした分析が盛んに行われている [3–8]．既存研究の分析結果を踏まえ，我々は，不具合が主に影響を与えるステークホルダの観点から，まず不具合を Process Bug と Product Bug に大別し，それぞれのカテゴリに 3 種類の不具合が存在することを示した [3]．以降では，Process Bug と Product Bug に属するそれぞれの不具合を紹介する．

2.1 Process Bug

Process Bug の主なステークホルダは開発者である．Process Bug に属する不具合は，開発スケジュールの

†1 和歌山大学大学
Wakayama University

変更を余儀なくさせる [4] とされており，開発者に大きな影響を与える．

- **Surprise Bug** [4]
  予期せぬ箇所（リリース前にあまり変更されないファイルなど）かつ，予期せぬ時期（特にリリース直後）に発生する不具合を指す．
- **Dormant Bug** [5]
  あるリリース (ver. x) に向けて開発を行っている際に埋め込まれるが，テスト中またはリリース後には見つからずに，次のリリース (ver. x+1) の後に見つかる不具合を指す．Dormant Bug は，Dormant Bug 以外の不具合よりも修正時間が短いことが報告されている．
- **Blocking Bug** [6]
  ソフトウェアコンポーネントの依存関係のために，他の不具合の修正を阻害している不具合を指す．Blocking Bug は，Blocking Bug 以外の不具合よりも修正時間が長いことが報告されている．

2.2 Product Bug

Product Bug の主なステークホルダはプロダクトのユーザである．Security Bug [7] や Performance Bug [8]，Breakage Bug [4] が該当する．これらの不具合は，プロダクトに対するユーザーの満足度や信頼度の低下などを招くとされている．

- **Security Bug** [7]
  それ自体が何かに影響を及ぼすものではなく，攻撃者により悪用される可能性がある不具合を指す．Security Bug は，Security Bug 以外の不具合よりも複雑であるため再発しやすいが，緊急度が高いため修正時間が短いと報告されている．
- **Performance Bug** [8]
  プロダクトのパフォーマンス低下を招く不具合である．ユーザーエクスペリエンスやレスポンス，スループットの低下などを含む．Performance Bug は，Performance Bug 以外の不具合よりも修正時間が長いことが報告されている．
- **Breakage Bug** [4]
  コード修正や機能追加により，以前使えていた機能を使えなくするような不具合を指す．

3. High Impact Bug 研究の今後

[3] において我々は，4 つの OSS プロジェクトの不具合報告をランダムに 100 件ずつ抽出し分析を行った．その結果，上述した 6 種類の High Impact Bug をすべてのプロジェクトにおいて観察できた．しかし，各プロジェクトの 100 件分の不具合に占める High Impact Bug の割合は小さいため，一般性のある一貫した知見を導くことはできなかった．そのため，現在我々は，1 プロジェクトあたり 1,000 件（プロジェクトに報告された全不具合の 10%程度）を対象に分析を進めている最中である．

本ワークショップでは，High Impact Bug として分析の対象にすべき新たな種類の不具合や，開発者やユーザに与える影響を計測する方法（例えば，既存研究 [4–8] では，不具合が開発者に割当てられるまでの時間や不具合が開発者に割当てられてから修正されるまでの時間，修正範囲（ファイル数）など）について議論したい．

謝辞 本研究の一部は，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．また，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」の支援を受けた．

参 考 文 献

1) Anvik, J., Hiew, L. and Murphy, G.C.: Who should fix this bug?, *ICSE '06*, pp. 361–370 (2006).
2) 柏祐太郎，大平雅雄，阿萬裕久，亀井靖高：大規模 OSS 開発における不具合修正時間の短縮化を目的としたバグトリアージ手法，情報処理学会論文誌 (2015). (to appear).
3) Kashiwa, Y., Yoshiyuki, H., Kukita, Y. and Ohira, M.: A Pilot Study of Diversity in High Impact Bugs, *Proceedings of ICSME'14*, pp. 536–540 (2014).
4) Shihab, E., Mockus, A., Kamei, Y., Adams, B. and Hassan, A.E.: High-impact Defects: A Study of Breakage and Surprise Defects, *Proceedings of ESEC/FSE '11*, pp.300–310 (2011).
5) Chen, T.-H., Nagappan, M., Shihab, E. and Hassan, A. E.: An Empirical Study of Dormant Bugs, *Proceedings of MSR '14*, pp.82–91 (2014).
6) Valdivia Garcia, H. and Shihab, E.: Characterizing and Predicting Blocking Bugs in Open Source Projects, *Proceedings of MSR '14*, pp. 72–81 (2014).
7) Gegick, M., Rotella, P. and Xie, T.: Identifying security bug reports via text mining: An industrial case study, *Proceedings of MSR '10*, pp.11–20 (2010).
8) Molyneaux, I.: *The Art of Application Performance Testing : Help for Programmers and Quality Assurance*, O'Reilly Medea (2009).