

Christian Plewnia
christian.plewnia@rwth-aachen.de

A Framework for Regression Test Prioritization and Selection

Master Thesis Final Presentation

December 9th, 2015



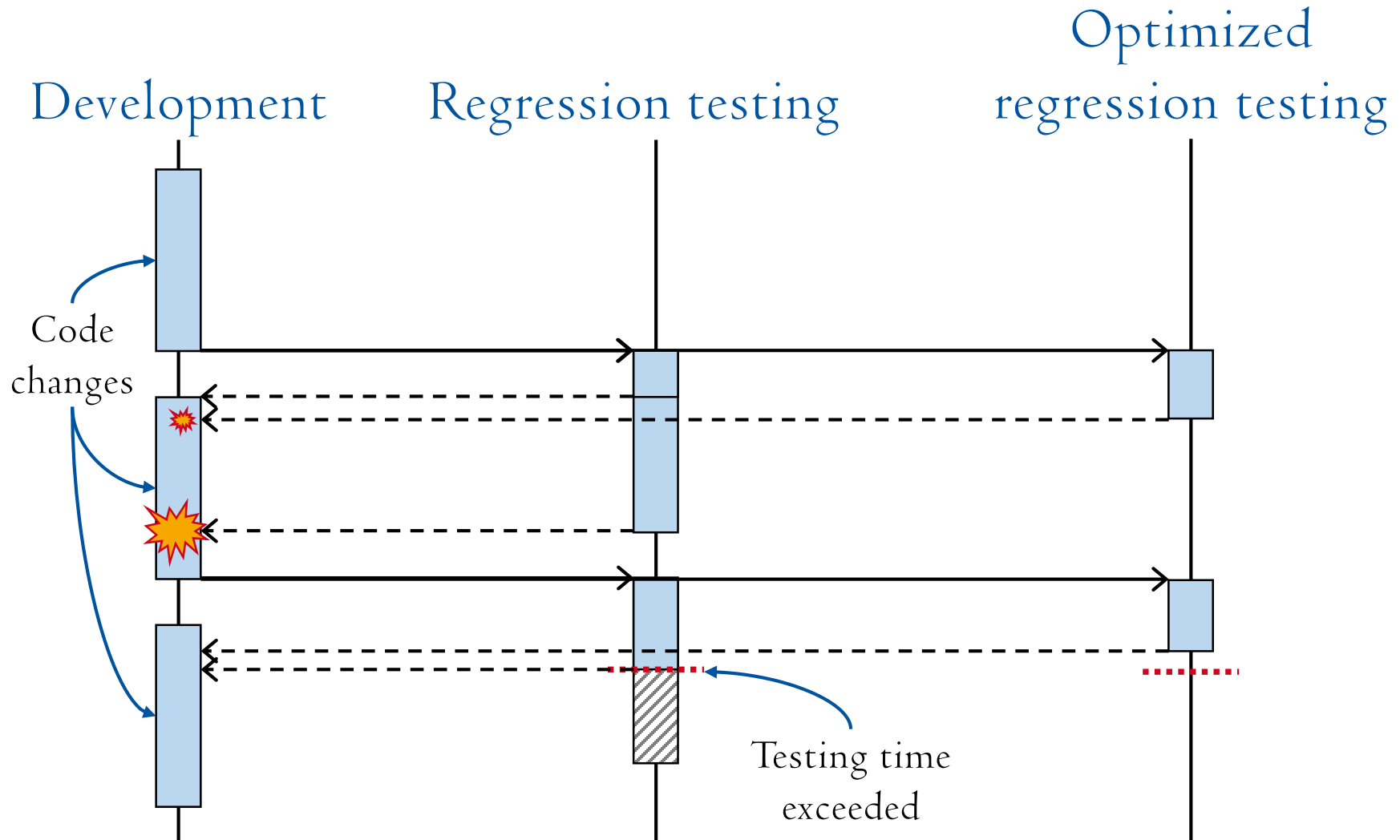
Regressions, Test Automation, and Regression Testing



ILLUSTRATION BY SEGUE TECHNOLOGIES

Source: <http://www.seguetech.com>

Challenge

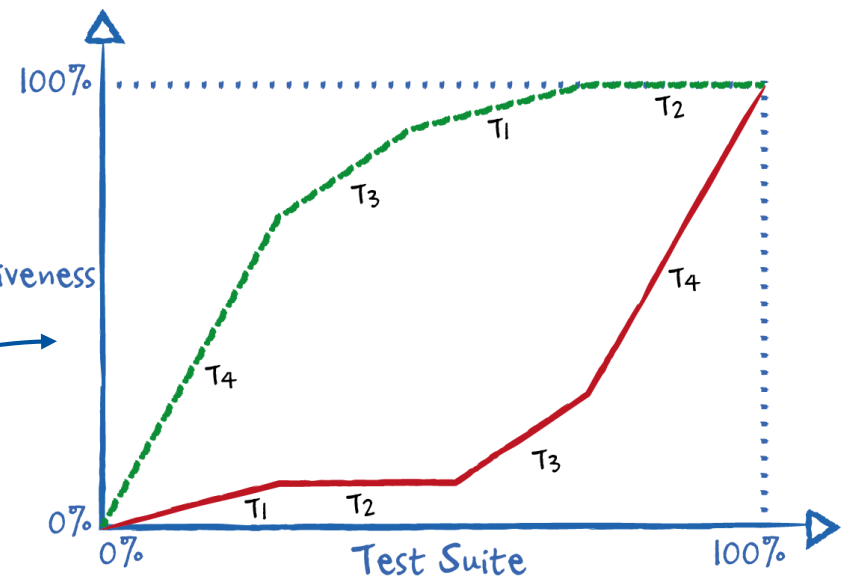
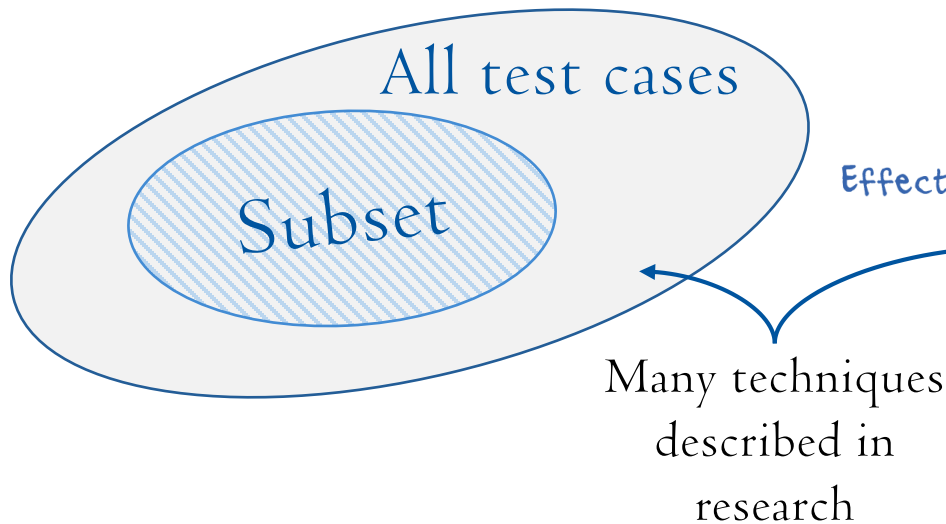


Regression Test Optimization

Regression Test Optimization (RTO)

Regression Test Selection (RTS)


Regression Test Prioritization (RTP)












RTO in Practice

- RTO is widely practiced **manually** by developers
 - techniques do not scale or
 - the **integration is difficult**

Challenges of integration

- Integration into **testing process** **maven**  **eclipse**
- Optimization requires **data gathering**
- **Execution** of optimized test suite

RTS & RTP Implementations

	Ekstazi	RTS
	InfiniTest	RTS (live testing)
	Test Load Balancer	RTP
	Echelon	RTP; internal tool by Microsoft
	Dependency Graph Tool*	RTS; internal tool by Google
	Cleanscape Testwise (ATAC)	RTS
	DejaVOO	RTS
	SPYDER	RTS
	Aristotle Analysis System	Library used by some RTO techniques

* Real name unknown



available



internal

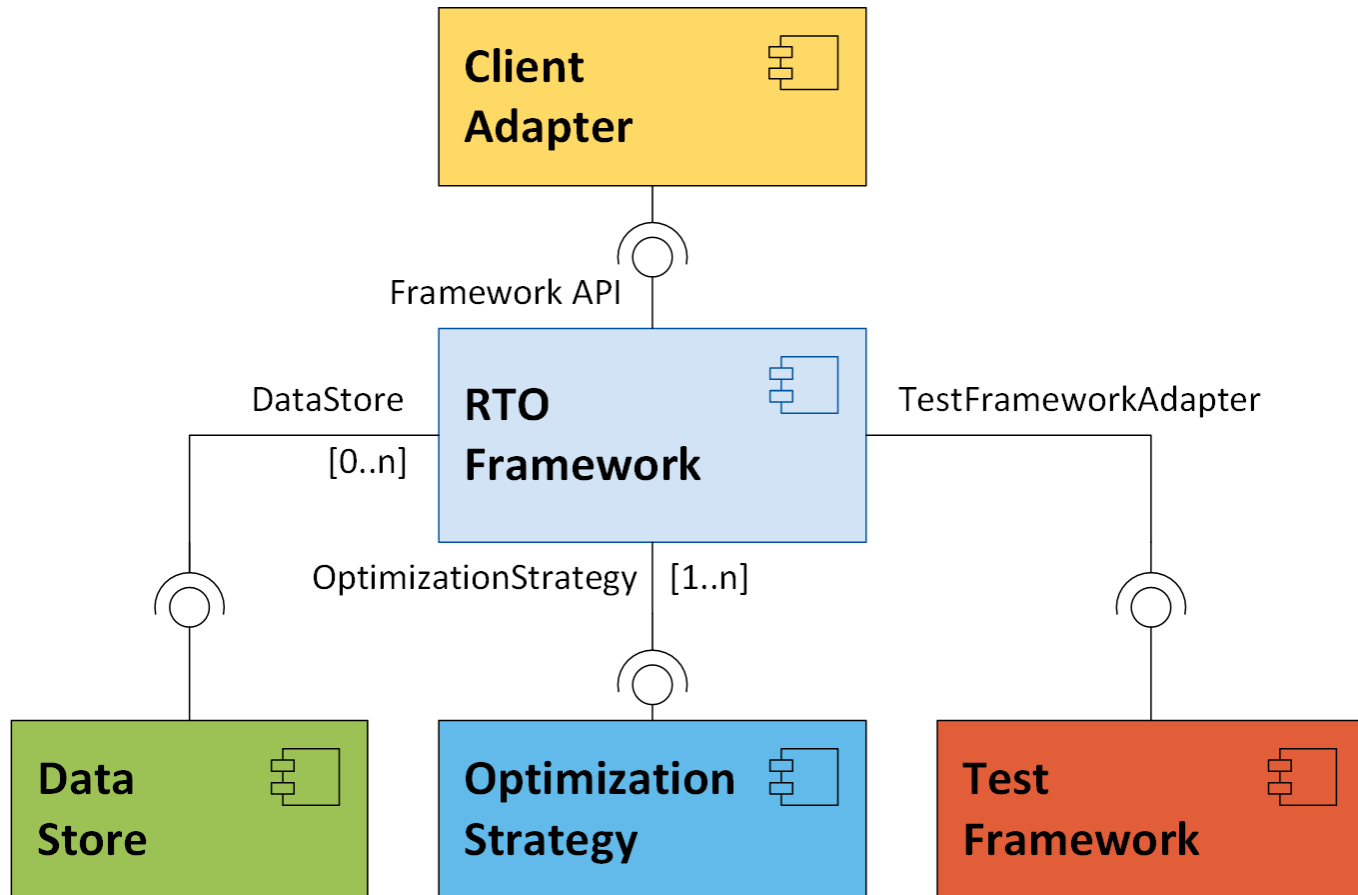


discontinued

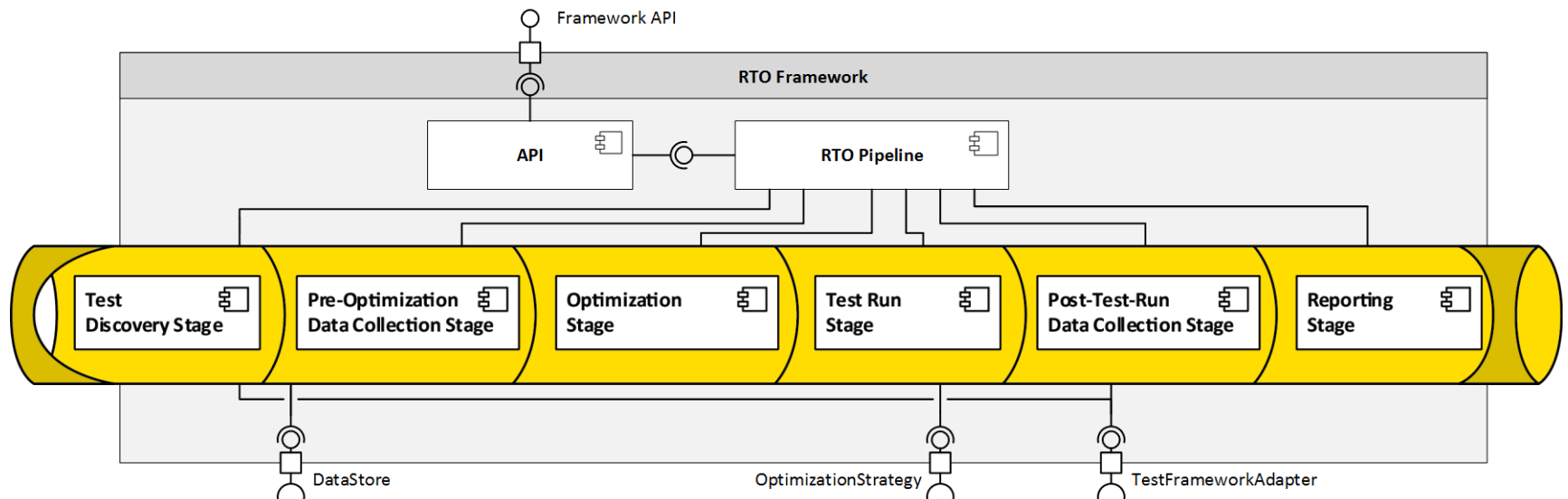
```
lazer-examples - [D:/repositories/MTCP/lazer.examples-develop] - [lazer-example-triangletester] - ...lazer-example-triangletester/src/main/logback-test.xml - IntelliJ IDEA 15.0.1
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
lazer.examples-develop > lazer-example-triangletester > pom.xml
Run lazer-example-triangletester [test]
Lazzer: 20:10:34.565 DEBUG d.r.s.l.t.j.l.TestRunLogger - Test SingleTest.test ..... IN PROGRESS
Lazzer: 20:10:34.566 DEBUG d.r.s.l.t.j.l.TestRunLogger - Test SingleTest.test ..... DONE
Lazzer: 20:10:34.566 DEBUG d.r.s.l.t.j.l.TestRunLogger - Test RobustnessBoundaryValueBoundaryValueTest.nomNomMinM ..... IN PROGRESS
Lazzer: 20:10:34.567 DEBUG d.r.s.l.t.j.l.TestRunLogger - Test RobustnessBoundaryValueBoundaryValueTest.nomNomMinM ..... DONE
Lazzer: 20:10:34.567 DEBUG d.r.s.l.t.j.l.TestRunLogger - Test RobustnessBoundaryValueBoundaryValueTest.nomMaxPNom ..... IN PROGRESS
Lazzer: 20:10:34.567 DEBUG d.r.s.l.t.j.l.TestRunLogger - Test RobustnessBoundaryValueBoundaryValueTest.nomMaxPNom ..... DONE
Lazzer: 20:10:34.567 DEBUG d.r.s.l.t.j.l.TestRunLogger - Test RobustnessBoundaryValueBoundaryValueTest.minMNomNom ..... IN PROGRESS
Lazzer: 20:10:34.567 DEBUG d.r.s.l.t.j.l.TestRunLogger - Test RobustnessBoundaryValueBoundaryValueTest.minMNomNom ..... DONE
Lazzer: 20:10:34.567 DEBUG d.r.s.l.t.j.l.TestRunLogger - Test RobustnessBoundaryValueBoundaryValueTest.nomNomMaxP ..... IN PROGRESS
Lazzer: 20:10:34.568 DEBUG d.r.s.l.t.j.l.TestRunLogger - Test RobustnessBoundaryValueBoundaryValueTest.nomNomMaxP ..... DONE
Lazzer: 20:10:34.568 DEBUG d.r.s.l.t.j.l.TestRunLogger - Test RobustnessBoundaryValueBoundaryValueTest.maxPNomNom ..... IN PROGRESS
Lazzer: 20:10:34.568 DEBUG d.r.s.l.t.j.l.TestRunLogger - Test RobustnessBoundaryValueBoundaryValueTest.maxPNomNom ..... DONE
Lazzer: 20:10:34.568 DEBUG d.r.s.l.t.j.l.TestRunLogger - Test RobustnessBoundaryValueBoundaryValueTest.nomMinMNom ..... IN PROGRESS
Lazzer: 20:10:34.568 DEBUG d.r.s.l.t.j.l.TestRunLogger - Test RobustnessBoundaryValueBoundaryValueTest.nomMinMNom ..... DONE
Lazzer: 20:10:34.569 DEBUG d.r.s.l.t.j.l.TestRunLogger - Test execution ended.
Lazzer: 20:10:34.571 DEBUG d.r.s.l.d.t.TestHistoryDataStore - Attempting to store test result in database.
Lazzer: 20:10:35.038 INFO d.r.s.l.f.i.p.s.ReportingStage - == Lazer Results Report ==
Lazzer: 20:10:35.038 INFO d.r.s.l.f.i.p.s.ReportingStage - Optimization Strategies
Lazzer: 20:10:35.038 INFO d.r.s.l.f.i.p.s.ReportingStage - [1/1] ExecutionTimePrioritization
Lazzer: 20:10:35.038 INFO d.r.s.l.f.i.p.s.ReportingStage - ~ Data Stores ~
Lazzer: 20:10:35.038 INFO d.r.s.l.f.i.p.s.ReportingStage - [1/1] TestHistoryDataStore
Lazzer: 20:10:35.038 INFO d.r.s.l.f.i.p.s.ReportingStage - ~ Tests ~
Lazzer: 20:10:35.039 INFO d.r.s.l.f.i.p.s.ReportingStage - [ 1/20] BoundaryValueBoundaryValueTest.nomNomMin ..... SUCCESS .... 0 ms
Lazzer: 20:10:35.039 INFO d.r.s.l.f.i.p.s.ReportingStage - [ 2/20] BoundaryValueBoundaryValueTest.nomNomNom ..... SUCCESS .... 0 ms
Lazzer: 20:10:35.039 INFO d.r.s.l.f.i.p.s.ReportingStage - [ 3/20] BoundaryValueBoundaryValueTest.nomNomMinP ..... SUCCESS .... 0 ms
Lazzer: 20:10:35.039 INFO d.r.s.l.f.i.p.s.ReportingStage - [ 4/20] BoundaryValueBoundaryValueTest.nomNomMaxM ..... SUCCESS .... 0 ms
Lazzer: 20:10:35.039 INFO d.r.s.l.f.i.p.s.ReportingStage - [ 5/20] BoundaryValueBoundaryValueTest.nomNomMax ..... SUCCESS .... 0 ms
Lazzer: 20:10:35.039 INFO d.r.s.l.f.i.p.s.ReportingStage - [ 6/20] BoundaryValueBoundaryValueTest.minPNomNom ..... SUCCESS .... 0 ms
Lazzer: 20:10:35.039 INFO d.r.s.l.f.i.p.s.ReportingStage - [ 7/20] BoundaryValueBoundaryValueTest.maxNomNom ..... SUCCESS .... 0 ms
Lazzer: 20:10:35.039 INFO d.r.s.l.f.i.p.s.ReportingStage - [ 8/20] BoundaryValueBoundaryValueTest.nomMinPNom ..... SUCCESS .... 0 ms
Lazzer: 20:10:35.039 INFO d.r.s.l.f.i.p.s.ReportingStage - [ 9/20] BoundaryValueBoundaryValueTest.nomMinNom ..... SUCCESS .... 0 ms
Lazzer: 20:10:35.039 INFO d.r.s.l.f.i.p.s.ReportingStage - [10/20] BoundaryValueBoundaryValueTest.nomMaxNom ..... SUCCESS .... 0 ms
Lazzer: 20:10:35.039 INFO d.r.s.l.f.i.p.s.ReportingStage - [11/20] BoundaryValueBoundaryValueTest.nomMaxMNom ..... SUCCESS .... 0 ms
Lazzer: 20:10:35.039 INFO d.r.s.l.f.i.p.s.ReportingStage - [12/20] BoundaryValueBoundaryValueTest.minNomNom ..... SUCCESS .... 0 ms
Lazzer: 20:10:35.039 INFO d.r.s.l.f.i.p.s.ReportingStage - [13/20] BoundaryValueBoundaryValueTest.maxMNomNom ..... SUCCESS .... 0 ms
Lazzer: 20:10:35.039 INFO d.r.s.l.f.i.p.s.ReportingStage - [14/20] SingleTest.test ..... SUCCESS .... 1 ms
Lazzer: 20:10:35.039 INFO d.r.s.l.f.i.p.s.ReportingStage - [15/20] RobustnessBoundaryValueBoundaryValueTest.nomNomMinM ..... SUCCESS .... 0 ms
```

Demo

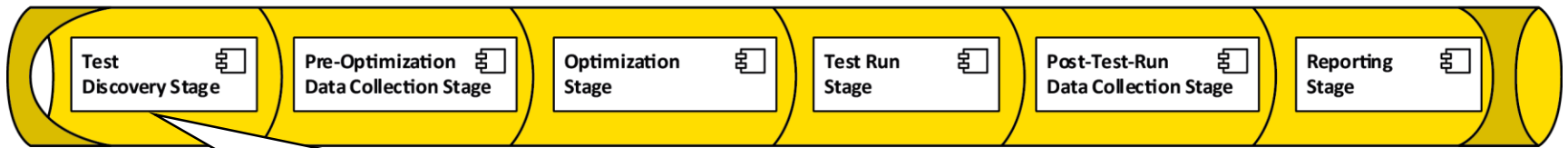
Lazzer Framework Architecture



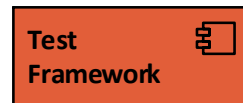
Lazzer Framework Architecture



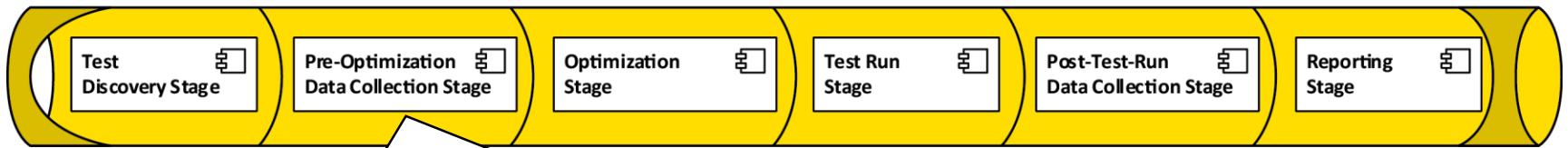
Lazzer Pipeline: Test Discovery



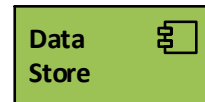
- Find all test cases in the given test suite
- Delegated to the test framework adapter



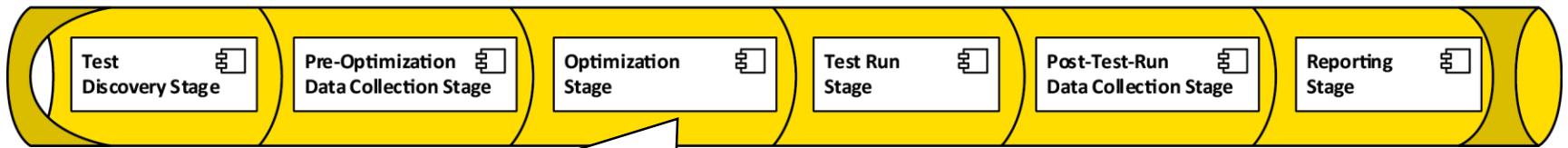
Lazzer Pipeline: Pre-Optimization Data Collection



- Enable data stores to collect information
- Example: determine source code changes



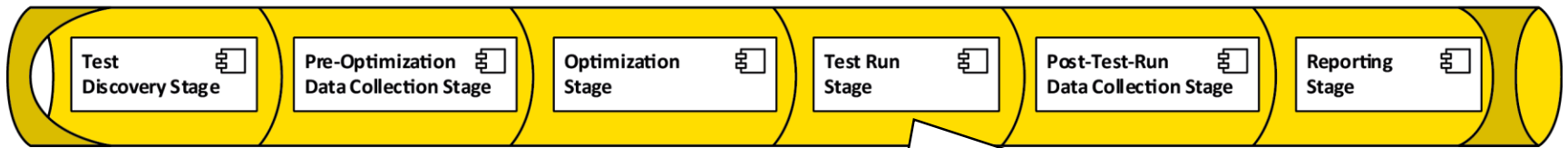
Lazzer Pipeline: Optimization



- Perform regression test optimization
- Delegated to optimization strategies
- The input and output is a test suite (a collection of test classes containing tests)



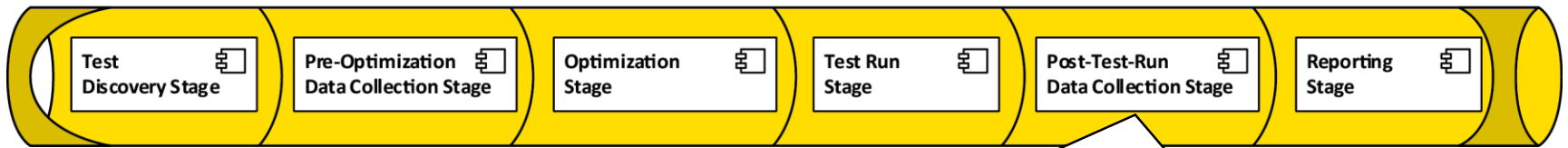
Lazzer Pipeline: Test Run



- Run the optimized test suite
- Delegated to the test framework

Test Framework

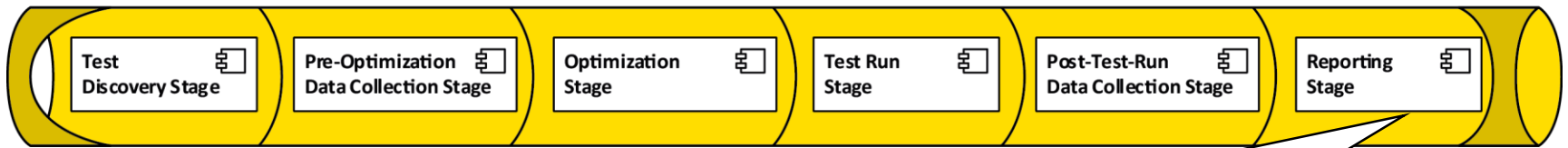
Lazzer Pipeline: Post-Test-Run Data Collection



- Enable data stores to collect information
- Example: Save test results in database

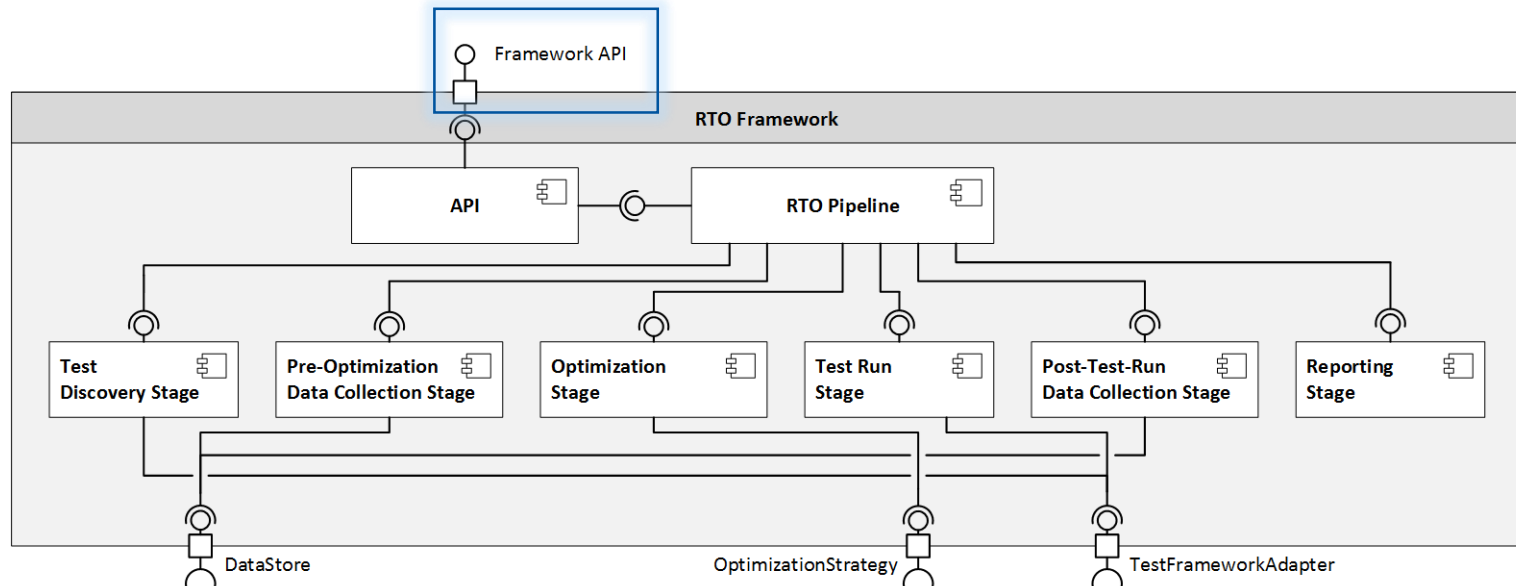


Lazzer Pipeline: Reporting



- Report test results
- Example: write to the log or an XML file

Lazzer Framework Architecture



Evaluation: Running Lazzer via its API

Client
Adapter

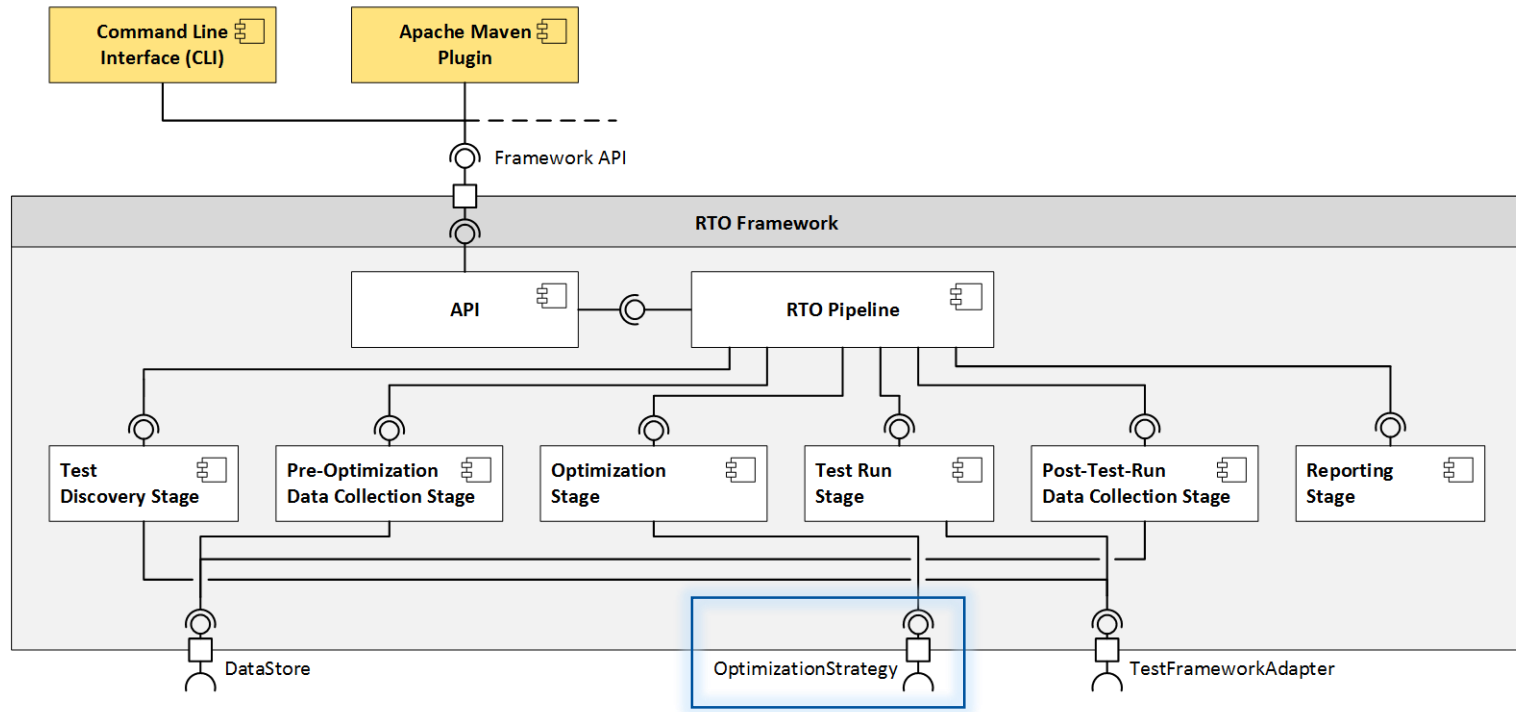


```
// Configuration
LazzerSettings settings =
    new LazzerSettings.Builder(
        // Test Framework Adapter
        new JUnit4Adapter(),
        // Optimization Strategy
        Arrays.asList(new FailedFirstPrioritization()),
        // Data Stores
        Arrays.asList(new TestHistory(...))
    ).build();

// Instantiate lazzer
Lazzer lazzer = LazzerFactory.createLazzer(settings);

// Execute test optimization & test run
lazzer.run();
```

Lazzer Framework Architecture



Developing a Lazzer Optimization Strategy

```
public interface OptimizationStrategy {
```

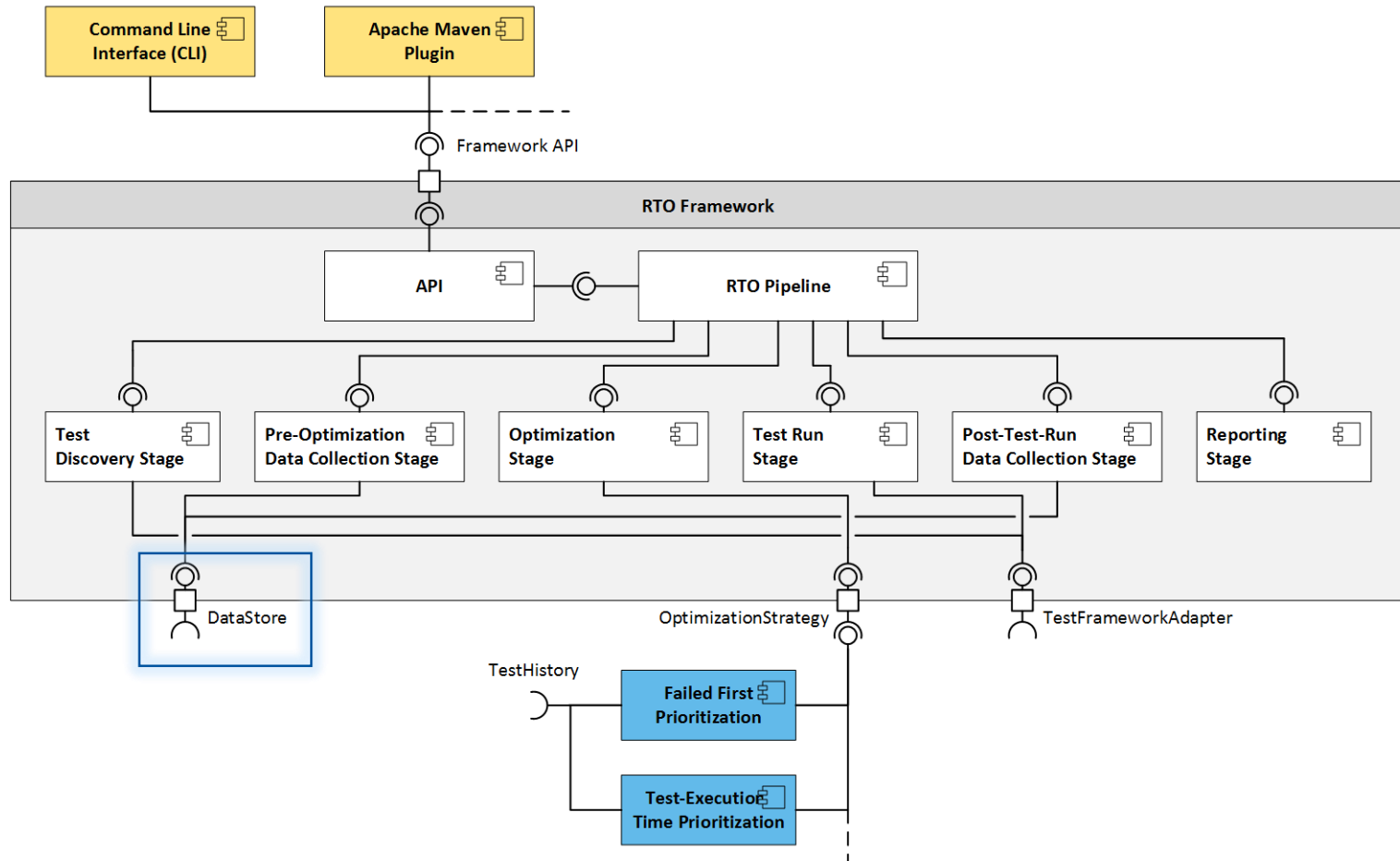
Optimization Strategy 

Called by the framework to
optimize the test suite

```
TestSuite optimize(TestSuite testSuite);
```

```
}
```

Lazzer Framework Architecture



Developing a Lazzer TestRunner

```
public interface DataStore {
```



Called by the framework before the optimization

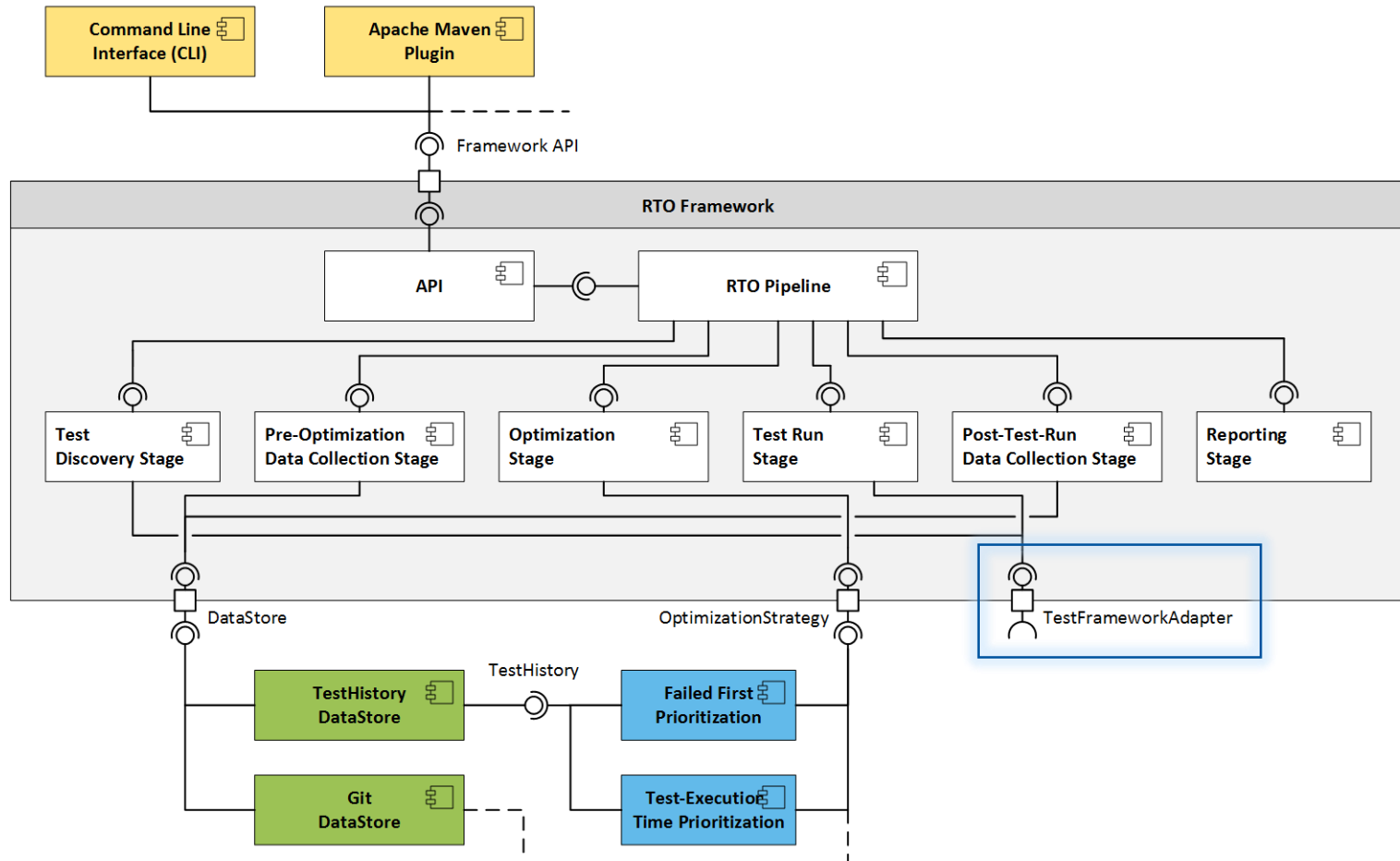
```
void preOptimizationDataCollection();
```

Called by the framework after the test run

```
void postTestRunDataCollection  
(TestRunReport testRunReport);
```

```
}
```

Lazzer Framework Architecture



Developing a Lazzer TestFrameworkAdapter

```
public interface TestFrameworkAdapter {
```

Test
Framework



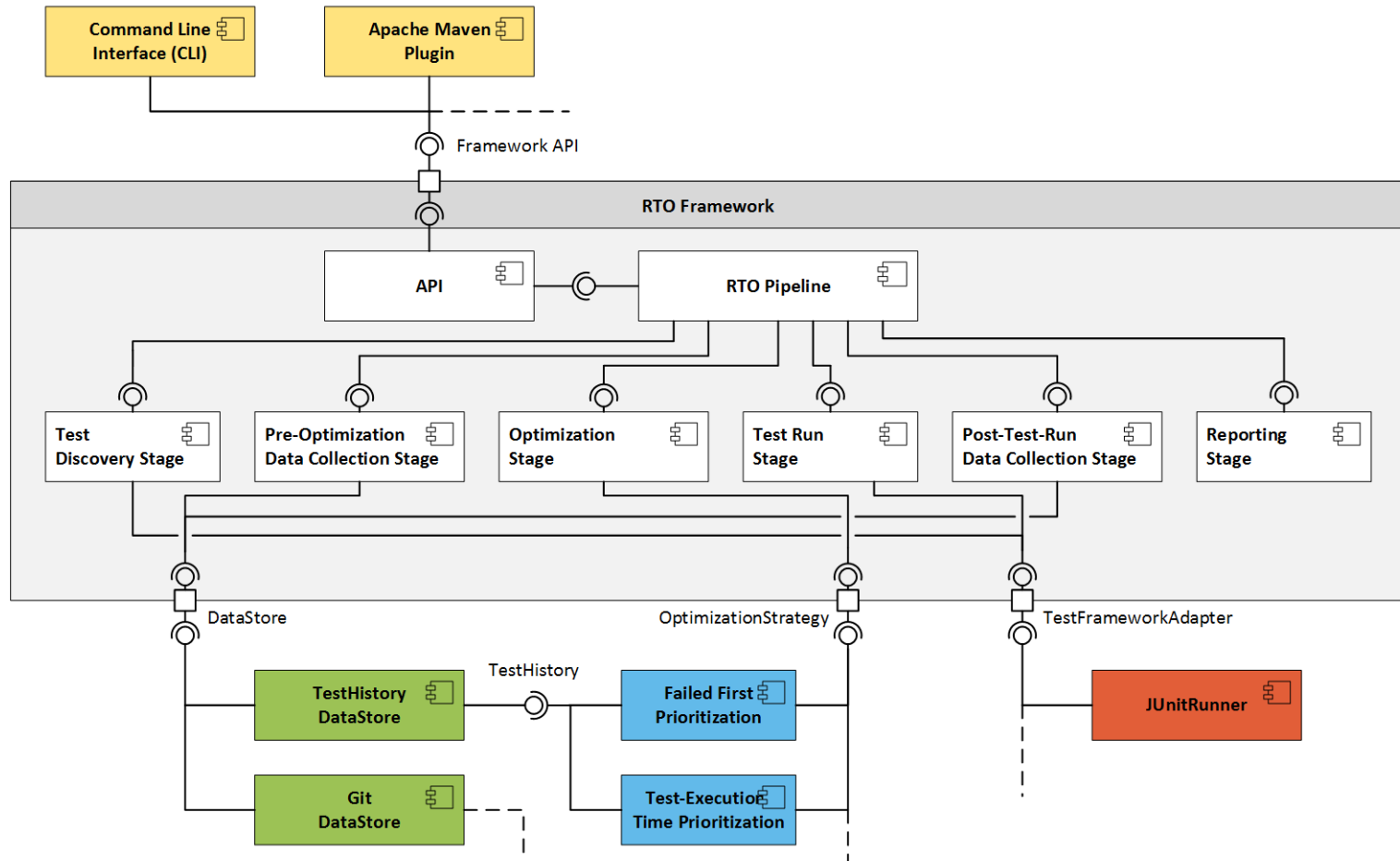
Finds test cases

```
TestDiscovery provideTestDiscovery (  
    Classpath classpathForSystemUnderTest,  
    Classpath classpathForTests  
);
```

Executes test run

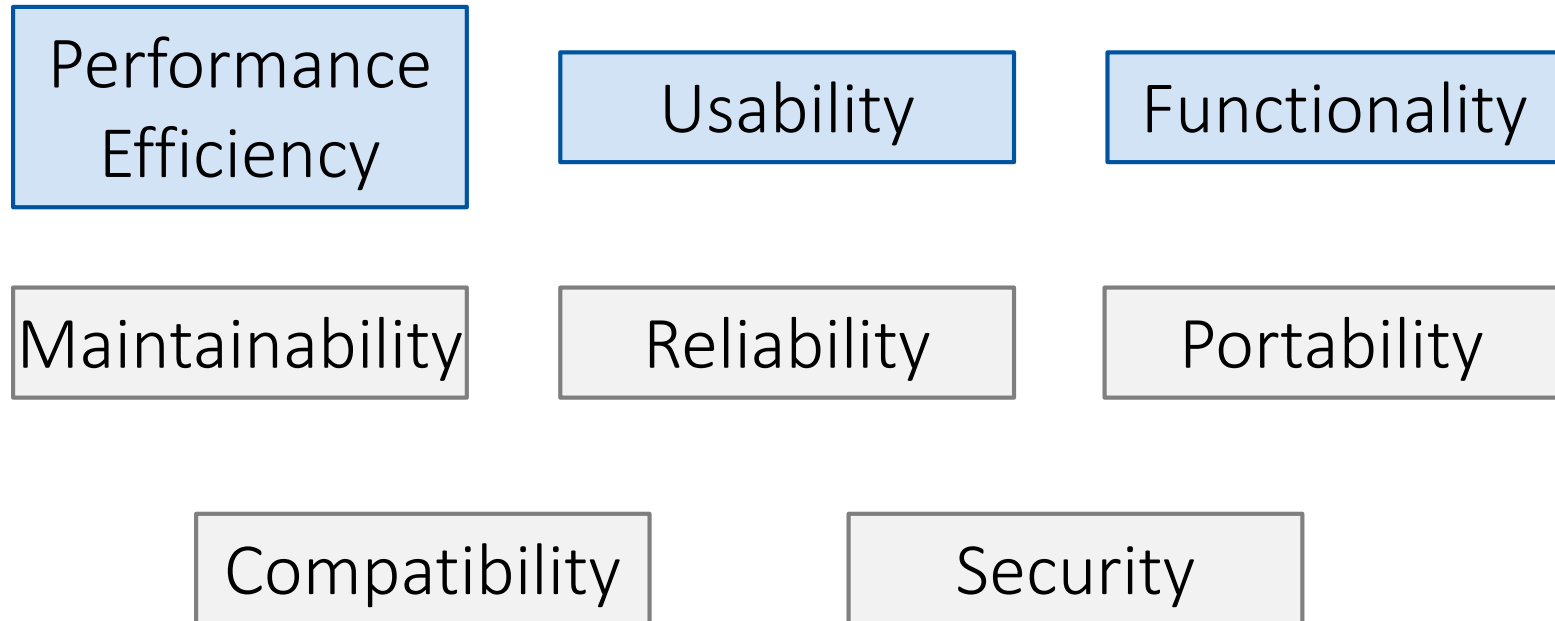
```
Optional<TestRunReport> runTests (  
    TestSuite testSuite,  
    Classpath classpathForSystemUnderTest,  
    Classpath classpathForTests);  
}
```

Lazzer Framework Architecture



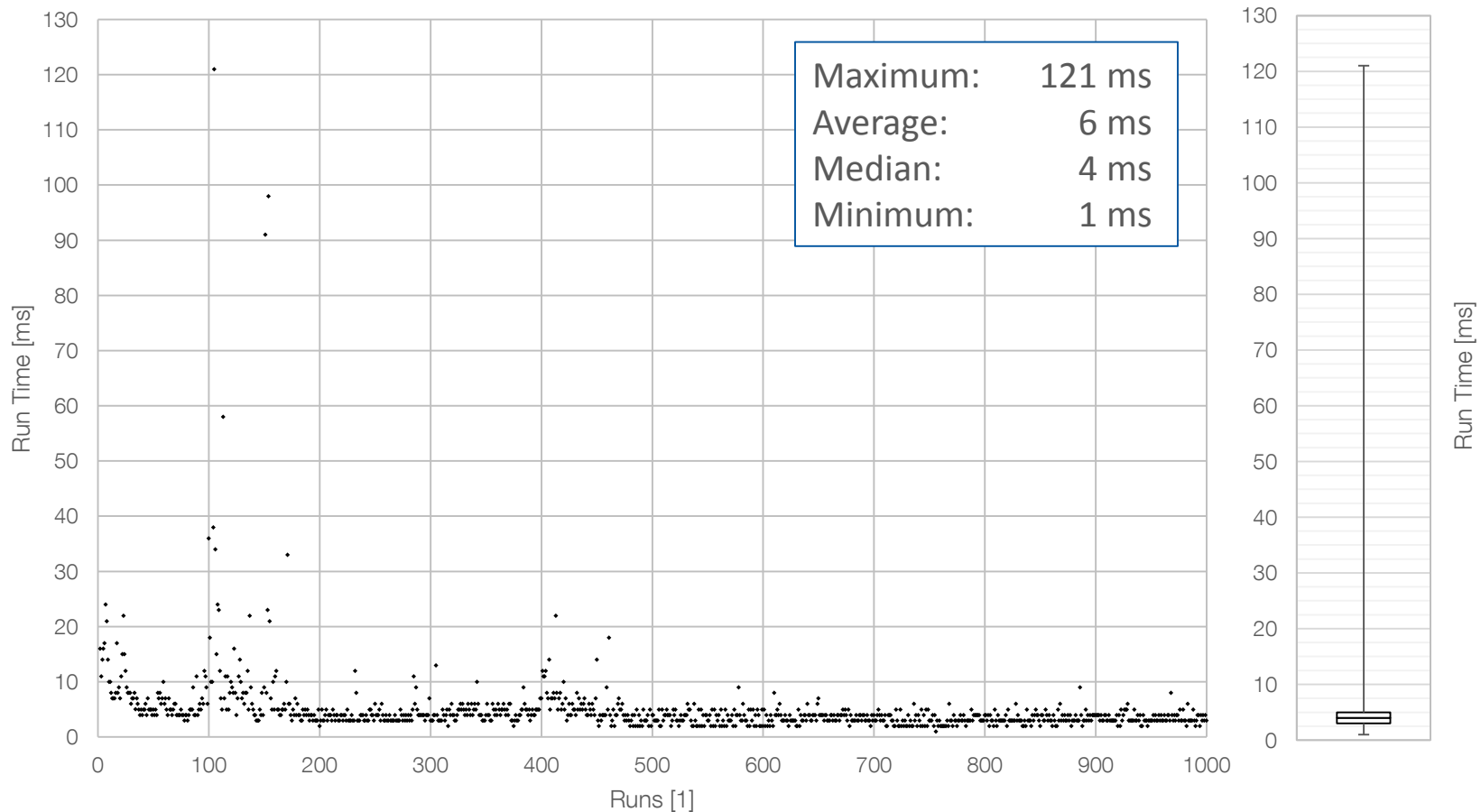
Evaluation

- Software quality assessed according to ISO/IEC 25010
- Aspects:



Evaluation: Performance Efficiency

Lazzer Run Time Evaluation
1,000 runs; 1 outlier (989ms) not considered



- Setup
 - Command Line: deploy framework JAR + extensions
 - Maven: add some lines to project configuration
- Invocation
 - Command Line: execution of a Java application
 - Maven: run “test” phase
- Inspection of **test results**
 - View log output

Evaluation: Functionality

- Working prototype ✓
- Limitations
 - Configuration of data stores difficult
 - Tests of JUnit version (<4) not supported
 - Parameterized tests not recognized
 - Test results are only logged ⇒ extend API for custom reporting
 - No prioritization on method level
 - Not supported order: TestA.x(), TestB.y(), TestA.y()

Future Work on the Prototype

Tasks

- Overcoming current limitations
- Support for parallel testing
- More framework configuration options
- Lazzer needs more tests

Some topics for future research:

- Implementation and evaluation of RTO techniques
- Support for other programming languages
- How to choose the best RTO technique?
- Recommendation for test suite reduction
- Test optimization as a service

Conclusion

Contributions

- Overview of state of research on RTO
 - Framework for RTO
 - Working prototype
- } First step towards an RTO platform

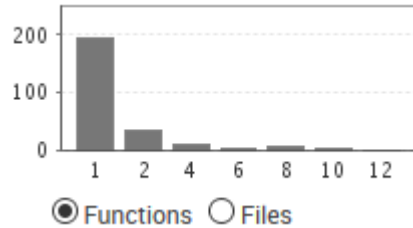
SonarQube Static Code Analysis of Lazzer Framework

- Complexity

Complexity

423 ↘

/Function 1,4 /Class 5,1 /File 5,3



- Rule Compliance

Debt

2h 30min ↗

Issues

4

🚫 Blocker 0
🚨 Critical 0
🔴 Major 4
🟢 Minor 0
🟡 Info 0

- Documentation

Documentation

76,1% ↘

Public API 259 ↗
Pub. Undoc. API 62 ↗

Comments

26,6%

Comment Lines 797 ↘

- Architecture

Lines Of Code

2.201 ↘

Java

Files

80 ↘

Directories

27 ↘

Lines

4.463 ↘

Functions

303 ↗

Classes

83

Statements

660 ↘

Accessors

26 ↘

- Unit Testing

Unit Tests Coverage

15,6%

Line Coverage

15,9%

Condition Coverage

13,5%

Version 51 / 7th December 2015 14:25 Uhr