

変異に基づく Fuzzing による E2E テストの自動生成手法

中川 尊雄^{1,a)} 徳井 翔梧¹ 山本 晃治¹ 徳本 晋¹ 宗像 一樹¹

概要: Web アプリケーションのテストケース作成にかかるコストを軽減する手法の一つとして、クローラベースの自動テスト生成手法がある。我々は、実世界の複雑なシステムのテストで成果を挙げている変異に基づく Fuzzing の考え方を、クローラベースの手法に応用し、人手で作成したテスト操作手順の近縁を重点的に探索できる新たなテスト生成手法を提案する。プロトタイプ実装を用いた評価実験により、単純なクローラベースの手法より高いカバレッジを獲得できる可能性が示された。

1. はじめに

Web アプリケーション開発ではしばしば、Web 画面からバックエンドまでのモジュールを結合して実施する End-to-End のシステムテストが実施される。

設計者の知識と多くの工数を要する End-to-End (E2E) のテスト作成を自動化する手法の一つとして、クローラベースのテスト生成 (e.g., [1] [2]) がある。これは操作手順の生成、実行、結果による画面遷移モデルの更新を繰り返しながら、より多くの操作手順を生成していく探索的な手法である。

同じ目的で利用されるモデルベースのテスト生成手法 (e.g., [3]) と比べると、テスト結果の判定機能が限定的であり、人手でのアサーション付与作業を要するが、モデルの開発・保守は不要である。従ってプロジェクトの引継ぎや熟練開発者の異動など、テスト対象に対する知識が限られるシーンで有効といえる。

クローラベースの手法の別の弱点として、生成される膨大な数の操作手順の中に、テストの観点で有意義といえるものがわずかしかなかったことが挙げられる。操作手順の生成には実際の操作を伴うため実行コストは低くなく、結果的に現実的な実行時間では画面遷移や操作パターンを網羅できないことも多い。

他方、実世界の複雑なシステムの欠陥を自動的に発見できる技術として、変異に基づく Fuzzing が注目を集めている [4]。変異に基づく Fuzzing では、テスト入力 (シード) に対して小さな摂動 (変異) を加えて新たなシードを生成し、実行結果に基づいてシードの有望さを評価することで、テスト生成を制御する。

本研究では、人手で作成した有意義な操作手順 (シード) の近縁を重点的に探索することを狙い、シードと変異の考え方をクローラベースのテスト生成手法に導入した新たなテスト生成手法を提案する。また、提案手法の評価のため、生成した操作手順の有意義さの指標として命令カバレッジを採用し、初歩的なクローラベースの手法との比較実験を行う。

2. クローラベースのテストケース生成

Web アプリは一般に、各画面を状態、操作とその結果を状態遷移とみなす状態遷移グラフの形でモデル化できる。クローラベースの手法は、既知のモデルがなくとも「未試行な操作は「未探索状態」への遷移を引き起こす」とみなす不完全なモデルを持つ。そして、未試行な操作手順を生成・実行しながら、実際に到達した状態に基づいて状態遷移グラフを更新していく。

よく知られたクローラベースのテスト生成ツールの一つとして、Mesbah らによる CrawlJax [1] がある。Zeller らによる Fuzzing の解説書 [5] でも、状態遷移モデルを形式文法の形で保管し、文法を網羅するように操作手順を生成するテスト手法の記載がある。

この手法の課題として、現実的な時間で探索できる範囲に限界があることが挙げられる。たとえば特定の固定値の入力や、複数の画面にまたがる操作を事前条件とする画面遷移は極めて引き起こしづらい。

これに対し、Fard らは、人手で作成したテストケースを参考にクローラの状態遷移モデルを更新する Testilizer [6] を提案し、問題の解決を試みた。ただし、Testilizer ではアサーションを再利用するために探索範囲を限定しており、人手で記述されなかった操作や遷移はクローラによる探索範囲外となっている。

¹ 富士通株式会社 人工知能研究所

^{a)} nakagawa-takao@fujitsu.com

3. 提案手法: Mutation-based GUI Fuzzer

クローラベースの手法の課題を軽減するため、我々は、人手で作成したテスト操作手順を初期シードとする新たなテスト生成手法を提案する。提案手法では、初期シードを読み込み、変異、実行、評価を繰り返しながら、人手で作成したテストケースの近縁に存在する有意なテスト手順の生成を目指す。

(手順 1) 初期シードの読み込み: シードは Selenium ベースの DSL を用い、クローラ内部では構文木の形で保持する。人手で作成した初期シードとクローラの内部表現は、要素の特定方法などの観点で一致しない場合があるため、クローラの生成した操作候補の中から等価な操作を見つけることで変換を行う。

(手順 2) シード変異・実行: 評価値の高いシードを変異し、新たなシードを作る。本手法では、フォーム入力値の変異と、ある画面状態での操作列の変異を行う。各変異では、他シードの同一ノードとの部分木交換と、(既存手法通りの) 状態遷移モデルに基づく生成のどちらかを実施する。

変異によって生成したシードは全て実行され、結果に基づいて状態遷移モデルが更新される。

(手順 3) シード評価: 生成したシードの有望さを評価する。本手法では、新たな画面を発見した場合と、以前試した操作で異なる画面遷移が起きた場合に評価値を加算する。これにより、新たな画面や、操作条件で遷移先が変わる画面を経由するシードが高く評価される。

4. 評価実験

Zeller らによるクローラベース手法の実装 [5] をベースに提案手法のプロトタイプを作成し、評価実験を行った。比較対象として、提案手法の機能を省いた単純なクローラベースのテスト生成ツールを作成した。

著者らが初期シードを作成することによるバイアスを防ぐため、実験対象は自動化結合テストケースを持つ Spring 製のサンプルアプリケーション^{*1}を採用した。評価では、各手法を 5 回ずつ各 10800 秒 実行し、300 秒ごとに測定した命令カバレッジを比較する。ただし、比較対象は初期シードのカバレッジ分だけ不利となるため、初期シードを実行した際のカバレッジとの和を結果とした。

図 1 に実験の結果を示す。中央のラインが差なしを表し、数値が正であれば提案手法が高いカバレッジを達成したといえる。図からは、実行初期と終盤（およそ 9000 秒経過以降）において、提案手法がより高いカバレッジを獲得できることが読み取れる。

なお、カバー可能な命令数は全体で 2642 命令あり、達成した網羅率の最大値は提案手法で 79.1%、比較対象で

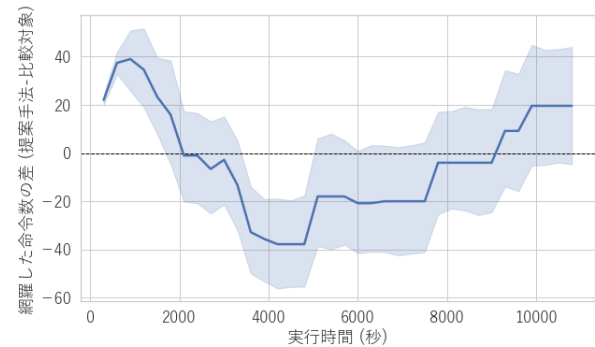


図 1 時間経過に対する獲得カバレッジの差 (背景の着色部は 95% 信頼区間を表示)

76.8%と、提案手法の方が最終的に高い網羅率を達成できた。提案手法でのみカバーできた部分は 6 メソッド 60 命令あり、逆のケースは存在しなかった。

また、中盤に見られる落ち込みの原因として、既存手法の達成カバレッジが 6000 秒程度で頭打ちになっており、提案手法では 10000 秒時点でも達成カバレッジの増加が見られることから、シード周辺での深さ優先的な探索が一時的にカバレッジの上昇を抑えたのではないかと考えられる。

5. まとめと今後の展望

本論文では、変異に基づく Fuzzing を応用した、Web アプリに対する自動テスト生成手法を提案・評価した。

今後の展望として、シード評価・変異戦略の多様化があげられる。たとえば、コードカバレッジによる評価を行う Coverage-Guided Fuzzing の応用は興味深い。

参考文献

- [1] Mesbah, A., Bozdog, E. and Van Deursen, A.: Crawling Ajax by inferring user interface state changes, *2008 Eighth International Conference on Web Engineering*, IEEE, pp. 122–134 (2008).
- [2] Thummalapenta, S., Lakshmi, K. V., Sinha, S., Sinha, N. and Chandra, S.: Guided test generation for web applications, *2013 35th International Conference on Software Engineering (ICSE)*, IEEE, pp. 162–171 (2013).
- [3] Andrews, A. A., Offutt, J. and Alexander, R. T.: Testing web applications by modeling with FSMs, *Software & Systems Modeling*, Vol. 4, No. 3, pp. 326–345 (2005).
- [4] Manès, V. J. M., Han, H., Han, C., Cha, S. K., Egele, M., Schwartz, E. J. and Woo, M.: The art, science, and engineering of fuzzing: A survey, *IEEE Transactions on Software Engineering* (2019).
- [5] Zeller, A., Gopinath, R., Böhme, M., Fraser, G. and Holler, C.: *The Fuzzing Book*, CISPA Helmholtz Center for Information Security (2021). Retrieved 2021-03-12 11:41:11+01:00.
- [6] Milani Fard, A., Mirzaaghaei, M. and Mesbah, A.: Leveraging existing tests in automated test generation for web applications, *Proceedings of the 29th ACM/IEEE international conference on Automated software engineering*, pp. 67–78 (2014).

^{*1} <https://github.com/selenide-examples/mybatis-spring-boot-jpetstore>