Enabling Run-Time System Verification through Built-in Testing

Daniel Brenner
dbrenner@uni-mannheim.de

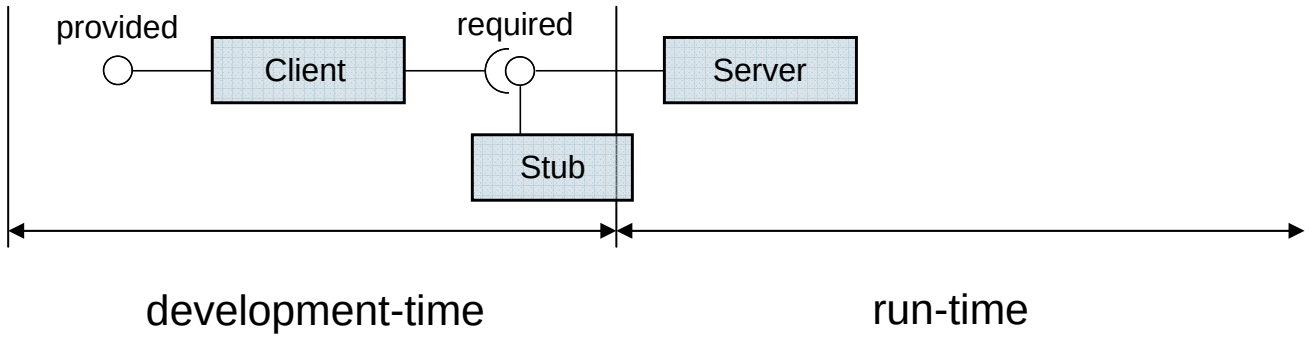
University of Mannheim
Prof. Colin Atkinson, Ph.D.

Overview

- Motivation
- Run-Time Testing
- Method Sequence Testing Language

Motivation

- Components in an ad-hoc system do not know their communication partners at development-time
  - Dynamic structure
- Dependencies on other (required) components



Run-Time Testing

- Testing system in actual run-time environment
- Reaction to test result drives run-time testing
  - Not possible to stop system
- Testing for same understanding of functionality
  - Contract

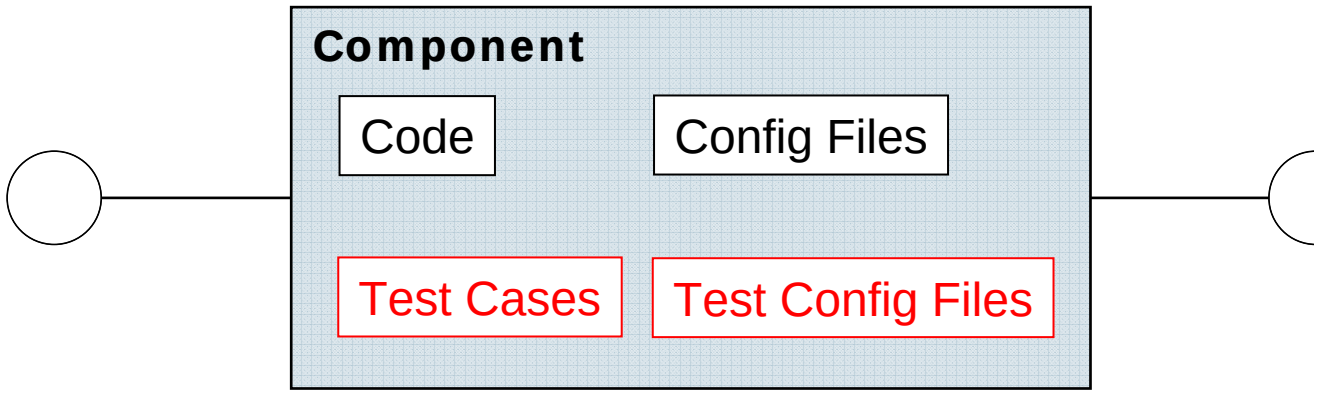
Bank

boolean transferMoney(String, String, double);

- Component's service delivery should not be affected

Built-In Testing

- Test cases are „built-into“ component
- Black-box testing
  - Test cases are written against interfaces



Quantitative Testing

- Qualitative Testing
  - Binary ‘passed’ vs. ‘failed’

Bank

boolean transferMoney(String from, String to, double amount);

- Quantitative Testing
  - Reliability assessment
  - Accuracy depends on sample size

Mail

void send(Message msg);

Method Sequence Testing Language

- Because of information hiding state not visible
  - Method sequence necessary
- Methods on their own are not testable
- Enhanceable with regular expressions

```
send(input);
List<Message> result = receive();
```

```
expects result.contains(input);
with input = new Message(...);
```

Mail

```
void send(Message msg);
List<Message> receive();
```

Future Work

- Implement MSTL
- Find good/appropriate reliability model
- Determine confidence measure
- ...

Thank you for your attention!