Software Testing
Introduction

Prof. Dr. Marie-Christine Jakobs
jakobs@sosy.ifi.lmu.de

Your Test Experience

- I executed (my) programs with example inputs.
- I wrote Unit tests.
- I used automatic testing tools.

Why Do We Test?

Writing correct code is hard

- Software systems are complex
- Intended functionality ambiguous, not precisely documented

$\Longrightarrow$ Most software will contain bugs (faults)

Software failures can have severe consequences

Threats of Software Failures 

- Reputation
  - Google's search engine classified all web sites as malicious in 2009
  - ...
- Costs
  - Ariane V rocket (destroyed due to an overflow error)
  - ...
- Harmful, possibly life-threatening
  - Northeast blackout of 2003 was caused by a data race
  - Therac-25 radiation therapy machine caused deaths
  - ...

source: `https://en.wikipedia.org/wiki/List_of_software_bugs`

Testing in Context of Software Quality Assurance

Testing is one means in software quality assurance

Testing Detection of failures
(deviation of expected behavior)

Debugging Detection of bug/fault (reason(s) for failure)

Fix Aims to remove the bug/fault from the coe

Software quality improves after the fix

Does This Program Work as Intended?

Method `avg` computes the average of all numbers in `array`

```
public static float avg(int[] array) {
  int sum = 0;
    for(int i=0;i<array.length;i++) {
      sum += array[i];
    }
    return sum/array.length;
}
```

Testing the Code with Empty Arrays

```
@Test
void testAverageEmpty() {
  int[] emptyArray = {};
  assertEquals(0, Average.avg(emptyArray));
}
```

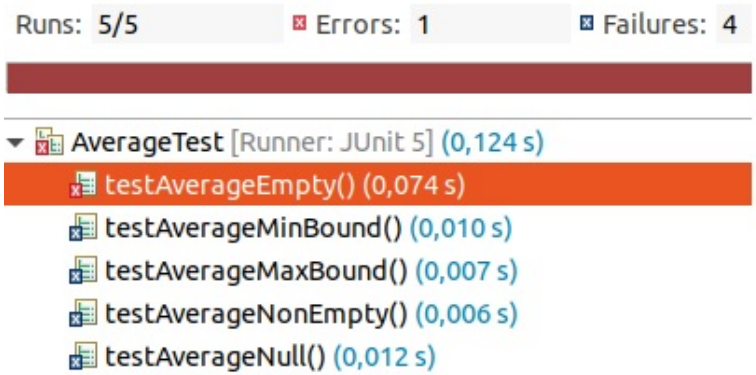
Test

- without precondition (true)
- input `emptyArray`
- expected result 0
- no postcondition (true)
- test instructions
  i.e., call of method avg with `emptyArray`
- comparison of observed and expected result
  automatic evaluation of test outcome

Further Tests for the Example Code

```
@Test
void testAverageMaxBound() {
  int[] maxArray = {Integer.MAX_VALUE, Integer.MAX_VALUE};
  assertEquals(Integer.MAX_VALUE, Average.avg(maxArray));
}

@Test
void testAverageMinBound() {
  int[] minArray = {Integer.MIN_VALUE, Integer.MIN_VALUE};
  assertEquals(Integer.MIN_VALUE, Average.avg(minArray));
}

@Test
void testAverageNonEmpty() {
  int[] array = {0,1};
  assertEquals(0.5, Average.avg(array));
}
```

Result of Test Execution



- Error: division by zero
- Failures: violation of assertions

Distinguishing between Fault, Failure, and Mistake

Failure
Deviation of a system, component, function, etc. from the expected behavior/result

Fault (also called defect, bug)
Incorrect instruction, step, process, etc.
May cause a failure

Mistake (also called error)
A human action that caused the fault

Demonstrating Terms Fault, Failure, and Mistake

```
public static float avg(int[] array) {
  int sum = 0;
    for(int i=0;i<array.length;i++) {
      sum += array[i];
    }
    return sum/array.length;
}
```

Failure calling avg on [0,1] returns 0 instead of 0.5

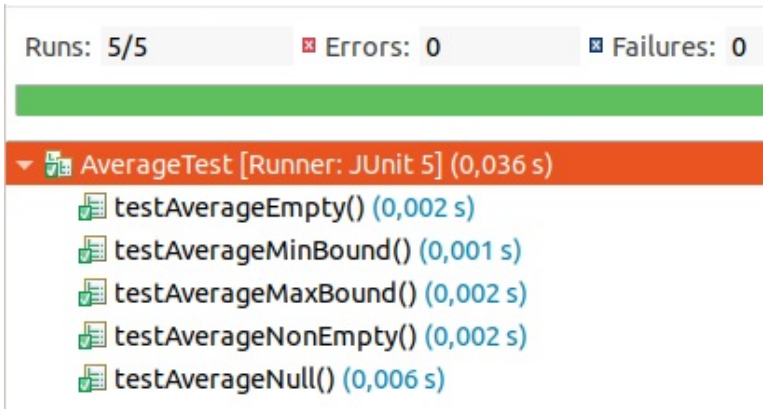
Fault Integer division

Mistake Neither dividend nor divisor casted to floating point number

Example Code with Patches

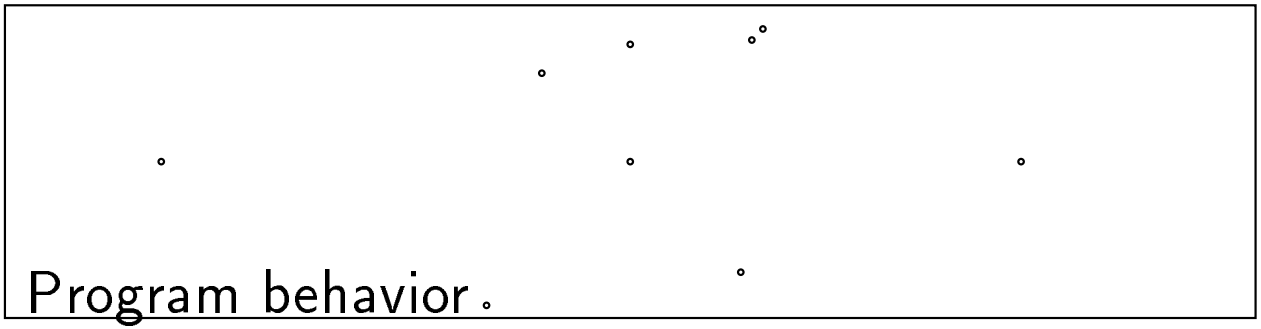
```
public static float avg(int[] array) {
  if(array == null || array.length == 0) {
    return 0;
  }

  long sum = 0;
  for(int i=0;i<array.length;i++) {
    sum+=array[i];
  }
  return (float) sum/array.length;
}
```

All Tests Pass – Is the Code correct?



- Does the method return the intended average for other arrays?
- Is 0 the intended value for empty or null arrays?

Incompleteness of Testing and its Consequences



Only test a subset of all possible behaviors

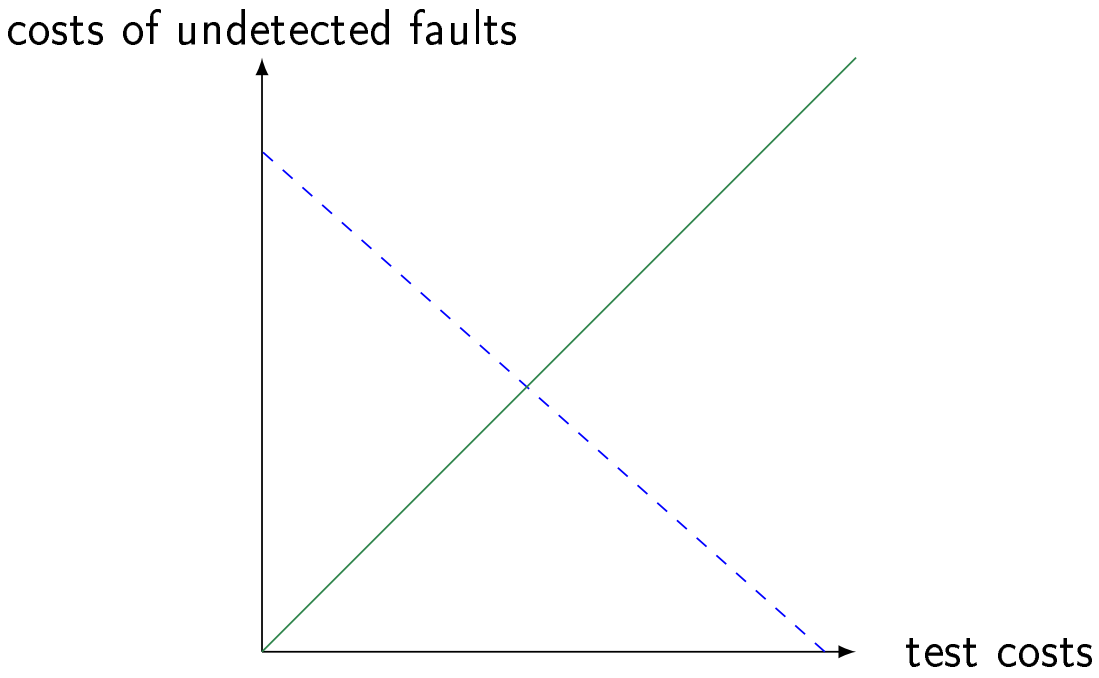
$\Longrightarrow$ underapproximate behavior

$\Longrightarrow$ can detect failures **but not** prove their absence

What to Test and How Much to Test?

(Practically) impossible to test every input

- Too many inputs
- Limited time and budget
- Not economic



Naive Idea

Distribute test effort equally

Ignores that not every software piece

- causes same costs/harms if fails
- has the same likelihood for failures

Risk-oriented Testing

Takes the individual risks of software pieces into account

Risk

- Combines likelihood of a failure with costs the failure causes if it occurs
- Aggregates risk for individual failures

Test riskier software more intensively
Prioritize software with higher risks

Likelihood of a Particular Failure

Need to consider different aspects like e.g.,

- how often will the functionality be used?
- how likely is it that it does not work as intended? (e.g., past experience of quality of component, performance of developer team, complexity of functionality, etc.)
- how likely it is that failure not captured by current tests?
- ...

Likelihood of Failure of Method avg Due to Overflow

- Expect developer to use a data type of int or larger for sum
  (otherwise explicit downcast needed)
- Expect that 90% of users call method with an array of less than 100 elements and elements in the range [0, 1000]

$\Longrightarrow$ Expected likelihood of sum of elements larger than Integer.MAX_VALUE below 10%

Costs of a Failure

Need to consider different aspects like e.g.,

- does one need to pay fines, penalties, etc.
- how costly it is to fix the code after the failure occurred?
- how many customers require the fix and how expensive/difficult it is to distribute the fix?
- what is the impact on reputation, future sells of this or other products?
- what are the legal consequences?
- ...

Costs of Failure of Method avg Due to Overflow

- License excludes liability of authors for software
- Software is offered as open source software without fees

$\Longrightarrow$ Expected costs are costs for debugging and fixing the overflow after it has been reported

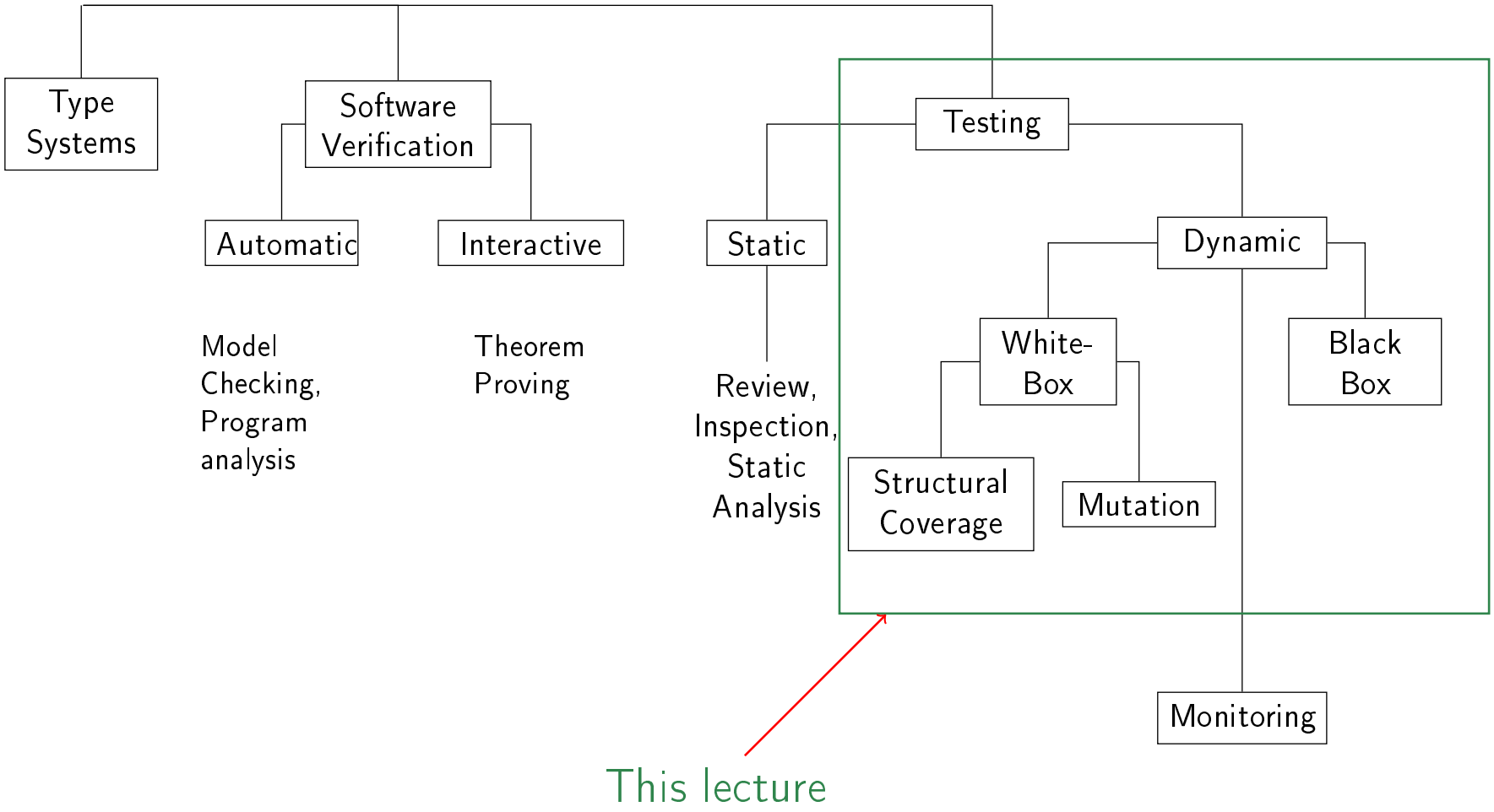
Different Levels of Testing

Component Tests Tests a single software unit (e.g., class)

Integration Tests Tests interfaces between components

System Tests Tests that system meets requirements

Overview over Verification and Testing Techniques



🏁 Learning Outcomes for Introduction

- Explain the terms testing, debugging, failure, fault, mistake, and risk
- Explain why software is tested and what can be achieved
- Name the elements of a test
- Name the different levels of testing
- Explain the idea of risk-oriented testing