Keith Stobie , Microsoft

Don't test too much! [or too little]
(*Lessons learned the hard way*)

Agenda

- Lessons Learned in Software Testing
- Using Simple Oracles
- Test Case size and Known Errors
- Never be Gatekeeper – know release goals
- Testing Exception handling
- Alternatives to execution
- Testing Combinations

Five-fold Testing System Classification

From: *Lessons Learned in Software Testing*

Who: Developers, testers, internal users, beta users, . . .
What: Coverage – Requirements, scenarios, functions, code, errors, . . .
Why: Potential problems – Risks you are testing for.
How: Activities – Regression, exploration, installation, . . .
Evaluation: How to tell whether the test passed or failed.

Lesson #283: "Apply diverse half measures"

Use Simple Oracles

Test SquareRoot() function:

1. ~~Assert(SquareRoot(X) == MySquareRoot(X))~~
2. ~~Assert(SquareRoot(X) == StdSquareRoot(X))~~
3. Y := SquareRoot(X)

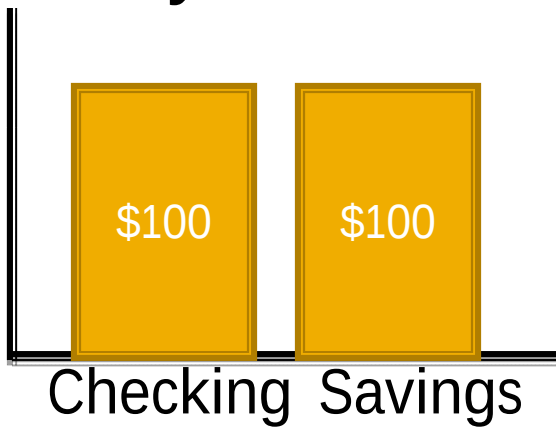
```
Assert(X == Y*Y)
```

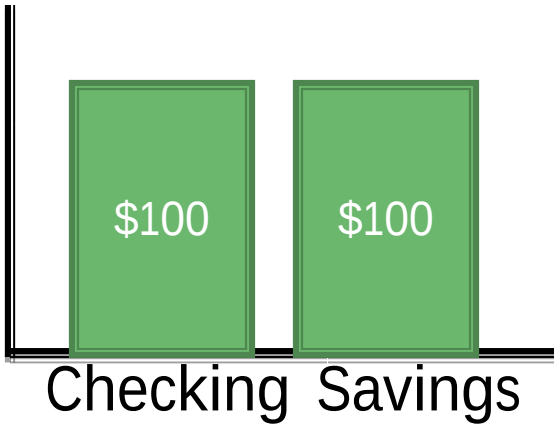
Over Testing (transactions)

- ACID Transaction
  Atomicity, Consistency, Isolation, Durability
- Over test: For each operation, repeat the same operation on a side “test” implementation.
- Test system must be as performant as Real system (to keep same level of concurrency)

Banking Transactions

- Transfer $100 from checking to savings
  1. Add money to savings
  2. Remove money from checking or Add to savings
- 2 Steps, both or neither must occur for consistency.





Simple Transaction Oracle

Maintain a consistency invariant outside of transactions (but violated inside)

1. Randomly choose first operation and value:
Transfer Savings -> Checking 1,234.57
2. Choose last operation to keep balances /1,000
   A. Add more: Savings -> Checking 765.43
   (2,000 total transfer)
   B. Add less : Checking -> Savings 234.57
   (1,000 total transfer)

Simple vs Accurate

Pros:

- Simple fast Check

(can be made post production)

Cons:

- Difficult Failure Diagnosis

(just like with real customers)

Related notions:

- Metamorphic testing
- Sampling & Consistency Oracles

lesson #125 "Avoid Complex logic in your Scripts."

Metamorphic Relation (MR)

- Reuse existing test cases to generate more tests. (Existing tests as Oracles)
- MR: Any relation among the inputs and the outcomes of *multiple* executions of SUT.

SqRt(X)*SqRt(Y) == SqRt(X*Y)

- Test 1: SqRt(9) == 3
- Gen Test 2: SqRt(9)*SqRt(2) == SqrRt(9*2)

Shortest Path examples

- ShortestPath(G, a, b)
  find the *shortest path between vertices a and b in graph G and also output* its length,
  where *G is an undirected graph with positive edge weights.* (*dijkstra's algorithm*)

- Reverse: The shortest path between B and A should be the reverse of the shortest path between A and B.
- Prefix: For any vertix, V, on shortest path between vertices A and B, the Shortest path between A and V must be the same sequence as the start of the path between A and B.

Test Boyer-Moore example

- Boyer-Moore algorithm
  returns the index of the first occurrence of a specified pattern within a given text.

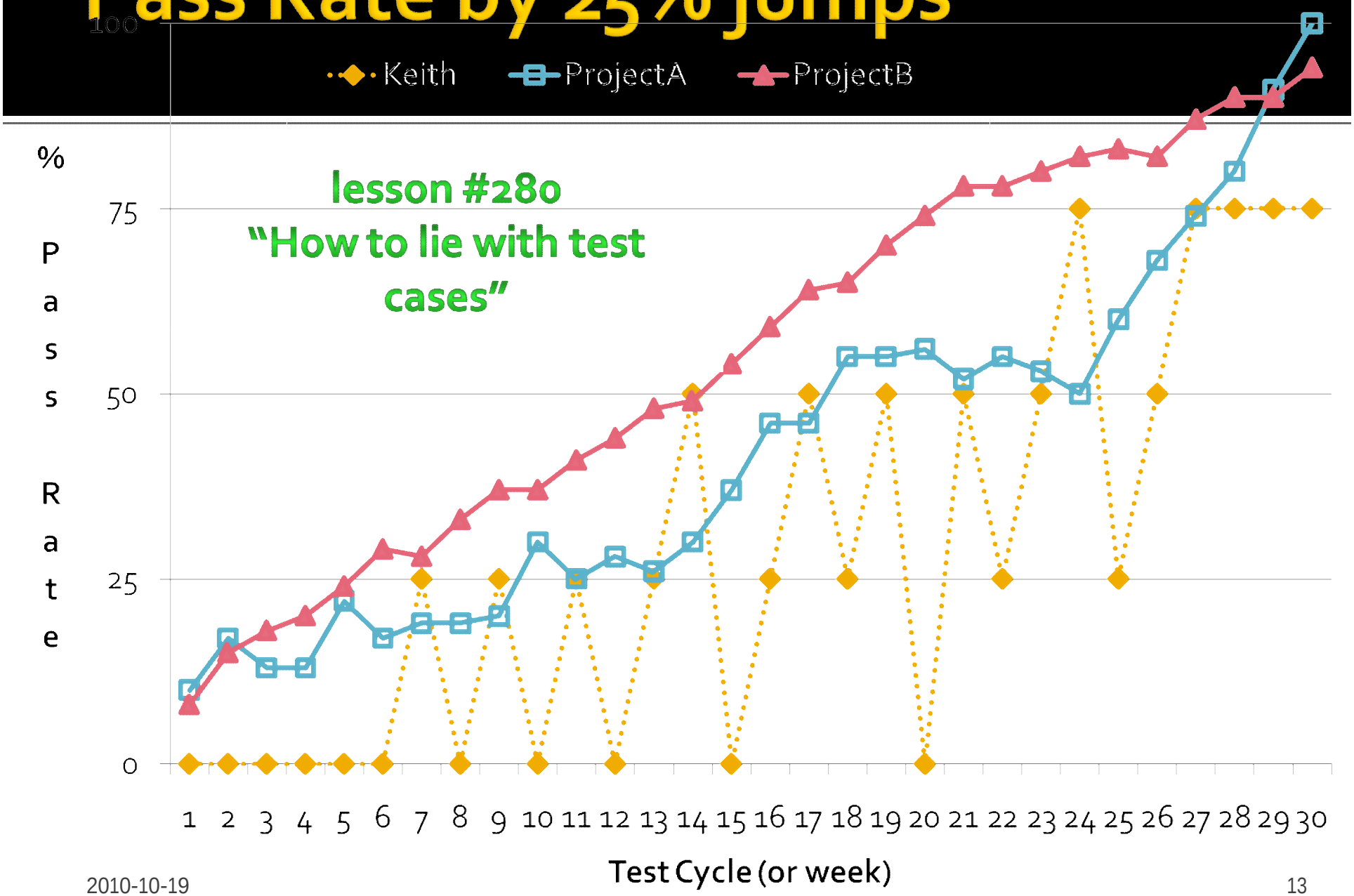
Metamorphic Relations

- Reverse: if string X exists in string Y,
  then the reverse of Y exists in the reverse of X.

Agenda

- Lessons Learned in Software Testing
- Using Simple Oracles
- Test Case size and Known Errors
- Never be Gatekeeper – know release goals
- Testing Exception handling
- Alternatives to execution
- Testing Combinations

Pass Rate by 25% jumps



Test in the Presence of Failures

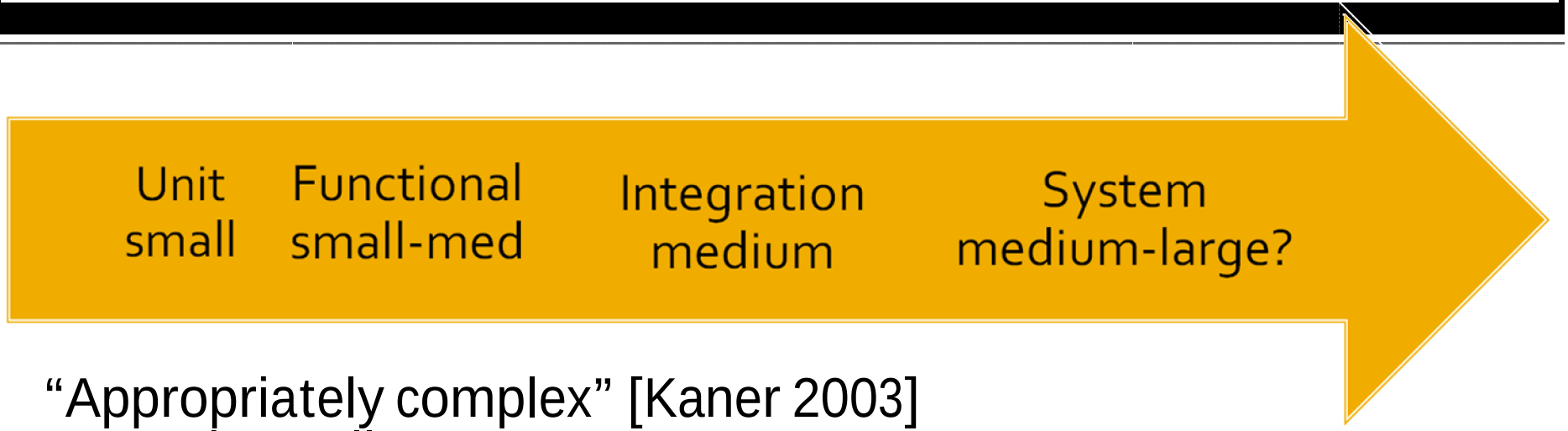
- Spec Explorer “long test”
  Ideal: Single test for all transitions in a model
  Pass/Fail test == Pass/Fail product
- Simplest way to design test cases is to fail fast.

  As soon as an error is detected, note the error and stop any further processing (Assert)
- Information loss when many Asserts.

  Symptom: Many “blocking” failures

lesson #34 “It Works” really means it appears to meet some requirement to some degree”

Test Case Size



"Appropriately complex" [Kaner 2003]

Towards smaller tests:

- Fewer blocking failures (less dependency/commonality)
- Parallel execution.
- Test Prioritization for regression (running < whole set)
- Automatic Failure Analysis tools

Towards larger tests:

- Reduced redundant setup/cleanup time
- Chance of inadvertently implementing a test idea that finds a bug. [Marick 2000]

Agenda

- Lessons Learned in Software Testing
- Using Simple Oracles
- Test Case size and Known Errors
- Never be Gatekeeper – know release goals
- Testing Exception handling
- Alternatives to execution
- Testing Combinations

Lesson #1
"N

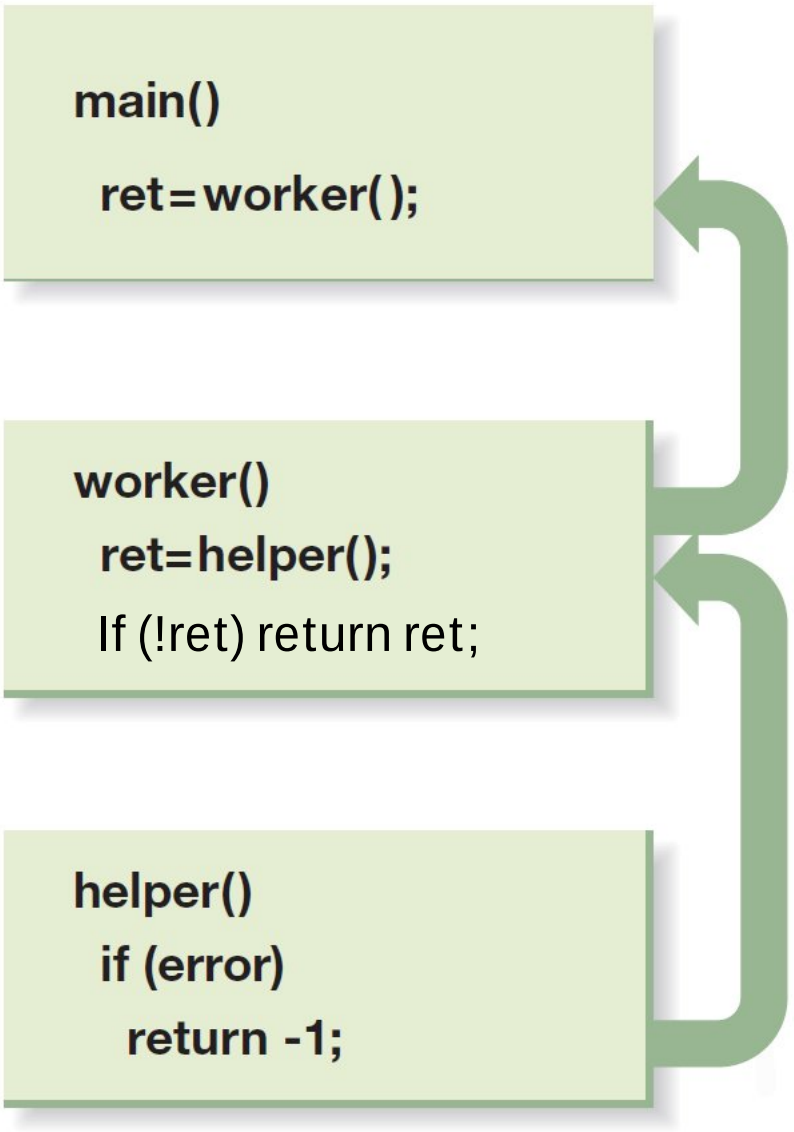
-
- J
- S

#205: "Don't sign-off to approve the release of a product"

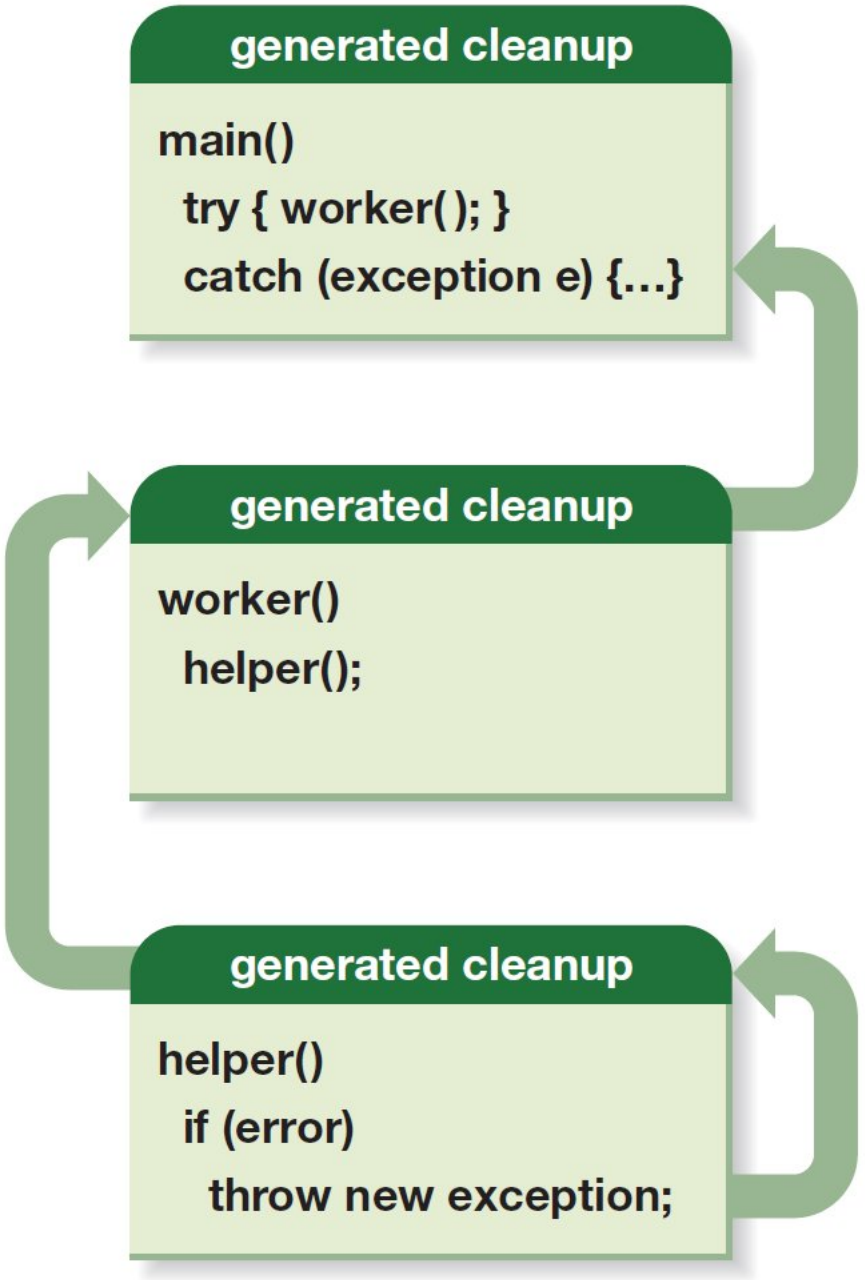
Agenda

- Lessons Learned in Software Testing
- Using Simple Oracles
- Test Case size and Known Errors
- Never be Gatekeeper – know release goals
- Testing Exception handling
- Alternatives to execution
- Testing Combinations

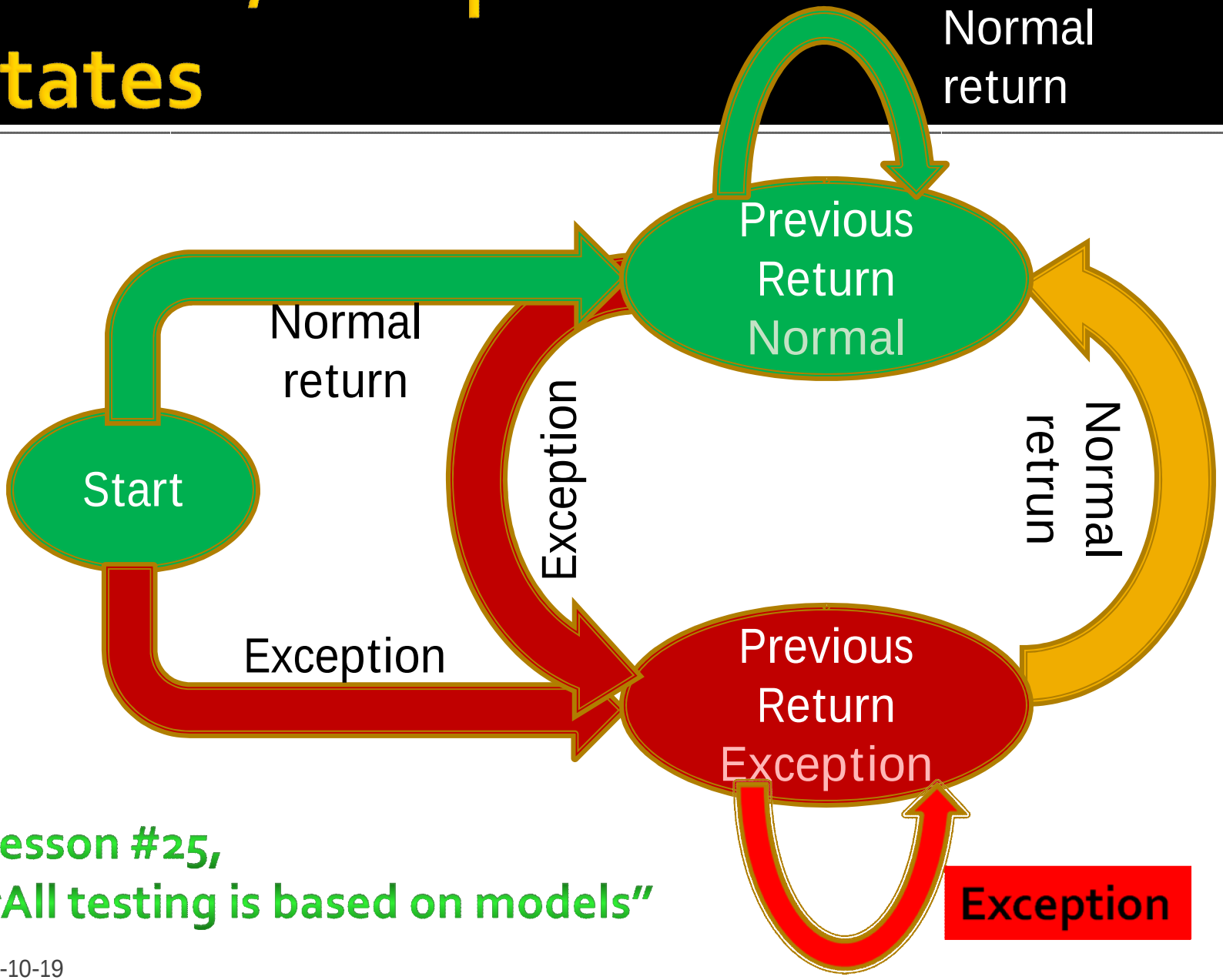
Old: Returning Error codes



Modern: Exception Propagation

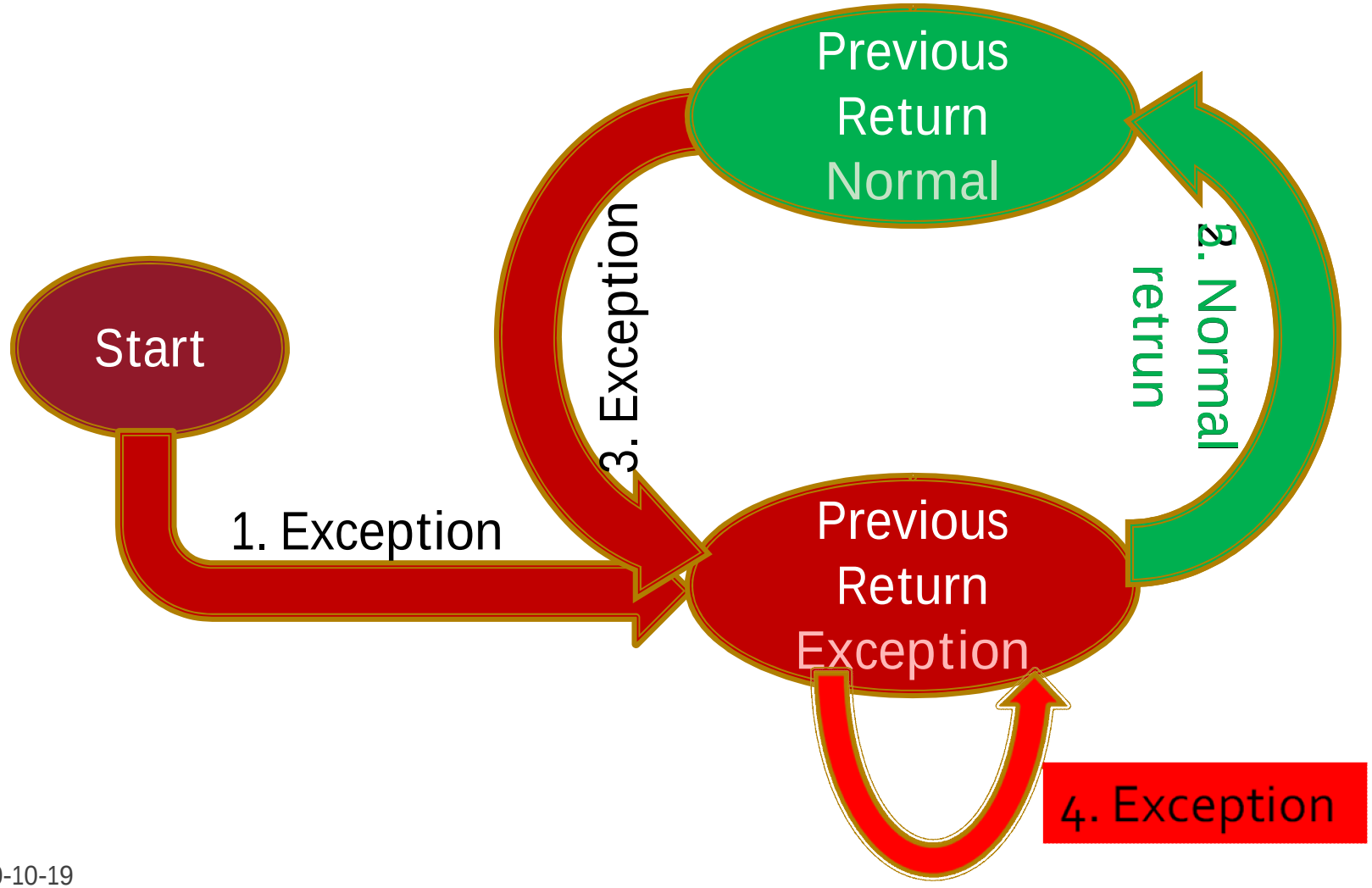


Normal/Exception States



Lesson #25,
"All testing is based on models"

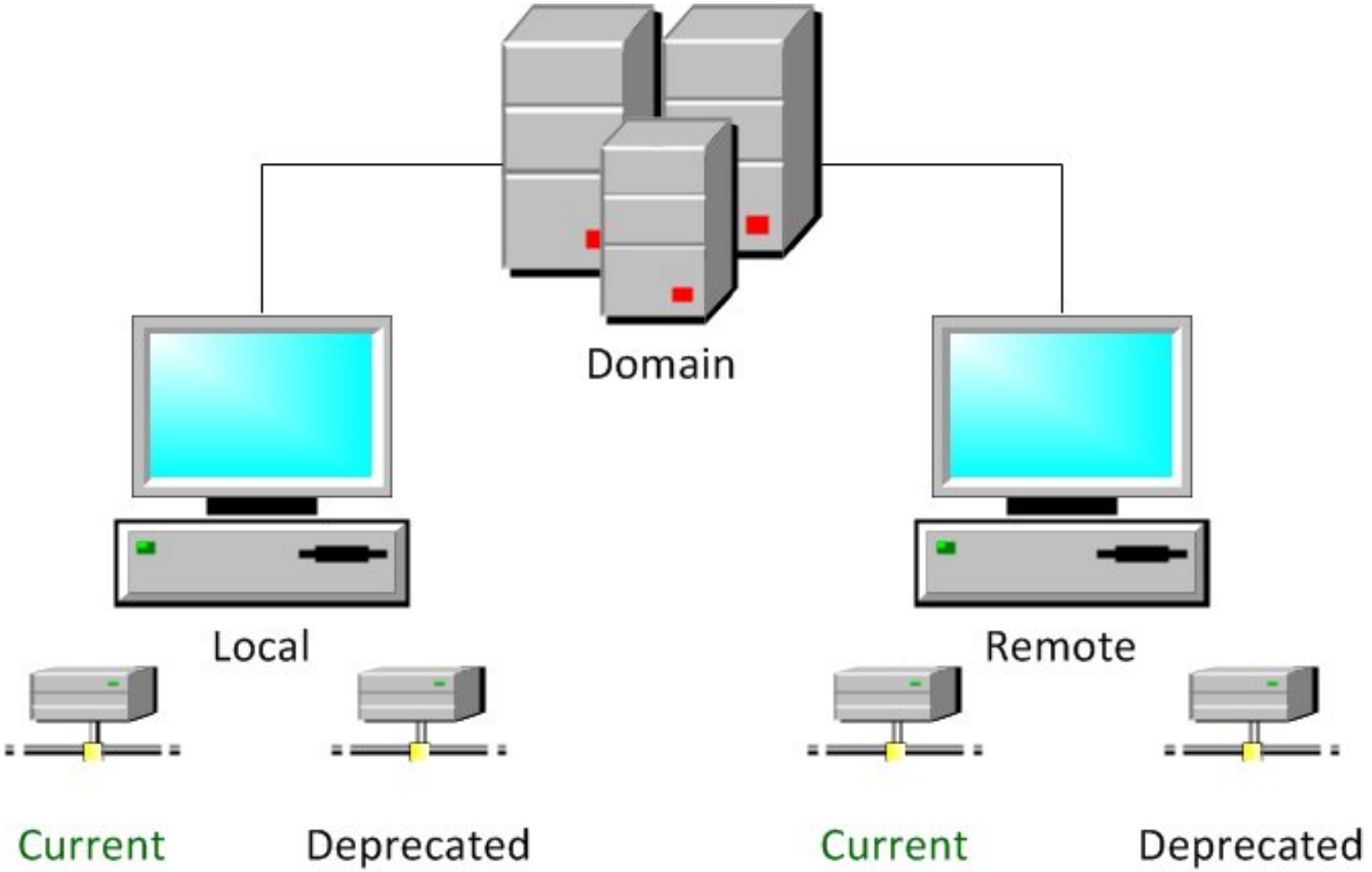
Pattern to Exercise Transitions



Agenda

- Lessons Learned in Software Testing
- Using Simple Oracles
- Test Case size and Known Errors
- Never be Gatekeeper – know release goals
- Testing Exception handling
- Alternatives to execution
- Testing Combinations

Avoiding hard stuff?

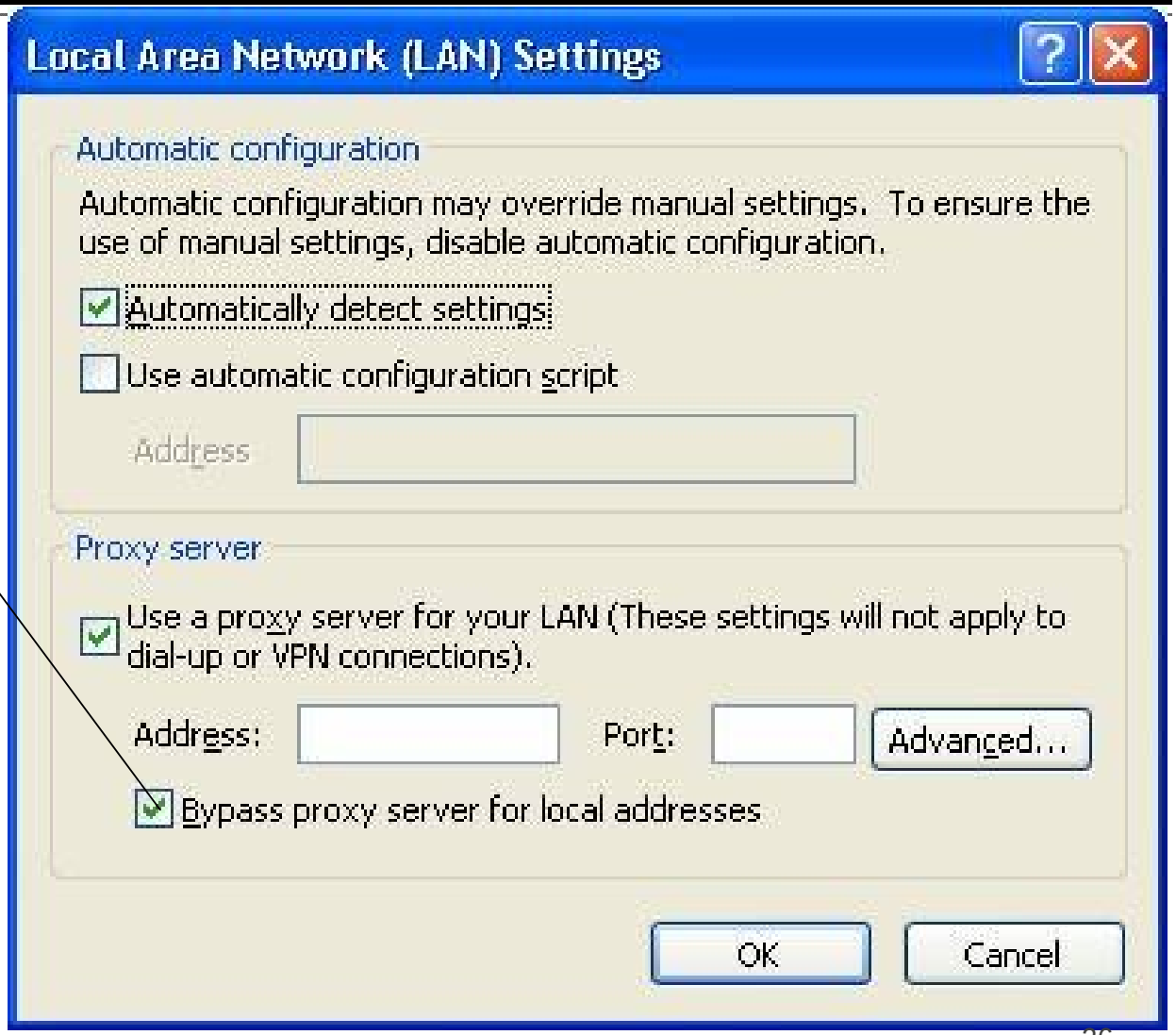


Agenda

- Lessons Learned in Software Testing
- Using Simple Oracles
- Test Case size and Known Errors
- Never be Gatekeeper – know release goals
- Testing Exception handling
- Alternatives to execution
- Testing Combinations

Testing Combinations

In the code,
the BypassProxyOnLocal property on the WebProxy object does not reflect the fact that it is checked



Pairwise PICT results

- Visual Basic installation testing:
  - 30% fewer test cases vs. minimal test suite created manually: 60 tests from PICT vs. 80-90 manual out of 200 possibilities (exhaustive). Testers had high confidence in quality
  - 235 tests generated for a domain with over 1M possible combinations. All these tests implemented in less than one week; traditional methods: 15-30 tests per week. An order of magnitude of a difference.
- Games testing:
  - *Asheron's Call* testing: the team was able to cut test resources by half due to use of PICT
  - *Vanguard* testing: 4 man-months to build and implement test cases vs. 44 man-months during last release. Process more efficient by an order of magnitude
- Windows command-line tools testing: (findstr.exe &)
  - *Attrib.exe* testing: 13 test cases generated by PICT achieved identical block coverage as 370 exhaustive test cases

Summary

Testing is a business activity with a cost.

- Understand the product and release goals,
- Deal with common issues like known errors,
- Anticipate likely errors
- Choose the right evaluation methods

Remember

- Use Simple Oracles
- Allow for Known Errors
- Choose "appropriately complex" Test Case size
- Know release quality goals ~~GateKeeeper~~
- Test Exception handling
- Consider Pairwise Testing of Combinations
- Consider Alternatives to Execution

Thanks!