

CS 5154: Software Testing

Regression Testing

Owolabi Legunsen

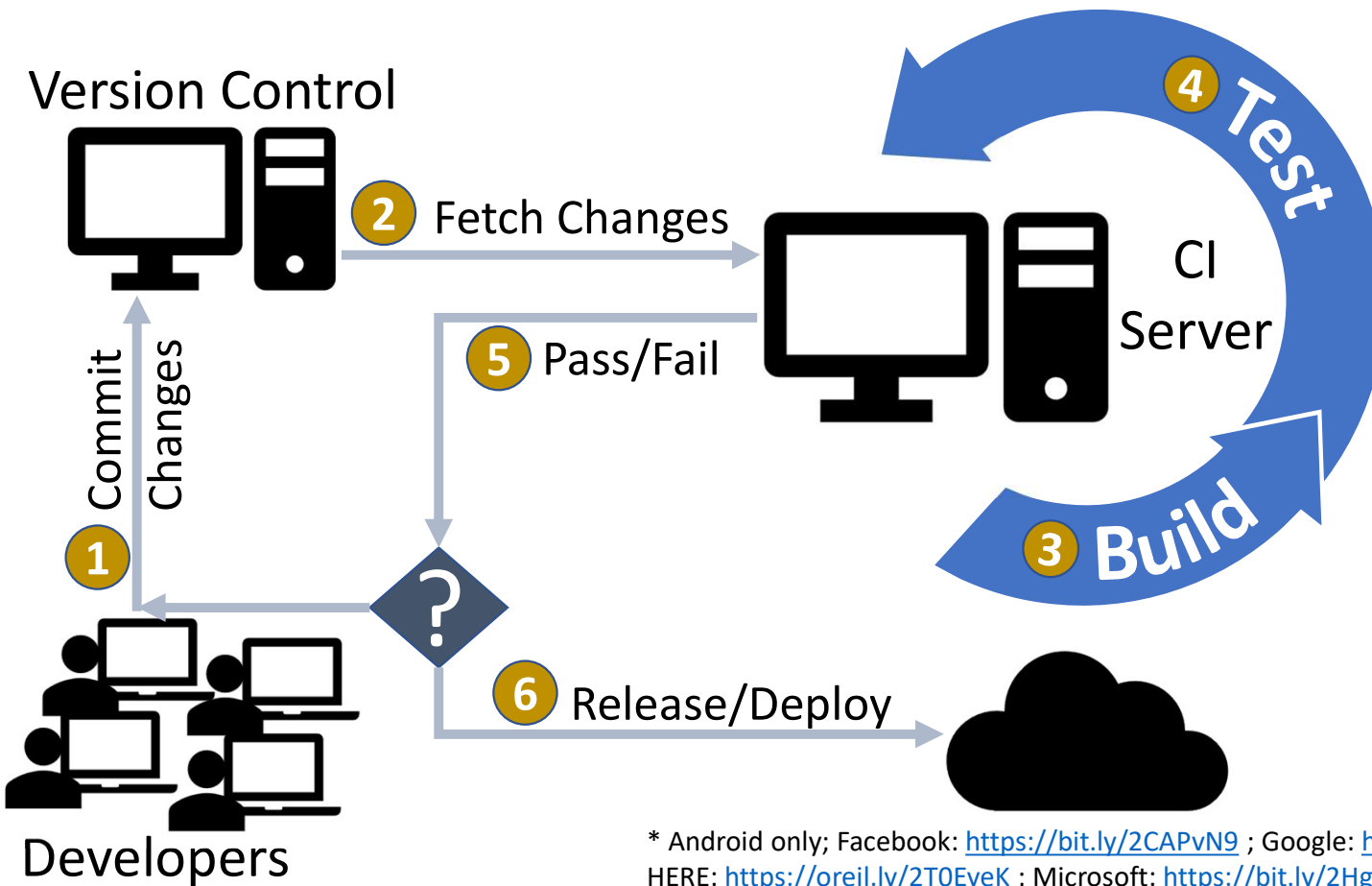
Review of the six CS5154 themes

1. How to automate the execution of tests? ✓
2. How to design and write high-quality tests? ✓
3. How to measure the quality of tests? ✓
4. How to automate the generation of tests? [??]
5. How to reduce the costs of running existing tests? ←
6. How to deal with bugs that tests reveal? [??]

What is regression testing?

Re-running tests to check that code changes do not break previously working functionality.

A common setting for regression testing: CI



Builds per day:

- Facebook: 60K*
- Google: 17K
- HERE: 100K
- Microsoft: 30K
- Single open-source projects: up to 80

Releases per day

- Etsy: 50

* Android only; Facebook: <https://bit.ly/2CAPvN9> ; Google: <https://bit.ly/2SYY4rR> ;
HERE: <https://oreil.ly/2T0EyeK> ; Microsoft: <https://bit.ly/2HgjUpw> ; Etsy: <https://bit.ly/2liSOJP> ;

What we'll talk about today

Problem: Regression testing can be very slow








Solution: Techniques to speed up regression testing



Developers

* Android only; Facebook: <https://bit.ly/2CAPvN9> ; Google: <https://bit.ly/2SYY4rR> ;
HERE: <https://oreil.ly/2T0EyeK> ; Microsoft: <https://bit.ly/2HgJUpw> ; Etsy: <https://bit.ly/2liSOJP> ;

Regression testing can be very slow

	test execution time	number of tests
	~5min	1667
 guava-libraries <small>Guava: Google Core Libraries for Java 1.6+</small>	~10min	641534
	~45min	1296
	~45min	361
	~45min	631
	~4h	4975
	~17h	8663

Re-run many times each day

The cost of regression testing is growing!



- 2015: ~10min 641,534 tests
- 2021: ~24min 1,713,729 tests

✓ Use Truth for better failure messages i...
CI #693: Commit 123c9a7 pushed by copybara-service master
bot 4 hours ago 23m 30s ...

Testing Google Guava locally: ~1/2 of lecture time

```
[INFO] -----  
[INFO] Reactor Summary for Guava Maven Parent HEAD-jre-SNAPSHOT:  
[INFO]  
[INFO] Guava Maven Parent ..... SUCCESS [ 4.414 s]  
[INFO] Guava: Google Core Libraries for Java ..... SUCCESS [02:13 min]  
[INFO] Guava BOM ..... SUCCESS [ 2.735 s]  
[INFO] Guava Testing Library ..... SUCCESS [05:06 min]  
[INFO] Guava Unit Tests ..... SUCCESS [25:20 min]  
[INFO] Guava GWT compatible libs ..... FAILURE [ 44.347 s]  
[INFO] -----  
[INFO] BUILD FAILURE  
[INFO] -----  
[INFO] Total time: 33:38 min  
[INFO] Finished at: 2021-11-22T20:27:28-05:00  
[INFO] -----
```


Why is the cost of regression testing growing?

- Number of changes per day is growing linearly
- Number of tests that are being run is growing linearly
- So, test execution time is growing quadratically



In 2011,

- 75+ million tests run per day
- **20+ revisions per minute**

*<https://testing.googleblog.com/2011/06/testing-at-speed-and-scale-of-google.html>

What are your ideas for speeding up testing?

What are your ideas for speeding up testing? (2)

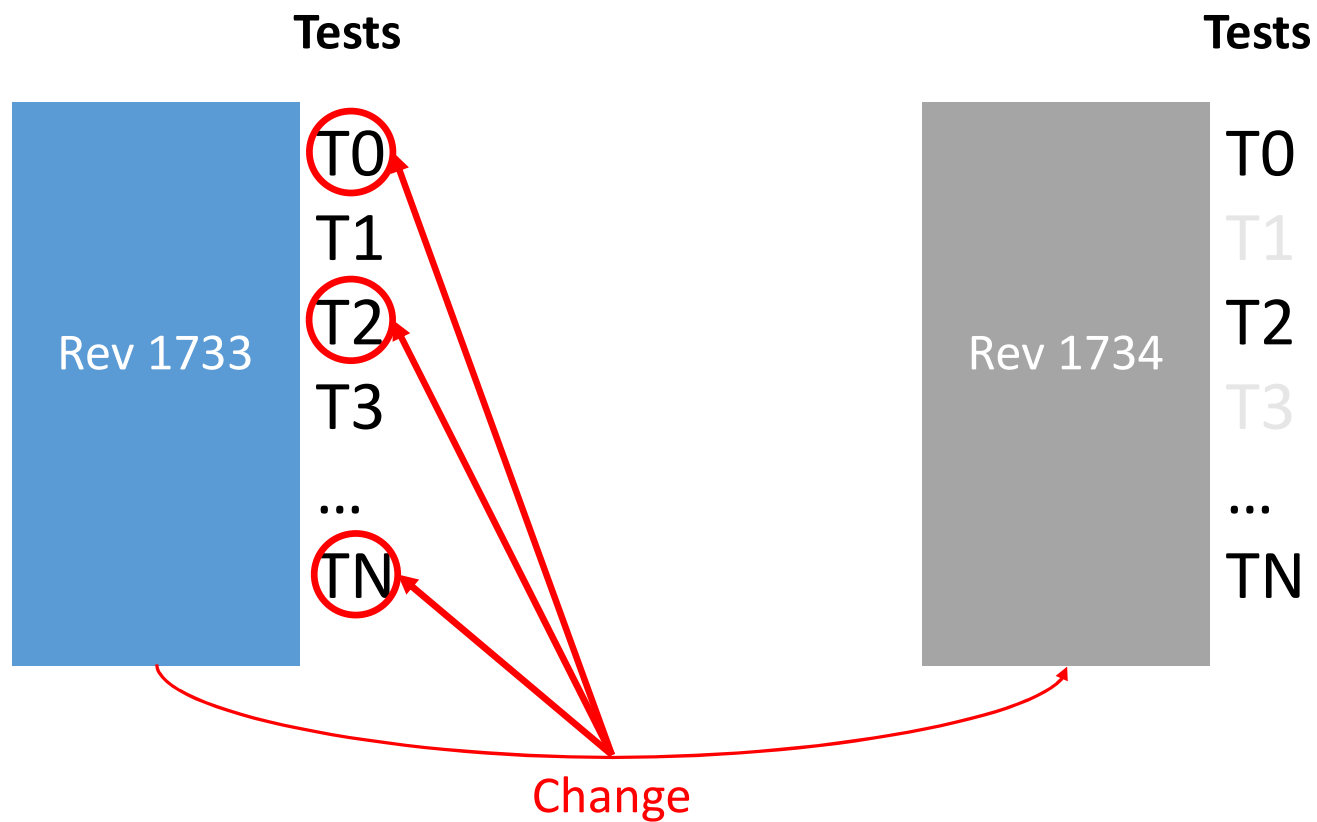
Goals for a regression testing technique

1. Detect regression faults as soon as possible
2. Reduce overall costs of testing
 - a. Costs in machine time to run tests
 - b. Costs in developer time to wait for test results

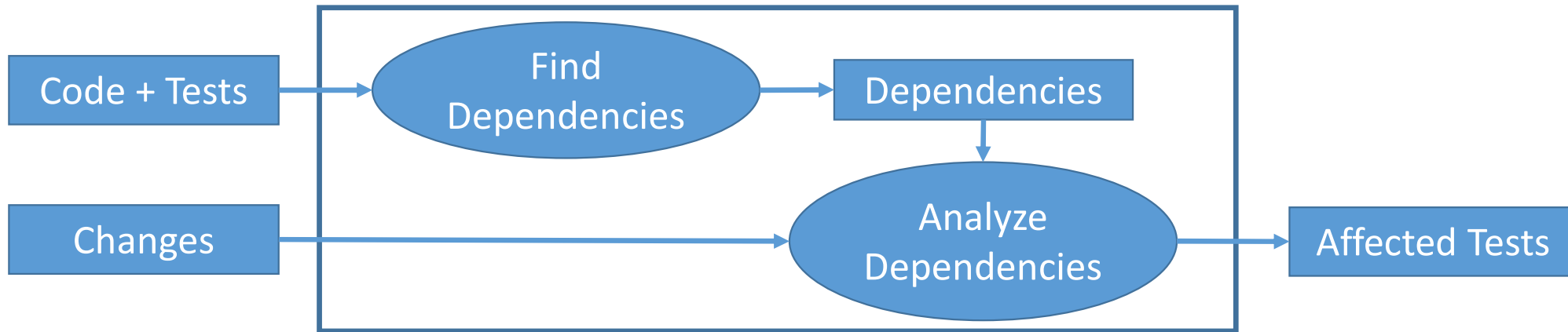
Some regression testing techniques

- **RetestAll** : Re-run all tests after a change
- **Regression Test Selection (RTS)** : Re-run subset of tests that are “affected” by code changes
- **Test-Suite Reduction (Minimization)** : Remove redundant tests
- **Test-Case Prioritization** : Order tests so that those that are “more important” are run first

Regression Test Selection (RTS)

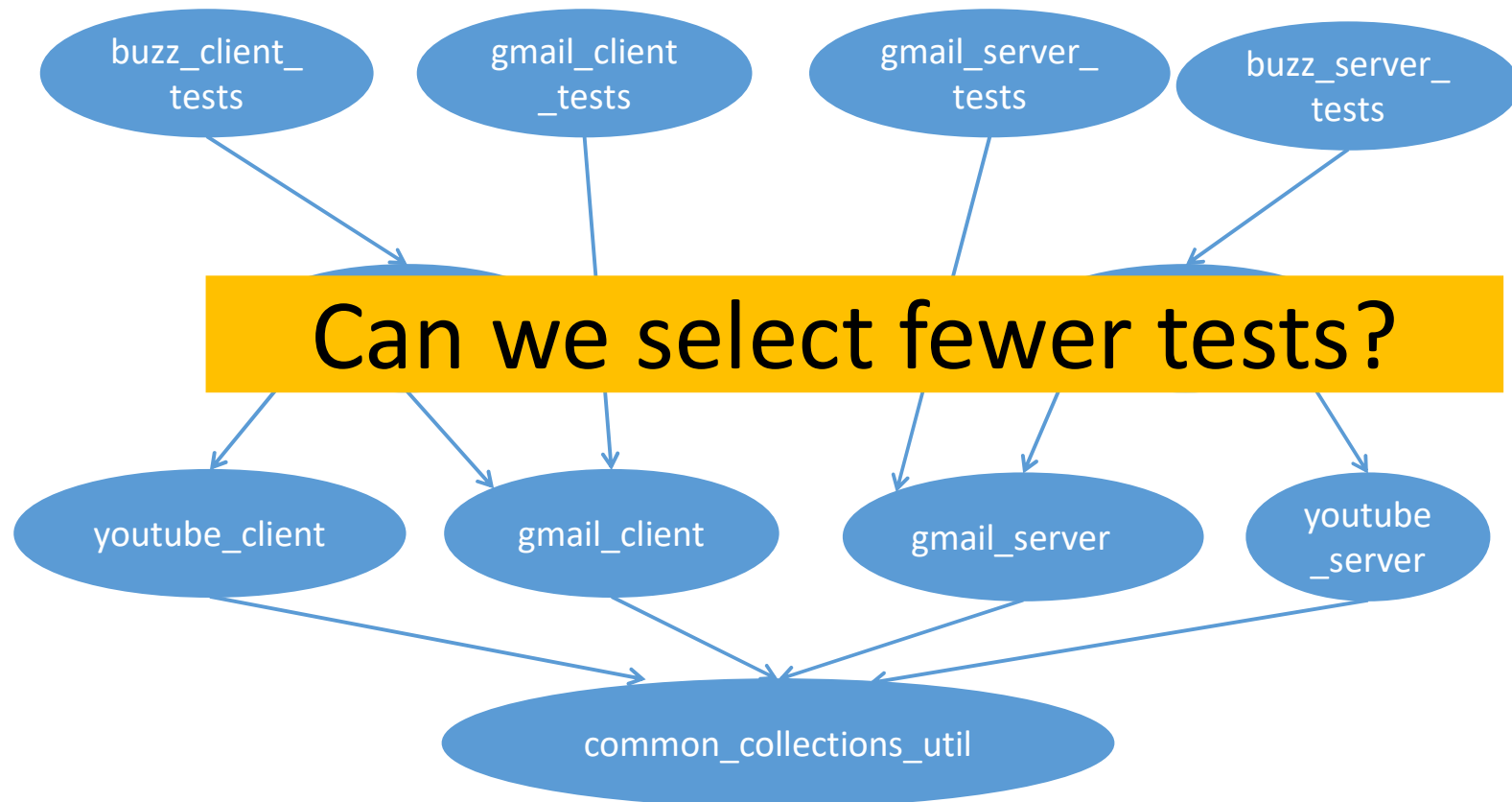


How RTS works



- An **affected test** can behave differently due to code changes *after ??*
- A test is affected if any of its dependencies changed

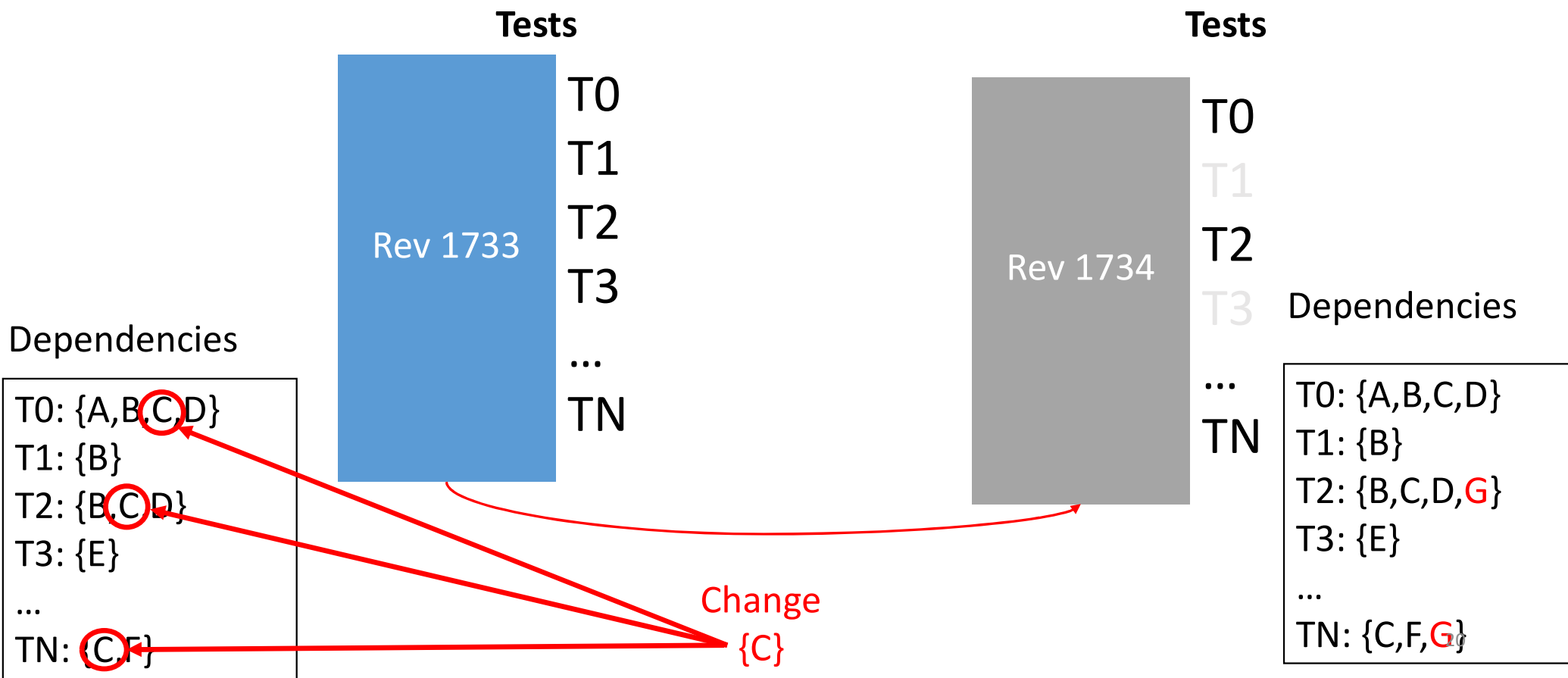
RTS at Google (Target/Module Level)



Class-level RTS

- Track dependencies between classes (in Java)
 - Collect changes at class level
 - Connect related classes
 - Select test classes (run all test methods in selected test class)
- How do we track test dependencies?
- How do we track changes?

Class-level Dynamic RTS (Ekstazi¹)





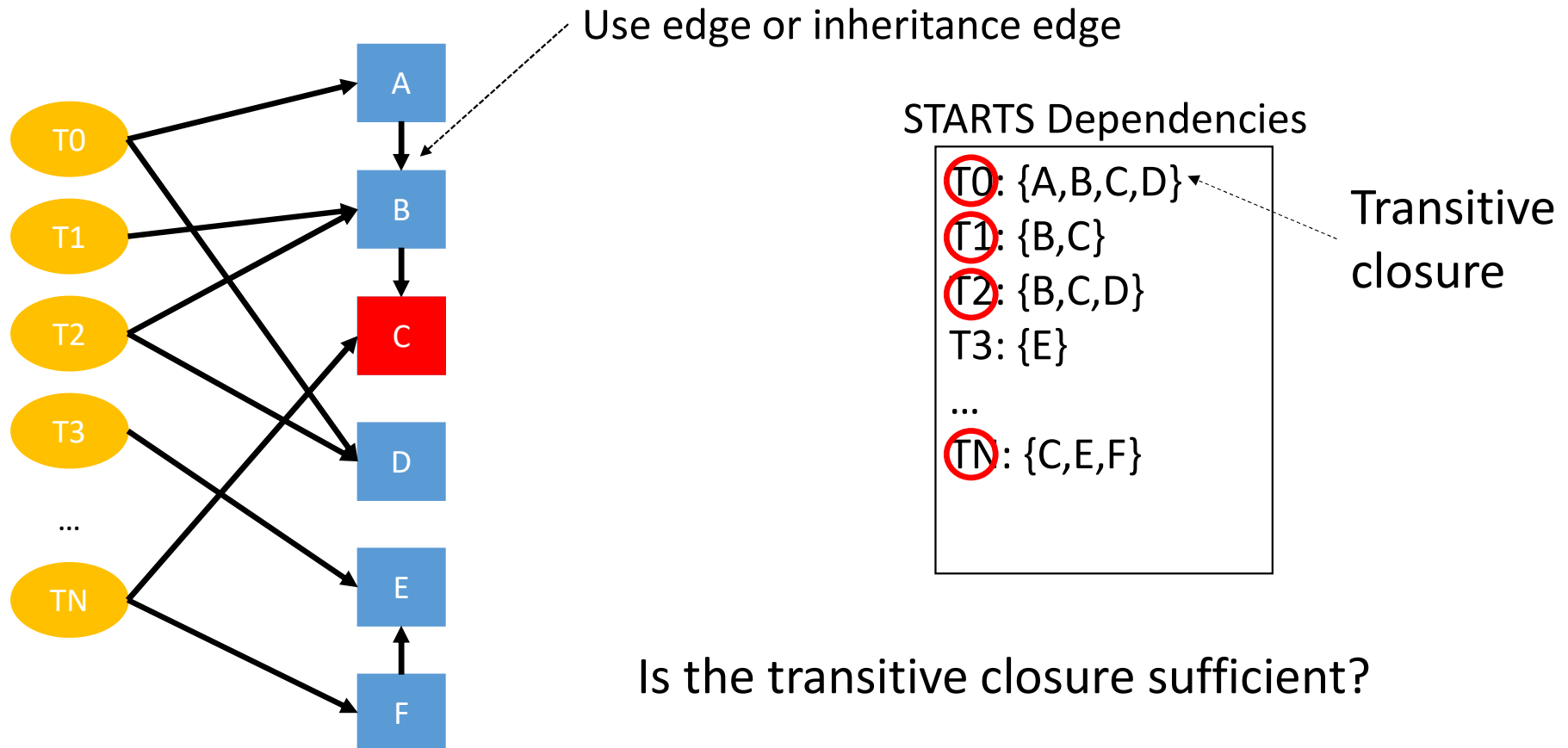
Ekstazi

How Ekstazi works

- **Find Dependencies:** dynamically track classes used while running each test class
 - Instrument classes to figure out which classes are used/loaded when running tests in some test class
- **Changes:** classes whose .class (bytecode) files differ
- **Analyze Dependencies:** select test classes for which any of its dependencies changed
 - Maintain dependencies between versions

¹Legunsen et al., *An Extensive Study of Static Regression Test Selection in Modern Software Evolution*. FSE 2016, <https://github.com/TestingResearchIllinois/starts>

Class-level STatic RTS (STARTS¹)



How STARTS works

- First, build a class dependency graph at compile time
 - Each class has an edge to direct superclass/interface and referenced classes
- **Find Dependencies:** classes reachable from each test class in the graph
- **Changes:** computed in same way as Ekstazi
- **Analyze Dependencies:** select test classes that reach a changed class in the graph

Some RTS tools you can use today

- Built by researchers (click on links below)
 - [STARTS](#)
 - [Ekstazi](#)
- Built by industry (click on links below)
 - [Microsoft Test Impact Analysis](#)
 - [OpenClover Test Optimization](#)
 - Ekstazi Gradle Plugin

Ekstazi “in the wild”



“Your tool is quite impressive; congratulations!”
an Apache Commons Math developer



Hangout with Google managers and developers

Several feature requests from various (Apache) developers

STARTS “in the wild”

- At least 6 dissertations built on or used STARTS
 - UIUC
 - KTH in Sweden
 - Hacettepe University in Turkey
 - Colorado State University

A conversation from 2021 lecture

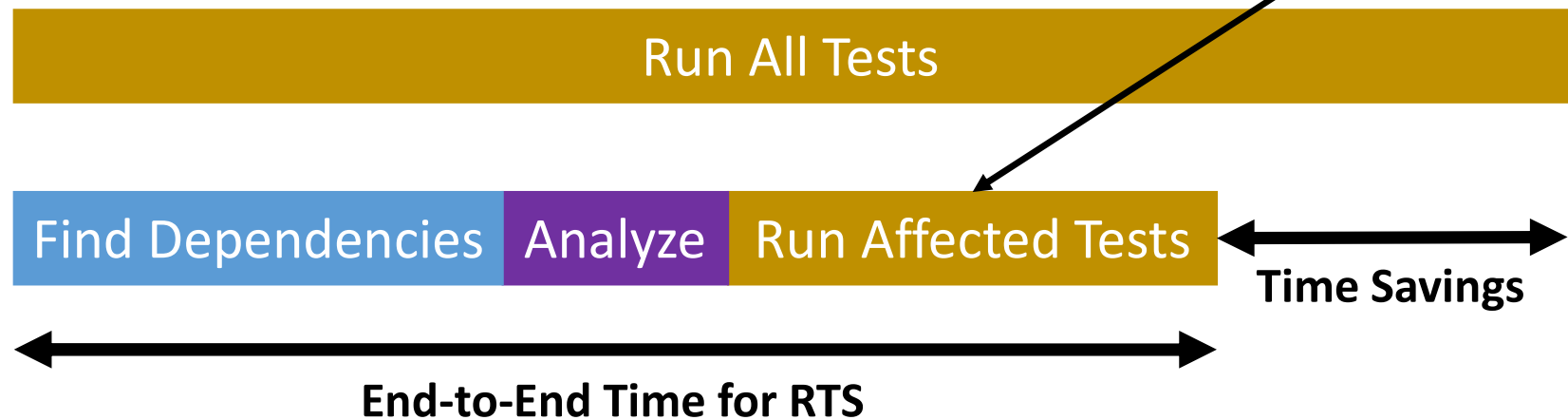
Owolabi : For an RTS technique to be useful, the end-to-end time of finding dependencies, analyzing dependencies+changes, and rerunning affected tests must be less than the time to simply re-run all tests

Student : What if RTS selects all tests to be re-run?

Owolabi : 😞

Important RTS Considerations

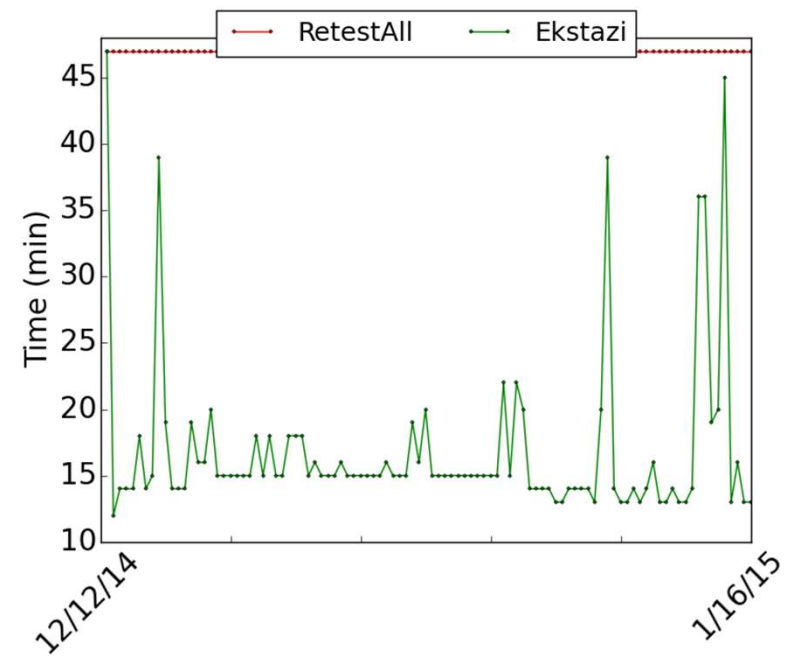
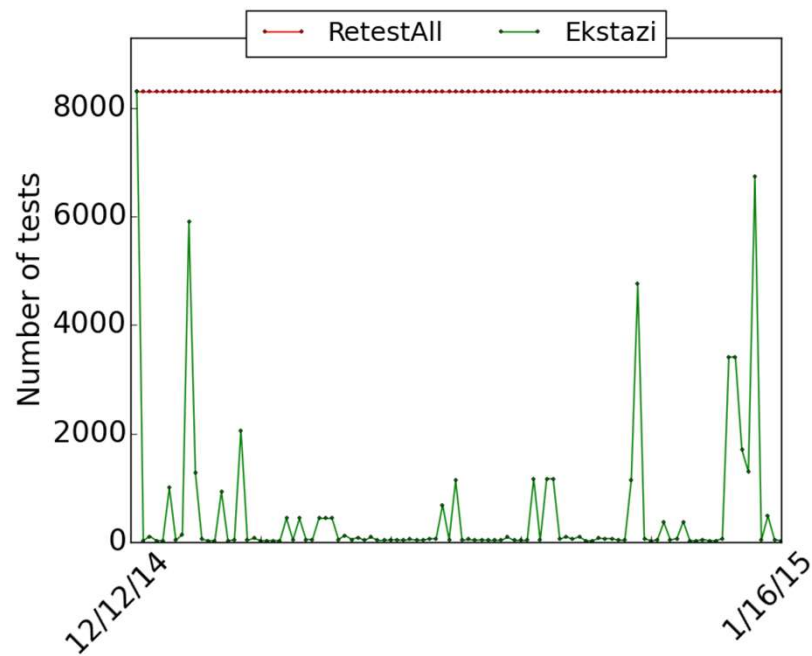
For Ekstazi, includes time to run and collect dependencies



- End-to-end time for RTS must be less than time to run all tests
- RTS should be **safe**: it should select to rerun *all* affected tests
- RTS should be **precise**: it should select to rerun *only* affected tests

Apache CXF

Benefit of RTS is measured across many versions



Reduces number of tests: **~15x** (10% more than dynamic method-level RTS)

Reduces test execution time: **~8x**

Dynamic vs Static

- Dynamic:
 - Pro
 - Finds exactly what tests depends on
 - Con
 - Requires executing tests to collect dependencies (overhead)
- Static:
 - Pro
 - Quick analysis without needing to execute tests
 - Con
 - Can over-approximate affected tests due to static analysis
 - May miss dependencies (reflection!)

A conversation from 2021 lecture

Owolabi: Module-level RTS saves costs but still runs too many tests because classes that changed may not be used by all modules that depend on changed module

Owolabi: So, we need to investigate class-level RTS

Student: But doesn't the same argument apply to class-level RTS?

Owolabi: 😊

Finer Granularity?

- Why not go even finer granularity of dependencies?
 - Method-level?
 - Statement-level?
- Collecting such dependencies (correctly) is harder/costlier
- More time to collect dependencies
 - Is the extra time worth it?
- Can be unsafe!

Safety Example (1)

Revision 0

```
class A {  
    A() {}  
    public void m() { ... }  
}
```

Revision 1

```
class A {  
    A() {}  
    public void m() { ... }  
+ public void n() { ... }  
}
```

```
Class Test {  
    @Test test() {  
        Method[] methods = A.class.getDeclaredMethods();  
        assertEquals(1, methods.length);  
    }  
}
```

Safety Example (1) – Dynamic Class-Level RTS

Revision 0

```
class A {  
    A() {}  
    public void m() { ... }  
}
```

Revision 1

```
class A {  
    A() {}  
    public void m() { ... }  
+ public void n() { ... }  
}
```

```
Class Test {  
    @Test test() {  
        Method[] methods = A.class.getDeclaredMethods();  
        assertEquals(1, methods.length);  
    }  
}
```

Would “Test” be selected?

***Should* “Test” be selected?**

Safety Example (1) – Static Class-Level RTS

Revision 0

```
class A {  
    A() {}  
    public void m() { ... }  
}
```

Revision 1

```
class A {  
    A() {}  
    public void m() { ... }  
+ public void n() { ... }  
}
```

```
Class Test {  
    @Test test() {  
        Method[] methods = A.class.getDeclaredMethods();  
        assertEquals(1, methods.length);  
    }  
}
```

Would “Test” be selected?

Should “Test” be selected?

Safety Example (1) – Dynamic Method-Level RTS

Revision 0

```
class A {  
    A() {}  
    public void m() { ... }  
}
```

Revision 1

```
class A {  
    A() {}  
    public void m() { ... }  
+ public void n() { ... }  
}
```

```
@Test test() {  
    Method[] methods = A.class.getDeclaredMethods();  
    assertEquals(1, methods.length);  
}
```

Would “test” be selected?

Should “test” be selected?

Safety Example (1) – Static Method-Level RTS

Revision 0

```
class A {  
    A() {}  
    public void m() { ... }  
}
```

Revision 1

```
class A {  
    A() {}  
    public void m() { ... }  
+ public void n() { ... }  
}
```

```
@Test test() {  
    Method[] methods = A.class.getDeclaredMethods();  
    assertEquals(1, methods.length);  
}
```

Would “test” be selected?

Should “test” be selected?

Safety Example (2)

Revision 0

```
class A {  
    A() {}  
    int m() { return 1; }  
}  
  
class B extends A {  
    B() {} // calls A()  
  
}
```

```
@Test test() {  
    B b = new B();  
    assertEquals(1, b.m());  
}
```

Revision 1

```
class A {  
    A() {}  
    int m() { return 1; }  
}  
  
class B extends A {  
    B() {} // calls A()  
+ @Override  
+ int m() { return 2; }  
}
```

Safety Example (2) – Dynamic Class-Level RTS

Revision 0

```
class A {  
    A() {}  
    int m() { return 1; }  
}  
  
class B extends A {  
    B() {} // calls A()  
}
```

Revision 1

```
class A {  
    A() {}  
    int m() { return 1; }  
}  
  
class B extends A {  
    B() {} // calls A()  
+ @Override  
+ int m() { return 2; }  
}
```

```
Class Test {  
    @Test test() {  
        B b = new B(); assertEquals(1, b.m());  
    }  
}
```

Would “Test” be selected?

***Should* “Test” be selected?**

Safety Example (2) – Static Class-Level RTS

Revision 0

```
class A {  
    A() {}  
    int m() { return 1; }  
}  
  
class B extends A {  
    B() {} // calls A()  
}
```

Revision 1

```
class A {  
    A() {}  
    int m() { return 1; }  
}  
  
class B extends A {  
    B() {} // calls A()  
+ @Override  
+ int m() { return 2; }  
}
```

```
Class Test {  
    @Test test() {  
        B b = new B(); assertEquals(1, b.m());  
    }  
}
```

Would “Test” be selected?

***Should* “Test” be selected?**

Safety Example (2) – Dynamic Method-Level RTS

Revision 0

```
class A {  
    A() {}  
    int m() { return 1; }  
}  
  
class B extends A {  
    B() {} // calls A()  
}
```

```
@Test test() {  
    B b = new B();  
    assertEquals(1, b.m());  
}
```

Revision 1

```
class A {  
    A() {}  
    int m() { return 1; }  
}  
  
class B extends A {  
    B() {} // calls A()  
+ @Override  
+ int m() { return 2; }  
}
```

Would “test” be selected?

***Should* “test” be selected?**

Safety Example (2) – Static Method-Level RTS

Revision 0

```
class A {  
    A() {}  
    int m() { return 1; }  
}  
  
class B extends A {  
    B() {} // calls A()  
}
```

```
@Test test() {  
    B b = new B();  
    assertEquals(1, b.m());  
}
```

Revision 1

```
class A {  
    A() {}  
    int m() { return 1; }  
}  
  
class B extends A {  
    B() {} // calls A()  
+ @Override  
+ int m() { return 2; }  
}
```

Would “test” be selected?

Should “test” be selected?

Class-level vs Target/Module-level

- Class-level test selection should be more precise than target/module-level test selection
 - Selects to run all tests in affected test class, not all tests in affected test target/module
- Why do companies not use class-level test selection?

Questions

