
Automating Results Comparison Isn't Always Easy

(Oracle-Based Automated Test Design)

C.E.S.A.R. Recife Summer School
February 18, 2009

**Douglas Hoffman, BACS, MBA, MSEE,
ASQ-CSQE, ASQ-CMQ/OE, ASQ Fellow
Software Quality Methods, LLC. (SQM)**

www.SoftwareQualityMethods.com
doug.hoffman@acm.org

Isn't Comparing Results Really Easy?

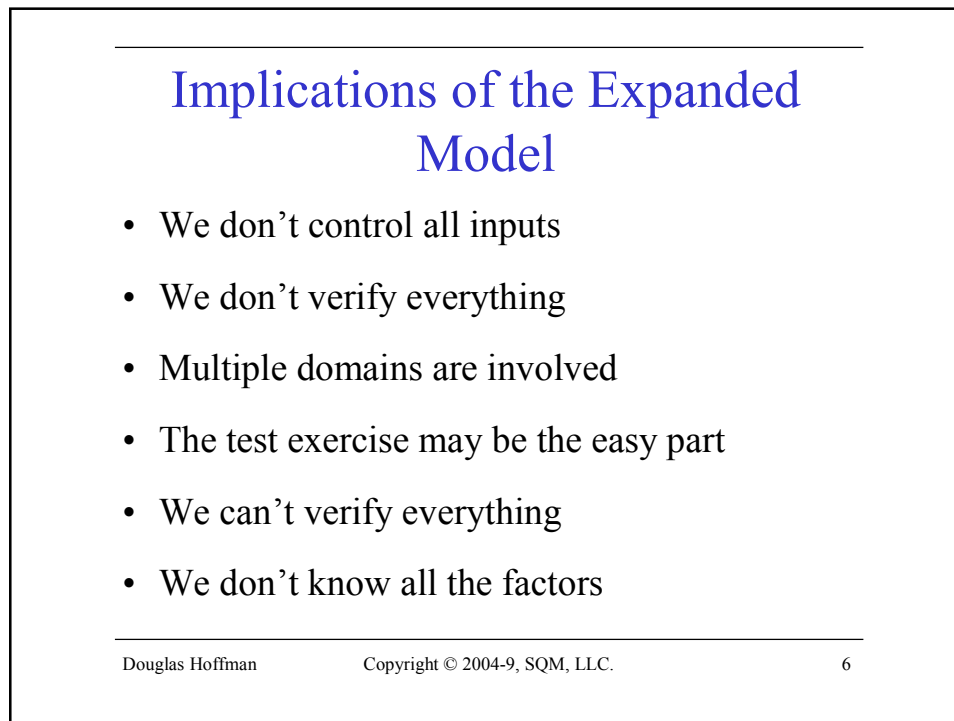
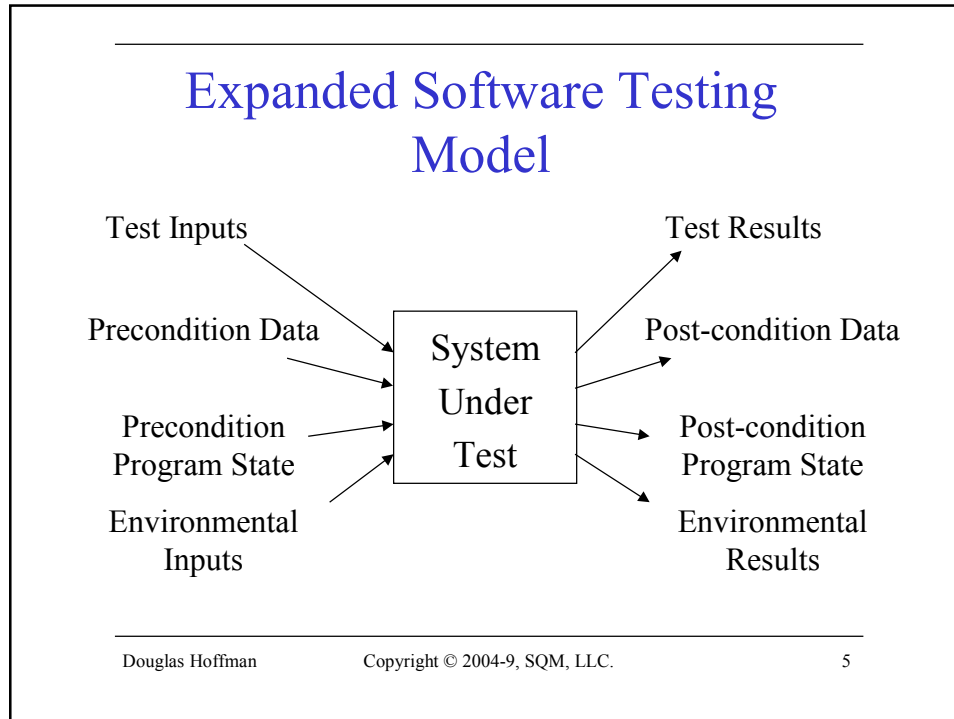
- Design the test
- Automate the test
- Run the test
- Save the result

- Run the test again
- Save the new result
- Compare the two saved results

What Did We Miss?

- What results are we talking about?
- Do we know how to capture them?
- Can we store the results?
- Do we know what they should be the first time? Did we check?
- How do we compare these results? (Are some differences OK)?

Software Test Automation: Foundational Concepts



Important Factors In Results Comparison

- Which results to check
- How do we know what to expect
- What differences matter
- Are “fuzzy” comparisons needed
- When to compare results

Software Test Automation: Test Oracles

Which Results To Check

- Expected results
- Anticipated likely errors
- Major environmental factors
- Available easy oracles

The Test Oracle

- Two slightly different views on the meaning of the word
 - *Reference Function*: You ask it what the “correct” answer is.
 - *Reference and Evaluation Function*: You ask it whether the program passed the test.
- Using an oracle, you can compare the program’s result to a reference value (predicted value) and decide whether the program passed the test.
 - *Deterministic oracle* (mismatch means program fails) (*This is the commonly analyzed case.*)
 - *Probabilistic oracle* (mismatch means program probably fails.)

A Source of Expected Results

- One common way to define an oracle is as a source of expected results
 - Under this view, you compare your results to those obtained from the oracle. If they match, the program passes. If they don't match, the program fails
 - The oracle is the method of generating the expected results
 - Comparison and evaluation may be handled separately
- *It is important to recognize that this evaluation is heuristic*
 - **We can have false alarms:** A mismatch between actual and oracle might not matter
 - **We can miss defects:** A match between actual and oracle might result from the same error in both

The Oracle Problem and Test Automation

- We often hear that all testing should be automated
- Automated testing depends on our ability to programmatically detect when the software under test behaves in expected or unexpected ways

Our ability to automate testing is fundamentally constrained by our ability to create and use oracles.

Automation Narrows the Oracle's Scope

An automated test is not equivalent to the most similar manual test:

- The mechanical comparison is typically more precise (and will be tripped by irrelevant discrepancies)
- The skilled human comparison will sample a wider range of dimensions, noting oddities that one wouldn't program the computer to detect
- The automated input is consistent, while humans cannot closely replicate their test activities

Oracles: Challenges

- Completeness of information
- Accuracy of information
- Usability of the oracle or of its results
- Maintainability of the oracle
- Complexity when compared to the SUT
- Temporal relationships
- Costs

Oracle Completeness

- Input Coverage
- Result Coverage
- Function Coverage
- Sufficiency
- Types of errors possible
- SUT environments

There may be more than one oracle for the SUT
Inputs may affect more than one oracle

Oracle Accuracy

- How similar to SUT
 - Arithmetic accuracy
 - Statistically similar
- How independent from SUT
 - Algorithms
 - Sub-programs & libraries
 - System platform
 - Operating environment

Close correspondence makes common mode faults more likely and reduces maintainability
- How extensive
 - The more ways in which the oracle matches the SUT, i.e. the more complex the oracle, the more errors
- Types of possible errors

Oracle Usability

- Form of information
 - Bits and bytes
 - Electronic signals
 - Hardcopy and display
- Location of information
- Data set size
- Fitness for intended use
- Availability of comparators
- Support in SUT environments

Douglas Hoffman

Copyright © 2004-9, SQM, LLC.

17

Oracle Maintainability

- COTS or custom
 - Custom oracle can become more complex than the SUT
 - More complex oracles make more errors
- Cost to keep correspondence through SUT changes
 - Test exercises
 - Test data
 - Tools
- Ancillary support activities required

Douglas Hoffman

Copyright © 2004-9, SQM, LLC.

18

Oracle Complexity

- Correspondence with SUT
- Coverage of SUT domains and functions
- Accuracy of generated results
- Maintenance cost to keep correspondence through SUT changes
 - Test exercises
 - Test data
 - Tools
- Ancillary support activities required

Douglas Hoffman

Copyright © 2004-9, SQM, LLC.

19

Oracle Temporal Relationships

- How fast to generate results
- How fast to compare
- When is the oracle run
- When are results generated
- When are results compared

Douglas Hoffman

Copyright © 2004-9, SQM, LLC.

20

Oracle Costs

- Creation or acquisition costs
- Maintenance of oracle and comparators
- Execution cost
- Cost of comparisons
- Additional analysis of errors
- Cost of misses
- Cost of false alarms

Software Test Automation: Types of Test Oracles

Oracle Taxonomy

	Definition	Advantages	Disadvantages
No Oracle	- Doesn't check correctness of results, (only that some results were produced)	- Can run any amount of data (limited only by the time the SUT takes)	- Only spectacular failures are noticed
Perfect Oracle	- Independent generation of all expected results	- No encountered errors go undetected - Can be used for many different tests	- Fast way to automate using an oracle - Expensive to implement and maintain - Complex and often time-consuming when run
Consistency	- Verifies current run results with a previous run	- Verification is straightforward - Can generate and verify large amounts of data	- Original run may include undetected errors
Self-Verifying	- Embeds answer within data in the messages	- Allows extensive post-test analysis - Does not require external oracles - Verification is based on message contents - Can generate and verify large amounts of complex data	- Must define answers and generate messages to contain them

Douglas Hoffman Copyright © 2004-9, SQM, LLC. 23

Oracle Taxonomy

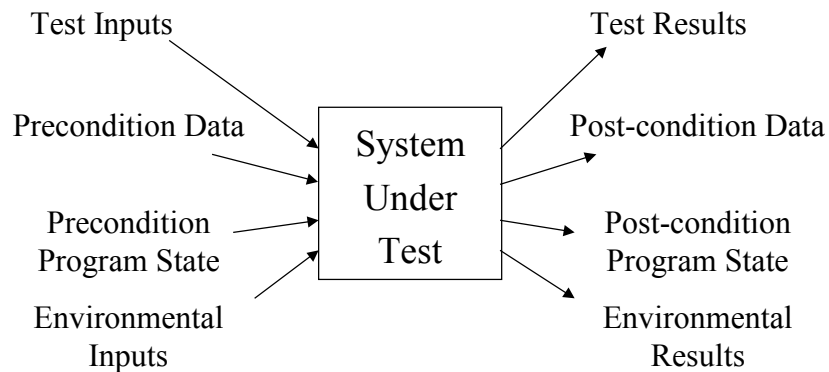
	Definition	Advantages	Disadvantages
Model Based	- Uses digital data model of SUT behavior	- May use digital model for multiple tests - Digital form of model easier to maintain than automated test - Tests may work for multiple SUTs by using different models	- Maintenance of complex SUT models is expensive - Model must match expected behavior
Hand Crafted	- Result is carefully selected by test designer	- Useful for some very complex SUTs - Expected result can be well understood	- Does the same thing every time - Limited number of cases can be generated
Heuristic	- Verifies some characteristics	- Faster and easier than Perfect Oracle - Much less expensive to create and use - May be reusable across SUTs and Tests	- Can miss systematic errors - Can miss obvious errors
Statistical	- Uses statistical correlation between inputs and outcomes	- Allows checking of very large data sets - Allows checking of live systems' data - Allows checking after the fact	- May miss systematic errors - Can miss obvious errors

Douglas Hoffman Copyright © 2004-9, SQM, LLC. 24

'No Oracle' Strategy

- Easy to implement
- Tests run fast
- Only spectacular errors are noticed
- False sense of accomplishment

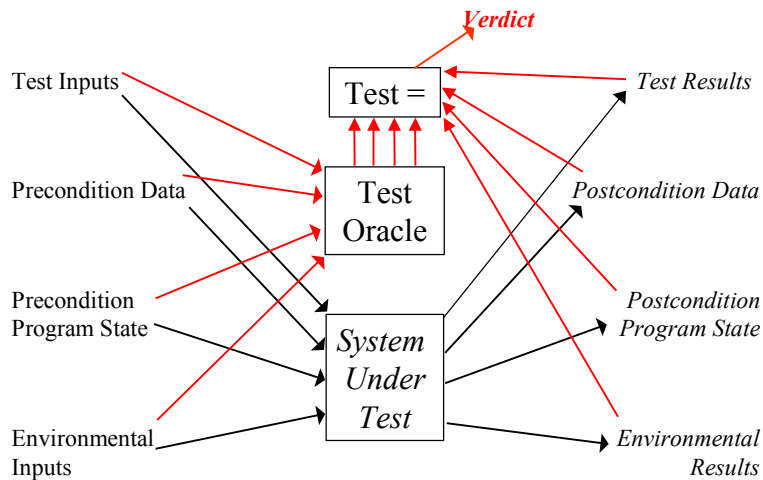
"No" Oracle Model



Perfect Oracle

- Independent implementation
- Complete coverage over domains
 - Input ranges
 - Result ranges
- “Correct” results
- Usually expensive

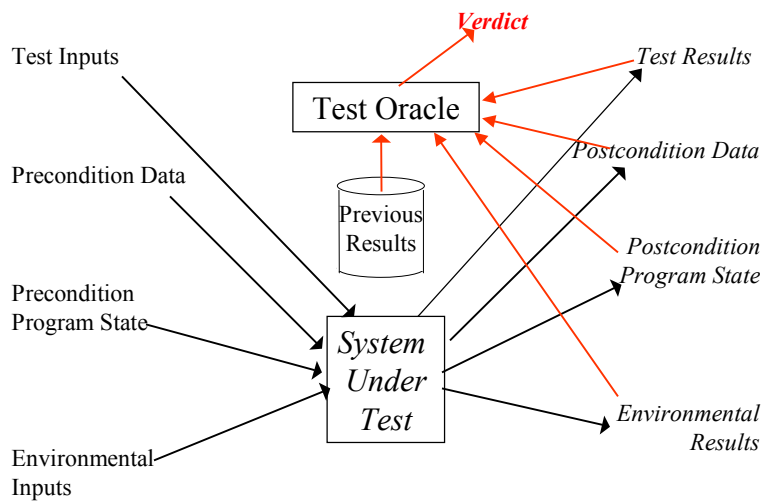
A Perfect Oracle



Consistency Strategy

- A / B compare
- Checking for changes
- Regression checking
 - Validated
 - Unvalidated
- Alternate versions or platforms
- Foreign implementations

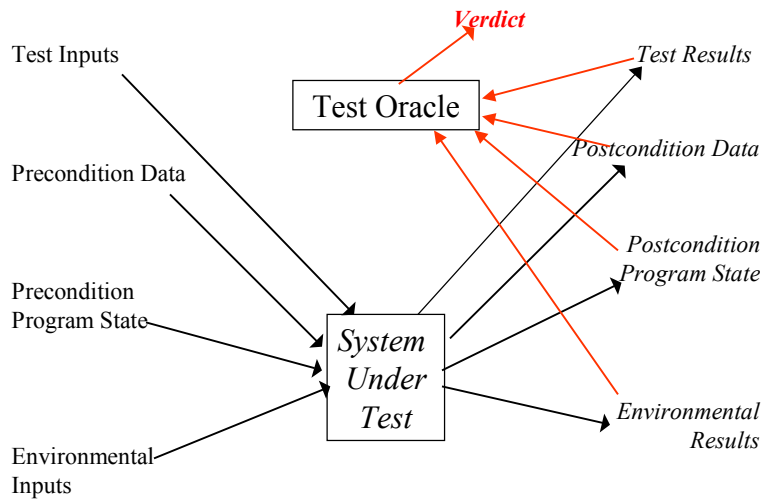
A “Consistency” Oracle



Self-Verifying Strategies

- Embed results in the data
- Cyclic algorithms
- Shared keys with algorithms

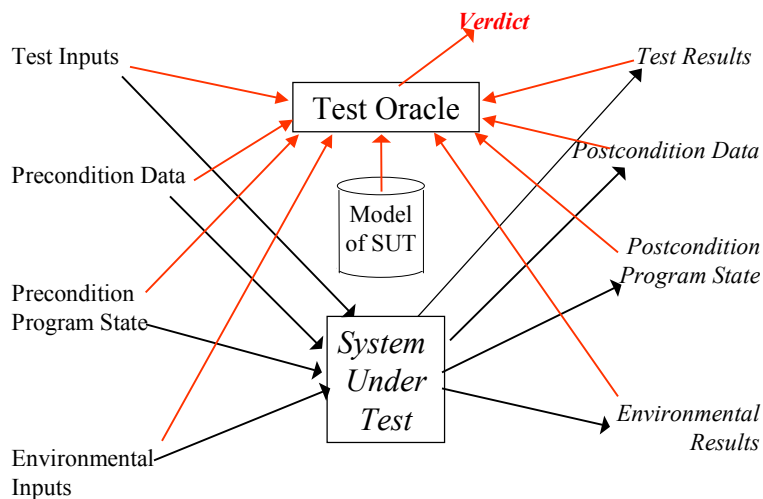
A “Self-Verifying Data” Oracle



Model Based Strategies

- Identify and describe a machine readable form of some aspect of the SUT
- Design test(s) for the modeled behavior based on having the model available
- Implement the test(s), reading in the model
- Update the model as needed

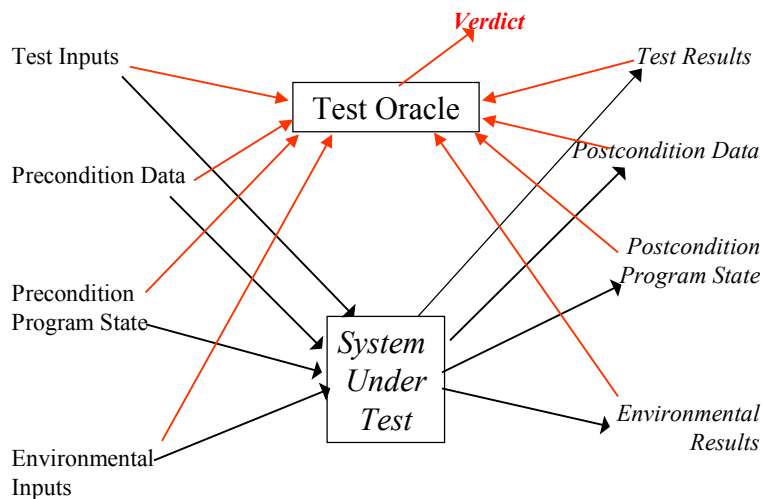
A “Model Based” Oracle



Hand Crafted Oracle Strategies

- Expected result is carefully crafted with input values
- Input and result are specified together
- Oracle is frequently built into test case

A “Hand Crafted” Oracle

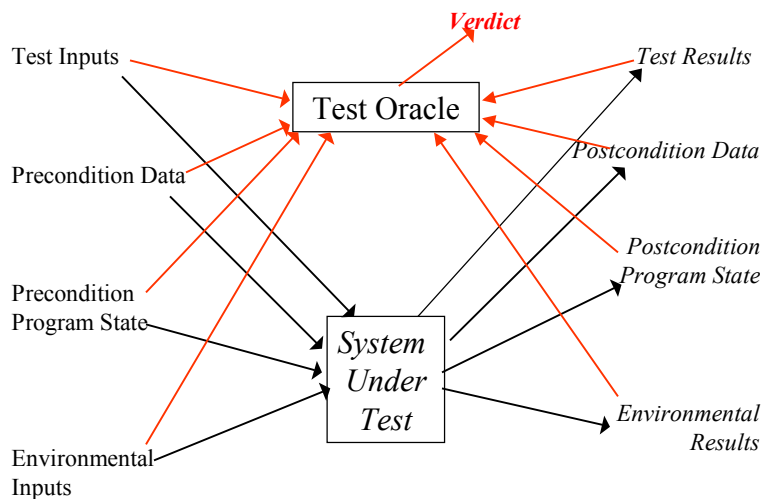


Heuristic Oracles

A Heuristic Oracle uses an approximation (a rule of thumb) or partial information that supports but does not mandate a given conclusion. We may use a probabilistic evaluation. This won't tell you that the program works correctly but it can tell you that the program is broken or something needs more investigation.

- This can be a cheap way to spot errors early in testing.
- (Note that most heuristics are prone to both Type I and Type II errors.)

A "Heuristic" Oracle



Statistical Oracles

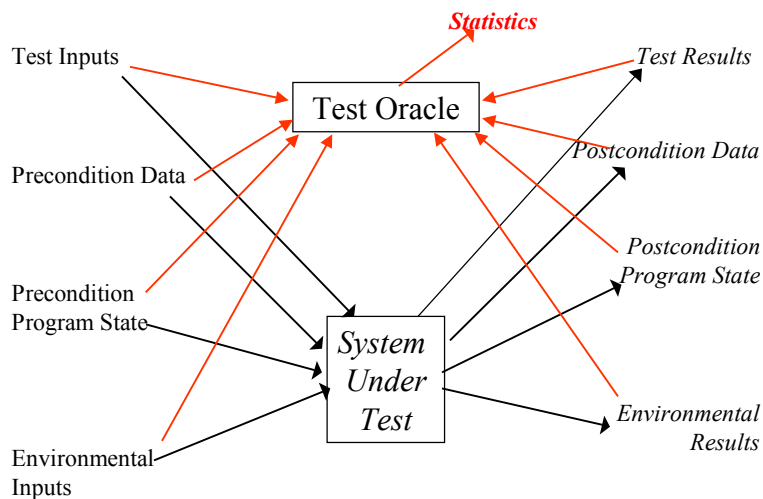
- Principle idea
 - High-volume testing with new cases
 - Results checking based on population statistical characteristics
- Fundamental goal is to have a huge numbers of tests
 - The individual tests are not all that powerful, nor all that compelling
 - Data is varied for each step
 - Individual results are not checked for correctness – population statistics are used
 - The power of the approach lies in the large number of tests

Douglas Hoffman

Copyright © 2004-9, SQM, LLC.

39

A Statistical Oracle



Douglas Hoffman

Copyright © 2004-9, SQM, LLC.

40

Software Test Automation: Test Comparators

Douglas Hoffman

Copyright © 2004-9, SQM, LLC.

41

Comparison Functions

- Data Comparisons (Oracle based)
 - Previous version
 - Competitor
 - Standard function
 - Custom model
- Computational or Logical Modeling
 - Inverse function
 - mathematical inverse
 - operational inverse (e.g. split a merged table)
 - Useful mathematical rules (e.g. $\sin^2(x) + \cos^2(x) = 1$)

Douglas Hoffman

Copyright © 2004-9, SQM, LLC.

42

An Evaluation Mechanism

- Another common way to define an oracle is as a mechanism for telling if results are plausible
 - Under this view, the oracle accepts the inputs and the test outputs to decide if the results are consistent with good SUT behavior. If not, the program fails
 - The oracle does generation and evaluation of the results
 - The squaring of the square root is an example of this type of oracle
- There is danger that the oracle could share common mode errors with the SUT

Reference Functions: Some Typical Examples

- Spreadsheet Version N and Version N-1
 - Single function comparisons
 - Combination testing
 - What about revised functions?
- Database management operations
 - Same database, comparable functions across DBMs or query languages
- Bitmap comparisons (output files)
 - The problem of random variation

Deterministic Reference Functions

- Saved result from a previous test.
- Parallel function
 - previous version
 - competitor
 - standard function
 - custom model
- Inverse function
 - mathematical inverse
 - operational inverse (e.g. split a merged table)
- Useful mathematical rules (e.g. $\sin^2(x) + \cos^2(x) = 1$)
 - Deterministic incidental or informative attributes
- Expected result encoded into data

Heuristic Evaluation Functions

- Compare (apparently) sufficiently complete attributes
 - compare calculated results of two parallel math functions (but ignore duration, available memory, pointers, display)
- Statistical distribution
 - test for outliers, means, predicted distribution
- Compare incidental but informative attributes
 - durations
 - sequences

Heuristic Evaluation Functions

- “Fuzzy” comparisons
 - display bitmaps
 - shading
- Check (apparently) insufficiently complete attributes
 - ZIP Code entries are 5 or 9 digits
- Check probabilistic attributes
 - X is usually greater than Y

Deterministic Test Result Possibilities

Situation Test Result	No Error in SUT	Error in SUT
As Expected (Pass)	Correct Pass	Silent Miss (Test Error!)
Unexpected (Fail)	False Alarm (Test Error!)	Caught it!

Heuristic Test Result Possibilities

Situation Test Result	No Error in SUT	Error in SUT
As Expected (Pass)	Correct Pass	Silent Miss (Expected)
Heuristic Check (Attention)	False Alarm (Expected)	Caught it!

Software Test Automation: Design of Automated Tests

Start With a Known State

- Data
 - Load preset values in advance of testing
 - Reduce dependencies on other tests
- Program State
 - External view
 - Internal state variables
- Environment
 - Decide on desired controlled configuration
 - Capture relevant session information

Build Variation Into the Tests

- Dumb monkeys
- Variations on a theme
- Configuration variables
- Data driven tests
- Pseudo-random event generation
- Model driven and model based automation

Check for Errors

- Periodically check for errors as the test runs
- Document expectations in the tests
- Capture prospective diagnostic information before an error is detected
- Capture information when the error is found (don't wait)
 - Results
 - Other domains
 - “Dump the world”
- Check as many things as possible

Summary

- Model for automated test execution
- Definition of a test oracle
- Oracle characteristics
- Types of test oracles
- Test comparators
- Design of automated tests

