Web アプリケーションに対する欠陥再現自動化手法†

高橋　黎*

Automated Defect Reproduction Methodology for Web Applications

Rei Takahashi

1　はじめに

今日の Web アプリケーション開発では，ソースコード管理ツールを用いた欠陥の報告や修正が行われているが，報告者によりバグレポートの記載内容が異なる.この際，再現手順がない場合や正しくない場合，開発者が欠陥を修正できない[1].

本研究では Selenium IDE[2]を用いた再現手順の記録，再生による欠陥再現の自動化と欠陥の修正前後の比較を容易にする 2 つの機能を組み合わせる事で，再現性がバグレポートの記述内容に依存しない再現自動化ツールを開発し，その評価を行った結果について述べる.

2　目的

本研究の目的は，バグレポートの記述内容に依存しない欠陥再現手法により欠陥の再現性を担保させることでソフトウェア品質の向上を支援することである.

Chapparo[3]は，欠陥再現に必要なバグレポートには 3 つの要素が含まれることが多いと述べている.（図 1）

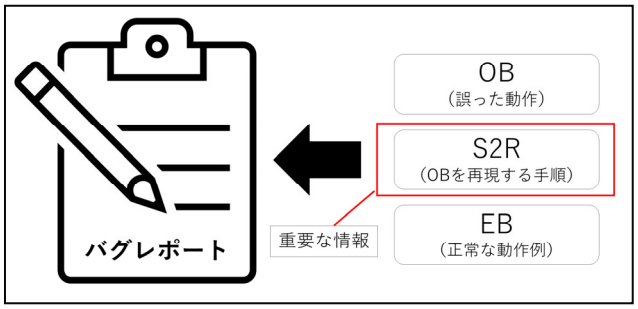

図 1 バグレポートを構成する 3 要素

「OB」はどのような欠陥が発生したかを説明するもので，「S2R」が欠陥を再現する再現手順の説明，「EB」はどのような動作が行われるべきであるかを説明する要素である.

Chapparo[3]は S2R の説明が欠陥再現に最も重要であると述べると共に，バグレポートの S2R と OB の明示的な記述が 51.4%以下に留まっていることを指摘している.例として S2R の情報が不足したバグレポートを図 2 に示す.

図 2 再現手順が画像のみのバグレポート

この例では，EC サイト上の購入確認ページで「クーポンボタン」と「お届け先変更ボタン」が下の要素と重なってしまっている欠陥である.

この欠陥再現に必要な再現手順は画像だけのため，欠陥自体の情報は得られるが，どのようにして画像の欠陥を引き起こすかが分からない.次に本来の再現手順を以下に示す.

1. 管理者ページからクーポン機能を有効化
2. ユーザー用トップページから任意の商品をカート追加
3. 購入ページへ進む
4. お届け先情報を入力して次に進む
5. 購入確認ページに欠陥が表示される

Web アプリケーションは複数のページに渡り処理を行うため再現に必要な操作が多くある．こうした再現手順が正確に開発者へ伝わらないために欠陥再現ができないことがある.

---

†本研究の一部は以下において発表した
・ソフトウェア工学の基礎 XXIX 第 29 回ソフトウェア工学の基礎ワークショップ（FOSE2022）

*電子情報メディア工学専攻　2218007　橋浦研究室

Copyright © 2023 Nippon Institute of Technology

3 提案手法

本研究では，バグレポートの記述内容によって欠陥再現ができない問題に対して，ブラウザ上において再現手順の記録と再生が可能な欠陥自動再現手法を提案する.また，自動再現手法を利用して修正差分の比較機能を容易にする機能を合わせた欠陥除去支援ツールを提案する.

本手法の提案機能を以下に示す.

1. 報告者による再現手順の記録と共有
2. 欠陥箇所の明示
3. 欠陥再現の自動化
4. 修正差分の確認

3.1 報告者による再現手順の記録と共有

再現に必要な再現手順は報告者が行った操作内容をレポートに記述した上で開発者に報告している.この時，書き方やニュアンス等によって正しく伝わらない可能性がある.

本研究では報告者が再現時に行ったブラウザ操作を記録し，再現手順ファイルとして共有することができ，開発者は共有されたファイルを再生することで再現が可能となる.

図 3 再現手順の記録と共有

3.2 欠陥箇所の明示

前章で Chaparro が OB の記述不足についても指摘するように，再現手順が明示されていても欠陥の箇所が分からなければ原因を特定できない.「画面の上部」などの抽象的な表現等により報告者と開発者で欠陥に対する認識の相違が無いように欠陥箇所を明示できるようにする.

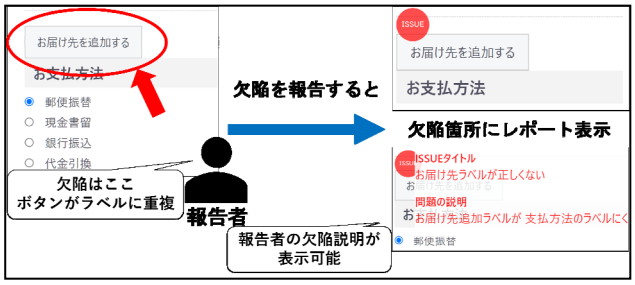

図 4 欠陥箇所へのレポート表示

3.3 欠陥再現の自動化

3.1 項で提案した機能の再現手順ファイルを用いて欠陥の自動再現を行う.開発者に共有された再現手順ファイルは任意のタイミングで再生可能で，報告者が行った再現操作をそのまま再生できることで再現性を担保している.

開発者がブラウザに再現手順ファイルを読込み再生を行うと，再現が自動で行われ欠陥ページまで遷移すると共に欠陥箇所が明示されることで欠陥原因を特定できる.

図 5 欠陥自動再現の流れ

3.4 修正差分の確認

3.3 項で提案した欠陥再現の自動化機能を修正前後のバージョンに対して同時に適用することで修正前後のバージョン間で正確な修正差分を確認できる.差分の確認は2つのバージョンを横並びにした専用のページ上で行う.

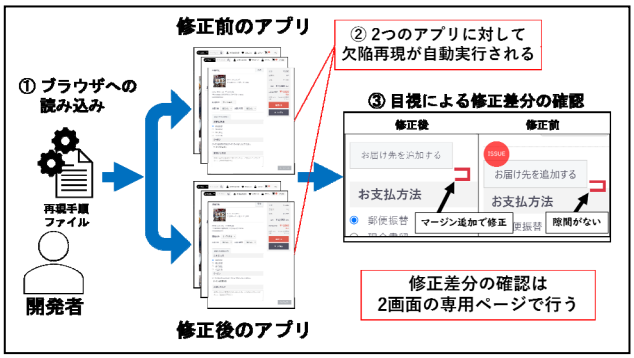

図 6 修正差分確認の流れ

4 実現手法

欠陥報告と欠陥再現は開発者・報告者共に煩雑な作業となる点を考慮し，Google Chrome[4]の拡張機能としてツールを実装し，GitHub API[5]と Selenium IDE[2]を連携させることでブラウザ上で欠陥報告や欠陥再現を自動的に行う修正支援が可能になった.

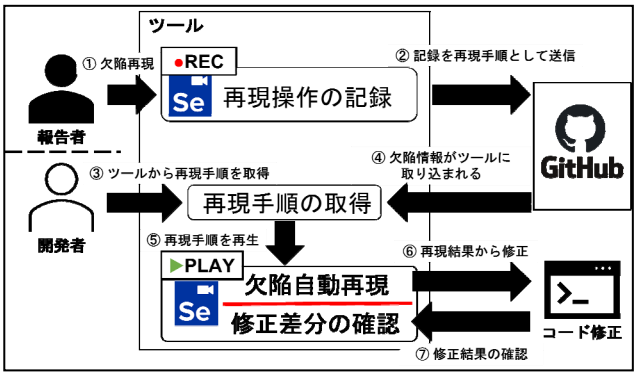

図 7 実装ツールによる欠陥修正の流れ

Copyright © 2023 Nippon Institute of Technology

このツールによる欠陥修正の流れは以下の通りである.

1. 報告者がブラウザ操作を記録して欠陥再現を行う(①)
2. 操作記録を再現手順として GitHub[6]へ送信（②）
3. 開発者がツール上で再現手順を取得することで報告済の欠陥が取り込まれ，再現作業が可能となる（③，④）
4. Selenium IDE[2]に再現手順を読み込み再現実行（⑤）
5. 再現結果から原因を特定し，コード修正を行う（⑥）
6. 修正後に修正差分の確認を行い修正結果確認（⑦）
7. 修正結果が正しくない場合，4 に戻る
8. 修正結果が正しいことが確認できれば修正終了

5 評価と考察

評価は OSS で EC サイトフレームワークを開発する EC-CUBE[7]へ報告された 1000 件の欠陥の中から，画面上で欠陥と分かるもの 251 件を対象とし本手法の有効性を確認した.

本研究の研究課題となるのは以下の 3 つである.

RQ1. 再現・差分機能はどのくらい対応したか

RQ2. 本手法で対応できない欠陥の要因は何か

RQ3. 開発において本手法で支援可能な欠陥は何か

5.1 RQ1:再現・差分機能はどのくらい対応したか

ツールの対応数を表 1 に示す．全体として，再現機能と修正差分機能ともに7割程度の欠陥に対応できた.機能ごとの対応数の差は 30 件（11.9%）で再現機能が修正差分機能よりも多くの欠陥に対応した.

表 1 ツール機能ごとの欠陥対応数

再現機能に対応	修正差分機能に対応	対応差
196 件（78.1%）	166 件（66.1%）	30 件（11.9%）

5.2 RQ2: 本手法で対応できない欠陥の要因は何か

前節で明らかになったとおり，本ツールは対象の欠陥すべてに対して対応することはできない.

したがって,本ツールを欠陥の修正作業に適用する際に,本手法による修正支援ができない欠陥がどのような要因で対応できないのかを確認することにした.

その結果,対応できない欠陥は提案手法の問題ではなく,ツール実装上の問題であることが分かった.

以下に要因を示す.

1. 欠陥の特性
2. Selenium IDE で記録できない
3. 自動再現機能への依存
4. 修正差分機能の実行環境問題

5.2.1 欠陥の特性

これらは再現手順がツールで支援可能な範囲を超えてしまったことが原因である.

具体例として,再現手順に認証コードを求められる場合,求められる認証コードが再現の度に変化するため，再現手順を記録しても，再利用することができないことがあげられる.

5.2.2 Selenium IDE で記録できない

これらはページのスクロール等の操作が Selenium IDE で記録できず，再現手順が生成できなかったことが原因である.

5.2.3 自動再現機能への依存

これらは修正差分の確認機能の実装方法が原因である.修正差分は欠陥自動再現機能を利用しており，5.2.1 項,5.2.2 項の要因による失敗が起こると,再現手順が生成できないため修正差分機能が利用できないことがあった.

5.2.4 修正差分機能の実行環境問題

これらは 2 つのアプリケーションの動作が干渉したことが原因である.Web アプリケーションのログイン機能はセッションや Cookie を利用して実現している.差分機能は同じ環境上で 2 つのアプリケーションに同じユーザ ID とパスワードを入力するため，セッションと Cookie の情報が干渉し，ログイン動作に失敗する等の支障が発生した.

5.3 RQ3: 開発において修正支援が可能な欠陥要因はなにか

Web アプリケーションはデータベースやロジックなど多くの技術要素で構成されており，本手法を用いて欠陥除去を行う場合，どの要素に混入する欠陥に対し有効かを確認する.そのため対象の欠陥を 7 つのカテゴリに分けた.

表 2 欠陥カテゴリごとの対応数

カテゴリ	対象欠陥数	再現対応数	修正差分数
データベース	4 (1.6%)	3	3
ロジック	146 (58.2%)	119	91
外観	100 (39.8%)	74	72
運用	0 (0%)	0	0
開発環境	0 (0%)	0	0
実行環境	1 (0.4%)	0	0

表 2 はカテゴリごとの欠陥数と再現機能，修正差分機能の対応数をカテゴリに分けた結果であるが，本手法で対象とした欠陥の 98%がロジックと外観であった.ロジックは,クーポンを使用しても合計金額に反映されない等のアプリケーション制御に関する欠陥で，外観はページの表示崩れ等の見た目上の欠陥としている.

この 2 要素について表 3 でツールの対応割合を確認する.

Copyright © 2023 Nippon Institute of Technology

表 3 2要素におけるツールの対応割合

カテゴリ	再現対応割合	修正差分対応割合
ロジック	81.5%	62.3%
外観	74.0%	72.0%

どちらの要素も再現機能が 7 割以上，修正差分機能が 6 割以上の欠陥に対応した.

ロジック要素では，再現と修正差分で 2 割程度の対応差があったが，再現手順にログイン操作が含まれていたため 5.2.4 項で述べた要因による影響を受けて対応数が減少している.外観要素は再現と修正差分の対応差が僅差であったことから修正の開始から終了まで本手法による欠陥除去支援が可能であると考えられる.

5.4 開発における本手法の有用性

本手法による欠陥再現の自動化は対象の 8 割程度の欠陥に，欠陥除去支援は 6 割程度の欠陥に対応できることが確認された.また本ツールで対象となる欠陥は，ロジックと外観が原因となるものが多いことが分かった.

ロジックや外観の欠陥では，欠陥を再現するために入力を行うことや操作を行わないと欠陥ページまで遷移できないことがあるため，欠陥の除去には正確な再現手順が必要であると考える.そのためロジックや外観の欠陥に対して本手法を適用することで Web アプリケーション開発におけるソフトウェア品質の維持や向上に寄与できる可能性があると考えられる.

また，対応できない欠陥の要因がツールの実装方法であったことから，実装方法を改良することで更なるソフトウェア品質の維持・向上に寄与できると考えられる.

6 関連研究

Chapparo ら[1]は，再現手順等の欠落を自動検出する手法を提案すると共に，得られた情報を報告者や開発者にフィードバックすることでバグレポートの品質を向上させる試みを行っている.

Chappar らの手法は，談話パターンと構文解析を組み合わせたニューラルネットワークを活用し，バグレポートに書かれた再現手順がどれだけ正確か，レポートの記述は具体的かを評価し報告者フィードバックしている.この手法を用いることで報告者がより品質の高いバグレポートを作成できるようになり，欠陥の再現性を向上させている.

Moran[8]はバグレポートに書かれる非構造な自然言語が欠陥特定を困難にさせる点を問題点とし，自然言語を用いない再現手順の作成支援ツール Fusion を提案している.

Moran の手法ではバグレポートを作成する際，どのページでどういった操作を行ったかを記述するのではなく，選択リストから選ぶことで再現手順を記録する.行った操作を選択するだけで再現手順が作成できるため自然言語による曖昧な説明や抽象的な表現がなくなり欠陥の再現性が向上する.また欠陥報告が行われると Fusion が再現手順書を自動生成することで，開発者が迅速な修正作業を支援している.

本研究ではバグレポートの品質によって再現性が変化しない手法を提案している.そのため Chapparo らが行っている報告者のバグレポート品質を向上させる必要がなく再現性を担保したバグレポートの作成が可能である.また, Moran は操作内容をリストから選択させたうえで，機械的に再現手順を生成して利用するが，本手法では報告者が行った操作そのものを再現手順として利用できる点が異なる.

7 結論

本研究では欠陥報告者が行った操作を再現手順として再生可能な欠陥再現自動化手法の提案し，ツールとして実装した.実際の OSS アプリケーション開発で報告される欠陥に対して実験を行った結果，欠陥自動再現が対象の 8 割程度，修正差分機能が 6 割程度対応し，開発において発生するロジックと外観の欠陥に対して本手法が有用である可能性が示唆された.

参 考 文 献

1) Oscar Chaparro, Carlos Bernal-C´ardenas, Jing Lu, Kevin Moran, Andrian Marcus, Massimiliano Di Penta, Denys Poshyvanyk, and Vincent Ng,“ Assessing the quality of the steps to reproduce in bug reports, ”ESEC/FSE 2019, Pages 86-96, August 2019.
2) SeleniumHQ.org, “SeleniumHQ/selenium-ide”, https://github.com/SeleniumHQ/, (accessed 2023-01-18).
3) O. Chaparro, “Improving Bug Reporting, Duplicate Detection, and Localization, 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C), Pages 421-424, May 2017.
4) Google LLC, “Google Chrome”, https://www.google.com/intl/ja_jp/chrome/, (accessed 2023-01-18).
5) GitHub Inc., “GitHub の REST API”, https://docs.github.com/ja/rest?apiVersion=2022-11-28, (accessed 2023-01-18).
6) GitHub Inc., “GitHub”, https://github.com/, (accessed 2023-01-18).
7) EC-CUBE CO.,LTD, “【公式】EC-CUBE”, https://www.ec-cube.net/, (accessed 2023-01-18).
8) Kevin Moran, “Enhancing Android application bug reporting, n Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2015). Pages 1045-1047, August 2015.

指導教授	審査委員（主査）	准教授	橋浦 弘明
	審査委員（副査）	教授	新井 啓之
	審査委員（副査）	教授	松田 洋

Copyright © 2023 Nippon Institute of Technology