Client-Specific Upgrade Compatibility Checking via Knowledge-Guided Discovery

Chenguang Zhu¹, Mengshi Zhang², Xiuheng Wu², Xiufeng Xu², Yi Li³
1. The University of Texas at Austin, USA
2. Meta Platforms, USA
3. Nanyang Technological University, Singapore

ICSE 2023
May 18, 2023
To upgrade, or not to upgrade

50%  A recent study\textsuperscript{[1]} shows 50% of the 408 studied open-source Java projects break after an upgrade to library

40%  40% of the breakages were runtime test failures (not caught by compilers)

82%  In another study\textsuperscript{[2]}, 82% of the developers of the studied systems keep outdated dependencies, leaving system open to zero-day attacks

\textsuperscript{[1]} Alex Gyori, Owolabi Legunsen, Farah Hariri, and Darko Marinov. Evaluating regression test selection opportunities in a very large open-source ecosystem. In ISSRE’18.

\textsuperscript{[2]} Raula Gaikovina Kula, Daniel M. German, Ali Ouni, Takashi Ishio, and Katsuro Inoue. Do developers update their library dependencies? In Empirical Software Engineering’18
Client-Specific Library Upgrade Incompatibility

- Will a library upgrade break a specific client (resulting in different behaviors)?

Client 1:

if (lib(s1, s2).length() > 0) {
    ...
}

*Incompatible for Client 1 (may throw NullPointerException)*

Client 2:

lib(
    x.concat("abc"),
    y.concat("abc")
);

*Compatible for Client 2*
Our Solution: Client-Specific Compatibility Checking (CompCheck)
Our Solution: Client-Specific Compatibility Checking (CompCheck)

```
public void testKryo() {
    Kryo kryo = new Kryo();
    ByteArrayOutputStream bytes = new ByteArrayOutputStream();
    Output output = new Output(bytes);
    Boo b = new Boo("hello");
    kryo.writeObject(output, b);
    output.close();
    ...
}
```
Our Solution: Client-Specific Compatibility Checking (CompCheck)
Our Solution: Client-Specific Compatibility Checking (CompCheck)

A new client method with matching context

```java
public static byte[] serialize(Kryo kryo, Object o) {
    ... 
    Output output = new Output(4096);
    kryo.writeObject(output, o);
    output.flush();
    return output.getBuffer();
}
```

New test generated (by reusing stored states)

---

Diagram:

- **Incompatibility Discovery**
  - Incompatibility-Revealing Tests
  - Validation

- **Test Suite Generation**
  - Matched Client Methods
  - Knowledge Matching

- **New Target Client**
  - Merged Client Contexts (FSMs)
  - Knowledge Storage

- **Open Source Projects**
  - Failed Tests
  - Library Version Pair
  - Execution Trace

**API Incompatibility Knowledge**

- **Knowledge Extraction**
- **Knowledge Aggregation**

**Module-Level Regression Testing**

**Knowledge Mining**

@Test
public void test0() {
    Kryo kryo = loadObj(arg_0_S1);
    Boo b = loadObj(arg_2_S1);
    serialize(kryo, b);
}
Evaluation Subjects

- 24 Backward Incompatible APIs from 8 popular libraries

- 35 high-starred Java client projects on GitHub, having 202 call sites in total
Evaluation Baselines

- Baselines
  - Sensor\textsuperscript{[1]}: generating tests to reveal library dependency conflicts
  - CIA+SBST: Uses change impact analysis (CIA) to find the call sites affected by the library upgrade, then perform search-based test generation
  - CompCheck--: Disable object reusing of CompCheck

- Comparison Goals
  - CompCheck vs Sensor vs CIA+SBST: An end-to-end comparison on the effectiveness of incompatibility discovery
  - CompCheck vs CompCheck--: Measure the benefit of object reusing

\textsuperscript{[1]} Ying Wang, Rongxin Wu, Chao Wang, Ming Wen, Yepang Liu, Shing-Chi Cheung, Hai Yu, Chang Xu, and Zhi-liang Zhu. Will Dependency Conflicts Affect My Program’s Semantics? TSE 2021.
1. CompCheck is effective in discovering incompatibility issues. It revealed 72.7% more issues than Sensor and 94.9% more issues than CIA+SBST

2. Object reusing significantly contributes to the overall effectiveness
Problem Highlight

Many library upgrades have client-specific compatibility issues, which needs to be analyzed case by case.

Tool: CompCheck

Source code and experiment data publicly available

https://sites.google.com/view/compcheck

Evaluation

Revealed 72.7% more issues with incompatibility tests, than existing techniques

Dataset: CompSuite

A newly compiled dataset for library behavioral incompatibilities

https://github.com/compsuite-team/compsuite