Interpreting a successful testing process: risk and actual coverage

Mariëlle Stoelinga, Mark Timmer
University of Twente

3rd IEEE International Symposium on Theoretical Aspects of Software Engineering

July 31, 2009
Why testing?

- Software becomes more and more complex
- Research showed that billions can be saved by testing better
- No need for the source code (black-box perspective)
Why testing?

- Software becomes more and more complex
- Research showed that billions can be saved by testing better
- No need for the source code (black-box perspective)

Model-based testing

- Precise and formal
- Automatic generation and evaluations of tests
- Repeatable and scientific basis for product testing
Why do we need risk and coverage?

- Testing is inherently incomplete
- Testing does increase our confidence in the system
- A notion of *quality* of a test suite is necessary
- Two fundamental concepts: risk and coverage

Informal calculation

Coverage: 

\[ \frac{6}{13} = 46\% \]

Risk: 

\[ 7 \cdot 0.1 \cdot \$10 = \$7 \]
Why do we need risk and coverage?

- Testing is inherently incomplete
- Testing does increase our confidence in the system
- A notion of *quality* of a test suite is necessary
- Two fundamental concepts: risk and coverage
Why do we need risk and coverage?

- Testing is inherently incomplete
- Testing does increase our confidence in the system
- A notion of *quality* of a test suite is necessary
- Two fundamental concepts: risk and coverage
Why do we need risk and coverage?

- Testing is inherently incomplete
- Testing does increase our confidence in the system
- A notion of *quality* of a test suite is necessary
- Two fundamental concepts: risk and coverage
Why do we need risk and coverage?

- Testing is inherently incomplete
- Testing does increase our confidence in the system
- A notion of *quality* of a test suite is necessary
- Two fundamental concepts: risk and coverage

Informal calculation

Coverage: \( \frac{6}{13} = 46\% \)
Why do we need risk and coverage?

- Testing is inherently incomplete
- Testing does increase our confidence in the system
- A notion of quality of a test suite is necessary
- Two fundamental concepts: risk and coverage

Informal calculation

Coverage: $\frac{6}{13} = 46\%$

Risk: $7 \cdot 0.1 \cdot $10 = $7$
## Existing coverage measures

- Statement coverage
- State/transition coverage

**Limitations:**

- All faults are considered of equal severity
- Likely locations for fault occurrence are not taken into account
- Syntactic point of view

**Existing risk measures**

- Bach
- Amland

**Limitations:**

- Informal
- Based on heuristics
- Only identify testing order for components
## Existing coverage measures

<table>
<thead>
<tr>
<th></th>
<th>Statement coverage</th>
<th>State/transition coverage</th>
</tr>
</thead>
</table>

### Limitations:
- all faults are considered of equal severity
- likely locations for fault occurrence are not taken into account
- syntactic point of view
## Existing coverage measures

- Statement coverage
- State/transition coverage

**Limitations:**
- all faults are considered of equal severity
- likely locations for fault occurrence are not taken into account
- syntactic point of view

## Existing risk measures

- Bach
- Amland
## Introduction – Existing approaches

<table>
<thead>
<tr>
<th>Existing coverage measures</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Statement coverage</td>
</tr>
<tr>
<td>• State/transition coverage</td>
</tr>
</tbody>
</table>

**Limitations:**
- all faults are considered of equal severity
- likely locations for fault occurrence are not taken into account
- syntactic point of view

<table>
<thead>
<tr>
<th>Existing risk measures</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Bach</td>
</tr>
<tr>
<td>• Amland</td>
</tr>
</tbody>
</table>

**Limitations:**
- Informal
- Based on heuristics
- Only identify testing order for components
Starting point: semantic coverage

Previous work by Brandán Briones, Brinksma and Stoelinga

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

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

Labelled transition systems

\[ s_0 \xrightarrow{10\text{ct}? \ coffee!} s_1 \quad s_0 \xleftarrow{20\text{ct}? \ tea!} s_2 \]
Starting point: semantic coverage

Previous work by Brandán Briones, Brinksma and Stoelinga

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

Labelled transition systems

\[
\begin{array}{c}
\delta \\
\end{array}
\]

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

Previous work by Brandán Briones, Brinksma and Stoelinga

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

Labelled transition systems

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

\[ \begin{align*}
s_1 \xleftarrow{\text{coffee!}} & \quad s_0 & \xrightarrow{\text{tea!}} \quad s_2
\end{align*} \]

\[ \begin{align*}
10\text{ct}? & \quad \text{coffee!} & 20\text{ct}? & \quad \text{tea!} & \quad \delta
\end{align*} \]
Starting point: semantic coverage

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

Labelled transition systems

\[ s_0 \xrightarrow{\delta} s_1 \quad \text{coffee!} \]
\[ s_0 \xrightarrow{\delta} s_2 \quad \text{tea!} \]

\[ s_1 \xrightarrow{10\text{ct?}} \text{coffee!} \]
\[ s_2 \xrightarrow{20\text{ct?}} \text{tea!} \]

Test cases

\[ \text{10ct?} \]
\[ \text{coffee!} \]
\[ \text{tea!} \]

Interpreting a successful testing process: risk and actual coverage
Starting point: semantic coverage

Previous work by Brandán Briones, Brinksma and Stoelinga

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

Labelled transition systems

\[
\begin{align*}
& s_1 \rightarrow s_0 \rightarrow s_2 \\
& \delta \\
& 10\text{ct}? \quad 20\text{ct}? \quad \text{coffee!} \quad \text{tea!} \\
& \end{align*}
\]

Test cases

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

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

Labelled transition systems

\[
\begin{align*}
\delta \quad 10\text{ct}? & \quad 20\text{ct}\
10\text{ct}? & \quad \text{coffee!} \quad 20\text{ct}? & \quad \text{tea!} \\
\text{coffee!} & \quad s_1 & \quad s_0 & \quad s_2 & \quad \text{coffee!} \quad \text{tea!} \\
10\text{ct}? & \quad \text{coffee!} & \quad 20\text{ct}? & \quad \text{tea!} & \quad \delta
\end{align*}
\]

Test cases

\[
\begin{align*}
10\text{ct}? & \quad \delta \quad \text{coffee!} \quad \text{tea!} \\
\text{fail} & \quad \delta \quad \text{coffee!} \quad \text{tea!} \\
\text{fail} & \quad \text{fail}
\end{align*}
\]
Starting point: 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

Interpreting a successful testing process: risk and actual coverage
A WFS consists of

- An LTS (expected system behaviour)
- An error function (probability of faults)
- A weight function (severity of faults)
A WFS consists of

- An LTS (expected system behaviour)
- An error function (probability of faults)
- A weight function (severity of faults)

\[
\delta(p_{err}(10\text{ct? coffee!}) = 0.2 \\
p_{err}(20\text{ct? tea!} = 0.03 \\
w(\epsilon) = 10 \\
w(10\text{ct?}) = 15 \\
w(10\text{ct? coffee!}) = 9.5
\]

Interpreting a successful testing process: risk and actual coverage
A WFS consists of

- An LTS (expected system behaviour)
- An error function (probability of faults)
- A weight function (severity of faults)

\[ p_{err}(10\text{ct? coffee!}) = 0.02 \]
\[ p_{err}(20\text{ct? tea!}) = 0.03 \]
A WFS consists of
- An LTS (expected system behaviour)
- An error function (probability of faults)
- A weight function (severity of faults)

\[ p_{err}(10\text{ct? coffee!}) = 0.02 \]
\[ p_{err}(20\text{ct? tea!}) = 0.03 \]
\[ w(\epsilon) = 10 \]
\[ w(10\text{ct?}) = 15 \]
\[ w(10\text{ct? coffee!}) = 9.5 \]
Fault weight: 10 + 15 = 25

(We are only interested in whether a fault can occur, not in which one)

Interpreting a successful testing process: risk and actual coverage

\[ w(\epsilon) = 10 \]
\[ w(10\text{ct?}) = 15 \]
\[ w(10\text{ct? coffee!}) = 9.5 \]
The WFS Model – Fault Weight

Interpreting a successful testing process: risk and actual coverage

- $w(\epsilon) = 10$
- $w(10\text{ct}?) = 15$
- $w(10\text{ct}? \text{ coffee!}) = 9.5$

Fault weight: $10 + 15 = 25$

(We are only interested in whether a fault can occur, not in which one)
Interpreting a successful testing process: risk and actual coverage

The WFS Model – Fault Weight

Fault weight: 10 + 15 = 25
(We are only interested in whether a fault can occur, not in which one)

Interpreting a successful testing process: risk and actual coverage

The WFS Model
Jul. 31, 2009 8 / 17
Fault weight: $10 + 15 = 25$
Fault weight: $10 + 15 = 25$

(We are only interested in whether a fault can occur, not in which one)
Definition

Given a test suite \( T \) and a passing execution \( E \), we define

\[
\text{risk}(T, E) = \mathbb{E}[w(\text{Impl}) \mid \text{observe } E]
\]

i.e., the fault weight still expected to be present after observing \( E \).
Given a test suite $T$ and a passing execution $E$, we define

$$\text{risk}(T, E) = \mathbb{E}[w(\text{Impl}) | \text{observe } E]$$

i.e., the fault weight still expected to be present after observing $E$.

Observe:

$$\text{risk}(\langle \rangle, \langle \rangle) =$$
Given a test suite $T$ and a passing execution $E$, we define

$$\text{risk}(T, E) = \mathbb{E}[\text{w(Impl)} \mid \text{observe } E]$$

i.e., the fault weight still expected to be present after observing $E$.

Observe:

$$\text{risk}('\langle \rangle', '\langle \rangle') = \sum_{\sigma} w(\sigma) \cdot p_{\text{err}}(\sigma)$$
Nondeterministic output behaviour yields difficulties.

How to calculate risk (expected fault presence)?

Interpreting a successful testing process: risk and actual coverage
Nondeterministic output behaviour yields difficulties.

How to calculate risk (expected fault presence)?

Interpreting a successful testing process: risk and actual coverage
Nondeterministic output behaviour yields difficulties. How to calculate risk (expected fault presence)?

Interpreting a successful testing process: risk and actual coverage
Nondeterministic output behaviour yields difficulties.
Nondeterministic output behaviour yields difficulties.

How to calculate risk (expected fault presence)?
Nondeterministic output behaviour yields difficulties.

How to calculate risk (expected fault presence)?

\[
\text{risk}(\langle \rangle, \langle \rangle) = \sum_{\sigma} w(\sigma) \cdot p_{\text{err}}(\sigma)
\]
Nondeterministic output behaviour yields difficulties.

How to calculate risk (expected fault presence)?

\[
\text{risk}(T, E) = \sum_{\sigma \neq 10\text{ct?}} w(\sigma) \cdot p_{\text{err}}(\sigma) + f(10\text{ct?})
\]
Nondeterministic output behaviour yields difficulties.

How to calculate risk (expected fault presence)?

\[
\text{risk}(T, E) = \sum_{\sigma \neq 10\text{ct?}} w(\sigma) \cdot p_{err}(\sigma) + f(10\text{ct?})
\]
Nondeterministic output behaviour yields difficulties.

How to calculate risk (expected fault presence)?

$$\text{risk}(T, E) = \sum_{\sigma \neq 10\text{ct?}} w(\sigma) \cdot p_{\text{err}}(\sigma) + w(10\text{ct?}) \cdot \mathbb{P}[\text{error after } 10\text{ct?} | E]$$
Weighted fault specification

A WFS consists of

- An LTS (expected system behaviour)
- An error function (probability of faults)
- A weight function (severity of faults)
- A failure function (probability of failure in case of fault)
Weighted fault specification

A WFS consists of

- An LTS (expected system behaviour)
- An error function (probability of faults)
- A weight function (severity of faults)
- A failure function (probability of failure in case of fault)
Weighted Fault Specifications (revisited)

A WFS consists of

- An LTS (expected system behaviour)
- An error function (probability of faults)
- A weight function (severity of faults)
- A failure function (probability of failure in case of fault)

\[ p_{\text{fail}}(\epsilon) = 1.0 \]
\[ p_{\text{fail}}(10\text{ct?}) = 0.5 \]
Interpreting a successful testing process: risk and actual coverage

\[
\text{risk}(T, E) = \sum_{\sigma \neq 10\text{ct?}} w(\sigma) \cdot p_{\text{err}}(\sigma) + w(10\text{ct?}) \cdot \mathbb{P}[\text{error after 10ct?} \mid E]
\]

\[
\text{risk} \left( T, E \right) = \sum_{\sigma \neq 10\text{ct?}} w(\sigma) \cdot p_{\text{err}}(\sigma) + w(10\text{ct?}) \cdot \mathbb{P}[\text{error after 10ct?} \mid E]
\]
risk\( (T, E) \)

\[
= \sum_{\sigma \neq 10\text{ct}?} w(\sigma) \cdot p_{\text{err}}(\sigma) + w(10\text{ct}?) \cdot \Pr[\text{error after 10ct?} | E]
\]

\[
= \sum_{\sigma \neq 10\text{ct}?} w(\sigma) \cdot p_{\text{err}}(\sigma) +
\]

\[
w(10\text{ct}?) \cdot \frac{(1 - p_{\text{fail}}(10\text{ct}?)) \cdot p_{\text{err}}(10\text{ct}?)}{(1 - p_{\text{fail}}(10\text{ct}?)) \cdot p_{\text{err}}(10\text{ct}?) + (1 - p_{\text{err}}(10\text{ct}?))}
\]
Calculation of risk

\[
\text{risk}(T, E) = \text{risk}() - \sum_{\sigma \in E} w(\sigma) \cdot \left( p_{\text{err}}(\sigma) - \frac{(1 - p_{\text{fail}}(\sigma))^{\text{obs}(\sigma, E)} \cdot p_{\text{err}}(\sigma)}{(1 - p_{\text{fail}}(\sigma))^{\text{obs}(\sigma, E)} \cdot p_{\text{err}}(\sigma) + 1 - p_{\text{err}}(\sigma)} \right)
\]

with \( \text{obs}(\sigma, E) \) the number of observations in \( E \) after \( \sigma \).
Risk

Calculation of risk

\[
\text{risk}(T, E) = \text{risk}(\langle \rangle, \langle \rangle) - \sum_{\sigma \in E} w(\sigma) \cdot \left( p_{\text{err}}(\sigma) - \frac{(1 - p_{\text{fail}}(\sigma))^{\text{obs}(\sigma, E)} \cdot p_{\text{err}}(\sigma)}{(1 - p_{\text{fail}}(\sigma))^{\text{obs}(\sigma, E)} \cdot p_{\text{err}}(\sigma) + 1 - p_{\text{err}}(\sigma)} \right)
\]

with \(\text{obs}(\sigma, E)\) the number of observations in \(E\) after \(\sigma\).

Although \(\text{risk}(\langle \rangle, \langle \rangle) = \sum_{\sigma} w(\sigma) \cdot p_{\text{err}}(\sigma)\) is an infinite sum, it can be calculated smartly:

- \(w\) defined by truncation: the sum is already finite
- \(w\) defined by discounting: system of linear equations
### Compute test suite quality in advance

- Estimate correct system behaviour
- Compute expected risk after passing the test suite
### Compute test suite quality in advance
- Estimate correct system behaviour
- Compute expected risk after passing the test suite

### Optimisation
- Find the optimal test suite of a given size
- Apply history-dependent backwards induction (Markov Decision Theory)
Other Applications

- Compute test suite quality in advance
  - Estimate correct system behaviour
  - Compute expected risk after passing the test suite

- Optimisation
  - Find the optimal test suite of a given size
  - Apply history-dependent backwards induction (Markov Decision Theory)

- Actual Coverage
  - Only consider the traces that were actually tested
  - Use error probability reduction as coverage measure
  - Methods very similar to risk
Probabilities might be hard to find, but

- We show what can be calculated, and the required ingredients
- We facilitate sensitivity analysis
- To compute numbers, we have to start with numbers...
### Limitations and Possibilities

Probabilities might be hard to find, but
- We show what can be calculated, and the required ingredients
- We facilitate sensitivity analysis
- To compute numbers, we have to start with numbers...

It looks like we need many probabilities and weights, but
- The framework can be applied at higher levels of abstraction
- Compute risk based on error / failure probabilities of modules
Conclusions and Future Work

Main results

- Formal notion of risk
- Both evaluation of risk and computation of optimal test suite
- Easily adaptable to be used as a coverage measure
Conclusions and Future Work

Main results
- Formal notion of risk
- Both evaluation of risk and computation of optimal test suite
- Easily adaptable to be used as a coverage measure

Directions for Future Work
- Validation of the framework: tool support, case studies
- Dependencies between errors
- On-the-fly test derivation