2009 Third IEEE International Symposium on Theoretical Aspects of Software Engineering

# Interpreting a Successful Testing Process: Risk and Actual Coverage

Mariëlle Stoelinga and Mark Timmer
*Formal Methods & Tools Group, Department of Computer Science*
*University of Twente, The Netherlands*
{*marielle, timmer*}*@cs.utwente.nl*

***Abstract*—Testing is inherently incomplete; no test suite will ever be able to test all possible usage scenarios of a system. It is therefore vital to assess the implication of a system passing a test suite. This paper quantifies that implication by means of two distinct, but related, measures: the *risk* quantifies the confidence in a system after it passes a test suite, i.e., the number of faults still expected to be present (weighted by their severity); the *actual coverage* quantifies the extent to which faults have been shown absent, i.e., the fraction of possible faults that has been covered. We provide evaluation algorithms that calculate these metrics for a given test suite, as well as optimisation algorithms that yield the best test suite for a given optimisation criterion.**

## I. Introduction

Software becomes more and more complex, making thorough testing an indispensable part of the development process. The U.S. National Institute of Standards and Technology has assessed that software faults cost the American economy almost sixty billion dollars annually [1]. More than a third of these costs could be eliminated if testing occurred earlier in the development process.

An important fact about testing is that it is inherently incomplete, since testing everything would require infinitely many input scenarios. On the other hand, passing a well-designed test suite does increase the confidence in the correctness of the tested product. Therefore, it is important to assess the quality of a test suite. Two fundamental concepts have been put forward to evaluate test suite quality: (1) *coverage metrics* determine which portion of the requirements and/or implementation-under-test has been exercised by the test suite; (2) *risk-based metrics* assess the risk of putting the tested product into operation.

Although existing coverage measures, such as code coverage in white-box testing ([2], [3]) and state and/or transition coverage in black-box testing ([4], [5], [6]), give an indication of the quality of a test suite, it is not necessarily true that higher coverage implies that more, or more severe, faults are detected. This is because these metrics do not take into account where in the system faults are most likely to occur. Risk-based testing methods do aim at reducing the expected number of faults, or their severity. However, these are often informal [7], based on heuristics [8], or indicate which components should be tested best [9], but rarely quantify the risk after a successful testing process in a precise way.

In this paper, we present a framework in which risk and coverage can be defined, computed and optimised in a black-box manner, for systems with nondeterminism. Key properties are a rigorous mathematical treatment based on solid probabilistic models, and the result that lower risk (or higher coverage) implies a lower expected number of faults.

**Overview.** The starting point in our theory is a *weighted fault specification (WFS)*, consisting of (1) a specification describing the desired system behaviour as an input-output labelled transition system (IOLTS), (2) a weight function describing the severity of faults, (3) an error function describing the probability that a certain error has been made, and (4) a failure function describing the probability that incorrectly implemented behaviour yields a failure. The error probabilities are assumed to be independent, which is partly justified by abstracting the actual inputs into equivalence classes. Still, this assumption is quite strict, but we think that a thorough understanding of simple models is the best start when tackling the more complex situation with dependent probabilities. The failure function is based on the fact that, due to nondeterminism, observing a correct response once does not yet imply correctness. That is, a system might respond differently to the same inputs during different executions.

From the WFS we derive its underlying probability model, i.e., a random variable that describes the distribution of (possibly erroneous) implementations. This allows us to define risk and actual coverage in an easy and precise way.

Given a WFS, we define the *risk* of a test suite as the expected fault weight that remains after this test suite passes. We show how to construct a test suite of a certain size with minimal expected risk. We also introduce *actual coverage* for a test suite, which quantifies the risk reduction obtained when an implementation passes it. Whereas risk is based on faults contained within the entire system, actual coverage

This research has been partially funded by NWO under FOCUS/BRICKS grant 642.000.505 (MOQS) and by the European Union under grant FP7-ICT-2007-1 (QUASIMODO).

978-0-7695-3757-3/09 $25.00 © 2009 IEEE
DOI 10.1109/TASE.2009.26

only relates to the part of the system that has been tested. This matches with the traditional interpretation of coverage.

Our methods refine the theory presented by Brandán Briones, Brinksma, and Stoelinga [10]. They introduced a concept we would call *potential coverage*, as it considers which faults *can* be detected during testing. Our measures, however, take into account the faults that *are* actually covered during a test execution, making them more precise.

While error probabilities and failure probabilities are important ingredients of our framework, techniques for obtaining them fall outside the scope of this paper. However, there is extensive literature on factors that determine them. For instance, estimations of the error probabilities can be based on the software change history [11]. They can also be based on McCabe's cyclomatic complexity number [12], Halstead's set of Software Science metrics [13], and requirements volatility [14]. The failure probabilities can be obtained by applying one of the many analysis techniques described in [15] and [16]. In practice it might still be difficult to estimate all the probabilities that are needed, asking for simplifying approximations. This paper could then serve as a baseline for sensitivity analysis [17], making it possible to assess the impact of these simplifications.

Finally, we note that our measures can easily be applied at higher abstraction levels. For instance, instead of defining behaviour in terms of basic actions, it could be defined in terms of function or module calls. A fault weight then denotes the severity of an error in a certain module, and error and failure probabilities describe respectively the expected presence of faults and occurrence of failures in the modules, providing risk and coverage measures for module testing.

**Organisation of the paper.** The model is described in Section II, risk in Section III, and actual coverage in Section IV. Section V discusses conclusions and future work.

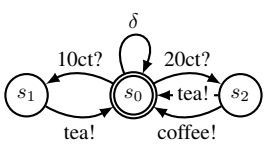
Due to space limitations, we refer the reader to the extended version of the current paper [18] for the proofs.

## II. The WFS Model

### A. Preliminaries and notations

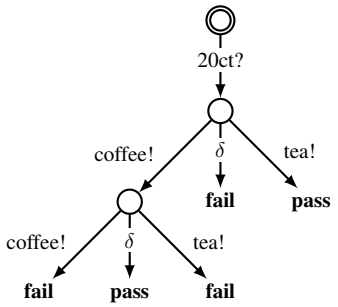
**Definition 1** (Preliminaries)**.** *Given a set $L$, the set of all sequences over $L$ is denoted by $L^*$, and the set of non-empty sequences by $L^+$. If $\sigma, \rho \in L^*$, then $\sigma$ is a* prefix *of $\rho$ (denoted $\sigma \sqsubseteq \rho$) if there is a $\sigma' \in L^*$ such that $\sigma\sigma' = \rho$. If $\sigma' \in L^+$, then $\sigma$ is a* proper prefix *of $\rho$ (denoted $\sigma \sqsubset \rho$).*

We model systems by IOLTSs, which describe behaviour by means of states and transitions [19]. Transitions are always caused by either an input action or an output action. In order to be able to model realistic systems using IOLTSs, we apply equivalence partitioning. That is, the inputs are partitioned into *equivalence classes* such that one input is representative for all others in its class [3]. This way, the action set can be kept finite, often even small.




| state | $w$ | $p_{\text{err}}$ | $p_{\text{fail}}$ |
|---|---|---|---|
| $s_0$ | 40 | 0.01 | 0.4 |
| $s_1$ | 25 | 0.07 | 0.5 |
| $s_2$ | 25 | 0.05 | 0.4 |

(a) A WFS $W$




(b) A test case $t$

Figure 1. A WFS and a test case

**Definition 2** (IOLTSs)**.** *An* IOLTS *$\mathcal{A}$ is a tuple $\langle S, s^0, L, \Delta \rangle$, where*

- *$S$ is a finite set of states, with $s^0 \in S$ the initial state;*
- *$L$ is a finite set of actions, partitioned into a set $L_{\text{I}}$ of inputs (suffixed by a question mark) and a set $L_{\text{O}}$ of outputs (suffixed by an exclamation mark);*
- *$\Delta \subseteq S \times L \times S$ is the transition relation, which is required to be (internally) deterministic. Formally, $(s, a, s') \in \Delta \wedge (s, a, s'') \in \Delta \implies s' = s''$.*

*We write $\mathcal{A}_{\text{spec}}$ to denote a specification. An IOLTS $\mathcal{A}_{\text{impl}}$ is a (potentially incorrect) implementation of $\mathcal{A}_{\text{spec}}$ if it has the same alphabet and is* input-enabled*, i.e., for all $s \in S$ and $a \in L_{\text{I}}$, there exists an $s' \in S$ with $(s, a, s') \in \Delta$. The set of possible implementations of $\mathcal{A}$ is denoted by $\text{IMPL}_{\mathcal{A}}$.*

**Definition 3** (Paths and traces)**.** *Let $\mathcal{A} = \langle S, s^0, L, \Delta \rangle$ be an IOLTS, then a* path *in $\mathcal{A}$ is a finite sequence of states and actions $\pi = s_0 a_1 s_1 a_2 \ldots a_n s_n$, with $s_0 = s^0$ and $\forall i \in \{0, \ldots, n-1\} : (s_i, a_{i+1}, s_{i+1}) \in \Delta$. The set of all paths in $\mathcal{A}$ is denoted by $\text{paths}_{\mathcal{A}}$.*

*Each path $\pi$ has a* trace *associated with it, denoted by $\text{trace}(\pi)$ and given by the sequence of the actions of $\pi$. From the set of all paths in $\mathcal{A}$ we can deduce the set of all traces in $\mathcal{A}$: $\text{traces}_{\mathcal{A}} = \{\text{trace}(\pi) \mid \pi \in \text{paths}_{\mathcal{A}}\}$.*

*We use $\mathcal{A}[\sigma]$ for the set of outputs that $\mathcal{A}$ can provide as a response to $\sigma$, i.e., $\mathcal{A}[\sigma] = \{b! \in L_{\text{O}} \mid \sigma b! \in \text{traces}_{\mathcal{A}}\}$.*

*A trace $\sigma$ is* implemented incorrected *by $\mathcal{A}_{\text{impl}}$ if $\mathcal{A}_{\text{impl}}$* might *respond incorrectly to it, i.e., if $\mathcal{A}_{\text{impl}}[\sigma] \not\subseteq \mathcal{A}_{\text{spec}}[\sigma]$.*

*Example* 1. The upper part of Figure 1(a) shows an IOLTS. Its states are represented by circles, and its initial state by an extra inner circle. The special action $\delta$ (quiescence) is used to denote that the absence of any output action is required.

An example path in $\mathcal{A}$ is ($s_0$ 20ct? $s_2$ coffee! $s_0$ 10ct? $s_1$ tea! $s_0$). The corresponding trace is (20ct? coffee! 10ct? tea!). It holds that $\mathcal{A}[\text{10ct?}] = \{\text{tea!}\}$. (For readability, parentheses are often placed around traces and paths.) □

### B. Weighted Fault Specifications

Since it is uncertain which faults are introduced, developing an implementation can be described by a random

experiment. For each trace, we specify the probability that an implementation might respond incorrectly; its *error probability*. These probabilities are assumed to be independent, which corresponds to the assumptions made in equivalence partitioning. There, the assumption is that correctness of a single input of an equivalence class implies correctness of all other inputs in its class, whereas it implies nothing about the correctness of inputs in other equivalence classes.

Since not all failures that can occur at a certain trace will actually occur when executing that trace once, we specify a *failure function*. This function yields, for any trace $\sigma$, the probability that an implementation produces an incorrect output directly after $\sigma$ during an arbitrary execution.

Finally, a *fault weight* is specified for each trace, denoting the severity of an incorrect implementation with respect to that trace (or rather, its equivalence class). The higher a fault weight, the higher the severity.

A specification together with fault weights, error probabilities, and failure probabilities, constitutes a *weighted fault specification*.

**Definition 4** (WFSs)**.** *A* WFS (weighted fault specification) *is a tuple* $W = \langle \mathcal{A}_{\text{spec}}, w, p_{\text{err}}, p_{\text{fail}} \rangle$*, with*

- $\mathcal{A}_{\text{spec}} = \langle S, s^0, L, \Delta \rangle$ *an IOLTS;*
- $w\colon \text{traces}_{\mathcal{A}_{\text{spec}}} \to \mathbb{R}^{\geq 0}$ *a* weight function *assigning a* fault weight *to each trace of* $\mathcal{A}_{\text{spec}}$*, such that* $0 < \sum_{\sigma \in \text{traces}_{\mathcal{A}_{\text{spec}}}} w(\sigma) < \infty$*. This constraint allows us to define a coverage notion relative to the total weight* $\sum_{\sigma \in \text{traces}_{\mathcal{A}_{\text{spec}}}} w(\sigma)$*. The fault weight* $w(\sigma)$ *denotes the severity of an erroneous output directly after* $\sigma$*;*
- $p_{\text{err}}\colon \text{traces}_{\mathcal{A}_{\text{spec}}} \to [0,1]$ *an* error function *assigning to each trace* $\sigma$ *of* $\mathcal{A}_{\text{spec}}$ *the probability* $p_{\text{err}}(\sigma)$ *that an implementation can provide an incorrect output directly after* $\sigma$*, i.e., that for an arbitrary* $\mathcal{A}_{\text{impl}}$ *it holds that* $\mathcal{A}_{\text{impl}}[\sigma] \not\subseteq \mathcal{A}_{\text{spec}}[\sigma]$*;*
- $p_{\text{fail}}\colon \text{traces}_{\mathcal{A}_{\text{spec}}} \to [0,1]$ *a* failure function *assigning to each trace* $\sigma$ *of* $\mathcal{A}_{\text{spec}}$ *the probability* $p_{\text{fail}}(\sigma)$ *that an arbitrary implementation* $\mathcal{A}_{\text{impl}}$ *with* $\mathcal{A}_{\text{impl}}[\sigma] \not\subseteq \mathcal{A}_{\text{spec}}[\sigma]$ *responds incorrectly to* $\sigma$*.*

*An implementation of* $W$ *is an implementation of* $\mathcal{A}_{\text{spec}}$*.*

Since $w$, $p_{\text{err}}$ and $p_{\text{fail}}$ have infinite domains, they cannot be specified directly, and we need a finite way of representing them. We will specify $p_{\text{err}}$ in an easy way by simply assigning a value $p_{\text{err}}(s)$ to each state $s$, and defining $p_{\text{err}}(\sigma) = p_{\text{err}}(\text{last}(\sigma))$, where $\text{last}(\sigma)$ is the last state of the path associated with $\sigma$. Analogously, $p_{\text{fail}}$ is defined.

Following [10], $w$ can be specified by truncation, i.e., explicitly specifying the fault weight of all traces smaller than a certain size and defining all others to have fault weight zero. Alternatively, fault weights can be assigned to the states and a discount factor $\lambda$ can be used to determine the fault weights of traces; that is, $w(\sigma) = w(\text{last}(\sigma)) \cdot \lambda^{|\sigma|}$, where $|\sigma|$ is the number of transitions of $\sigma$. Choosing $0 \leq \lambda < \frac{1}{m}$, with $m$ the maximal outdegree of the IOLTS, this keeps the accumulated fault weight of all traces finite. Note that it is also sufficient to only apply this restriction to the derivation for traces larger than some threshold, and use a different (or no) discount factor for the shorter traces.

Our framework does not rely on the way $w$, $p_{\text{err}}$, and $p_{\text{fail}}$ are specified, and thus can handle any of the above methods.

*Example* 2. In all examples we will use $\lambda = 0.9$ for short traces (all traces explicitly used in examples being short). A specification of $w$, $p_{\text{err}}$ and $p_{\text{fail}}$ for the states of the IOLTS of the previous example is shown in Figure 1(a). Given for example $\sigma$ = (20ct? coffee!), consequently $w(\sigma) = 40 \cdot 0.9^2 = 32.4$. Also, $p_{\text{err}}(\sigma) = 0.01$ and $p_{\text{fail}}(\sigma) = 0.4$. □

The fault weight of an implementation $\mathcal{A}_{\text{impl}}$ is defined as the total fault weight of all incorrectly implemented traces.

**Definition 5** (Fault weight)**.** *Let* $W = \langle \mathcal{A}_{\text{spec}}, w, p_{\text{err}}, p_{\text{fail}} \rangle$ *be a WFS and* $\mathcal{A}_{\text{impl}}$ *an implementation of* $W$*, then the* fault weight *of* $\mathcal{A}_{\text{impl}}$ *is defined by*

$$w(\mathcal{A}_{\text{impl}}) = \sum_{\substack{\sigma \in \text{traces}_{\mathcal{A}_{\text{spec}}} \\ \mathcal{A}_{\text{impl}}[\sigma] \not\subseteq \mathcal{A}_{\text{spec}}[\sigma]}} w(\sigma) \ ,$$

*which is less than infinity by the assumptions on* $w$*.*

### C. Test Cases and Test Suites

To investigate the quality of systems, test cases and suites are used. Following ioco theory [19], we require test cases for IOLTSs to be *fail fast*; they stop directly after observing a failure. Test cases repeatedly either perform an input action or observe which output action a system provides.

**Definition 6** (Test cases and suites)**.** *(i) A* test case $t$ *for an IOLTS* $\mathcal{A}_{\text{spec}} = \langle S, s^0, L, \Delta \rangle$ *is a prefix-closed, finite subset of* $L^*$*, such that for all* $\sigma \in L^*$*,* $a? \in L_{\text{I}}$*, and* $a! \in L_{\text{O}}$

- *if* $\sigma a? \in t$*, then* $\forall b \in L\colon b \neq a? \implies \sigma b \notin t$*;*
- *if* $\sigma a! \in t$*, then* $\forall b! \in L_{\text{O}}\colon \sigma b! \in t$*;*
- *if* $\sigma \notin \text{traces}_{\mathcal{A}_{\text{spec}}}$*, then* $\forall \sigma' \in L^+\colon \sigma\sigma' \notin t$*.*

*A* test suite $T$ *is a tuple of test cases, denoted* $\langle t_1, \ldots, t_n \rangle$*.*

*(ii) An* execution *of* $t$ *is a trace* $\sigma \in t$ *such that there is no* $\rho \in t$ *with* $\sigma \sqsubset \rho$*, i.e.,* $\sigma$ *is a maximal element of* $t$*. The set of all executions of* $t$ *is denoted by* $\text{exec}_t$*. An* observing trace *of* $t$ *is a trace that is followed by an observation, i.e., a trace* $\sigma \in t$ *such that* $\forall b! \in L_{\text{O}}\colon \sigma b! \in t$*.*

*(iii) An execution of a test suite* $\langle t_1, \ldots, t_n \rangle$ *is a sequence* $E = \langle \sigma_1, \ldots, \sigma_n \rangle$*, such that* $\sigma_i$ *is an execution of* $t_i$ *for all* $i$*. It is a* correct execution *if* $\sigma_i \in \text{traces}_{\mathcal{A}_{\text{spec}}}$ *for all* $i$*.*

*(iv) For each test suite execution* $E = \langle \sigma_1, \ldots, \sigma_n \rangle$ *and trace* $\sigma \in L^*$*, we define* $\text{obs}(\sigma, E)$ *as the number of times* $E$ *observed directly after* $\sigma$*, i.e.,* $\text{obs}(\sigma, E) = |\{i \mid \forall b! \in L_{\text{O}}\colon \sigma b! \sqsubseteq \sigma_i\}|$*. We use* $\text{obs}(\sigma, T)$ *to denote the number of times an execution of* $T$ *might observe after* $\sigma$*, i.e.,* $\text{obs}(\sigma, T) = |\{i \mid \forall b! \in L_{\text{O}}\colon \sigma b! \in t_i\}|$*. The set of all* observing traces *of* $T$ *is given by* $\text{obs}_T = \bigcup_{t_i \in T} \{\sigma \in t_i \mid \forall b! \in L_{\text{O}}\colon \sigma b! \in t_i\}$*.*

*Example* 3. A test case $t$ for the IOLTS of the previous examples is shown in Figure 1(b). We have $\mathrm{exec}_t = \{(20\text{ct? } \delta), (20\text{ct? tea!}), (20\text{ct? coffee! coffee!}), (20\text{ct? coffee! } \delta), (20\text{ct? coffee! tea!})\}$. The set of observing traces of the test suite $T = \langle t \rangle$ is $\mathrm{obs}_T = \{(20\text{ct?}), (20\text{ct? coffee!})\}$.

Let $T = \langle t, t \rangle$ be a test suite containing $t$ twice. The maximum number of times an execution of $T$ may observe after $\sigma = (20\text{ct? coffee!})$ is $\mathrm{obs}(\sigma, T) = 2$. However, given the execution $E = \langle (20\text{ct? tea!}), (20\text{ct? coffee! } \delta) \rangle$ there was only one such observation, so $\mathrm{obs}(\sigma, E) = 1$. □

### D. Underlying Probability Model

Since we only care which traces are handled incorrectly, and we do not care about the incorrect responses, we partition the possible implementations into classes of implementations that respond correctly to exactly the same traces.

**Definition 7** (Classification relation)**.** *Let $\mathcal{A}_{\mathrm{spec}}$ be a specification, and $\mathcal{A}_{\mathrm{impl}_1}$ and $\mathcal{A}_{\mathrm{impl}_2}$ two of its implementations, then the relation $\sim_{\mathcal{A}_{\mathrm{spec}}}$ is defined by*

$$\mathcal{A}_{\mathrm{impl}_1} \sim_{\mathcal{A}_{\mathrm{spec}}} \mathcal{A}_{\mathrm{impl}_2} \textit{ iff}$$
$$\forall \sigma \in \mathrm{traces}_{\mathcal{A}_{\mathrm{spec}}} : \mathcal{A}_{\mathrm{impl}_1}[\sigma] \subseteq \mathcal{A}_{\mathrm{spec}}[\sigma] \Leftrightarrow \mathcal{A}_{\mathrm{impl}_2}[\sigma] \subseteq \mathcal{A}_{\mathrm{spec}}[\sigma]$$

*We use $[\![\mathcal{A}_{\mathrm{impl}}]\!]_{\mathcal{A}_{\mathrm{spec}}}$ to denote the equivalence class of $\mathcal{A}_{\mathrm{impl}}$ with respect to the relation $\sim_{\mathcal{A}_{\mathrm{spec}}}$, and leave out the subscript $\mathcal{A}_{\mathrm{spec}}$ whenever no confusion arises.*

We lift the fault weight of implementations to classes of implementations by saying that $w([\![\mathcal{A}_{\mathrm{impl}}]\!]) = w(\mathcal{A}_{\mathrm{impl}})$.

The function $p_{\mathrm{err}}$ of a WFS induces a random variable $A_W$ over the equivalence classes of $\sim_{\mathcal{A}_{\mathrm{spec}}}$. Because the number of possible implementations is uncountable, $A_W$ is a continuous random variable and hence the probability of every individual implementation is 0.

**Definition 8** ($A_W$)**.** *Let $W = \langle \mathcal{A}_{\mathrm{spec}}, w, p_{\mathrm{err}}, p_{\mathrm{fail}} \rangle$ be a WFS, then we define $A_W : \Omega \to \mathit{IMPL}_{\mathcal{A}_{\mathrm{spec}}}/\!\sim$ to be the random variable representing the equivalence class of an arbitrary implementation of $W$.*

We will often use the event that $A_W = [\![\mathcal{A}_{\mathrm{impl}}]\!]$ such that $\mathcal{A}_{\mathrm{impl}}[\sigma] \subseteq \mathcal{A}_{\mathrm{spec}}[\sigma]$, and denote this by $A_W[\sigma] \subseteq \mathcal{A}_{\mathrm{spec}}[\sigma]$. Note that by definition $\mathbb{P}[A_W[\sigma] \subseteq \mathcal{A}_{\mathrm{spec}}[\sigma]] = 1 - p_{\mathrm{err}}(\sigma)$.

For each test suite $T$, the error function and failure function also induce a random variable representing the execution of $T$. After all, due to nondeterminism the same test suite might test different traces during different executions.

**Definition 9** ($R_{W,T}$)**.** *Let $W = \langle \mathcal{A}_{\mathrm{spec}}, w, p_{\mathrm{err}}, p_{\mathrm{fail}} \rangle$ be a WFS and $T = \langle t_1, \ldots, t_n \rangle$ a test suite for $\mathcal{A}_{\mathrm{spec}}$, then we define $R_{W,T} : \Omega \to \mathrm{exec}_{t_1} \times \mathrm{exec}_{t_2} \times \cdots \times \mathrm{exec}_{t_n}$ to be the random variable representing the result of executing $T$ against an arbitrary implementation of $W$.*

Note that the distribution of $R_{W,T}$ depends on the distribution of outputs the system provides. However, we do not need the explicit distribution of $R_{W,T}$ here.

## III. Risk

### A. Test Evaluation with Respect to Risk

Having defined a formal framework, we can now define the measures of interest. First of all, when a test suite passes, we want to estimate the number of faults that remained undetected. To also incorporate the severity of these faults, we define the *risk* of an implementation after passing a test suite as its expected remaining fault weight, i.e., the expected number of remaining faults weighted by their severity.

**Definition 10** (Risk)**.** *Let $W = \langle \mathcal{A}_{\mathrm{spec}}, w, p_{\mathrm{err}}, p_{\mathrm{fail}} \rangle$ be a WFS, $T$ a test suite for $\mathcal{A}_{\mathrm{spec}}$, and $E$ a correct execution of $T$. Then, the* risk *of an arbitrary implementation of $W$ after executing $T$ yielded $E$ is defined by*

$$\mathrm{risk}_W(T, E) = \mathbb{E}[w(A_W) \mid R_{W,T} = E] \ .$$

For this conditional expectation to be defined properly, $\mathbb{P}[R_{W,T} = E]$ has to be nonzero. However, since $E$ is exactly the execution we observed, $\mathbb{P}[R_{W,T} = E]$ is obviously nonzero and consequently no extra restriction is imposed.

Due to nondeterminism, the absence of a failure during testing does not yet prove its absence. Therefore, to compute the risk we need the probability that a trace has been implemented incorrectly even though a test suite passes; its *posterior error probability*.

**Definition 11** (PEP)**.** *Let $W = \langle \mathcal{A}_{\mathrm{spec}}, w, p_{\mathrm{err}}, p_{\mathrm{fail}} \rangle$ be a WFS, $T$ a test suite for $\mathcal{A}_{\mathrm{spec}}$, $E$ a correct execution of $T$, and $\sigma \in \mathrm{traces}_{\mathcal{A}_{\mathrm{spec}}}$. Then, the* posterior error probability (PEP) *of $\sigma$, after $T$ yielded $E$, is defined by*

$$\mathrm{PEP}_W(\sigma, T, E) = \mathbb{P}[A_W[\sigma] \not\subseteq \mathcal{A}_{\mathrm{spec}}[\sigma] \mid R_{W,T} = E] \ .$$

Some executions of $E$ may have performed an input action after $\sigma$, and are therefore not able to detect incorrect behaviour directly after $\sigma$. Hence, to compute the PEP we have to count the executions reaching $\sigma$ *and* observing afterwards; this is precisely given by $\mathrm{obs}(\sigma, E)$. Keeping this in mind, the following proposition can be obtained using Bayes' formula. A proof of this, and of all other propositions and theorems of this paper, can be found in [18].

**Proposition 1.** *Let $W = \langle \mathcal{A}_{\mathrm{spec}}, w, p_{\mathrm{err}}, p_{\mathrm{fail}} \rangle$ be a WFS, $T$ a test suite for $\mathcal{A}_{\mathrm{spec}}$, $E$ a correct execution of $T$, and $\sigma \in \mathrm{traces}_{\mathcal{A}_{\mathrm{spec}}}$. Then*

$$\mathrm{PEP}_W(\sigma, T, E) = \frac{(1 - p_{\mathrm{fail}}(\sigma))^{\mathrm{obs}(\sigma,E)} \cdot p_{\mathrm{err}}(\sigma)}{(1 - p_{\mathrm{fail}}(\sigma))^{\mathrm{obs}(\sigma,E)} \cdot p_{\mathrm{err}}(\sigma) + 1 - p_{\mathrm{err}}(\sigma)} \ .$$

Since $\mathrm{PEP}_W(\sigma, T, E)$ only depends on $W$, $\sigma$ and $\mathrm{obs}(\sigma, E)$, we use $\mathrm{PEP}_W(\sigma, n)$ to denote $\mathrm{PEP}_W(\sigma, T, E)$ with $\mathrm{obs}(\sigma, E) = n$.

It is not difficult to see that the value of $\mathrm{risk}_W(T, E)$ in principle can be computed by ranging over all traces of

$\mathcal{A}_{\text{spec}}$, summing their fault weight multiplied by their PEP:

$$\text{risk}_W(T,E) = \sum_{\sigma \in \text{traces}_{\mathcal{A}_{\text{spec}}}} w(\sigma) \cdot \text{PEP}_W(\sigma, T, E) \ .$$

However, as there are infinitely many traces, this formula cannot be evaluated in practice (unless truncation was used to specify $w$). To solve this, we first compute the initial risk:

$$\text{risk}_W(\langle\rangle, \langle\rangle) = \sum_{\sigma \in \text{traces}_{\mathcal{A}_{\text{spec}}}} w(\sigma) \cdot p_{\text{err}}(\sigma) \ .$$

In case of discounting this formula can easily be evaluated using a system of linear equations, very similar to how [10] computes the total coverage $\sum_{\sigma \in L^*} w(\sigma)$. Now, for all traces $\sigma \in \text{obs}_T$ we subtract the risk reduction that was obtained by $E$. Considering that the initial risk of every trace $\sigma$ is $w(\sigma) \cdot p_{\text{err}}(\sigma)$, the following theorem easily follows.

**Theorem 1.** *Let $W = \langle \mathcal{A}_{\text{spec}}, w, p_{\text{err}}, p_{\text{fail}} \rangle$ be a WFS, $T$ a test suite for $\mathcal{A}_{\text{spec}}$, and $E$ a correct execution of $T$. Then*

$$\text{risk}_W(T,E) = \text{risk}_W(\langle\rangle, \langle\rangle) - \sum_{\sigma \in \text{obs}_T} w(\sigma) \cdot (p_{\text{err}}(\sigma) - \text{PEP}_W(\sigma, T, E)) \ .$$

*Complexity.* The complexity of risk evaluation is in $O(n^3 + p \log(m))$, with $n$ the number of states of $\mathcal{A}_{\text{spec}}$, $m$ the size of $T$, and $p$ the size of $\text{obs}_T$.

The term $n^3$ comes from calculating $\text{risk}_W(\langle\rangle, \langle\rangle)$, which is shown to be of this complexity in [10]. Then, the summation yields $p$ summands, each of them requiring some exponentiations (worst case in $O(\log(m))$).

*Example* 4. Again consider the WFS of the previous examples. Assume that discounting was defined such that $\text{risk}_W(\langle\rangle, \langle\rangle) = 10$.

Let $T$ be the test suite $\langle t, t \rangle$, with $t$ the test case of Figure 1(b). Suppose that $T$ is executed, yielding $E = \langle (\text{20ct? tea!}), (\text{20ct? coffee! } \delta) \rangle$. To determine $\text{risk}_W(T,E)$, we calculate the risk reduction obtained by $E$, and subtract this from the initial risk (following Theorem 1). As risk can only be reduced by observing traces, we only have to consider the traces that are in the set $\text{obs}_T = \{(\text{20ct?}), (\text{20ct? coffee!})\}$.

Since $E$ observed twice after $\sigma_1 = (\text{20ct?})$, the risk reduction by this trace is

$$w(\sigma_1) \cdot (p_{\text{err}}(\sigma_1) - \text{PEP}_W(\sigma_1, 2)) = (25 \cdot 0.9) \left(0.05 - \frac{0.6^2 \cdot 0.05}{0.6^2 \cdot 0.05 + 1 - 0.05}\right) = 0.707.$$

Since $E$ observed once after $\sigma_2 = (\text{20ct? coffee!})$, the risk reduction by this trace is

$$w(\sigma_2) \cdot (p_{\text{err}}(\sigma_2) - \text{PEP}_W(\sigma_2, 1)) = (40 \cdot 0.9^2) \left(0.01 - \frac{0.6^1 \cdot 0.01}{0.6^1 \cdot 0.01 + 1 - 0.01}\right) = 0.129.$$

Thus, we obtain

$$\text{risk}_W(T,E) = 10 - (0.707 + 0.129) = 9.164 \ . \quad \square$$

### *B. Estimating Output Behaviour to Predict Risk Reduction*

Due to nondeterministic behaviour, risk has been defined for IOLTSs based on a test suite *and* an execution. However, to find an optimal test suite, we need to estimate risk without knowing the execution in advance. It is therefore necessary to estimate output behaviour, for which we extend the WFS model to also include *output probabilities* $p_{\text{out}}$.

**Definition 12** (WFS$^+$)**.** *A WFS$^+$ is a tuple $W = \langle \mathcal{A}_{\text{spec}}, w, p_{\text{err}}, p_{\text{fail}}, p_{\text{out}} \rangle$, where the first four elements constitute a WFS, and $p_{\text{out}} \colon \text{traces}_{\mathcal{A}_{\text{spec}}} \times L_{\text{O}} \to [0,1]$ is a function assigning to each trace $\sigma$ and output action $a!$ the probability $p_{\text{out}}(\sigma, a!)$ that the system provides an $a!$ after $\sigma$, given that no failures occur.*

Given a test case $t$ and a trace $\sigma \in t$, the probability $p_{\text{reach}}(\sigma)$ of actually reaching $\sigma$ when $t$ is executed (given that no failures occur) can easily be calculated using $p_{\text{out}}$. As inputs are chosen by the test case, we set $p_{\text{out}}(\sigma, a?) = 1$ for all traces $\sigma$ and input actions $a? \in L_{\text{I}}$.

**Proposition 2.** *Let $W = \langle \mathcal{A}_{\text{spec}}, w, p_{\text{err}}, p_{\text{fail}}, p_{\text{out}} \rangle$ be a WFS$^+$ and $t$ a test case for $\mathcal{A}_{\text{spec}}$, then $p_{\text{reach}}(\sigma) = \prod_{i=1}^{n} p_{\text{out}}(a_1 \dots a_{i-1}, a_i)$ for all $\sigma = a_1 a_2 \dots a_n \in t$.*

Now, letting $\text{risk}_W(T)$ be the random variable representing the expected risk after a random execution of $T$, the following result holds.

**Theorem 2.** *Let $W = \langle \mathcal{A}_{\text{spec}}, w, p_{\text{err}}, p_{\text{fail}}, p_{\text{out}} \rangle$ be a WFS$^+$ and $T$ a test suite for $\mathcal{A}_{\text{spec}}$. Then*

$$\mathbb{E}[\text{risk}_W(T)] = \text{risk}_W(\langle\rangle, \langle\rangle) - \sum_{\sigma \in \text{obs}_T} w(\sigma) \cdot \left( \sum_{i=0}^{\text{obs}(\sigma,T)} \binom{\text{obs}(\sigma,T)}{i} \cdot p_{\text{reach}}(\sigma)^i \cdot (1 - p_{\text{reach}}(\sigma))^{\text{obs}(\sigma,T)-i} \cdot (p_{\text{err}}(\sigma) - \text{PEP}_W(\sigma, i)) \right) \ .$$

*Proof (sketch).:* To obtain this formula, we started with Theorem 1 and replaced $p_{\text{err}}(\sigma) - \text{PEP}_W(\sigma, T, E)$ (the error probability reduction for $E$) by the expected error probability reduction for an arbitrary execution. This reduction depends on the number of times the execution observes after $\sigma$, which is by definition between 0 and $\text{obs}(\sigma, T)$. The probability of observing $i$ times is equal to obtaining $i$ successes in a binomially distributed experiment with $n = \text{obs}(\sigma, T)$ and $p = p_{\text{reach}}(\sigma)$. Using these observations, Theorem 1, and the familiar formula for the binomial distribution, we obtain Theorem 2. ■

*Complexity.* The complexity of risk prediction is in $O(n^3 +$

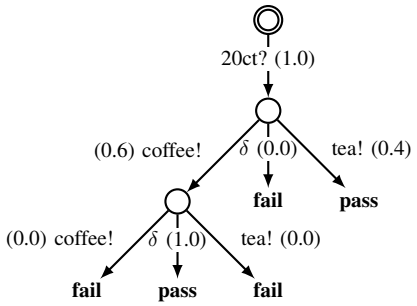



Figure 2. A test case with output probabilities

$pm\log(m) + p^3)$, with $n$ the number of states of $\mathcal{A}_{\text{spec}}$, $m$ the size of $T$, and $p$ the size of $\text{obs}_T$.

Again, the term $n^3$ comes from calculating $\text{risk}_W(\langle\rangle, \langle\rangle)$. Then, the outer summation yields $p$ summands. For each of these the inner summation yields at most $m$ summands, each of them requiring some exponentiations (worst case in $O(\log(m))$). Finally, the binomials $\binom{s}{0}, \ldots, \binom{s}{s}$ are required for all $s \in \{1, \ldots, p\}$. Since each of them requires at most $2s$ multiplications, this is in $O(p^3)$.

*Example* 5. Using the WFS $W$ and test suite $T$ of the previous examples again, we compute $\mathbb{E}[\text{risk}_W(T)]$. First, we specify the relevant output probabilities to make it into a WFS$^+$; see Figure 2. Since we assume that test suite executions pass (otherwise we improve the system and test again), the incorrect outputs have been given probability 0.

Now, we can easily calculate $p_{\text{reach}}(\text{20ct?}) = 1.0$, and $p_{\text{reach}}(\text{20ct? coffee!}) = 1.0 \cdot 0.6 = 0.6$. Using Theorem 2, we obtain the following (the full calculation is in [18]).

$$\mathbb{E}[\text{risk}_W(T)] = \cdots = 10 - 0.707 - 0.136 = 9.157 \ .$$

The risk that was calculated in Example 4 is indeed almost equal to this expected value. □

### C. Test Optimisation with Respect to Risk

Using risk prediction we can now compute optimal test suites with respect to risk. Let $W = \langle \mathcal{A}, w, p_{\text{err}}, p_{\text{fail}}, p_{\text{out}} \rangle$ be a WFS$^+$, $T$ a test suite for $\mathcal{A}$, $\sigma$ a trace of $\mathcal{A}$, and assume that $\text{obs}(\sigma, T) = n$. Since each trace $\sigma$ contributes $w(\sigma) \cdot p_{\text{err}}(\sigma)$ to $\text{risk}_W(\langle\rangle, \langle\rangle)$, Theorem 2 implies that the contribution of $\sigma$ to the expected risk after $T$ passes is

$$c(\sigma, n) = w(\sigma) \cdot \sum_{i=0}^{n} \binom{n}{i} p_{\text{reach}}(\sigma)^i (1 - p_{\text{reach}}(\sigma))^{n-i} \cdot \text{PEP}_W(\sigma, i) \ .$$

Note that $c(\sigma, n)$ only takes into account the contribution of $\sigma$ itself, not of its prefixes. It is easy to see that adding a new test case to $T$ that observes after $\sigma$ yields an expected risk reduction (ERR) of $r(\sigma, n) = c(\sigma, n) - c(\sigma, n+1)$.

Using these insights, we can now construct

$$T^{\text{opt-risk}}_{W,k,d} = \underset{T=\langle t_1, \ldots, t_k \rangle}{\arg\min} \ \mathbb{E}[\text{risk}_W(T)] \ ,$$

i.e., a test suite of size $k$ with minimal expected risk. As an extra restriction, we limit the depth of each test case to $d$ to obtain a finite test suite.

To compute the best test case (i.e., the one having maximal ERR) of depth $d$ to add to a test suite $T$, we first derive a recursive equation for the maximal ERR obtained by such a test case. To express this as a function of the maximal ERR of its sub test cases of depth $d-1$, we also have to keep track of the trace seen thus far. We therefore let $\mathcal{M}_T(\sigma, d')$ denote the maximal ERR to be obtained by a sub test case of depth $d'$ with history $\sigma$. Note that we are looking for $\mathcal{M}_T(\epsilon, d)$.

For $d' = 0$, trivially $\mathcal{M}_T(\sigma, d') = 0$. For an arbitrary $d' > 0$, $\mathcal{M}_T(\sigma, d')$ is calculated inductively by looking at the first step of the test case. Starting with an input $a? \in L_{\text{I}}$ such that $\sigma a? \in \text{traces}_{\mathcal{A}}$, no ERR is obtained directly and we are left with $\mathcal{M}_T(\sigma a?, d'-1)$. Starting with observation, an ERR of $r(\sigma, \text{obs}(\sigma, T))$ is obtained. Then, some output $b!$ is provided, leaving us with $\mathcal{M}_T(\sigma b!, d'-1)$. As the probability of choosing each individual $b!$ is already accounted for in $c(\sigma, n)$, no weighted average is taken.

Formalising these observations, we obtain

$$\mathcal{M}_T(\sigma, d') = \begin{cases} 0 & \text{if } d' = 0 \\ \max(\text{doInput}, \text{observe}) & \text{if } d' > 0 \end{cases} \ ,$$

where

$$\text{doInput} = \max_{\substack{a? \in L_{\text{I}} \\ \sigma a? \in \text{traces}_{\mathcal{A}}}} \mathcal{M}_T(\sigma a?, d'-1) \ ,$$

$$\text{observe} = r(\sigma, \text{obs}(\sigma, T)) + \sum_{\substack{b! \in L_{\text{O}} \\ \sigma b! \in \text{traces}_{\mathcal{A}}}} \mathcal{M}_T(\sigma b!, d'-1) \ .$$

To construct $T^{\text{opt-risk}}_{W,k,d}$, we start with $T = \langle\rangle$ and compute $\mathcal{M}_T(\epsilon, d)$: the maximum expected risk reduction to be obtained by a test case of depth $d$. Algorithmically, we start bottom-up by calculating $\mathcal{M}_T(\sigma_{d-1}, 1)$ for all $\sigma_{d-1} \in \text{traces}_{\mathcal{A}}$ of length $d-1$. Based on these values, we can calculate $\mathcal{M}_T(\sigma_{d-2}, 2)$ for all $\sigma_{d-2} \in \text{traces}_{\mathcal{A}}$ of length $d-2$. Working our way up, we arrive at $\mathcal{M}_T(\epsilon, d)$. During the calculations we record for each $\mathcal{M}_T(\rho, l)$ whether it was obtained by observation or an input (and which one).

Now, the first step of the best test case $t_1$ is the action (or observation) that was chosen to maximise $\mathcal{M}_T(\epsilon, d)$. Then, suppose that $a \in L$ was performed, we do an input or observe, according to which was chosen for $\mathcal{M}_T(a, d-1)$, and so on, until depth $d$ has been reached.

After having set $T = \{t_1\}$, the best test case $t_2$ to add to $T$ can be found by repeating the procedure described above. Since only a part of the state space changes, this can be calculated efficiently. We just continue in this way until $|T| = k$ and then set $T^{\text{opt-risk}}_{W,k,d} = T$.

In [18] the algorithm is formalised and proved correct.

*Complexity.* Note that the above method is very similar to history-dependent backwards induction, known from Markov decision theory [20]. The complexity of finding each test case to add to $T$ is therefore exponential in its depth, and because of history-dependence cannot be improved to polynomial complexity.

*Example* 6. Using the WFS$^+$ $W$ of the previous examples once more, we calculate the optimal test suite $T^{\text{opt-risk}}_{W,2,3}$ of size 2 and depth 3. We still assume $p_{\text{reach}}(\text{20ct? coffee!}) = 0.6$ and $p_{\text{reach}}(\text{20ct? tea!}) = 0.4$, and a discount rate of $\lambda = 0.9$ for short traces.

We first calculate $\mathcal{M}_T(\sigma_2, 1)$ for all traces of length 2:

$$\begin{aligned}
\mathcal{M}_T(\delta\ \delta, 1) &= \max\big(\mathcal{M}_T(\delta\ \delta\ \text{10ct?}, 0), \mathcal{M}_T(\delta\ \delta\ \text{20ct?}, 0), \\
&\qquad r(\delta\ \delta, 0) + \mathcal{M}_T(\delta\ \delta\ \delta, 0)\big) \\
&= \max(0, 0, 0.129) = 0.129 \\
\mathcal{M}_T(\delta\ \text{10ct?}, 1) &= \cdots = 0.683 \\
\mathcal{M}_T(\delta\ \text{20ct?}, 1) &= \cdots = 0.393 \\
\mathcal{M}_T(\text{10ct? tea!}, 1) &= \cdots = 0.129 \\
\mathcal{M}_T(\text{20ct? tea!}, 1) &= \cdots = 0.052 \\
\mathcal{M}_T(\text{20ct? coffee!}, 1) &= \cdots = 0.077
\end{aligned}$$

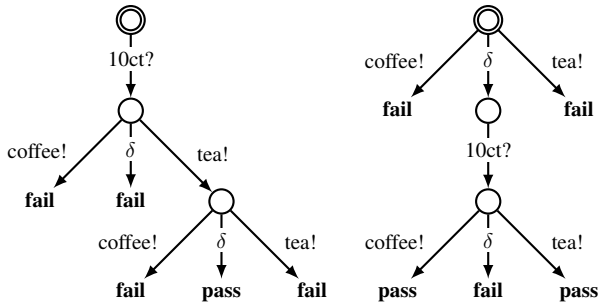
Note that this confirms the intuition that it is best to observe in the final step, since performing an input has no effect on the risk. Continuing, we calculate

$$\begin{aligned}
\mathcal{M}_T(\delta, 2) &= \max\big(\mathcal{M}_T(\delta\ \text{10ct?}, 1), \mathcal{M}_T(\delta\ \text{20ct?}, 1), \\
&\qquad r(\delta, 0) + \mathcal{M}_T(\delta\ \delta, 1)\big) \\
&= \max(0.683, 0.393, 0.143 + 0.129) = 0.683 \\
\mathcal{M}_T(\text{10ct?}, 2) &= \max\big(r(\text{10ct?}, 0) + \mathcal{M}_T(\text{10ct? tea!}, 1)\big) \\
&= \max(0.759 + 0.129) = 0.889 \\
\mathcal{M}_T(\text{20ct?}, 2) &= \max\big(r(\text{20ct?}, 0) + \mathcal{M}_T(\text{20ct? tea!}, 1) \\
&\qquad + \mathcal{M}_T(\text{20ct? coffee!}, 1)\big) \\
&= \max(0.436 + 0.052 + 0.077) = 0.565 \\
\\
\mathcal{M}_T(\epsilon, 3) &= \max\big(\mathcal{M}_T(\text{10ct?}, 2), \mathcal{M}_T(\text{20ct?}, 2), \\
&\qquad r(\epsilon, 0) + \mathcal{M}_T(\delta, 2)\big) \\
&= \max(0.889, 0.565, 0.159 + 0.683) = 0.889
\end{aligned}$$

Apparently, the maximum expected risk reduction that can be obtained by a test case of depth 3 is 0.889. Based on the calculations above, we can deduce the corresponding test case, which is depicted in Figure 3(a).

To find the second test case to include in $T$, we perform the same calculations, only now using $r(\text{10ct?}, 1)$ and $r(\text{10ct? tea!}, 1)$ instead of $r(\text{10ct?}, 0)$ and $r(\text{10ct? tea!}, 0)$, since $\text{obs}(\sigma, T)$ is now 1 for them. In fact, only a part of the calculations has to be repeated.

We find the test case shown in Figure 3(b), which yields an *additional* expected risk reduction of 0.842. As this calculation uses the fact that the first test case was already present, the values can just be added to find that the optimal test suite of size two has an expected risk reduction of $0.889 + 0.842 = 1.731$. □




(a) The first test case (b) The second test case

Figure 3. An optimal test suite

## IV. Actual Coverage

Whereas risk gives us information about the entire system, coverage only relates to the part of a system we tested.

Basically, we define the *absolute actual coverage* (absCov) of a test suite $T$ as the accumulated fault weight of the traces that are known to be implemented correctly after $T$ passes. However, due to nondeterminism we often only reduce the probability of the presence of faults, so we need a more precise notion. We therefore introduce *relative error probability reduction* (REPR) as the extent to which the error probability of a trace decreases as a result of passing a test suite. Then, absCov is defined as the sum of all fault weights, each weighted by the corresponding REPR.

To assess the quality of a test suite, we calculate its absolute actual coverage relative to the total amount of fault weight that could potentially be present in the system. This measure will be called its *relative actual coverage* (relCov). (See [10] for an algorithm to compute totCov efficiently).

**Definition 13** (Coverage measures)**.** *Let $W = \langle \mathcal{A}_{\text{spec}}, w, p_{\text{err}}, p_{\text{fail}} \rangle$ be a WFS, $T$ a test suite for $\mathcal{A}_{\text{spec}}$, and $E$ a correct execution of $T$. Then we define*

$$\begin{aligned}
\text{REPR}_W(\sigma, T, E) &= \frac{p_{\text{err}}(\sigma) - \text{PEP}_W(\sigma, T, E)}{p_{\text{err}}(\sigma)}\ ; \\
\text{absCov}_W(T, E) &= \sum_{\sigma \in \text{traces}_{\mathcal{A}_{\text{spec}}}} w(\sigma) \cdot \text{REPR}_W(\sigma, T, E)\ ; \\
\text{totCov}_W &= \sum_{\sigma \in \text{traces}_{\mathcal{A}_{\text{spec}}}} w(\sigma)\ ; \\
\text{relCov}_W(T) &= \frac{\text{absCov}_W(T)}{\text{totCov}_W}\ .
\end{aligned}$$

Note that since weight functions never sum up to zero or infinity, relative actual coverage is properly defined.

Using Proposition 1, the following result can easily be obtained, providing a formula for computing actual coverage. Note that it reduced to a finite sum, since the traces not in $\text{obs}_T$ have no REPR and can therefore be omitted.

**Theorem 3.** *Let $W = \langle \mathcal{A}_{\text{spec}}, w, p_{\text{err}}, p_{\text{fail}} \rangle$ be a WFS, $T$ a test suite for $\mathcal{A}_{\text{spec}}$, and $E$ a correct execution of $T$. Then*

$$\text{absCov}_W(T, E) = \sum_{\sigma \in \text{obs}_T} w(\sigma) \cdot \left(1 - \frac{(1 - p_{\text{fail}}(\sigma))^{\text{obs}(\sigma, E)}}{1 - p_{\text{err}}(\sigma) + (1 - p_{\text{fail}}(\sigma))^{\text{obs}(\sigma, E)} \cdot p_{\text{err}}(\sigma)}\right).$$

Optimisation with respect to actual coverage can be done in exactly the same way as optimisation with respect to risk, only using expected coverage increase instead of expected risk reduction.

## V. Conclusions and Future Work

While testing is an important part of today's software development process, little research has been devoted to the interpretation of a successful testing process. In this paper, we introduced a weighted fault specification (WFS) to describe the required behaviour of a system and the estimation of its probabilistic behaviour. Based on such a WFS, we presented two measures: *risk* and *actual coverage*. Risk denotes the confidence in the system after testing is successful, whereas actual coverage denotes how much was tested.

We presented a method to compute the risk of a system after it successfully passes a test suite, as well as a way to calculate the quality of a given test suite with respect to risk. We also gave an optimisation strategy enabling the tester to obtain a test suite of a given size that will obtain minimal risk. All are easily adaptable to work with actual coverage. Although we made some strict assumptions on error independence, we think that a thorough understanding of simple models is a useful start when tackling these complicated problems.

Our work gives rise to several directions for future research. First, it is crucial to validate our framework by developing tool support and performing case studies. Second, it seems useful to include fault dependencies in our model. Third, the possibilities of on-the-fly test derivations (e.g., as performed by the tool TorX [21]) based on risk or actual coverage could be investigated. This may yield a tool that, during testing, calculates probabilities and decides how to test optimally. Finally, our framework may be used to study the sensitivity of the probabilities, validating potential simplifying approximations for risk and actual coverage.

## References


[1] M. Newman, "Software errors cost U.S. economy 59.5 billion annually, NIST assesses technical needs of industry to improve software-testing," Press Release, http://www.nist.gov/public_affairs/releases/n02-10.htm, 2002.

[2] T. Ball, "A Theory of Predicate-Complete Test Coverage and Generation," in *Proceedings of the 3rd International Symposium on Formal Methods for Components and Objects (FMCO '04)*, ser. Lecture Notes in Computer Science, vol. 3657. Springer, 2004, pp. 1–22.

[3] G. J. Myers, C. Sandler, T. Badgett, and T. M. Thomas, *The Art of Software Testing, Second Edition*. Wiley, 2004.

[4] D. Lee and M. Yannakakis, "Principles and methods of testing finite state machines - a survey," *Proceedings of the IEEE*, vol. 84, no. 8, pp. 1090–1123, 1996.

[5] L. Nachmanson, M. Veanes, W. Schulte, N. Tillmann, and W. Grieskamp, "Optimal strategies for testing nondeterministic systems," *SIGSOFT Software Engineering Notes*, vol. 29, no. 4, pp. 55–64, 2004.

[6] H. Ural, "Formal methods for test sequence generation," *Computer Communications*, vol. 15, no. 5, pp. 311–325, 1992.

[7] F. Redmill, "Exploring risk-based testing and its implications," *Software Testing, Verification and Reliability*, vol. 14, no. 1, pp. 3–15, 2004.

[8] J. Bach, "Heuristic risk-based testing," *Software Testing and Quality Engineering Magazine*, November/December 1999.

[9] S. Amland, "Risk-based testing: risk analysis fundamentals and metrics for software testing including a financial application case study," *Journal of Systems and Software*, vol. 53, no. 3, pp. 287–295, 2000.

[10] L. Brandán Briones, E. Brinksma, and M. I. A. Stoelinga, "A semantic framework for test coverage," in *Proceedings of the 4th International Symposium on Automated Technology for Verification and Analysis (ATVA '06)*, ser. Lecture Notes in Computer Science, vol. 4218. Springer, 2006, pp. 399–414.

[11] T. L. Graves, A. F. Karr, J. S. Marron, and H. Siy, "Predicting fault incidence using software change history," *IEEE Transactions on Software Engineering*, vol. 26, no. 7, pp. 653–661, 2000.

[12] T. J. McCabe, "A complexity measure," *IEEE Transactions on Software Engineering*, vol. 2, no. 4, pp. 308–320, 1976.

[13] M. H. Halstead, *Elements of Software Science*. Elsevier, 1977.

[14] Y. K. Malaiya and J. Denton, "Requirements volatility and defect density," in *Proceedings of the 10th International Symposium on Software Reliability Engineering (ISSRE '99)*. IEEE, 1999, pp. 285–294.

[15] K. W. Miller, L. J. Morell, R. E. Noonan, S. K. Park, D. M. Nicol, B. W. Murrill, and M. Voas, "Estimating the probability of failure when testing reveals no failures," *IEEE Transactions on Software Engineering*, vol. 18, no. 1, pp. 33–43, 1992.

[16] J. Voas, L. Morell, and K. Miller, "Predicting where faults can hide from testing," *IEEE Software*, vol. 8, no. 2, pp. 41–48, 1991.

[17] A. Saltelli, S. Tarantola, F. Campolongo, and M. Ratto, *Sensitivity Analysis in Practice: A Guide to Assessing Scientific Models*. Halsted, 2004.

[18] M. I. A. Stoelinga and M. Timmer, "Interpreting a successful testing process: risk and actual coverage (extended version)," CTIT, University of Twente, Tech. Rep. TR-CTIT-09-17, 2009.

[19] G. J. Tretmans, "Test Generation with Inputs, Outputs and Repetitive Quiescence," *Software—Concepts and Tools*, vol. 17, no. 3, pp. 103–120, 1996.

[20] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. Wiley, 2005.

[21] A. Belinfante, J. Feenstra, R. G. de Vries, J. Tretmans, N. Goga, L. M. G. Feijs, S. Mauw, and L. Heerink, "Formal test automation: A simple experiment," in *Proceedings of the 12th International Workshop on Testing Communicating Systems (IWTCS '99)*, ser. IFIP Conference Proceedings, vol. 147. Kluwer, 1999, pp. 179–196.