SAND2008-3346P

A Code Verification Checklist

P. Knupp and C. Ober

Software Engineering Seminar Series
May 13, 2008

ASME definition of Code Verification:

activities that “establish confidence, through the collection of evidence, that the mathematical model and solution algorithms are working correctly.”

ASME Guide describe the kinds of activities one does to verify codes.

How does this document (and others like it) relate to Modeling & Simulation at SNL?

How do I incorporate this activity in relation to other project activities such as
--Development of new Capabilities,
--Software Quality,
--Calculation Verification,
--Validation,
--UQ, and
--Modeling & Simulation?

In particular, if I desire to do Code Verification within my Project, what should I actually do?

No consensus on these issues has been achieved at SNL.
We would not be surprised if the Checklist document generates a lot of discussion.

This is a preliminary attempt, based on many discussions with SNL developers & verification experts. Can perhaps use this framework to develop an approach for your code.

A Context for Code Verification

I. Within Software

Software – Any software, software system, code, libraries, practices, development, testing, platforms

Software Quality

– Fitness for Use (Juran), Conformance to Requirements (Crosby), The result of managing vulnerabilities to a targeted risk (Peercy) Product-focused.

Software Engineering

– The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software.

Software Verification

– Verifying that the software compiles on all targeted platforms, all components are properly linked and interact correctly, speed & memory requirements are met, expected behavior occurs, errors & exceptions are properly handled, gives stable results from one version to the next, and lots of other stuff.

Numerical Algorithm Verification (NAV)

– Demonstrating that the relevant numerical algorithms within a code have been implemented in agreement with their known mathematical properties such as order-of-accuracy, discrete conservation, stability, and consistency. (part of Code Verification)

A Context for Code Verification

II. Within Modeling & Simulation

Modeling & Simulation

– The use of software to simulate physical phenomena for exploration, design, qualification, or certification.

Analysis

– Doing calculations with a code and analyzing the results to answer questions about a physical system.

UQ

– Quantification of the uncertainties in a calculation or set of calculations due to parameter uncertainty, model uncertainty, discretization error, and other uncertainties.

Validation

– Demonstrating that the equations solved within a code are an adequate representation of physical reality. Are we solving the right equations?

A Context for Code Verification

II. Within Modeling & Simulation

Verification

– Are we solving the equations right?

Calculation (or Solution) Verification

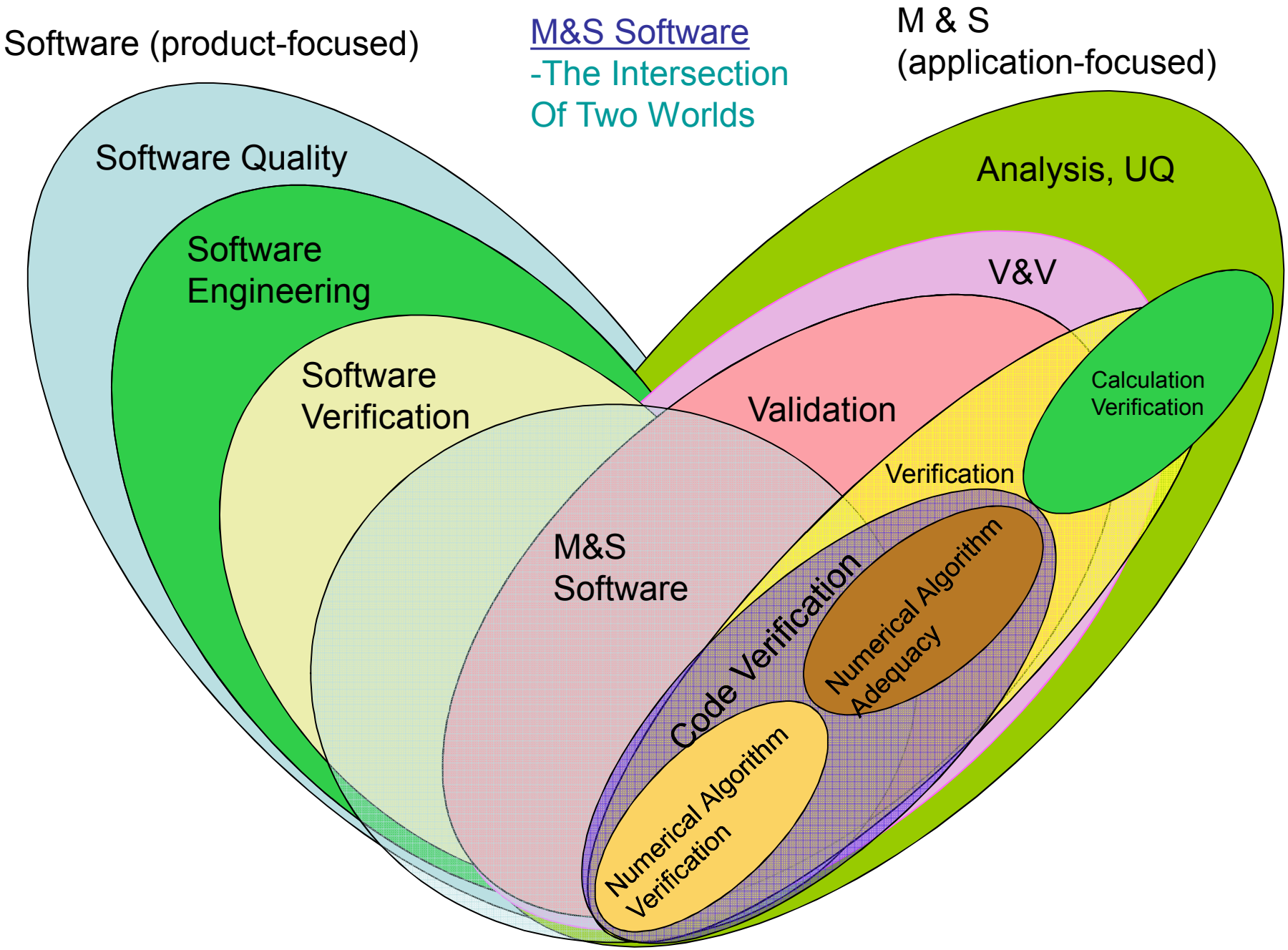
– Determination of the discretization error and error bars in a particular simulation intended to say something about an application. Exact solution to equations is unknown. Tests the calculation, not the code. Sanity check.

Numerical Algorithm Adequacy (NAA)

– Determining that a numerical algorithm within a code meets the accuracy, robustness, and speed requirements with respect to a specific application. (the other part of Code Verification)

Examples of NAA. On the specific application:

- The residual of the discrete system of equations can be driven down to an acceptable level,
- A numerical solution is likely to be obtained, provided the input stays within some range,
- The observed order-of-accuracy is sufficient to permit reaching the asymptotic regime for the anticipated practical mesh sizes,
- The numerical solution does not exhibit numerical artifacts that corrupt the results,
- The numerical algorithm adequately captures the important features of well-known benchmark problems.


Software (product-focused)
M&S Software
-The Intersection
Of Two Worlds
M & S
(application-focused)
Software Quality
Software
Engineering
Software
Verification
Validation
V&V
Analysis, UQ
Calculation
Verification
Verification
M&S
Software
Code Verification
Numerical Algorithm
Adequacy
Numerical Algorithm
Verification

Remarks on Code Verification

1. M & S software includes both PDE-codes and non-PDE codes.

2. Numerical algorithm verification (NAV) can find a host of coding mistakes and more.
   - Acceptance Criteria based on observing known mathematical properties of algorithms.
   - If properties are unknown, then one is doing NAA.

3. NAA is application-specific. Acceptance Criteria based on meeting the requirements of the application.

4. Algorithm verification (NAV) must precede numerical algorithm adequacy (NAA).

5. In V&V, the model is initially not adequate, so that some equations and models are preliminary. The model matures during the course of a V&V exercise. Fidelity to physics.

6. In code verification, it is irrelevant if the model is preliminary or mature because it is only concerned with correctness & adequacy, i.e., fidelity to algorithms & requirements.

7. Both preliminary and mature models must be verified because code verification serves as the foundation for Calculation Verification, Validation, & UQ.

8. Order-Verification Tests: Comparing the numerical solution on a sequence of uniformly refined meshes or time-steps to a known exact solution to establish an observed order-of-accuracy.

Now that you know what code verification is,
do you know what to do in order to verify the code in your SNL project?

Commercial Code Verification vs. Code Verification for M&S

I. Commercial codes with many potential users

- Code verification necessary for any “supported” code feature or capability

- Includes all valid combinations of FC’s

- A huge effort

II. M & S codes sponsored by “Application” Teams

- Code verification necessary for any code feature or capability
  that is relevant to the Application.

- Includes all combinations required by the Application (a lot fewer)

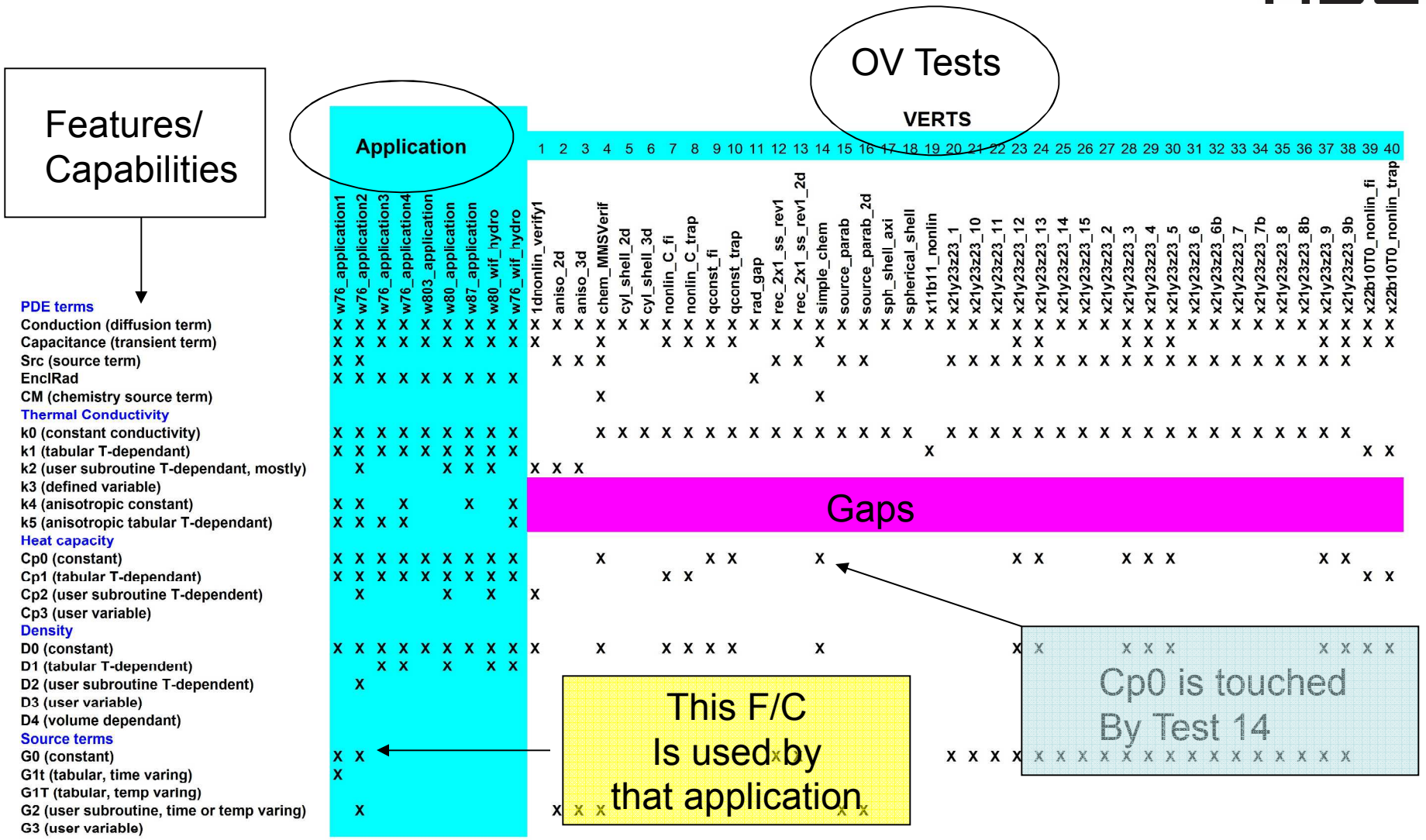
- The more customers you have, the more like a commercial code you become.

- Code Verification = Feature & Capability Verification? (What is a F/C, anyway?)

- Is everything else Software Verification?

Code F/C Coverage for VERTS

K. Dowding



An Extended F/C Coverage Table

The Features & Capabilities in this Table are those needed in Application Y.

Passed Tests Only

Features/ Capabilities

Date:05/12/08	T1	T2	T3	T4	T5	T6	Exhaustiveness Level
FC1			X				3
FC2	X						4
FC3							GAP
FC4						X	5
FC5	X		X	X			2
FC6				X			2
FC7							1
FC8					X		1

FC8 is touched by Test 5

- F/C's are derived from the application.
- A different Table may be created for each new application.
- Tests in the table only include PASSED tests (A/C met).
- The tests should include order-verification, but might include other kinds.
- Exhaustiveness is the bottom line.

FC4 is Exhaustively Tested by Test 6

This is the primary Evidence that needs to be supplied.

Exhaustiveness Levels

How exhaustive, in aggregate, are the passed tests that correspond to a particular F/C ?

Exhaustiveness measured in two dimensions:
- The appropriateness & rigor of the Acceptance Criteria (A/C),
- The degree to which the tests collectively cover the F/C the application requirements

Subjective, graded scale (perhaps Levels 0-3)

Example:

Level	Description
0	No passed tests for this F/C; a gap exists
1	No gap. Weak A/Cs and poor coverage.
2.	No gap. Good coverage, but weak A/C. (or Strong A/Cs, but poor coverage).
3.	No gap. Strong A/Cs and good coverage.

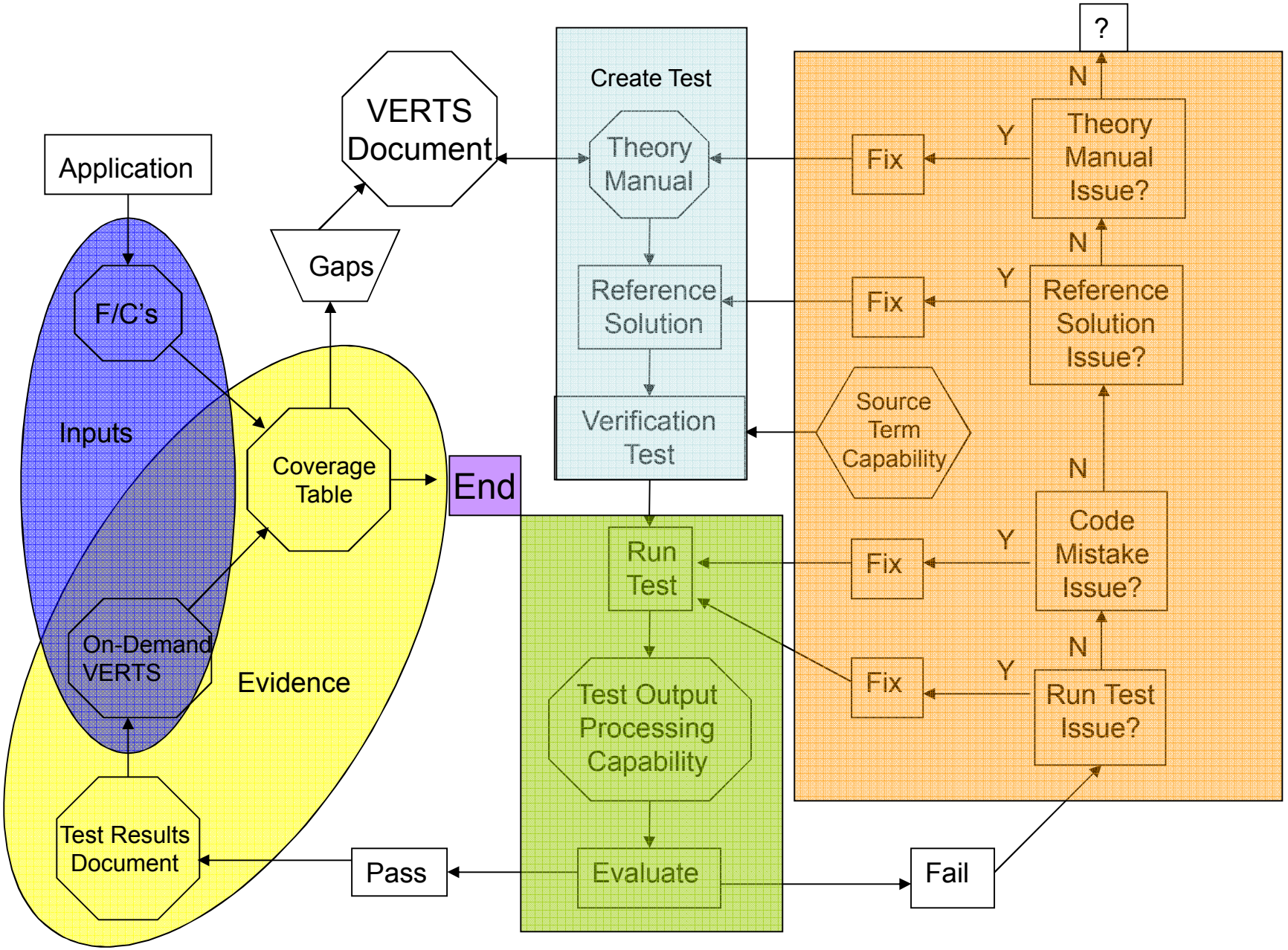
Weak A/C vs. Strong

Weak – Judgment, eyeball norm, regression test with lost gold copy
Medium – Comparison to another numerical solution
Strong – Comparison to an exact solution, with mesh refinement

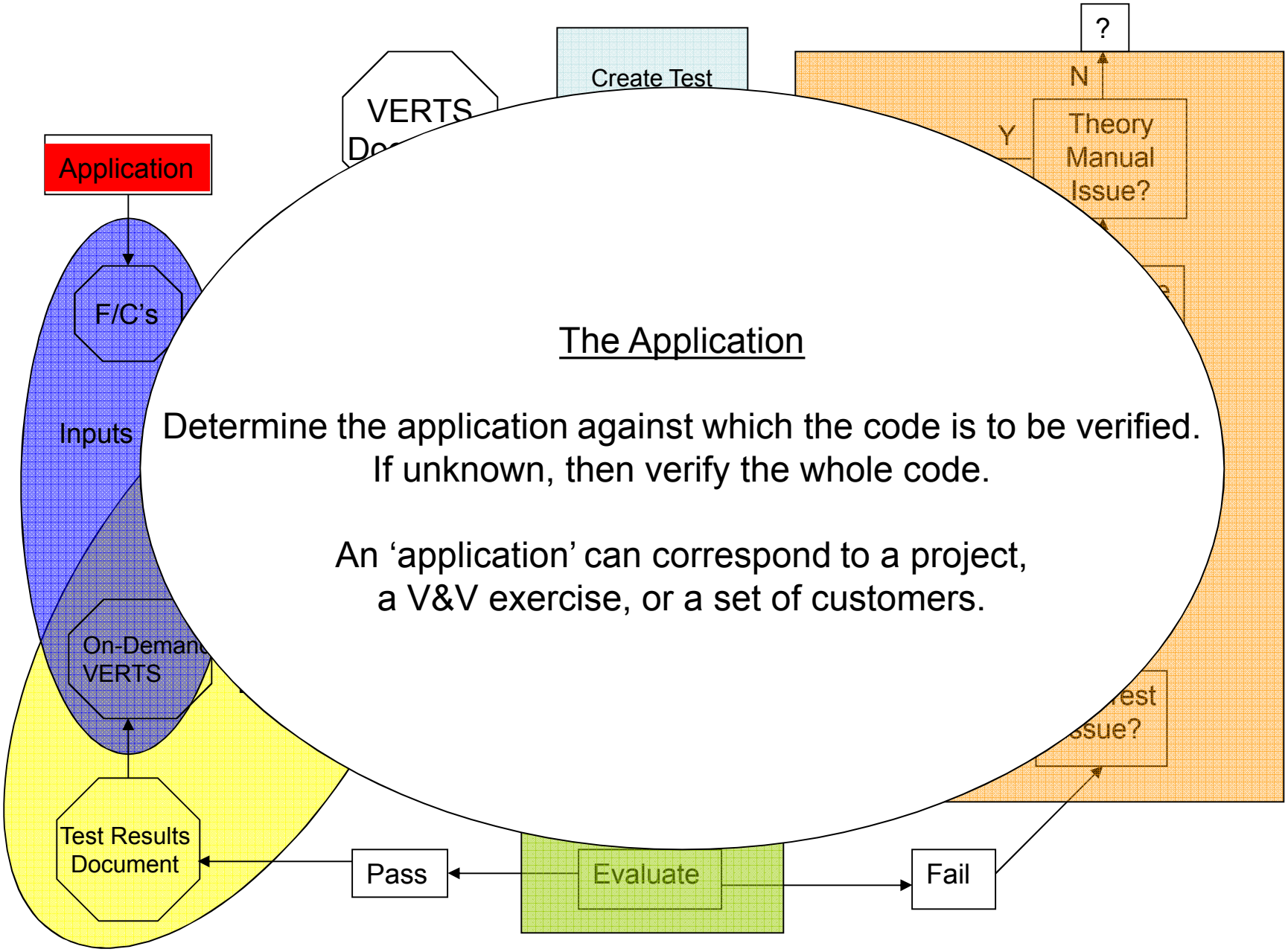
Poor Coverage vs. Good Coverage

Poor – F/C partially tested
Good – all aspects of the F/C are tested

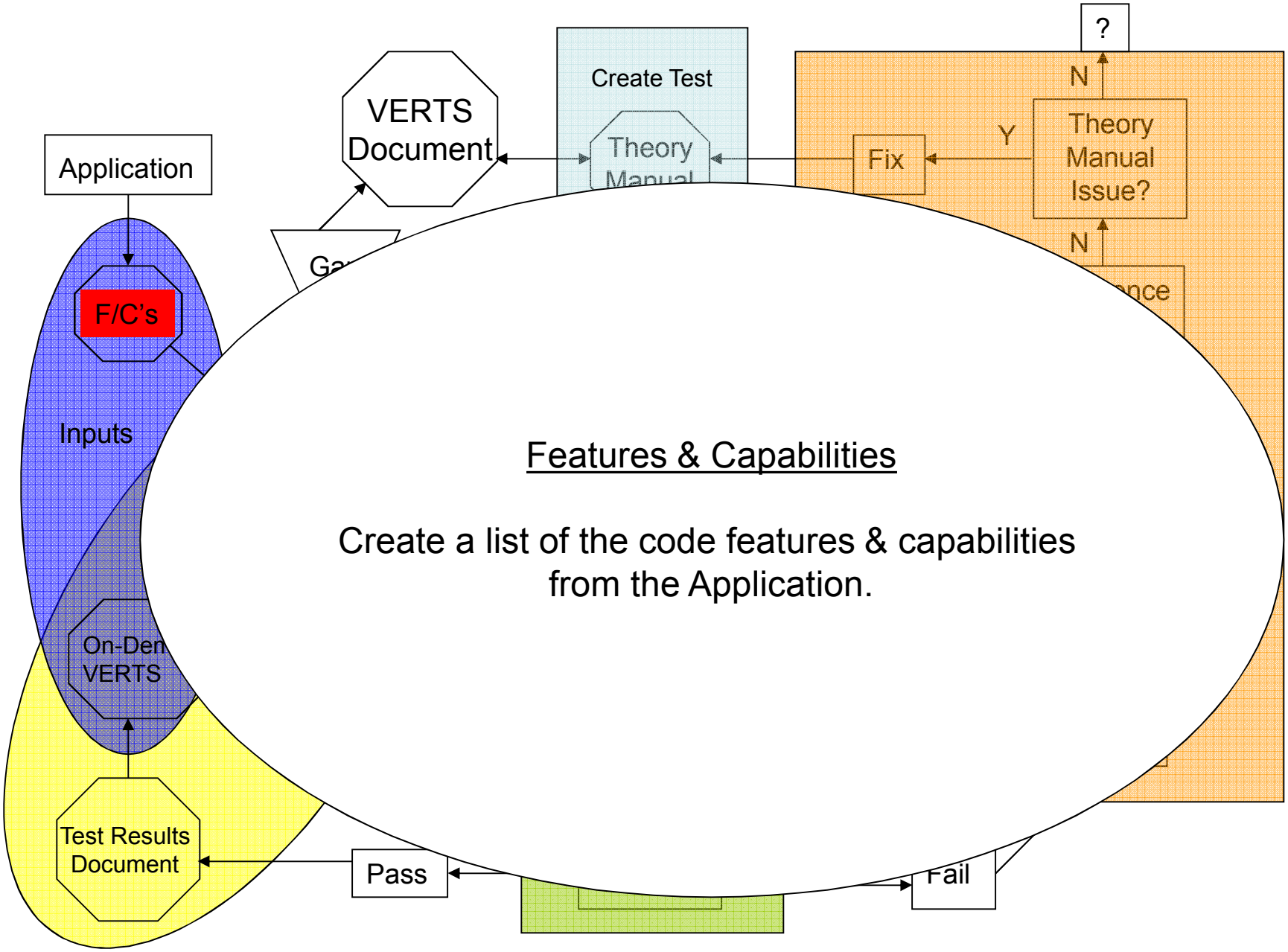
Coverage Table Flow-Chart



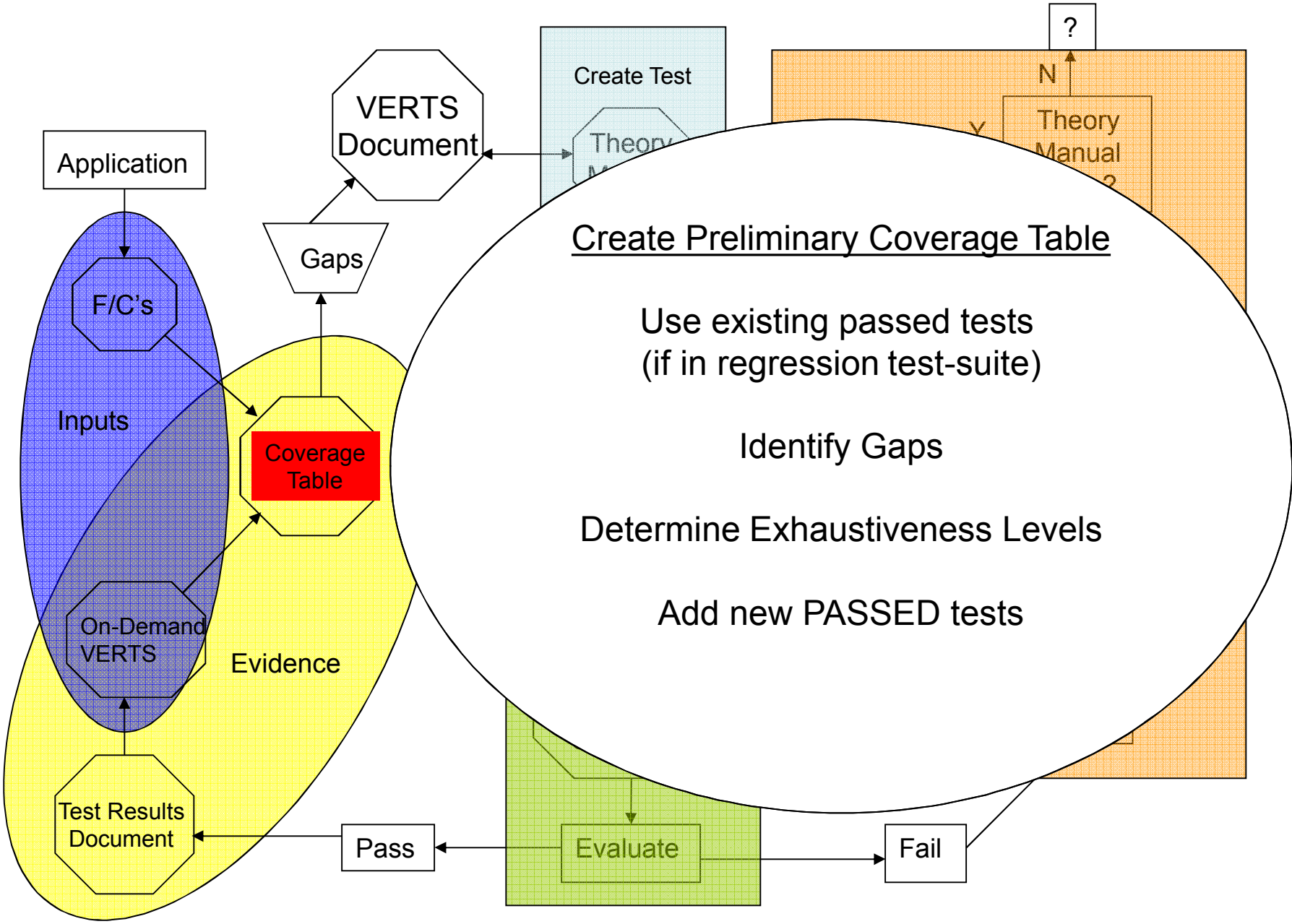
Coverage Table Activities



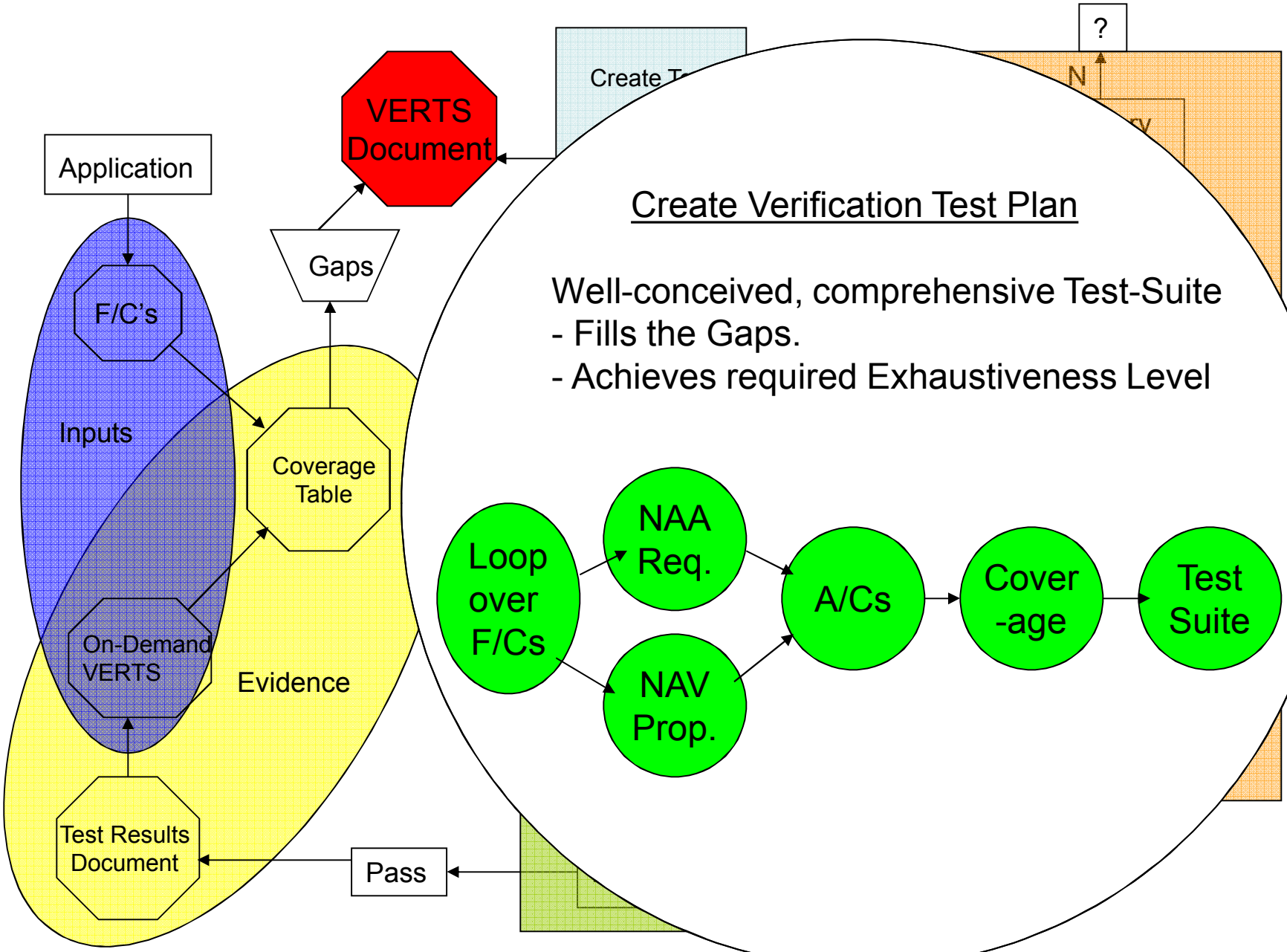
Coverage Table Activities



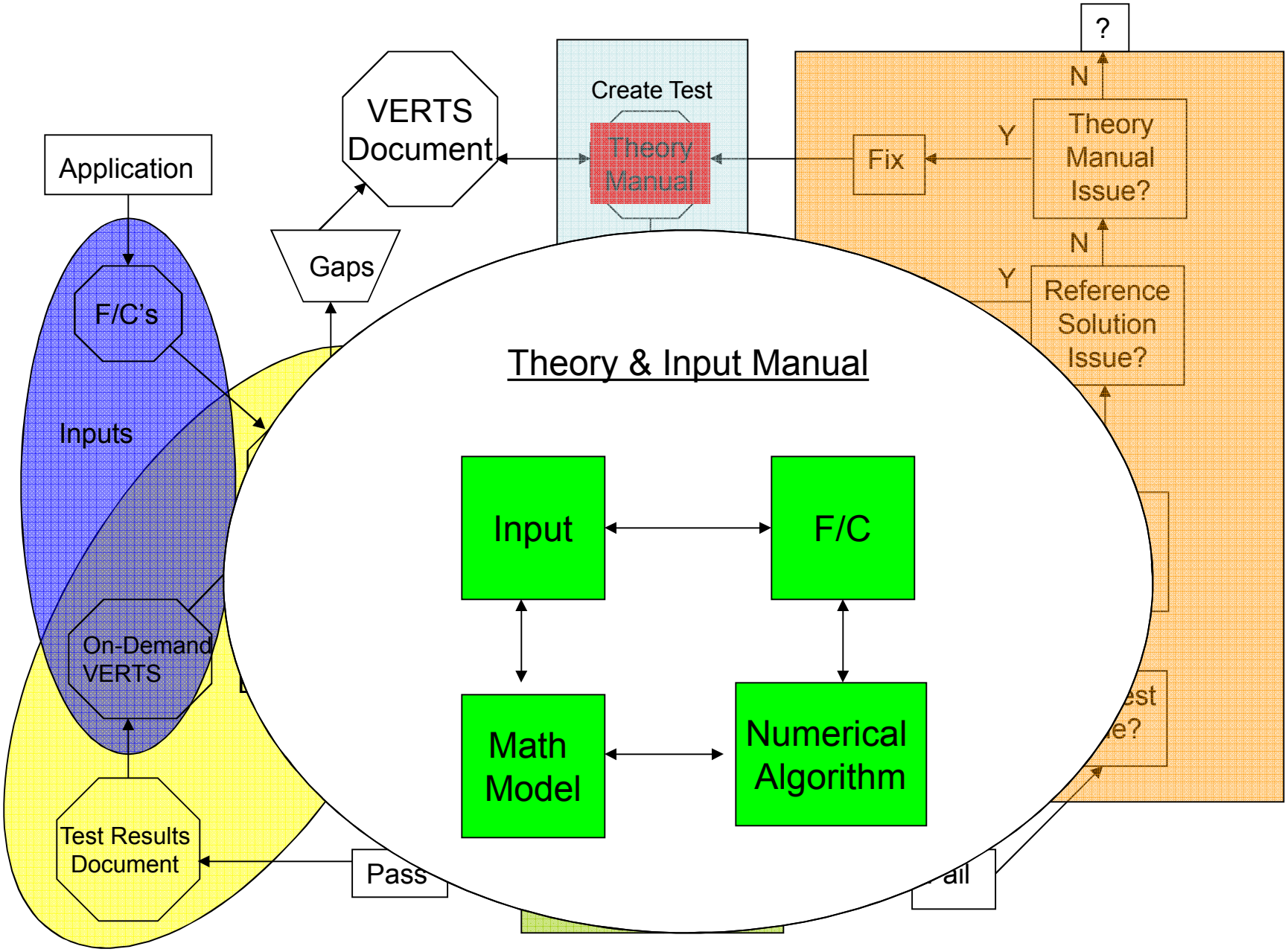
Coverage Table Activities



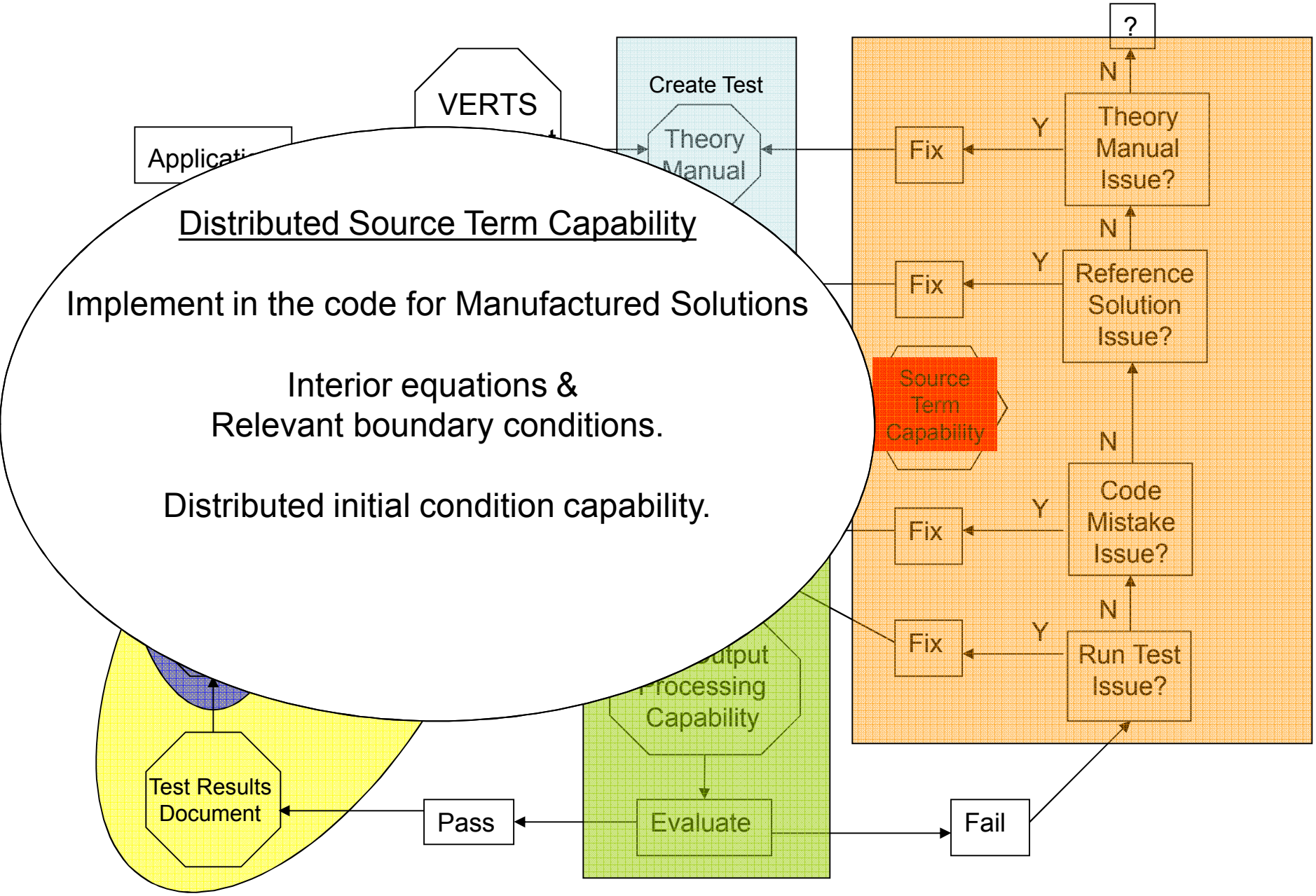
Coverage Table Activities



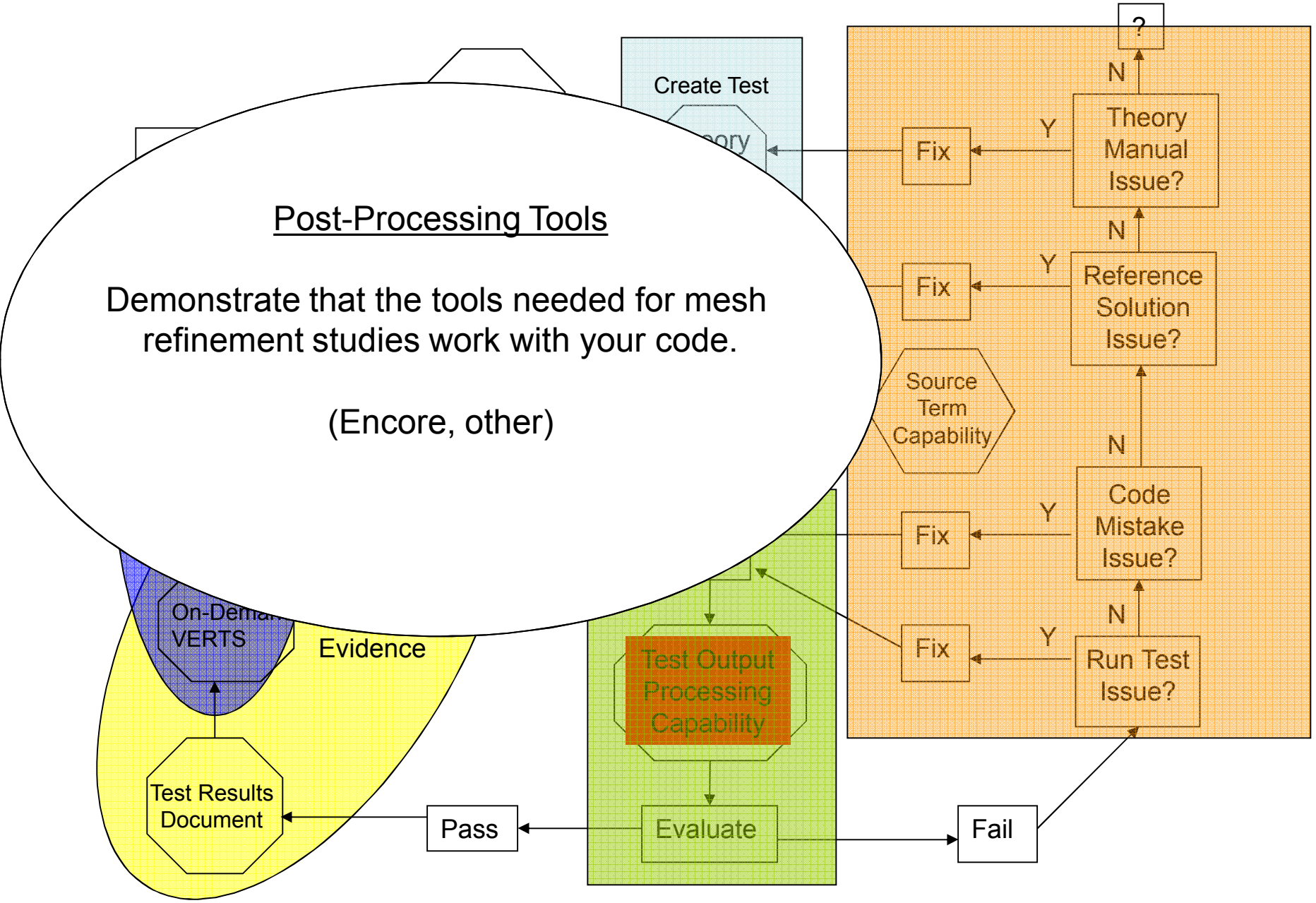
Coverage Table Activities



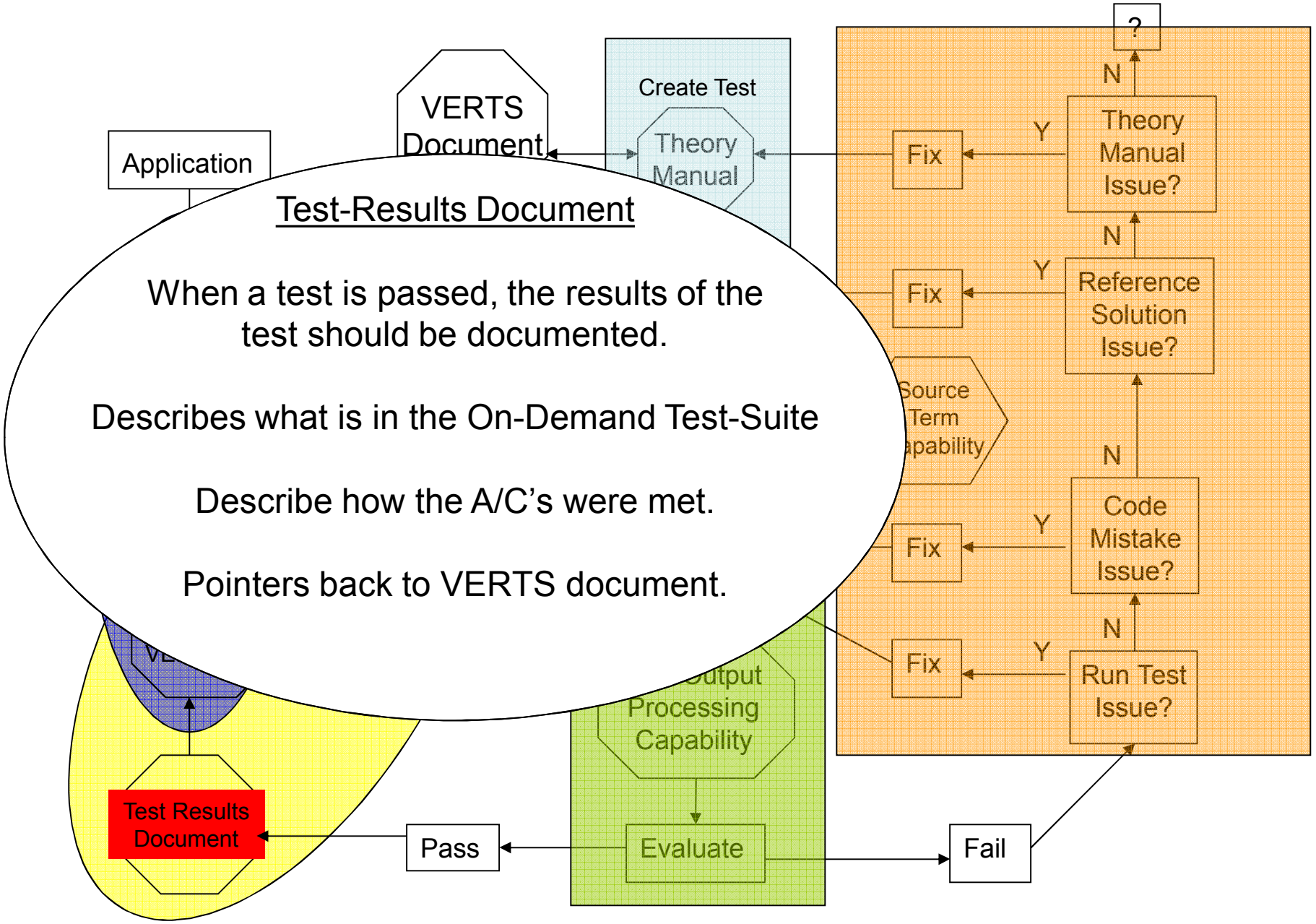
Coverage Table Activities



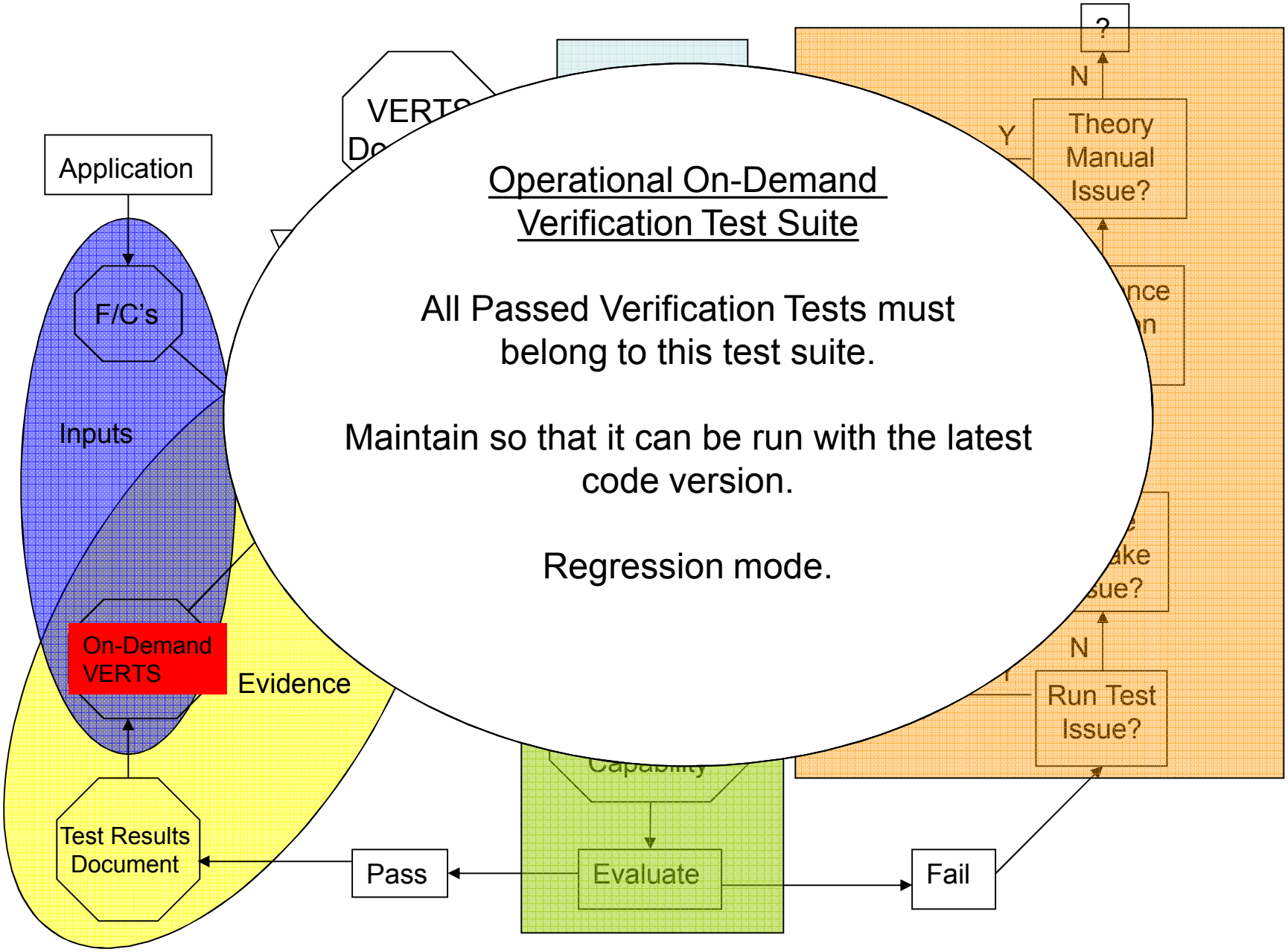
Coverage Table Activities



Coverage Table Activities



Coverage Table Activities



The Code Verification Checklist

Part I

Ensure the code is “Test-able”

a. Document the mathematical model and solution algorithms.

b. Ensure the PDE code supports user-defined distributed source terms.

c. Document the F/C’s in the code in relation to the mathematical model.

d. Document the Input in relation to the F/C’s.

e. Ensure that compatible software tools that facilitate refinement studies exist and meet your requirements.

The Code Verification Checklist

Part II

Maintain Well-Conceived, Comprehensive, and Documented Test-Suites

1. Identify the specific application against which the code is to be verified.
2. Create and Maintain Coverage Tables for the Test Suite
3. Document Verification Test Plan
4. Newly-passed tests are
    a. Scrupulously added to the on-demand test suite
    b. Documented in terms of equations solved, acceptance criteria, and results.

Evidence Generated by the Coverage Table Activities

Primary:

1. The List of F/C's relevant to the Identified Application,
2. Code Verification Plan (VERTS) document,
3. The Test Results Document,
4. The Code Coverage Table (vs. time)

Secondary:

5. The On-Demand Verification Test Suite,
6. The theory manual (math & algorithms to F/C's),
7. Input manual (Input to F/C's),
8. Source term capability as found in Input manual,

Management & Customers should be asking for the evidence of code verification.

Maturity Model

Maturity Model helps one Bootstrap into Code Verification, and provides a means of assessment.

Measure maturity in terms of levels.

Coverage Table:

Each F/C in a coverage table has an Exhaustiveness Level (E_k, k = 1… # F/Cs.)

Level	Description
0	No passed tests for this F/C; a gap exists
1	No gap. Weak A/Cs and poor coverage.
2.	No gap. Good coverage, but weak A/C. (or Strong A/Cs, but poor coverage).
3.	No gap. Strong A/Cs and good coverage.

Combine the individual Exhaustiveness Levels for each FC into a single FC-level for multiple F/Cs.

FC-Level = Min_{k=1… #F/C} { E_k }

Predictive Capability Maturity Model (PCMM)

	Maturity Level 0 Low Consequence, Minimal M&S Impact (e.g. scoping studies)	Maturity Level 1 Moderate Consequence, Some M&S Impact (e.g., design support)	Maturity Level 2 High Consequence, High M&S Impact (e.g., Qualification Support)	Maturity Level 3 High Consequence, Decisions-making based on M&S (Qualification or Certification)
Code Verification + M&S Software Verification	FC Level = 0, Little or no SQE	FC Level = 1, Code managed by SQE procedures	FC Level = 2, Some peer review conducted	FC Level = 3, Independent peer review conducted

Comparison to Current Practice

What's the same?

- Application-driven,
- Regression test-suites,
- VERTS,
- Manuals
- Software Verification

What's different?

- Code coverage table,
- Levels of Exhaustiveness, Maturity Model,
- Specific Evidence of Code Verification,
- Code is testable,
- Emphasis on both NAA and NAV,
- Manuals provide seamless connections between input, F/C, algorithms, & math model