Software Quality Research Laboratory

SQRL

Software Production Essentials

Beyond the Buzz Words

Tom Swain
Software Quality Research Laboratory
University of Tennessee
Department of Electrical Engineering and Computer Science

SQRL

Why Process?

- Quality
  - Maximize Customer Satisfaction
  - Minimize Rework and Repair
- Productivity
  - Optimize Production Cost
  - Shorten Time to Market

This is not a tradeoff. Quality is Free.

Post ~2000 Scientific Computation
What's Different?

- Collaboration, Collaboration, Collaboration
- Results affect national policy NOW (*e.g.*, climate models)
- Results can have major economic impact SOON (e.g., materials, energy, and IT infrastructure)

SQRL

Process Objectives in a Scientific HPC Environment

- Produce reliable software for community use
- Implement functionality tailored to user needs and expectations
- Maximize resource commitment to scientific innovation and productivity
- Minimize resources required for rework and maintenance
- Meet regulatory SQA requirements, where applicable

SQRL

Basic Process Elements

- Requirements Definition
- Coding
- Release

Everybody does it, but the order may vary.

SQRL

How Do We Fill the Gaps?

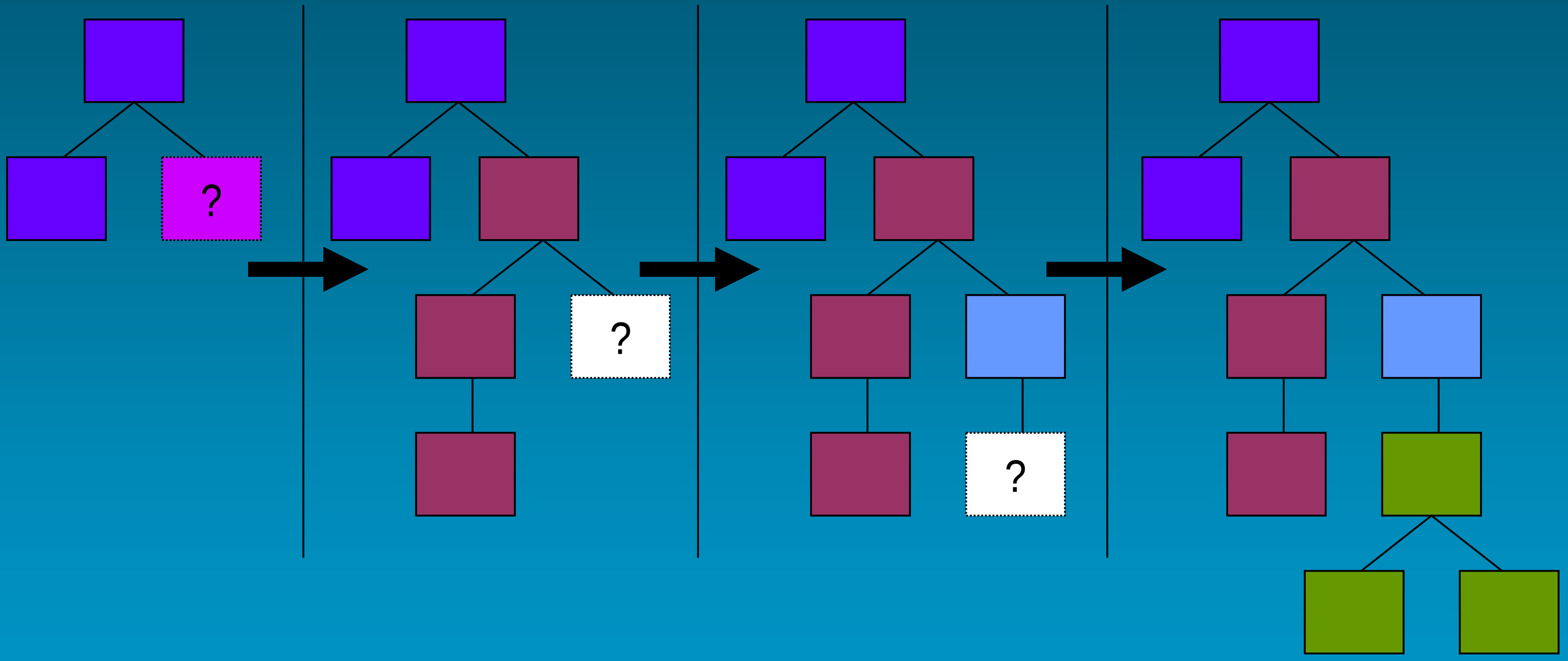
- Requirements Definition
- Specification?/Design?/Verification?/?
- Coding
- Test?/ Inspection?/?
- Release

No shortage of lifecycle/process candidates.

SQRL

SQRL

Software Production is Incremental



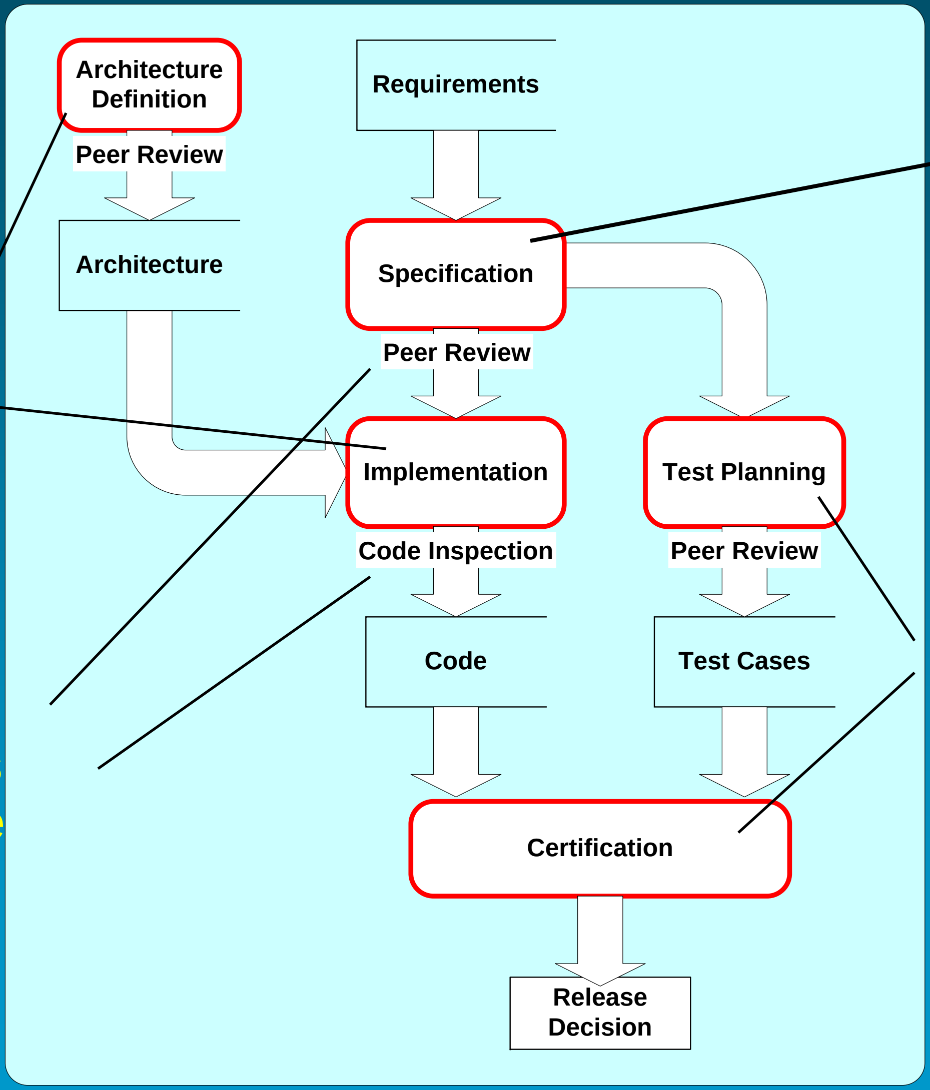
An efficient, repeatable process is necessary for expanding, changing requirements.

Software Production Keys

SQRL

(2) Mapping specified code to a robust, maintainable architecture.

(3) Peer review of key work products including code inspection.



(1) A seamless, traceable transformation from requirements to behavior specification.

(4) Certification via quantitative testing.

Behavior Specification Objectives

- **Completeness**
  - a response is defined for every stimulus history
- **Consistency**
  - each stimulus history maps to only one response
- **Correctness**
  - the specification is explicitly traceable to the requirements

SQRL

SQRL

Requirements

- Individual requirements tagged for traceability.
- Initial requirements assumed to be incomplete, inconsistent, and possibly incorrect.

SQRL

The Simple Case: Static Calculations

(Things that run to completion without user interaction)

1. Partition input space into domains bounded by discontinuities
2. Specify response function for EVERY domain

SQRL

Function Specifications

- Describe function mathematically for each distinct region of input space
- Specify correspondence between program variables and math symbols used
- Include responses to invalid inputs
- Specify all function results - returned values, state variable modification, modified globals

Practical Consideration: Embed Static Specs in Code Using Doxygen, JavaDoc, or equivalent

Function Specifications: Arguments and Return Values

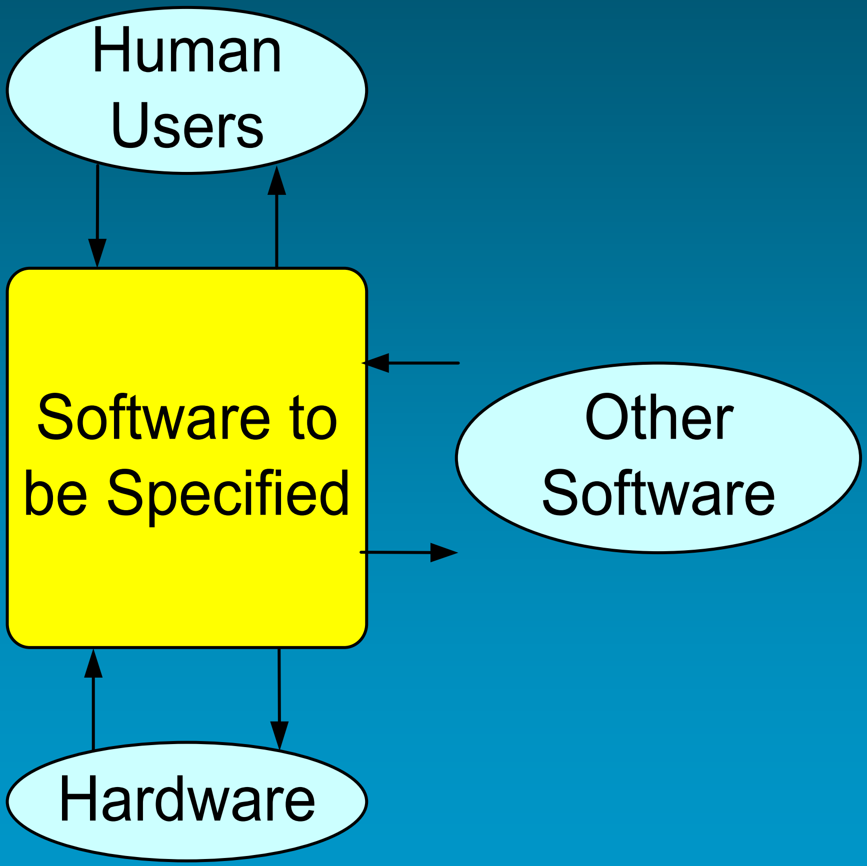
- Scalars and arrays of fundamental types
  - Specify type, definition, units, valid range, and default value
- Pointers to fundamental types
  - Same as above for dereferenced values
- Compound types – classes, templates, etc.
  - Specify recursively by providing above info wherever data members of fundamental types are declared

SQRL

The General Case: Stateful Systems (GUIs, datacom, control, etc.)

- Establish system boundary in terms of human/software/hardware interfaces.
- Itemize stimuli.
- Itemize responses.
- Perform enumeration of stimulus sequences.
- Perform canonical sequence analysis.
- Generate state machine specification.

System Boundary and Interfaces



SQRL

Important Definitions

- Stimulus - an event resulting in information flow from the outside to the inside of the system boundary
- Output – externally observable item of information flow from inside to outside the system boundary
- Response – occurrence of one or more outputs caused by a stimulus

SQRL

Enumeration Mechanics

- For each stimulus sequence of length n:
  - If illegal, mark it illegal and do not extend further.
  - Document correct response based on requirements.
  - If no requirement found, create derived requirement.
  - Record requirements trace.
  - Check for equivalence with previous sequences.
- Extend only those sequences that are not illegal or equivalent.
- Continue until all sequences of a given length are illegal or equivalent to previous sequences.

SQRL

Canonical Sequence Analysis

- Identify canonical sequences – all legal sequences not equivalent to earlier sequences.
- List the canonical sequences in the order enumerated.
- Define state variables such that each canonical sequence corresponds to a unique state vector.

Canonical sequences define all system states

SQRL

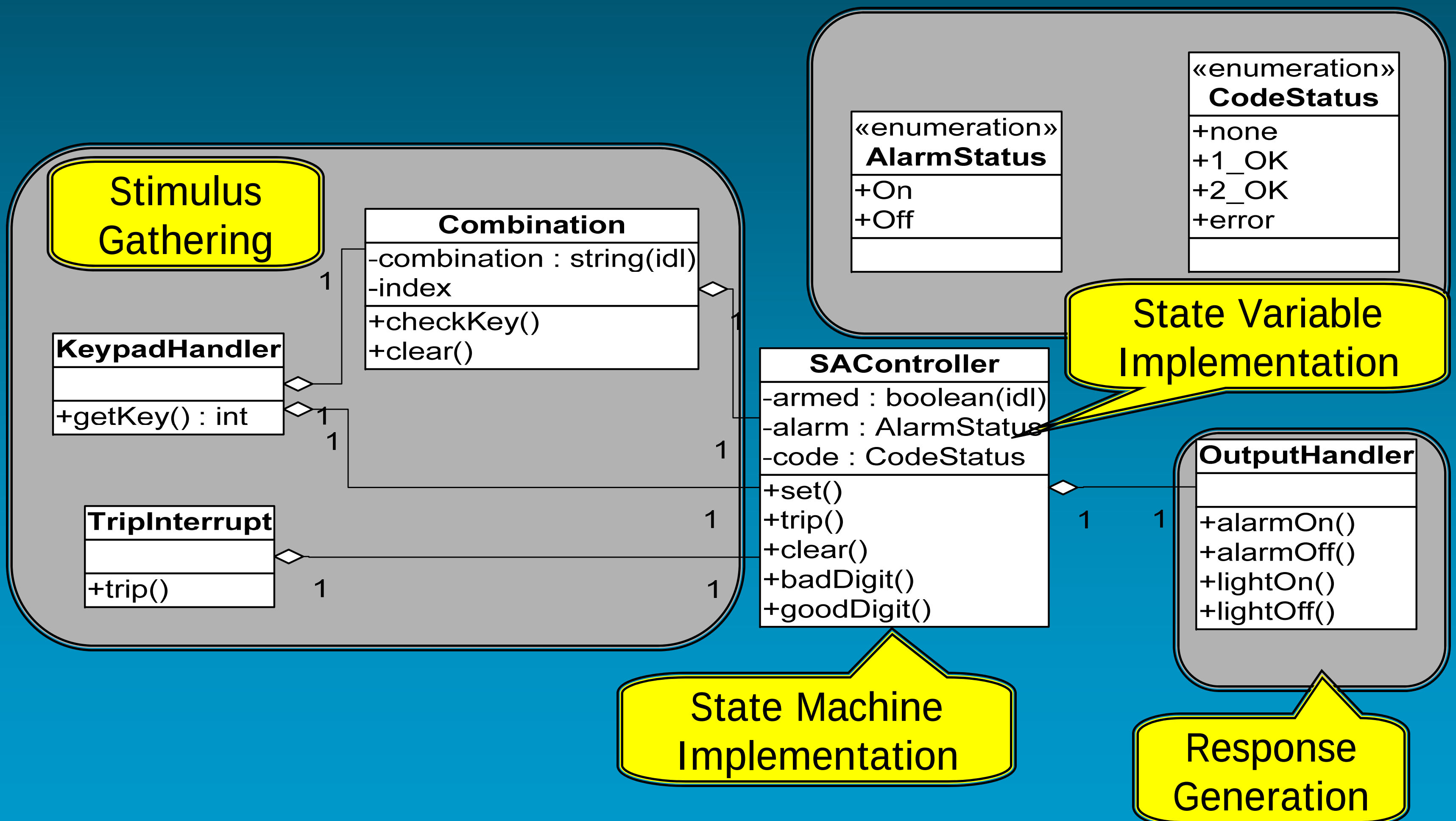
State Machine Generation

- For each stimulus
  - For each canonical sequence (CS)
    - Get state variable values
    - Find sequence (CS+stimulus) in enumeration
    - Get response
    - Get new CS (CS+stimulus or its equivalence)
    - Get new state variable values

SQRL

Map Code Specified in Behavior Specification to Architecture

SQRL



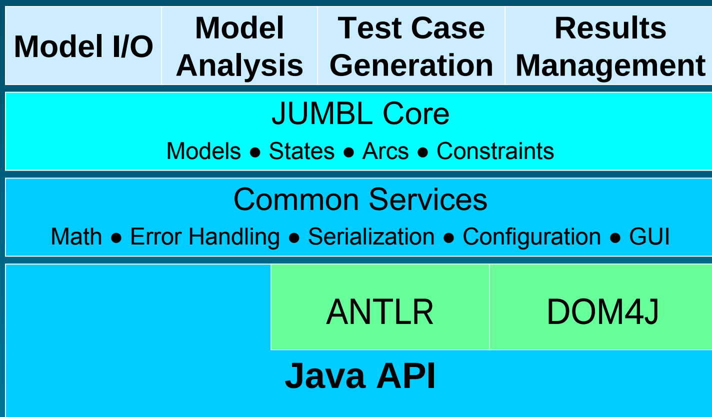
SQRL

Specification Procedure Summary

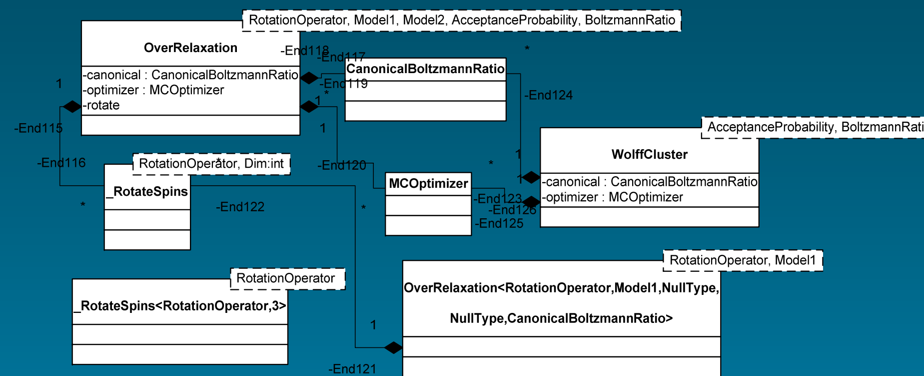
- Specification must be complete, consistent, and traceably correct
- Partition system into manageable components for scalability
- Use enumeration to discover and correct ambiguity and omissions in requirements
- Completed enumeration converts to state machine specification
- Map stimulus gathering, response generation, and state machine spec to architecture for code generation

High Level Architecture Examples

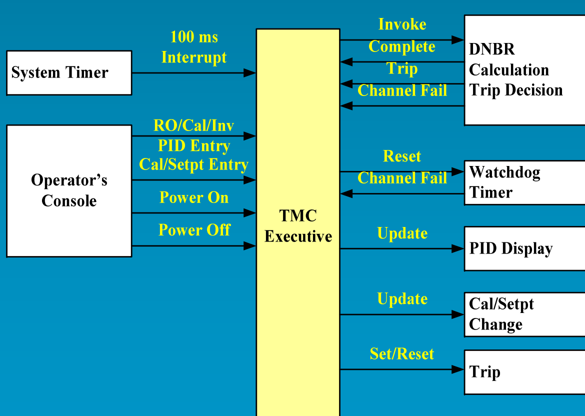
SQRL



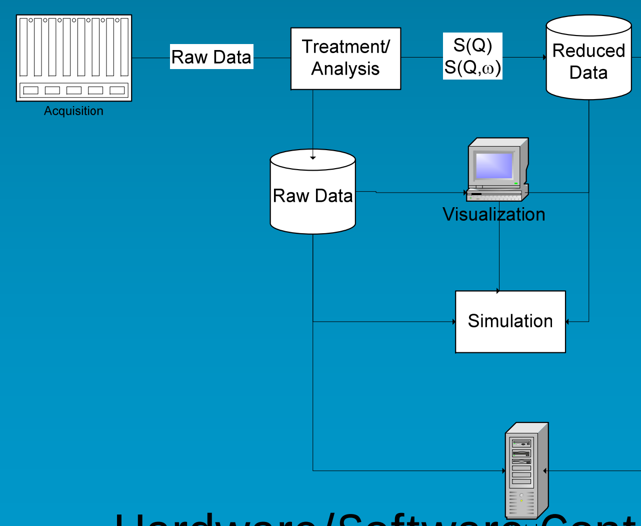
Library



Computational Component



System Abstractions



Hardware/Software Context

SQRL

Architecture Specification Contents

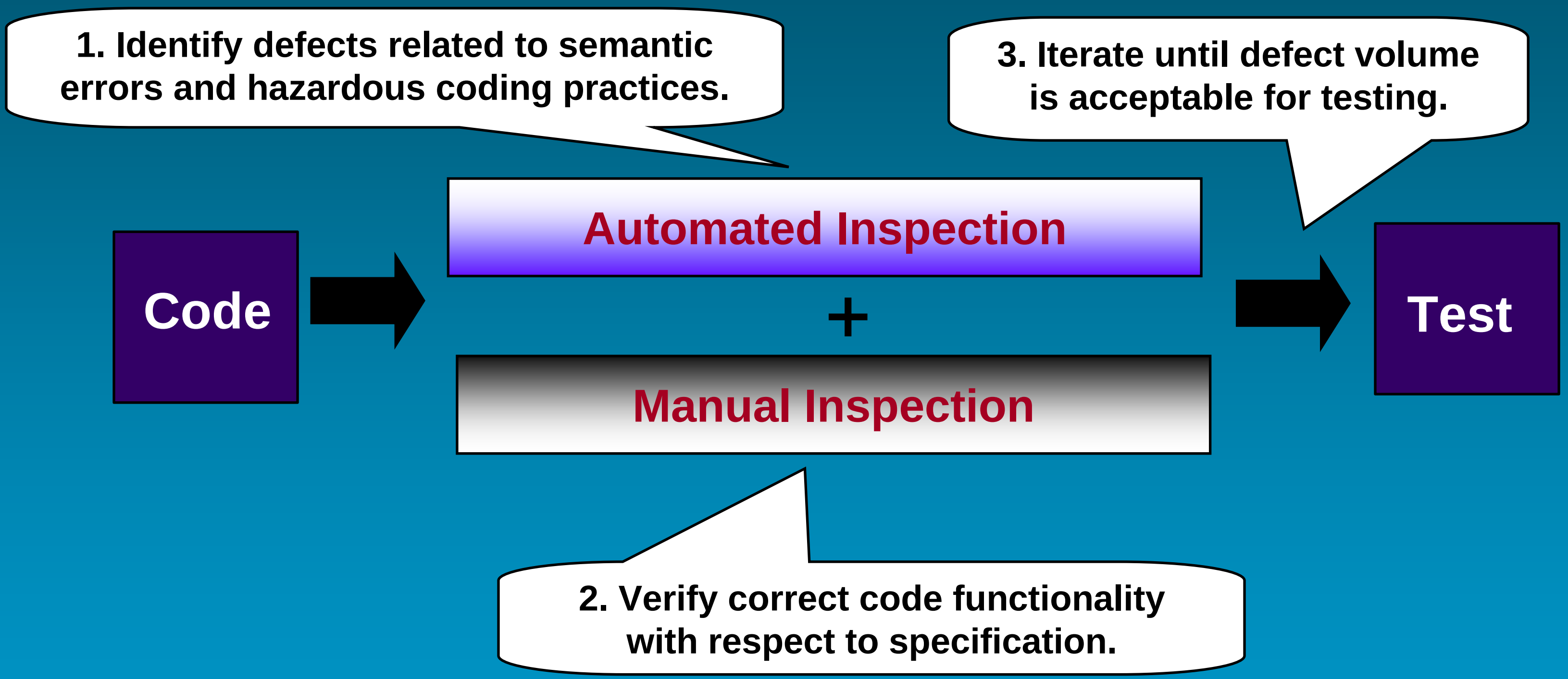
- Define components and their responsibilities
- Specify relationships among components
- Specify intra-element and external interfaces
- Ensure unidirectional use hierarchy
- Specify assumptions regarding platform/environment

SQRL

Independent Peer Review

- Domain Expert Review
  - Initial Requirements
  - Derived Requirements from Specification
- Development Team Peer Review
  - Architecture Specification
  - Test Plan
- Code Inspection
  - Automated Enforcement of Coding Standards
  - Manual Verification of Functional Correctness

Manual and Automated Code Inspection

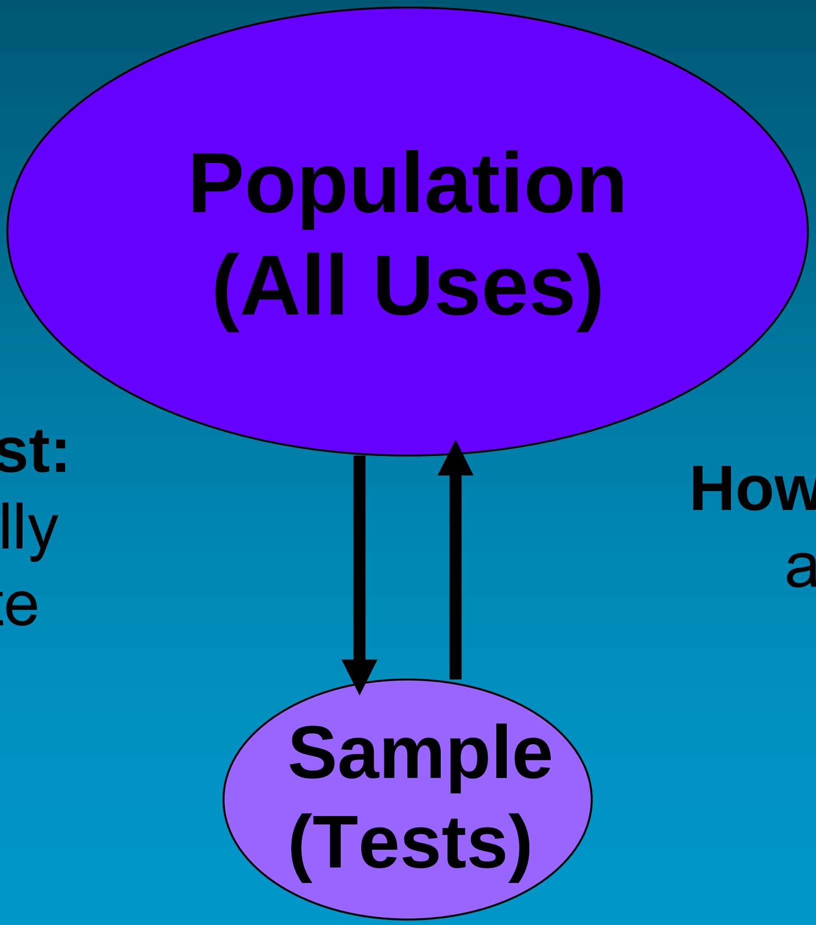


SQRL

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.

Testing is Always Sampling

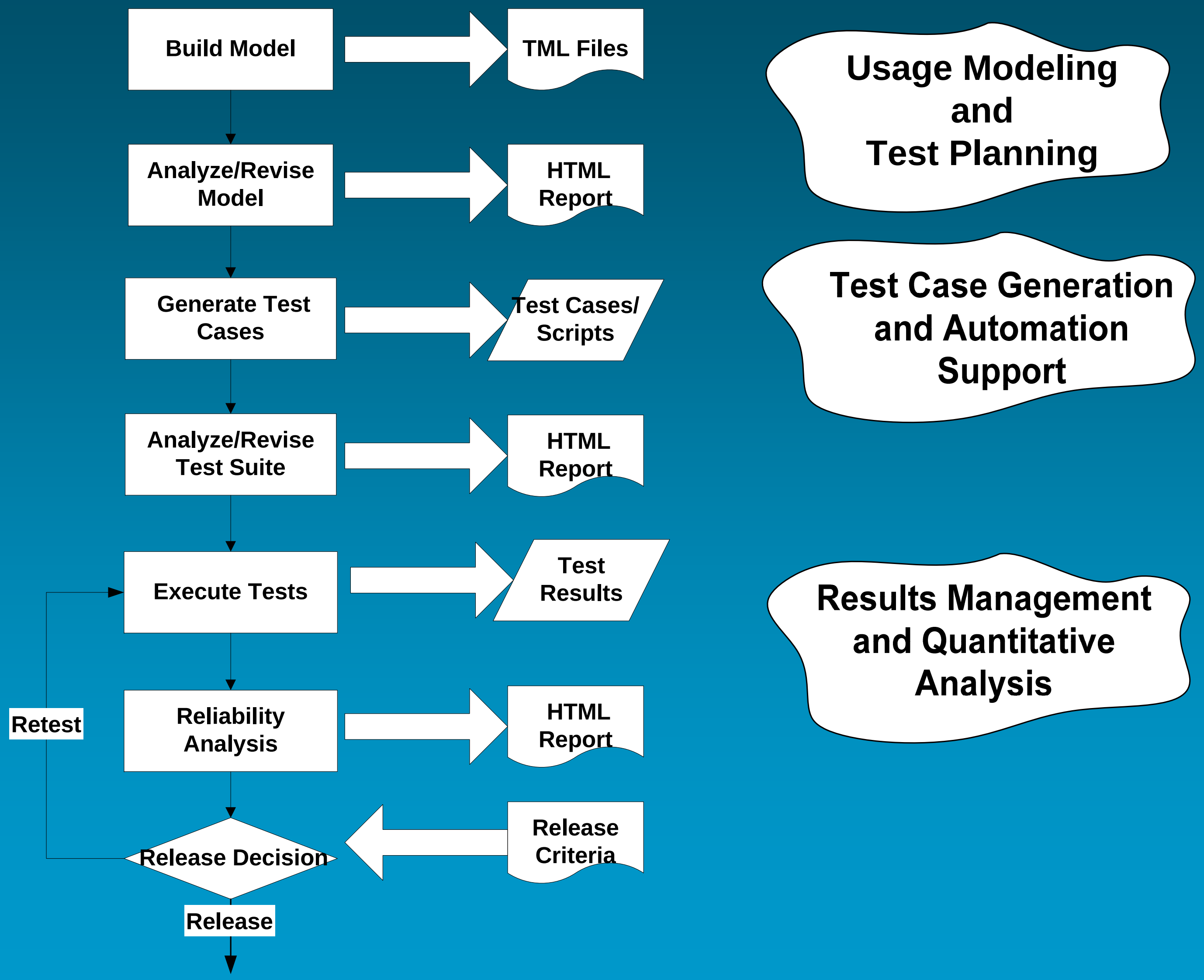


What to test:
a statistically appropriate sample

How much to test:
a risk/benefit tradeoff

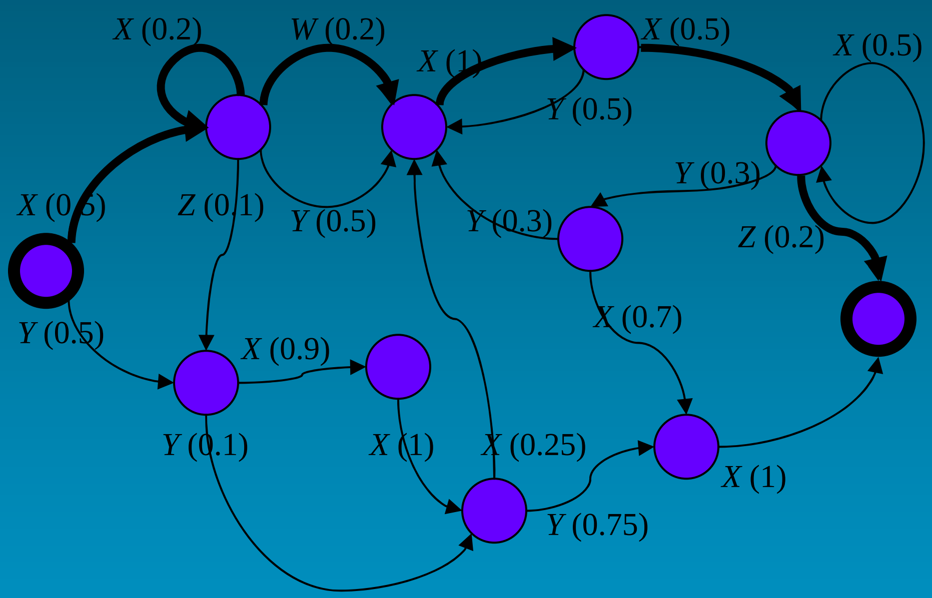
SQRL

Model-Based Statistical Testing (MBST)



SQRL

The Population of All Uses Is Represented by a Markov Chain Usage Model



$\Pr[\,x \quad x \quad w \quad x \quad x \quad z\,] = 0.0002$

- Nodes represent “states of use”
- Arcs represent stimuli/events
- Probabilities represent likelihood of a stimulus, given the current state.

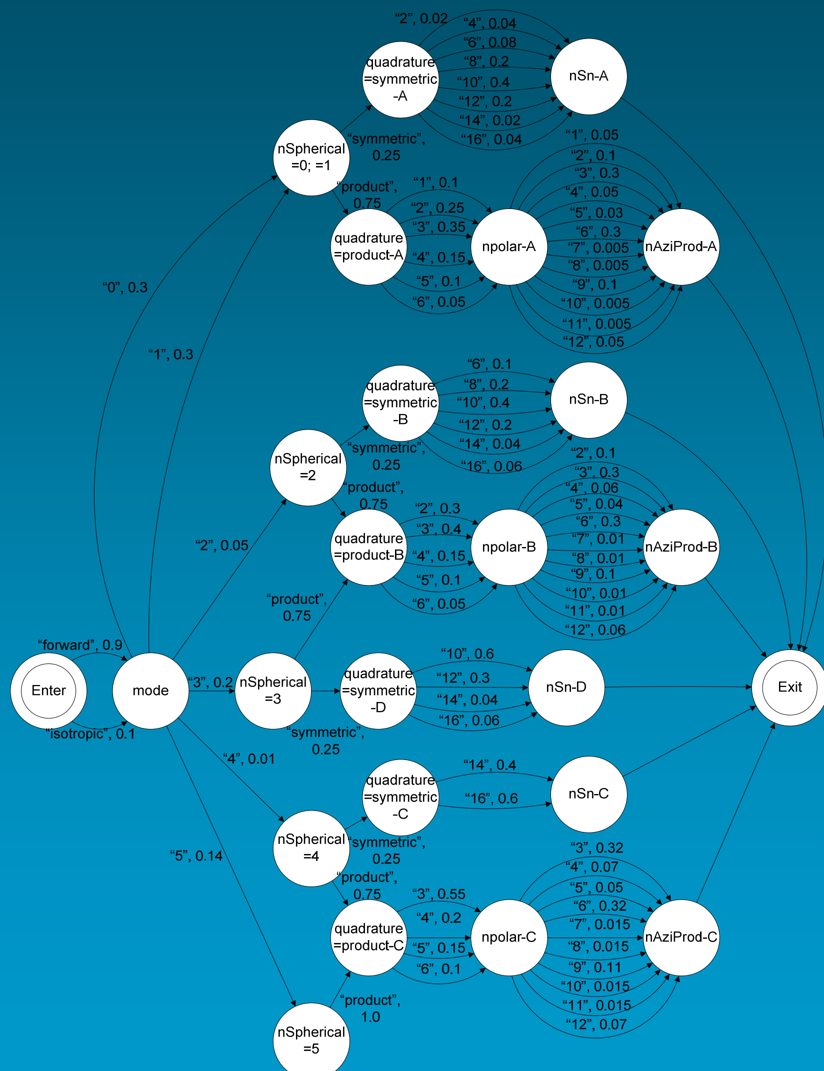
A “use” (or test) is any path from the source to the sink.

Special Case: Static Computation with Large Input Space

- Partition input space via abstractions
- Model input selection
- Provide specific parametric input at run time
- Probability weighted generation can give all possible input combinations after partitioning
- Test oracles
  - diverse implementations
  - constraints based on science
  - interpolate between benchmark points
  - favor clarity over performance
  - incentive for good specifications

SQRL

For Static Computation Model Input Selection



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
- Describes "use" of product and not the product itself

SQRL

Useful Properties Available via Markov Analysis

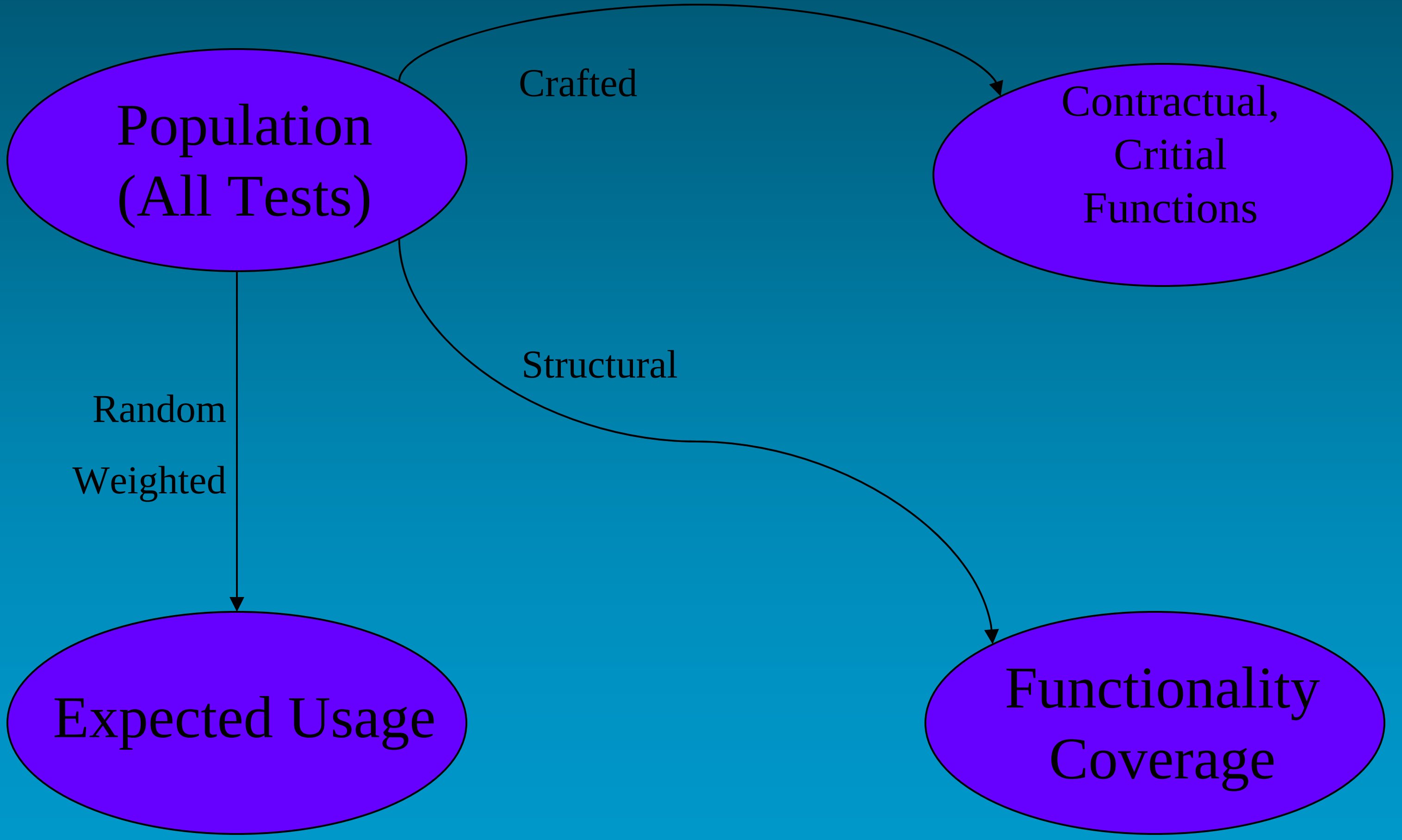
- Expected Test Case Length: average number of stimulus events from start to end
- Arc/Stimulus Occupancy: fraction of all transitions performed by each Arc/Stimulus
- State/Arc Probability of Occurrence: probability a State/Arc will be visited during a single use
- State/Arc Visits per Test Case: Average number of visits to each State/Arc per use
- Mean First Passage: average number of test cases required to exercise a particular state/function

SQRL

Model Revision and Validation

- Analytical results are inescapable, given the model.
- If results do not square with what is known of the real-world application, the model must be revised.
- Continue the analyze-revise cycle until the model is an acceptable description of use of the system.

Sampling Options (Test Case Generation)



SQRL

Nonrandom Testing

SQRL

- Coverage tests
  (cover all arcs at the least cost of testing)
- Importance tests
  (generate tests in order of probability or cost)
- Crafted tests
  (contractual, safety issues, critical functions)

Random sample testing

- Test cases are generated by random walks through the usage model.
- Permits statistical analysis of the sample and generalization to the population of uses.
- Each test case is a sequence of stimuli and random test sets may be reused.

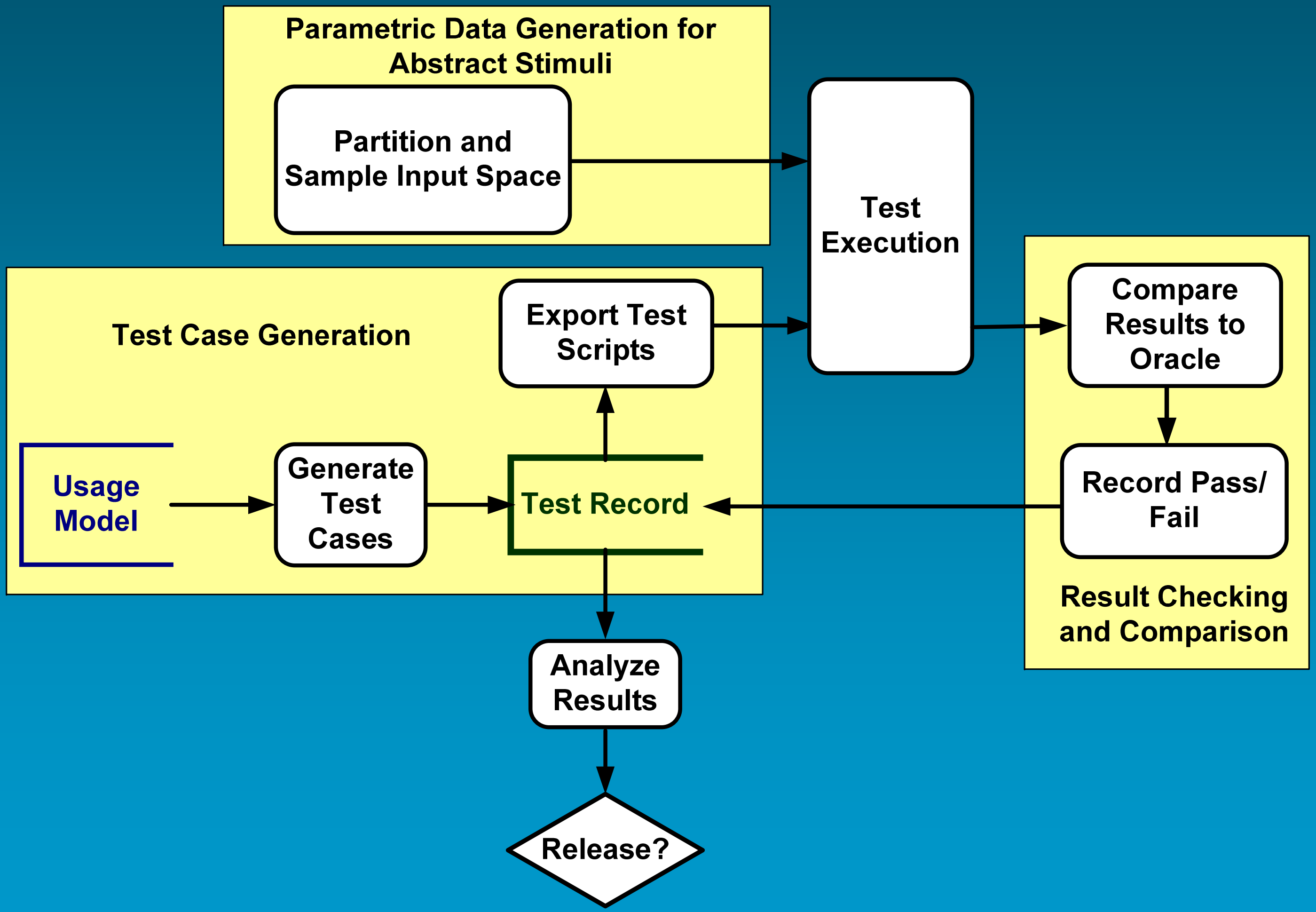
SQRL

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
  - signals for testing equipment
  - commands for driver software (*e.g.*, X-runner)
  - statements in a programmed test driver

Test Automation



SQRL

Test Automation

- Application-Specific Tools
  - Generate parametric data for abstract stimuli
  - Compute expected results
  - Make pass/fail determination
- Generic Tools
  - Generate test cases as test scripts
  - Associate test instructions with arcs and states in a model
  - Perform statistical analysis of test results

SQRL

Test Results

- Record failures by test case and transition
- Estimate reliability based on testing experience
- Evaluate stopping criteria

SQRL

Reliability Estimation

- 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

SQRL

SQRL

MBST Benefits

- 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 release decisions
  - Quantification of expected reliability
  - Measurement tool for continuous process improvement

Essential Process Elements

- Requirements Definition
- Robust Architecture Definition
- Rigorous Specification
- Domain Expert and Peer Review
- Coding
- Code Inspection
- Statistical Testing/Certification
- Release

Fill the gaps with rigorous engineering practices.