Supplying Compiler's Static Compatibility Checks by the Analysis of Third-party Libraries

Kamil Jezek, Lukas Holy, Premek Brada

Department of Computer Science and Engineering
University of West Bohemia
Pilsen, Czech Republic
{kjezek,lholy,brada}@kiv.zcu.cz

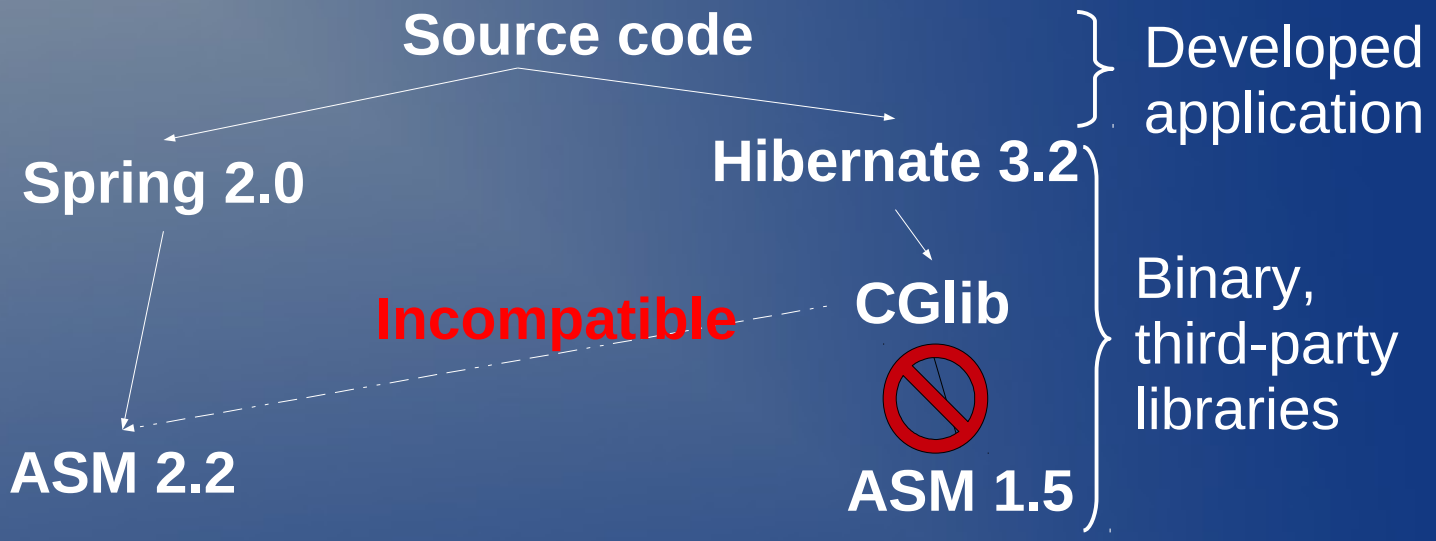
CSMR 2013

Problem Definition

- Statically typed languages prevent type errors
  - They check source-code against library dependencies (e.g. method signatures)
- Libraries have transitive dependencies
- Mutual library dependencies ignored by compilation – problem with 3rd party libraries
- Resulting issues
  - Library inconsistencies manifest at runtime
  - Compile time checks powerless

Motivating Example

Common JEE technological stack:



Source code compiles well against its direct libraries

But transitive library dependencies cause inconsistent application

→ Discovered at runtime! **E.g. NoSuchMethodException**

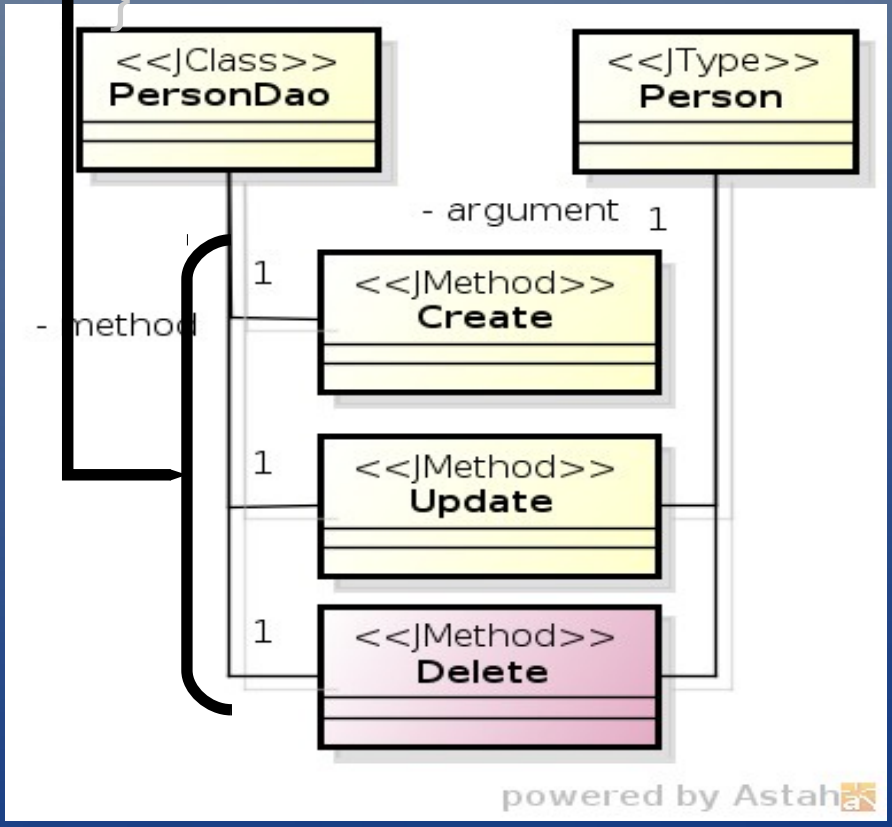
Approach: Reconstruction and verification of dependencies

- Libraries examined
  - Mutual dependencies reconstructed
  - Consistency verified
- Typed-based approach
  - No runtime behavior, workflow, ...
  - Motivated by statically typed compilers
- Inconsistent dependencies discovered
  - As they would be found by compilation

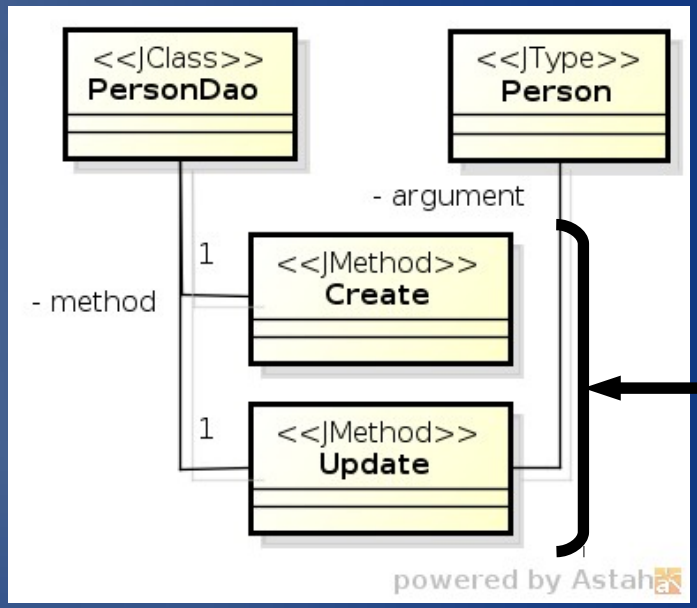
Dependencies Reconstruction

Library API:

```
public interface PersonDao {
    Person create();
    void update(Person person);
    void delete(Person person);
}
```



Library use:

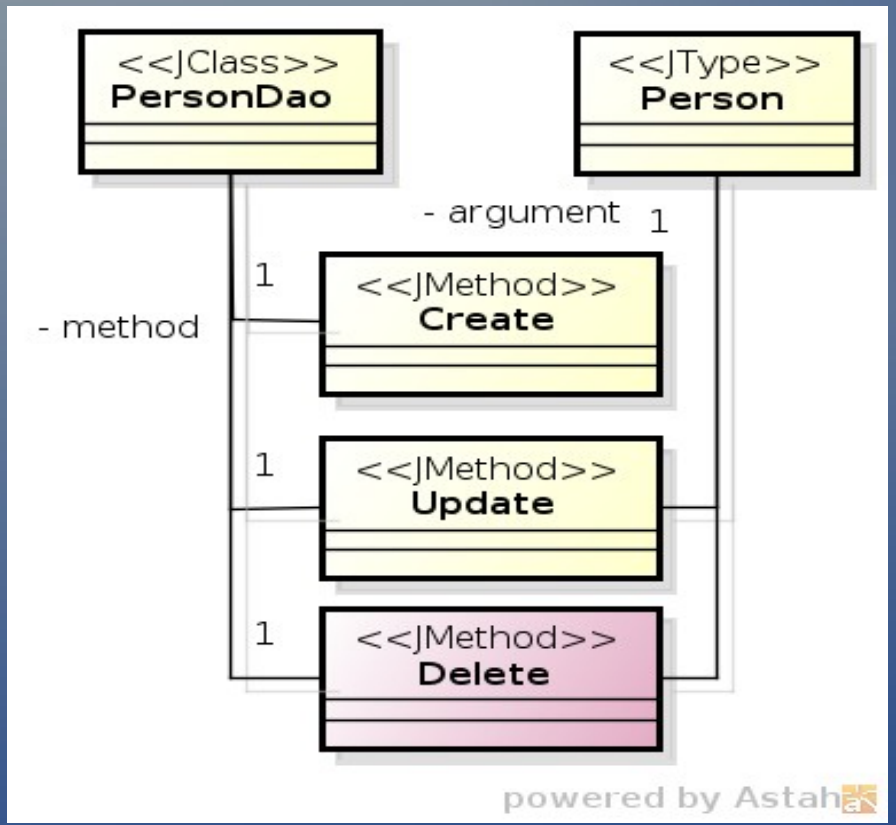



```
public class PersonService {
    private PersonDao personDao;

    public void newCustomer(String id) {
        Person p = personDao.create();
        personDao.update(p);
    }
```

Dependencies Verification

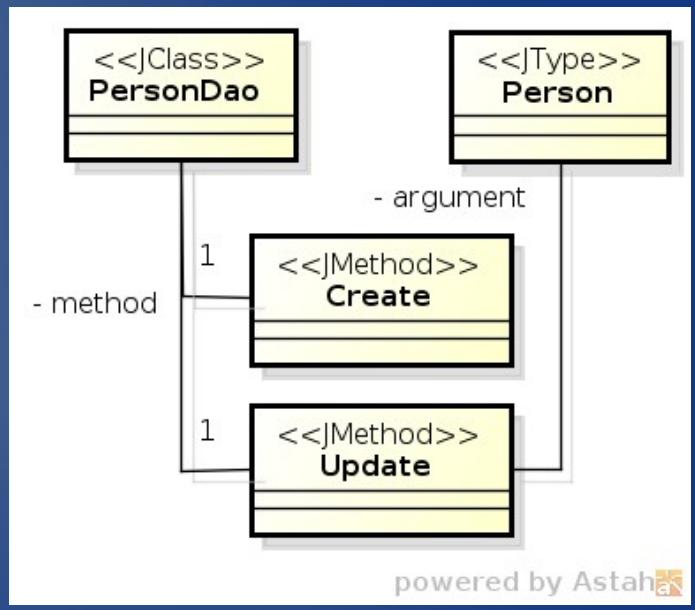
Library API:



?

$\neq$ VI $\vee$ V $\vee$ IV $<$

Library use:



Incompatible

ASM 2.2 ---------- CGlib

java.lang.NoSuchMethodError: org.objectweb.asm.ClassWriter.<init>(Z)V

Implementation

- Java byte-code analyser
  - Based on ASM
- Model of Java elements (class, method, field,...)
  - Inspired by Java Reflection API
- Model element comparators

- Published as open-source

Conclusion

- Approach to reconstruct and verify library mutual dependencies
- Implemented for Java byte-code
- Future work: integration to development tools
  - Eclipse IDE
  - Maven

Thank you

- Projects available at

  JaCC: https://www.assembla.com/spaces/jacc/
  OBCC: https://www.assembla.com/spaces/obcc/

Contact us

Kamil Jezek, Lukas Holy, Premek Brada

{kjezek,lholy,brada}@kiv.zcu.cz