

BreakingOutBugFinderの 評価実験

オープンソースソフトウェア工学研究室

学籍番号 60206299

吉富 楓雅

背景と目的

背景

- 一般ユーザのオープンソースソフトウェアの使用率が増加
 - * ソフトウェアの品質保証が重要
- OSSにおいてもソフトウェアテストの実施は普及しつつある

問題点

- プロジェクトが大きくなるにつれて、プロダクトコードが膨大になるため、テストコードも膨大になる
- テストコードそのものの品質に関する研究は少ない

既存研究

- Vahabzadehらはテストコードの不具合を，既存の不具合検出ツールで検出できるのかについて実証研究
 - * テストコードの不具合を分類
- **本研究との違い**
 - * **既存研究ではテストコードにのみ着目し，プロダクトコードを除外している**
 - * **本研究では，プロダクトコードも考慮に入れる**

目的

- テストコードにおける不具合を検出するツール
BreakingOutBugFinderを使用し，正しく不具合を検出
できているか評価実験を行う

テストコードの不具合

テストコードの不具合

- Cunninghamはテストコードの不具合により発生する現象を2種類に分類[1]
 - * **Silent Horror** : テストコードに不具合が存在するためにテストが成功する
 - * **False Alarm** : テストコードに不具合が存在するためにテストが失敗する

Silent Horror

- VahabzadehらはSilent Horrorを2種類に分類[2]
 - * **Missing Assertion** : 誤った例外処理によって発生
 - * **Assertion Fault** : 例外処理の欠落によって発生

Missing Assertionの例

✳	@@ -410,6 +443,7 @@	public void testTriggerNegative() throws SQLException {
410	443	expectedRows = new String[][] { {"5","two"}, {"6","four"}, {"8","eight"} };
411	444	//--- check trigger is not created
412	445	rs = s.executeQuery("select count(*) from SYS.SYSTRIGGERS where triggername='TEST_TRIG'");
446	+	JDBC.assertFullResultSet(rs, new String[][] { {"0"} });
413	447	s.execute("drop trigger create_trigger_trig");
414	448	//--- create a trigger to test we cannot drop it from a procedure called by a trigger
415	449	s.execute("create trigger test_trig AFTER delete on t1 for each STATEMENT insert into t1 values(20, 'twenty
✳	@@ -420,6 +454,7 @@	public void testTriggerNegative() throws SQLException {
420	454	JDBC.assertFullResultSet(rs, new String[][] { {"1","2"}, {"2","4"} });
421	455	//--- check trigger is not dropped
422	456	rs = s.executeQuery("select count(*) from SYS.SYSTRIGGERS where triggername='TEST_TRIG'");
457	+	JDBC.assertFullResultSet(rs, new String[][] { {"1"} });
423	458	s.execute("drop trigger drop_trigger_trig");
424	459	//- use procedures which create/drop index on trigger table and some other table
425	460	s.execute("create trigger create_index_trig AFTER delete on t2 for each STATEMENT call create_index_proc()");
✳	@@ -430,6 +465,7 @@	public void testTriggerNegative() throws SQLException {
430	465	JDBC.assertFullResultSet(rs, new String[][] { {"1","2"}, {"2","4"} });
431	466	// -- check index is not created
432	467	rs = s.executeQuery("select count(*) from SYS.SYSCONGLOMERATES where CONGLOMERATENAME='IX' and ISINDEX=1");
468	+	JDBC.assertFullResultSet(rs, new String [][] { {"0"} });
433	469	s.execute("drop trigger create_index_trig");
434	470	//--- create an index to test we cannot drop it from a procedure called by a trigger
435	471	s.execute("create index ix on t1(i,b)");
✳	@@ -438,6 +474,7 @@	public void testTriggerNegative() throws SQLException {
438	474	// -- check delete failed
439	475	rs = s.executeQuery("select * from t1");
440	476	expectedRows = new String[][] { {"5","two"}, {"6","four"}, {"8","eight"} };
477	+	JDBC.assertFullResultSet(rs, expectedRows);
441	478	// -- check index is not dropped
442	479	rs = s.executeQuery("select count(*) from SYS.SYSCONGLOMERATES where CONGLOMERATENAME='IX' and ISINDEX=1");
443	480	JDBC.assertFullResultSet(rs, new String[][] { {"1"} });

False Alarm

- VahabzadehらはFalse Alarmを5種類に分類[2]
 - * Semantic Bug : 仕様やコードに矛盾が生じる
 - * Flaky Test : 同バージョンでテストが成功・失敗を繰り返す
 - * Environmental Bug : パス区切りなど環境による違いによる
 - * Inappropriate Handling of Resource : テストの依存性による
 - * Obsolete Test : 古いテストを使用した際に予期せぬエラーが起こる

BreakingOutBug Finder

調査項目

- ツールが検出したSilent Horror, False Alarmを引き起こすテストコードの不具合は各プロジェクトの全不具合のうち何件か？
- 実際にSilent Horror, False Alarmを引き起こす不具合は何件か？

ファイルの変更部分(chunk)

- ファイルの変更のかたまりを**chunk**と呼ぶ
 - * **DELETE** : 行の削除
 - * **INSERT** : 行の追加
 - * **REPLACE** : 行の修正(削除および追加)

ファイルの変更

✚	@@ -68,7 +58,6 @@	
68	58	+ * Out-of-order writes are handled gracefully. Time range overlapping among store files is tolerated
69	59	* and the performance impact is minimized. Configuration can be set at hbase-site or overridden at
70	60	* per-table or per-column-family level by hbase shell. Design spec is at
71	-	>>>>>> 3f749fd... HBASE-15368 Add pluggable window support
72	61	* https://docs.google.com/document/d/1_AmLnB2N8Us1xICsTeGDLKIqL6T-oHoRLZ323MG_uy8/
73	62	*/
74	63	@InterfaceAudience.LimitedPrivate(HBaseInterfaceAudience.CONFIG)
✚	@@ -121,13 +110,20 @@ public boolean needsCompaction(final Collection<StoreFile> storeFiles,	
121	110	public boolean shouldPerformMajorCompaction(final Collection<StoreFile> filesToCompact) throws IOException {
122	111	long mcTime = getNextMajorCompactTime(filesToCompact);
123	112	if (filesToCompact == null mcTime == 0) {
113	+	if (LOG.isDebugEnabled()) {
114	+	LOG.debug("filesToCompact: " + filesToCompact + " mcTime: " + mcTime);
115	+	}
124	116	return false;
125	117	}
126	118	
127	119	// TODO: Use better method for determining stamp of last major (HBASE-2990)
128	120	long lowTimestamp = StoreUtils.getLowestTimestamp(filesToCompact);
129	121	long now = EnvironmentEdgeManager.currentTimeMillis();
130	-	if (lowTimestamp <= 0L lowTimestamp >= (now - mcTime)) {
122	+	if (lowTimestamp <= 0L lowTimestamp >= (now - mcTime)) {
123	+	if (LOG.isDebugEnabled()) {
124	+	LOG.debug("lowTimestamp: " + lowTimestamp + " lowTimestamp: " + lowTimestamp + " now: " +
125	+	now + " mcTime: " + mcTime);
126	+	}
131	127	return false;
132	128	}
133	129	
✚		

DELETE

INSERT

REPLACE

検出の手順

- 1.変更のあったメソッドのうち、コードが追加された
(INSERT, REPLACE)行を取得
- 2.(1)の親コミットを取得し、コードが削除された
(DELETE, REPLACE)行を取得
- 3.(1)(2)について、ファイル名が".java"で終わらないファイル
を除外

検出の手順

4.各ファイルの**chunk**について

A. メソッド内の変更は, **INSERT, DELETE, REPLACE**のうちどの変更か?

B. どのメソッドが変更されたのか?

▶ メソッドの変更については, **INSERTがX行以内, かつDELETEがX行以内(X=1, 3, 5)**

C. (B)を満たす場合, **SilentHorrorProne**

検出の手順

- フィルタ： Type: Bug, Resolution: fixed
- フィルタされた不具合からそれぞれ10%を目視
 - * 実際にSilent Horror, False Alarmを引き起こしたか

評価実験

対象プロジェクト

プロジェクト	プロダクトコード の行数(KLOC)	テストコード の行数(KLOC)	不具合報告数 (件)
Derby	386	370	614
HBase	587	195	440
Hive	836	124	295
Hadoop HDFS	101	57	286
Hadoop Common	1,249	380	279
Hadoop Map/Reduce	60	24	231
Accumulo	405	78	187
Qpid	553	93	152
Jackrabbit	247	107	145
Cloud Stack	1,361	228	111

目視結果

現在目視中