
Mutating Automated Tests

ASQ Western Regional Conference '02

Douglas Hoffman

Software Quality Methods, LLC.

24646 Heather Heights Place

Saratoga, California 95070-9710

Phone 408-741-4830

Fax 408-867-4550

doug.hoffman@acm.org

www.SoftwareQualityMethods.com

Copyright © 2000 - 2002, Software Quality Methods, LLC. No part of these graphic overhead slides may be reproduced, or used in any form by any electronic or mechanical duplication, or stored in a computer system, without written permission of the author.

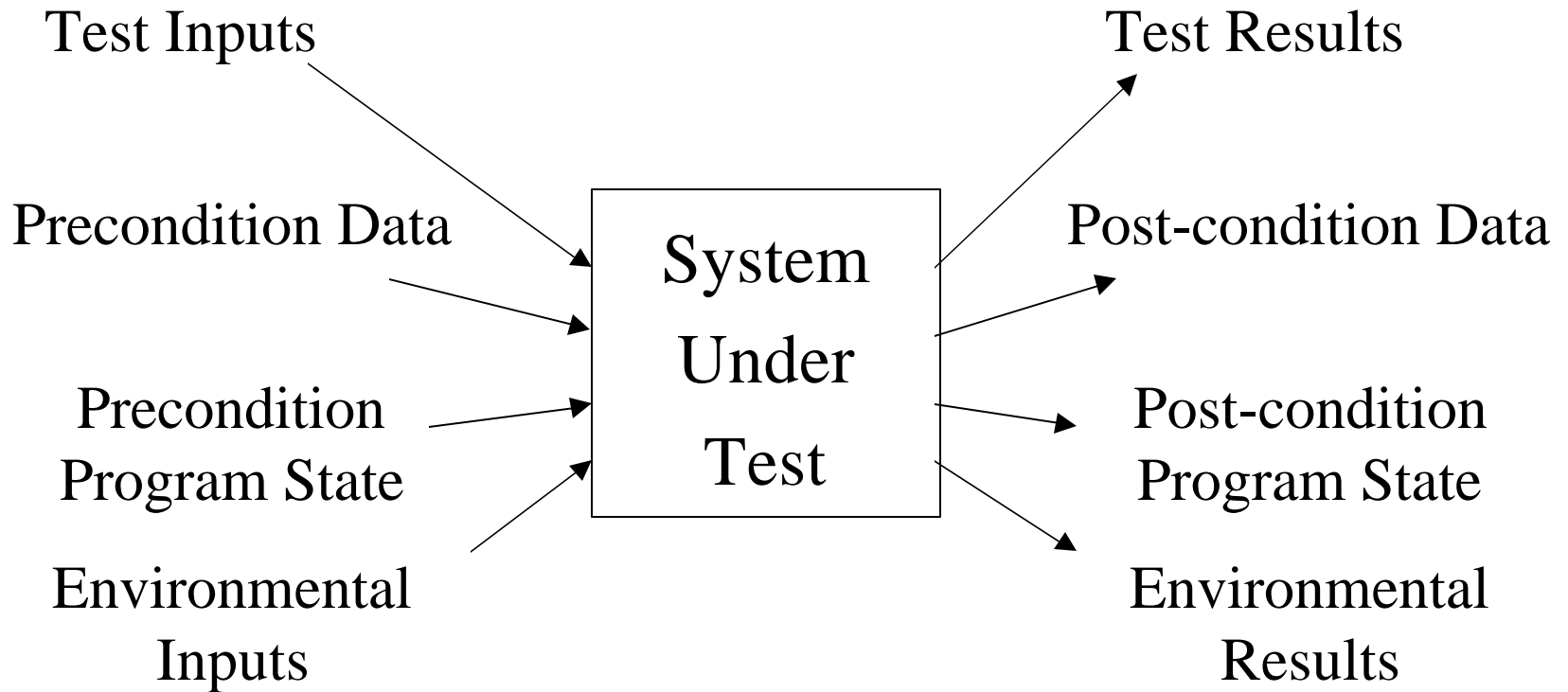
Automated Software Tests

- No intervention needed after launching tests
- Automatically sets-up and/or records relevant test environment
- Runs test exercise
- Captures relevant results
- Evaluates actual against expected results
- Reports analysis of pass/fail

Levels of Automation

- Fully automated software testing
- Semi-automated software testing
- Manual software testing

Expanded Black Box Testing Model



Designing Good Automated Tests

- Start with a known state
- Design variation into the tests
 - configuration variables
 - specifiable (e.g. table-loadable) data values
- Check for errors
- Put your analysis into the test itself
- Capture information when the error is found (not later)
 - test results
 - environment results
- Don't encourage error masking or error cascades

Advantages To Automation

- Repeatable
- Faster running
- Reusable components
- Standardized formats
- Easy to generalize
- Better environment control

Disadvantages To Automation

- Requires test tools
 - Expensive
 - Confine the paradigm
- Time consuming test creation
- Does the same thing each time
- Limit possible observations
- Tests and tools require real maintenance

First Generation Automation



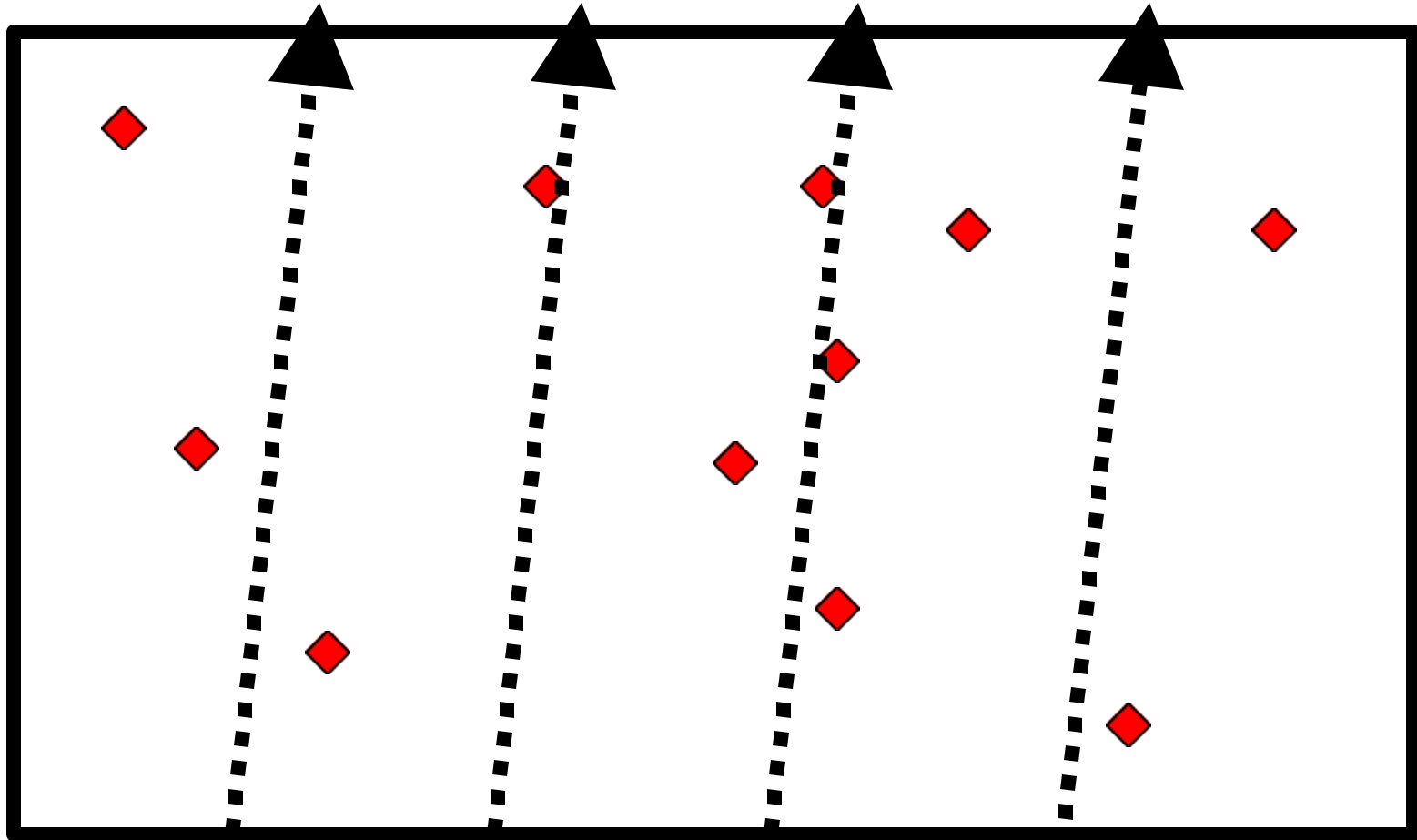
- Automate existing tests by creating equivalent exercises
- Small improvements
- Test scripting
- Reuse test components
- Hard coded oracles

Automating Regression Testing

This is the most commonly discussed regression automation approach:

- create a test case
- run it and inspect the output
- if the program fails, report a bug and try again later
- if the program passes the test, save the resulting outputs
- in future tests, run the program and compare the output to the saved results
- report an exception whenever the current output and the saved output don't match.

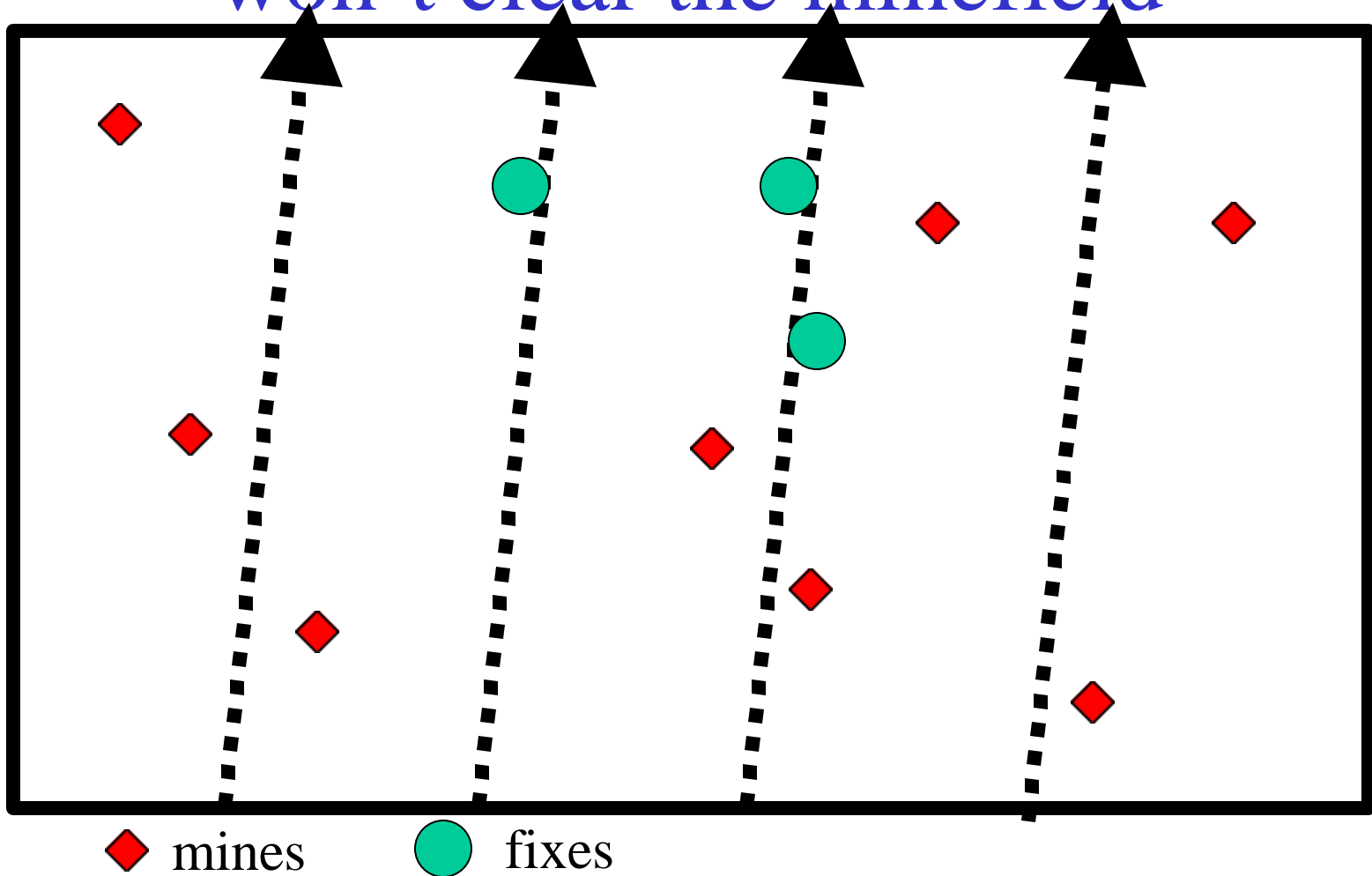
Testing Analogy: Clearing Mines



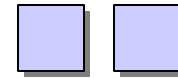
◆ mines

Thanks to James Bach for letting me use his slides.

Totally repeatable tests won't clear the minefield



Second Generation Automation

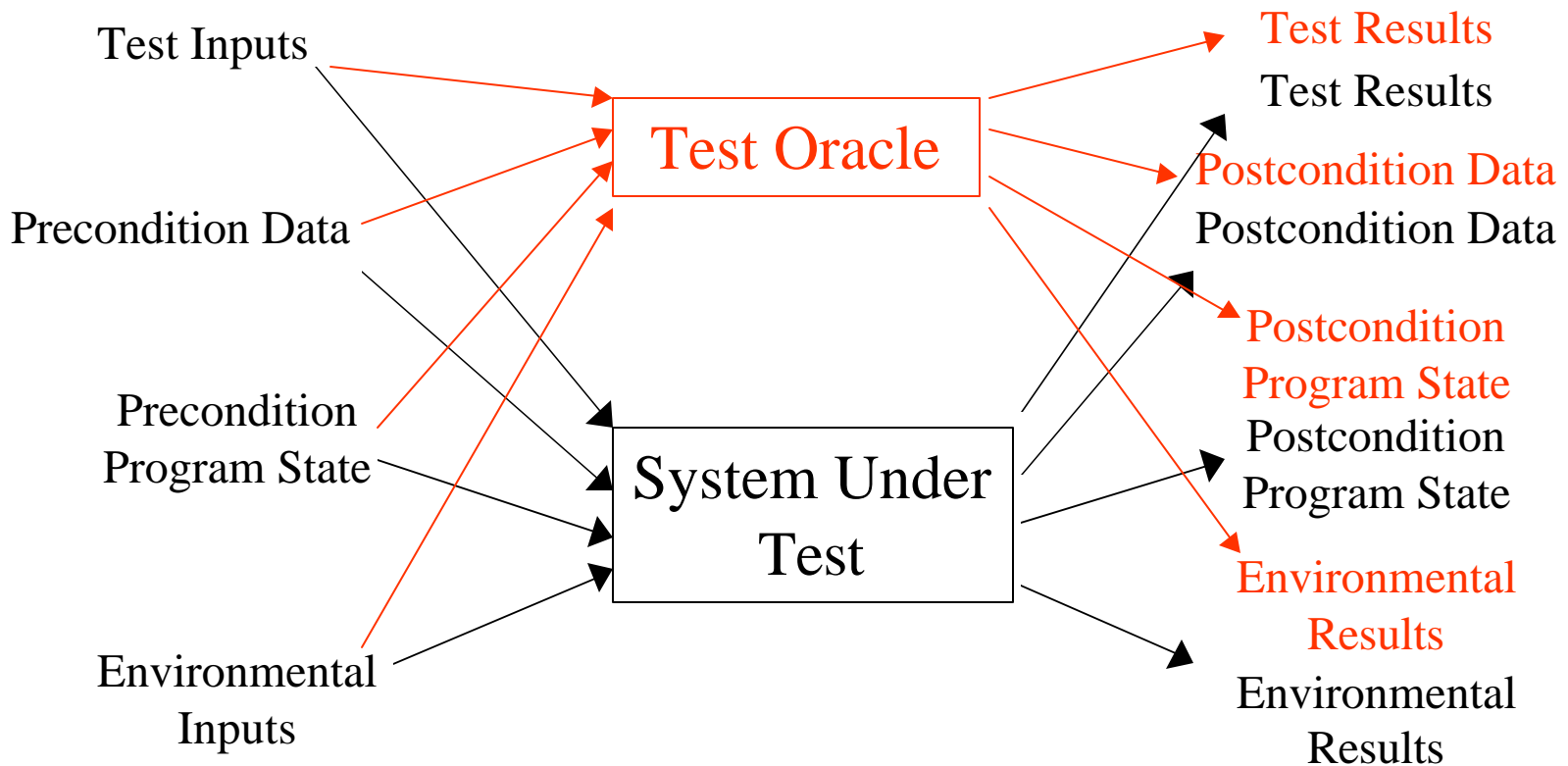


- Automated oracles
- Exhaustive/extensive/intensive exercises
- Auto generated tests and data
- More powerful exercises
- Random selections among alternatives

Increasing The Power Of Tests

- Self verifying tests
 - Check results ASAP in the test
 - Dump data only on errors
- More general exercises
 - All alternatives in turn
 - “Walk the tree” approach
- Include positive and negative cases

Testing With An Oracle



More Powerful Exercises

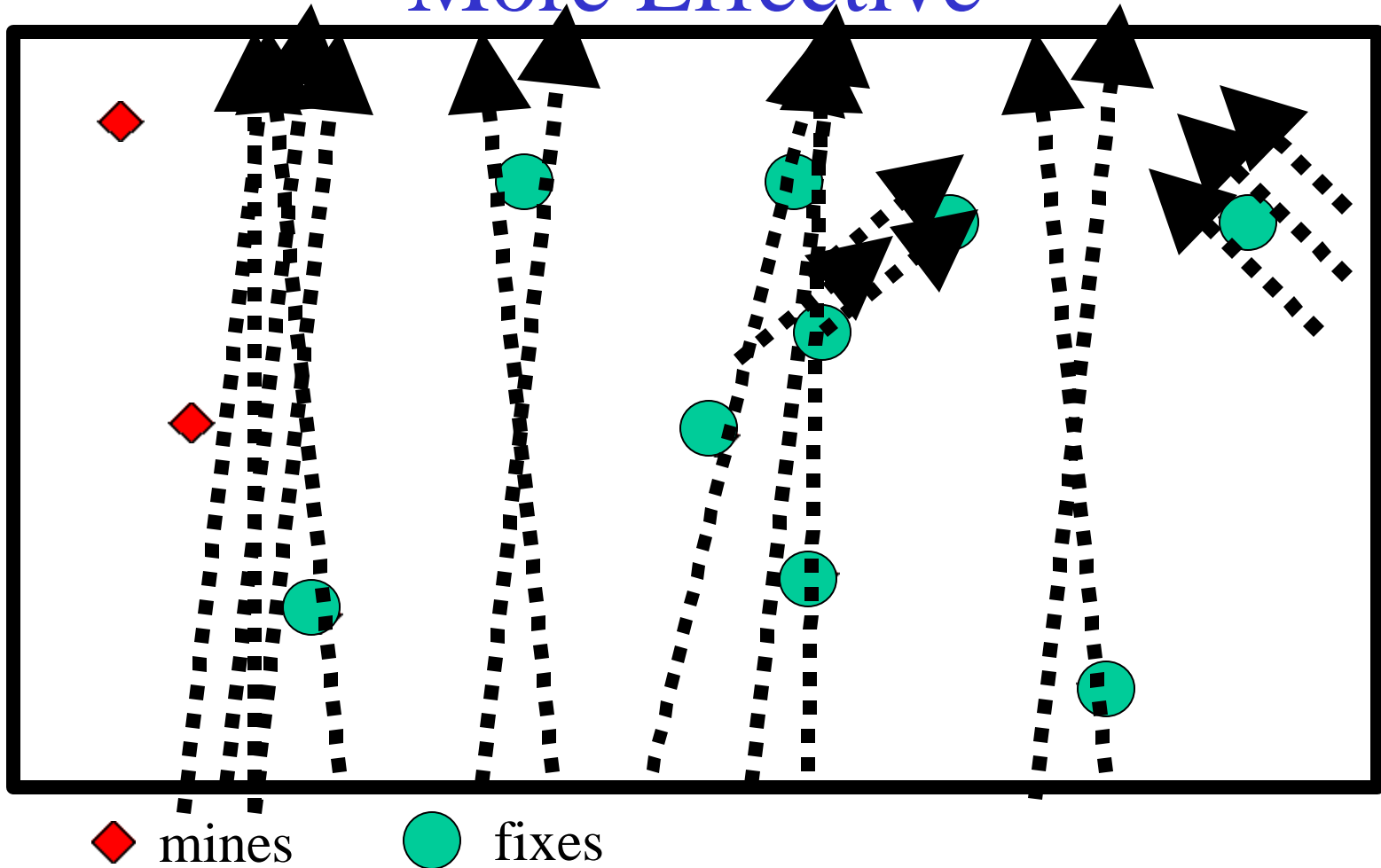
- Increase the number of combinations
- Self-verifying tests and diagnostics
- More frequency, intensity, duration
- Increase the variety in exercises

Random Selection Among Alternatives

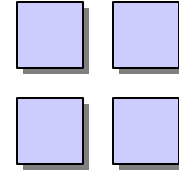
- Partial domain coverage
- Small number of combinations
- Requires an oracle for verification
- Pseudo random number generation

The beginnings of mutating tests!

Variable Tests are More Effective



Third Generation Tests



- System instrumentation
- Multi-threaded tests
- Fall back compares
- Heuristic oracles
- Diagnostics

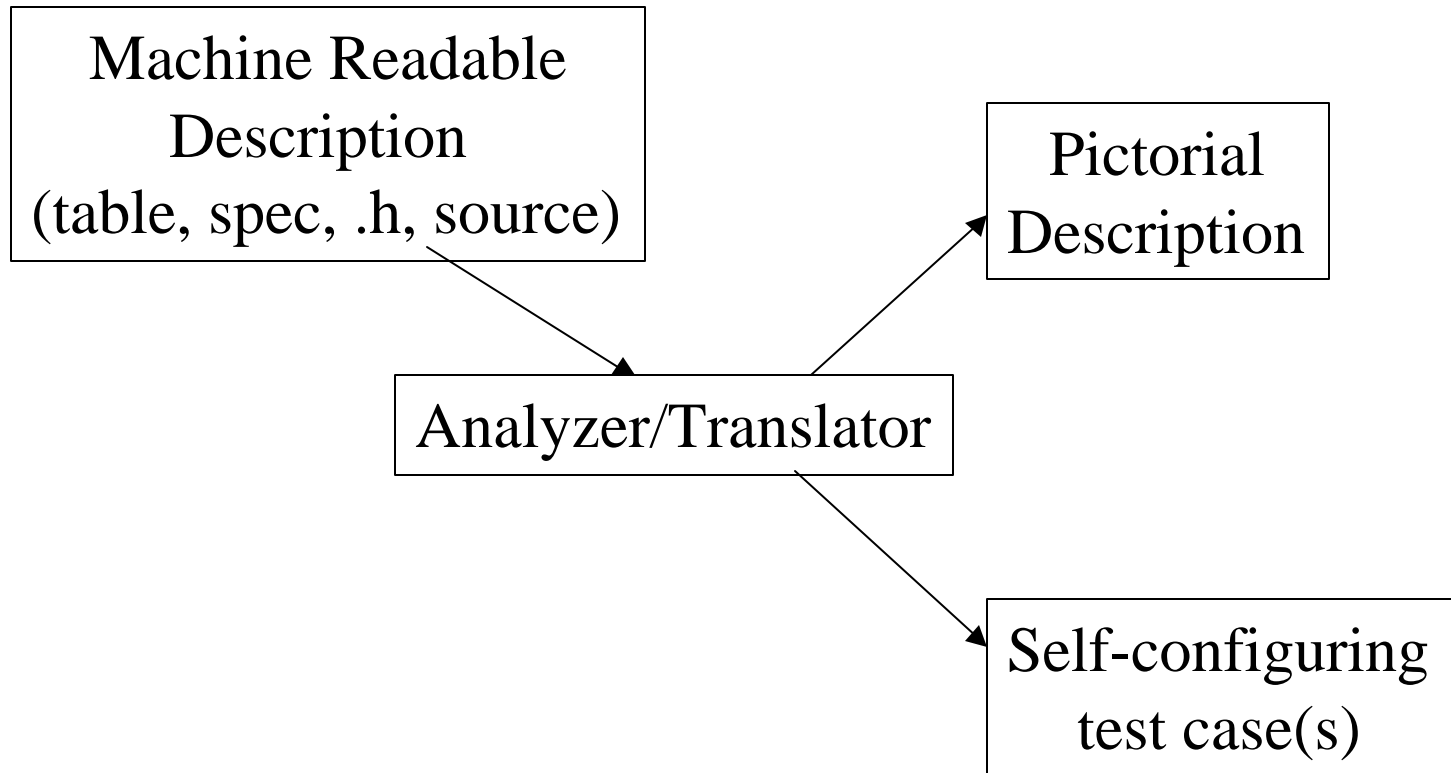
Tools For Third Generation

- Software instrumentation
 - Observations
 - Controls
 - Stimulation
- Oracles and comparators
- Test execution control

Data-Driven Architecture

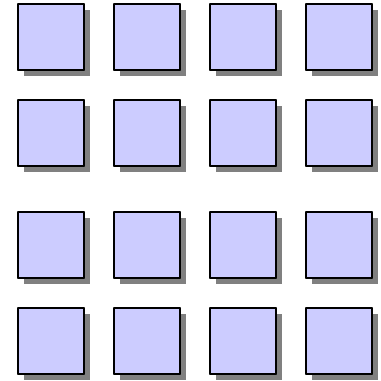
- In general, we can benefit from separating the treatment of one type of data from another with an eye to:
 - optimizing the maintainability of each
 - optimizing the understandability (to the test case creator or maintainer) of the link between the data and whatever inspired those choices of values of the data
 - minimizing churn that comes from changes in the UI, the underlying features, the test tool, or the overlying requirements
- You store and display the different data in whatever way is most convenient for you

Auto-Configuring Tests



Mutating Automated Tests

- Closely tied to instrumentation and oracles
- Using pseudo random numbers
 - Random selection from domains
 - Nesting and looping
 - With and without replacement
- Positive and negative cases possible
- Drill down on error



Mutating Tests Examples

- Data base contents
- Processor instruction sets
- Compiler language syntax
- Stacking of data objects

Summary

- Automated tests can be powerful
- Static automated tests are unlikely to find defects
- More powerful automated tests cannot be duplicated manually

