**COVERAGE-BASED VS. USAGE-BASED TESTING: WHEN TO STOP**

**TESTING?**

For most of the testing situations, the answer to the question “when to stop testing?’ depends on the completion of some pre-planned activities, coverage of certain entities, or whether a pre-set goal has been achieved. We next describe the use of different exit criteria and the corresponding testing techniques.

**When to stop testing?**

The question, “when to stop testing”, can be refined into two different questions:

* On a small or a local scale, we can ask: “When to stop testing for a specific test activity?’ This question is also commonly associated with different testing subphases.
* On a global scale, we can ask: “When to stop all the major test activities?” Because the testing phase is usually the last major development phase before product release, this question is equivalent to: “When to stop testing and release the product?’ These questions may yield different answers, leading us to different testing techniques and related activities. Without a formal assessment for decision making, decision to stop testing can usually be made in two general forms:
* *Resource-based criteria,* where decision is made based on resource consumptions. The most commonly used such stopping criteria are

- “Stop when you run out of time.”

- “Stop when you run out of money.”

Such criteria are irresponsible, as far as product quality is concerned, although they

may be employed if product schedule or cost are the dominant concerns for the

product in question.

* *Activity-based criteria,* commonly in the form:

- “Stop when you complete planned test activities.”

This criterion implicitly assumes the effectiveness of the test activities in ensuring the quality of the software product. However, this assumption could be questionable without strong historical evidence based on actual data from the project concerned. Because of these shortcomings, informal decisions without using formal assessments have very limited use in managing the testing process and activities for large software systems. We next examine exit criteria based on formal analyses and assessments. On the global level, the exit from testing is associated with product release, which determined the level of quality that a customer or a user could expect. In our overall software quality engineering process, this decision is associated with achieving quality goals, as well as achieving other project goals in the overall software development process. Therefore, the most direct and obvious way to make such product release decisions is the use of various reliability assessments. When the assessment environment is similar to the actual usage environment for customers. the resulting reliability measurement would be directly meaningful to these customers.

The basic idea in using reliability criterion is to set a reliability goal in the quality planning activity during product planning and requirement analysis, and later on to compare the reliability assessment based on testing data to see if this pre-set goal has been reached. If so, the product can be released. Otherwise, testing needs to continue and product release needs to be deferred. Various models exist today to provide reliability assessments and improvement based on data from testing,

One important implication of using this criterion for stopping testing is that the reliability assessments should be close to what actual users would expect, which requires that the testing right before product release resembles actual usages by target customers. This requirement resulted in the so-called usage-based testing. On the other hand, because of the large number of customers and usage situations, exhaustive coverage of all the customer usage scenarios, sequences, and patterns is infeasible. Therefore, an unbiased statistical sampling is the best that we can hope for, which results in usage-based statistical testing (UBST) that we will describe later in this section. Some specific techniques for such testing,

For earlier sub-phases of testing, or for stopping criteria related to localized test activities, reliability definition based on customer usage scenarios and frequencies may not be meaningful. For example, many of the internal components are never directly used by actual users, and some components associated with low usage frequencies may be critical for various situations. Under these situations, the use of reliability criterion may not be meaningful or may lead to inadequate testing of some specific components. Alternative exit criteria are needed. For example, as a rule of thumb:

“Products should not be released unless every component has been tested,”

Criteria similar to this have been adopted in many organizations to test their products and related components. We call these criteria coverage criteria, which involve coverage of some specific entities, such as components, execution paths, statements, etc. The use of coverage criteria is associated with defining appropriate coverage for different testing techniques, linking what was tested with what was covered in some formal assessments.

One implicit assumption in using coverage as the stopping criterion is that everything covered is defect free with respect to this specific coverage aspect, because all defects discovered by the suite of test cases that achieved this coverage goal would have been fixed or removed before product release. This assumption is similar to the one above regarding the effectiveness of test activities, when we use the completion of planned test activities as the exit criterion. However, this assumption is more likely to be enforced because specific coverage is closely linked to specific test cases.

From the quality perspective, the coverage criteria are based on the assumption that higher levels of coverage mean higher quality, and a specific quality goal can be translated into a specific coverage goal. However, we must realize that although there is a general positive correlation between coverage and quality, the relationship between the two is not a simple one. Many other factors need to be considered before an accurate quality assessment can be made based on coverage. For example, different testing techniques and sub-phases may be effective in detecting and removing different types of defects, leading to multistage reliability growth and saturation patterns (Horgan and Mathur, 1995). Nevertheless, coverage information gives us an approximate quality estimate, and can be used as the exit criterion when actual reliability assessment is unavailable, such as in the early sub-phases of testing.

**Usage-based statistical testing (UBST) and operational profiles (OPs)**

At one extreme, actual customer usage of software products can be viewed as a form of usage-based testing. When problems are experienced by these customers, some information about the problems, can be reported to software vendors, and integrated fixes can be constructed and delivered to all the customers to prevent similar problems from occurring. However, these post-product-release defect fixing activities could be very expensive because of the massive numbers of software installations. Frequent fixes could also damage the software vendor’s reputation and long-term business viability. The so-called beta test makes use of this usage-and-fix to the advantage of software vendors, through controlled software release so that these beta customers help software development organizations improve their software quality.

In general, if the actual usage, or anticipated usage for a new product, can be captured and used in testing, product reliability could be most directly assured. In usage-based statistical testing (UBST), the overall testing environment resembles the actual operational environment for the software product in the field, and the overall testing sequence, as represented by the orderly execution of specific test cases in a test suite, resembles the usage scenarios, sequences, and patterns of actual software usage by the target customers. Because the massive number of customers and diverse usage patterns cannot be captured in an exhaustive set of test cases, statistical sampling is needed, thus the term “statistical” in the descriptive title of this strategy. For the same reason, “random testing” and “usage-based random testing” are also used in literature. However, we prefer to use the term “usage-based statistical testing” in this book to avoid the confusion between random testing and “ad hoc” testing, where no systematic strategy is implied in “ad hoc” testing.

For practical implementation of such a testing strategy, actual usage information needs to be captured in various models, commonly referred to as “operational profiles” or OPs. Different OPs are associated with different testing techniques for UBST. Two primary types of usage models or OPs are:

* Flat OPs, or Musa OPs (Musa, 1993; Musa, 1998), which present commonly used operations in a list, a histogram, or a tree-structure, together with the associated occurrence probabilities. The main advantage of the flat OP is its simplicity, both in model construction and usage**.**
* Markov chain based usage models, or Markov OPs (Mills, 1972; Mills et al., 1987b; Whittaker and Thomason, 1994; Kallepalli and Tian, 2001; Tian et al., 2003), which present commonly used operational units in Markov chains, where the state transition probabilities are history independent (Karlin and Taylor, 1975). Complete operations can be constructed by linking various states together following the state transitions, and the probability for the whole path is the product of its individual transition probabilities. Markov models based on state transitions can generally capture navigation patterns better than flat OPs, but are more expensive to maintain and to use. This testing technique.

Usage-based statistical testing (UBST) is generally applicable to the final stage of testing, typically referred to as acceptance testing right before product release, so that stopping testing is equivalent to releasing the product. Other late sub-phases of testing, such as integration and system testing, could also benefit from the knowledge of actual customer usage situations to drive effective reliability improvement before product release. Naturally, the termination criterion used to stop such testing is achievement of reliability goals.

**Coverage and coverage-based testing (CBT)**

Most traditional testing techniques, either functional testing (BBT) or structural testing (WBT), use various forms of test coverage as the stopping criteria. The simplest such criterion is in the form of completing various checklists, such as a checklist of major functions based on product specifications when BBT is used, or a checklist of all the product components or all the statements when WBT is used. Testing can be performed until all the items on the respective checklist have been checked off or exhausted. For most of the systematic testing techniques, some formal models beyond simple checklists are used. Some specific examples of such models and related coverage include:

Formally defined partitions can be used as the basis for various testing techniques**,** which are similar to checklists but ensure:

- mutual exclusion of checklist items to avoid unnecessary repetition,

- complete coverage defined accordingly.

* **A** specialized type of partitions, input domain partitions into sub-domains, can also be used to test these sub-domains and related boundary conditions.
* Various programming or functional states can be defined and linked together to form finite-state machines (FSMs) to model the system as the basis for various testing techniques to ensure state coverage and coverage of related state transitions and execution sequences.

The above FSMs can also be extended to analyze and cover execution paths and data dependencies through various testing techniques.

The generic steps and major sub-activities for CBT model construction and test preparation are described below:

* *Defining the model:* These models are often represented by some graphs, with individual nodes representing the basic model elements and links representing the interconnections. Some additional information may be attached to the graph as link or node properties (commonly referred to as weights in graph theory).
* *Checking individual model elements* to make sure the individual elements, such as links, nodes, and related properties, have been tested individually, typically in isolation, prior to testing using the whole model. This step also represents the self-checking of the model, to make sure that the model captures what is to be tested.
* *Dejning coverage criteria:* Besides covering the basic model elements above, some other coverage criteria are typically used to cover the overall execution and interactions. For example, with the partition-based testing, we might want to cover the boundaries in addition to individual partitions; and for FSM-based testing, we might want to cover state transition sequences and execution paths.
* *Derive test cases:* Once the coverage criteria are defined, we can design our test cases to achieve them. The test cases need to be sensitized, that is, with its input values selected to realize specific tests, anticipated results defined, and ways to check the outcomes planned ahead of time.

Model construction and test preparation are more closely linked to individual testing techniques, which are described when each testing technique. The other major testing related activities, including test execution, measurement, analysis, and follow-up activities, are typically similar for all testing techniques. Coverage analysis plays an important role in guiding testing and coverage criterion is used to determine when to stop testing. Automated tool support by for this analysis and related data collection.

**Comparing CBT with UBST**

To summarize, the key questions that distinguish coverage-based testing (CBT) from usage based statistical testing (UBST) are the “perspective” question and the related stopping criteria:

* *Perspective:* UBST views the objects of testing from a user’s perspective and focuses on the usage scenarios, sequences, patterns, and associated frequencies or probabilities; while CBT views the objects from a developer’s perspective and focuses covering functional or implementation units and related entities.
* *Stopping criteria:* UBST use product reliability goals as the exit criterion; and CBT using coverage goals - surrogates or approximations of reliability goals - as the exit criterion. CBT and UBST can also be compared by the way in which they address the following questions:
* *Objects:* Although the objects tested may overlap, CBT is generally used to test and cover small objects, such as small software products, small units of large software products, or large systems at a high level of abstraction, such major functions or components; while UBST is generally more suitable for large software systems as a whole.
* *Verification vs. validation:* Although both CBT and UBST can be used for both verification test and validation test, UBST is more likely to be used for validation test because of their relationship to customers and users.
* *Timeline:* For large software systems, CBT is often used in early sub-phases of testing, such as unit and component testing, while UBST is often used in late sub-phases of testing, such as system and acceptance testing.
* *Defect detection:* In UBST, failures that are more likely to be experienced by users are also more likely to be observed in testing, leading to corresponding faults being detected and removed for reliability improvement. In CBT, failures are more closely related to things tested, which may lead to effective fault removal but may not be directly linked to improved reliability due to different exposure ratios for software faults.
* *Testing environment:* UBST uses testing environment similar to that for in-field operation at customer installations; while CBT uses environment specifically set up for testing.
* *Techniques:* Various techniques can be used to build models and generate test cases to perform systematic CBT. When these models are augmented with usage information, typically as the probabilities associated with checklist items, partitions, states, and state transitions, they can be used as models for UBST also. This is why we cover UBST models and techniques together with corresponding basic CBT models and techniques .
* *Customer and user roles:* UBST models are constructed with extensive customer and user input; while CBT models are usually constructed without active customer or user input. UBST is also more compatible with the customer and user focus in today’s competitive market.
* *Tester:* Dedicated professional testers typically perform UBST; while CBT can be performed by either professional testers or by developers themselves.