Source-Code Level Regression Test Selection: The Model-Driven Way

Thibault Béziers la Fosse, Jean-Marie Mottu
Massimo Tisi, Gerson Sunyé

ECMFA'19 • 07.18.19

Regression Testing

Type of software testing ensuring that **recent** program changes do not break **existing** features.

Regression Testing

50%

of software maintenance cost is dedicated to **software testing**

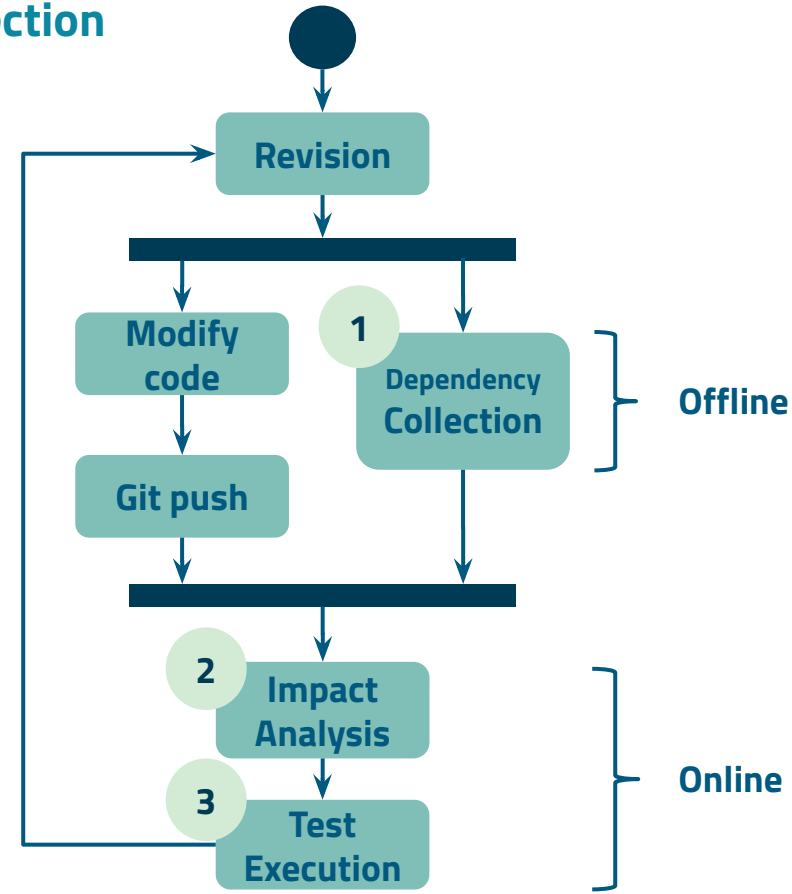
80%

of software testing cost is dedicated to **regression testing**

Regression Test Selection

- Running all tests **takes time**.
- **Reducing this duration** improves productivity.
- Running only the test cases **impacted by code changes** instead of all tests is the purpose of Regression Test Selection (RTS).

Regression Test Selection Approach



Properties of RTS

- Safety: A RTS approach is considered as **safe** if **all** the test cases impacted by changes are **selected**.
- Precision: A RTS approach is considered as **precise** if **only** test cases impacted by changes are **selected**.

Granularity of RTS

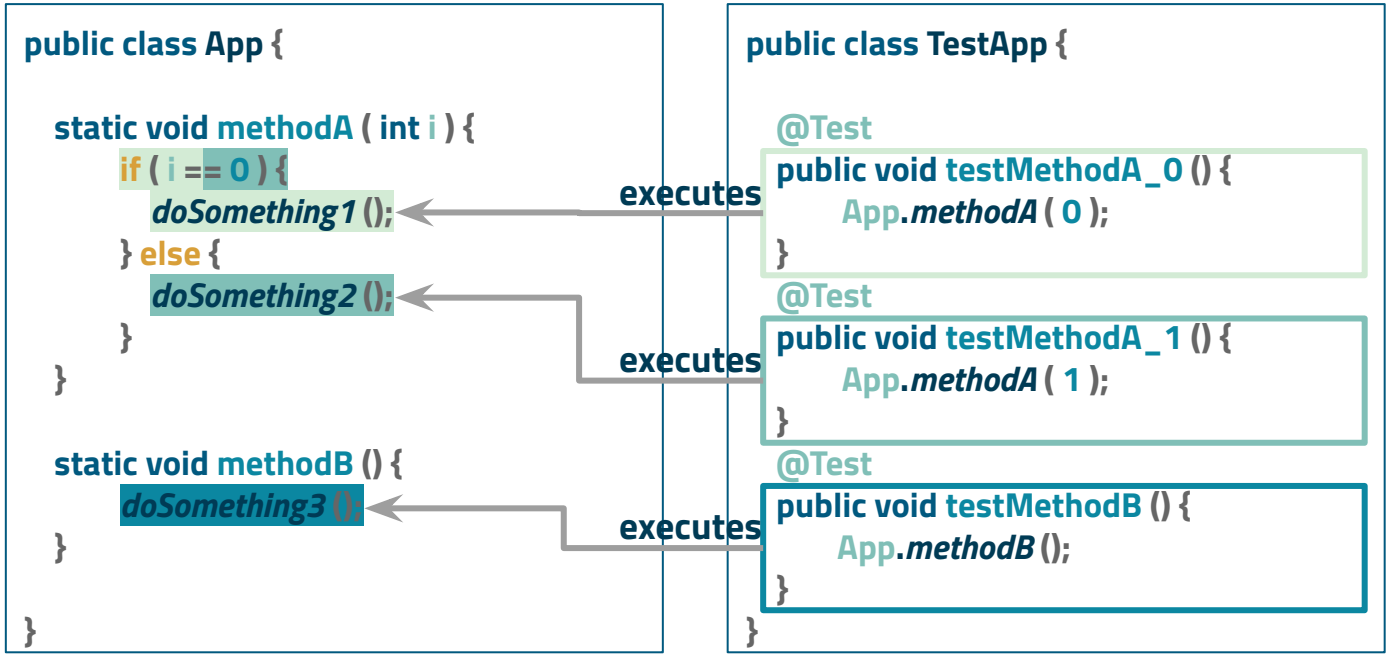
- Defines the granularity at which **code changes** are considered.
- A statement-grained RTS computes the impacts of **statement** modifications.
- Fine-grained RTS is **slower** but selects **less tests** to execute (i.e., more **precise**).

Regression Test Selection: Example

```
public class App {

   static void methodA ( int i ) {
         if ( i == 0 ) {
            doSomething1 ();
         } else {
            doSomething2 ();
         }
   }

   static void methodB () {
         doSomething3 ();
   }

}
```

```
public class TestApp {

   @Test
   public void testMethodA_0 () {
         App.methodA ( 0 );
   }
   @Test
   public void testMethodA_1 () {
         App.methodA ( 1 );
   }
   @Test
   public void testMethodB () {
         App.methodB ();
   }
}
```

Regression Test Selection: Collection



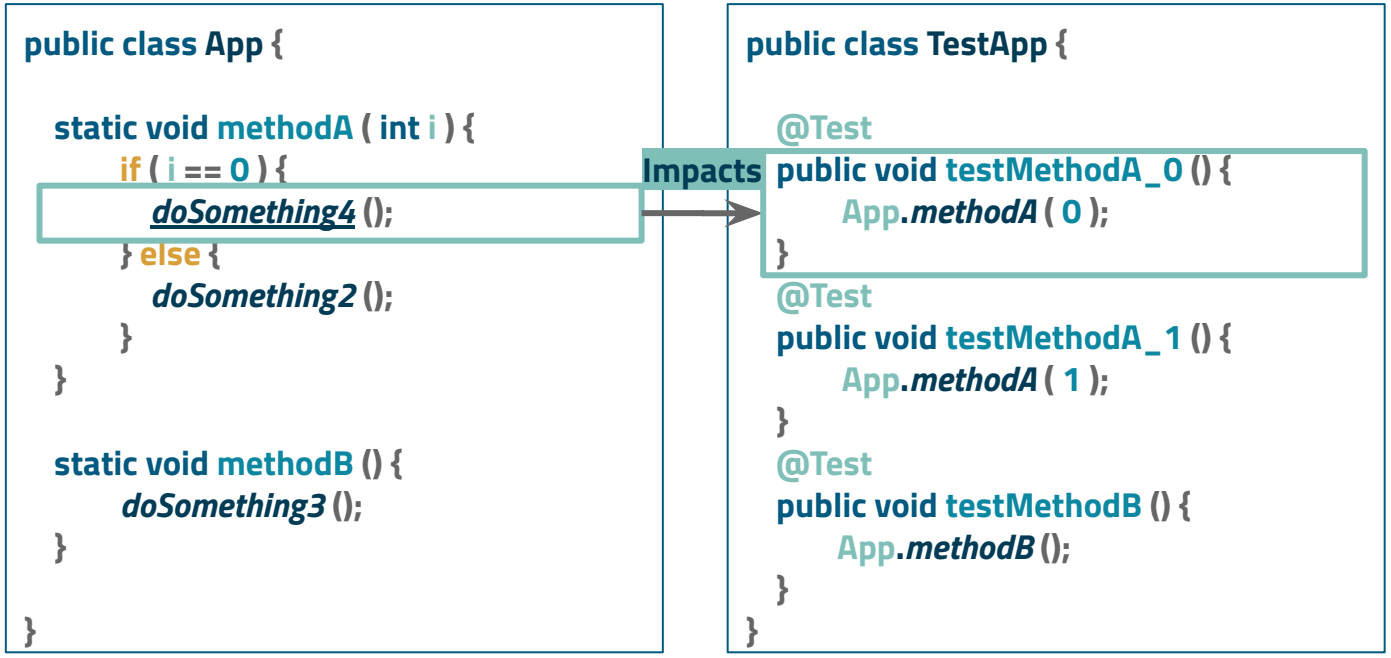
Regression Test Selection: Analysis

```
public class App {

   static void methodA ( int i ) {
        if ( i == 0 ) {
           doSomething4 ();
        } else {
           doSomething2 ();
        }
   }
```

```
--- a/App.java
+++ b/App.java
@@ -2,7 +2,7 @@ public class App {
- doSomething1 ();
+ doSomething4 ();
```

```
public class TestApp {

   @Test
   public void testMethodA_0 () {
        App.methodA ( 0 );
   }
   @Test
   public void testMethodA_1 () {
        App.methodA ( 1 );
   }
   @Test
   public void testMethodB () {
        App.methodB ();
   }
}
```

Regression Test Selection: Analysis



Regression Test Selection: Execution

```
public class App {

   static void methodA ( int i ) {
         if ( i == 0 ) {
            doSomething4 ();
         } else {
            doSomething2 ();
         }
   }

   static void methodB () {
         doSomething3 ();
   }

}
```

```
public class TestApp {

   @Test                                   Execution
   public void testMethodA_0 () {
         App.methodA ( 0 );
   }
   @Test
   public void testMethodA_1 () {
         App.methodA ( 1 );
   }
   @Test
   public void testMethodB () {
         App.methodB ();
   }
}
```

Model-Driven Regression Test Selection

1- RTS approaches focus on precision and/or performances, not on **reusability** & **flexibility**.

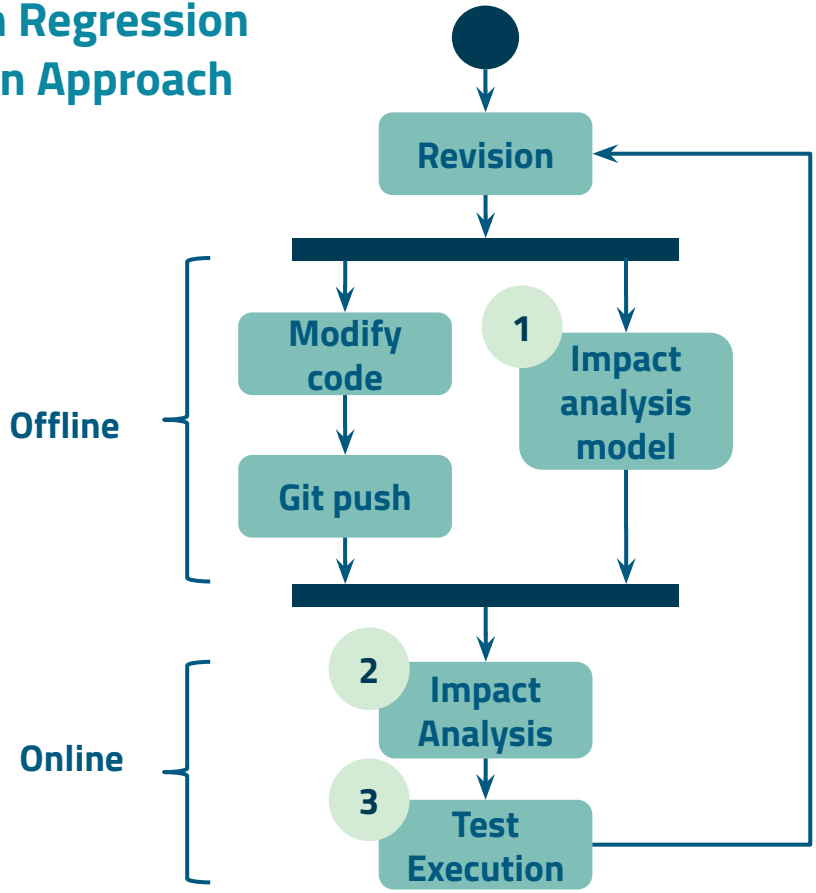
2- Using MDE for RTS would allow a **configurable** approach with variable **precision** and better **reusability**.

3- An **impact analysis model** produced for RTS can be reused for other purposes.

Model-Driven RTS Requirements

1. Safety
2. Precision
3. Customizability
4. Reusability
5. Analysis & Execution phases have to be **faster** than running all tests.

Model-Driven Regression Test Selection Approach

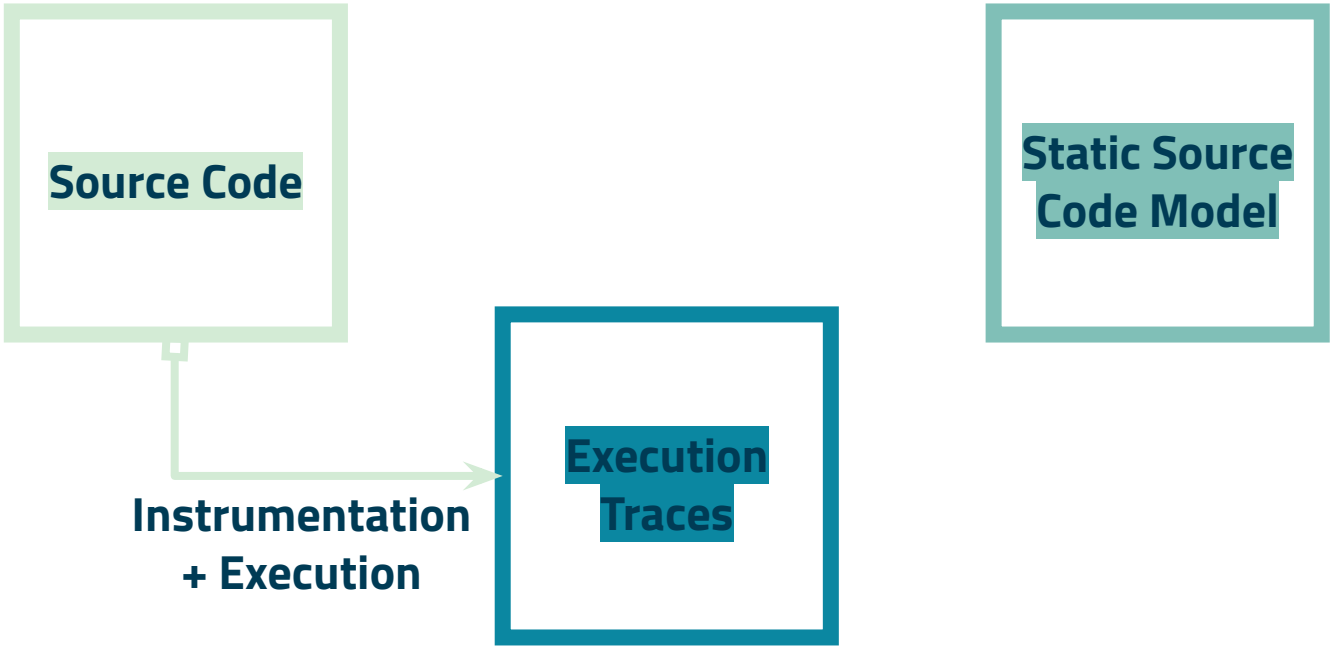


I- Building an Impact Analysis Model

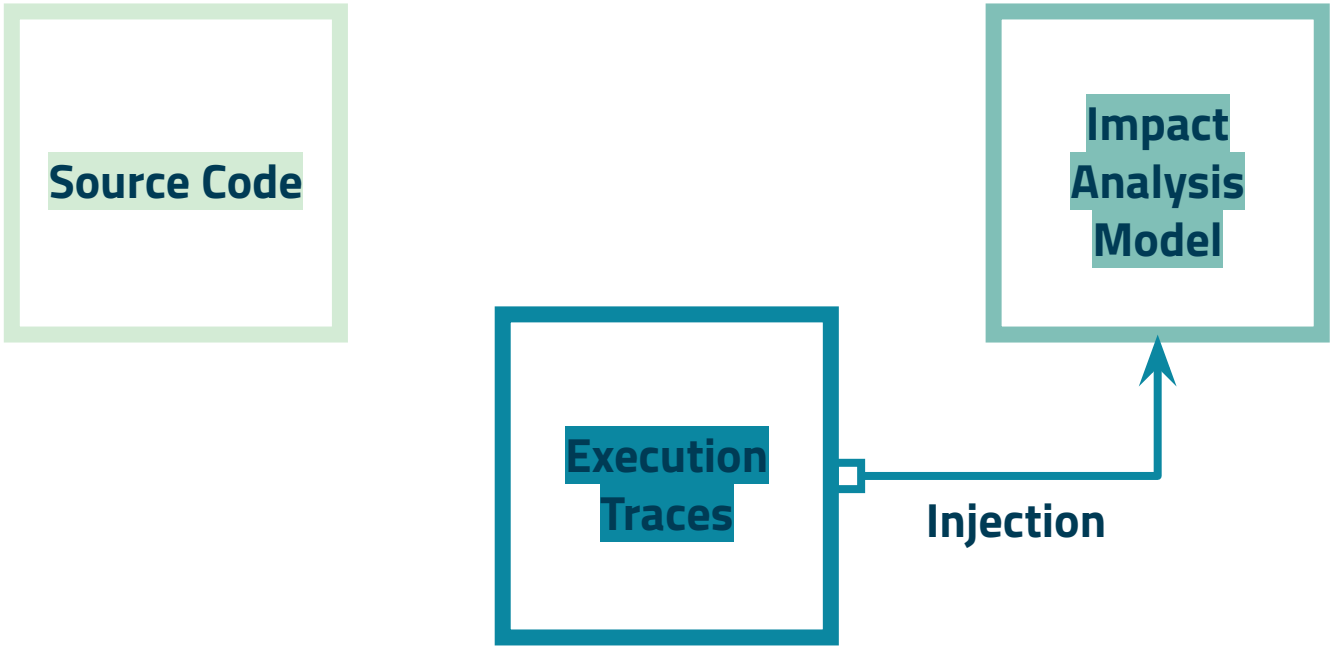
Source Code

I- Building an Impact Analysis Model

I- Building an Impact Analysis Model



I- Building an Impact Analysis Model

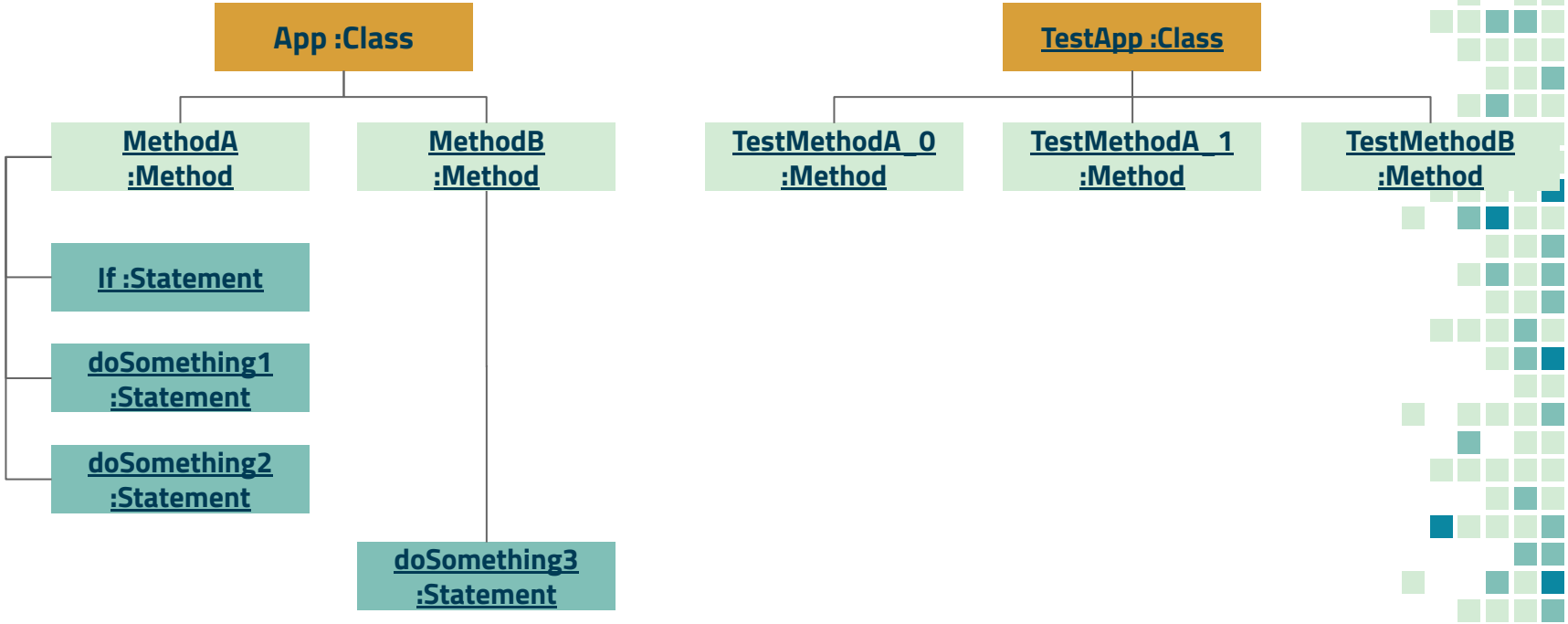


I- Building an Impact Analysis Model

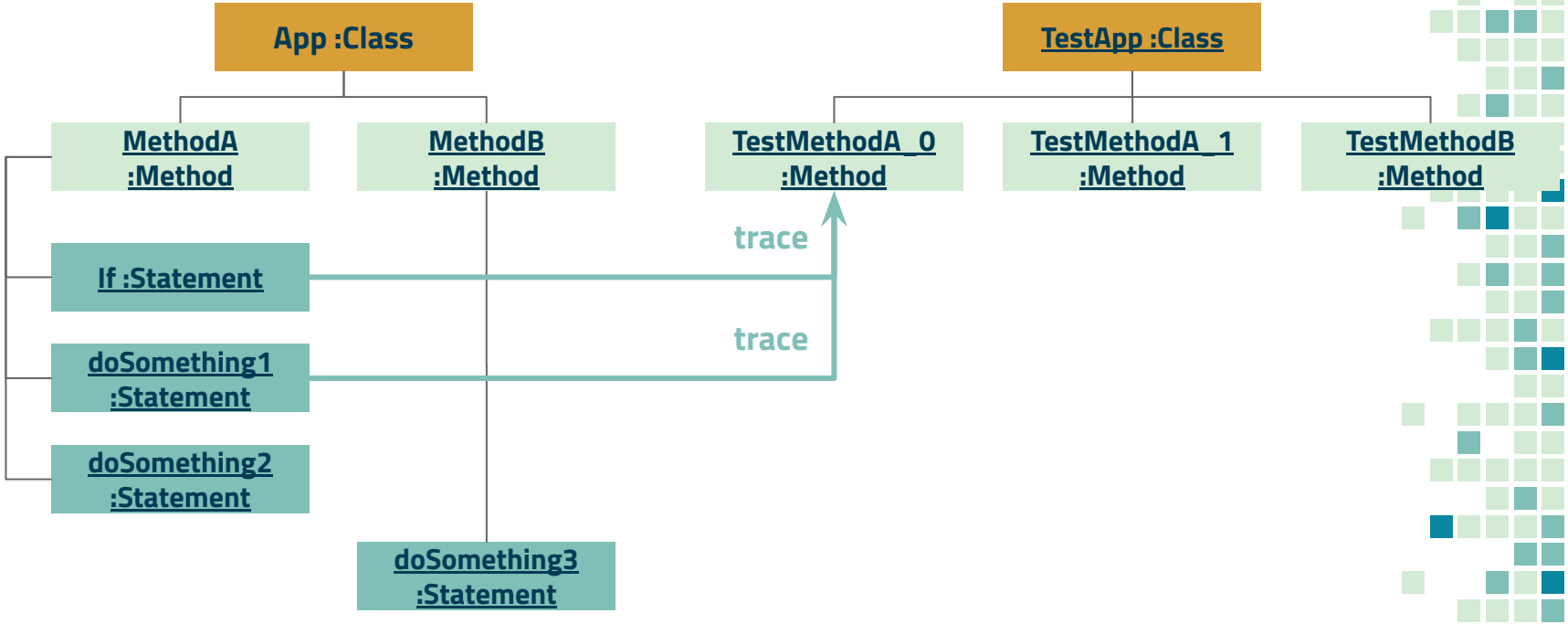
```
public class App {

   static void methodA ( int i ) {
         if ( i == 0 ) {
            doSomething1 ();
         } else {
            doSomething2 ();
         }
   }

   static void methodB () {
         doSomething3 ();
   }

}
```

```
public class TestApp {

   @Test
   public void testMethodA_0 () {
         App.methodA ( 0 );
   }
   @Test
   public void testMethodA_1 () {
         App.methodA ( 1 );
   }
   @Test
   public void testMethodB () {
         App.methodB ();
   }
}
```

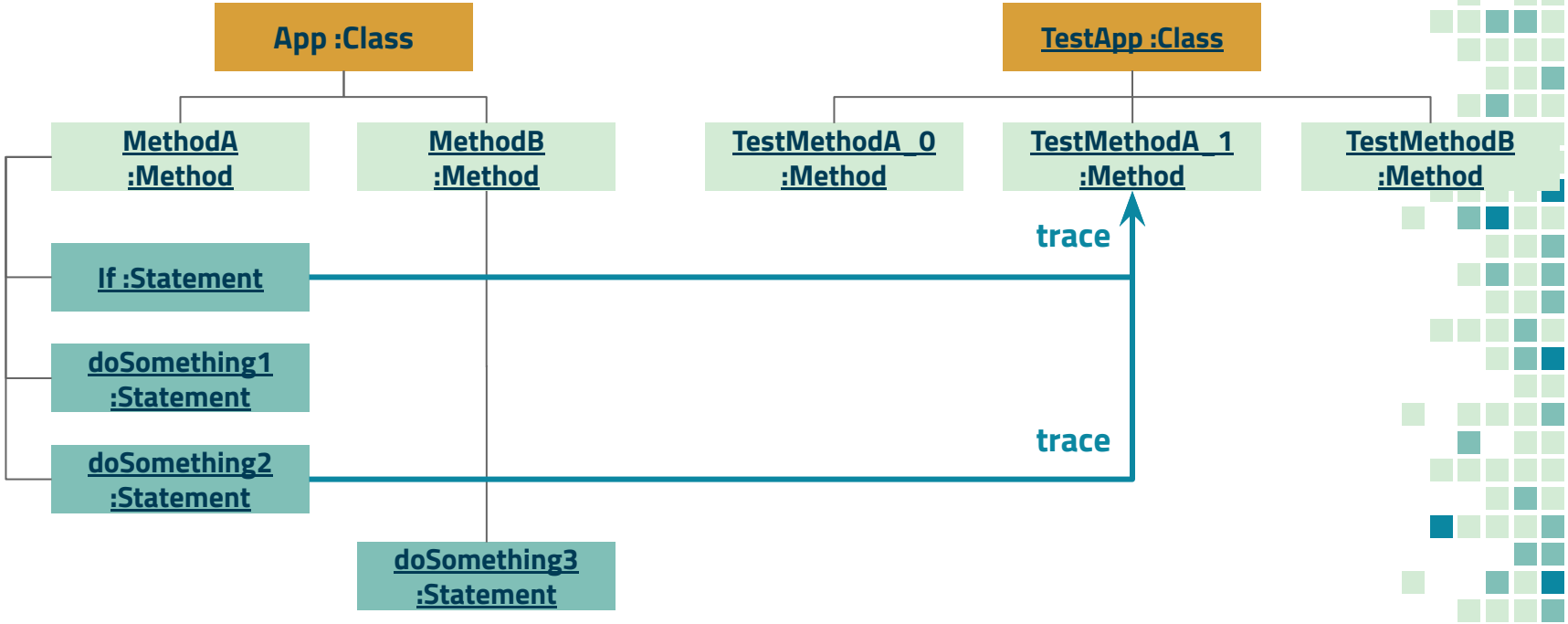
I- Building an Impact Analysis Model



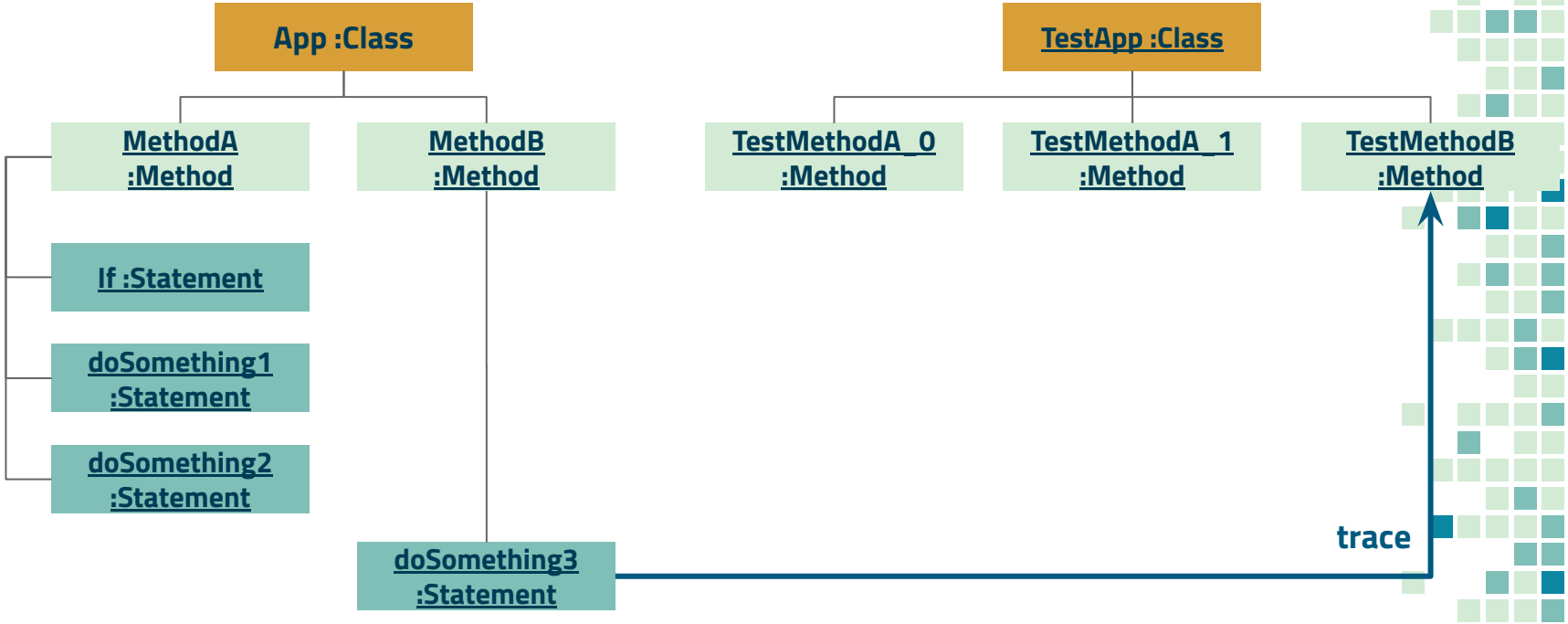
I- Building an Impact Analysis Model



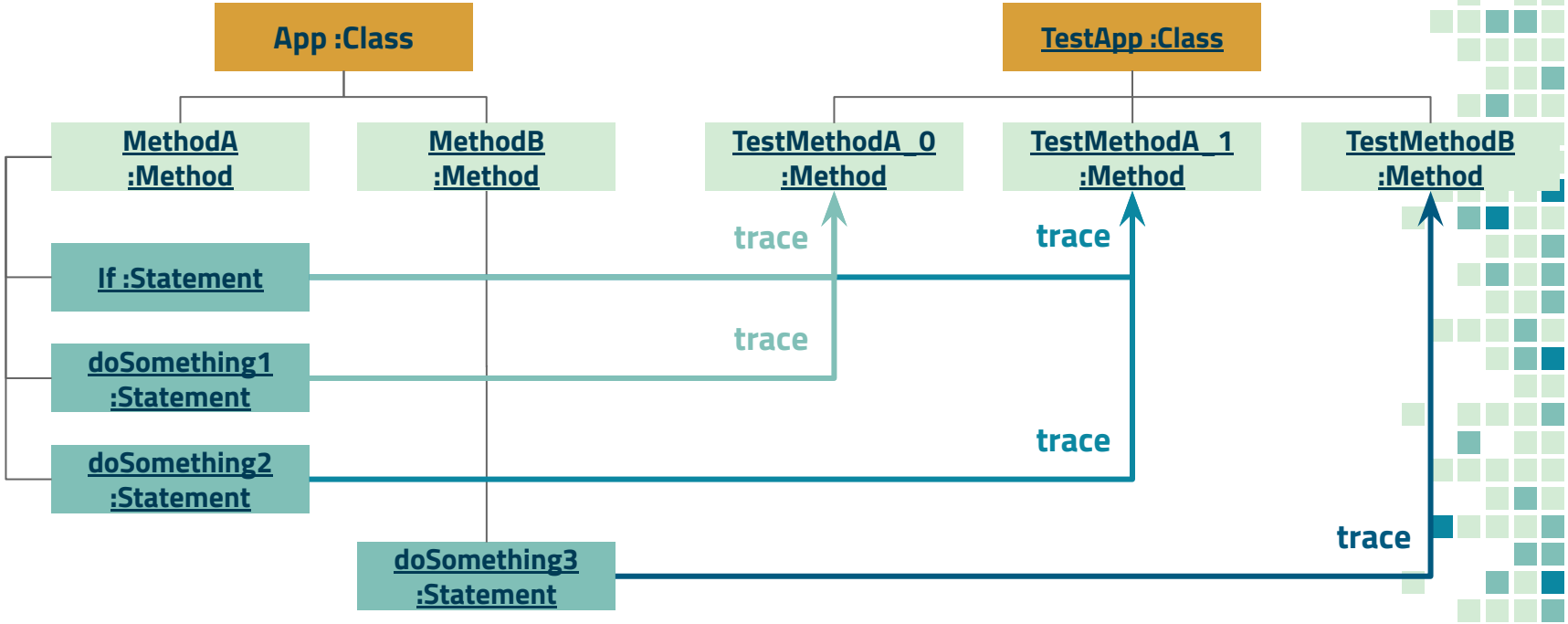
I- Building an Impact Analysis Model



I- Building an Impact Analysis Model



I- Building an Impact Analysis Model



II- Change Impact Analysis

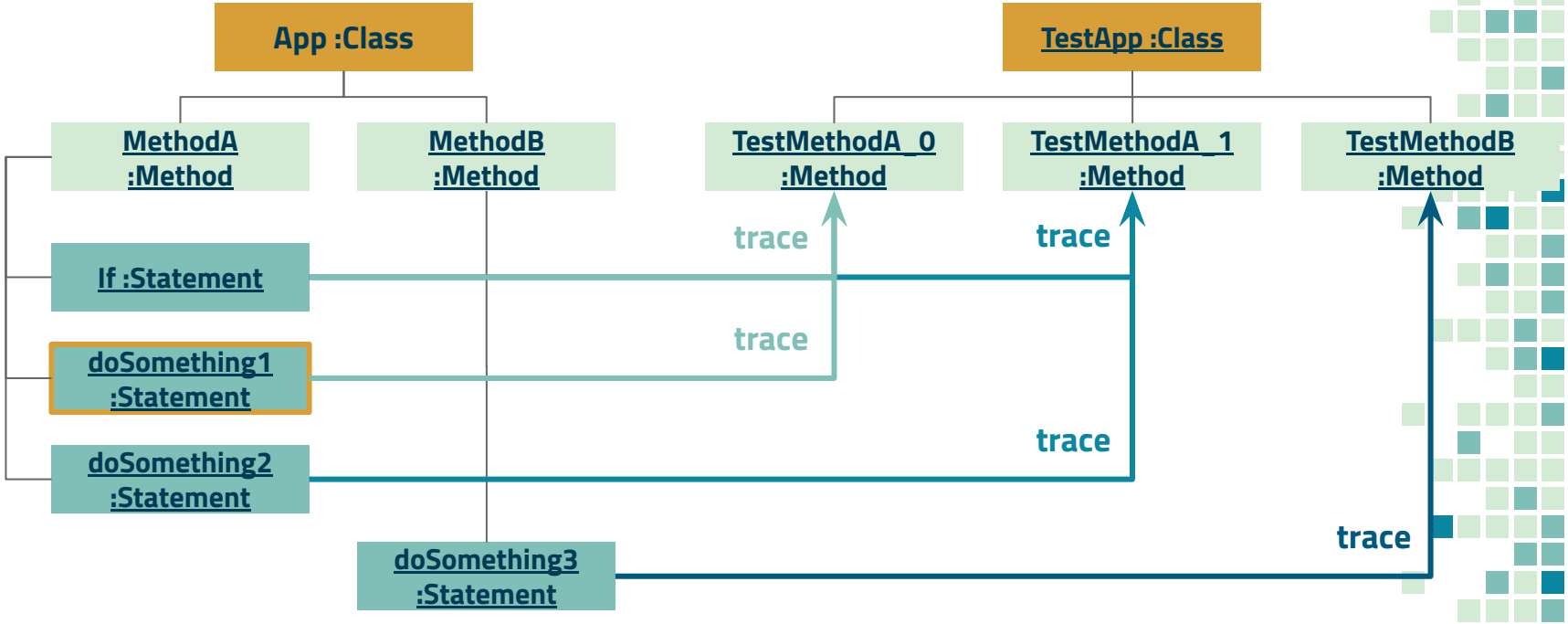
```
public class App {

   static void methodA ( int i ) {
         if ( i == 0 ) {
            doSomething1 ();
         } else {
            doSomething2 ();
         }
   }

   static void methodB () {
         doSomething3 ();
   }

}
```

II- Change Impact Analysis

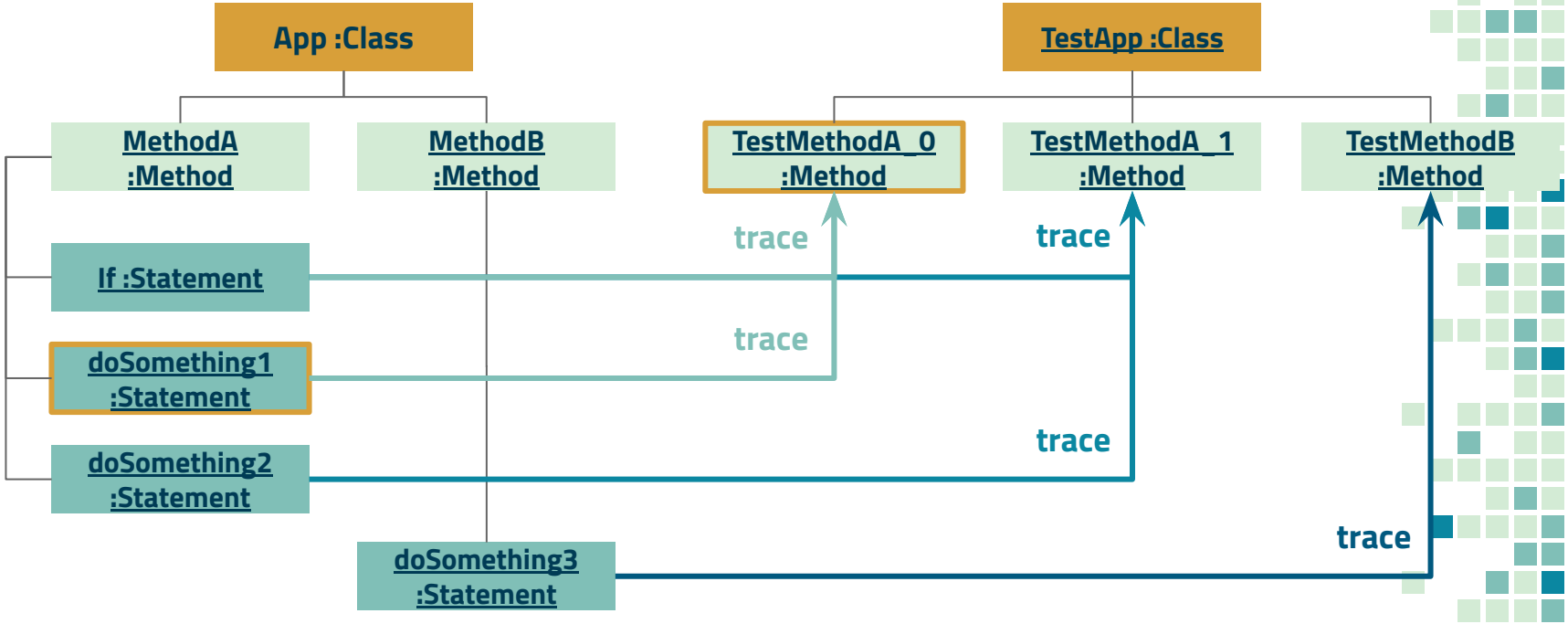
```
public class App {

   static void methodA ( int i ) {
         if ( i == 0 ) {
            doSomething4();
         } else {
            doSomething2 ();
         }
   }

   static void methodB () {
         doSomething3 ();
   }

}
```

Statement doSomething1 updated

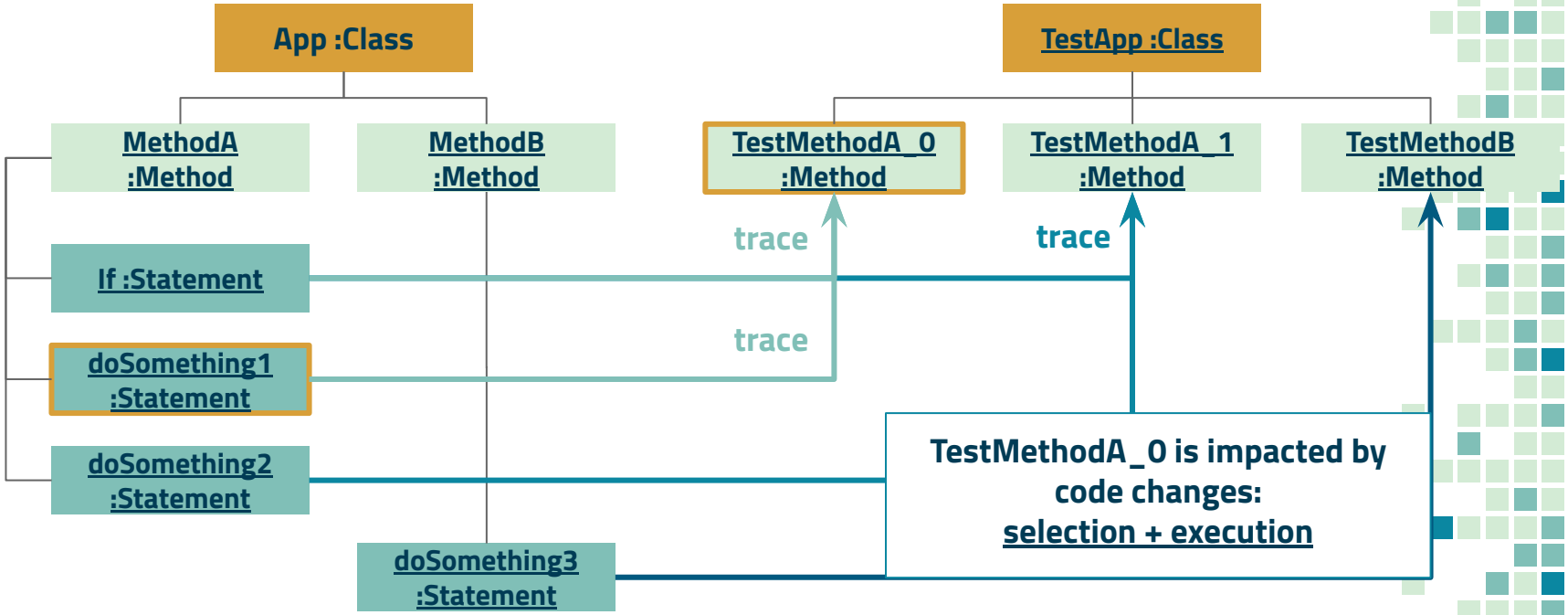
II- Change Impact Analysis



II- Change Impact Analysis



III- Impacted Test Case Execution

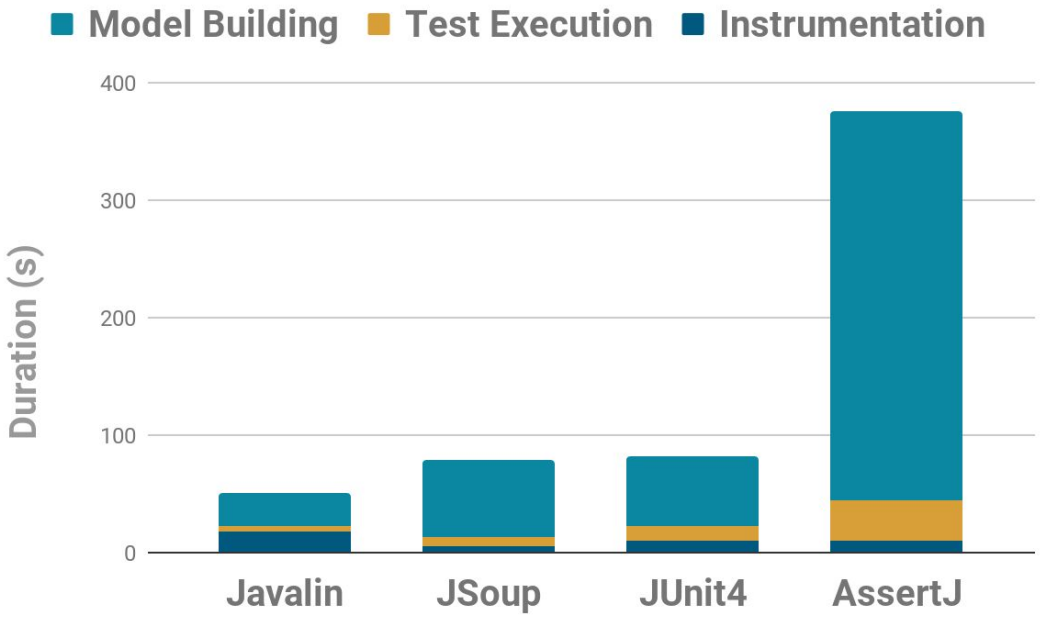


Evaluation

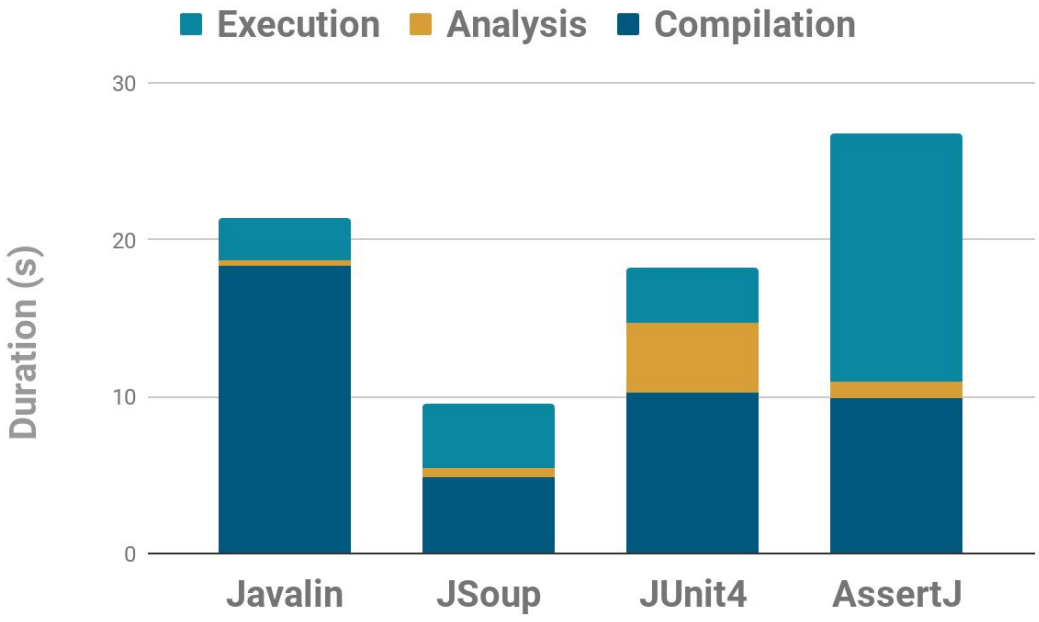
- Applied Model-Driven RTS on **100 commits** of popular open-source projects

Framework	**LOC**	**#Classes Tested**	**#Test Classes**
Javalin	2500	13	23
JSoup	25 000	61	31
JUnit4	100 000	126	174
AssertJ	250 000	368	1903

Offline phase

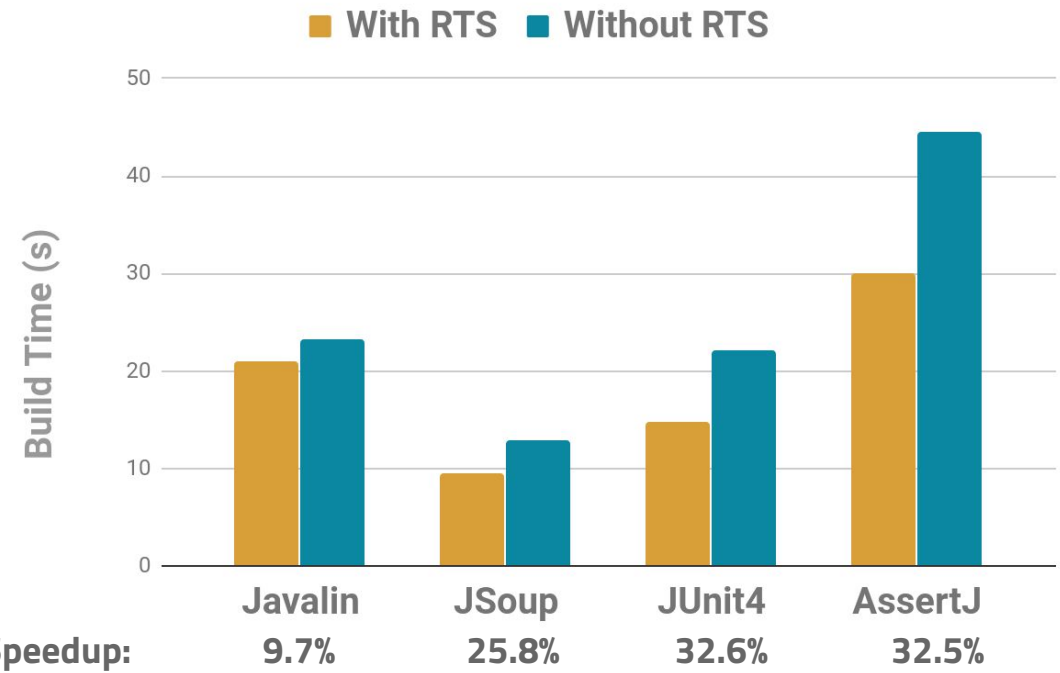


Online phase



Test cases selected:	Javalin	JSoup	JUnit4	AssertJ
	6.31%	34.01%	3.51%	18.24%

Results


With RTS
Without RTS
Build Time (s)
50
40
30
20
10
0
Javalin
JSoup
JUnit4
AssertJ
Speedup:
9.7%
25.8%
32.6%
32.5%

Complementary use of the Model

- Better **understanding** of test suites.
- Can provide an accurate overview of testing **code coverage**.
- Easy **coupling** with other models:
  - Association with Structured Metrics Metamodel (**SMM**) for energy modeling[1].

[1] Béziers la Fosse, T, Mottu, J. M., Tisi, M., & Sunyé, G. Characterizing a Source Code Model with Energy Measurements. *MeGSuS@ ESEM*. 2016.

Model-Driven RTS Requirements

1. Safety
2. Precision
3. Customizability
4. Reusability
5. Analysis & Execution phases have to be **faster** than running all tests.

Model-Driven RTS Requirements

1. Impact analysis ensures **all** impacted tests are selected.
2. Precision
3. Customizability
4. Reusability
5. Analysis & Execution phases have to be **faster** than running all tests.

Model-Driven RTS Requirements

1. Impact analysis ensures **all** impacted tests are selected.
2. **Statement-grained** RTS.
3. Customizability
4. Reusability
5. Analysis & Execution phases have to be **faster** than running all tests.

Model-Driven RTS Requirements

1. Impact analysis ensures **all** impacted tests are selected.
2. **Statement-grained** RTS.
3. MDE enables granularity **variation**.
4. Reusability
5. Analysis & Execution phases have to be **faster** than running all tests.

Model-Driven RTS Requirements

1. Impact analysis ensures **all** impacted tests are selected.
2. **Statement-grained** RTS.
3. MDE enables granularity **variation**.
4. MDE enables **coupling** with other models.
5. Analysis & Execution phases have to be **faster** than running all tests.

Model-Driven RTS Requirements

1. Impact analysis ensures **all** impacted tests are selected.
2. **Statement-grained** RTS.
3. MDE enables granularity **variation**.
4. MDE enables **coupling** with other models.
5. Analysis & Execution **are** faster than running all tests.

Limits & Future work

- Low performances of models make the approach only suitable for **offline** RTS.
- Evaluate RTS with **variable granularities** & **comparison** with state-of-the-art tools.
- Perform **more operations** on the impact analysis model: fault localization, code coverage, debugging ...

Conclusion

- Model-Driven approach for **Safe** & highly **Precise** RTS.
- Can effectively **reduce** regression testing time.
- Offers better **reusability** and **modularity** than existing RTS approaches.

THANKS!

Contact:

Thibault Béziers la Fosse
IMT Atlantique - ICAM
Nantes, France

thibault.beziers-la-fosse@ls2n.fr
@ThibaultBLF

Regression Test Selection: Granularity

```
public class App {

   static void methodA ( int i ) {
         if ( i == 0 ) {
            doSomething4 ();
         } else {
            doSomething2 ();
         }
   }

   static void methodB () {
         doSomething3 ();
   }

}
```

```
public class TestApp {

   @Test
   public void testMethodA_0 () {
         App.methodA ( 0 );
   }                                  line
   @Test
   public void testMethodA_1 () {
         App.methodA ( 1 );
   }                                  method
   @Test
   public void testMethodB () {
         App.methodB ();
   }                                  class
}
```

Change Impact Analysis

Statement-Level Change

```
static void methodB () {
    doSomething3 ();
}
```

```
static void methodB () {
    doSomething4 ();
}
```

Declaration-Level Change

```
static void methodB () {
    doSomething3 ();
}
```

```
static void methodC () {
    doSomething3 ();
}
```

Change Impact Analysis (SUT)

Change Level	Statement-Level			Declaration-Level		
Type of Change	**Insert**	**Modify**	**Delete**	**Insert**	**Modify**	**Delete**
Impact Granularity	Method Level	Statement Level	Statement Level	Method Level	Method Level	Method Level

Change Impact Analysis (SUT)

```
abstract class SuperC {
  void m() {
    doSomething();
  }
}
```

Revision 1

```
class C extends SuperC {


}
```

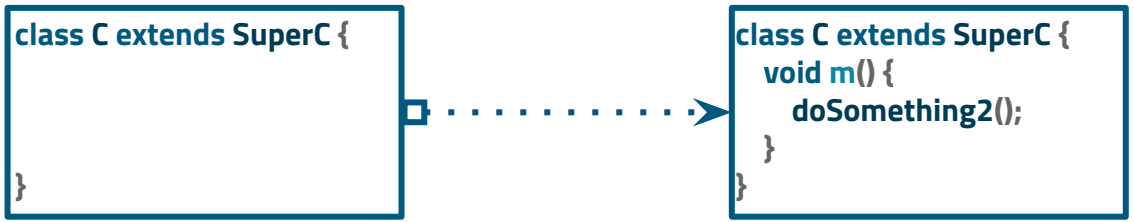
```
class TestC {
  @Test
  void test() {
    new C().m();
  }
}
```

```
abstract class SuperC {
  void m() {
    doSomething();
  }
}
```

Revision 2

```
class C extends SuperC {
  void m() {
    doSomething2();
  }
}
```

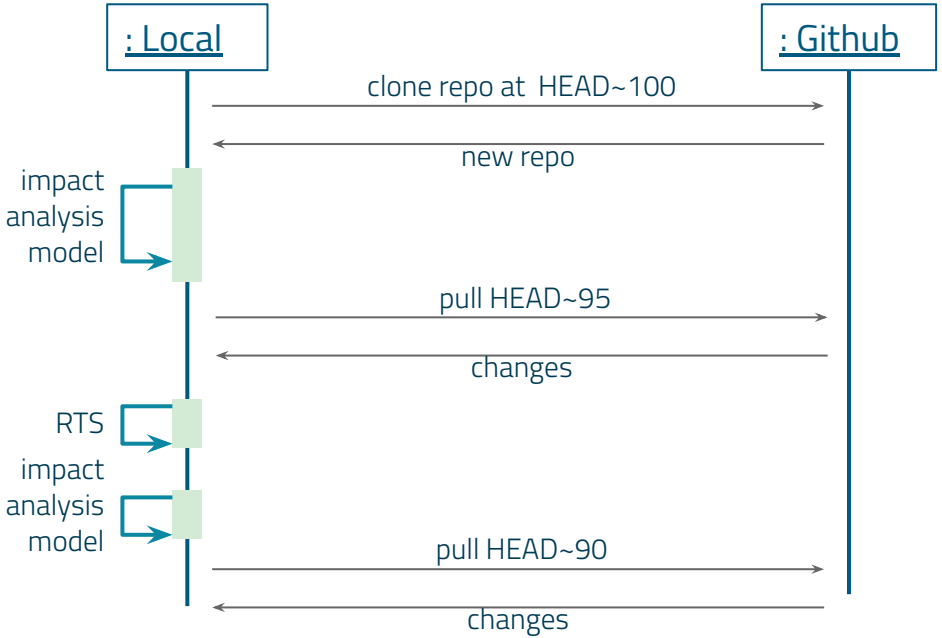
```
class TestC {
  @Test
  void test() {
    new C().m();
  }
}
```



Change Impact Analysis (Tests)

Change Level	Statement-Level			Declaration-Level		
Type of Change	**Insert**	**Modify**	**Delete**	**Insert**	**Modify**	**Delete**
Impact Granularity	Re-run	Re-run	Re-run	Run	Run	Ignore

Evaluation Workflow



...

Regression Test Selection