Actual Test Coverage for Embedded Systems

Mark Timmer
University of Twente

14th Dutch Testing Day
November 27, 2008
Why coverage?

- Testing is inherently incomplete
- Testing does increase our confidence in the system
- A notion of *quality* of a test suite is necessary
- Coverage: ‘amount’ of specification / implementation examined by a test suite
Why coverage?

- Testing is inherently incomplete
- Testing does increase our confidence in the system
- A notion of *quality* of a test suite is necessary
- Coverage: ‘amount’ of specification / implementation examined by a test suite
Introduction – coverage

Why coverage?

- Testing is inherently incomplete
- Testing does increase our confidence in the system
- A notion of *quality* of a test suite is necessary
- Coverage: ‘amount’ of specification / implementation examined by a test suite
Why coverage?

- Testing is inherently incomplete
- Testing does increase our confidence in the system
- A notion of *quality* of a test suite is necessary
- Coverage: ‘amount’ of specification / implementation examined by a test suite

Coverage: \( \frac{6}{13} = 46\% \).
Early work on coverage

- Statement coverage
- Path coverage

Limitations:
- All faults are considered of equal severity
- No probabilities
- Syntactic point of view
# Introduction – Existing approaches to coverage

## Early work on coverage

- Statement coverage
- Path coverage

**Limitations:**
- All faults are considered of equal severity
- No probabilities
- Syntactic point of view

---

### Recipe 1: Vegetable Soup
- Chop an onion
- Slice carrots and mushroom
- Boil one liter of water
- Put everything in the water
- Wait a while

**Quality:** 4 · 5 = 20

### Recipe 2: Vegetable Soup
- Chop an onion
- Slice a few carrots
- Slice a mushroom
- Boil one liter of water
- Put everything in the water
- Wait a while

**Quality:** 4 · 6 = 24
## Early work on coverage

- **Statement coverage**
- **Path coverage**

Limitations:
- all faults are considered of equal severity
- no probabilities
- syntactic point of view

### Recipe 1: vegetable soup
- Chop an union
- Slice carrots and mushroom
- Boil one liter of water
- Put everything in the water
- Wait a while

### Quality:

$$4 \cdot 5 = 8$$

### Recipe 2: vegetable soup
- Chop an union
- Slice a few carrots
- Slice a mushroom
- Boil one liter of water
- Put everything in the water

### Quality:

$$4 \cdot 6 = 6$$
Early work on coverage

- Statement coverage
- Path coverage

Limitations:
- all faults are considered of equal severity
- no probabilities
- syntactic point of view

Recipe 1: vegetable soup

- Chop an union
- Slice carrots and mushroom
- Boil one liter of water
- Put everything in the water
- Wait a while

Recipe 2: vegetable soup

- Chop an union
- Slice a few carrots
- Slice a mushroom
- Boil one liter of water
- Put everything in the water

Quality: $4 \times 5 = 20$
Introduction – Existing approaches to coverage

Early work on coverage

- Statement coverage
- Path coverage

Limitations:
- All faults are considered of equal severity
- No probabilities
- Syntactic point of view

Recipe 1: vegetable soup

- Chop an union
- Slice carrots and mushroom
- Boil one liter of water
- Put everything in the water
- Wait a while

Quality: \( \frac{4}{5} \cdot 10 = 8 \)
Early work on coverage

- Statement coverage
- Path coverage

Limitations:
- all faults are considered of equal severity
- no probabilities
- syntactic point of view

Recipe 1: vegetable soup
- Chop an union
- Slice carrots and mushroom
- Boil one liter of water
- Put everything in the water
- Wait a while

Quality: \[ \frac{4}{5} \cdot 10 = 8 \]

Recipe 2: vegetable soup
- Chop an union
- Slice a few carrots
- Slice a mushroom
- Boil one liter of water
- Put everything in the water
- Wait a while
Early work on coverage

- Statement coverage
- Path coverage

Limitations:
- all faults are considered of equal severity
- no probabilities
- syntactic point of view

Recipe 1: vegetable soup
- Chop an union
- Slice carrots and mushroom
- Boil one liter of water
- Put everything in the water
- Wait a while

Quality: \( \frac{4}{5} \cdot 10 = 8 \)

Recipe 2: vegetable soup
- Chop an union
- Slice a few carrots
- Slice a mushroom
- Boil one liter of water
- Put everything in the water
- Wait a while

Quality: \( \frac{4}{6} \cdot 10 = 6 \)
**Early work on coverage**

- Statement coverage
- Path coverage

Limitations:
- All faults are considered of equal severity
- No probabilities
- Syntactic point of view

**Recipe 1: vegetable soup**
- Chop an union
- Slice carrots and mushroom
- Boil one liter of water
- Put everything in the water
- Wait a while

Quality: \( \frac{4}{5} \cdot 10 = 8 \)

**Recipe 2: vegetable soup**
- Chop an union
- Slice a few carrots
- Slice a mushroom
- Boil one liter of water
- Put everything in the water
- Wait a while

Quality: \( \frac{4}{6} \cdot 10 = 6.7 \)
Starting point for my work: semantic coverage

Previous work by Brandán Briones, Brinksma and Stoelinga

- System considered as black box
- Semantic point of view
- Fault weights
Starting point for my work: semantic coverage

Previous work by Brandán Briones, Brinksma and Stoelinga
- System considered as black box
- Semantic point of view
- Fault weights

Labelled transition systems

\[
\begin{array}{c}
{s_1} & \xrightarrow{20\text{ct}^?} & s_0 & \xrightarrow{10\text{ct}^?} & s_2 \\
\text{coffee!} & \quad & \quad & \quad & \text{tea!}
\end{array}
\]
Starting point for my work: semantic coverage

Previous work by Brandán Briones, Brinksma and Stoelinga

- System considered as black box
- Semantic point of view
- Fault weights

Labelled transition systems

\[
\begin{array}{c}
{s_1} \quad 20\text{ct?} \quad 10\text{ct?} \\
{s_0} \quad \delta \\
{s_2} \quad \text{coffee!} \quad \text{tea!}
\end{array}
\]
Starting point for my work: semantic coverage

Previous work by Brandán Briones, Brinksma and Stoelinga

- System considered as black box
- Semantic point of view
- Fault weights

Labelled transition systems

\[ s_0 \rightarrow \delta \rightarrow s_1, s_2 \]

- 20ct?
- 10ct?
- Coffee!
- Tea!

20ct?
Starting point for my work: semantic coverage

Previous work by Brandán Briones, Brinksma and Stoelinga

- System considered as black box
- Semantic point of view
- Fault weights

Labelled transition systems
Starting point for my work: semantic coverage

Previous work by Brandán Briones, Brinksma and Stoelinga

- System considered as black box
- Semantic point of view
- Fault weights

Labelled transition systems

Actual Test Coverage for Embedded Systems
Starting point for my work: semantic coverage

Previous work by Brandán Briones, Brinksma and Stoelinga

- System considered as black box
- Semantic point of view
- Fault weights

Labelled transition systems

\[
\begin{align*}
\delta & \quad 20\text{ct}? & 10\text{ct}?
\end{align*}
\]

\[
\begin{array}{c}
s_0 \quad \text{coffee!} \quad \text{tea!} \\
\quad s_1 \quad \quad \quad \quad \quad \quad s_2
\end{array}
\]

20ct? coffee! 10ct? tea!
Starting point for my work: semantic coverage

Previous work by Brandán Briones, Brinksma and Stoelinga

- System considered as black box
- Semantic point of view
- Fault weights

Labelled transition systems

20ct? coffee! 10ct? tea!

Test cases

10ct? tea! coffee!

fail
Starting point for my work: semantic coverage

Previous work by Brandán Briones, Brinksma and Stoelinga

- System considered as black box
- Semantic point of view
- Fault weights

Labelled transition systems

\[
\begin{align*}
S_1 & \xrightarrow{\delta} S_0 \\
S_0 & \xrightarrow{20\text{ct}?} S_1 \\
S_0 & \xrightarrow{10\text{ct}?} S_2 \\
S_2 & \xrightarrow{\text{coffee!}} S_0 \\
S_0 & \xrightarrow{\text{tea!}} S_2
\end{align*}
\]

Test cases

\[
\begin{align*}
\text{10ct?} & \xrightarrow{\delta} \text{tea!} \\
\text{tea!} & \xrightarrow{\text{coffee!}} \text{fail} \\
\text{coffee!} & \xrightarrow{\text{fail}} \text{fail}
\end{align*}
\]
Starting point for my work: semantic coverage

Previous work by Brandán Briones, Brinksma and Stoelinga
- System considered as black box
- Semantic point of view
- Fault weights

Labelled transition systems

Test cases
Starting point for my work: semantic coverage

Previous work by Brandán Briones, Brinksma and Stoelinga
- System considered as black box
- Semantic point of view
- Fault weights

Labelled transition systems

Test cases
Absolute potential coverage: 4 + 5 + 3 + 2 = 14
Absolute potential coverage: 10 + 20 + 4 + 5 = 39
Actual test coverage: 2 + 3 + 4 + 5 = 14
Absolute potential coverage

4 + 5 + 3 + 2 = 14
Introduction - Limitations of potential coverage

Some remarks:
- Not all faults can be detected at once.
- Single executions cover only some faults.
- Executing more often could increase coverage.
- Necessary to include probabilities.

Absolute potential coverage:
$$10 + 15 = 25$$
Introduction - Limitations of potential coverage

Absolute potential coverage

$$10 + 15 = 25$$
Introduction - Limitations of potential coverage

Some remarks

- Not all faults can be detected at once
- Single executions cover only some faults
- Executing more often could increase coverage
- How many executions are needed?
- Necessary to include probabilities!

Absolute potential coverage

\[10 + 15 = 25\]
Introduction - Limitations of potential coverage

Absolute potential coverage

\[ 10 + 15 = 25 \]

Some remarks

- Not all faults can be detected at once
- Single executions cover only some faults
- Executing more often could increase coverage
- How many executions are needed?
- Necessary to include probabilities!
Some remarks

- Not all faults can be detected at once
- Single executions cover only some faults
- Executing more often could increase coverage
- How many executions are needed?
- Necessary to include probabilities!

Absolute potential coverage

$$10 + 15 = 25$$
Introduction - Limitations of potential coverage

Some remarks
- Not all faults can be detected at once
- Single executions cover only some faults
- Executing more often could increase coverage
- How many executions are needed?
- Necessary to include probabilities!

Absolute potential coverage

\[ 10 + 15 = 25 \]
Overview of actual coverage

Actual coverage

1. Probabilistic execution model:
   - Branching probabilities \( p^{br} \)
   - Conditional branching probabilities \( p^{cbr} \)
Overview of actual coverage

Actual coverage

1. Probabilistic execution model:
   - Branching probabilities ($p^{br}$)
   - Conditional branching probabilities ($p^{cbr}$)

2. Evaluating actual coverage:
   Calculating the actual coverage of a given execution or sequence of executions
Overview of actual coverage

Actual coverage

1. Probabilistic execution model:
   - Branching probabilities ($p^{br}$)
   - Conditional branching probabilities ($p^{cbr}$)

2. Evaluating actual coverage:
   Calculating the actual coverage of a given execution or sequence of executions

3. Predicting actual coverage:
   Predicting the actual coverage a test case or test suite yields.
Fault coverage

A fault is *covered* by an execution if the execution gives us information about whether the fault is present or absent.
A fault is *covered* by an execution if the execution gives us information about whether the fault is present or absent.
Fault coverage

A fault is *covered* by an execution if the execution gives us information about whether the fault is present or absent.
A fault is *covered* by an execution if the execution gives us information about whether the fault is present or absent.
Fault coverage

Actual Test Coverage for Embedded Systems  Evaluating actual coverage  Nov. 27, 2008  10 / 24
Fault coverage

Actual Test Coverage for Embedded Systems  Evaluating actual coverage  Nov. 27, 2008  10 / 24
Fault coverage

- Coffee! (1.0)
- Tea! (0.0)
- Pass
- Fail

- Coffee! (1.0)
- Tea! (0.0)
- Pass
- Fail

- Coffee! (0.6)
- Tea! (0.4)
- Pass
- Fail

Conditional branching probabilities

Actual Test Coverage for Embedded Systems
Fault coverage

- (1.0) tea!  coffee! (0.0)
  - pass  fail

- (1.0) tea!  coffee! (0.0)
  - pass  fail

- (0.6) tea!  coffee! (0.4)
  - pass  fail

- (0.99) tea!  coffee! (0.01)
  - pass  fail
Fault coverage

Conditional branching probabilities $p_{cbr}$
Fault coverage

1. If a fault is shown present, it is *completely covered*.
2. If a fault is shown absent, it is *partially covered*. 
If a fault is shown present, it is *completely covered*.

If a fault is shown absent, it is *partially covered*.

---

**Fault coverage**

1. If a fault is shown present, it is *completely covered*.
2. If a fault is shown absent, it is *partially covered*. 
Fault coverage

1. If a fault is shown present, it is *completely covered*.
2. If a fault is shown absent, it is *partially covered*.

Fault coverage *coffee!* *coffee!*

10

Fault coverage *tea!* *tea!*

0
Fault coverage

1. If a fault is shown present, it is *completely covered*
2. If a fault is shown absent, it is *partially covered*.

Fault coverage *coffee! coffee!*

Fault coverage *tea! tea!*

10

(0.4)

15

(0.4)
Fault coverage

1. If a fault is shown present, it is *completely covered*.
2. If a fault is shown absent, it is *partially covered*.

Fault coverage *coffee! coffee!*

4 − 6.4 − 7.8 − 8.7

Fault coverage *tea! tea!*

0
Actual coverage

*Actual coverage* of an execution or sequence of executions:
The sum of all fault coverages
Actual coverage

*Actual coverage* of an execution or sequence of executions:
The sum of all fault coverages

![Actual Test Coverage for Embedded Systems](image)
Evaluating actual coverage

**Actual coverage** of an execution or sequence of executions:

The sum of all fault coverages

---

**Actual Test Coverage for Embedded Systems**

---

Nov. 27, 2008 12 / 24
Actual coverage

*Actual coverage* of an execution or sequence of executions:
The sum of all fault coverages

\[
faultCov(b! a? d!) =
\]

---

Evaluating actual coverage
Actual coverage

Actual coverage of an execution or sequence of executions:
The sum of all fault coverages

\[
\text{faultCov}(b! a? d!) = 4
\]
Evaluating actual coverage

Actual coverage of an execution or sequence of executions:
The sum of all fault coverages

\[
\text{faultCov}(b! a? d!) = 4 \\
\text{faultCov}(b! a? c!) =
\]
Actual coverage of an execution or sequence of executions:
The sum of all fault coverages

\[
\text{faultCov}(b! a? d!) = 4 \\
\text{faultCov}(b! a? c!) = 4.8
\]
Evaluating actual coverage

Actual coverage of an execution or sequence of executions:
The sum of all fault coverages

\[
\begin{align*}
\text{faultCov}(b! \ a? \ d!) &= 4 \\
\text{faultCov}(b! \ a? \ c!) &= 4.8 \\
\text{faultCov}(d! \ a? \ b!) &= 0 \\
\text{faultCov}(d! \ a? \ d!) &= 0
\end{align*}
\]
Evaluating actual coverage

Actual coverage of an execution or sequence of executions:
The sum of all fault coverages

\[
\begin{align*}
\text{faultCov}(b! \ a? \ d!) &= 4 \\
\text{faultCov}(b! \ a? \ c!) &= 4.8 \\
\text{faultCov}(d! \ a? \ b!) &= 0 \\
\text{faultCov}(d! \ a? \ d!) &= 0 \\
\text{faultCov}(c!) &= \\
\end{align*}
\]
Actual coverage of an execution or sequence of executions:
The sum of all fault coverages

\[
\text{faultCov}(b! a? d!) = 4 \\
\text{faultCov}(b! a? c!) = 4.8 \\
\text{faultCov}(d! a? b!) = 0 \\
\text{faultCov}(d! a? d!) = 0 \\
\text{faultCov}(c!) = 1.4
\]
Actual coverage

*Actual coverage* of an execution or sequence of executions:
The sum of all fault coverages

\[
\begin{align*}
faultCov(b! a? d!) &= 4 \\
faultCov(b! a? c!) &= 4.8 \\
faultCov(d! a? b!) &= 0 \\
faultCov(d! a? d!) &= 0 \\
faultCov(c!) &= 1.4 \\
absCov &= 10.2
\end{align*}
\]
Actual coverage distribution of a test case

The *actual coverage distribution* of a test case predicts its actual coverage.
Actual coverage distribution of a test case

The actual coverage distribution of a test case predicts its actual coverage.

- Coffee!
- Tea!
- Tea!
- Coffee!
- Coffee!
- Tea!
- Fail (0.4)
- Fail (0.4)
- Pass
- 10
- 15
The actual coverage distribution of a test case predicts its actual coverage.
The actual coverage distribution of a test case predicts its actual coverage.

\[ \text{absCov} \]

\[
\begin{array}{c}
10 \\
15
\end{array}
\]
The actual coverage distribution of a test case predicts its actual coverage.
The actual coverage distribution of a test case predicts its actual coverage.
Actual coverage distribution of a test case

The *actual coverage distribution* of a test case predicts its actual coverage.

\[
\text{absCov} | P \\
10 | 0.015 \\
15 | 0.005 \\
4  | 0.735 \\
6  | 0.245 \\
\]
The actual coverage distribution of a test case predicts its actual coverage.

\[
\begin{array}{c|c|c}
\text{absCov} & P & \times \\
\hline
10 & 0.015 & 0.150 \\
15 & 0.005 & 0.075 \\
4 & 0.735 & 2.940 \\
6 & 0.245 & 1.470 \\
\end{array}
\]
The actual coverage distribution of a test case predicts its actual coverage.

\[
\begin{array}{ccc}
\text{absCov} & \mathbb{P} & \times \\
10 & 0.015 & 0.150 \\
15 & 0.005 & 0.075 \\
4 & 0.735 & 2.940 \\
6 & 0.245 & 1.470 + \\
\hline
\mathbb{E}(\text{absCov}) = 4.635
\end{array}
\]
The branching probabilities $p^{br}$ describe how the implementation is expected to behave.
The branching probabilities $p^{br}$ describe how the implementation is expected to behave.
The branching probabilities $p^{br}$ describe how the implementation is expected to behave.

![Branching probabilities diagram]

- (0.75) coffee! (0.25) tea!
- (0.98) tea! (0.02) coffee! (0.02) tea!
- (0.98) tea! (0.02) coffee! (0.02) tea!
- (0.4) pass  (0.4) fail
- (10)  (15)
Suppose we perform three executions of

\[
\begin{array}{c}
(0.75) \text{ coffee!} \\
(0.25) \text{ tea!}
\end{array}
\]

Possible observation: [blue, blue, red]

Actual coverage: 15 + 6.4 = 21.4

Many observations possible:
Suppose we perform three executions of

\[ \text{(0.75) coffee!} \quad \text{tea! (0.25)} \]

\[ \text{(0.98) tea!} \quad \text{coffee! (0.02)} \quad \text{tea! (0.02)} \quad \text{coffee! (0.98)} \]

Possible observation: [blue, blue, red]
Suppose we perform three executions of

![Diagram showing sequences of executions with probabilities and outcomes.]

- Possible observation: [blue, blue, red]
- Actual coverage: $15 + 6.4 = 21.4$
Suppose we perform three executions of

![Execution Diagram]

Possible observation: [blue, blue, red]

Actual coverage: 15 + 6.4 = 21.4

Many observations possible: $O(|\text{exec}|^n)$
Expected actual coverage for a sequence of executions

**Theorem**

\[ E(\text{actCov}_n) = \sum_{\sigma a \in \text{err}_t} f(\sigma a) \cdot \left( (1 - (1 - p^{\text{to}}(\sigma a))^n) \cdot 1 + \sum_{k=0}^{n} \binom{n}{k} p^{\text{to}}(\sigma)^k (1 - p^{\text{to}}(\sigma))^{n-k}. \right) \]

\[ \left( (1 - p^{\text{br}}(a | \sigma))^k. \right) \]

\[ (1 - (1 - p^{\text{cbr}}(a | \sigma)^k)) \]
Theorem

\[ \mathbb{E}(\text{actCov}_n) = \sum_{\sigma a \in \text{err}_t} f(\sigma a) \cdot \left( \left(1 - (1 - p^{\text{to}}(\sigma a))^n \right) \cdot 1 + \sum_{k=0}^{n} \binom{n}{k} p^{\text{to}}(\sigma)^k \left(1 - p^{\text{to}}(\sigma)\right)^{n-k} \cdot \left(1 - p^{\text{br}}(a | \sigma))^k \cdot \left(1 - (1 - p^{\text{cbr}}(a | \sigma)^k) \right) \right) \]
Expected actual coverage for a sequence of executions

**Theorem**

\[ E(\text{actCov}_n) = \sum_{\sigma a \in \text{err}_t} f(\sigma a) \cdot \left( 1 - (1 - p^{\text{to}}(\sigma a))^n \right) \cdot 1 + \sum_{k=0}^{n} \binom{n}{k} p^{\text{to}}(\sigma)^k (1 - p^{\text{to}}(\sigma))^{n-k} \cdot (1 - p^{\text{br}}(a | \sigma))^k \cdot (1 - (1 - p^{\text{cbr}}(a | \sigma)^k)) \]
Asymptotic behaviour of actual coverage

<table>
<thead>
<tr>
<th>Pass</th>
<th>Fail</th>
<th>Expected Actual Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0.75) coffee!</td>
<td>tea! (0.25)</td>
<td></td>
</tr>
<tr>
<td>(0.98) tea!</td>
<td>coffee! (0.02) (0.02)</td>
<td></td>
</tr>
<tr>
<td>pass</td>
<td>fail</td>
<td></td>
</tr>
<tr>
<td>(0.4)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

Expected actual coverage vs. Number of executions

Theorem: \( \lim_{n \to \infty} E(\text{actual coverage}_n) = \text{potential coverage} \)
Theorem

$\lim_{n \to \infty} \mathbb{E}(\text{actual coverage}_n) = \text{potential coverage}$
Actual coverage for test suites

- Very similar to actual coverage for test cases: sum all the fault coverages
- Take into account in how many test cases an erroneous trace is contained
- Again, an efficient formula for the expected actual coverage exists

**Theorem**

\[ \lim_{n \to \infty} E(\text{actual coverage}_n) = \text{potential coverage} \]
Example – chemical dispenser

Actual Test Coverage for Embedded Systems

Nov. 27, 2008 19 / 24
Example – chemical dispenser

Actual Test Coverage for Embedded Systems
Example – chemical dispenser

(a) Test case $t_1$ (pot.cov.: 1938)

(b) Test case $t_2$ (pot.cov.: 1938)
<table>
<thead>
<tr>
<th>$\mathbb{E}(A_{t_1}^1)$</th>
<th>197.0</th>
<th>$\mathbb{E}(A_{t_2}^1)$</th>
<th>156.8</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbb{E}(A_{t_1}^5)$</td>
<td>729.1</td>
<td>$\mathbb{E}(A_{t_2}^5)$</td>
<td>639.8</td>
</tr>
<tr>
<td>$\mathbb{E}(A_{t_1}^{10})$</td>
<td>1076.9</td>
<td>$\mathbb{E}(A_{t_2}^{10})$</td>
<td>1032.1</td>
</tr>
<tr>
<td>$\mathbb{E}(A_{t_1}^{50})$</td>
<td>1704.6</td>
<td>$\mathbb{E}(A_{t_2}^{50})$</td>
<td>1848.0</td>
</tr>
<tr>
<td>$\mathbb{E}(A_{t_1}^{250})$</td>
<td>1917.7</td>
<td>$\mathbb{E}(A_{t_2}^{250})$</td>
<td>1938.0</td>
</tr>
</tbody>
</table>
Prediction of actual coverage

\[ \mathbb{E}(A_{t1}^1) = 197.0 > \mathbb{E}(A_{t2}^1) = 156.8 \]
\[ \mathbb{E}(A_{t1}^5) = 729.1 > \mathbb{E}(A_{t2}^5) = 639.8 \]
\[ \mathbb{E}(A_{t1}^{10}) = 1076.9 > \mathbb{E}(A_{t2}^{10}) = 1032.1 \]
\[ \mathbb{E}(A_{t1}^{50}) = 1704.6 < \mathbb{E}(A_{t2}^{50}) = 1848.0 \]
\[ \mathbb{E}(A_{t1}^{250}) = 1917.7 < \mathbb{E}(A_{t2}^{250}) = 1938.0 \]
## Simulation and evaluation actual coverage

<table>
<thead>
<tr>
<th>( n )</th>
<th>( \mathbb{E}(A^n_{t_1}) ) Sim. std.</th>
<th>( \mathbb{E}(A^n_{t_2}) ) Sim. std.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>197.0 213.3 50.1</td>
<td>156.8 155.1 60.8</td>
</tr>
<tr>
<td>5</td>
<td>729.1 762.1 84.0</td>
<td>639.8 629.7 107.0</td>
</tr>
<tr>
<td>10</td>
<td>1076.9 1112.6 104.8</td>
<td>1032.1 1013.3 114.8</td>
</tr>
<tr>
<td>50</td>
<td>1704.6 1743.3 62.4</td>
<td>1848.0 1831.2 39.5</td>
</tr>
<tr>
<td>250</td>
<td>1917.7 1925.8 11.2</td>
<td>1938.0 1938.0 0.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( n )</th>
<th>( \mathbb{E}(A^n_{t_1}) ) Sim. std.</th>
<th>( \mathbb{E}(A^n_{t_2}) ) Sim. std.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>197.0 229.1 48.4</td>
<td>156.8 174.7 52.0</td>
</tr>
<tr>
<td>5</td>
<td>729.1 813.9 56.5</td>
<td>639.8 711.6 89.4</td>
</tr>
<tr>
<td>10</td>
<td>1076.9 1167.9 94.8</td>
<td>1032.1 1133.6 92.9</td>
</tr>
<tr>
<td>50</td>
<td>1704.6 1757.3 11.1</td>
<td>1848.0 1890.7 19.4</td>
</tr>
<tr>
<td>250</td>
<td>1917.7 1926.0 11.1</td>
<td>1938.0 1938.0 0.0</td>
</tr>
</tbody>
</table>
Conclusions and future work

Main results

- New notion of coverage: actual coverage
- Evaluating actual coverage of a given execution
- Predicting actual coverage of a test case or test suite
Main results

- New notion of coverage: actual coverage
- Evaluating actual coverage of a given execution
- Predicting actual coverage of a test case or test suite

For more details, see my Master's Thesis
(www.home.cs.utwente.nl/~timmer)
Main results

- New notion of coverage: *actual coverage*
- Evaluating actual coverage of a given execution
- Predicting actual coverage of a test case or test suite

For more details, see my Master’s Thesis
(wwwhome.cs.utwente.nl/~timmer)

Directions for future work

- Validation of the framework: tool support, case studies
- Dependencies between errors
- Accuracy of approximations
- On-the-fly test derivation
Questions

Actual Test Coverage for Embedded Systems