Christian Plewnia
christian.plewnia@rwth-aachen.de

A Framework for Regression Test Prioritization and Selection

Master Thesis Final Presentation

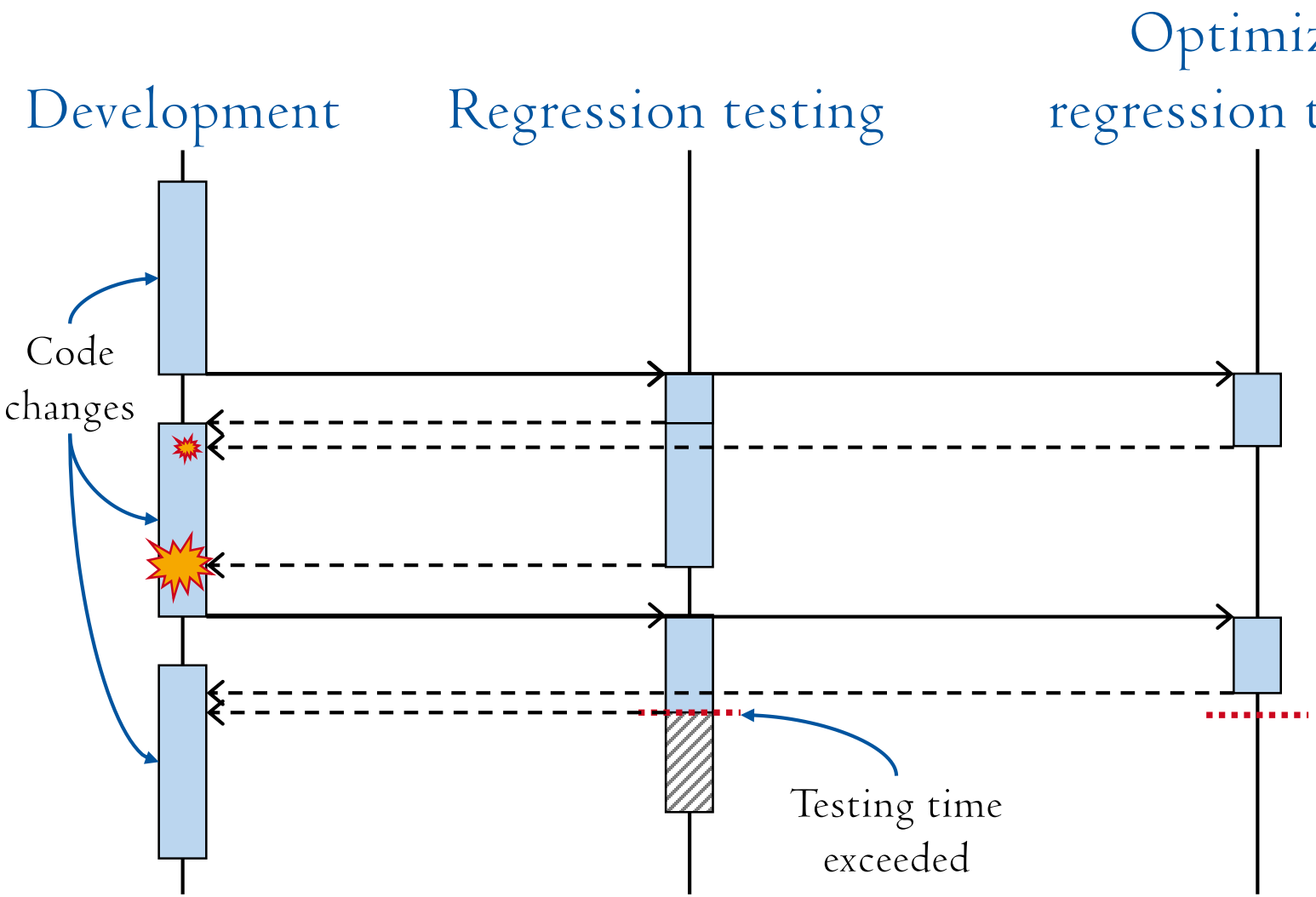
December 9th, 2015

SWC Software Construction | RWTH AACHEN UNIVERSITY

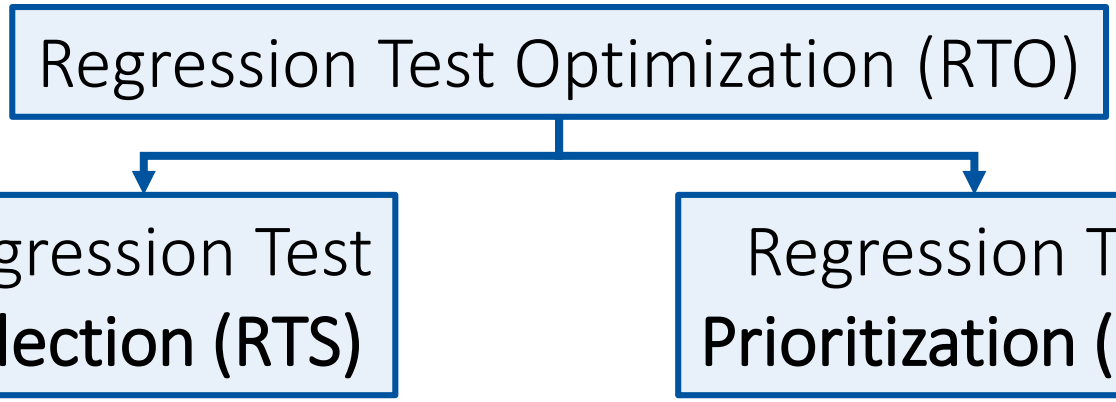
Regressions, Test Automation, and Regression Testing

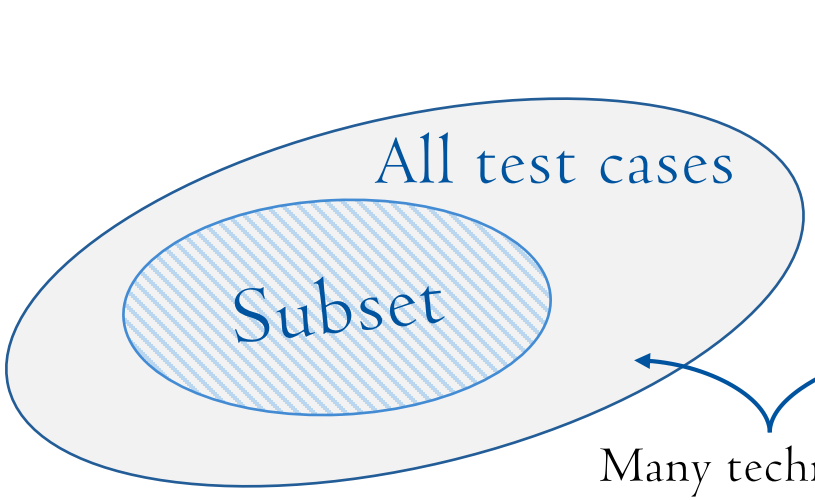
Source: http://www.seguetech.com

Challenge

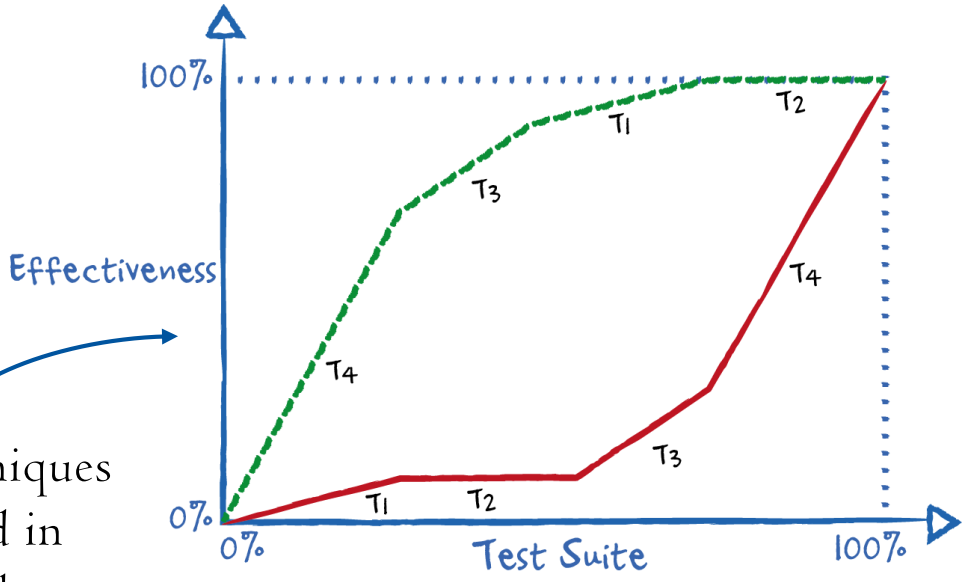

Development
Regression testing
Optimized regression testing
Code changes
Testing time exceeded

Regression Test Optimization





Many techniques described in research



RTO in Practice

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

Challenges of integration

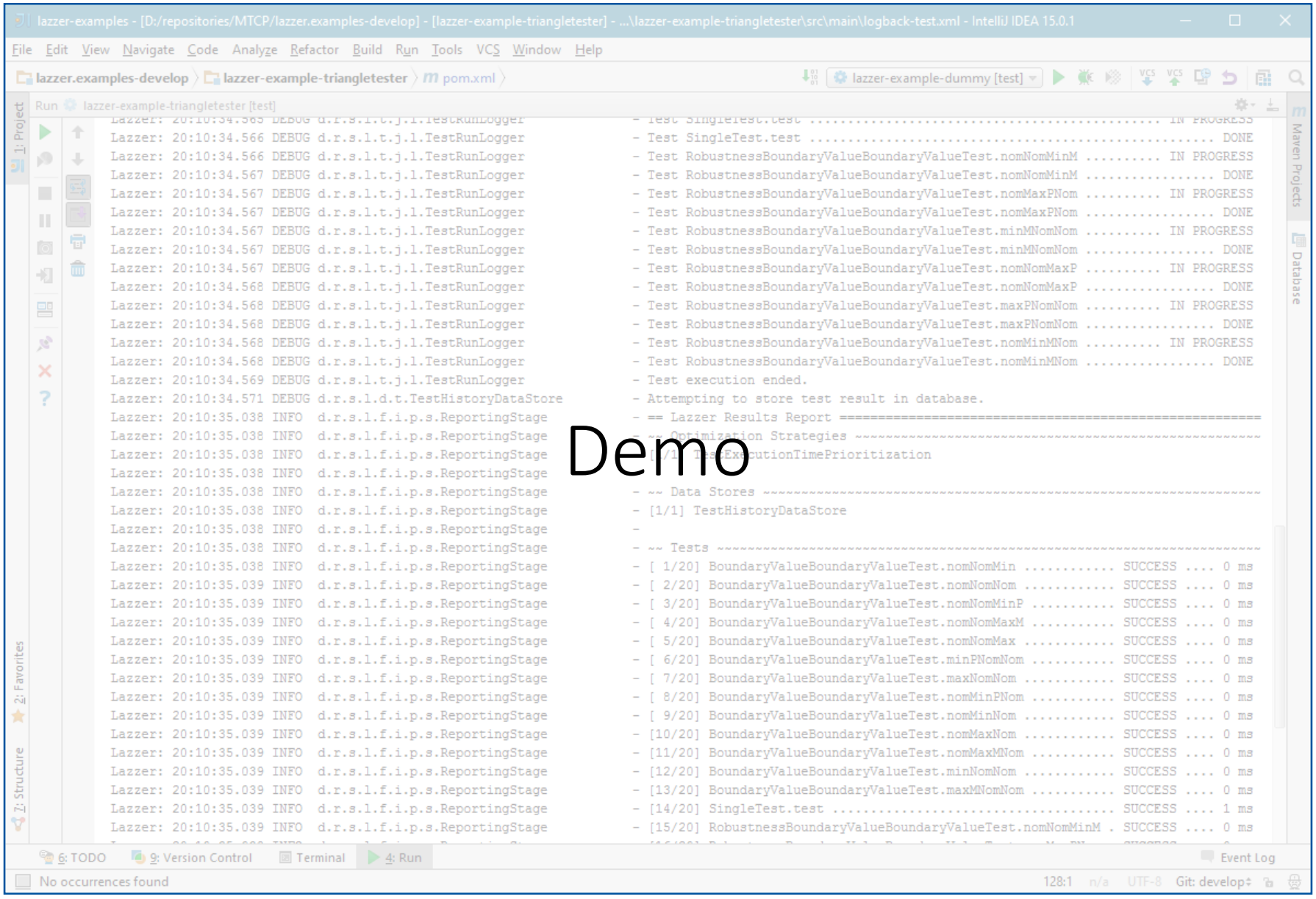
- Integration into testing process
- Optimization requires data gathering
- Execution of optimized test suite

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

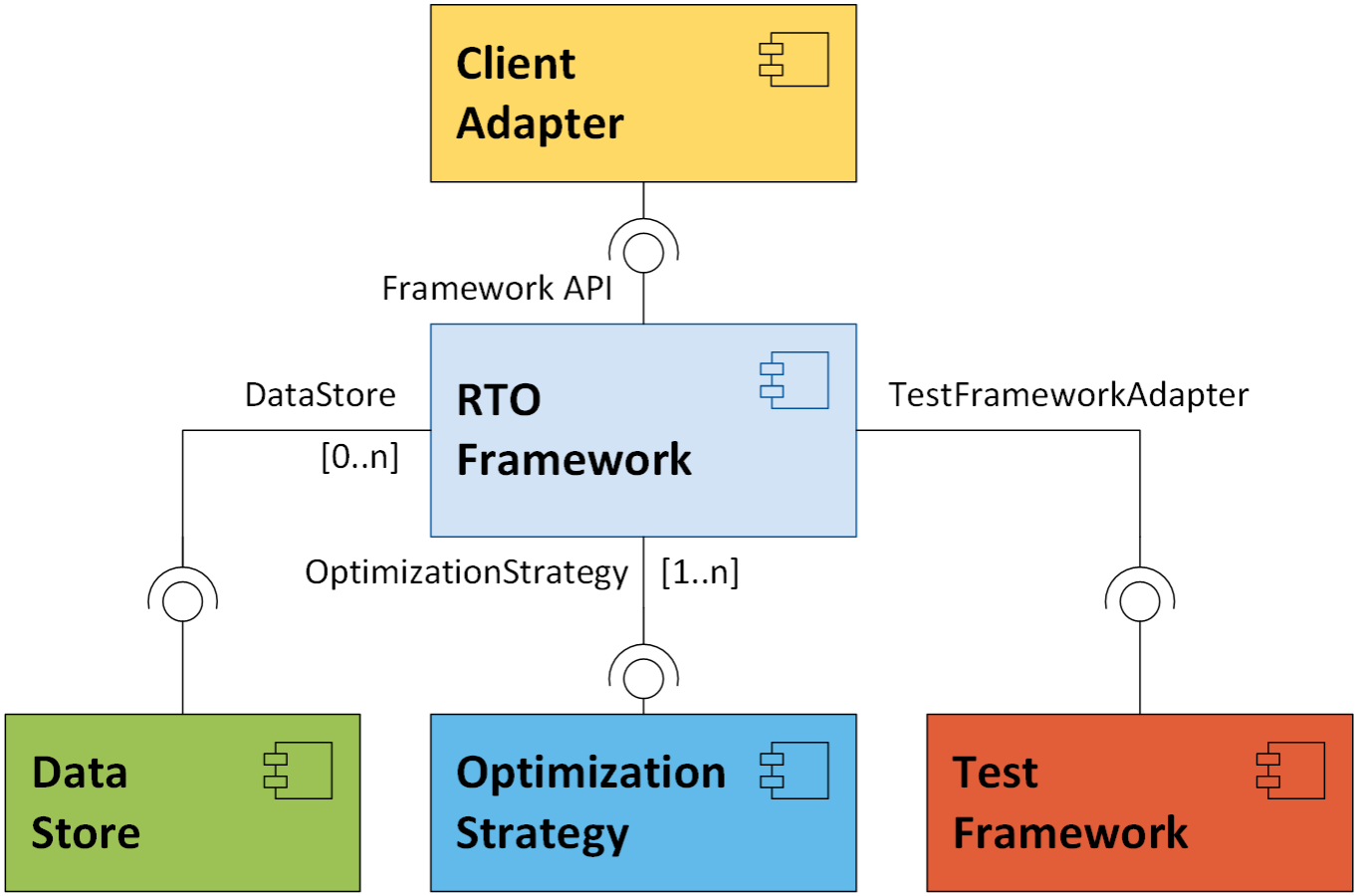
* Real name unknown

✓ available – internal ✗ discontinued

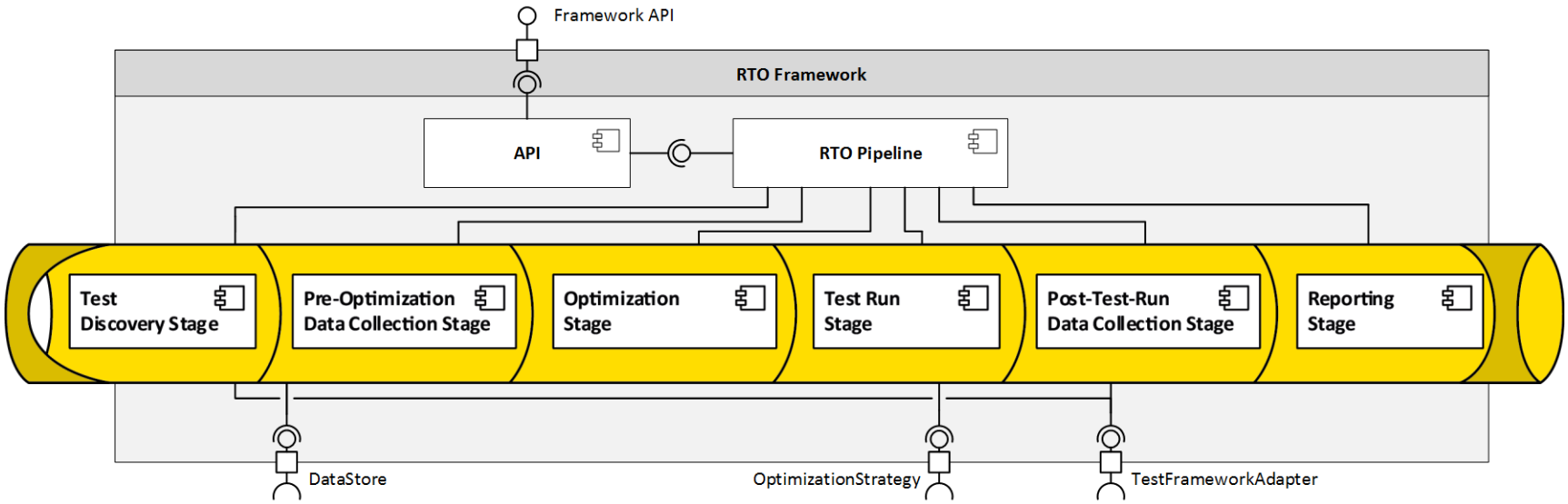


Software Construction | RWTH AACHEN UNIVERSITY

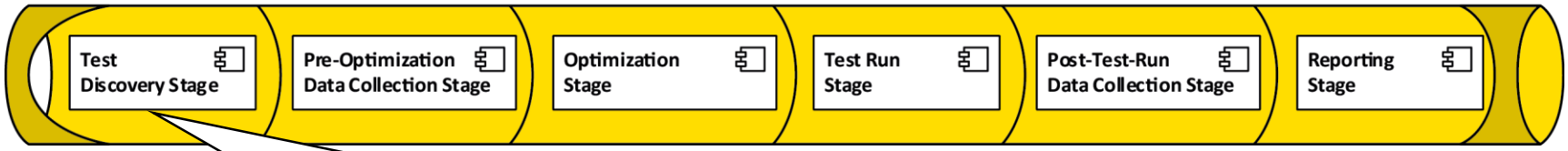
Lazzer Framework Architecture



Lazzer Framework Architecture



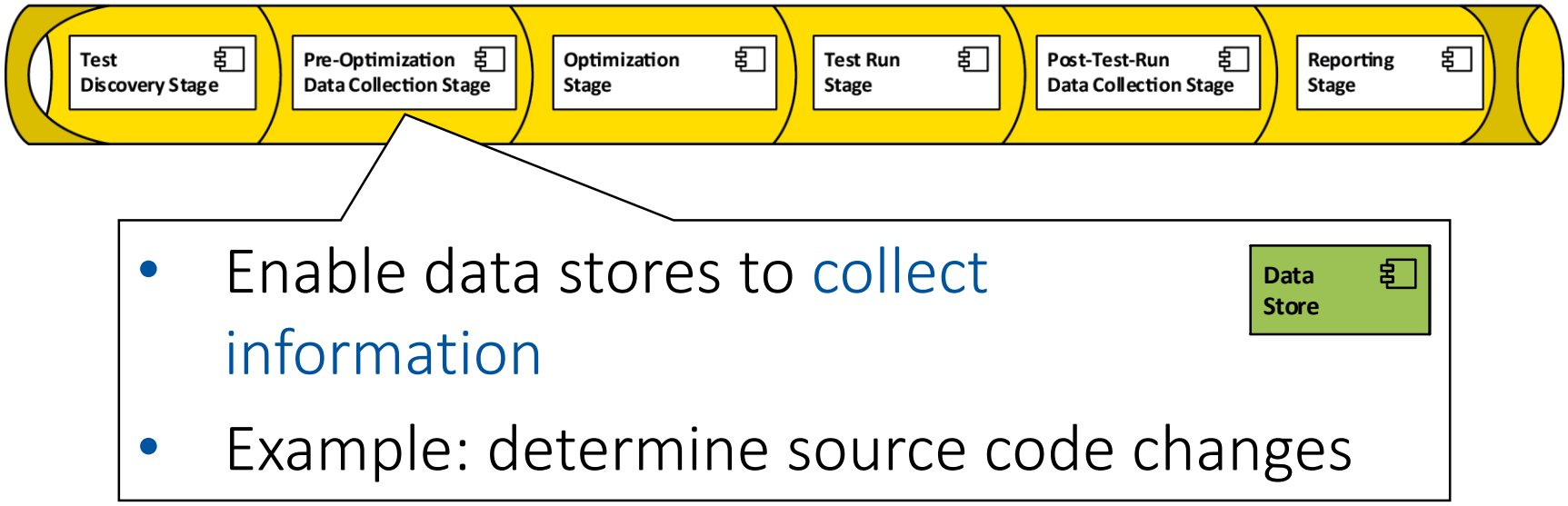
Lazzer Pipeline: Test Discovery



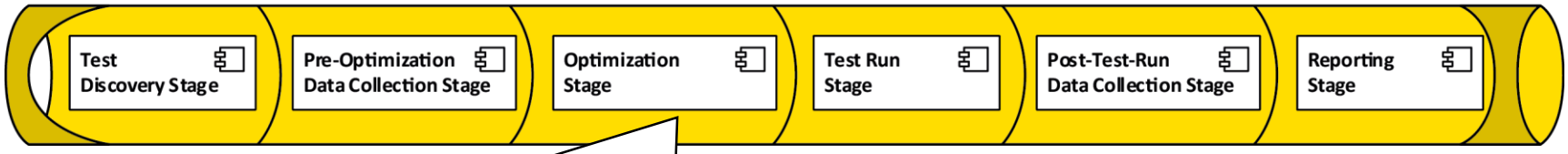
- Find all test cases in the given test suite
- Delegated to the test framework adapter

Test Framework

Lazzer Pipeline: Pre-Optimization Data Collection



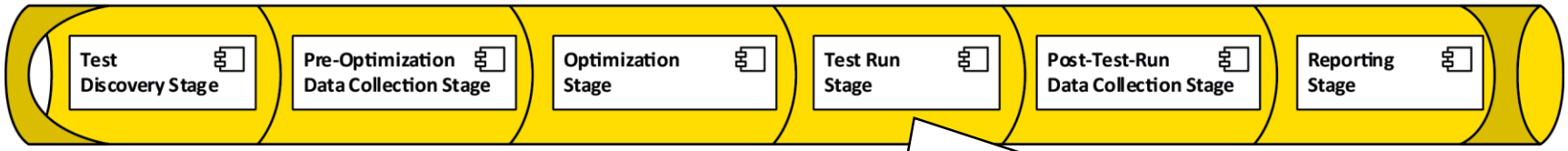
Lazzer Pipeline: Optimization



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

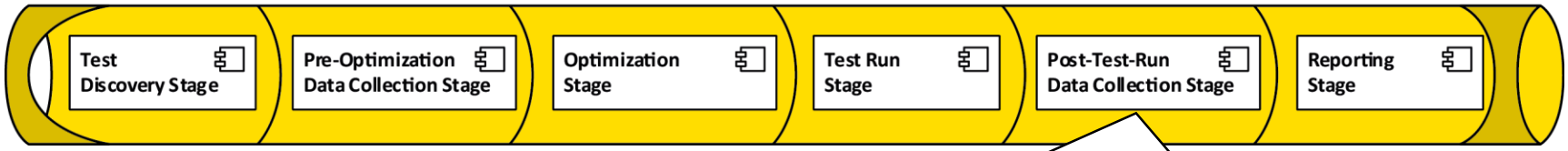
Optimization Strategy

Lazzer Pipeline: Test Run



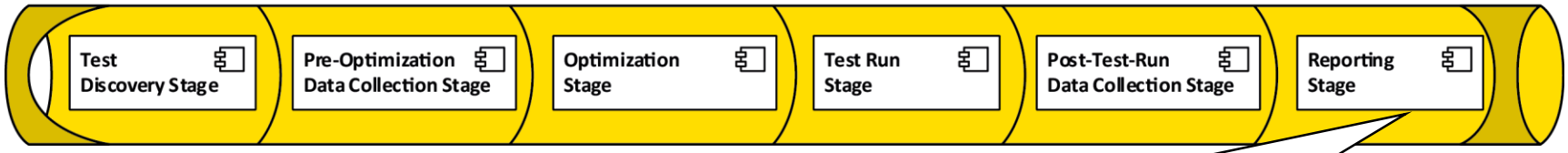
- Run the optimized test suite
- Delegated to the test framework

Lazzer Pipeline: Post-Test-Run Data Collection



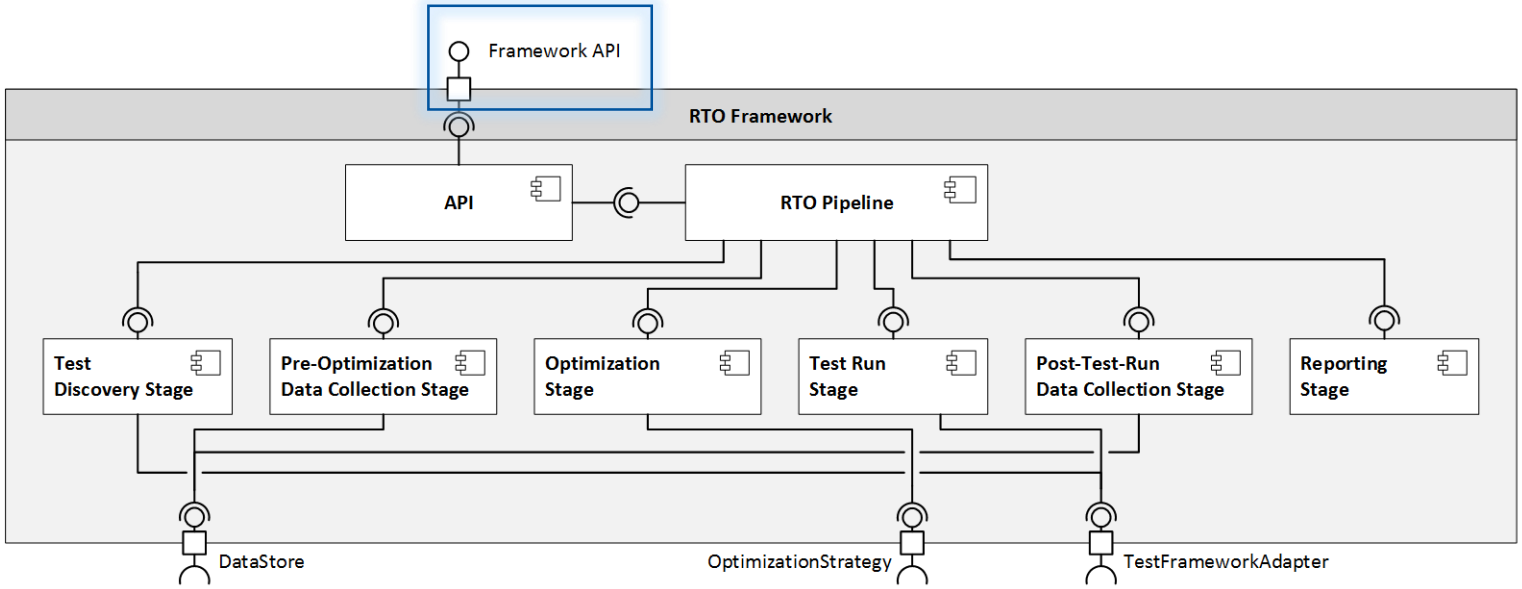
- Enable data stores to collect information
- Example: Save test results in database

Lazzer Pipeline: Reporting



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

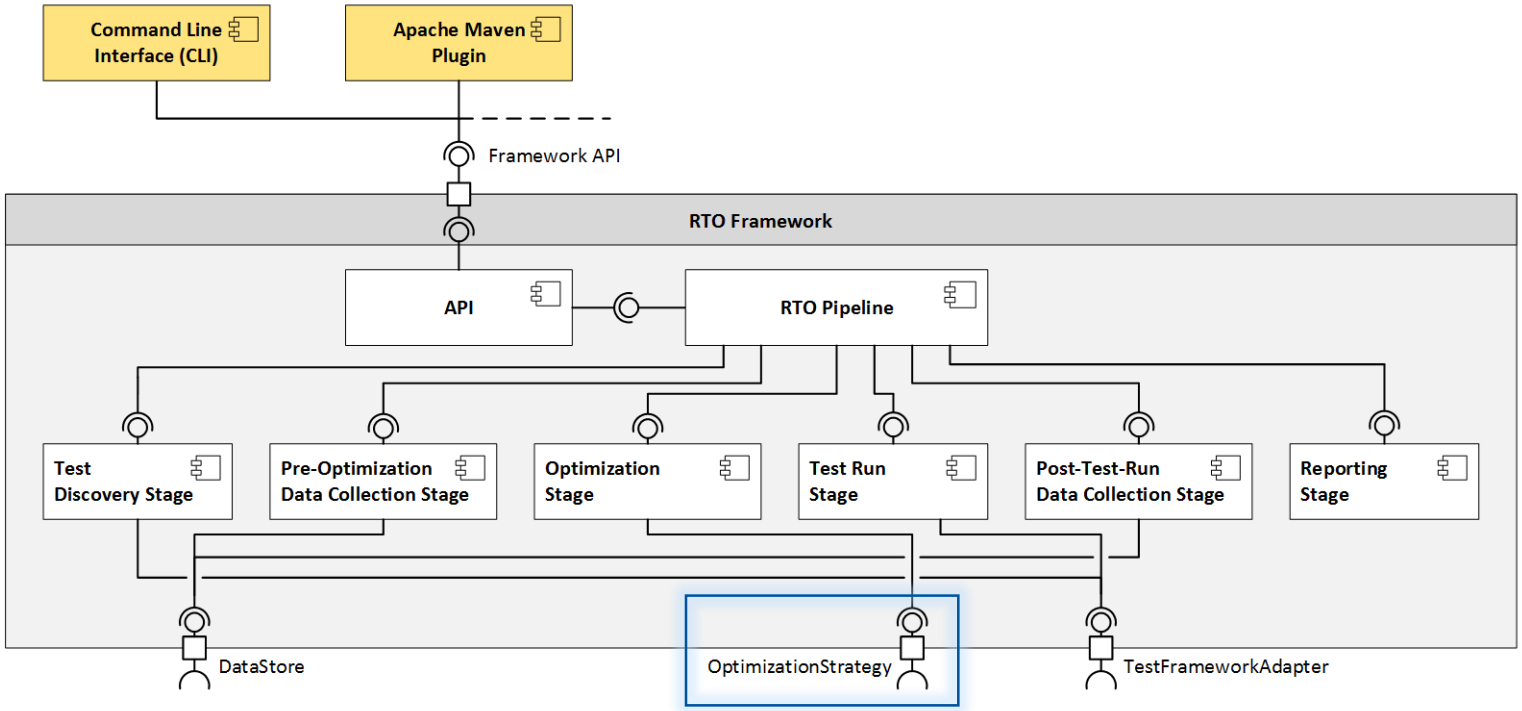
Lazzer Framework Architecture



Evaluation: Running Lazzer via its API


```
// 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

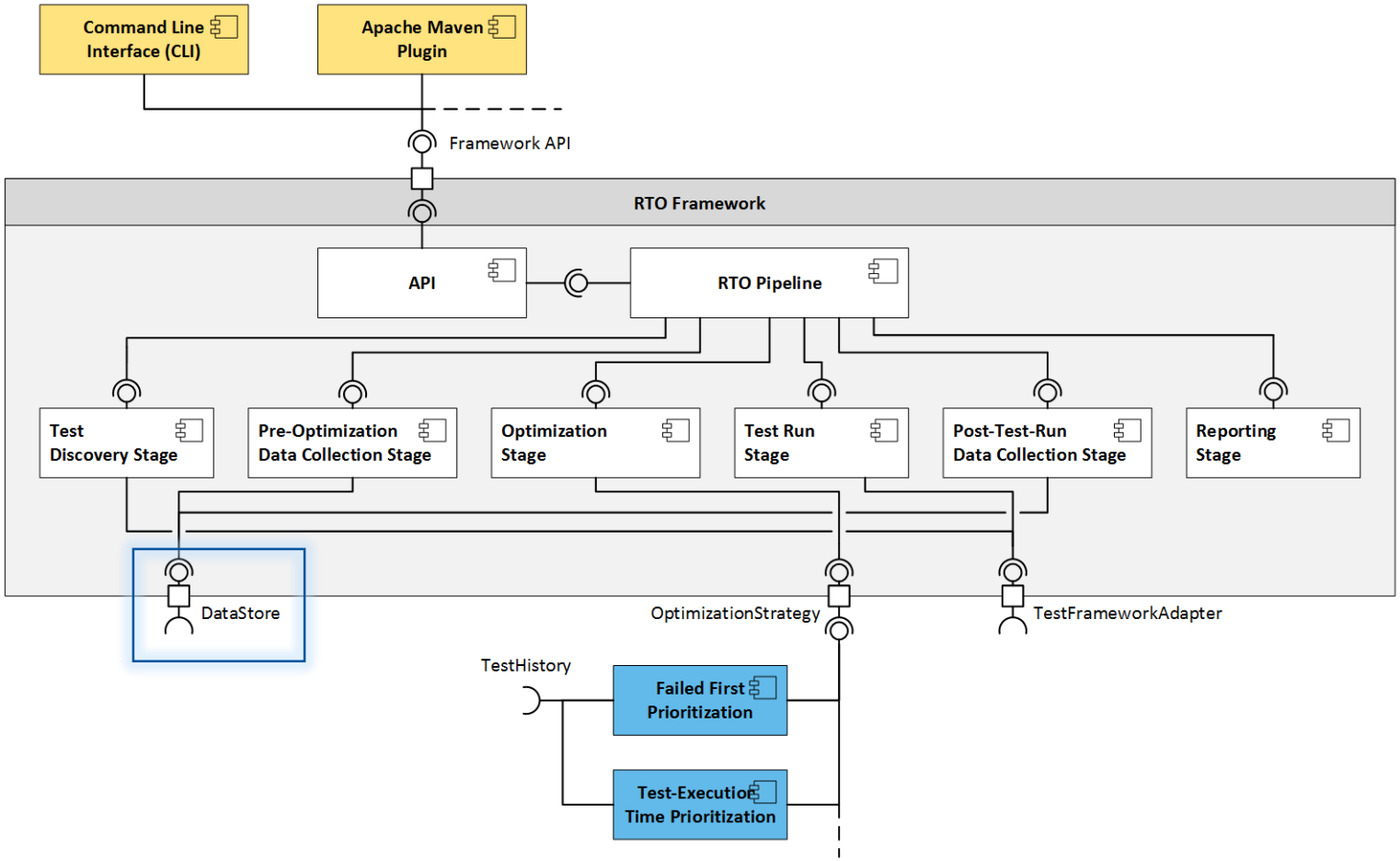


Developing a Lazzer Optimization Strategy

```
public interface OptimizationStrategy {

    TestSuite optimize(TestSuite testSuite);
}
```

Called by the framework to optimize the test suite

Lazzer Framework Architecture



Developing a Lazzer TestRunner


```
public interface DataStore {
```

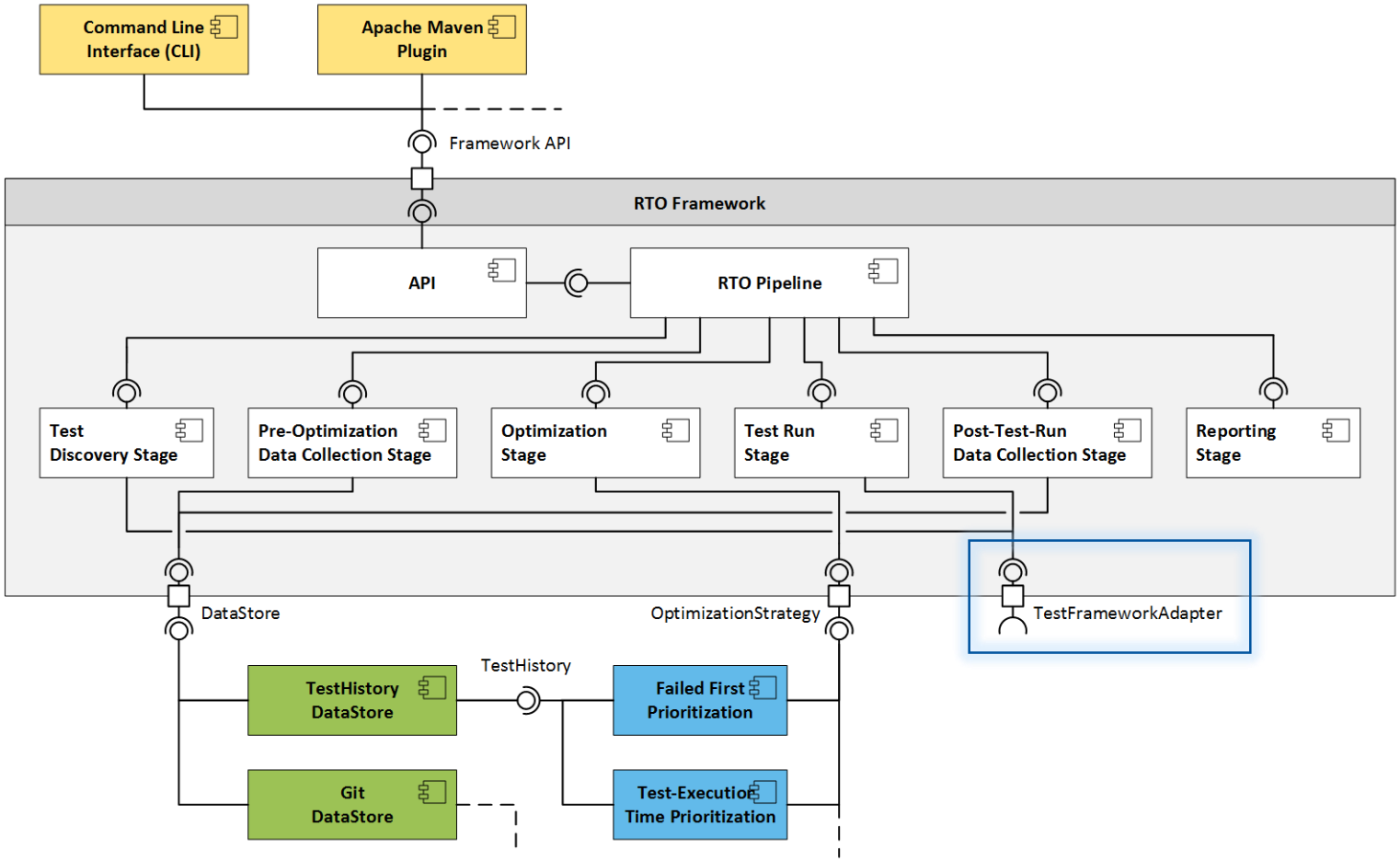
Called by the framework before the optimization

```
    void preOptimizationDataCollection();
```

Called by the framework after the test run

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

Lazzer Framework Architecture



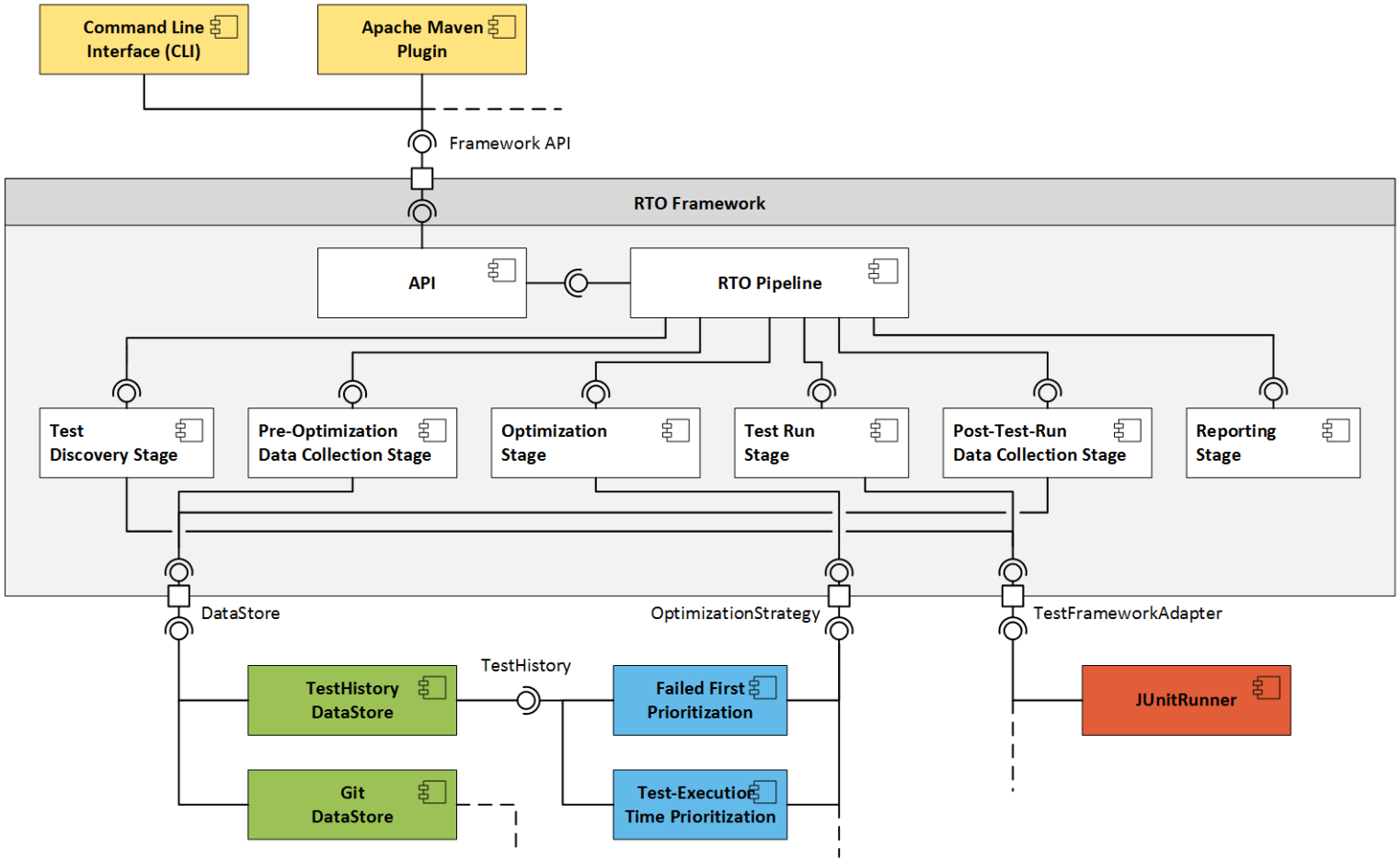
Developing a Lazzer TestFrameworkAdapter


```
public interface TestFrameworkAdapter {

    TestDiscovery provideTestDiscovery(
        Classpath classpathForSystemUnderTest,
        Classpath classpathForTests
    );

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

Finds test cases

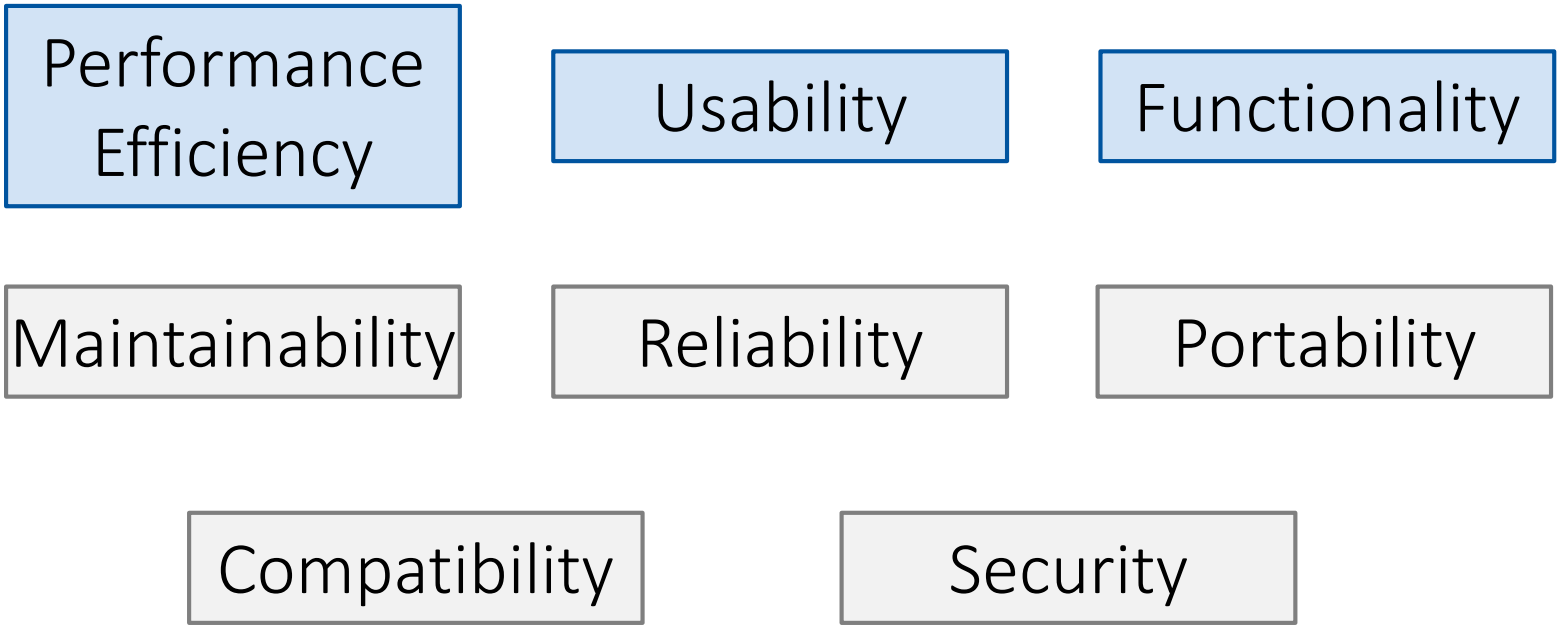
Executes test run

Lazzer Framework Architecture

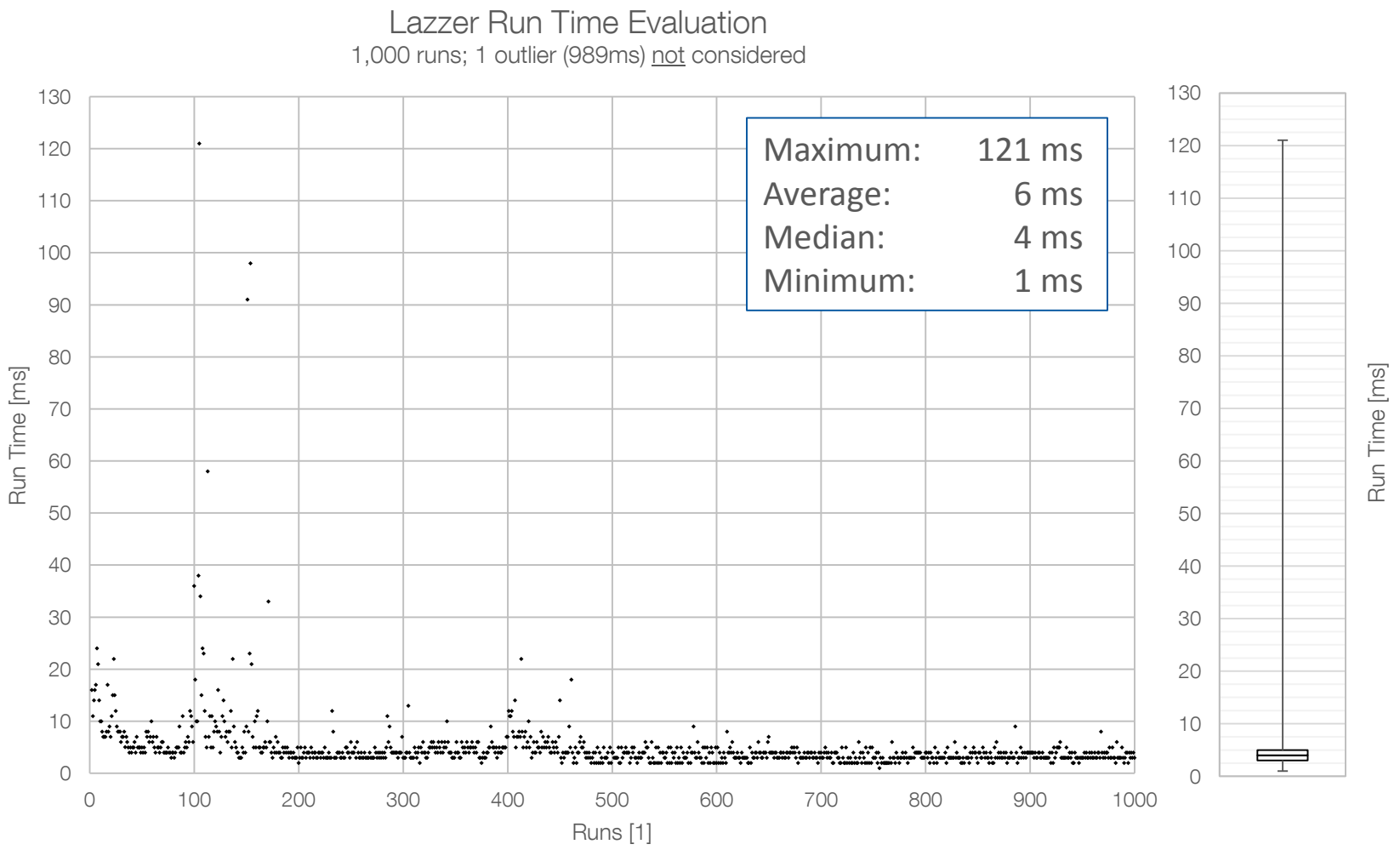


Evaluation

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



Evaluation: Performance Efficiency



Software Construction | RWTH AACHEN UNIVERSITY

Evaluation: Usability

- 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: Test`A.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

Future Research

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

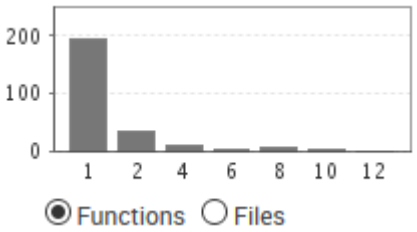
} First step towards an RTO platform

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% | Comments: 26,6%

Public API: 259 | Pub. Undoc. API: 62 | Comment Lines: 797

- ## Architecture

Lines Of Code: 2.201 | Files: 80

Java | 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