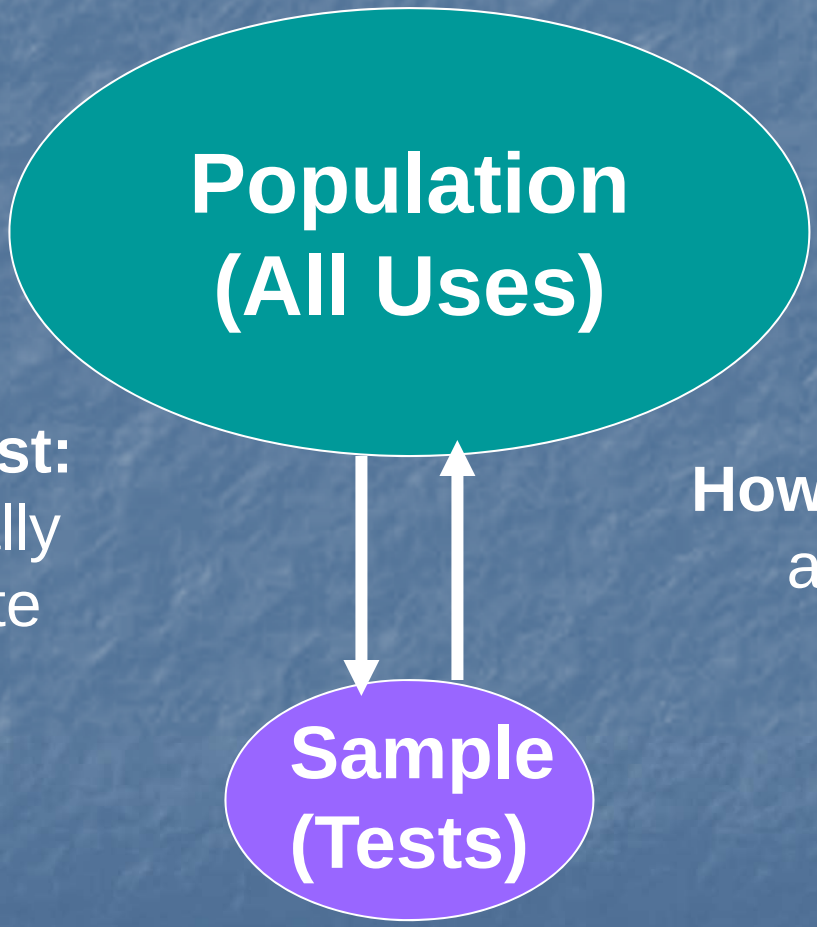
Model-Based Statistical Testing

Tom Swain
tomswain@comcast.net

A Statistical Approach Frames Testing in Science

- The question is not whether to test, but:
  - what to test
  - how much to test
- Statistical science supports rational, economical, and defensible answers to these questions.

Testing is Always Sampling



What to test:
a statistically appropriate sample

How much to test:
a risk/benefit tradeoff

Statistical Usage Testing

Statistical Usage Testing involves

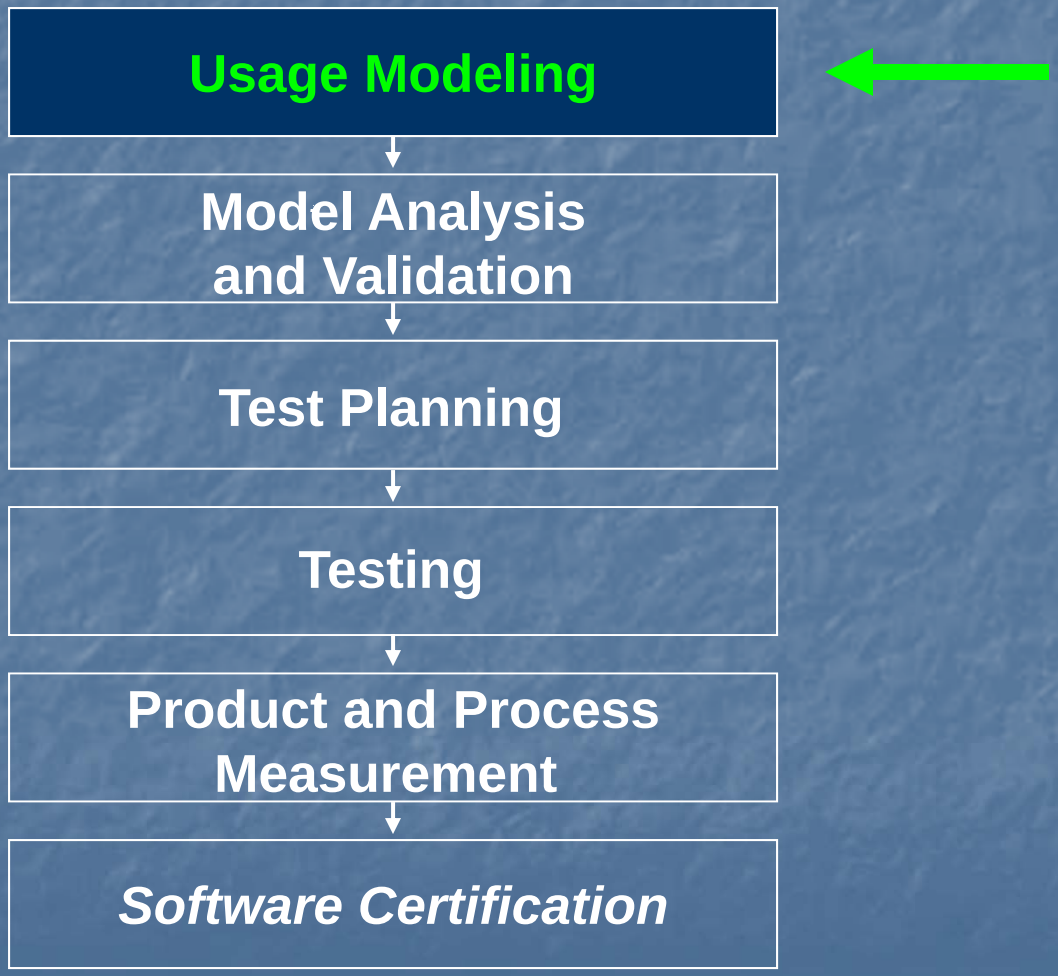
- Modeling the population of uses
- Selecting a sample for testing
- Estimating field performance from test sample results
- Making a release decision

Statistical Testing can be used across the Life Cycle

- *Development testing assesses evolving product quality*
- *Certification testing estimates fielded product performance*

The Statistical Testing Process

Software Testing Process

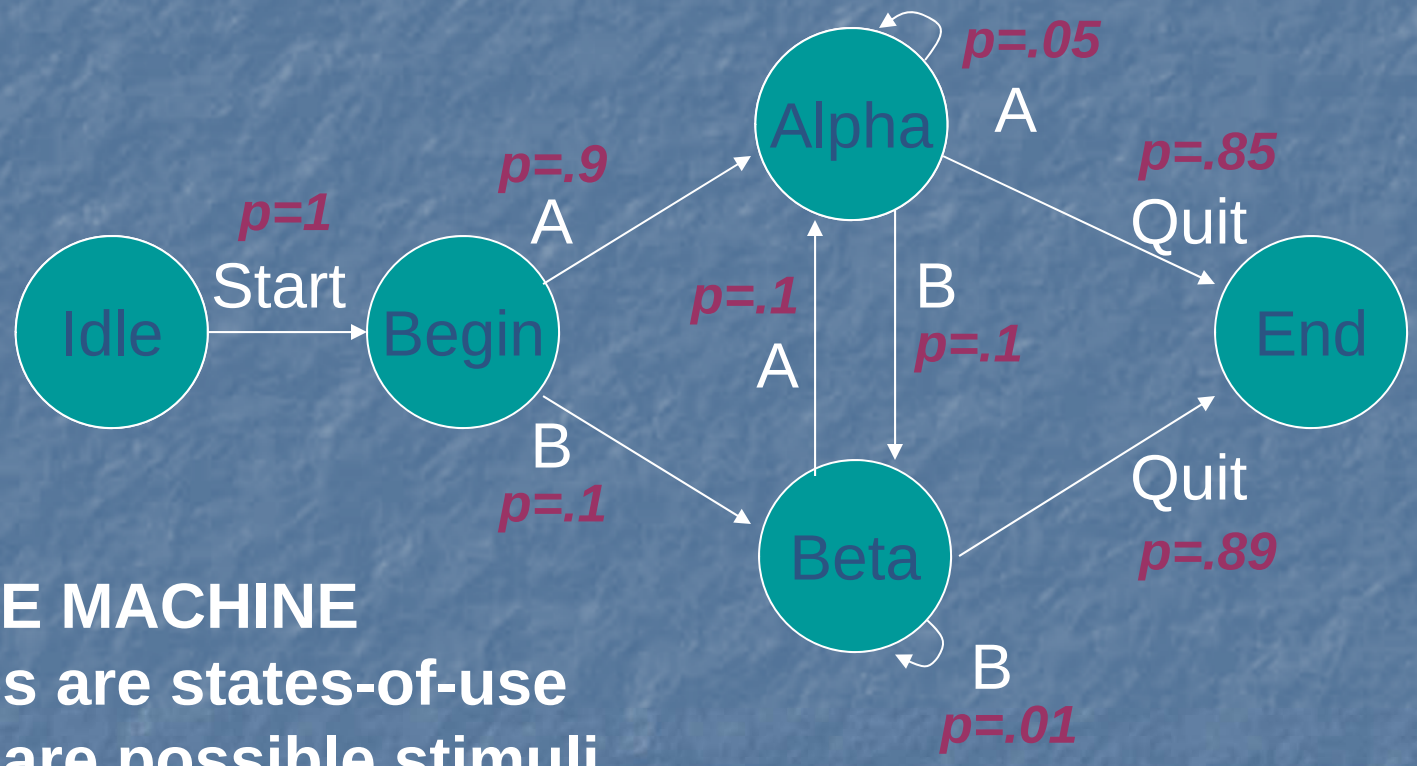


Usage Modeling

A usage model is a statistical representation of <u>all</u> possible uses of the system.

- model structure is a directed graph
  - nodes are states-of-use
  - arcs are possible transitions
- usage profile
  - probability of making each transition
- one structure, many profiles

Usage Model: All Possible Uses and their Likelihood



STATE MACHINE
Nodes are states-of-use
Arcs are possible stimuli
Probabilities define expected usage
Test case is path from initial to terminal state

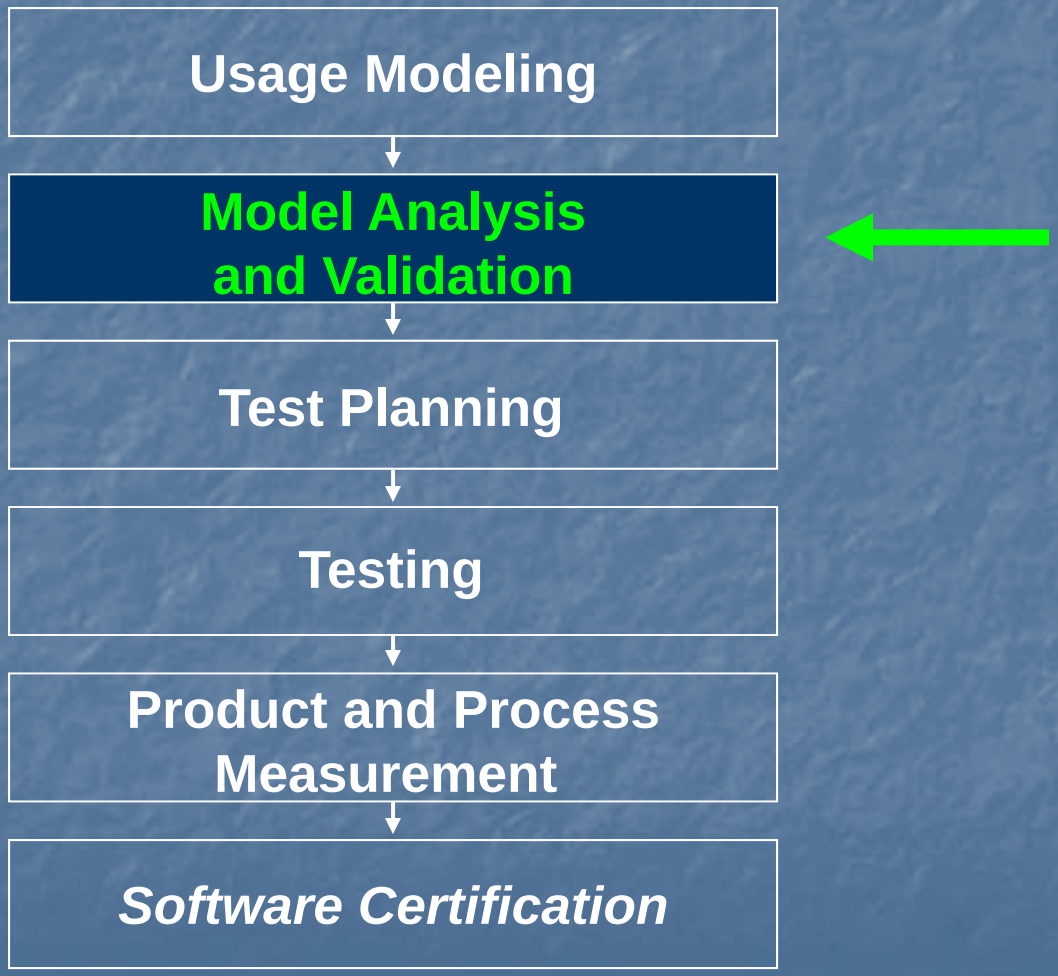
Example:
Two Usage Profiles

One structure *Two sets of transition probabilities*

From-State	To-State	Uniform Probability	Specific Profile
Idle	Ready	1	1
Ready	Entry Error	.25	.2398
Ready	1_OK	.25	.7193
Ready	Ready	.25	.0400
Ready	Alarm	.25	.0010
Entry Error	Ready	.33	.9590
Entry Error	Entry Error	.33	.0400
Entry Error	Alarm and Entry Error	.33	.00001
1_OK	Entry Error	.2	.2304
1_OK	Ready	.2	.0096
1_OK	2_OK	.2	.7200
1_OK	1_OK	.2	.0400
1_OK	Alarm and Entry Error	.2	.00001

...

Software Testing Process



A Usage Model is a Finite-State *Markov Chain*

- Well-understood formalism
- Rich body of analytical results
- Engineering basis for testing
- Objectivity in test planning and management
- Simulates “use” of product and not the product itself

Standard Analysis

All standard statistics for Markov chains have meaningful interpretations in the application to software testing.

- Long run occurrence of each state
- Expected number of transitions in a test case
- Expected number of transitions until a particular state first appears
- Estimate of the necessary and sufficient number of test cases to run

Model Revision and Validation

- Analytical results are inescapable, given the model.
- If results do not reflect what is known of the real-world usage, the model must be revised.
- Analyze and revise the model until the it is an acceptable description of expected use.

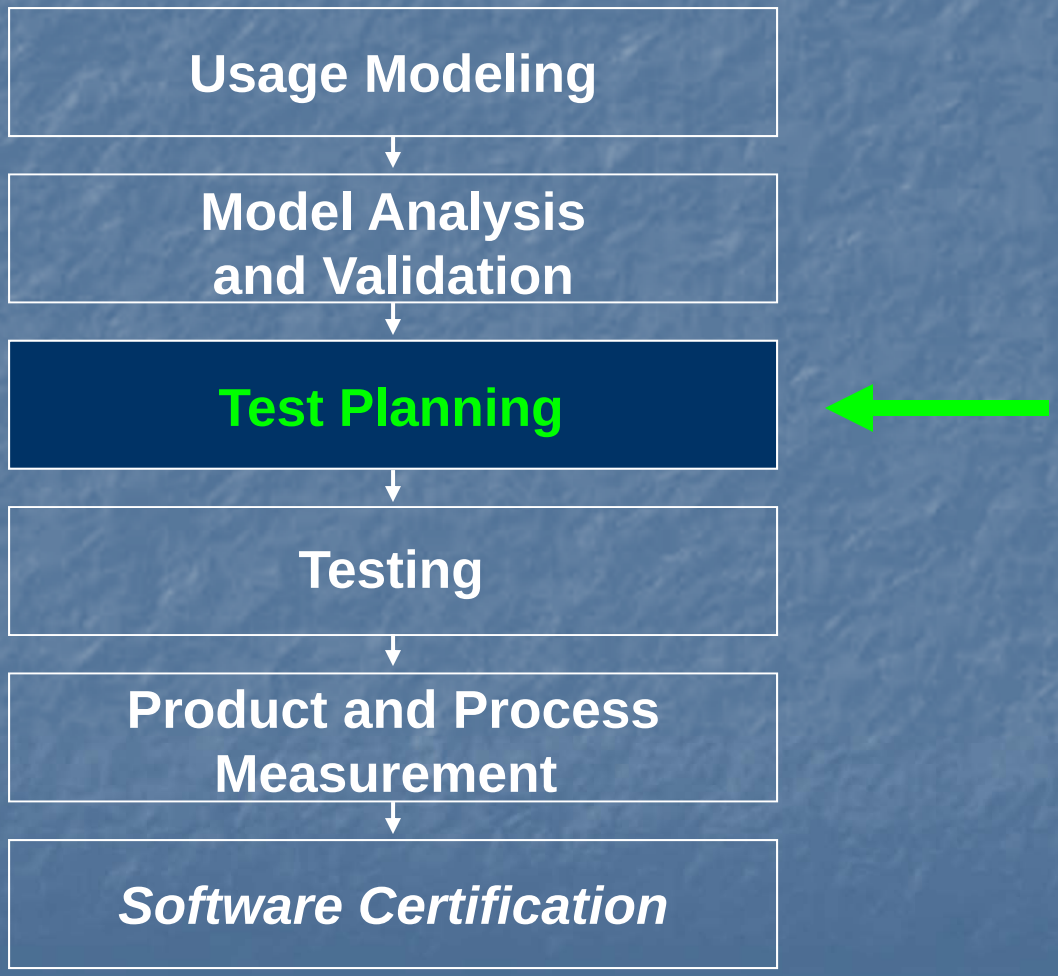
Generating Models From Systems of Constraints

- An alternative to the analyze-revise cycle.
- Describe the model as a system of constraints.
- Set objective functions to optimize.
- Generate the Markov chain transition matrix as the solution to the system of constraints.

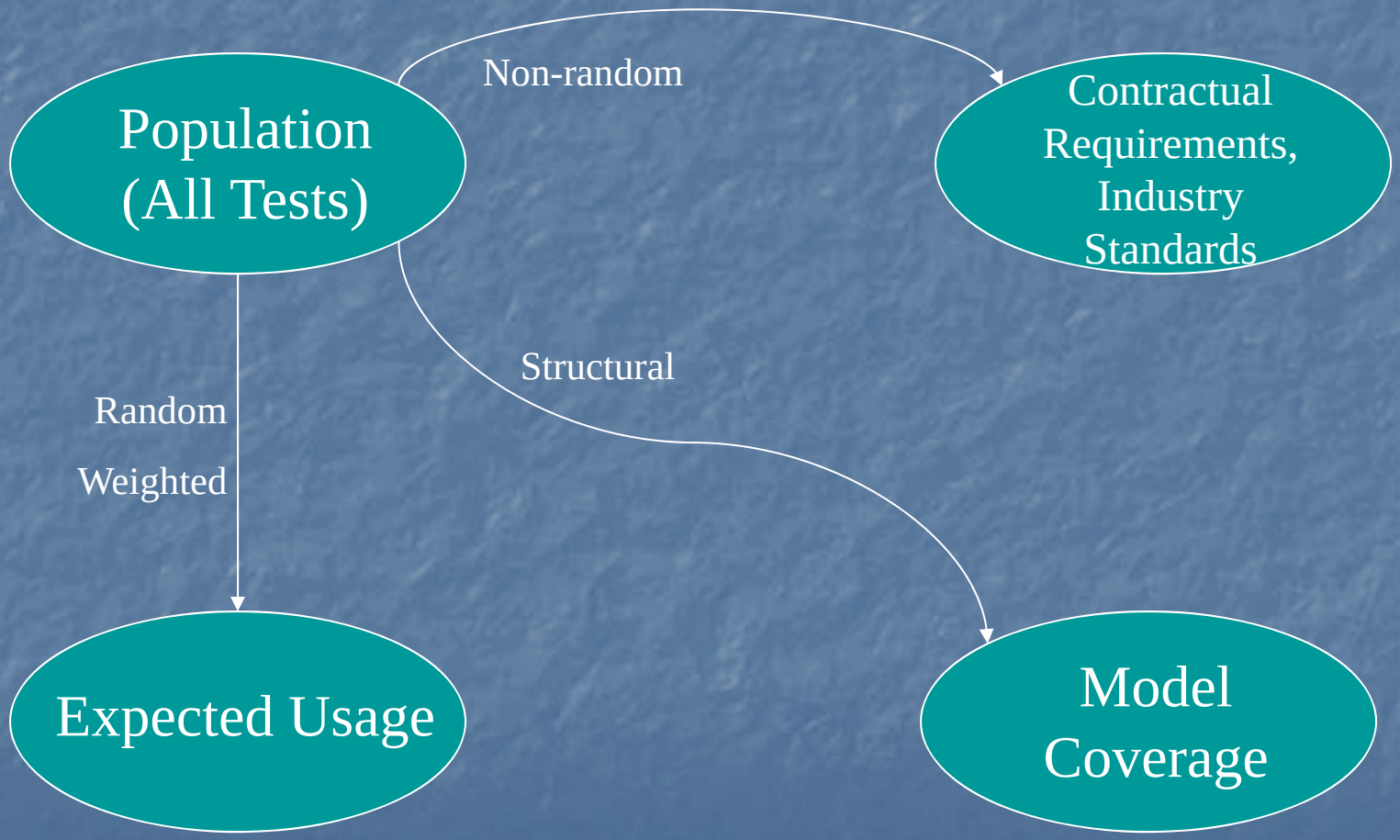
Model Constraint Examples

- One transition is a function of another transition
  e.g.: prob (GoodDigit) = 3 (prob (BadDigit))
- A probability is within a range
  e.g.: .1 < prob (BadDigit) <.3

Software Testing Process



Sampling Options



Nonrandom Testing

- Coverage tests
  (cover all arcs at the least cost of testing)
- Mandatory tests
  (policy, contract, regulatory, safety, security)
- Regression tests
- Importance tests
  (generate tests in order of probability or cost)

Random sample testing

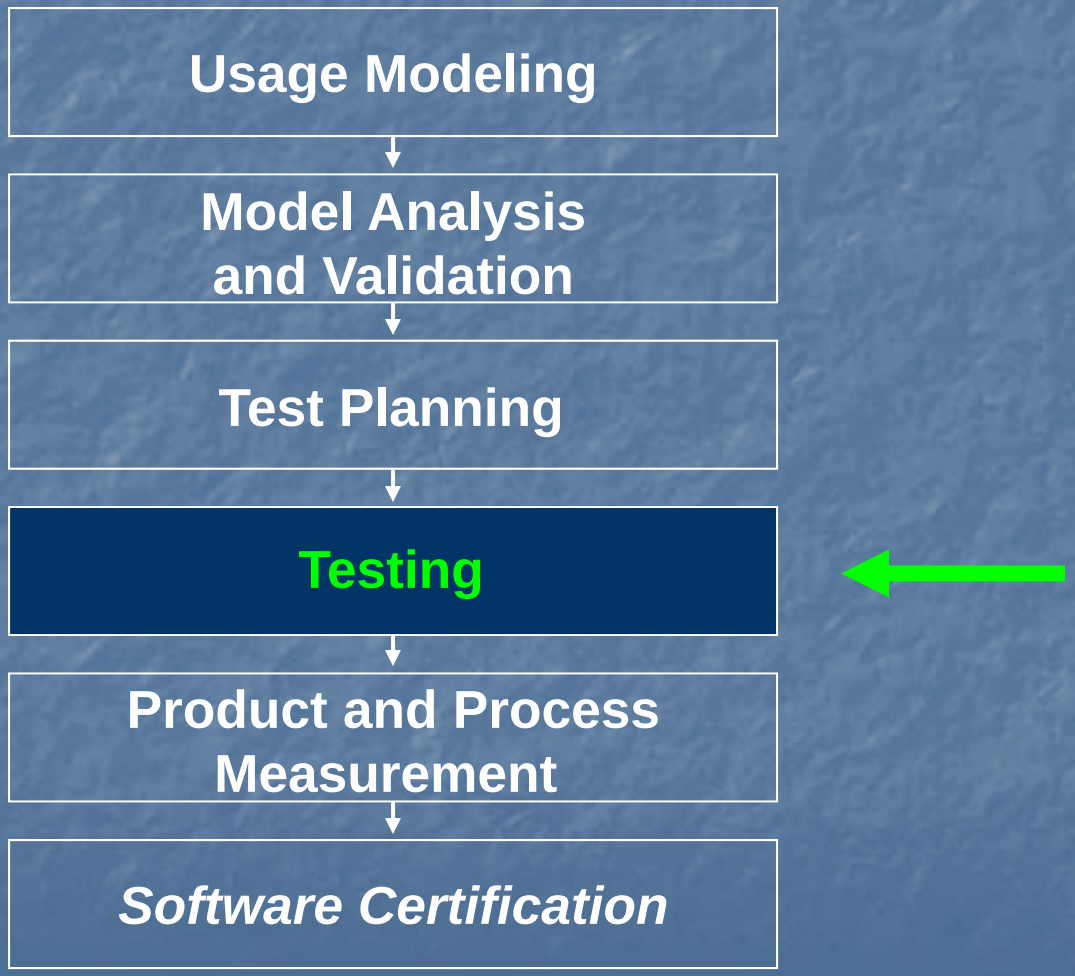
- Random test cases are generated from the usage model.
- Random testing permits statistical analysis of the sample and generalization to the population.
- Each test case is a sequence of stimuli, and random test cases may be reused.

Testing Scripts

Script commands are attached to arcs and give the instructions for testing the transition:

- Manual testing
  - written instructions
  - data to use
  - items to check
- Automated testing
  - commands for testing equipment
  - commands for testing software
  - statements in a programming language

Software Testing Process

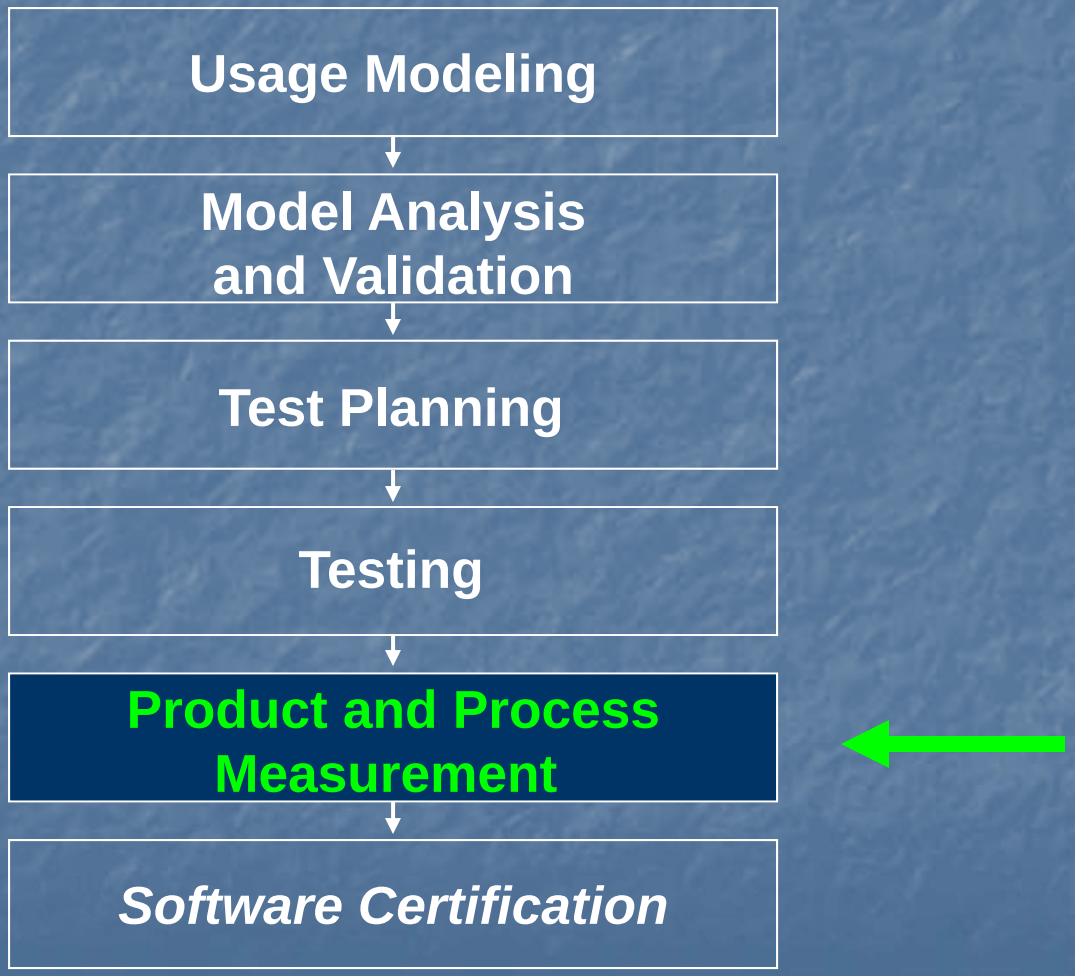


Testing

Record all failures by test case number and transition number where failure occurs, then:

- Estimate reliability based on testing experience
- Evaluate stopping criteria

Software Testing Process



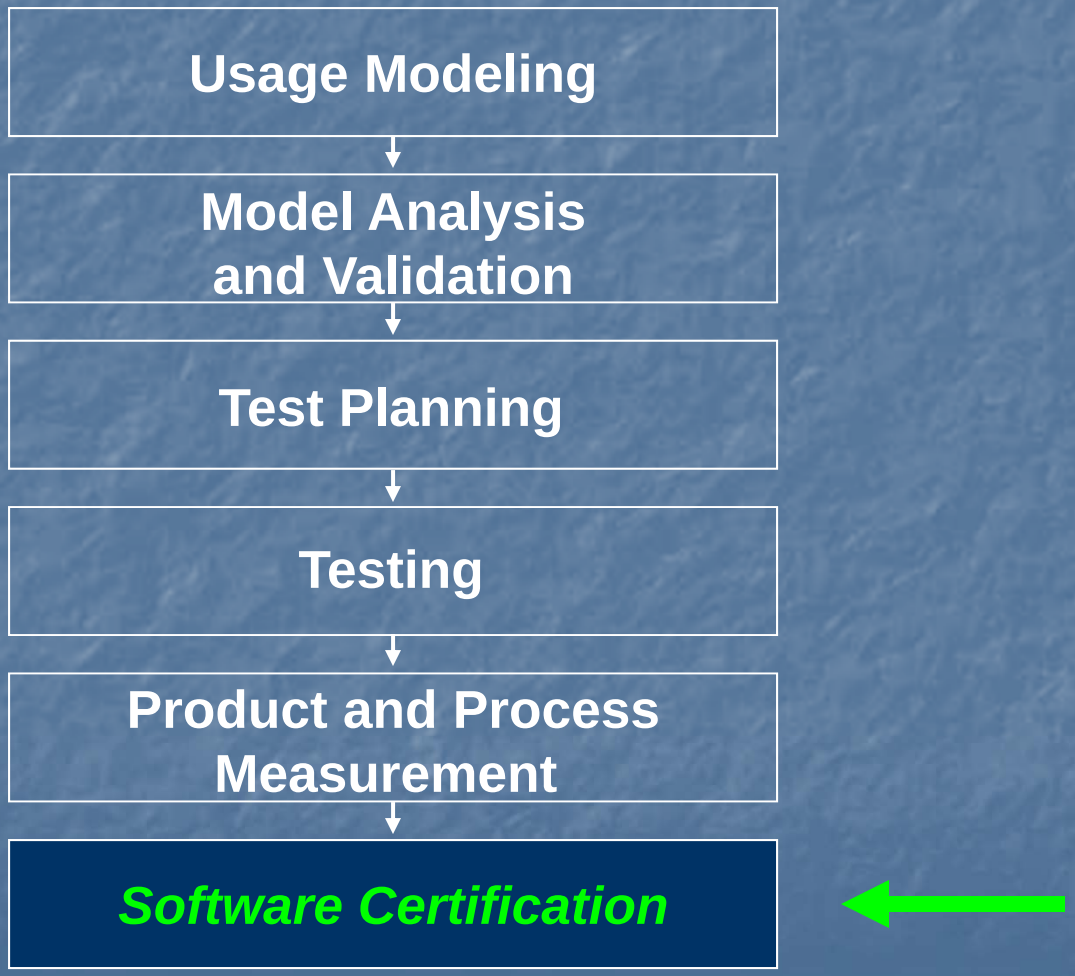
Process Measurement

- Statistical testing supports incremental development with feedback on the development process.
- Reliability growth modeling can be applied to incremental development.
- Reliability estimates can be used to project product support effort.

Reliability Measurement

- Test case pass/fail statistics give reliability and confidence based on binomial distribution.
- Bayesian models
  - provide reliability estimates regardless of whether failures are observed
  - allow use of prior reliability information

Software Testing Process



Software Certification

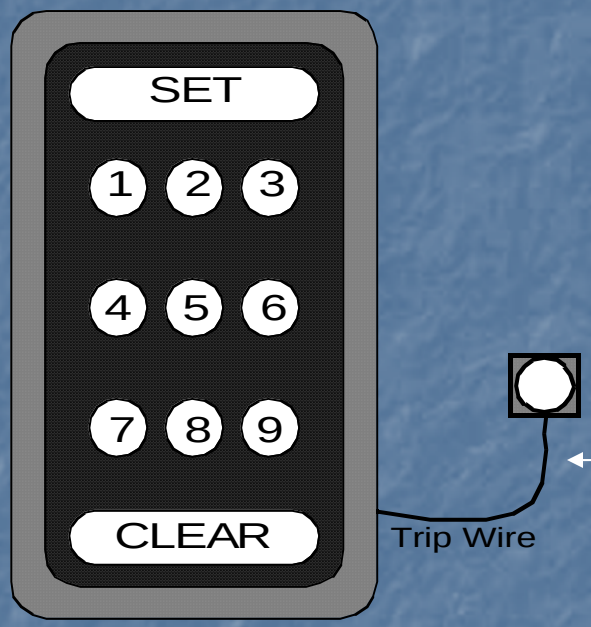
- Certification establishes product conformance with well-defined standards.
- Product certification requires a process that is independently repeatable within statistical variation.
- Statistical testing supports quantitative certification through statistical characterization of system use and reliability.

EXAMPLE:
The Security Alarm

Security Alarm Stimuli

SET activates device

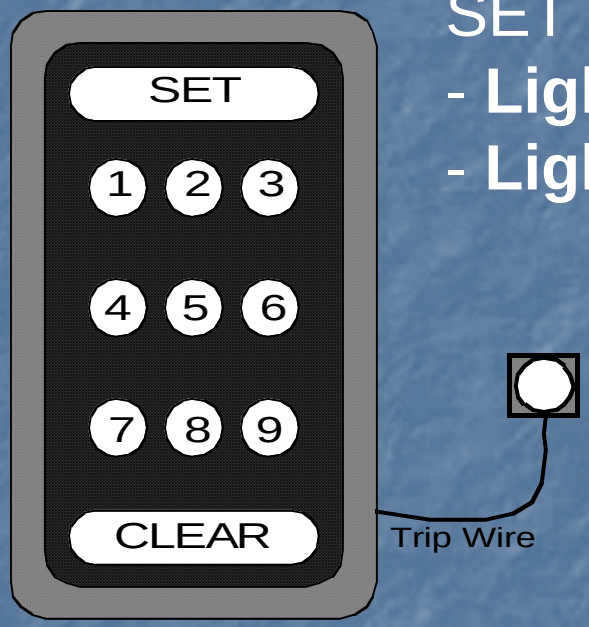
A 3-digit code deactivates device, so possible inputs: **GoodDigit, BadDigit**



Trip activates alarm

CLEAR erases all digits that have been entered

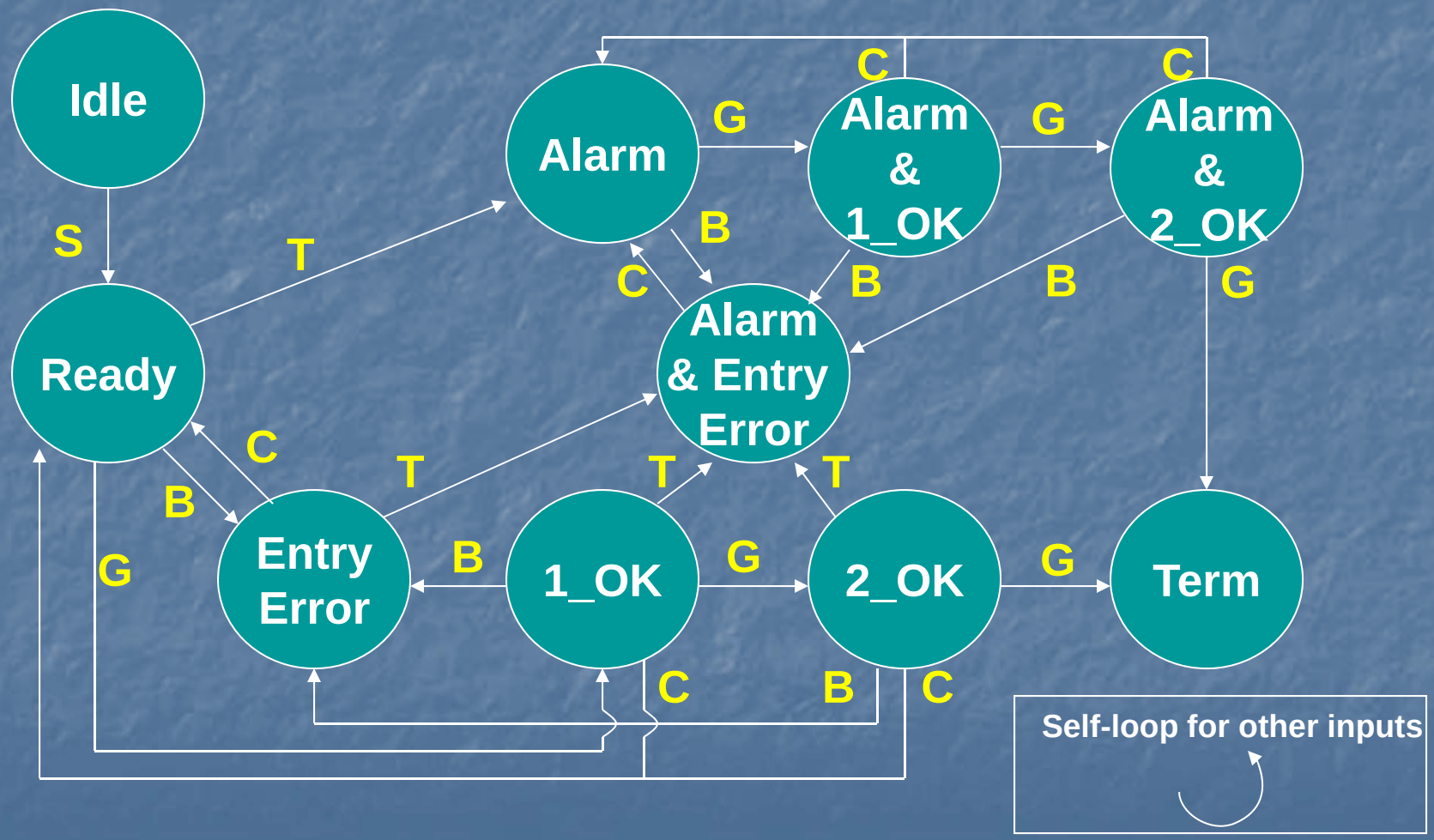
Security Alarm Responses



SET button
- **Light on** when device is activated
- **Light off** when correct code entered

Alarm siren
- **Alarm on** when tripped
- **Alarm off** when correct code entered

Security Alarm Model Structure

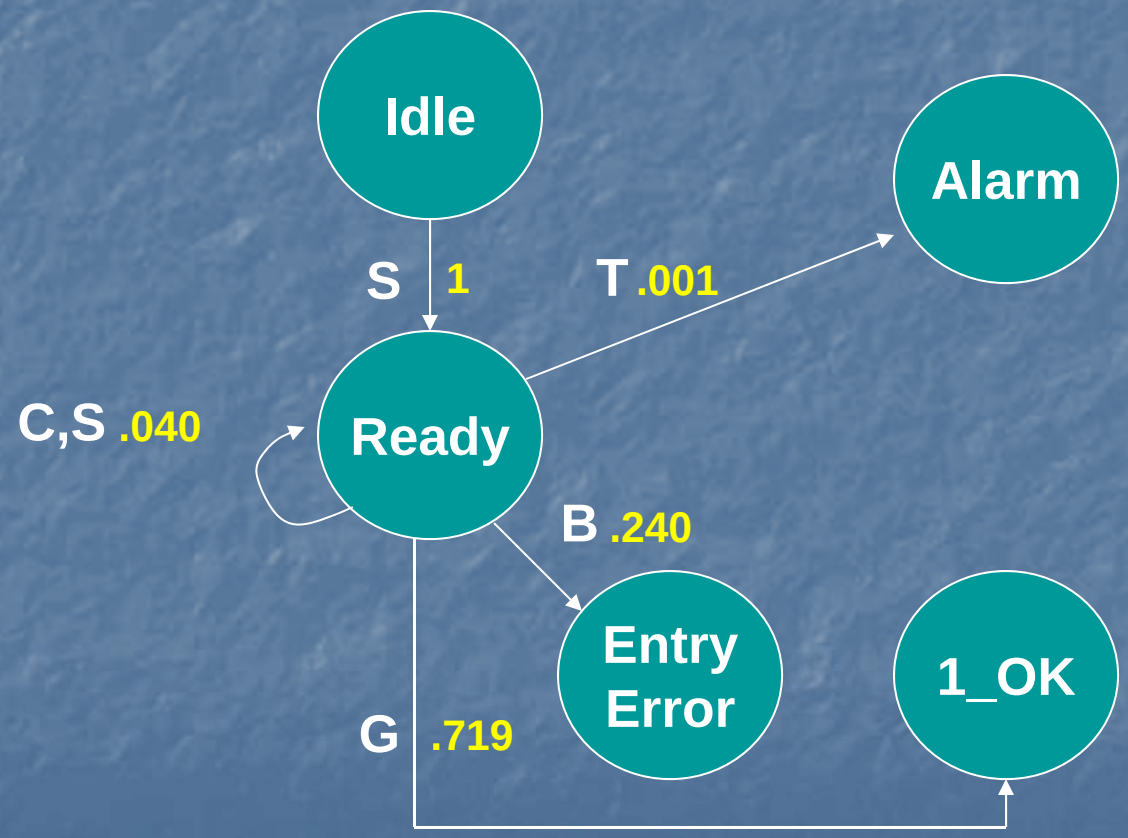


Graph: Possible Use
Arc Probabilities: Expected Use

- The structure of the usage model (the graph) represents all possible uses of the software.
- Transition probabilities on the arcs represents the expected use.

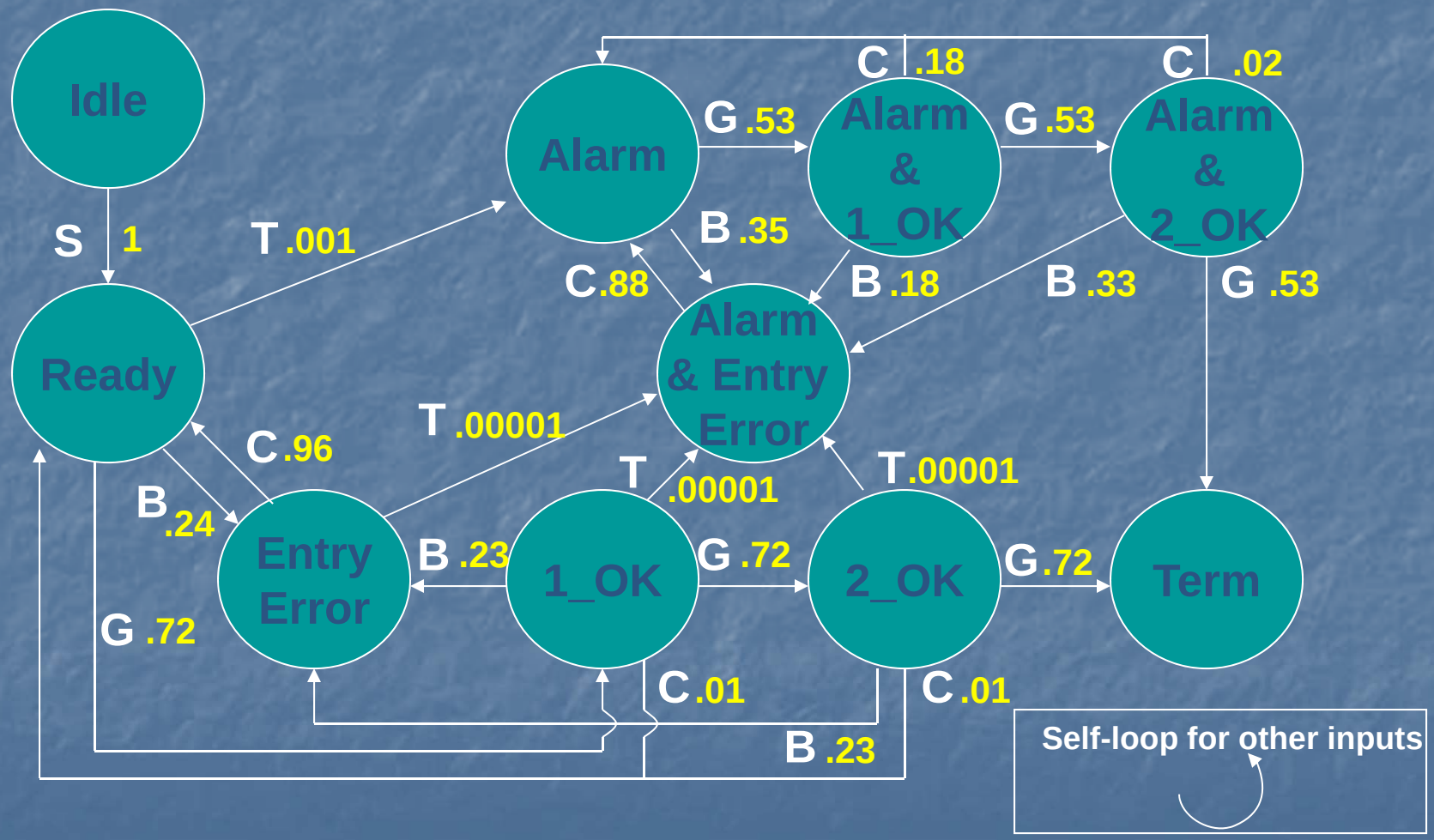
Transition Probabilities on *Exit Arcs* sum to 1

Probabilities come from historical data or engineering judgment



.719 G
.240 B
.040 C,S
.001 T
1.000

Security Alarm Usage Model



Operational Profile from Markov Analysis

State	Long run occupancy
Idle	.109597
Ready	.270206
Entry Error	.152519
1_OK	.202452
2_OK	.151837
Alarm	.001447
Alarm & 1_OK	.000868
Alarm & 2_OK	.000521
Alarm & Entry Error	.000956
Term	.109597

Most likely state

Least likely state

These results are validated against data and expert opinion.

Operational Profile from Markov Analysis

FROM STATE	Arc	Long run occupancy	
Idle	Set	.12308685	
Ready	BadDigit	.07275884	
	GoodDigit	.21827653	← Most likely arc
	Set, Clear	.01212647	
	Trip	.00030316	
Entry Error	Clear	.16443191	
	Set, BadDigit, GoodDigit	.00685811	
	Trip	.00000171	
… 2_OK	Trip	.00000170	← Least likely arc

These results are validated against data and expert opinion.

Operational Profile from Markov Analysis

Stimulus	Long run occupancy
Set	0.13900261
GoodDigit	0.50644176
BadDigit	0.16537277
Clear	0.16937841
Trip	0.00030885
All self-loops	0.0195

Most likely stimulus

Least likely stimulus

These results are validated against data and expert opinion.

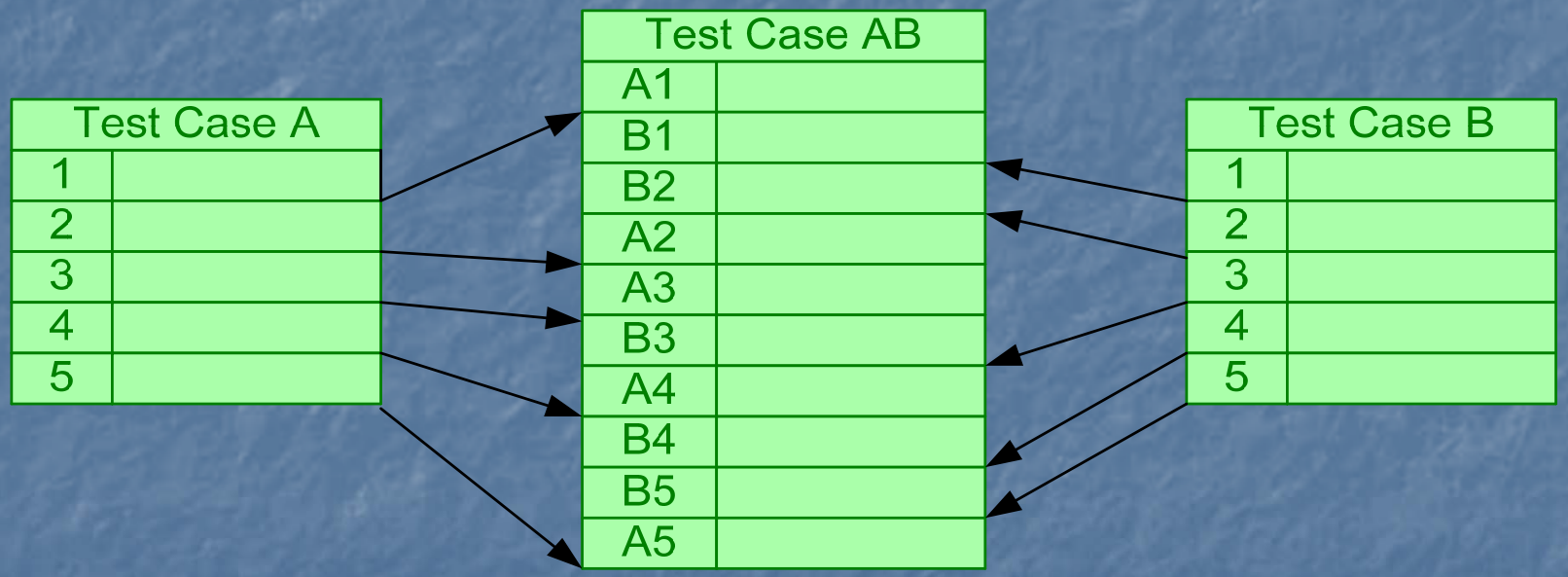
Statistical Tests

- Random tests
  random scenarios based on transition probabilities
- Importance tests
  scenarios that traverse “most important” arc sequences, where importance may be related to cost, time, probability, etc.

Sample Security Alarm Test Script

```
# ========================================
# Trajectory: 0
# Model:     Security Alarm
# Key:
#
# Events:    14
# ========================================
# Step: 1, Trajectory: 0
[S0]."S"
# Step: 2, Trajectory: 0
[S1]."G"
# Step: 3, Trajectory: 0
[S3]."B"
...
# Step: 13, Trajectory: 0
[S6]."G"
# Step: 14, Trajectory: 0
[S7]."G"
```

Test Case Interleaving



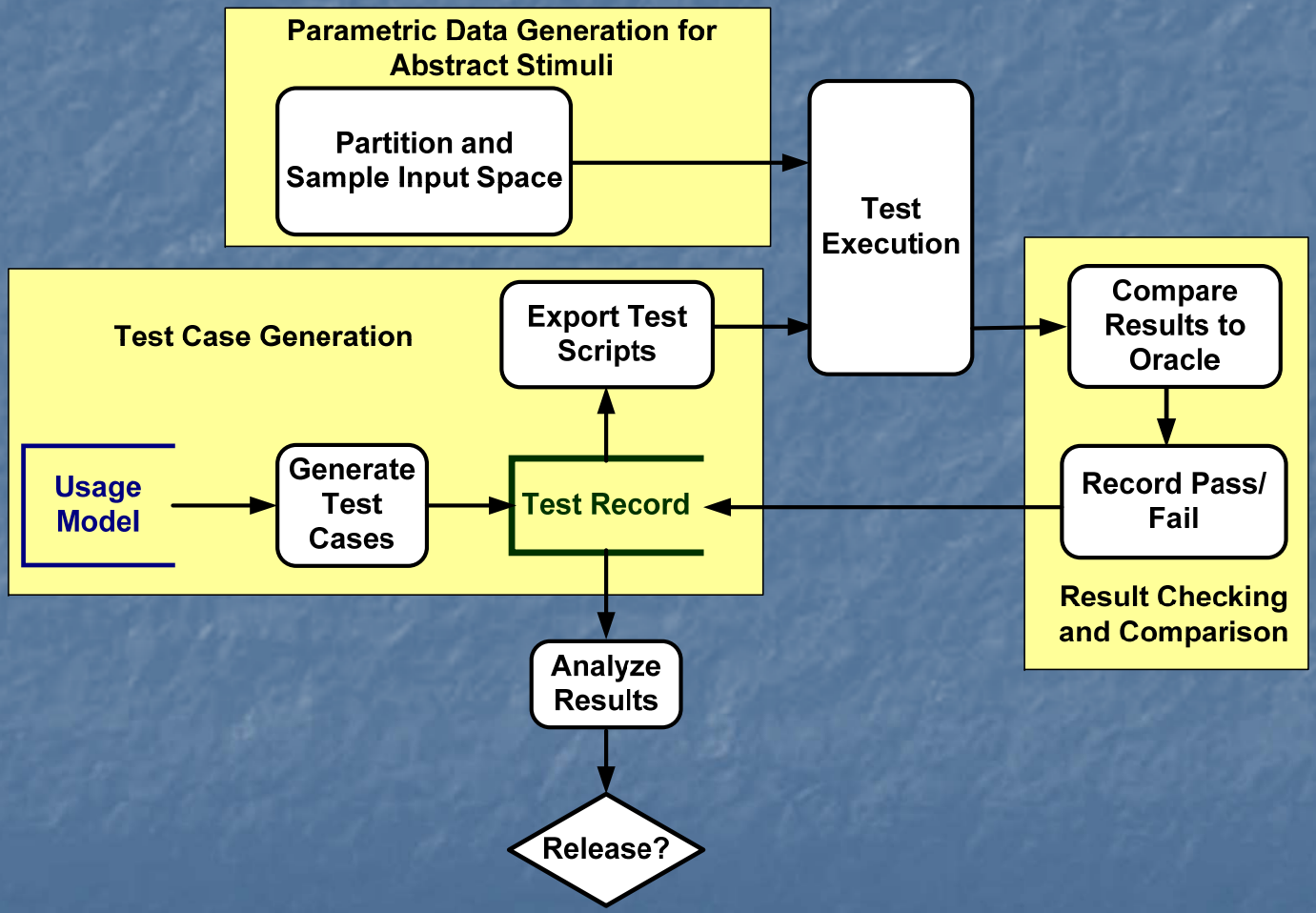
Interleaving Uses

- Servers to Multiple Clients
- Multiplexed Input Streams
- Non-modal GUI

Test Automation

- Many automated test tools already exist
- Convert test cases to test scripts which can be executed by a test tool
- Associate test instructions with arcs and states in a model

Test Automation



Test Oracle

- Must be able to decide pass / fail for each test.
- Automation techniques:
  - Build oracle into test script
  - Use self-checking data
  - Use known cases
  - For some cases, let a human decide

Test Record Management

- Record pass/fail information for each test case
  - Specify arc(s) where failure occurred
  - Specify if and where test case aborted

Security Alarm Test Record

SecurityAlarm_1.str,5,19

On the test case 1 for model SecurityAlarm, failures occurred on events 5 and 19, but testing continued to the end of the test.

SecurityAlarm_2.str,17,21,S

On the test case 2 for model SecurityAlarm, failures occurred on events 17 and 21, and testing stopped.

Test Results Analysis

- State/Stimulus/Arc coverage
- Stimulus/Arc reliability
- Single use reliability

Benefits of Model Based Testing

- Better Product
  - Clearer requirements, improved specification
- Better Use of Resources
  - Optimization of testing strategy
  - Reusable assets: models, test plans, scripts, test cases
- Shorter Life Cycle
  - Test planning done in parallel with development
  - Easier test automation
- Better Management
  - Quantitative support for management decisions
  - Quantification of expected field reliability
  - Clear path to continuous process improvement