重大不具合に対するOSS開発者の認識調査

松野 祐介
和歌山大学 システム工学部
s171050@sys.wakayama-u.ac.jp

山谷 陽亮
和歌山大学 システム工学研究科
s151049@sys.wakayama-u.ac.jp

大平 雅雄
和歌山大学 システム工学部
masao@sys.wakayama-u.ac.jp

要旨

本研究では，重大不具合 *(High Impact Bug)* に対する *OSS* 開発者の認識を理解することを目的として，*GitHub* に登録されている *OSS* 開発者に対してアンケート調査を行う．大規模・複雑化する近年のソフトウェア開発において，日々報告される多くの不具合に対処することは困難である．そのため，不具合修正プロセスの改善を目的とした研究が盛んに行われているが，個々の不具合の種類については十分に考慮されていない．このような背景から，特に近年，*High Impact Bug (HIB)* が注目されている．*HIB* とはユーザや開発者に重大な影響を与える不具合のことである．しかしながら，*HIB* は主に研究者が着目して研究が進められているものであり，実際には開発者がどのように認識しているのかは十分に明らかではない．そこで本研究では，*GitHub* に登録されている *OSS* 開発者に対して， *HIB* に関するアンケート調査を実施し，その結果を分析した．分析の結果，以下の知見が得た．*(1) OSS* 開発者は，*HIB* の内 *Security* バグを最も重要視している．*(2) OSS* 開発者は，開発スケジュールなど開発に影響を与える不具合よりも，ユーザに影響を与える不具合を *HIB* として認識している．*(3) OSS* 開発者は，ソースコードの複雑さ，不十分なテスト，および，外部環境の変化を *HIB* の主な発生原因と認識している．*(4) OSS* 開発者は，修正パッチのテストや不具合の再現に時間をかけることで *HIB* を修正している．*(5)* 開発者の役割には違いはないが，開発するプロダクトのドメインによって重視する *HIB* は異なる．

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，多くの不具合が報告される．プロジェクトの管理者は，報告された不具合に対して，修正の優先順位の決定，修正コストの見積もり，修正担当者の決定，といった修正プロセスにおける意思決定を行う必要がある．しかし，人員やスケジュールに限りがある中で大量に報告される不具合に迅速かつ適切に対処することは困難である．そのため，不具合修正プロセスの改善を目的として，

- 重複不具合報告の検出 [10, 16]
- 不具合の修正時間予測 [21, 22]
- 不具合箇所の特定 [20, 17]
- 不具合修正タスクの割り当て [1, 23]

などの研究が行われている．

しかし，これらの研究では個々の不具合の種類については十分に考慮されていない．そのため，タイプミスなどの軽微な不具合からセキュリティに関する優先度の高い不具合まで同等に扱われていることが問題となっている．例えば，不具合修正タスクの割り当てを行う際に，個々の不具合に適性のある開発者に割り当てを行う．しかし，重要な不具合である場合に誰が修正を担当すべきであるか考慮されていない．このことから，個々の不具合が与える影響を考慮した分類・分析が必要とされ，近年 High Impact Bug（以降では HIB と呼ぶ）と呼ばれるユーザや開発者に大きな影響を与える不具合に関する研究が行われている [6, 9].

近年の研究では，6 種類の HIB (Surprise バグ，Dormant バグ，Blocking バグ，Security バグ，Performance バグ，Breakage バグ）が注目されている．しかし，先行研究において挙げられている 6 種類の HIB は，いずれも研究者の知識や経験に基づいて定義された不具合であり，開発者が実際に重要な不具合だと認識しているかは明らかではない．そのため，HIB の分析および予測手法の提案を開発者が求めていない可能性がある．

本研究では，オープンソースソフトウェア (OSS) 開発者を対象にアンケート調査を実施し，HIB に対する OSS 開発者の認識を理解することを目的とする．本研究では特に，GitHub に登録されている OSS 開発者を対象としてアンケート調査を実施し，得られた 321 件の回答に対して分析を行う．

以降ではまず，2 章において本研究で対象とする 6 種類の HIB に関する先行研究について述べ，本研究の立場を明らかにする．次に，3 章では，実施するアンケート調査の方法について説明する．4 章では，HIB に対する OSS 開発者の認識理解を目的とするアンケート調査において用いる調査項目の意図について説明する．5 章で分析結果を示し，その結果に踏まえ 6 章で今後必要となる研究の方向性について議論する．最後に 7 章において本論文のまとめと今後の課題を述べる．

2. High Impact Bug

本章では，先行研究で挙げられている 6 種類の HIB の特徴について述べる．その後，本研究のモチベーションを述べる．

2.1. HIB の特徴と先行研究

Surprise バグ [12]

Surprise バグとは，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．Surprise バグが発生すると，スケジュールの変更が必要になる等，主に開発者に影響を与える不具合として考えられている．Shihab ら [12] は，Surprise バグを予測するモデルを構築する際には，同時変更ファイル数やリリース前の最終変更からの日数等が精度向上に寄与することを示している．

Dormant バグ [2]

Dormant バグとは，あるリリースに向けて開発を行っている際に埋め込まれ，テスト中またはリリース後には見つからずに，次のリリースの後に見つかる不具合を指す．リリース後に報告された不具合は，そのリリースの品質の指標として用いられているが，Dormant バグは，本来不具合が埋め込まれたリリースとは別のリリースの不具合として判断されるため，各リリースの品質評価を誤る可能性がある．そのため，Dormant バグは，他の不具合と比べて影響力が高いため優先して修正が行われる傾向があり，Dormant バグ以外の不具合よりも修正規模は大きいが修正時間は短くなることが Chen らの研究 [2] において報告されている．

Blocking バグ [15]

Blocking バグとは，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 バグが解消されない限り，修正を阻害されている側の不具合が例え軽微なものであっても修正できないため，Blocking バグの存在はプロジェクト全体の不具合修正の長期化を招くとされている．実際，Garcia らの研究 [15] において，Blocking バグは，Blocking バグ以外の不具合に比べて修正時間が長くなることが報告されている．

Security バグ [3]

Security バグとは，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 バグの代表的な例として，HeartBleed と呼ばれるものが存在する．ソフトウェアに Security バグが含まれていると，攻撃者によってユーザに悪影響が与えられる危険性があるため，影響力の高い不具合として考えられている．Zaman ら [18] は，Security バグは Security バグ以外の不具合よりも修正時間が短いことを報告している．

Performance バグ [8]

Performance バグとは，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，プロダクトのパフォーマンス低下を招く不具合を指す．ソフトウェアのパフォーマンスはユーザの満足度に影響を与える非機能要求として重要視され，プロジェクトの成功に大きく関わると考えられている．明確な障害が発生する不具合から，ユーザに

よって認識が分かれる軽微なパフォーマンス低下まで，様々な不具合が Performance バグとして扱われる．不具合報告のテキスト情報に含まれるキーワード（perf, slow, hang, throughput など）を用いて Performance バグを分類する研究 [19, 8] などがある．

Breakage バグ [12]
Breakage バグとは，不具合の修正や新しい機能の追加によって以前使えていた機能を使えない状態にする不具合を指す．Breakage バグが発生すると，既存の機能が失われることで，ユーザが既存の機能で行っていた日常業務に支障をきたすため，主にユーザに影響を与える不具合として考えられている．Shihab ら [12] は，Breakage バグの予測にはファイル変更回数やリリース前のバグ数が重要であると報告している．

2.2. 本研究のモチベーション

先行研究で取り上げられている HIB はすべて，研究者の知識や経験に基づいて問題視されている不具合であり，必ずしも実務者の認識と一致しているとは限らない．研究者が提案する HIB 検出のための予測モデルや分析手法が実際に役立つかどうかが不明なままであり，実務者のニーズを正しく理解しないままさらに研究を行うことは好ましくない．

そこで本研究では，実務者の HIB に対する認識をインタビュー調査することにより，実務者の観点から今後の研究の方向性や方針についての知見を得ることを目指す．ただし，先行研究のほとんどは，OSS プロジェクトから収集した不具合データを用いて予測モデルの検証結果や分析結果を示している．本研究においても，結果の一般性を高めることを目的として様々な OSS プロジェクトの開発者を対象として HIB に対する認識を調査する．企業実務者のニーズとは依然として乖離が生じる可能性もあるが，本研究で得られる知見を今後の議論の叩き台として活用したいと考えている．

3. 調査方法

本章では，実施するアンケート調査の対象と手順について説明する．

3.1. 調査対象

本研究では，OSS における開発者の HIB に対する認識を調査するために，GitHub[1] に登録されている開発者を調査対象とする．GitHub とはソフトウェア開発プロジェクトのためのホスティングサービスであり，開発者は Git によって管理されるリポジトリを GitHub 上で公開することにより，世界中の開発者とソースコードなどを共有することができる．ユーザ数 1,000 万，リポジトリ数 3,500 万を超える最も大規模なホスティングサービスであるため，多様な種類の OSS の開発に携わる開発者から HIB に対する認識を調査できると考えた．

3.2. 調査手順

3.2.1. 開発者の選定

本研究では，GitHub に登録されている開発者の中から活動的な開発者に対してアンケート調査を実施する．活動量の指標として GitHub API[2] を用いて取得できる contribution（リポジトリに対してコミットした回数）を用いるが，GitHub ユーザ個々人の contribution を直接カウントする方法はないため，以下の手順により contribution の多い開発者を選定した．

1. リポジトリ情報の取得
   GitHub API を用いてリポジトリ一覧を取得する．リポジトリは 3,500 万件以上存在するが，今回の調査では 3,500 万件までを用いる．リポジトリ一覧に基づいて再び GitHub API を用いて各リポジトリから contribution の多い順に 30 名のユーザ名（開発者のアカウント名）を取得する．

2. リポジトリの選定
   各リポジトリの contributor を参照し，全員（最大 30 名）の contribution を合計した値を各リポジトリの contribution とする．contribution が多い順にソートしリポジトリのランキングを作成する [3]．ランク付けしたリポジトリの内，contribution が 1,000

[1] GitHub:https://github.com/
[2] https://api.github.com/
[3] フォークにより作成されたリポジトリにはフォーク元リポジトリの contributor のデータもコピーするため，重複してデータ取得してしまう問題が発生する．この問題を回避するために，オーナーが異なる複数の同名リポジトリが存在する場合には，contoribution の一番多いリポジトリのみを残し，それ以外をランキングから除外した．

回以上のリポジトリ 1,211,913 件を活発なリポジトリとして選定した．

3. 開発者情報の取得
   GitHub API を用いて，手順 2 で選定したリポジトリに登録されている開発者のメールアドレスを取得する．ここで，開発者のアカウント名と contribution およびメールアドレスを紐付けておく．

4. 開発者の選定
   手順 3 の処理を行うことで，異なるリポジトリで活動している同じ開発者のデータが複数得られる．同じ開発者のデータが複数あれば contribution を合計する．contribution の多い順にソートし開発者のランキングを作成する．メールアドレスが無記入だった開発者を除いた 54,282 人の内，contribution が 100 回以上の開発者 22,228 人をアンケート対象として選定した．

3.3 アンケートの実施

前述の手順によって選定した 22,228 人の開発者の内，メールサーバの制限により，18,299 人にメールでアンケート調査への参加を依頼した．その結果，321 人からの回答が得た．アンケート調査は 2015 年 7 月から 2015 年 8 月の間に行った．アンケートには，開発者のプロファイルに関する情報（開発経験，開発目的，プロジェクトでの役割，主要参加プロジェクトなど）の他，次章で説明する調査項目が含まれる．回答方法は，一部の調査項目を除いて，著者らが用意した選択肢から 1 つ選ぶ形式のものであり，適切な選択肢がない場合には「その他 (other)」を選択した上で自由形式でその内容を記述できるようにした．

4. 調査内容

本章では，開発者の HIB に対する認識を理解するために行う調査の内容について説明する．

4.1. High Impact Bug の重要性

Q1：6 種類の HIB の中で最も重要なものはどれか？
調査の意図：近年研究者が注目している 6 種類の HIB の内どの HIB が，OSS の開発現場で重要な不具合として認識されているかは明らかではない．Q1 は，OSS 開発者が最も重要視する HIB を特定するための調査項目である．

Q2：6 種類の HIB 以外の HIB は存在するか？
調査の意図：本研究が対象とする 6 種類の HIB は，あくまでも研究者が研究対象として近年取り上げているものであり，OSS の開発現場では 6 種類の HIB 以外にも HIB として認識されている不具合が存在する可能性がある．Q2 は，6 種類の HIB 以外に重要視する HIB を特定するための調査項目である．

4.2. High Impact Bug の影響範囲

Q3：HIB は誰に対して最も影響があるか？
調査の意図：先行研究が 6 種類の HIB に着目している理由は，それらがユーザあるいは開発者に対して大きな影響を与えるためである．例えば，Surprise バグは予期しないタイミングで予期しない箇所で発見されるため，開発スケジュールの再調整に至るケースがあり，開発者に大きな影響を与えるとされている．Breakage バグは以前使えていた機能を使えなくする不具合であるため，ユーザの業務や利用体験に大きな影響を与えるとされている．直接的，間接的にはいずれの HIB もユーザおよび開発者の双方に影響を与えるものではあるものの，Q3 は，OSS 開発者が各 HIB の影響範囲をどのように捉えているかをより正確に把握するための調査項目である．

4.3. High Impact Bug の発生原因・対処方法

Q4：HIB の発生原因はどのようなものがあるか？
調査の意図：先行研究では HIB の発生原因に対する分析が行われている．例えば，Dormant バグの発生原因は主に，境界値や制御フローであることが報告されている [2]．Q4 は，各 HIB の発生原因に対する OSS 開発者の認識を理解するための調査項目である．

Q5：HIB をどのように修正するか？
調査の意図：先行研究では HIB の修正方法に関する分析が行われている．例えば，Security バグは多くの開発者が修正活動に関与し，特に熟練の開発者に担当されやすいことが報告されている [18]．Q5 は，OSS 開発者が HIB をどのように修正しているのかを理解するための調査項目である．

4.4. 役割とドメインによる認識の違い

Q6：プロジェクト内の役割によって HIB に対する認識の違いがあるか？
調査の意図：OSS 開発では，開発者にはプロジェクト全体の管理，ソフトウェアのテスト，不具合報告，修正パッチの投稿・レビュー・反映といった様々な役割が存在し，プロジェクト内で担う役割によって HIB に対する認識に違いが存在する可能性がある．例えば，不具合修正を担当する開発者は，他の不具合修正を妨害する Blocking バグを最も重要視するかもしれない．Q6 は，OSS 開発者の役割の違いによる HIB に対する認識の違いを明らかにするための調査項目である．
Q7：開発するプロダクトのドメインによって HIB に対する認識の違いがあるか？
調査の意図：OSS 開発では，ビジネス向けアプリケーション（OpenOffice など）から開発者向けアプリケーション（Eclipse IDE など），ゲームアプリケーションまで様々なドメインのプロダクトが開発されている．OSS 開発者がどのドメインのプロダクトを開発しているかによって，HIB に対する認識に違いが存在する可能性がある．例えば，ブラウザなどの一般向けに広く普及しているプロダクトの開発者は，Performance バグを重要視するかもしれない．Q7 は，開発するプロダクトのドメインの違いによる HIB に対する認識の違いを明らかにするための調査項目である．

5. 調査結果

本章では，アンケート調査の結果を示す．

5.1. High Impact Bug の重要性

Q1：6 種類の HIB の中で最も重要なものはどれか？
回答結果を図 1 に示す．縦軸は回答の割合を，横軸は 6 種類の HIB およびその他 (other) を表す．6 種類の HIB の内，Security バグが 53.1%と最も高く，次点で Breakage バグが 22.4%だった．Surprise バグ，Dormant バグ，Blocking バグ，Performance バグに関しては回答の割合は 2〜6%であった．
Q2：6 種類の HIB 以外の HIB は存在するか？
Q1 においてその他 (other) を選択した開発者が 7.5%存在した．other を選択した場合には自由記述ができるようにしていたため，いくつか具体的な回答が得られた．

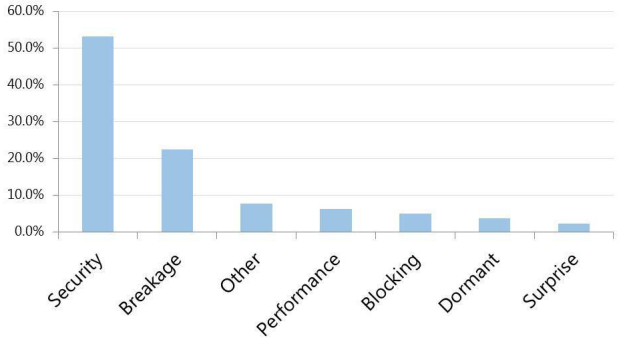

図 1. Q1 の調査結果（other はその他を選択した割合）

自由記述欄にコメントがあった回答の回答番号と回答内容を表 1 に示す．表 1 より，6 種類の HIB と類似する不具合も含まれていたが，ユーザデータの消失やユーザ体験の低下など，ユーザに対して大きな影響を与える不具合を HIB としている開発者が多く見受けられた．

5.2. High Impact Bug の影響範囲

Q3：HIB は誰に対して最も影響があるか？
回答結果を図 2 に示す．縦軸は回答の割合を表す．横軸は回答者全員 (all) と各 HIB を選択した開発者を表す．Blocking バグ以外の HIB に対しては，開発者の過半数が「主にユーザに影響を与える」不具合として認識していることが分かる．特に Performance バグについては，95%の開発者がユーザに影響を与える不具合と捉えている．一方，Blocking バグについては，50.0%の開発者が「主に開発者に影響を与える」不具合と考えており，先行研究が示した Blocking バグの問題意識については OSS 開発者と認識を共有しているものと言える．Surprise バグおよび Dormant バグについては，先行研究の知見とは異なり，ユーザ視点からその影響を考えている開発者が大半を占めることが分かった．

5.3. High Impact Bug の発生原因・対処方法

Q4：HIB の発生原因はどのようなものがあるか？
回答結果を図 3 に示す．縦軸，横軸の意味は Q3 と同様である．回答全体としては，「コードの複雑さ」が 32.7%，「不十分なテスト」が 30.5%，「外部環境の変化」が 18.9%と

表 1. Q1 における other の自由既述（Q2 の調査結果）

回答番号	回答内容
#13	A bug resulting in false results
#28	threatening users' data security
#32	a bug impacting user experience
#38	bug affecting many users
#72	a mix between security/performance/core bugs
#114	lost user data
#122	a bug reducing user satisfaction
#128	bug having most user impact (as this is judged critical by management)
#157	bug corrupting the database silently
#164	bug in core functionality that affects only particular platform and cannot be easily worked around
#169	a bug threatening systems features
#187	calculation error
#195	bug blocking main functions
#240	causing data loss for users
#276	a bug preventing usage of the project, e.g. crashes
#309	Concurrency bugs

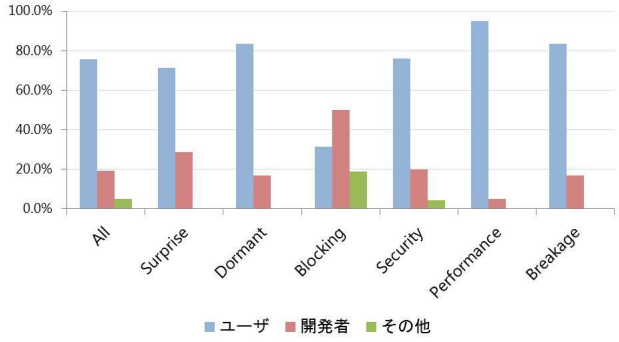

図 2. Q3 の調査結果

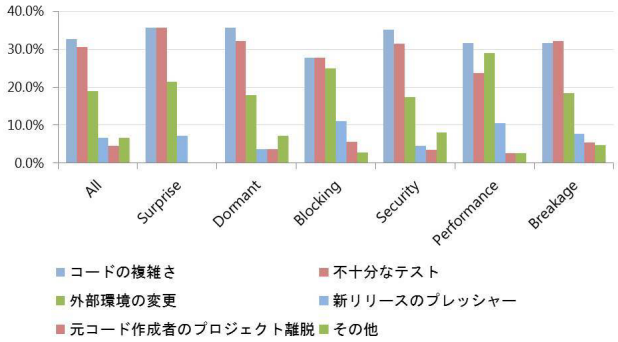

図 3. Q4 の調査結果

いう結果となり，HIB の主な発生原因であることが分かった．HIB 毎の結果も同様の傾向があるが，特に Performance バグが最も重要とした開発者の 28.9%は「外部環境の変化」を主要な発生原因と回答しており，他の HIB よりも高い割合となっていた．OS のアップデートや計算機環境の変更により，OSS においてもパフォーマンスに関する問題が発生しやすいことが伺える．

Q5：HIB をどのように修正するか？

回答結果を図 4 に示す．縦軸，横軸の意味は Q3 と同様である．回答全体としては,「修正パッチのテストを慎重に行う」が 32.5%,「不具合の再現に時間をかける」が 24.5%,「熟練の開発者に修正を割り当てる」が 20.7%,「十分に議論を行う」が 16.8%,「その他」が 5.5%の順となった．HIB 毎の結果も概ね同様の傾向があるが，特に Surprise バグおよび Performance バグが最も重要とした開発者は,「不具合の再現に時間をかける」を最も多く選択している（Surprise: 37.5%, Performance: 35.0%）．Surprise バグは予期しない箇所やタイミングで発見されることや，Performance バグは絶対的な判断基準が存在しないことから，不具合を再現し，開発者同士で問題を共有することが重要になるため，他の HIB よりも不具合の再現を重要視していると推察される．

5.4. 役割とドメインによる認識の違い

Q6：プロジェクト内の役割によって HIB に対する認識の違いがあるか？

回答結果を図 5 に示す．縦軸は回答の割合を，横軸は回答者全員（全体）と開発者の役割を表す．全体の結果を含め，全ての役割において Security バグが最も重要視されていることが分かる．次点では，Breakage バグを選

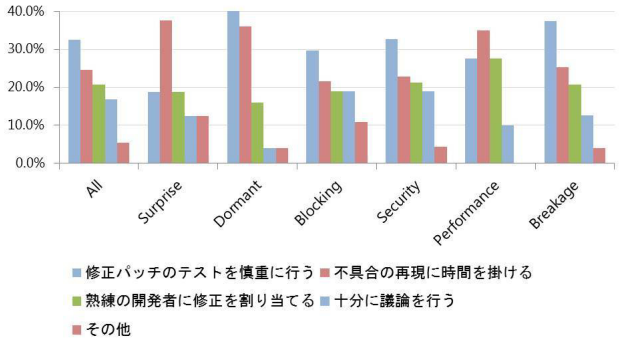

図 4. Q5 の調査結果

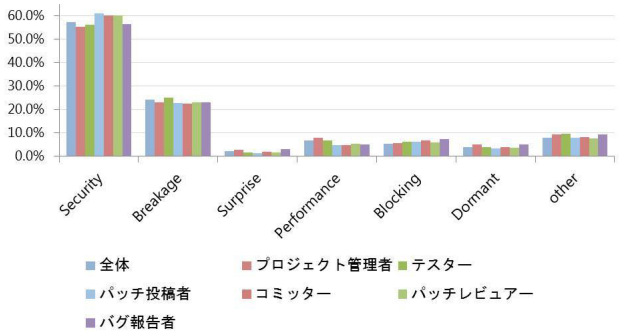

図 5. Q6 の調査結果

択する開発者の割合が高い結果となっている．このことから，OSS 開発者の役割の違いによって重要視する HIB に大きな違いは見られず，開発者間で一致した見解が存在することが分かった．

Q7：開発するプロダクトのドメインによって HIB に対する認識の違いがあるか？

Q7 は開発するプロダクトの種類（ドメイン）によって HIB に対する認識の違いが存在するかを調査するためのものであるが，GitHub には階層型のドメイン分類が存在しないため，本研究では，SourceForge[4] の分類を参考に，本調査で回答した開発者が開発している OSS を目視で確認し分類した．参考にしたカテゴリは，Audio & Video，Business & Enterprise，Communications，Development，Home & Education，Games，Graphics，Science & Engineering，Security & Utilities，System Administration の 10 種である．図 6 に分類結果を示す．縦軸は回答の割合を，横軸は回答の割合を，縦軸は全

[4]Source Forge:http://sourceforge.net/

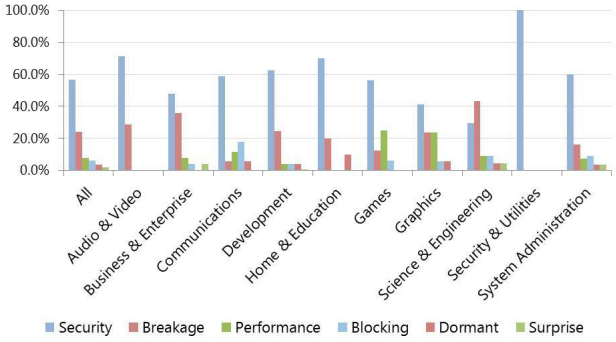

図 6. Q7 の調査結果

カテゴリ (all) と各カテゴリを表す．全体の結果としては，Security バグが 56.7%，Breakage バグが 24.1%と，他の HIB よりも突出して割合が高いことが見て取れる．ドメイン別に見ても概ね全体の結果と一致する傾向にあるが，Communications に関しては Security バグについで重要視されるのが Blocking バグであり，Science & Engineering では最も重要視されるのが Breakage バグとなっている．また，Games や Graphics においては，Security バグに続いて Performance バグが次点として選ばれている．なお，Security & Utilities では，(当然ではあるかもしれないが) Security バグが 100.0%となっている．これらの結果から，開発しているプロダクトのドメインによって重要視する HIB に特徴があることが分かった．

6. 考察

本章では，本研究で行ったアンケート調査の結果を基に考察を行う．

6.1. High Impact Bug の重要性

Q1 の結果，半数以上の OSS 開発者が Security バグを HIB だと認識していることが分かった．これは Security バグが攻撃者に悪用されることによってユーザに損害を与える可能性があるため，多くの開発者にとって優先して修正すべきだと認識されていると考えられる．また，次点で 22.4%の OSS 開発者が Breakage バグを HIB だと認識していることが分かった．これも同様に既存の機能が失われることによってユーザの日常業務が滞り，損害を与える可能性があるためと考えられる．

Q2 の結果からも同様の認識が示唆される．表 1 が示すとおり，#28 の OSS 開発者はユーザデータのセキュリティを脅かす不具合を HIB だと認識している．その他にも，#72 はセキュリティに関する問題を含む複合的な不具合，#114 および#240 はユーザデータの損失を引き起こす不具合が HIB として挙げている．

このことから，Security バグや Breakage バグを含む，ユーザに損害を与える可能性のある不具合に着目した研究が今後も重要であることが分かる．例えば，報告される不具合が Security バグ [3] であるかの予測や，Breakage バグの特定 [12] といった手法をさらに洗練させることが OSS 開発者にとっては特に有益であると思われる．

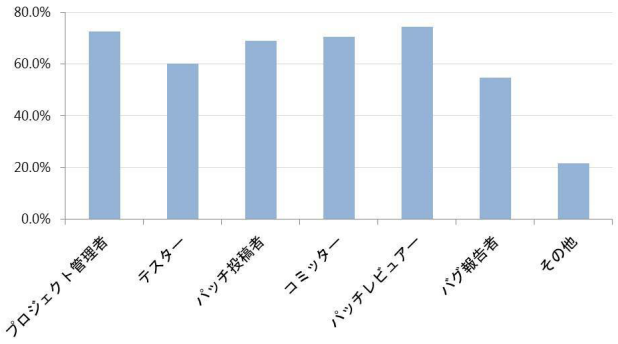

図 7. アンケートに回答した OSS 開発者の役割（複数選択あり）

6.2. High Impact Bug の影響範囲

Q3 の結果，Blocking バグを除いて，開発者の大半が HIB はユーザに大きな影響を与えるものであると認識していることが分かった．特に Performance バグを HIB であると回答した開発者の 95.0%がユーザに影響を与える不具合と考えており，ユーザ視点で HIB を重視する開発者には，前述の Security バグおよび Breakage バグに関する研究に加え，Performance バグに関する研究が求められていると言える．[19, 8] では，Performance バグの特徴や修正方法などについて分析されているのみであり，今後は Performance バグの再現や原因特定を支援する研究が必要である．

6.3. High Impact Bug の発生原因・対処方法

Q4 の結果，HIB は「ソースコードの複雑さ」,「不十分なテスト」,「外部環境の変化」が主な発生原因と捉えられており，どの HIB においても概ね同様の傾向があることが分かった．Eclipse プロジェクトなどの大規模 OSS 開発プロジェクト以外の OSS プロジェクトでは，厳格なリリース計画やマイルストーンを設定していることは少ない．また，特に GitHub では，プルリクエストベースの開発が行われているため，コアメンバー以外の開発者（貢献者）からのコードやパッチ提供を受けつつ，実装と不具合修正が同時進行する形で開発が行われる．小規模あるいは中規模プロジェクトでは人的資源にも限界があるため，リファクタリングやテストを十分行わずに（行えずに）リリースする状況が多く発生するのではないかと考えられる．近年では，HeartBleed 問題で注目されたように，少人数で開発されている OSS が社会的に広く普及していることも少なくない．技術的な支援や研究も必要であるが，人的な支援も今後はより重要になると考える．

Q5 の結果，特に Security バグおよび Performance バグの修正方法として,「不具合の再現に時間をかける」と回答した開発者の割合が高いことが分かった．これらの不具合は，不具合の再現が困難なだけではなく修正も困難を伴う場合が多く，熟練の開発者に不具合の修正を割り当てることが多いと指摘されている [18]．不具合の再現方法や特定方法に加え，バグトリアージ技術 [1, 23] の向上により，効率的に適任の開発者に不具合修正タスクを依頼できるようにすることが重要であると思われる．

6.4. 役割とドメインによる認識の違い

Q6 の結果，OSS 開発者の役割によって HIB に対する認識に大きな差は見られなかった．これは本研究でアンケートに回答した OSS 開発者の役割に明確な差がなかったことが一因と考えられる．図 7 に，本調査に参加した OSS 開発者の役割の分布を示す．プロファイルの調査では，複数選択を可能としたため多くの開発者が複数の役割を選択していた．このことが，役割間の認識の差を小さくした原因と思われる．今後は，各役割毎に HIB の認識を調査できるようにする（「テスターとして重要視する HIB はどれか？」などの質問形式にする）など工夫が必要である．

Q7 の結果，開発するプロダクトのドメインによって HIB に対する認識に大きな差が見られた．ドメインの認識を考慮することで，プロジェクトに則した効果的な支

援手法の構築が期待できる．例えば，バグトリアージ技術やバグローカリゼーション技術 [11, 17] を構築する際には，プロジェクトが重要視する HIB に重みを置いた手法を用いることで，より効果的な支援が行える可能性がある．

6.5. 制約

本研究では，GitHub に登録されている OSS 開発者を対象にアンケート調査を実施した．そのため，GitHub で活動していない OSS 開発者や企業の開発者は調査対象から外れている．また，GitHub に登録されている開発者の中で活動的な開発者に対してアンケート調査を実施するために，活動量の指標として GitHub API で取得できる contribution を用いた．この contribution にはコミット数のみが反映され，それ以外の issue 開設，コメント投稿といった活動が反映されていない．そのため，コミット以外の活動で多く貢献している OSS 開発者が調査対象から外れている．また，活動的な開発者として選定した中で，メールアドレスを GitHub 上で公開していない OSS 開発者は調査対象から外れている．その他，メールサーバの制限により，対象者の全員にメール送信ができなかった．一般性の高い結果を得るためには，本研究で対象にできなかった開発者に対しても調査を行う必要がある．

なお，本研究ではアンケート調査の回答を多く得るために質問数が多くならないようにしたが，開発者の認識をより良く理解するためには，今後さらに多くの質問を実施する必要がある．また，回答のほとんどは複数の選択肢から 1 つだけを選択する単一回答方式を用いた．そのため，OSS 開発者が重要視する HIB に対して回答の偏りが生じている可能性は否定できない．今後は，回答形式に順位回答を用いることで懸念を解消できる可能性がある．

7. まとめと今後の課題

本研究では，OSS 開発者の HIB に対する認識を理解するために，GitHub に登録されている OSS 開発者を対象にアンケート調査を実施した．得られた 321 件の回答を分析した結果，主に以下の知見が得られた．

- OSS 開発者は，HIB の内 Security バグを最も重要視している．
- OSS 開発者は，開発スケジュールなど開発に影響を与える不具合よりも，ユーザに影響を与える不具合を HIB として認識している．
- OSS 開発者は，ソースコードの複雑さ，不十分なテスト，および，外部環境の変化を HIB の主な発生原因と認識している．
- OSS 開発者は，修正パッチのテストや不具合の再現に時間をかけることで HIB を修正している．
- 開発者の役割には違いはないが，開発するプロダクトのドメインによって重視する HIB は異なる．

また，アンケート調査の結果から，不具合の特定や再現，割り当て方法に関して HIB を考慮した研究が今後必要となることが分かった．今後は，アンケート対象を拡充するとともに調査項目を充実させて調査を継続したい．また，企業実務者との意見交換などを踏まえて，不具合修正プロセスの改善技術の方向性についても有益な知見を提供したいと考えている．

謝辞

本研究の一部は，文部科学省科学研究補助金（基盤 (C): 15K00101）による助成を受けた．

参考文献

[1] John Anvik, Lyndon Hiew, and Gail. C. Murphy. Who should fix this bug? In *Proceedings of the 28th International Conference on Software Engineering (ICSE '06)*, pp. 361–370, 2006.

[2] 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)*, pp. 82–91, 2014.

[3] Michael Gegick, Pete Rotella, and Tao Xie. Identifying security bug reports via text mining: An industrial case study. In *Proceedings of the 7th Working Conference on Mining Software Repositories (MSR '10)*, pp. 11–20, 2010.

[4] Emanuel Giger, Martin Pinzger, and Harald Gall. Predicting the fix time of bugs. In *Proceedings of the 2nd International Workshop on Recommendation Systems for Software Engineering (RSSE '10)*, pp. 52–56, 2010.

[5] Gaeul Jeong, Sunghun Kim, and Thomas Zimmermann. Improving bug triage with bug tossing graphs.

In *Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering (ESEC/FSE '09)*, pp. 111–120, 2009.

[6] 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)*, pp. 536–540, 1 2014.

[7] Lionel Marks, Ying Zou, and Ahmed E. Hassan. Studying the fix-time for bugs in large open source projects. In *Proceedings of the 7th International Conference on Predictive Models in Software Engineering (PROMISE '11)*, pp. 11:1–11:8, 2011.

[8] 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)*, pp. 237–246, 2013.

[9] Masao Ohira, Yutaro Kashiwa, Yosuke Yamatani, Hayato Yoshiyuki, Yoshiya Maeda, Nachai Limsettho, Keisuke Fujino, Hideaki Hata, Akinori Ihara, and Kenichi Matsumoto. A dataset of high impact bugs: Manually-classified issue reports. In *Proceedings of the 12th Working Conference on Mining Software Repositories (MSR '15)*, pp. 518–521, 2015.

[10] Per Runeson, Magnus Alexandersson, and Oskar Nyholm. Detection of duplicate defect reports using natural language processing. In *Proceedings of the 29th International Conference on Software Engineering (ICSE '07)*, pp. 499–510, 2007.

[11] Ripon K. Saha, Matthew Lease, Sarfraz Khurshid, and Dewayne E. Perry. Improving bug localization using structured information retrieval. In *Proceedings of the 28th IEEE/ACM International Conference on Automated Software Engineering (ASE'13)*, pp. 345–355, 2013.

[12] 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)*, pp. 300–310, 2011.

[13] Chengnian Sun, David Lo, Xiaoyin Wang, Jing Jiang, and Siau-Cheng Khoo. A discriminative model approach for accurate duplicate bug report retrieval. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering (ICSE '10)*, pp. 45–54, 2010.

[14] Ahmed Tamrawi, Tung Thanh Nguyen, Jafar M. Al-Kofahi, and Tien N. Nguyen. Fuzzy set and cache-based approach for bug triaging. In *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering (ESEC/FSE '11)*, pp. 365–375, 2011.

[15] 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)*, pp. 72–81, 2014.

[16] Xiaoyin Wang, Lu Zhang, Tao Xie, John Anvik, and Jiasu Sun. An approach to detecting duplicate bug reports using natural language and execution information. In *Proceedings of the 30th International Conference on Software Engineering (ICSE '08)*, pp. 461–470, 2008.

[17] Chu-Pan Wong, Yingfei Xiong, Hongyu Zhang, Dan Hao, Lu Zhang, and Hong Mei. Boosting bug-report-oriented fault localization with segmentation and stack-trace analysis. In *Proceedings of 30th International Conference on Software Maintenance and Evolution (ICSME2014)*, pp. 181–190, 2014.

[18] Shahed Zaman, Bram Adams, and Ahmed E. Hassan. Security versus performance bugs: A case study on firefox. In *Proceedings of the 8th Working Conference on Mining Software Repositories (MSR '11)*, pp. 93–102, 2011.

[19] Shahed Zaman, Bram Adams, and Ahmed E. Hassan. A qualitative study on performance bugs. In *Proceedings of the 9th Working Conference on Mining Software Repositories (MSR '12)*, pp. 199–208, 2012.

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

[21] 正木仁, 大平雅雄, 伊原彰紀, 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, Vol. 52, No. 2, pp. 933–944, 2 2013.

[22] 吉行勇人, 大平雅雄, 戸田航史. Oss 開発における管理者と修正者の社会的関係を考慮した不具合修正時間予測. コンピュータソフトウェア, Vol. 32, No. 2, pp. 128–134, 6 2015.

[23] 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, Vol. 56, No. 2, pp. 669–681, 2 2015.