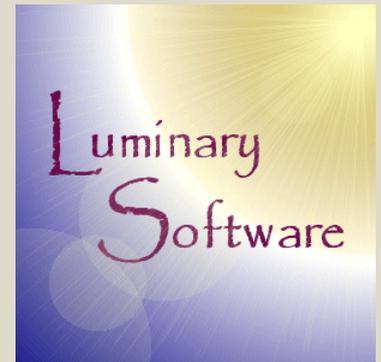

Validation & Verification Implications for Modeling and Simulation Reuse

John D. McGregor
Luminary Software
Clemson University



Goal

Achieve significant, strategic reuse in both product and test assets from concept definition to deployment

Focus

Two possible targets:

Model – is it correct?

→ Program – is it accurate? ←

To reuse an M&S system we must have confidence in the system, which we can only have (practically) through V&V.

Reuse *!%@*

Many professionals don't take reuse "schemes" seriously anymore

Schemes are initiated by people without sufficient breadth of responsibility

Schemes are initiated by people without sufficient depth of control

Test assets

Test plans

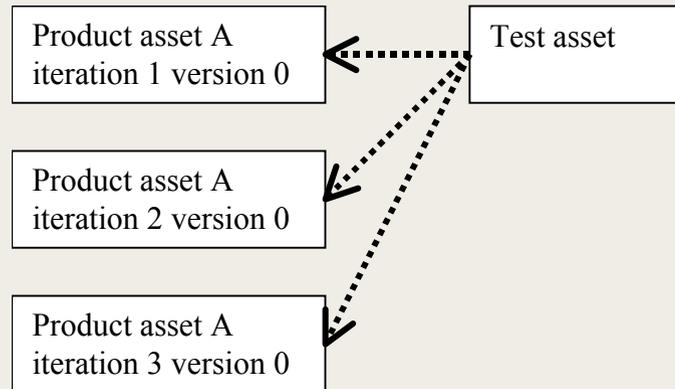
Test cases

Test reports

Used repeatedly on same system

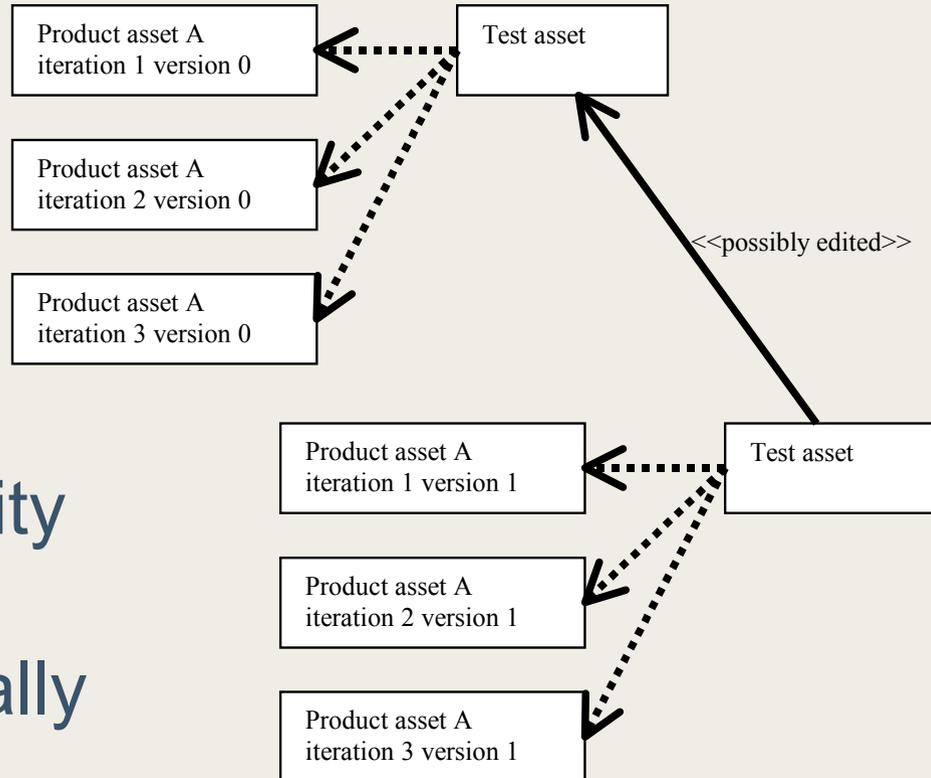
Used repeatedly across multiple systems

Reuse in same version of a product



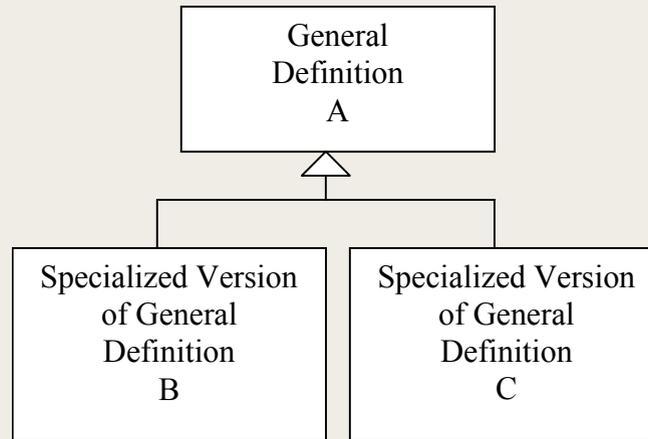
Same functionality, correcting defects
Functionality developed iteratively

Reuse across versions



Functionality
developed
incrementally

Definitional reuse



Assumptions - 1

- The greatest impact of “multiple use” of assets comes at the definition level rather than the operation level.
- Traceability between product assets and test assets facilitate the management of changes to assets.
- The software assets being produced capture and use domain expertise that is important to the success of the organization.

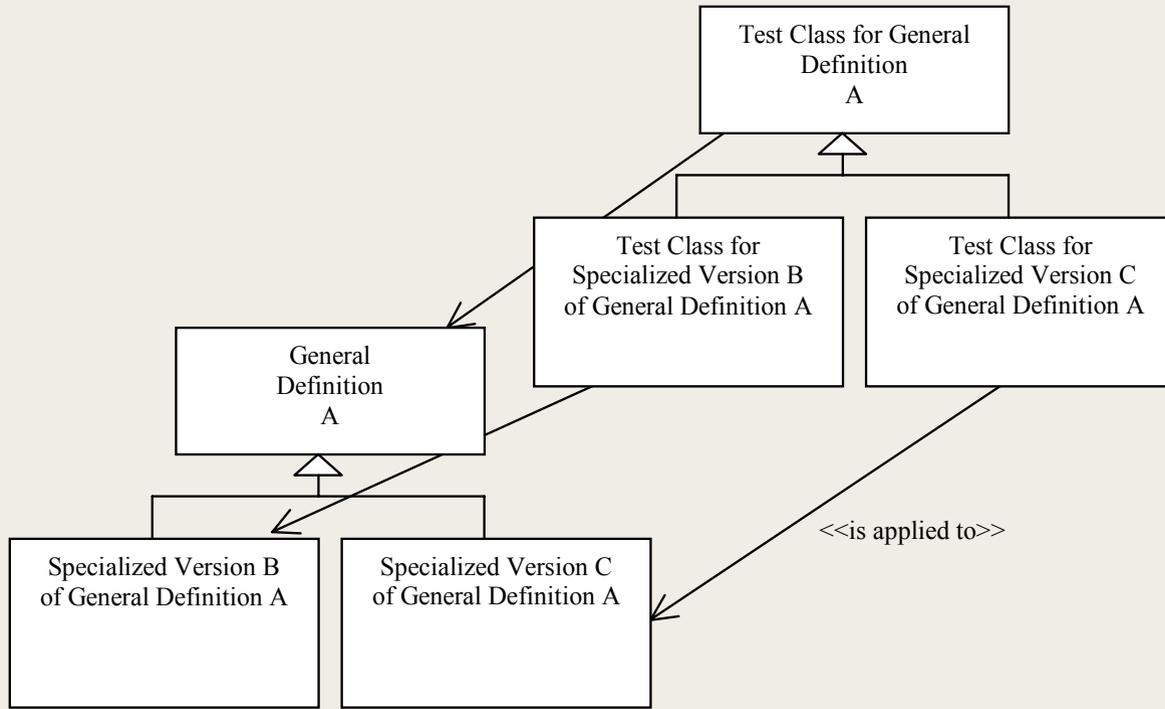
Assumptions - 2

- The V & V activities include static techniques such as inspections and reviews as well as dynamic tests.
- The various verification and validation activities form a process that is independent of, but intertwined with, the product development process.
- The later in the product creation process that V & V begins, the less effective it will be

Principles - 1

- An asset is only usable within a certain context.
- Test assets are more likely to be used for multiple systems if the product assets they are used to test are used in those same systems.
- Designing tests to have the same structure as the portion of the system to be tested produces test assets with the least amount of effort possible and maintains close traceability.

Parallel Architecture for Component Testing



Principles - 2

- The concept of an interface as a specification separate from implementations of that interface supports the use of the specification information across multiple implementations.
- Abstraction, encapsulation and information hiding are standard devices for supporting multiple in software design.

Software Product Lines

A software product line is a strategy for planning and producing multiple products that share a related set of features

A software product line organization adopts a comprehensive set of practices that span the areas of organizational management, technical management, and software engineering.

These practices include review, inspection, and testing practices that emphasize the use of assets across the products in the product line.

Product Line Results

Raytheon experienced 10 fold improvement in quality, 7 fold improvement in productivity

Cummins, Inc. experienced the time to build typical system reduced from 1 year to 1 week

Commonality and Variability Analysis

Determine the elements in common across products

Determine the allowed variabilities across products

This should be done during system creation but they may be conducted after the fact to aid in test creation.

Software architecture

- The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of these components, and the relationships among them
- The Architecture Trade-off Analysis Method is an architecture evaluation method that utilizes a set of generic techniques that are used to specialize a set of general constructs into the specific questions to be used for evaluation
- Use cases are the source of scenarios that drive ATAM

Test assets in a product line

- test plans and processes,
- architecture evaluation scenarios,
- standard component interface test assets,
 - system level test assets
 - Each asset created in a product line context is guaranteed to be used in several products in the product line by virtue of the planning, scoping, and architecture processes.

Proactive reuse is more profitable than reactive reuse

- comprehensive planning process
- domain analysis
- use of test assets follows the use of the corresponding product assets

Component test packages

encapsulates a product component with:

- A specification,
- A test plan for the component,
- Test cases,
- Test data sets
- Previous test reports

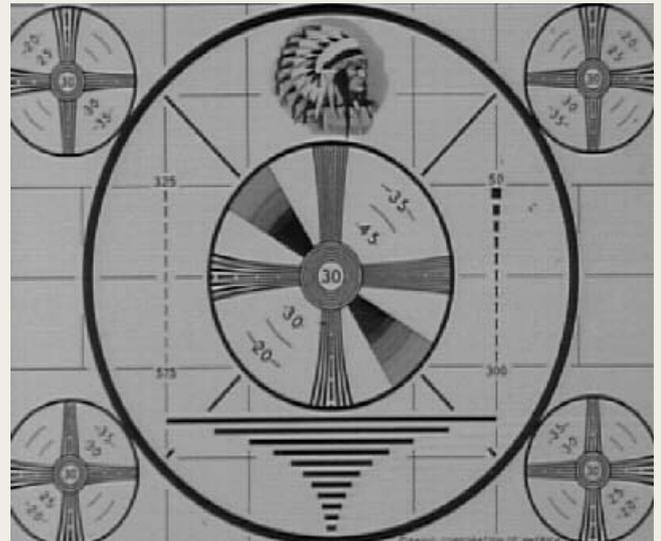
System test packages

groups together:

- a family of use cases,
- the test plan for that set of requirements,
- inspection scenarios involving those use cases,
- the functional tests for that aspect of the system, and
- test data sets

Design and test patterns

- Design pattern captures a canonical solution to a design problem
- Test patterns capture how to test software designed using a specific design pattern

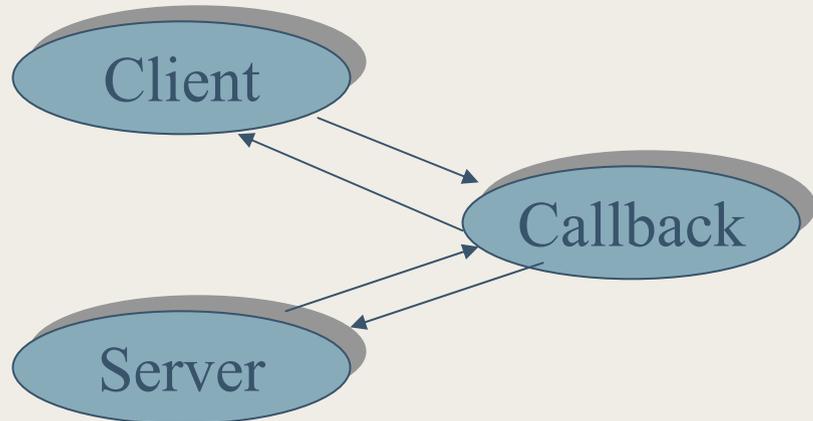
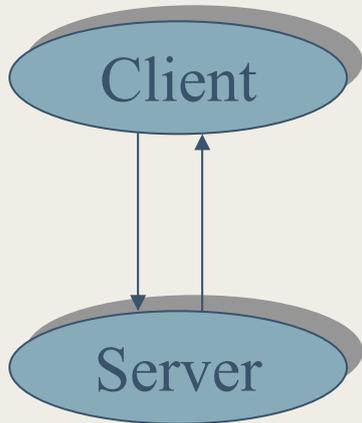


Content of Test Pattern

- Problem - Description of pattern to be tested
- Context - Special testing conditions
- Forces - What types of faults are we looking for?
- Solution - Test case selection strategy that tests the interactions among the components that implement the pattern

Problem - Distributed Callbacks

The synchronous communication between two objects is modified to be asynchronous by adding a callback object



Testing Distributed Callbacks

Context

- An intermediate object forwards messages

Forces

- Possible to intermingle successive messages/responses

Solution

- Build test cases that submit a second message prior to receiving the response from the first message
- Construct multiple clients, submit multiple requests through multiple callback objects

Thanks

Contact me at:

- johnmc@lumsoft.com

