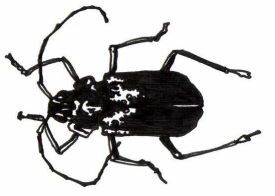


Why RATs don‘t eat Bugs

The MORABIT Approach to Run-Time Testing

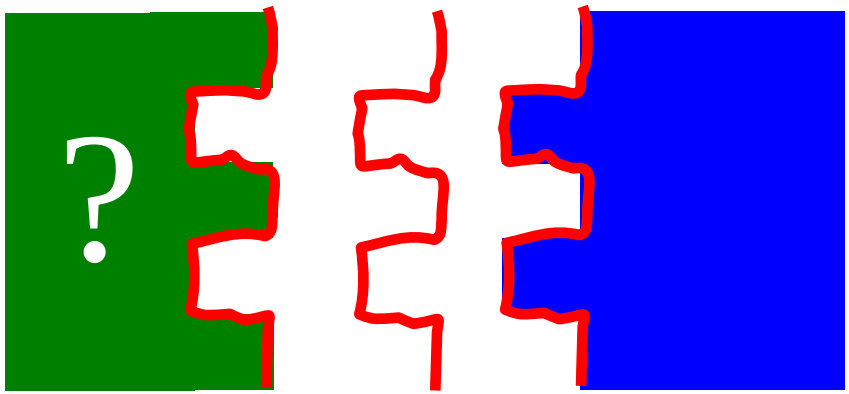
Matthias Merdes
EML Research gGmbH
Heidelberg

EML Research

MORA BIT

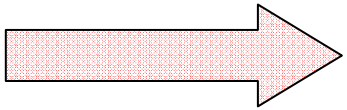
Software: Mobile, Ad-Hoc

- Dynamic formation of software systems
- Components with services (interfaces)
- Implementation of servers unknown



Software: Mobile, Ad-Hoc

- Dynamic formation of software systems
- Components with services (interfaces)
- Implementation of servers unknown



Traditional integration testing impossible

RATs to the Rescue

Resource-

Aware Run-Time

Tests

- Why?
- What?
- How?
- When?

EML Research

MORA BIT

Run-time Tests / BIT - History

- Wang et al. 1998
- Edwards 2001
- Vincent et al. 2002
- Gross 2004

- Our focus:
  - Resource-awareness
  - Middleware support

Do RATs eat Bugs?

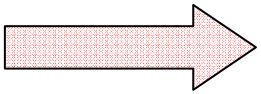
NO!

EML Research

MORA BIT

Goals

- Purpose NOT to find errors
  – NOT development time testing
- Check suitability for *concrete* purpose,
  NOT *general* quality
- Assure common interpretation of contract
- Detect misunderstandings



Example

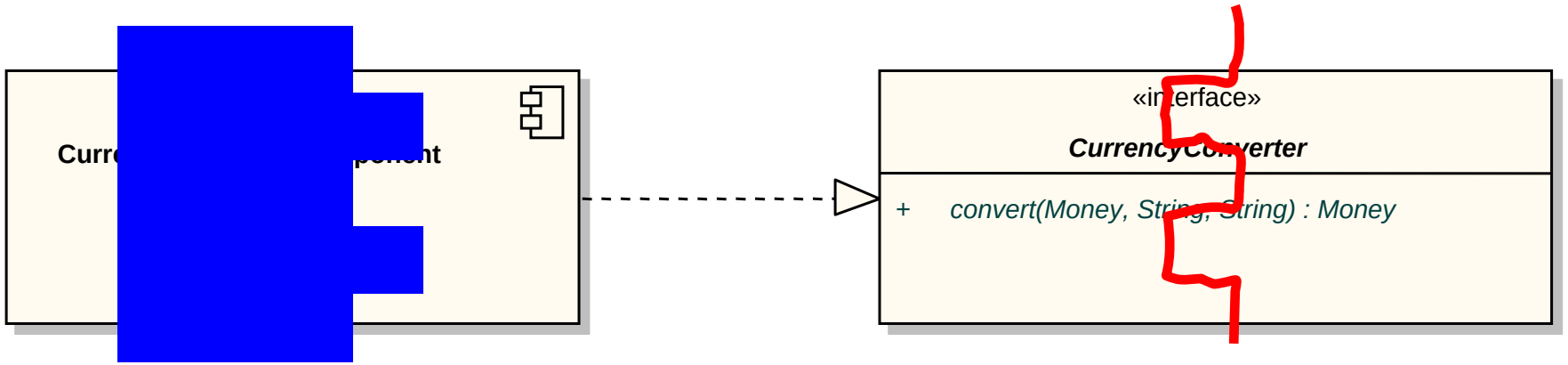
EML Research

MORA BIT

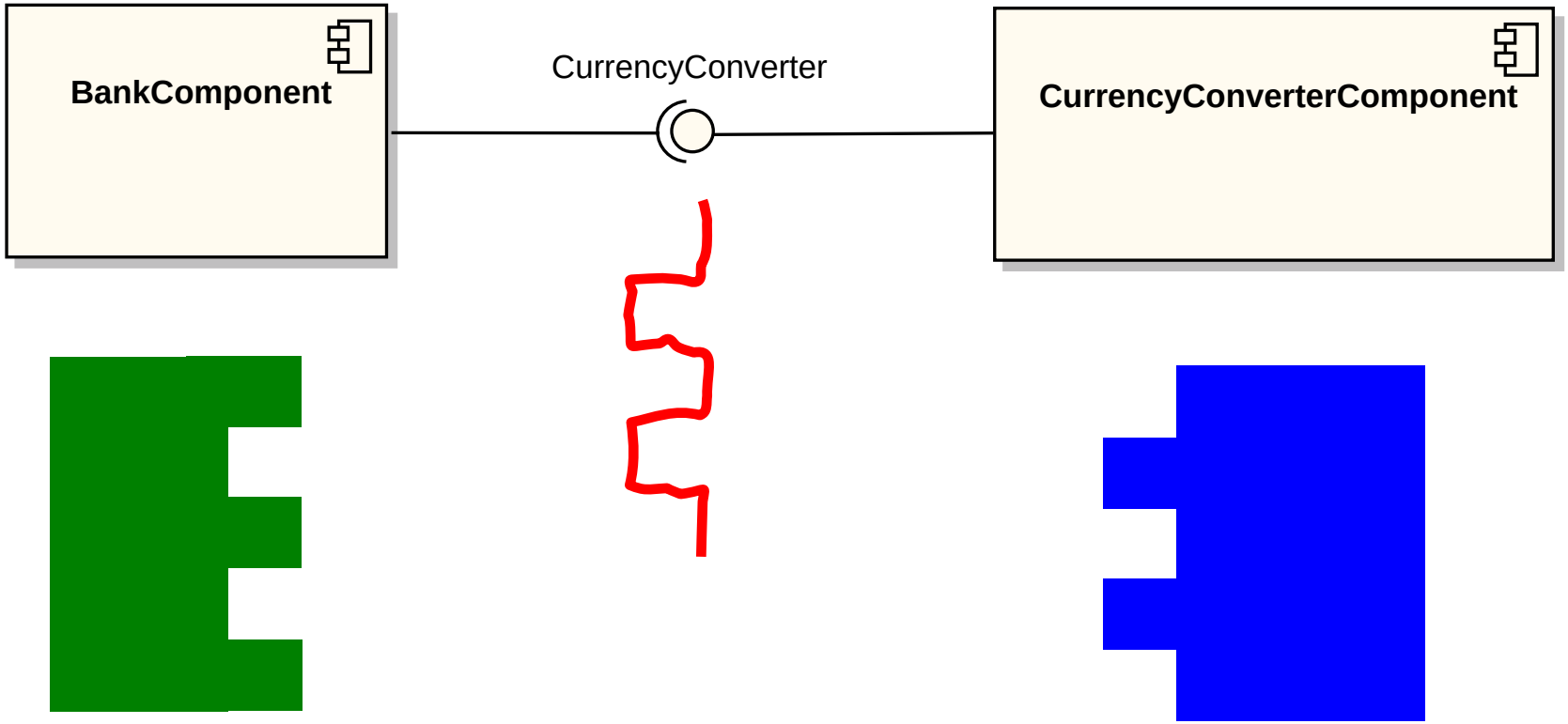
Example: CurrencyConverter

CurrencyConverter.java

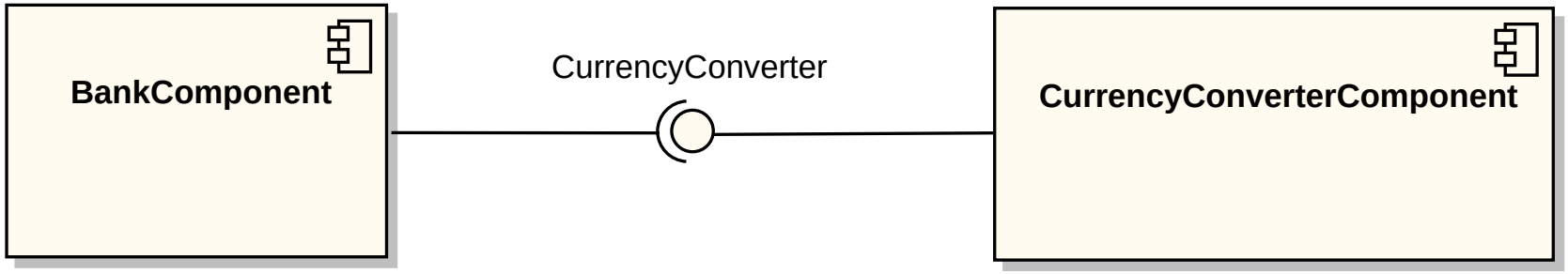
```
package org.morabit;

public interface CurrencyConverter
{
    public Money convert(Money amount, String from, String to);
}
```



Example: CurrencyConverter

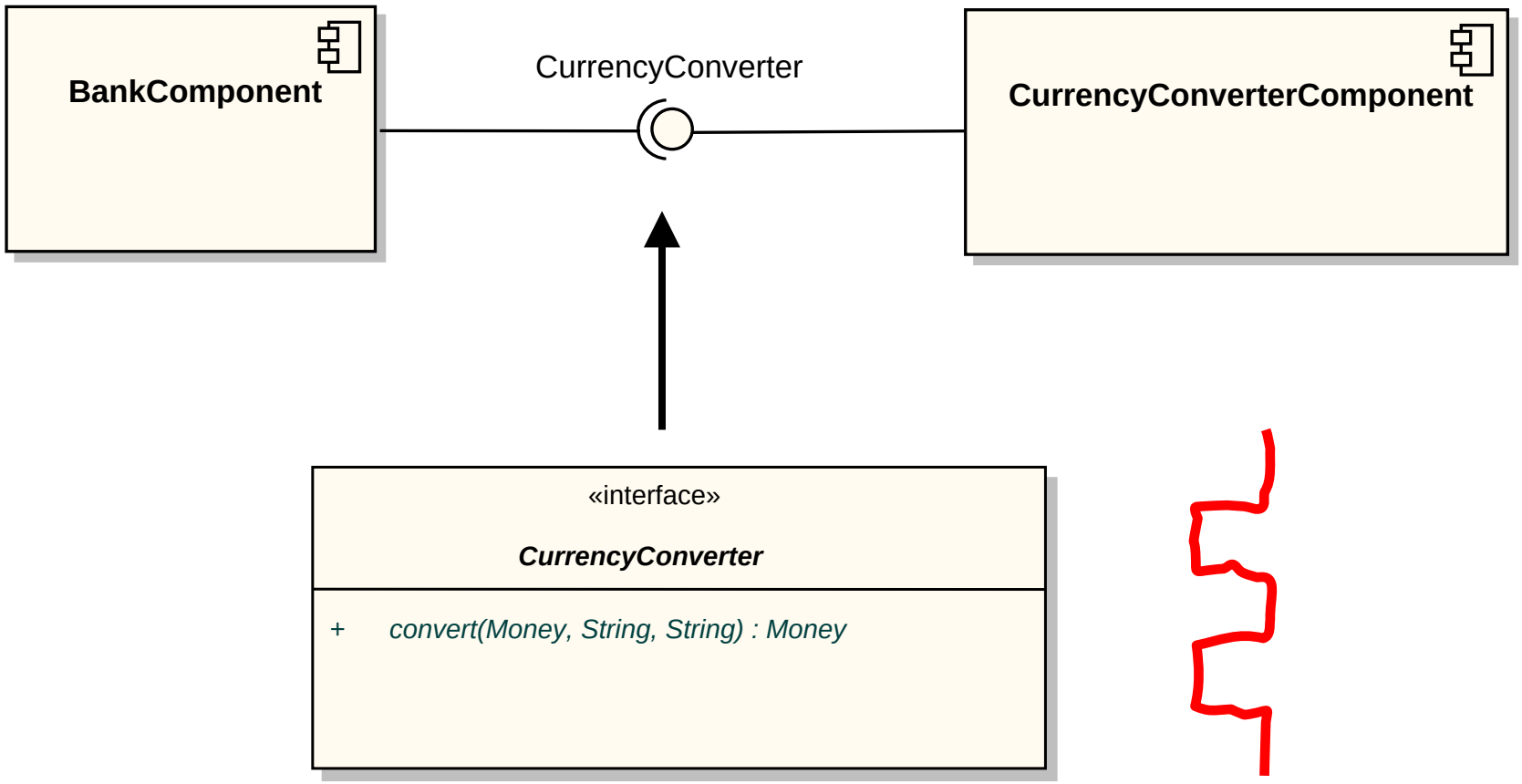


Example: CurrencyConverter



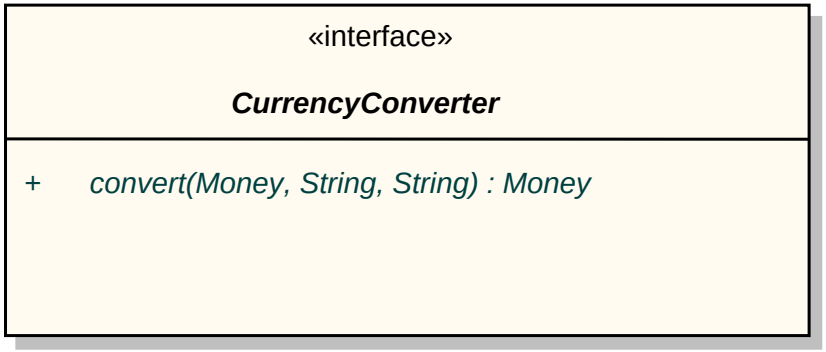
- Server: CurrencyConverterComponent
  - implements service interface
- Client: BankComponent
  - uses service interface

What can go wrong?



What can go wrong?

```
converter.convert(50, „$“, „€“);
```



Misunderstanding: Order

```
converter.convert(50, „$“, „€“);
```

EML Research

MORA BIT

Misunderstanding: Order

```
converter.convert(50, „€“, „$“);
```

EML Research

MORA BIT

Misunderstanding: Units

```
converter.convert(50, „$“, „€“);
```

or

```
converter.convert(50, „USD“, „EUR“);
```

or

```
converter.convert(50, „Dollar“, „Euro“);
```

?

EML Research

MORA BIT

CNN - NASA's metric confusion caused Mars orbiter loss - September 30, 1...

File Edit View Bookmarks Widgets Feeds Tools Help

http://www.cnn.com/TECH/space/9909/30/mars.metric/

Google

CNN.com

sci-tech > space > story page

MAIN PAGE
WORLD
U.S.
LOCAL
POLITICS
WEATHER
BUSINESS
SPORTS
TECHNOLOGY
SPACE
HEALTH
ENTERTAINMENT
BOOKS
TRAVEL
FOOD
ARTS & STYLE
NATURE
IN-DEPTH
ANALYSIS
myCNN
Headline News brief

exploringmars

in-depth specials

NASA's metric confusion caused Mars orbiter loss

September 30, 1999
Web posted at: 1:46 p.m. EDT (1746 GMT)

(CNN) -- NASA lost a $125 million Mars orbiter because one engineering team used metric units while another used English units for a key spacecraft operation, according to a review finding released Thursday.

NASA's Climate Orbiter was lost September 23, 1999

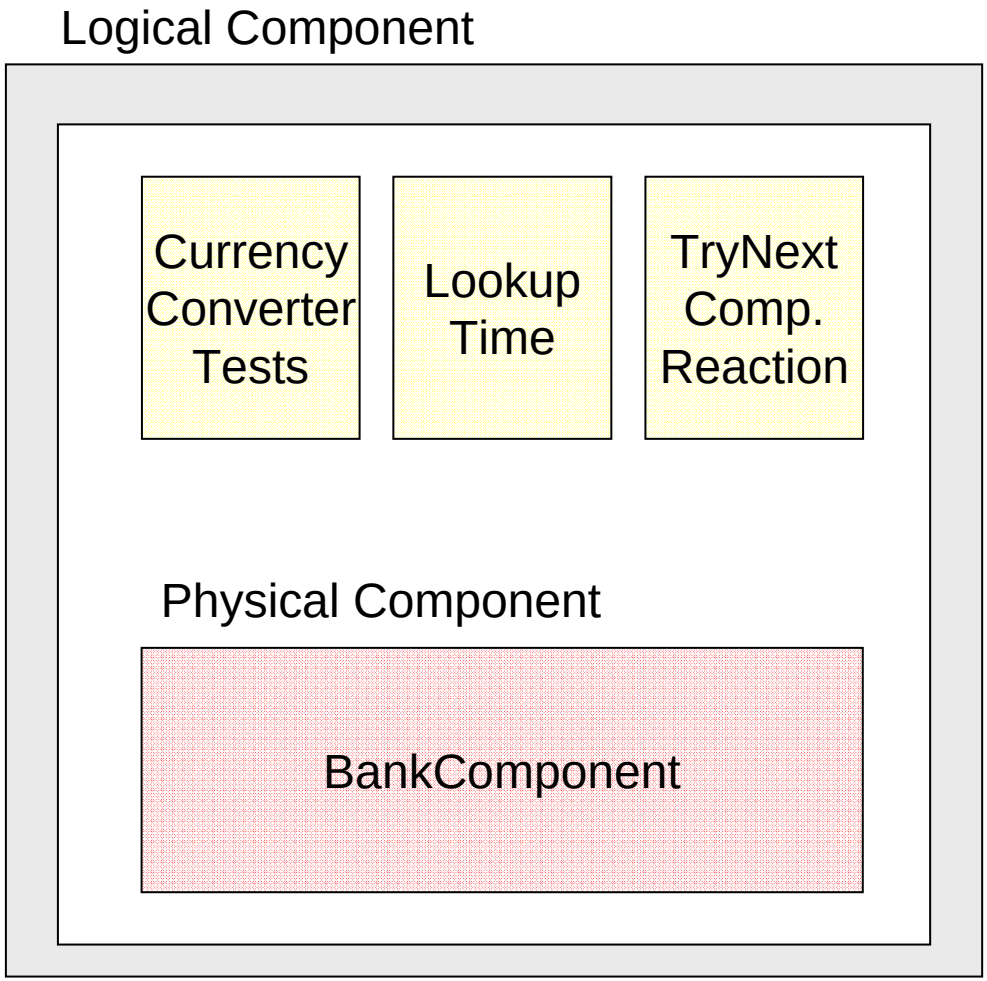
Misunderstanding: Accuracy

expect 5 digits of accuracy,

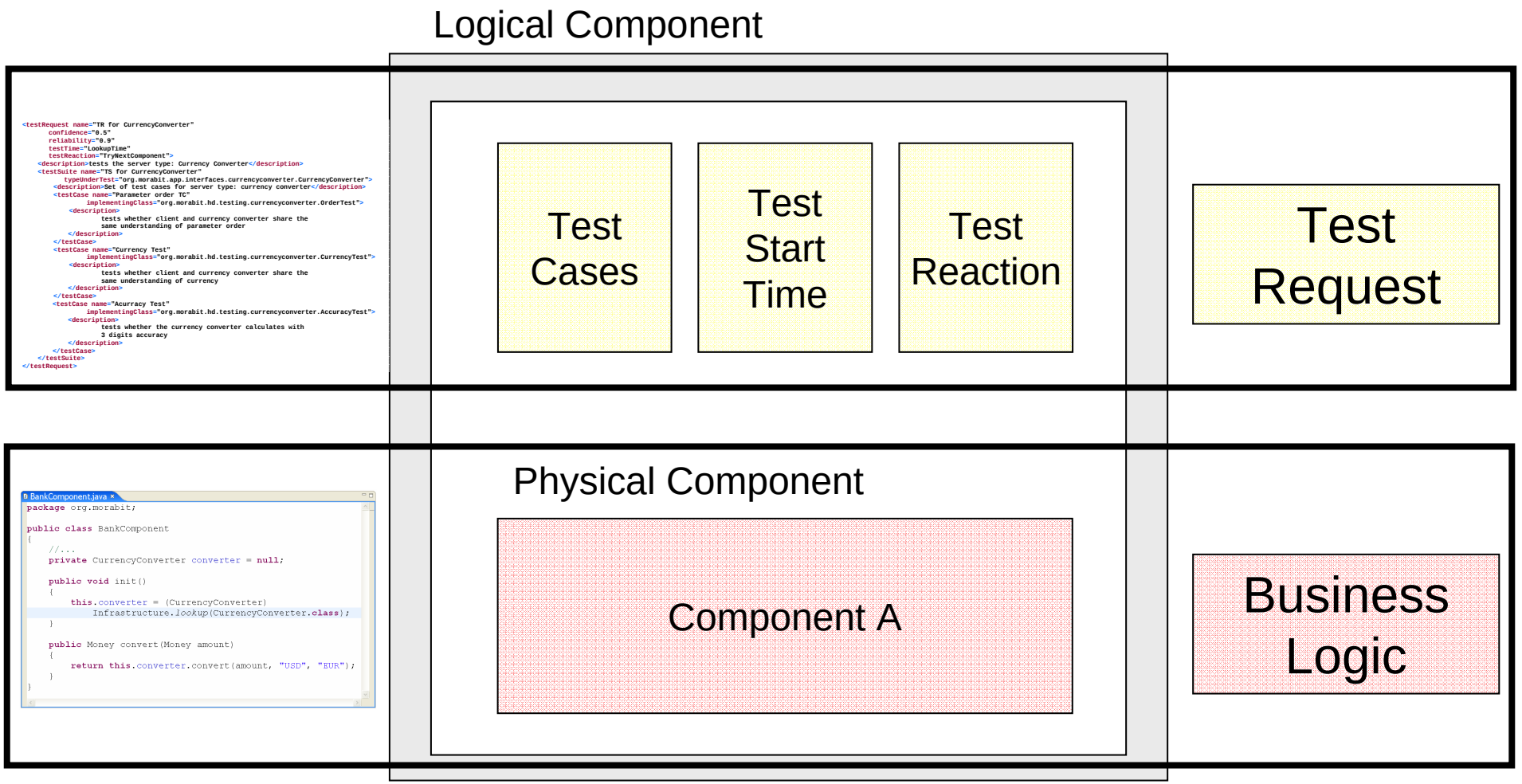
but only get 3...

- Many sources of misunderstandings
- Better check compatibility at run-time
- But: How?

Built-In Tests (BIT)



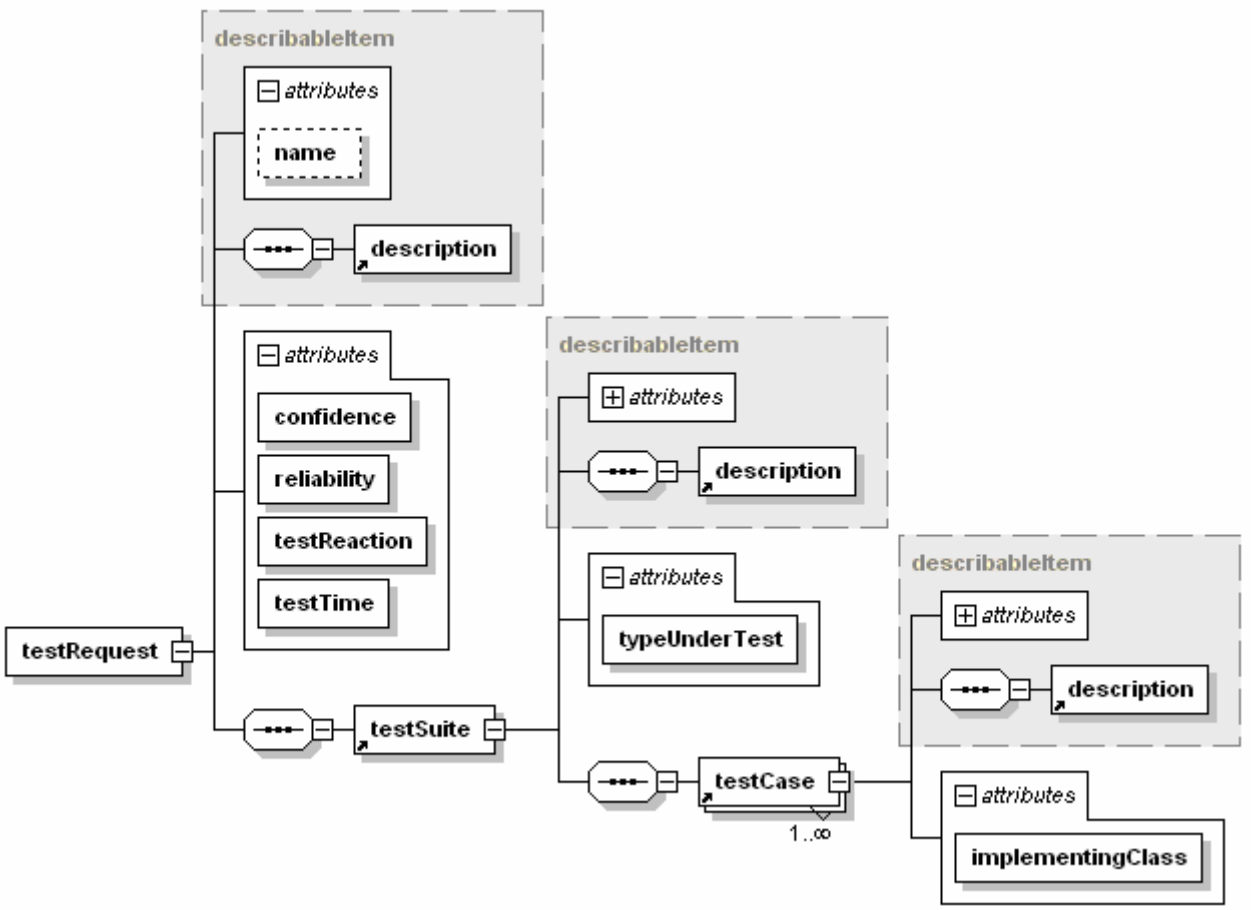
Built-In Tests (BIT)



EML Research

MORA BIT

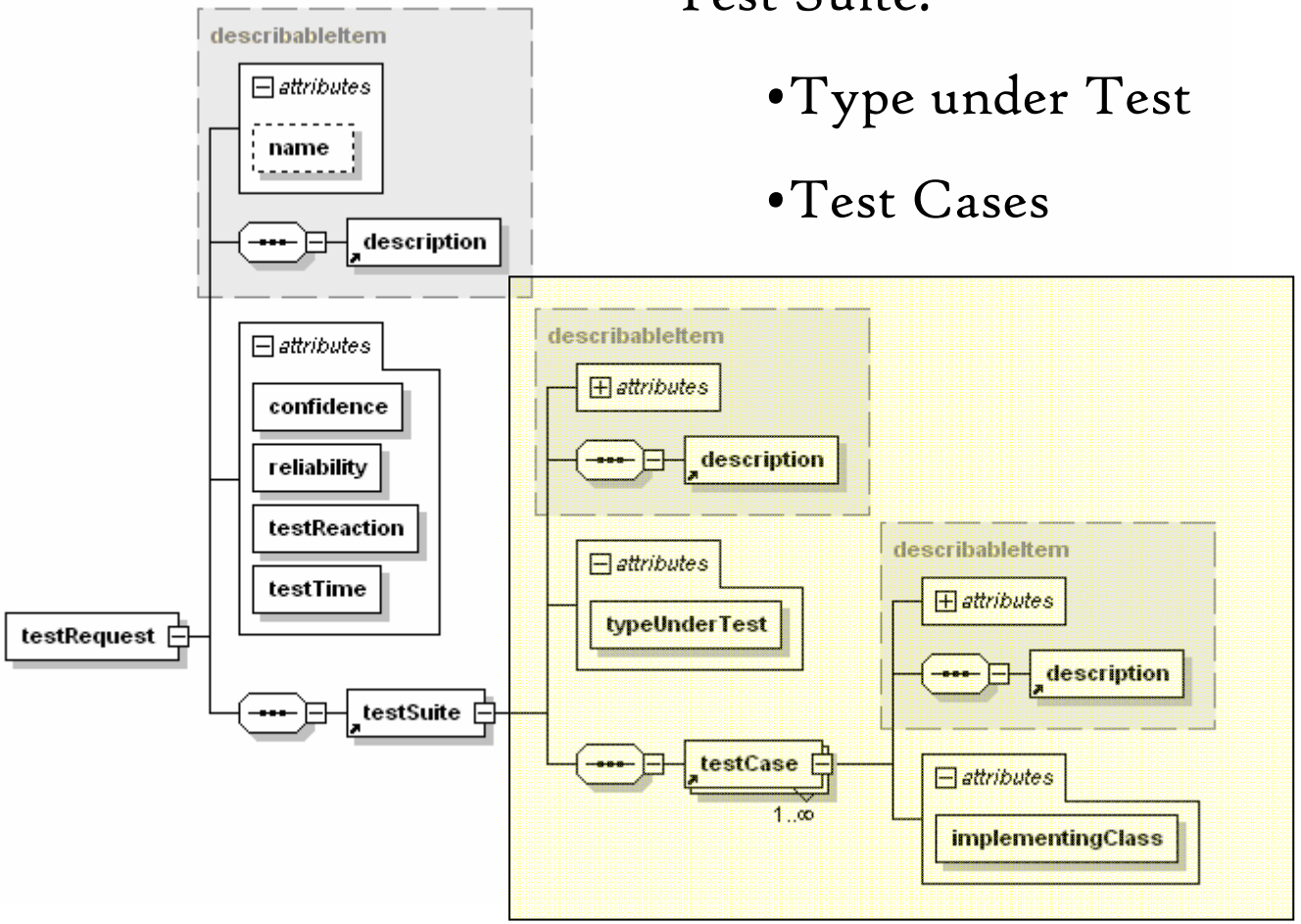
Test Request



EML Research

MORA BIT

Test Request



Test Suite:

- Type under Test
- Test Cases

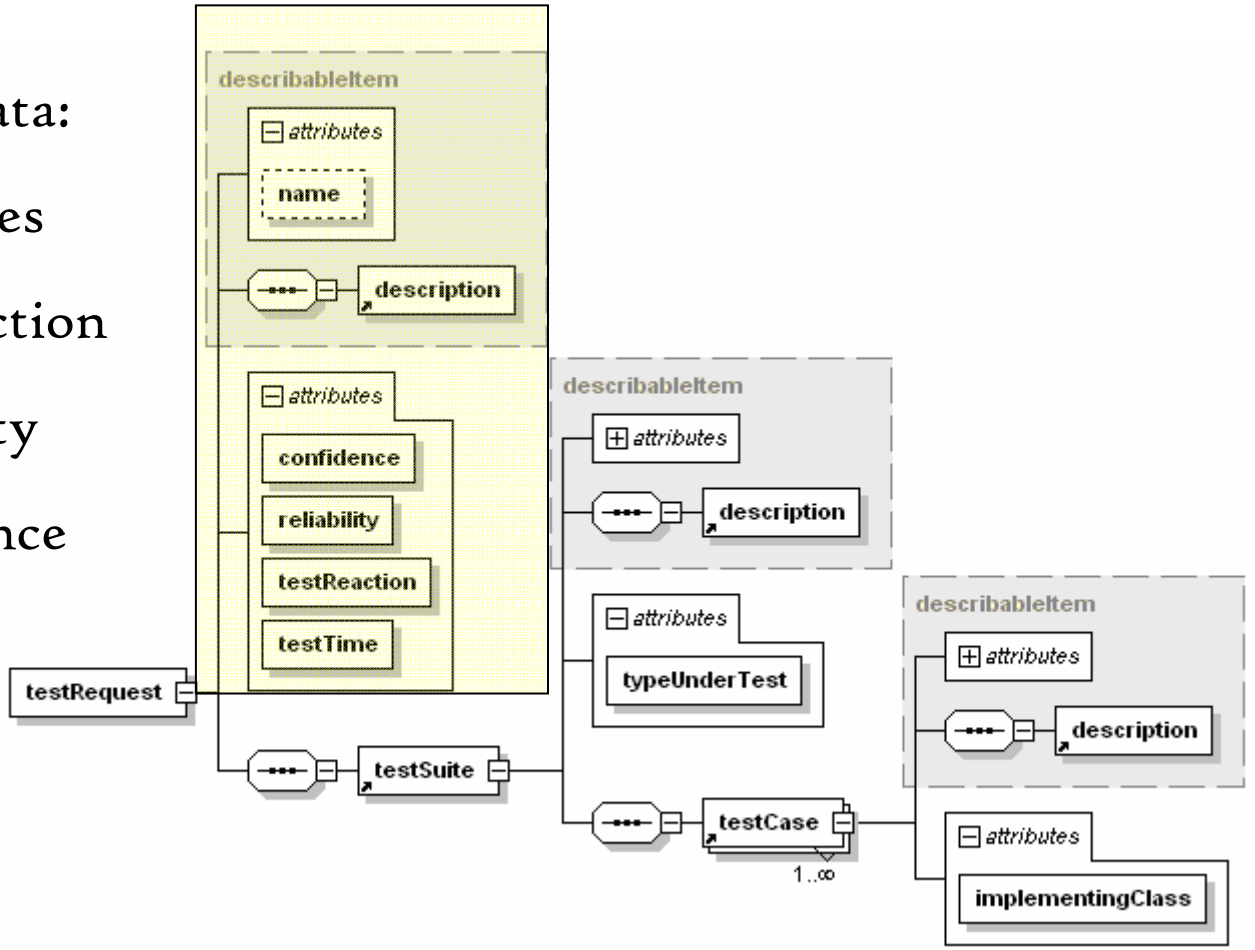
EML Research

MORA BIT

Test Request

Test Meta-Data:

- Test times
- Test reaction
- Reliability
- Confidence



EML Research

MOBIT

Test Request: Example

Test Meta-Data

3 Test Cases

```
<testRequest name="TR for CurrencyConverter"
    confidence="0.5"
    reliability="0.9"
    testTime="LookupTime"
    testReaction="TryNextComponent">
  <description>tests the server type: Currency Converter</description>
  <testSuite name="TS for CurrencyConverter"
        typeUnderTest="org.morabit.app.interfaces.currencyconverter.CurrencyConverter">
    <description>Set of test cases for server type: currency converter</description>
    <testCase name="Parameter order TC"
          implementingClass="org.morabit.hd.testing.currencyconverter.OrderTest">
      <description>
          tests whether client and currency converter share the
          same understanding of parameter order
      </description>
    </testCase>
    <testCase name="Currency Test"
          implementingClass="org.morabit.hd.testing.currencyconverter.CurrencyTest">
      <description>
          tests whether client and currency converter share the
          same understanding of currency
      </description>
    </testCase>
    <testCase name="Acurracy Test"
          implementingClass="org.morabit.hd.testing.currencyconverter.AccuracyTest">
      <description>
          tests whether the currency converter calculates with
          3 digits accuracy
      </description>
    </testCase>
  </testSuite>
</testRequest>
```

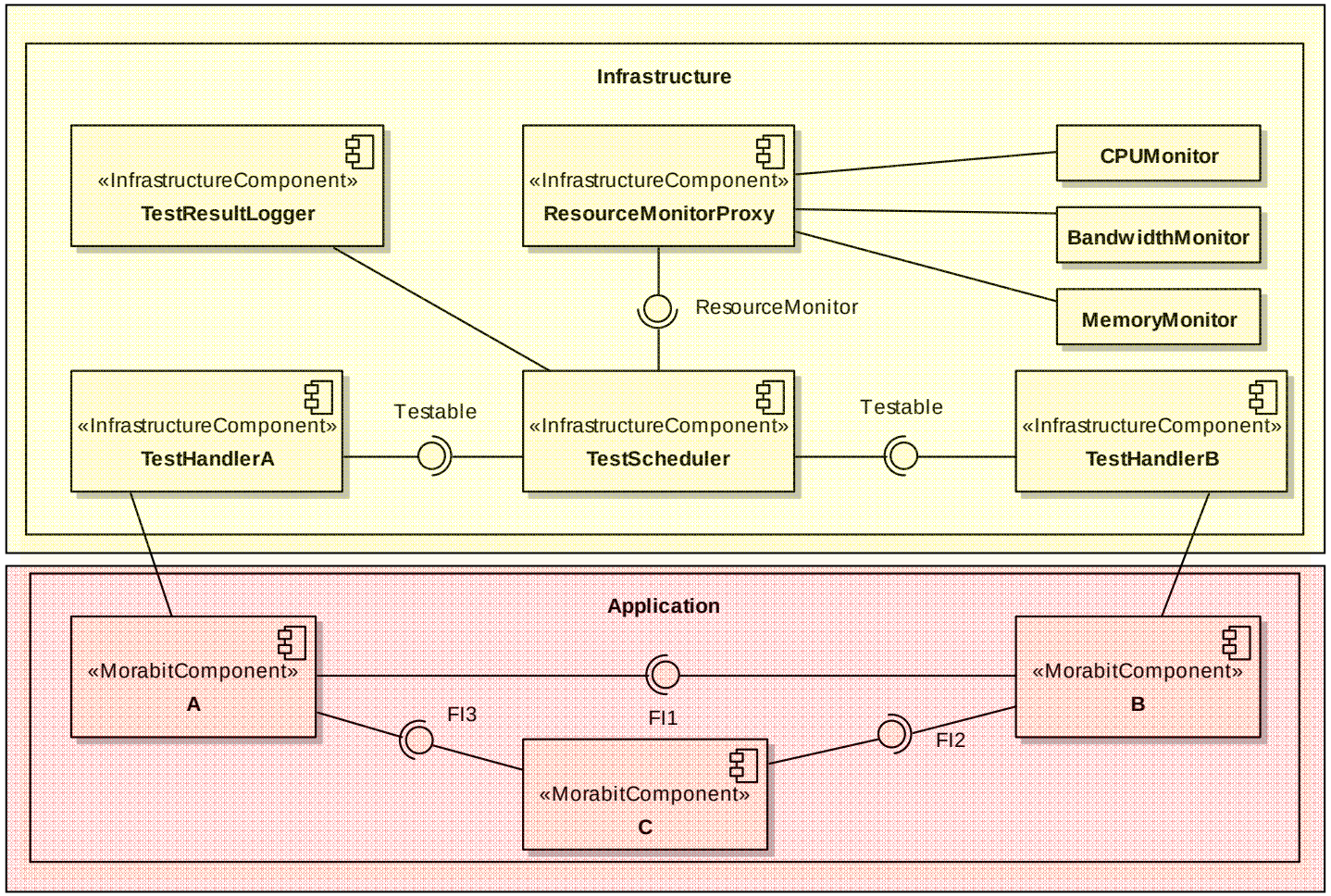
EML Research

MORA BIT

Infrastructure

- Run-time testing with BITs
  - component middleware
  - test execution support
- Resource-awareness
  - resource measurement
  - test execution dependent on resource situation
- “Separation of Concerns“
  - don‘t mix testing and “normal“ logic

Infrastructure: Architecture



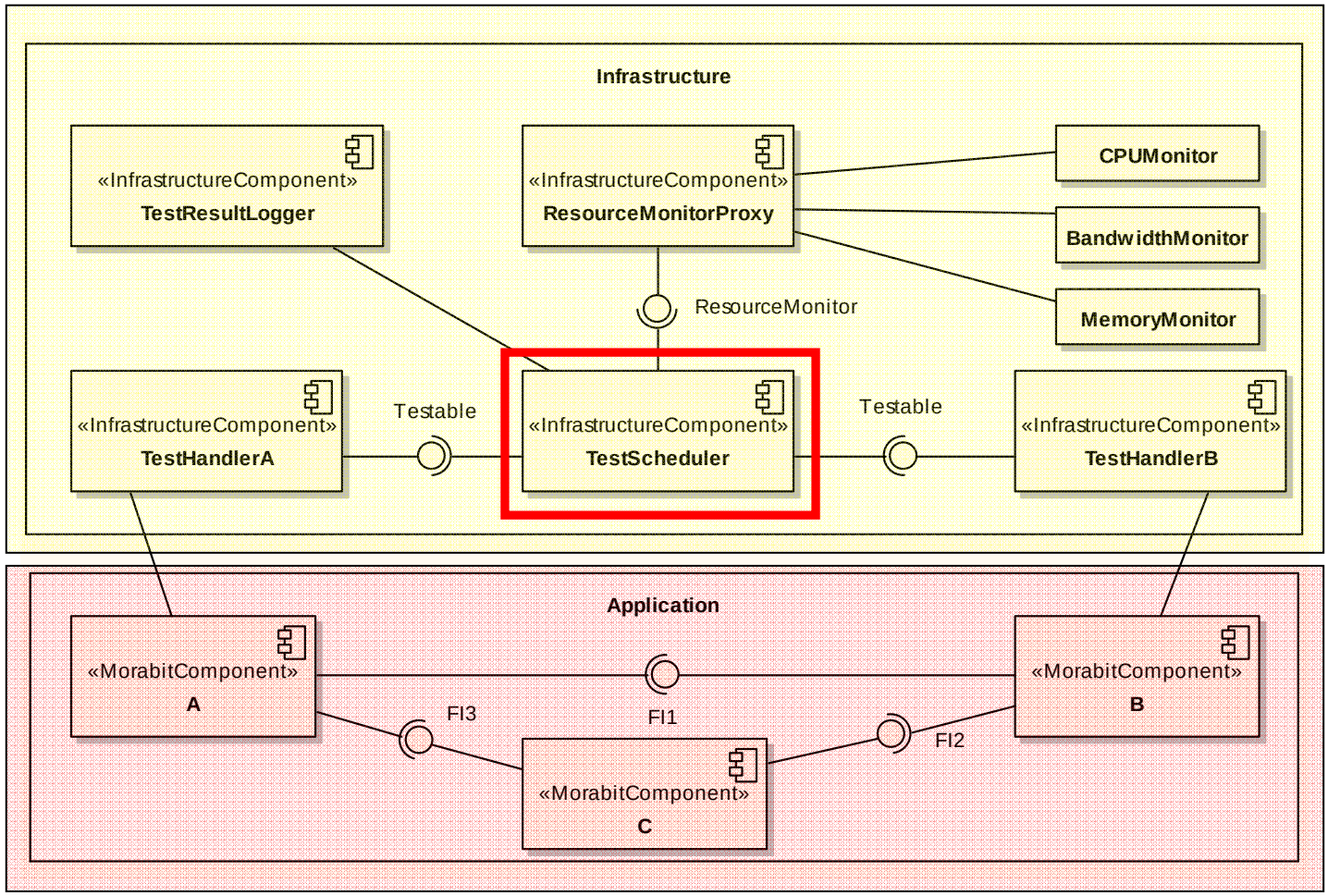
Infrastructure: Middleware for run-time tests (RATs)

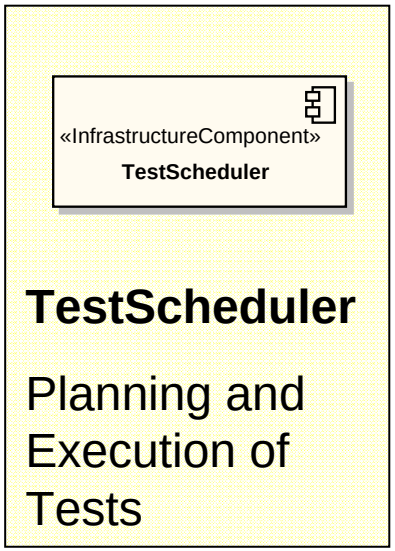
User application: business logic

EML Research

MORA BIT

Infrastructure: Architecture

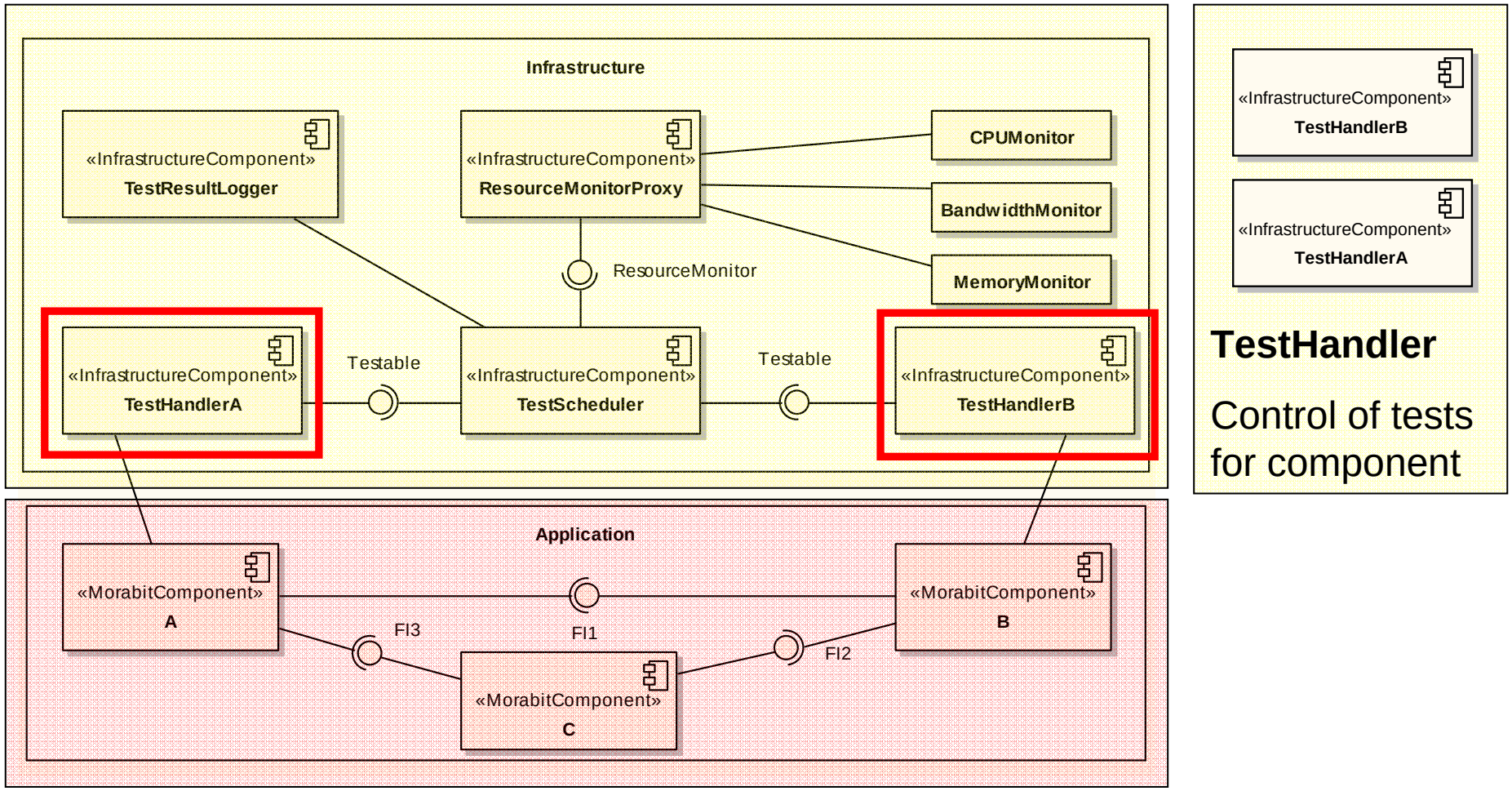




EML Research

MORA BIT

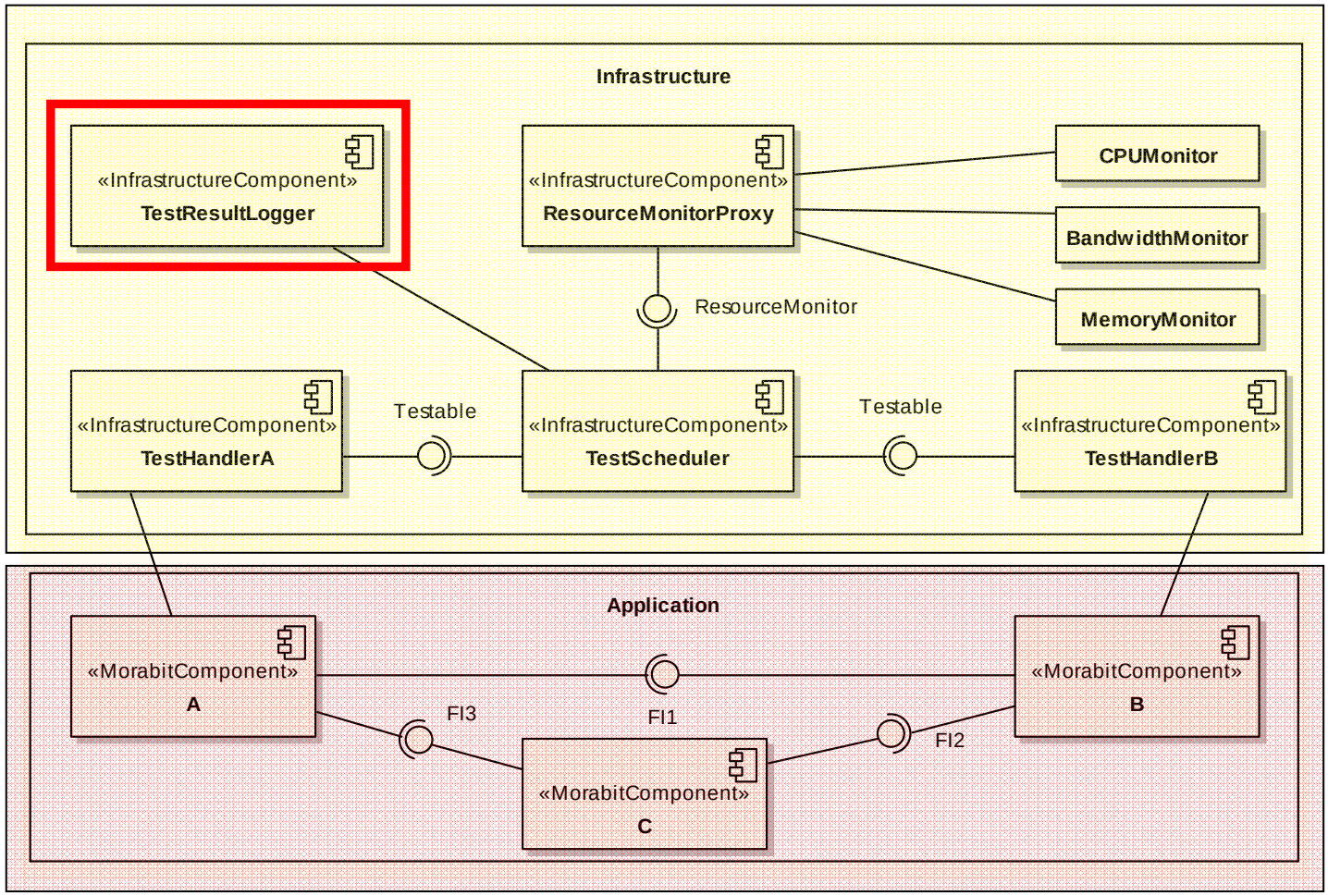
Infrastructure: Architecture



EML Research

MORA BIT

Infrastructure: Architecture



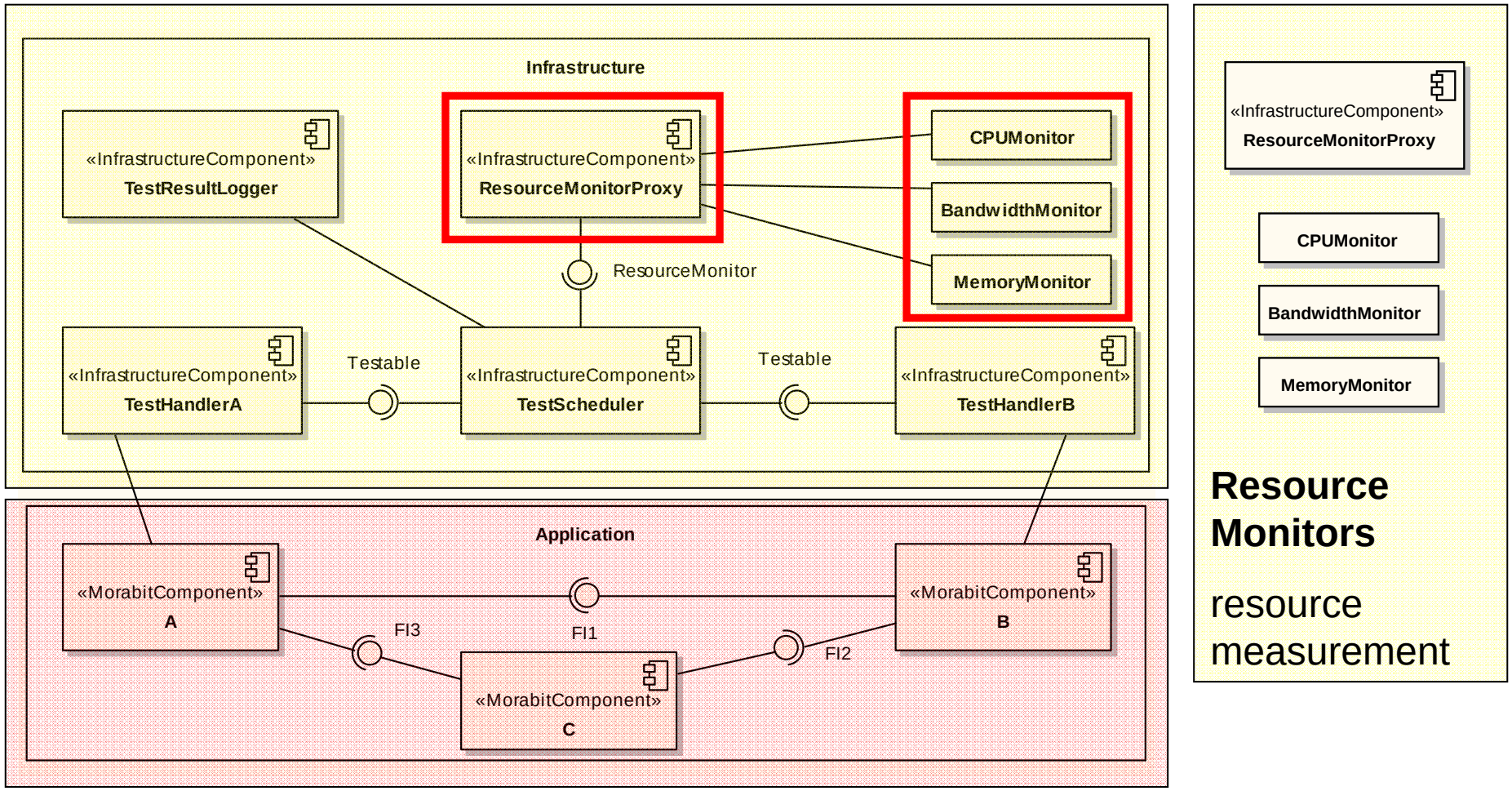
«InfrastructureComponent»
TestResultLogger

TestResult Logger

Visualization of test results

Test history

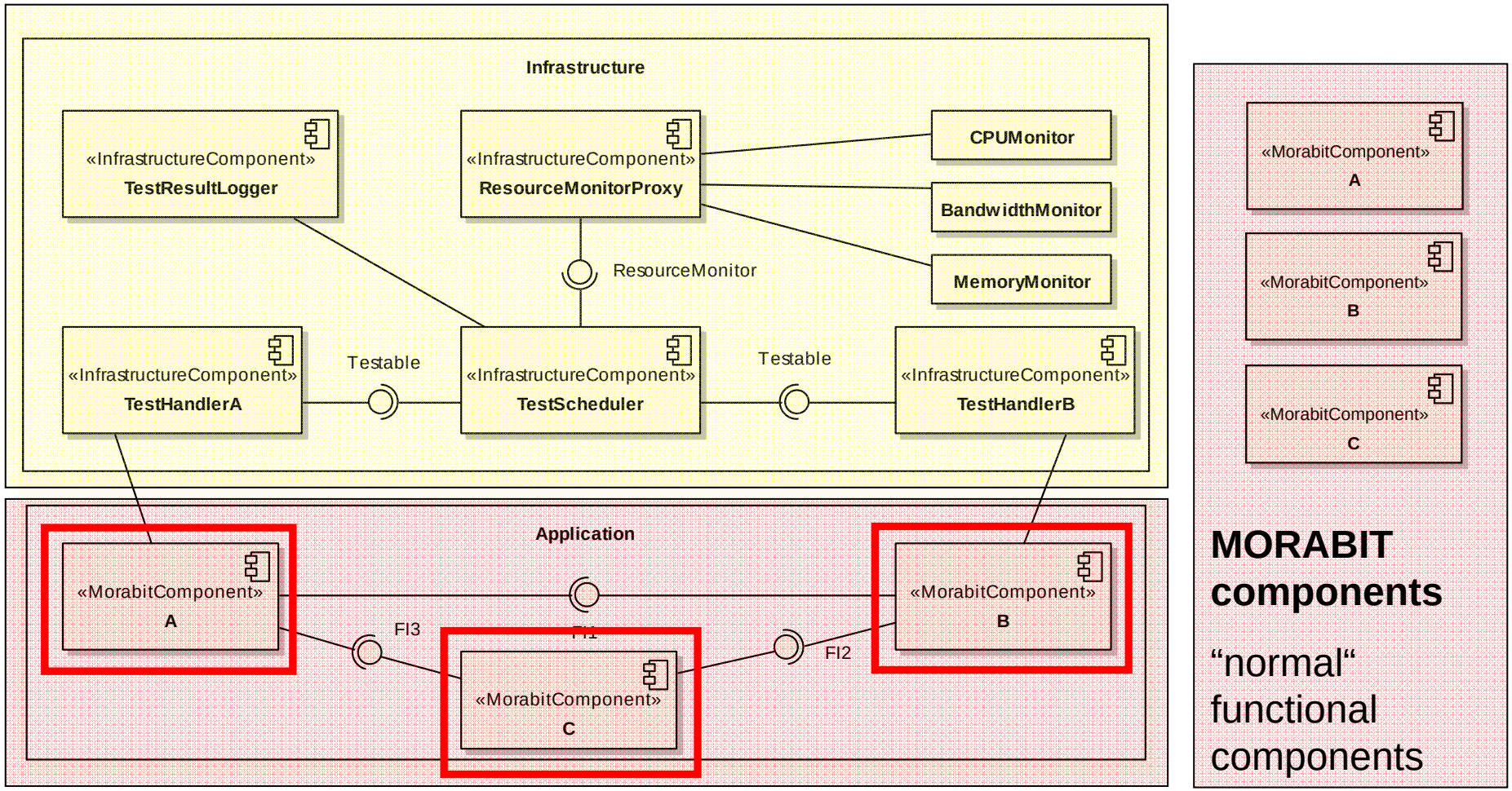
Infrastructure: Architecture



EML Research

MORA BIT

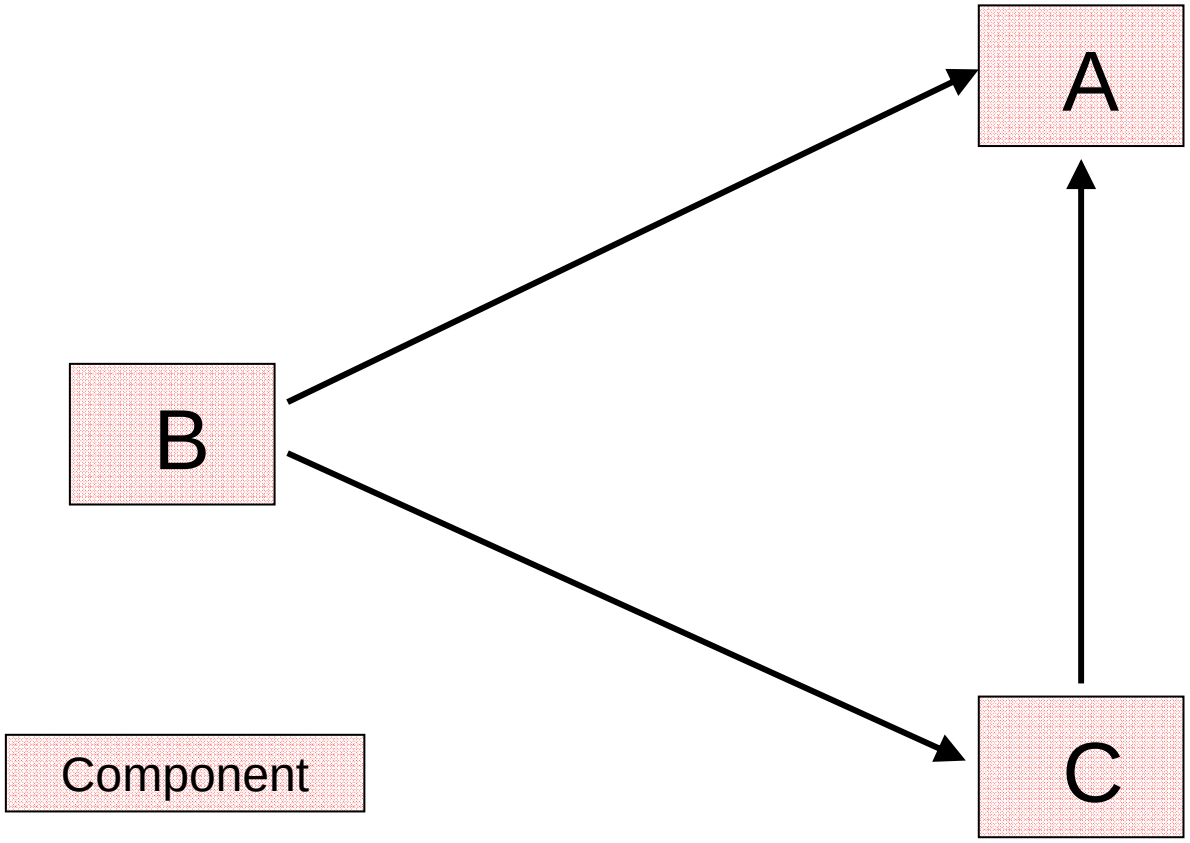
Infrastructure: Architecture



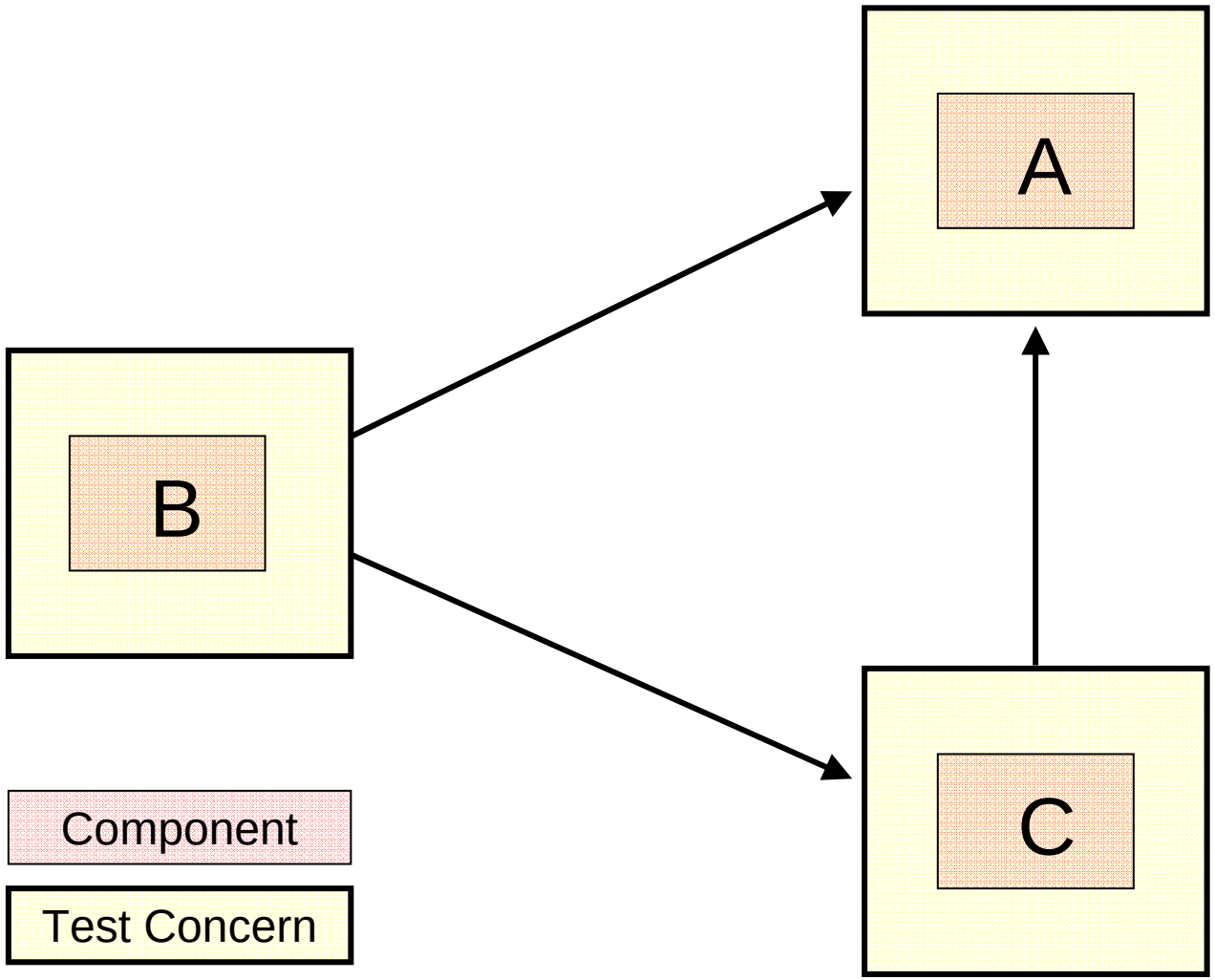
Infrastructure: Dynamic View

- Run-time view of components & tests
- Testing details:
  - Test start times
  - Test execution strategies
  - Test reactions

Infrastructure: Runtime View



Infrastructure: Runtime View



Test Start Times

- Component life cycle
  - lookup-time
  - call-time
- System state
  - idle-time
  - topology-change-time
- Chronological time
  - periodic-time
  - random-time

EML Research

MORA BIT

Test Start Time: lookup-time

BankComponent.java

```
package org.morabit;

public class BankComponent
{
    //...
    private CurrencyConverter converter = null;

    public void init()
    {
        this.converter = (CurrencyConverter)
            Infrastructure.lookup(CurrencyConverter.class);
    }

    public Money convert(Money amount)
    {
        return this.converter.convert(amount, "USD", "EUR");
    }
}
```

Test Start Time: call-time

BankComponent.java

```
package org.morabit;

public class BankComponent
{
    //...
    private CurrencyConverter converter = null;

    public void init()
    {
        this.converter = (CurrencyConverter)
            Infrastructure.lookup(CurrencyConverter.class);
    }

    public Money convert(Money amount)
    {
        return this.converter.convert(amount, "USD", "EUR");
    }
}
```

Mobile & Ubiquitous Software

- Limited resources
- Resources:
  - Memory
  - CPU
  - Bandwidth
  - Battery Charge

MObile Resource-Aware Built-In Tests

EML Research

MORA BIT

Mobile & Ubiquitous Software

- Limited resources
- Resources:
  - Memory
  - CPU
  - Bandwidth
  - Battery Charge

MORABIT

EML Research

MORA BIT

Resource-Aware Test Execution

- Tests use resources
- Trade-off between
  - tests and
  - business logic
- Options:
  - cancel or postpone tests
  - limit resources
  - partial test suites

Resource-Aware Test Execution

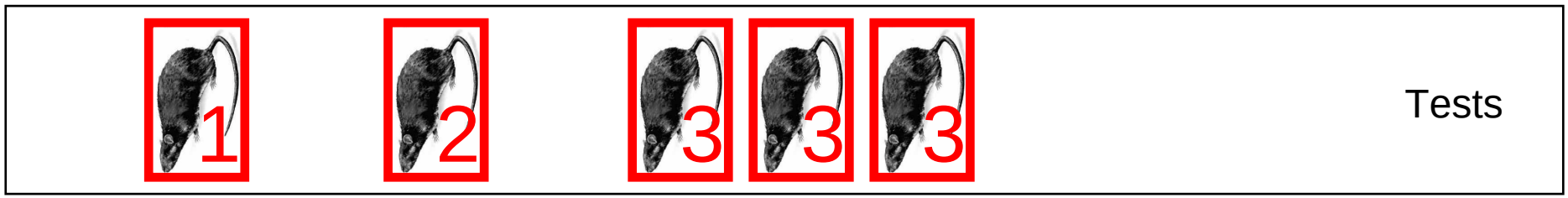
- Many test execution strategies
- Example: “Delay“ strategy

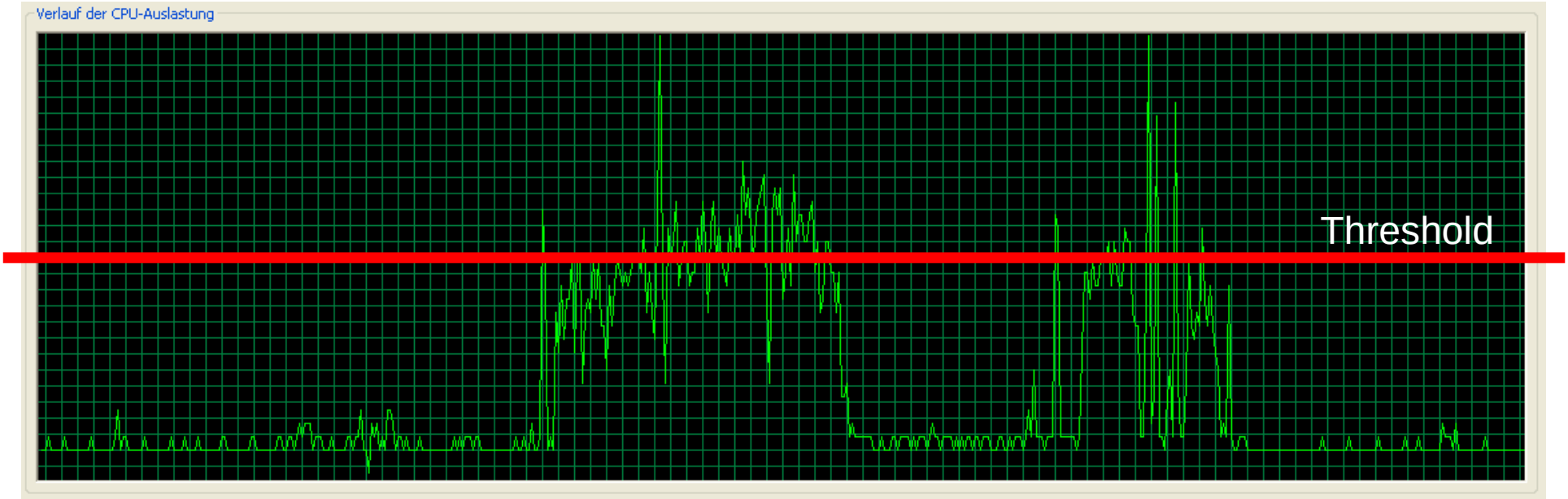
Details:
Merdes M, Malaka R, Suliman D, Paech B, Brenner D, Atkinson C (2006): *Ubiquitous RATs: How Resource-Aware Run-Time Tests can improve Ubiquitous Software Systems*. Proceedings of the Sixth International Workshop on Software Engineering and Middleware (SEM 2006). ACM Press, New York, NY

Example: “Delay“ Strategy

- Controlled postponement of tests
- Avoid load maxima
- Cheap! no expensive planning
- “Poor man‘s“-optimization

Example: “Delay“ Strategy



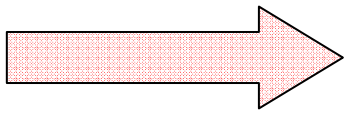


EML Research

MORA BIT

Test Reactions

- Development-time:
  - Go and fix that bug!
- Run-time:
  - Cannot fix bug...



„Automatic“ Reactions?

EML Research

MORA BIT

Test Reactions

- Generic reactions
- Declarative specification
- Some examples

Test Reactions

- Generic reactions
- Declarative specification

Use-anyway Reaction:

- non-critical situations
- warning to user
- logging, debugging

EML Research

MORA BIT

Test Reactions

- Generic reactions
- Declarative specification

Try-next-component Reaction:

- find "better" component with same service
- "transparent" improvement

EML Research

MORABIT

Test Reactions

- Generic reactions
- Declarative specification

Shut-down Reaction:

- for critical tests
- thread or system shut-down

Conclusion

- Run-time tests improve software systems
- Run-time tests differ from dev.-time tests
  - nature of tests
  - test times
  - test execution
  - test reaction

- RATs consider resource situation
- Supported by run-time infrastructure
  - separation of test and application concerns

Future Work

- Implementation
  - more test execution strategies
- Empirical comparison
  - no tests vs. run-time tests
  - “normal“ run-time tests vs. RATs
- Distribution
  - more realistic
  - test execution as load balancing?

Acknowledgements

- Rainer Malaka, Dirk Dorsch
  Barbara Paech, Dima Suliman
  Daniel Brenner, Colin Atkinson

RUPRECHT-KARLS-UNIVERSITÄT HEIDELBERG

UNIVERSITÄT MANNHEIM

- Klaus Tschira Foundation

KTS

- Landesstiftung
  Baden-Württemberg:
  MORABIT project

LANDESSTIFTUNG Baden-Württemberg

EML Research

MORA BIT

Thank you for your attention!

Questions?

EML Research

MORA BIT