A Safe Regression Test Selection Technique for Modelica

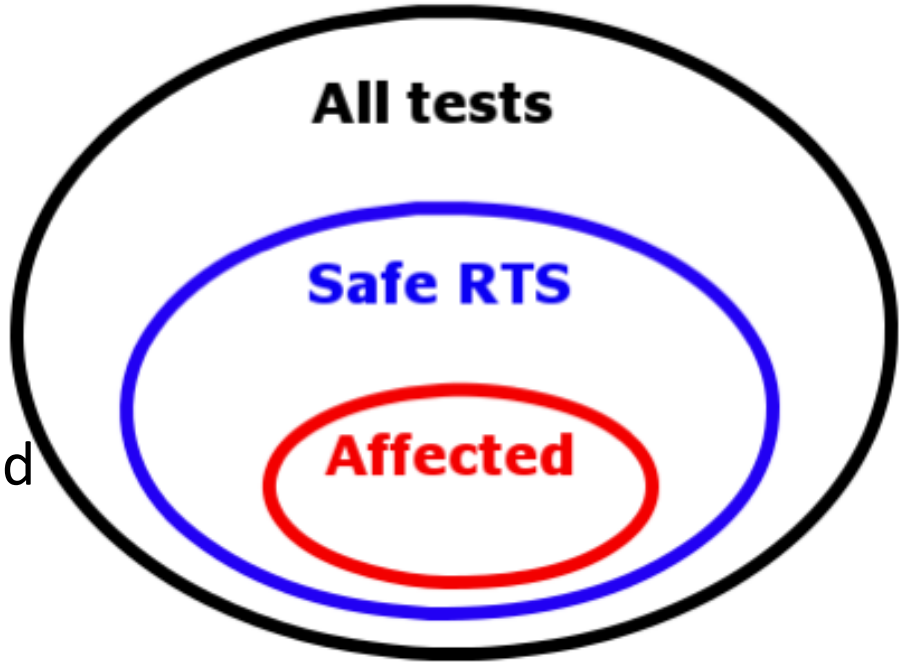
Niklas Fors, Lund University, Sweden

Jon Sten, Markus Olsson, Filip Stenström, Modelon, Sweden

American Modelica Conference 2018, 10 October

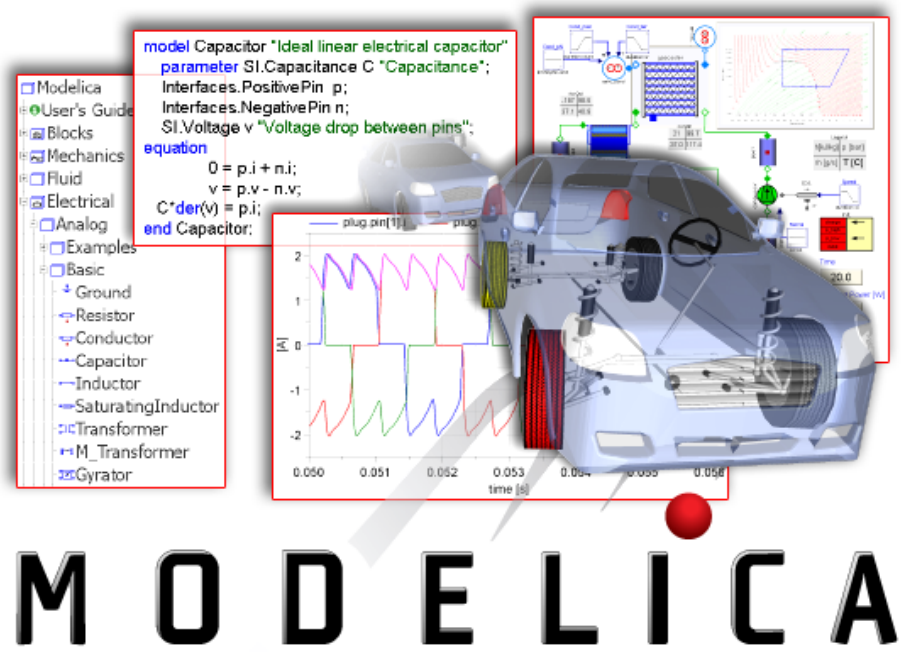
Regression Testing

- **Regression Testing**
  - Re-run tests to ensure that previous functionality still work **after a change**
- **Regression Test Selection (RTS)**
  - Run a subset of all tests
- **Safe RTS**
  - Run all **affected tests**
  - Conservative ⇒ might run unaffected tests
  - Tradeoff between analysis time and time saved



RTS for Modelica

- Motivation: testing Modelica Standard Library takes 2-3 hours
- We have implemented a **Safe RTS** technique for Modelica
- Static **dependency analysis** between classes, defined as **dependency rules**



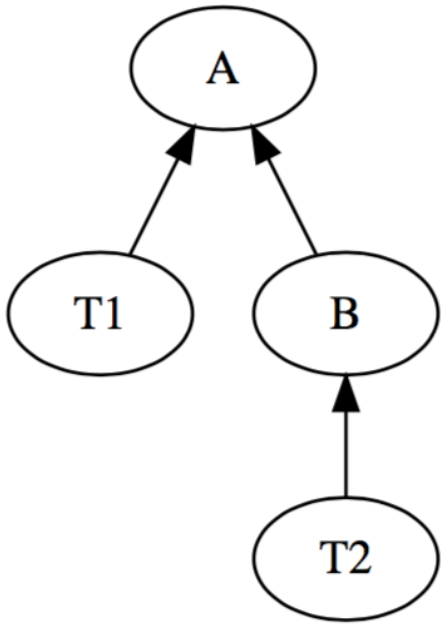
Master's theses

This work is based on two master’s theses:

- *Improved precision and verification for test selection in Modelica*
  by Markus Olsson and Filip Stenström
- *Safe test selection for modelica using static analysis*
  by Erik Hedblom and Kasper Rundquist

Dependency Analysis

```
model A
  ...
end A;

model B
  A a;
end B;

model T1
  A a;
end T1;

model T2
  B b;
end T2;
```

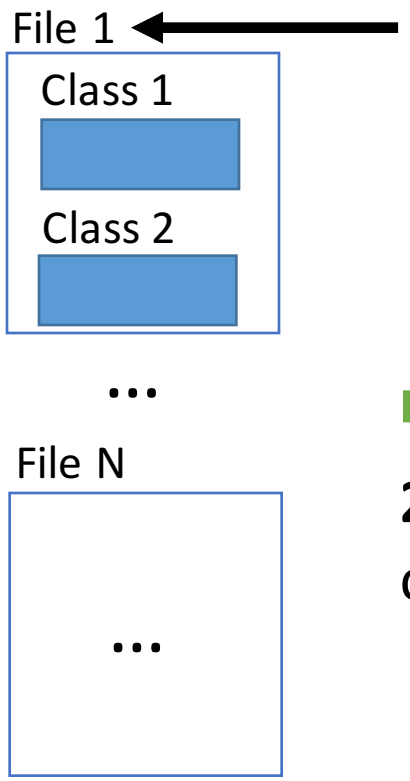


We analyze dependencies between classes to select test classes that need to run given a change.

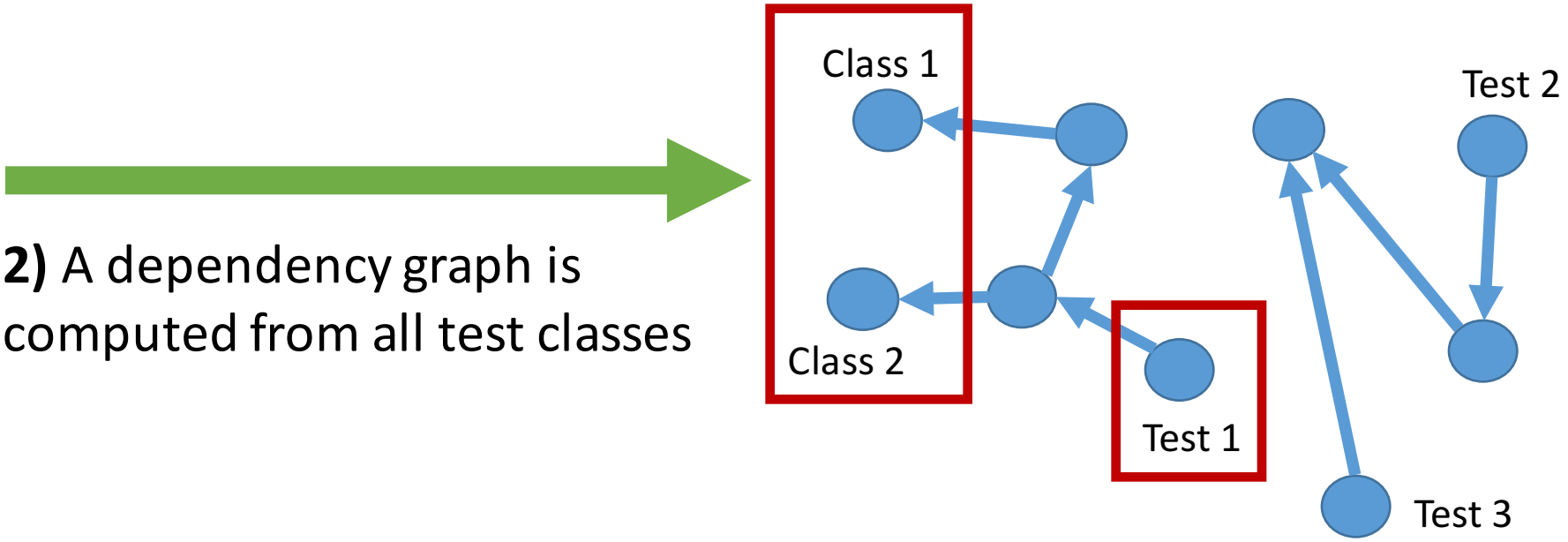
Examples:

- If **A** is changed ⇒ run **T1** and **T2**
- If **B** is changed ⇒ run **T2**

Overview - Changing A File



1) **File 1** is changed, all classes in the file are considered changed.



2) A dependency graph is computed from all test classes

3) All test classes that reach one of the changed classes need to be run (Test 1 in this example)

Dependency Rules

Rule 1: *A class has a dependency on an **accessed class**, including all parts of the qualified name.*

Rule 2: *A class has a dependency on its **enclosing class**.*

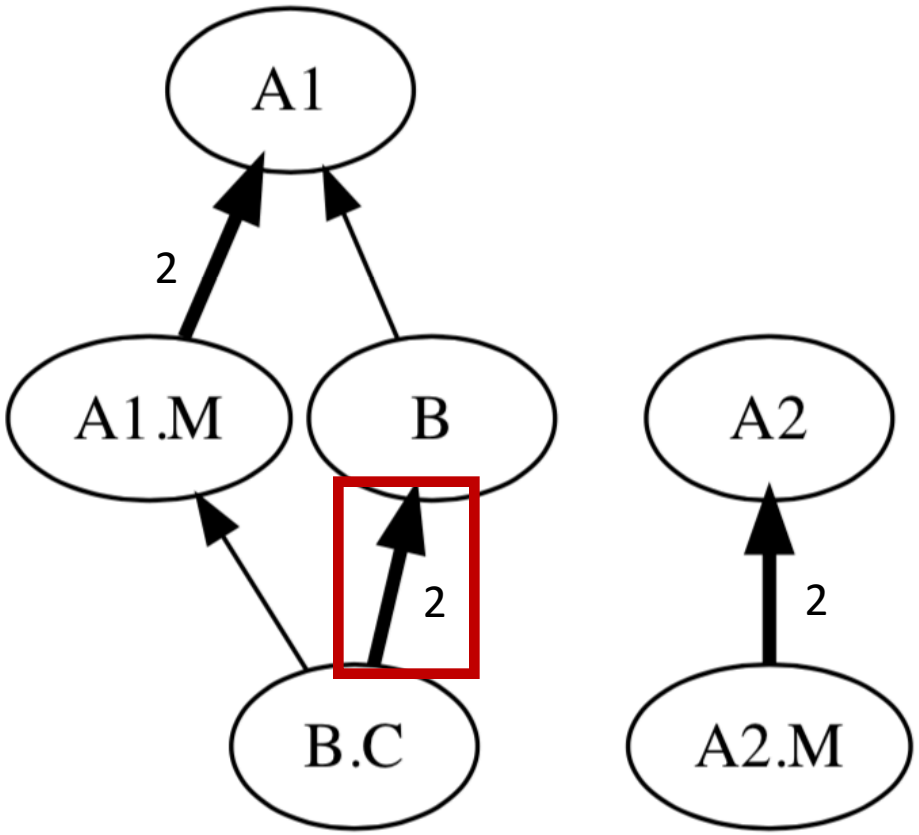
Rule 3: *A class that contains a **redeclaration** depends on all super classes and enclosed classes of the replacing class (and all their enclosed classes and super classes recursively).*

Rule 4: *A class has a dependency on **implicitly called classes**. This includes a record or type enclosing a function named **equalityConstraint**, and a class extending the class **ExternalObject** has dependency on enclosed function destructor.*

Rule 2: *A class has a dependency on its **enclosing class**.*

```
package A1
  model M
  end M;
end A1;

package A2
  model M
  end M;
end A2;

package B
  extends A1;

  model C
    M m;
  end C;
end P;
```

uses

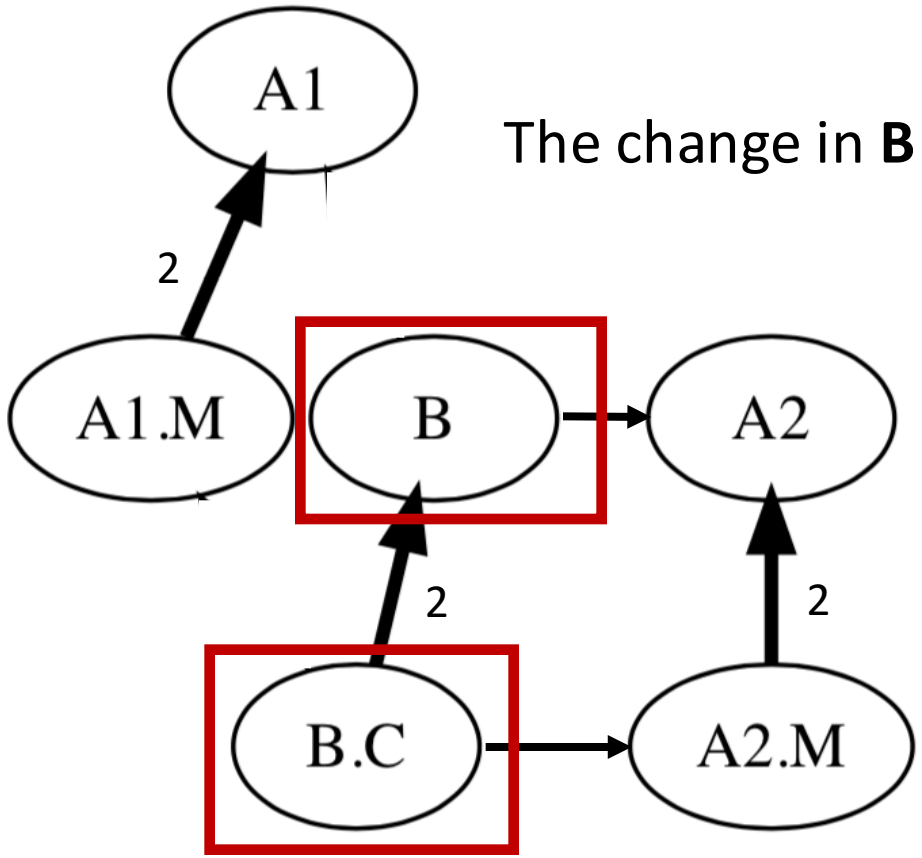


8

Rule 2: *A class has a dependency on its **enclosing class**.*

```
package A1
  model M
  end M;
end A1;

package A2
  model M
  end M;
end A2;

package B
  extends A1;
          A2;
  model C
    M m;
  end C;
end P;
```

uses



The change in **B** affects the meaning of **C**!

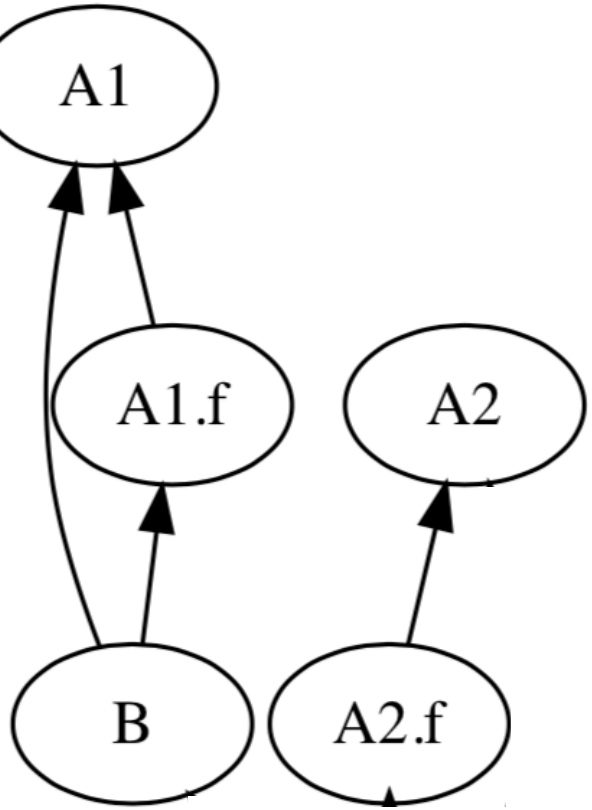
9

Rule 3:

*A class that contains a **redeclaration** depends on all super classes and enclosed classes of the replacing class (and all their enclosed classes and super classes recursively).*

```
package A1
  function f
  end f;
end A1;

package A2
  function f
  end f;
end A2;

model B
  replaceable package P = A1;
  Real x = P.f();
end B;
```

uses



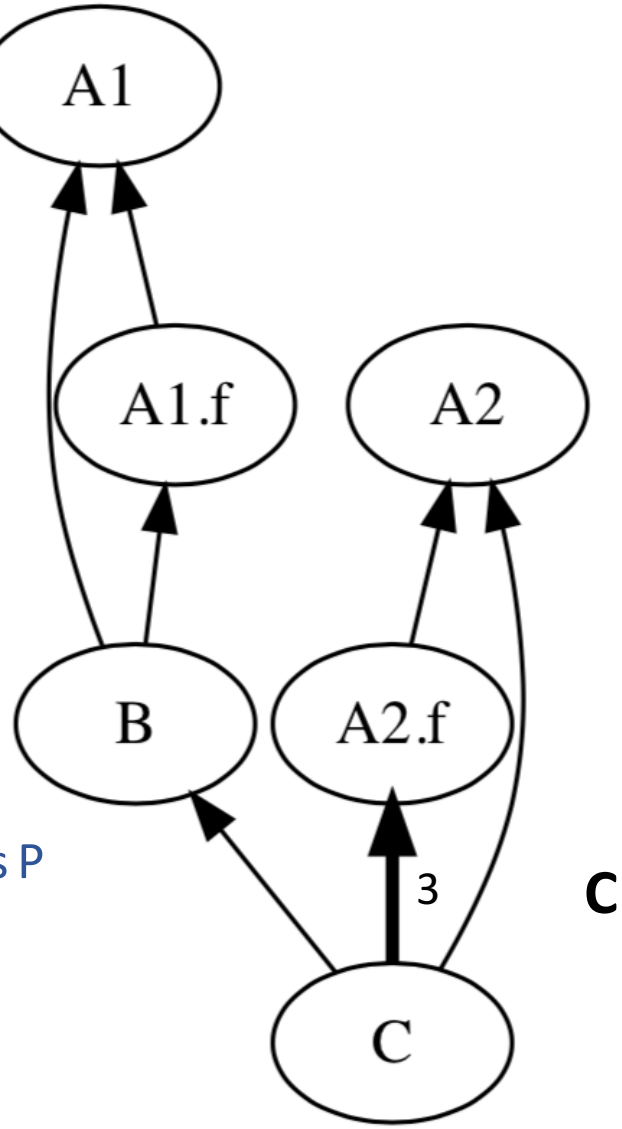
Rule 3:

*A class that contains a **redeclaration** depends on all super classes and enclosed classes of the replacing class (and all their enclosed classes and super classes recursively).*

```
package A1
  function f
  end f;
end A1;

package A2
  function f
  end f;
end A2;

model B
  replaceable package P = A1;
  Real x = P.f();
end B;

model C
  B b(redeclare package P = A2)
end C;
```

in context of C

redeclares P to A2



C depends on **A2.f**

Implementation

- Dependency analysis implemented in the OPTIMICA Compiler Toolkit by Modelon
- 201 source lines of code (JastAdd code)

Performance Results

Library	**Avg. testing runtime saved / changed class**	**Avg. testing runtime saved / changed file**	**Dependency analysis**
Modelica Standard Library	95.5%	88.9%	0.04%
Heat Exchanger Library	78.9%	80.5%	0.1%

Verification

Is our RTS technique safe?

- Mutation testing! On MSL.

Normal mutation testing	Our mutation testing
How good is the test suite?	Does the dependency analysis find all dependencies? Compute actual dependencies and compare to our RTS technique

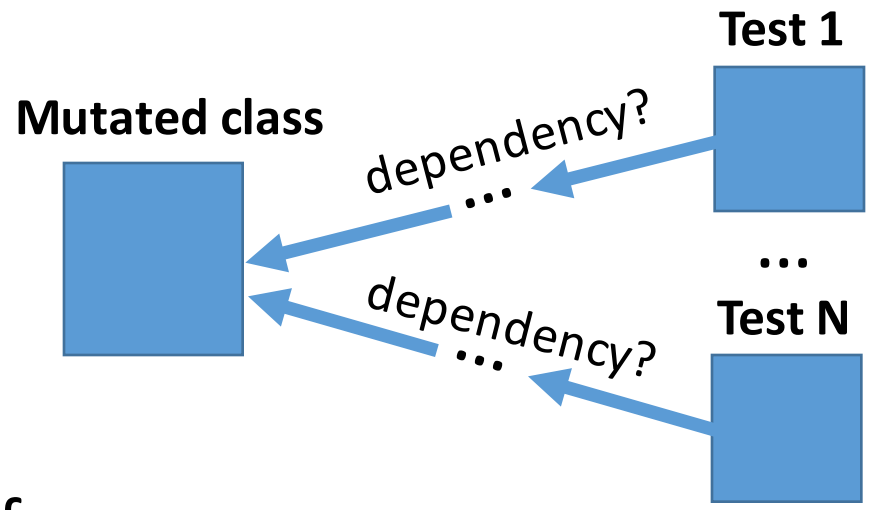
Examples of Mutations

Type	**Before**	**After**
Literal	39	40
Arithmetic	1 + 2	2 + 1
Logical	f() > 0	f() <= 0
Comment	M m “comment”;	M m “mutated”;

Mutating Classes to Find Dependencies

Is there a dependency from
test class to mutated class?

If yes, then our analysis
should also find that!



Otherwise, our technique is not safe:

- Dependency rule missing or
- Implementation bug

Background: Modelica Flattening

Flattening:

- Removes class and component hierarchy
  $\Rightarrow$ one equation system
- Compilation step before simulation

source code

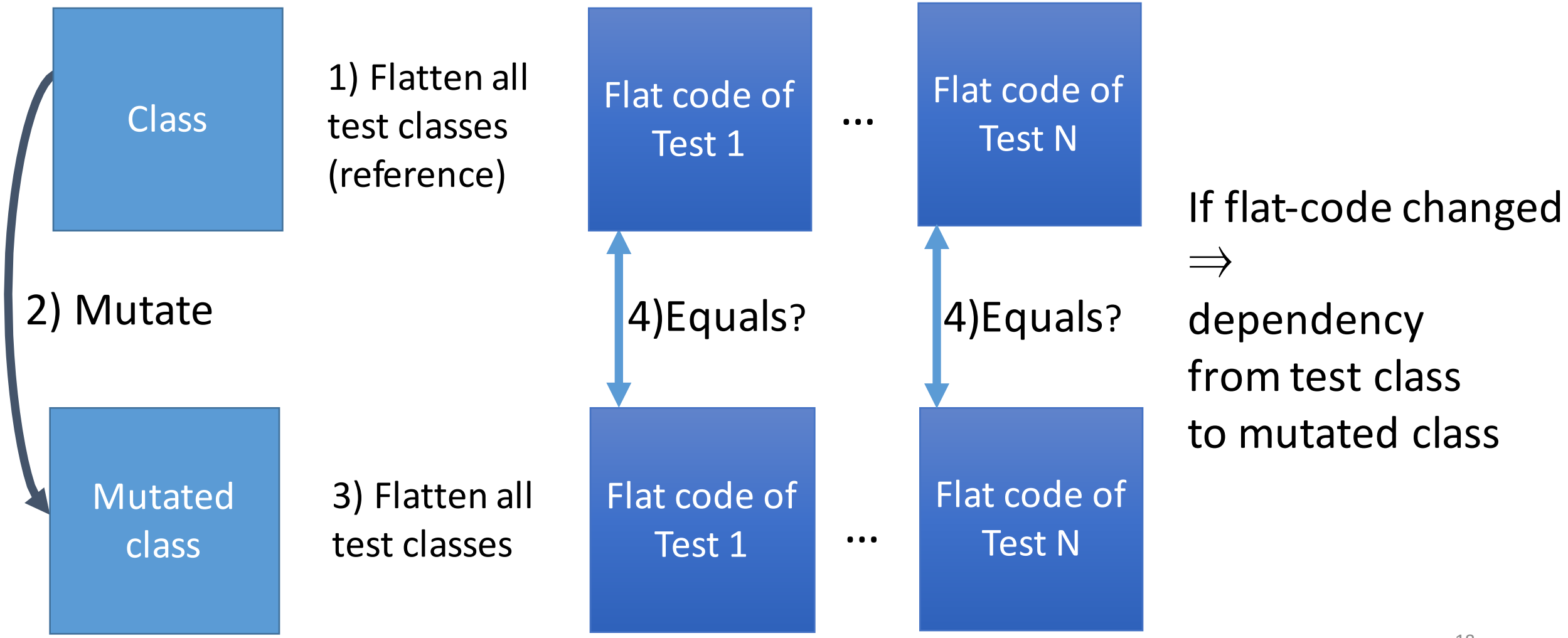
```
model A
   B b;
   Real x;
equation
   x = b.y + 1;
end A;

model B
   Real y;
   Real z;
equation
   y = z;
   z = time;
end B;
```

flatten(A)

flat-code

```
fclass A
 Real b.y;
 Real x;
equation
 x = b.y + 1;
 b.y = time;
end A;
```

Verification: Method



Verification Results

- Results
  - 6 implementation bugs found
  - Rule 3 generalized
  - New Rule 4
- Previous technique not safe *(by Hedblom and Rundquist)*
- Total execution time: 280 days
  - Run on Jenkins cluster
  - Important with good mutations

Verification Results

Mutated classes found dependencies to	2345 (39.4%)
Classes attempted to mutate	4587 (77.1%)
Classes in MSL	5946

Open Source Test Suite for Modelica RTS

modelon / MCDTS

Watch 5 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Insights

https://github.com/modelon/MCDTS

5 commits | 1 branch | 0 releases | 2 contributors | MIT

Branch: master | New pull request | Find file | Clone or download

Markus-O Added License — Latest commit b4a13c0 on 25 May

msl_dependencies	Added the database of MSL test dependencies	5 months ago
testfiles	Added test case	5 months ago
LICENSE	MIT LICENSE	5 months ago
README.md	Added the database of MSL test dependencies	5 months ago

Conclusions

- Regression test selection technique for Modelica
  - Implementation-independent dependency rules
  - Savings: MSL: 96%, HXL: 79%
  - Safety verified using mutation testing. Not 100%
  - Open source test suite
- Future Work
  - Code instrumentation
  - More mutation testing