カバレッジ情報を活用してリグレッションテスト数を削減する最適テスト選択サービスSmallTests

SmallTests Service to Reduce Number of Regression Tests in Software Development Using Code Coverage Information

山口　裕希 YAMAGUCHI Yuki

東芝グループのソフトウェア開発では，品質保持や開発サイクル短縮を目的として，自動テストの導入を進めている。しかし，ソフトウェアの機能開発時のリグレッションテストが多数になる場合，自動化しても実行に時間が掛かり，開発期間を圧迫する問題があった。

東芝は，これを解決するために，最適テスト選択サービスSmallTestsを開発した。SmallTestsは，テスト実行時のカバレッジ情報及び開発時のソースコード変更情報を活用して，ソースコード変更箇所に関連するテストを選択する。バグデータセットを対象とした評価実験では，SmallTestsの適用により，実行するテスト数を最小21.0 %，最大99.1 %削減できることを確認した。

The Toshiba Group is engaged in introducing automated tests to ensure the quality of software and shorten development cycles. However, even automated tests tend to prolong the development period due to a large number of regression tests which consume a considerable amount of time.

To address this issue, Toshiba Corporation has developed SmallTests, an optimal test selection service capable of selecting tests related to changes in the source code based on code coverage information during test execution using information on source codes changed during the software development process. Evaluation experiments applying SmallTests to the Defects4J bug dataset have confirmed that it reduces the number of executed tests of a minimum 21.0% and maximum 99.1% score, respectively, compared with our conventional process.

1. まえがき

東芝グループのソフトウェア開発では，品質保持や開発サイクル短縮のために，自動テストの導入を進めている。一方，自動テスト数が多い場合，機能開発時のリグレッションテストの実行に時間が掛かる。テスト工程が，開発工程全体の1/2以上の時間を占めるプロジェクトも多く，開発期間を圧迫する要因となっている。

そこで東芝は，最適テスト選択サービスSmallTestsを開発した。SmallTestsは，テスト実行時のカバレッジ情報に基づき，全テストからソースコード変更箇所に関連するテストだけを選択するサービスである。カバレッジ情報とは，プログラム実行時に実行された命令行がどれであったかなどの情報である。

ソフトウェアのある関数のソースコードが変更されると，その関数を実行するテストは結果が変わる可能性がある。一方で，その関数を実行しないテストの結果は変わらない。ユーザーは，変更と関係のあるテストだけを実行することで，効率的にリグレッションテストを実施できる。

SmallTestsは，ソースコード変更箇所を実行するテストだけを選択するので，リグレッションテスト数を最小限にできる。これにより，ユーザーは不具合を早期に発見でき，リグレッションテストに掛かる時間を削減できる。

ここでは，SmallTestsに用いられる技術の概要と評価結果について述べる。

2. SmallTestsの概要

図1に，SmallTestsの構成を示す。SmallTestsは，カバレッジ登録サービスと，テスト計画サービスの二つから成る。ユーザーはまず，カバレッジ登録サービスにより，テストで実行されたクラス・メソッド・関数（以下，関数等と略記）をカバレッジ情報データベース（以下，カバレッジ情報DBと略記）に登録する（ステップ1）。次に，テスト対象のソフトウェアに対して機能開発を行い，ソースコードを変更する。その後，テスト計画サービスにより，ソースコード変更箇所に関連するテストを取得する（ステップ2）。

SmallTestsの各ステップで利用するサービスの詳細について，2.1節，2.2節で述べる。

2.1　カバレッジ登録サービス

ユーザーは，カバレッジ登録サービスを利用して**図2**の手順(1)～(3)を実行することで，カバレッジ情報を登録する。

(1)　カバレッジ測定リクエスト　　ユーザーが，カバレッジ登録サービスにカバレッジ測定リクエストを送ると，カバレッジ登録サービスはテスト対象のカバレッジ測定

を開始する（1.1）。

⑵　テスト実行　　ユーザーは，テスト対象上でテストを1件実行する。

⑶　カバレッジ登録リクエスト　　ユーザーは，カバレッジ登録サービスにカバレッジ登録リクエストを送る。カバレッジ登録サービスは，テスト対象からテスト実行時のカバレッジ情報を取得し（3.1），ソースコード管理システムからテスト対象のリビジョンのソースコードを取得する（3.2）。これらの情報を突き合わせ，テストで実行された関数等を取得する。最後に，“実行したテストのid（識別情報）”，“テストで実行された関数等の名前”，及び“テスト実行時のテスト対象のリビジョン”の組を，カバレッジ情報DBに登録する（3.3）。

ユーザーは，（1.1）と（3.1）の実行のために，カバレッジ登録サービスからのリクエストを処理するカバレッジ取得モジュールを，テスト対象に設置する。

各テストについて⑴〜⑶を行うことで，テスト実行時のカバレッジ情報を登録する。

カバレッジ情報の出力方法や出力形式はプログラム言語ごとに異なるため，SmallTestsはテスト対象の言語に合わせたカバレッジ記録の処理を実装している。2025年7月時点では，Java，PHP，及びPythonの言語に対応しており，随時拡充していく。

2.2　テスト計画サービス

ユーザーは，テスト計画サービスを利用して**図3**の手順⑷を実行することで，テスト選択する。

⑷　テスト選択リクエスト　　ユーザーは，テスト計画サービスにテスト選択リクエストを送る。テスト計画サービスは，ソースコード管理システムから機能開発前と後のリビジョンを取得（（4.1），（4.2））し，抽象構文木を用いて両方のリビジョンを解析する。これらを比較し，機能開発で変更された関数等を取得する。その後，カバレッジ情報DBから，これらの関数を実行するテストの一覧を取得（4.3）し，ユーザーへ提示する（4.4）。

なお，抽象構文木とは，ソースコード中のクラスや，関数，命令などを木構造で表す方法である。二つの抽象構文木の木構造を比較することで，リビジョン間で関数等の変更を検出する。また，抽象構文木を利用することで，コメントや空行の追加・削除などのテスト対象の動作に影響を及ぼさない変更を除外できる。

このようにして，ユーザーはテスト計画サービスを用いて，ソースコード変更箇所に関連するテストだけを取得し，実行できる。

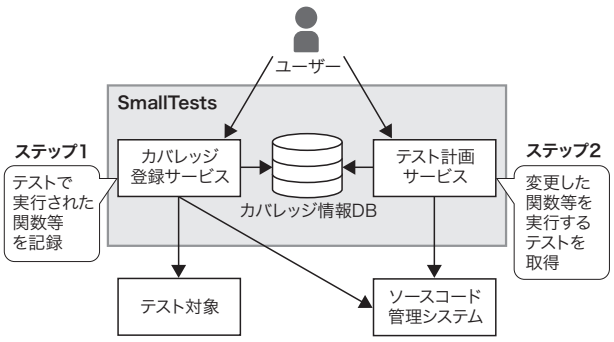

図1．SmallTestsの構成

SmallTestsは，カバレッジ登録サービスとテスト計画サービスから成り，テスト対象やソースコード管理システムと通信する。

SmallTests service configuration

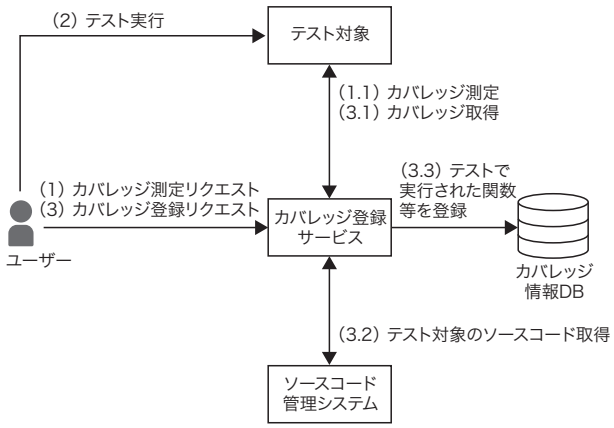

図2．カバレッジ情報の登録手順

ユーザーは，カバレッジ登録サービスを利用して，それぞれのテストについて⑴〜⑶の手順でカバレッジ情報を登録する。

Code coverage information registration process

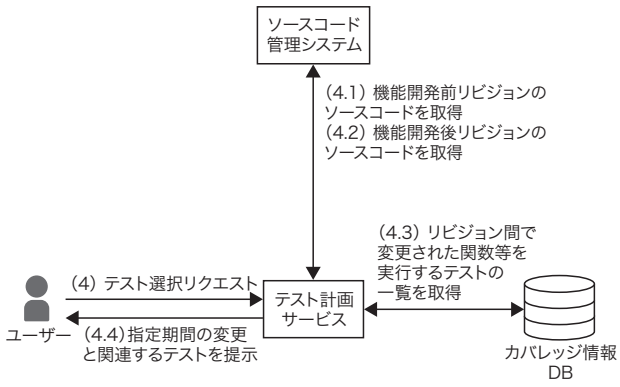

図3．テスト選択の実行手順

ユーザーは，テスト計画サービスを利用して，ソースコード変更箇所に関連するテストを取得する。

Test selection process

3. SmallTestsの評価

3.1 テスト選択の妥当性

SmallTestsによるテスト選択の妥当性を評価するため，オープンソースプロジェクトから収集されたバグデータセットDefects4J[1]を利用して，実験を行った。Defects4Jでは，対象のオープンソースプロジェクトで実際に検出された複数のバグの幾つかについて，バグがあるプログラムのリビジョン（以下，バグリビジョンと略記）のidの後にb，修正されたプログラムのリビジョン（以下，修正リビジョンと略記）のidの後にfと，それぞれラベル付けしている。この実験では，実際にバグを検出したテストをSmallTestsが選び出せるかどうかを確認した。

表1に実験結果を示す。実験では，Defects4JのCLIプロジェクト[2]の1から5，及び8から10のバグリビジョンと修正リビジョンの間の変更について，SmallTestsによるテスト選択を行った。リビジョン6,7はバグが再現不能又は不当なリビジョンであるため，評価対象から除外した。表1のリビジョン期間1bと1fの変更に対するテスト選択では，SmallTestsが選択したテストにバグを検出したテスト94が含まれていた。

表1. SmallTestsによるバグ検出テストの選択

Tests including bugs selected by SmallTests

リビジョン期間	バグを検出したテストid	SmallTestsが選択したテストid
1b–1f	**94**	1，2，…，**94**
2b–2f	**96**	6，8，…，**96**
3b–3f	**67**	**67**
4b–4f	**55**	1，2，…，**55**，…，103
5b–5f	**74**，**98**	1，2，…，**74**，…，**98**，…，105
8b–8f	**38**	**38**
9b–9f	**62**	1，2，…，**62**，…，111
10b–10f	**68**	1，2，…，**68**，…，114

太字の数字：バグを検出したテストid

表1に示したとおり，8種類のリビジョン期間でSmallTestsが選択したテストは，いずれもバグを検出したテストを含んでいて，SmallTestsが選び出せることを確認した。

3.2 テスト数削減効果

SmallTestsのテスト数削減効果を評価するため，CLIプロジェクトのバグリビジョンと修正リビジョンの間の変更に対してSmallTestsを適用し，削減されたテスト数を評価した。

表2に，CLIプロジェクトの1から5，及び8から10のバグリビジョンと修正リビジョンの組に対して，SmallTestsでテスト選択した結果を示す。CLIプロジェクトにはリビジョン期間ごとに41から47のファイルがあり，それぞれのリビジョン期間で1ファイルだけが修正されている。1ファイル中のソースコード変更箇所の数は，リビジョン期間ごとに異なる。

リビジョン3の組では，全101件のテストのうち1件だけが選択されている。これは，ソースコード変更箇所を実行するテストが少ないことを示しており，SmallTestsで多くのテストを削減できる。一方，リビジョン1の組では全95件のテストのうち，74件が選択されている。これは，ソースコード変更箇所が多くのテストで実行されることを示しており，テスト数削減効果が低い。これらから，SmallTestsで削減できるテスト数は，ソースコード変更箇所に依存する。ソースコード変更箇所が多くのテストで実行されるリビジョン1，4，5，9，10の場合は，2割程度テスト数を削減でき，最小でも21.0％のテスト数削減効果が得られた。また，ソースコード変更箇所がテストで実行されることが少ないリビジョン2，3，8の場合は，8割以上のテスト数を削減でき，最大で99.1％のテスト数削減効果が得られた。

3.3 ソースコード変更箇所の数と選択されるテスト数の関係

ソースコード変更箇所の数によるテスト数削減効果への影響を確認するため，CLIプロジェクトのバグリビジョン1と，それ以降の修正リビジョンとの間の変更に対してSmallTests

表2. SmallTests適用で削減されたテストの割合

Test reduction rates when using SmallTests

リビジョン期間	変更されたファイル数／全ファイル数	SmallTestsが選択したテスト数	全テスト数	削減されたテスト割合
1b–1f	1/41	74	95	22.1％
2b–2f	1/42	14	96	85.4％
3b–3f	1/44	1	101	99.0％
4b–4f	1/44	75	103	27.2％
5b–5f	1/46	83	105	21.0％（最小）
8b–8f	1/47	1	110	99.1％（最大）
9b–9f	1/47	79	111	28.8％
10b–10f	1/47	86	114	24.6％

表3. 変更されたファイル数とSmallTestsで削減されたテストの割合

Number of modified files and test reduction rates when using SmallTests

リビジョン期間	変更されたファイル数／全ファイル数	SmallTestsが選択したテスト数	全テスト数	削減されたテスト割合
1b–1f	1/41	74	95	22.1％
1b–2f	3/42	75	96	21.9％
1b–3f	10/44	80	101	20.8％
1b–4f	41/44	82	103	20.4％
1b–5f	43/46	86	105	18.1％
1b–8f	46/47	109	110	0.9％
1b–9f	46/47	110	111	0.9％
1b–10f	46/47	114	114	0.0％

を適用し，変更されたファイル数と削減されたテスト数について評価した。

表3に，結果を示す。全体の傾向として，変更されたファイル数が多くなるほど，テストに関連するソースコード変更箇所が多くなる傾向があるため，選択されるテスト数が多くなる。この結果から，小規模変更ごとにSmallTestsでテストを実施することにより，実行するテスト数を効果的に削減できる。

また，一つ目のバグリビジョンから四つ目の修正リビジョンの間のファイル変更では，テスト対象の動作を変更しないコメント修正が主であり，これらの変更はテスト選択に影響を与えない。そのため，変更されたファイル数が41件に対し20.4 %のテスト数を削減できた。コメントや改行による変更は，選択されるテスト数に影響しない。

4. あとがき

当社は，テストを効率良く実施することで不具合を早期に発見するSmallTestsサービスを開発した。バグデータセットを対象とした評価実験では，SmallTestsの適用により，実行するテスト数を最小21.0 %，最大で99.1 %削減できた。SmallTestsは，東芝グループ向けにWebサービスとして提供しており，テスト対象として，Java，PHP，及びPythonのプログラム言語に対応している。ユーザーはカバレッジ取得モジュールを用意するだけで，容易にリグレッションテスト数を削減できる。

今後は，対応言語の拡充や，クラウドサービスとしての展開を行い，ユーザーの利便性向上に取り組んでいく。

文　献

(1) Just, R. et al. "Defects4J: A Database of Existing Faults to Enable Controlled Testing Studies for Java Programs". Proceedings of the 2014 International Symposium on Software Testing and Analysis (ISSTA 2014). San Jose, CA, 2014-07, 2014, p.437–440. <https://homes.cs.washington.edu/~mernst/pubs/bug-database-issta2014.pdf>, (accessed 2025-02-21).

(2) Apache Commons. "Apache Commons CLI". <https://commons.apache.org/proper/commons-cli/>, (accessed 2025-02-21).

山口　裕希　YAMAGUCHI Yuki
総合研究所 デジタルイノベーション技術センター
ソフトウェアエンジニアリング技術部
Software Engineering Technology Dept.