EXPOSING OPAQUE CHANGES BY CONTRASTING ANALYSES

Reid Holmes
David Notkin

"[We should do] our utmost best to shorten the conceptual gap between the static program and the dynamic process"

--Dijkstra
MOTIVATION

- Predicting the runtime behaviour of a source code change can be difficult
- Inconsistency can arise when the static and dynamic components of a change differ
- Most changes are not interesting or surprising
- By identifying inconsistent changes when they happen, we hope to reduce latent, hard-to-detect errors
PROCESS

Version 1

Version 2
PROCESS

Version 1

Static Analysis

Dynamic Analysis

Version 2
PROCESS

Version 1

Static Analysis

Dynamic Analysis

Version 2
private void genStore() {
    ...
    cache()
}

private void cache() {
    LocalType l = new LocalType();
    _collection.add(l);
}
**EXAMPLE - STATIC VIEW**

```java
private void genStore() {
    ...
    cache();
}

private void cache() {
    LocalType l = new LocalType();
    _collection.add(l);
}
```
private void genStore() {
    ...
    cache()
}

private void cache() {
    LocalType l = new LocalType();
    _collection.add(l);
}
PARTITIONED EXAMPLE

s_2 \rightarrow \text{genStore()} \rightarrow \text{cache()}
\text{cache()} \rightarrow \text{cache()}
\text{cache()} \rightarrow \text{LocalType()}
\text{cache()} \rightarrow \text{Collection.add(...)}
\text{d}_2 \rightarrow \text{LocalType.equals(...)}
\text{d}_1 \rightarrow \text{...}

s_1 \rightarrow \text{...}

\text{genStore()} \rightarrow \text{cache()}
\text{cache()} \rightarrow \text{LocalType()}
\text{cache()} \rightarrow \text{Collection.add(...)}
\text{Collection.add(...)} \rightarrow \text{LocalType.equals(...)}
COMBINED ANALYSES
COMBINED ANALYSES

genStore() → cache()
cache() → LocalType()
cache() → Collection.add(…)

Reid Holmes - UW CSE Affiliates 2009
COMBINED ANALYSES

genStore() → cache()
cache() → LocalType()
cache() → Collection.add(...)

Collection.add(...) → LocalType.equals(...)
PARTITIONS

LEGEND

- \( s \) statically unchanged
- \( s^+ \) statically added
- \( s^- \) statically removed
- \( d \) dynamically unchanged
- \( d^+ \) dynamically added
- \( d^- \) dynamically removed
COMPACT NOTATION

LEGEND
s  statically unchanged
s⁺  statically added
s⁻  statically removed
d  dynamically unchanged
d⁺  dynamically added
d⁻  dynamically removed
COMPACT NOTATION

LEGEND
s  statically unchanged
s⁺  statically added
s⁻  statically removed
d  dynamically unchanged
d⁺  dynamically added
d⁻  dynamically removed

s⁻  s⁻d⁻
s⁺  s⁺d⁺
d⁻  sd⁻
d⁺  sd⁺

Reid Holmes - UW CSE Affiliates 2009
QUALITATIVE FINDINGS

• Visualizer v22 → v23
  • Callback disappears (d⁻)
  • Callback originally happened within an anonymous class

```
com.google.common.collect.Ordering.givenOrder(List) →
AggregationColumn.equals(Object)
```
QUALITATIVE FINDINGS

- JodaTime v1366→v1367
  - Inconsistency between s^+ and s^+d^+
    - A thrown exception prevents an added method call
  - Three unexpected method calls (d^+)
  - New tests executed by JUnit

```
TestDuration_Basics.testToStandardSeconds_overflow() → Assert.fail()
```

LEGEND

- s  statically unchanged
- s^+ statically added
- s^- statically removed
- d  dynamically unchanged
- d^+ dynamically added
- d^- dynamically removed
COMPARING LIBRARIES

• Run the same program with two library versions
• Run JodaTime with JDK5 and JDK6
  • One new method call
    • [via reflection to new API]
  • One old method call disappeared

\[
\begin{align*}
s^+ &\quad \text{DateTimeUtils.getDateFormatSymbols(} \text{Locale}) \rightarrow \text{Method.invoke(} \text{Object, Object[]}\text{)} \\
\text{sd}^- &\quad \text{DateTimeUtils.getDateFormatSymbols(} \text{Locale}) \rightarrow \text{DateFormatSymbols(} \text{Locale})
\end{align*}
\]
CONCISENESS

- Total # of facts in each partition across all 30 program versions evaluated
  - $d^+$: 77% fewer facts to consider
  - $d^-$: 80% fewer facts to consider
CONCISENESS

- Total # of facts in each partition across all 30 program versions evaluated
  - \(d^+\): 77% fewer facts to consider
  - \(d^-\): 80% fewer facts to consider

99+% reduction in interesting partitions
FUTURE DIRECTIONS

• Longitudinal analysis: Behavioural changes could be tracked through time by analyzing the past data collected through our analysis

• Impact analysis: By comparing actual inconsistent changes with expected inconsistent changes, we could evaluate if the impact of the change is sufficiently understood
• Program behaviours can be hard to discern from the source code.
• While most behaviours are not surprising, by combining static and dynamic analyses we are able to concisely partition inconsistent behaviour.