

不具合見逃しの原因となる テストコードの検出支援ツールの 評価

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

- **ソフトウェア開発において品質保証活動が重要**
 - * 品質が保証されたソフトウェアをリリースするには、不具合を早期に発見することが必要
- **品質保証活動の1つにソフトウェアテストがある**
 - * システムの動作が仕様通りであるか確認し、仕様を満たさない動作をする不具合を発見する作業

```
public class Calculator () {  
    public int multiply (int x, int y) {  
        return x * y;  
    }  
}
```

Calculator.java

2変数の積が
正しく行われているかをテスト

```
public class CalculatorTest () {  
    @Test  
    public void multiplyTest () {  
        Calculator calc = new Calculator();  
        int expected = 12;  
        int actual = calc.multiply (3, 4);  
        assertThat(actual, is(expected));  
    }  
}
```

テストコードの不具合

- プロダクトコードの増加に伴いテストコードも増加
 - * 現在のソフトウェア開発では、テストコードはプロダクトコードと同量かそれ以上のコードを記述することが多い
- テストコードに不具合が混入する可能性が高まる



プロダクトコード



テストコード

テストコードの不具合により生じる問題

Silent Horror : テストコードに不具合があるために
プロダクトコードの不具合があってもテストが成功する

```
// must throw a NamespaceException.  
try {  
    nsp.unregisterNamespace("ThisNamespaceIsNotCurrentlyRegistered");  
+    fail("Trying to unregister an unused prefix must fail");  
} catch (NamespaceException e) {  
    // expected behaviour  
}
```

fail()の書き忘れ

Description

In two places, the test doesn't fail if an expected exception isn't thrown.

Proposal: test should fail if expected exception isn't thrown.

— NamespaceRegistryTest.java (revision 422074)

+++ NamespaceRegistryTest.java (working copy)

@@ -150,6 +154,7 @@

for (int t = 0; t < SYSTEM_PREFIXES.length; t++) {

try

{ nsp.unregisterNamespace(SYSTEM_PREFIXES[t]); + fail("Trying to unregister " + SYSTEM_PREFIXES[t] + " must fail"); }

catch (NamespaceException e)

{ // expected behaviour }

@@ -159,6 +164,7 @@

// must throw a NamespaceException.

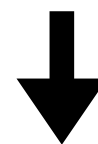
try

{ nsp.unregisterNamespace("ThisNamespacelsNotCurrentlyRegistered"); + fail("Trying to unregister an unused prefix must fail"); }

catch (NamespaceException e)

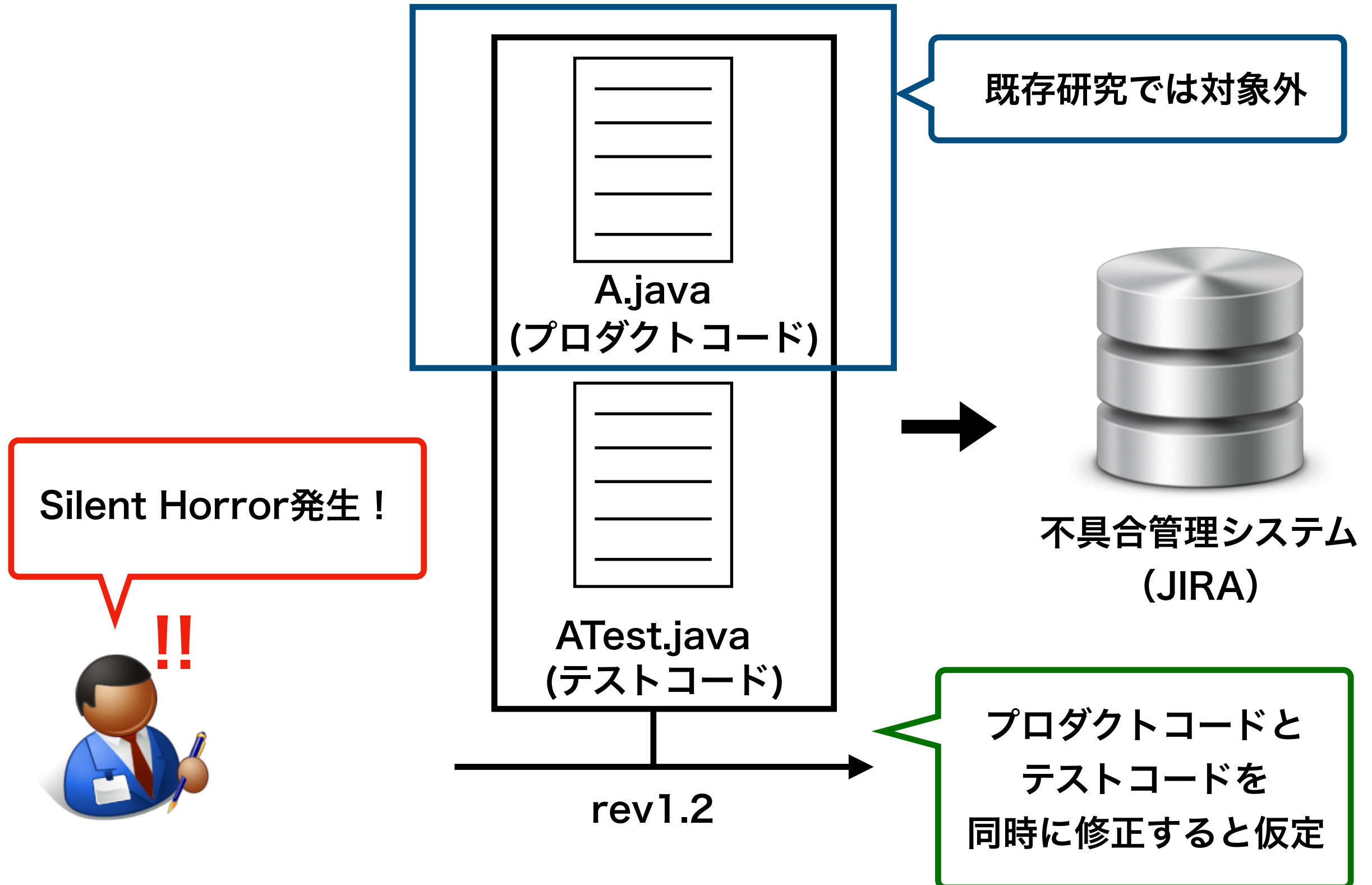
{ // expected behaviour }

期待している
例外がスローされない



テストは失敗せず
プロダクトコードの
不具合が発見できない

既存研究との違い



研究の目的

- 目的

- * Kashiwaらが提案したSilent Horrorを検出するツールの評価

- アプローチ

- * 目視により実際にSilent Horrorが発生しているか確認

ツール概要

- 目的

- * ソフトウェアシステムにおけるSilent Horrorの検出

- ▶ テストメソッドからX行以内の変更が行われたコミットを特定

検出手順

javaファイルを
対象. それ以外の
ファイルは除外

ディレクトリ名
に"test"があれば
テストコード

コメントアウト
空行は除外

```
5 src/test/java/org/apache/hadoop/hbase/zookeeper/TestHQuorumPeer.java
@@ -132,14 +132,11 @@
132 132 assertEquals("foo.bar", server.addr.getHostName());
133 133 }
134 134
135 - /**
136 -  * Test Case for HBASE-2305
137 -  */
138 135 @Test public void testShouldAssignDefaultZookeeperClientPort() {
139 136 Configuration config = HBaseConfiguration.create();
140 137 config.clear();
141 138 Properties p = ZKConfig.makeZKProps(config);
142 139 assertNotNull(p);
143 - assertEquals(2181, p.get("hbase.zookeeper.property.clientPort"));
140 + assertEquals(2181, p.get("clientPort"));
144 141 }
145 142 }
```

削除行

追加行

1. プロダクトコードとテストコードを含むコミットであることを確認
2. テストメソッドで追加・削除された行を特定
3. 各メソッドの変更行数をカウント
4. 各メソッドの追加行, 削除行のそれぞれの合計がX行以内であれば,
Silent Horrorを引き起こす候補

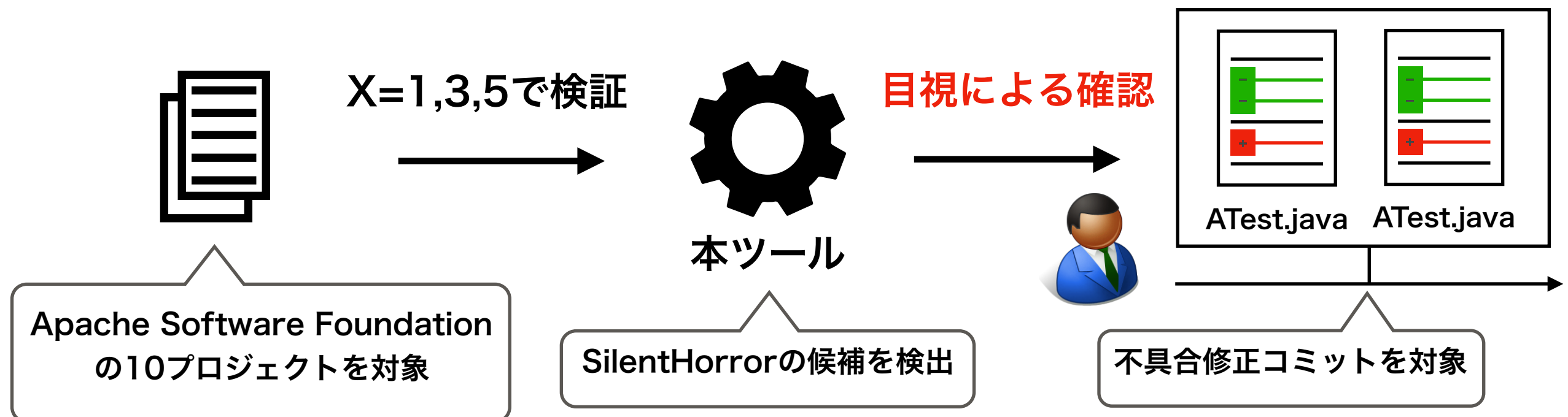
実験概要

- 目的

- * 実際にSilent Horrorが引き起こされている件数を調査する

- 評価項目

- * 各プロジェクトでSilentHorrorの候補の件数を確認
 - * 実際にSilent Horrorを引き起こした件数を確認



目視方法



不具合票

A.javaで異なるポート番号が返される

<不具合詳細>

A.javaではポート番号2181が返されるべきであるが、異なるポート番号が返された

<コメント>

hyohyo 2018年10月1日 12:00

調査の結果、ポート番号3314が誤って返されていました。原因を追求します。

tommy 2018年10月2日 16:30

ATest.javaの285行目は unnecessary です。
異なるポート番号3314を返す原因です。

hyohyo 2018年10月2日 20:00

tommy, ありがとう。
修正パッチを添付します。

160	
161	-
162	+
163	
164	-

A.java

280	
281	-
282	-
283	
284	
285	+

ATest.java

③

ATest.javaによって
見逃された不具合が
修正されている

②

285行目の不具合を確認

①

テストコードの不具合が
報告されている

Silent Horror !



実験結果：SilentHorrorの候補

	SilentHorrorの候補の件数			
	X = 1	X = 3	X = 5	ベースライン
Derby	153	106	65	321
HBase	594	335	127	622
Hive	412	242	98	472
Hadoop HDFS	26	23	8	44
Hadoop Common	216	145	75	381
Hadoop Map/Reduce	47	43	16	67
Accumulo	96	57	27	83
Qpid	26	15	11	35
Jackrabbit	53	40	13	176
Cloud Stack	81	38	29	93

実験結果：Silent Horrorの件数

	Silent Horrorの件数			
	X = 1	X = 3	X = 5	ベースライン
Derby	0	0	0	0
HBase	1	0	0	0
Hive	0	1	0	0
Hadoop HDFS	0	0	0	0
Hadoop Common	0	0	0	0
Hadoop Map/Reduce	0	0	0	0
Accumulo	0	1	0	0
Qpid	0	0	0	0
Jackrabbit	0	1	0	0
Cloud Stack	0	0	0	0

考察

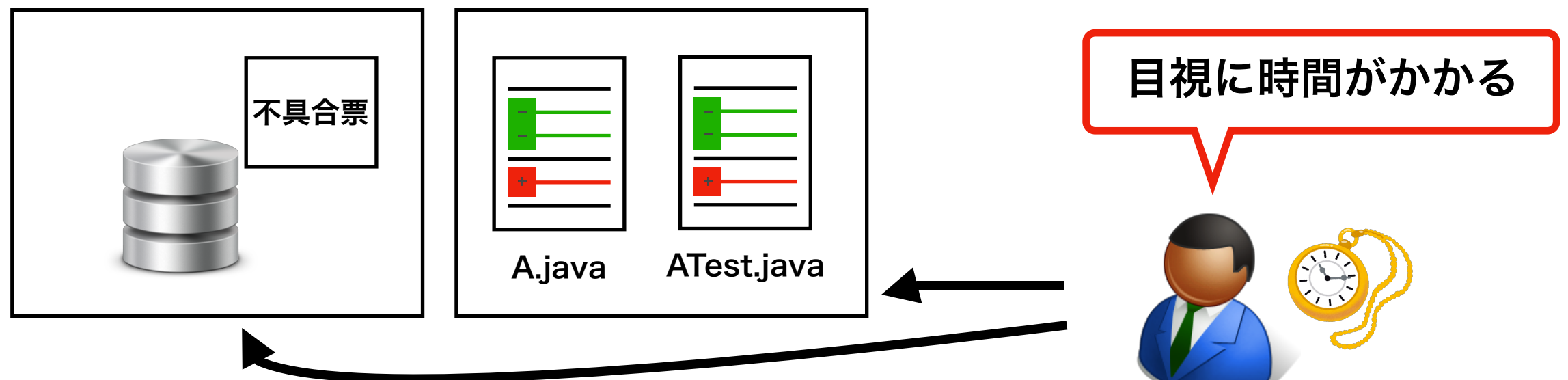
- ツールの問題点

- * Silent Horrorの候補の件数は依然として多数あり目視に時間がかかる
- * 565件を目視：約250時間を要した

- 目視件数を削減するために再度目視

- テストメソッドの追加が散見された

- * 4件にはテストメソッドの追加は行われていなかった



考察：テストメソッドの追加を含む

Silent Horrorの候補の件数

	テストメソッドの追加を含むSilent Horrorの候補の件数			
	X = 1	X = 3	X = 5	ベースライン
Derby	12 / 15 (80.0%)	8 / 11 (72.7%)	4 / 7 (57.1%)	24 / 32 (75.0%)
HBase	25 / 60 (41.7%)	19 / 34 (55.9%)	9 / 13 (69.2%)	51 / 62 (82.3%)
Hive	9 / 42 (21.4%)	9 / 25 (36.0%)	2 / 10 (20.0%)	26 / 47 (55.3%)
Hadoop HDFS	1 / 3 (33.3%)	1 / 3 (33.3%)	1 / 1 (100.0%)	3 / 4 (75.0%)
Hadoop Common	11 / 22 (50.0%)	6 / 15 (40.0%)	2 / 8 (25.0%)	30 / 38 (78.9%)
Hadoop Map/Reduce	4 / 5 (80.0%)	3 / 5 (60.0%)	1 / 2 (50.0%)	4 / 7 (57.1%)
Accumulo	3 / 13 (23.1%)	2 / 6 (33.3%)	1 / 4 (25.0%)	6 / 8 (75.0%)
Qpid	0 / 3 (0.0%)	1 / 2 (50.0%)	1 / 1 (100.0%)	3 / 4 (75.0%)
Jackrabbit	5 / 5 (100.0%)	3 / 4 (75.0%)	1 / 1 (100.0%)	14 / 18 (77.8%)
Cloud Stack	2 / 8 (25.0%)	4 / 4 (100.0%)	2 / 3 (66.7%)	6 / 10 (60.0%)

考察：テストメソッドの追加を除く

Silent Horrorの候補の件数

	テストメソッドの追加を除く Silent Horrorの候補の件数			
	X = 1	X = 3	X = 5	ベースライン
Derby	3	3	3	8
HBase	35	15	4	11
Hive	33	16	8	21
Hadoop HDFS	2	2	0	1
Hadoop Common	11	9	6	8
Hadoop Map/Reduce	1	2	0	3
Accumulo	10	4	3	2
Qpid	3	1	0	1
Jackrabbit	0	1	0	4
Cloud Stack	6	0	1	4

まとめ

- Kashiwaらが提案したSilent Horrorを検出するツールの評価を行った
- 10のプロジェクト(10%抽出)のうち4件がSilent Horrorであった
- 本ツールは依然として目視に時間を要するため、テストメソッドを含む不具合修正コミットを除外することでSilent Horrorの候補の件数を削減できると考えられる