“What Programmers Do With Inheritance in Java”* – Replicated on Source Code

Cigdem Aytekin | Dr. Tijs van der Storm | University of Amsterdam

Problem

Replicate the study “What Programmers Do with Inheritance in Java” (1) to verify its results.

Research Questions

How are the results of our replication study with respect to the research questions of the original study? How often do we see:

Late-bound self-reference

Subtyping

Reuse

Other uses

Project Overview

- The original study proposes a model for usage of defined inheritance relationships in Java,
- Using this model, the authors analyzed the byte code of 93 open source Java projects from Qualitas Corpus (2),
- They found that defined inheritance relationships are also used intensively.
- We replicated this study on Java source code,
- We analyzed 90 projects from Qualitas.class Corpus (3).
- Our results are similar to the original results, but they are not the same.

Inheritance Usage - Examples

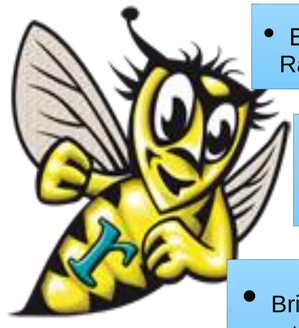
```
public class P {
  void p() {
    q();  // down-call  }
  void q() {}
  void a() {}
}
public class C extends P {
  void q() {}
  void c() {
    a(); // internal reuse
  }
}
```

```
public class N {
  void n() {
    P aP = new C(); // subtype
    C aC = new C();
    aC.a(); // external reuse
  }
}
```

Inheritance Usage Model

Inheritance Use	Definition
Down-call:	Late-bound self-reference. A parent method issues a call to another parent method, which is overridden by child class.
Subtyping:	A child object is supplied where a parent is expected.
External Reuse:	Child object reuses code of parent object (outside of child class code).
Internal Reuse:	Child object reuses code of parent object (inside of child class code).
Other Uses:	Other uses like: constant, marker, category, super, etc.

Replication - Implementation



- Build Java source AST’s with Rascal.
- Visit the relevant AST nodes and use the Rascal M3 model
- Bring the results together for different metrics and put them in Rascal relations.

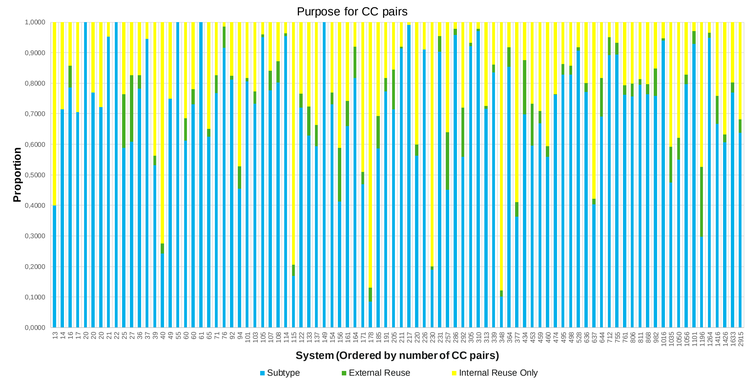
Differences in Study Set-up

Original	Replication
Java byte code	Java source code
Qualitas Corpus (2)	Qualitas.class Corpus (3)

Replication – Major Limitations

- 25 out of 90 projects have different versions in Qualitas.class Corpus. For the projects with same versions, the content of source and byte code distributions are quite different.
- Limitation about analysis of methods defined outside of the project: fewer cases for subtype and external reuse.

Results



Results (median %)	Original	Replication
Down-call	34	27
Subtype - between classes	76	76
External Reuse	22	4
Internal Reuse	2	20
Other uses	Not significant	Not significant

Conclusion

- Our results verify the results of the original study for all research questions to a great extent.
- Most differences are the result of differences between the analyzed projects and our analysis limitation about external methods.
- We suspect some false positives for down-call and external reuse in the original study, but there are not many of them. We suspect that byte code analysis can be misleading in these cases.

Works Cited

- (1) : “What Programmers Do with Inheritance in Java”, by E. Tempero, H. Y. Yang and J. Noble, ECOOP 2013,p: 577 - 601
- (2) Tempero, Ewan, et al. "The Qualitas Corpus: A curated collection of Java code for empirical studies." *Software Engineering Conference (APSEC), 2010 17th Asia Pacific*. IEEE, 2010.
- (3) Terra, Ricardo, et al. "Qualitas. class Corpus: A compiled version of the Qualitas Corpus." *ACM SIGSOFT Software Engineering Notes* 38.5 (2013): 1-4.