UNIVERSITÄT BERN

Towards Detecting Inconsistent Comments in Java Source Code Automatically*

**to appear in the NIER track of SCAM'20*

***Nataliia Stulova**, Arianna Blasi, Alessandra Gorla, Oscar Nierstrasz*

Software Composition Seminar
22 September 2020

Problem: commits introducing inconsistencies

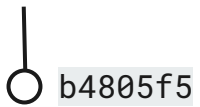


```
/**
* Checks if one of the graphs is from unsupported graph type and throws
* IllegalArgumentException if it is. The current unsupported types are
* graphs with multiple-edges.
*
* @param graph1
* @param graph2
*
*
* @throws IllegalArgumentException
*/
protected static void assertUnsupportedGraphTypes(
    Graph g1,
    Graph g2)

    throws IllegalArgumentException
{...}
```

Problem: commits introducing inconsistencies

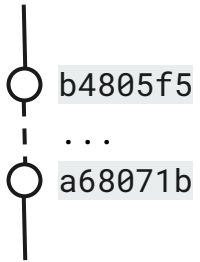
```
 /**
 * Checks if one of the graphs is from unsupported graph type and throws
 * IllegalArgumentException if it is. The current unsupported types are
 * graphs with multiple-edges.
 *
 * @param graph1
 * @param graph2
 *
 *
 * @throws IllegalArgumentException
 */
 protected static void assertUnsupportedGraphTypes(
-    Graph g1,
-    Graph g2)
+    Graph g)
      throws IllegalArgumentException
 {...}
```



code and comment diverged!..

Problem: commits introducing inconsistencies

```
 /**
 * Checks if one of the graphs is from unsupported graph type and throws
 * IllegalArgumentException if it is. The current unsupported types are
 * graphs with multiple-edges.
 *
-* @param graph1
-* @param graph2
+* @param g
 *
 * @throws IllegalArgumentException
 */
 protected static void assertUnsupportedGraphTypes(
     Graph g)
     throws IllegalArgumentException
 {...}
```



… still inconsistent
7.5 years later

Solution idea: detect inconsistencies with NLP

Step 0: get the diff of the change

Step 1: extract *natural language cues* from both code and comments and store as Bag-of-Words (BoW)

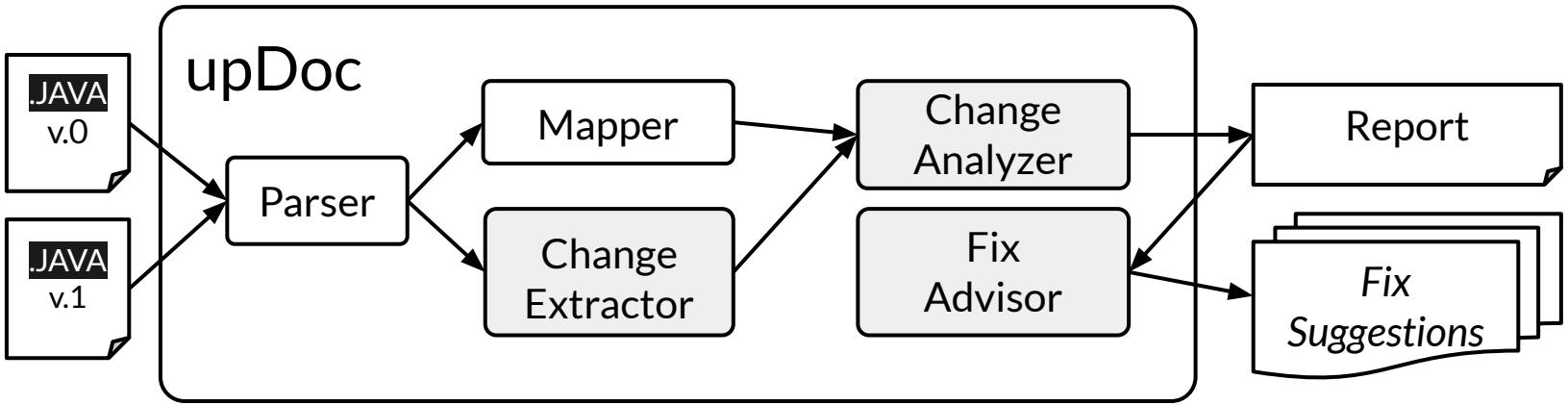
stems

Step 2: measure BoW similarities

Word Mover's Distance

		WMD similarity
comment	`[1:illeg, 1:argument, 1:throw, 1:one, 1:except, 1:check, 1:unsupport, 1:type, 2:graph]`	
code v.0	`[1:illeg, 1:argument, 1:throw, 1:void, 1:assert, 1:except, 1:unsupport, 1:type, 4:graph, 1:first, 1:second]`	70%
code v.1	`[1:illeg, 1:argument, 1:throw, 1:void, 1:assert, 1:g, 1:except, 1:unsupport, 1:type, 2:graph]`	66% (**-4%**)

Implementation prototype



Implementation details

Implemented

- **Parser**: processes Java source code
- **Mapper**:
  - produces bag-of-words representations for code and comments
  - relates them based on similarity measures: cosine similarity, word mover's distance

Working on

- **Change extractor**: work with AST-based diffs for clear code snippet scopes
- **Change analyzer**: combine information from the mapping and from the diff
- **Fix advisor**: notify the programmer if the comment needs more changes

Preliminary evaluation of mapping accuracy

- get public inconsistencies dataset from ICPC'19
- take first 50 commits and filter out fixes not detected (yet) by upDoc
- analyze 67 fixes in 20 commits that remained after filtering

Results:

- For 50 fixes the similarity scores improve as expected
- For 10 fixes the similarity scores did not change
- For 7 fixes upDoc reports unexpected decreases in the similarity scores

upDoc 's mapping between methods and doc comments accurately reflects inconsistencies

in 90% of the cases!

Evaluation highlights 1/3

Inconsistency fix, correctly detected:

```
-* This automatically calls {@link #escapeHTML} so make it safe for HTML too.
+* This automatically calls {@link #sanitizeForHtml} so make it safe for HTML too.
+*
+* @param string
+* @return the sanitized string or null (if the parameter was null).
 */
 public static String sanitizeForJs(String str){...}
```

new version has **better** similarity score than old version

Evaluation highlights 2/3

Readability fix, correctly skipped:

```
-* Invoked when a refresh of current IndexReaders is detected as necessary.
-* The implementation is blocking to maximize reuse of a single IndexReader (better for buffer usage,
-* caching, ..) and to avoid multiple threads to try and go opening the same resources at the same time.
-* @return the refreshed IndexReader
+* Invoked when a refresh of current {@code IndexReader}s is detected necessary.
+*
+* The implementation is blocking to maximize reuse of a single {@code IndexReader} (better for buffer usage,
+* caching, ..) and to avoid multiple threads trying and opening the same resources at the same time.
+*
+* @return the refreshed {@code IndexReader}
+*/
 private synchronized IndexReader refreshReaders() {
```

old and **new** versions have **same** similarity scores

Evaluation highlights 3/3

Inconsistency fix, NOT correctly detected:

```
 * Fetch an {@link InputStream} for each {@link Resource} in this {@link ResourceList}, pass the
 * {@link InputStream} to the given {@link InputStreamConsumer}, then close the {@link InputStream} after the
-* {@link InputStreamConsumer} returns.
+* {@link InputStreamConsumer} returns, by calling {@link Resource#close()}.
   ...
 public void forEachInputStreamThenClose(final InputStreamConsumer inputStreamConsumer,
           final boolean ignoreIOExceptions) {
```

old version has a **better** similarity score than the new, fixed version