Building Reusable Testing Assets for a Software Product Line

John D. McGregor
Visiting Scientist - SEI
Partner – Luminary Software, LLC
Associate Professor - Clemson University
johnmc@cs.clemson.edu

Qualifications

- Worked on large software systems - 500 developer FTE
- Worked on early product line efforts
- Working on current product line efforts
- Domains:
  - telephony
  - wireless
  - insurance
  - ...

Motivation

- Test effort estimates range from 50 - 200% that of the actual production system
- If effective use of test assets can reduce this by 10%, this represents substantial savings
- In fact we can reduce the effort by considerably more than that depending upon the initial maturity

Outline

- Overview
- Definitions
- Activities & Assets
- An example company

Overview

- *A software product line is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way.*

Overview

- *sharing a common, managed set of features*
  - *Tests that are tied to features should be applied in many products*
  - *These tests can be managed in the same way as the production assets to keep they synchronized*

Overview

- *particular market segment or mission*
  - *The market segment will determine specific quality levels*
  - *Which in turn determine test coverage criteria*

Overview

- *common set of core assets*
  - *There are test-related core assets*

Overview

- *a prescribed way*
  - *The test infrastructure must be compatible with the production strategy and infrastructure*

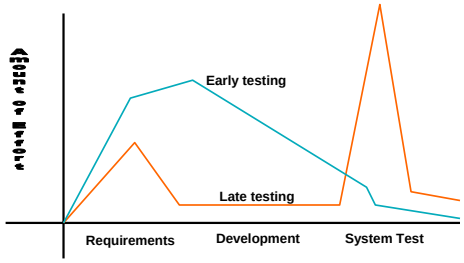
Overview

- The basic test assets are
  - test plans
    - summary of strategies for test case selection
  - test cases (including test data)
    - <pre-conditions, input stimulus, expected results>
  - test reports
    - standard summary of results of the tests that have been run
  - test harnesses
    - architecture-based

Overview

- Making assets reusable requires both management and technical support
  - resources must be allocated on a different schedule which changes management practice
  - "correct" design choices are different
- Reuse can occur both vertically between the product line and the products and horizontally between products
  - vertical reuse propagates interfaces
  - horizontal reuse propagates opportunistic components

Overview



Overview

- Testing is conducted during and after development
  - during development we are concerned that the work does what it does correctly
  - after development we are concerned that it does what it is supposed to do
- Testing is conducted before and after code is written
  - before code is written test cases guide the inspection technique
  - after code is written test cases are executed using the product code

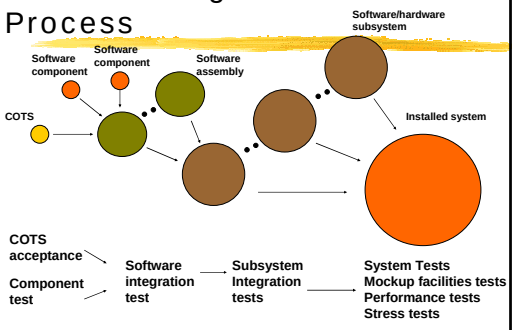
Overview

- A major focus for testing in a product line environment is interactions
- It is not possible for the core asset builders to test every possible product configuration that can be built from the assets
- Each product team will need an extensive program of incremental assembly intertwined with integration tests of the assembly at each increment

Outline

- Overview
- Definitions
- Activities & Assets
- An example company

Basic Testing Process



Definitions

- Testing - the detailed examination of a product guided by specific information
- Tester - any person evaluating an artifact against expectations
- Developer - the person constructing the product under test
- Domain expert - person knowledgeable in the subject matter of the product under test

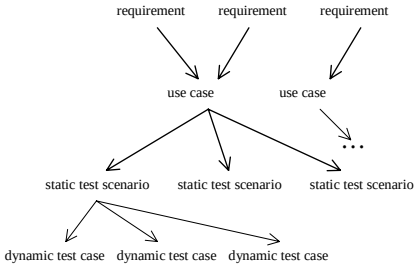
Test points - 1

- Requirements model
- Analysis and design models
  - Domain analysis
  - Application analysis
  - Architectural design
  - Mechanism designs
  - Detailed designs
- Individual components
  - Interfaces
  - Implementations

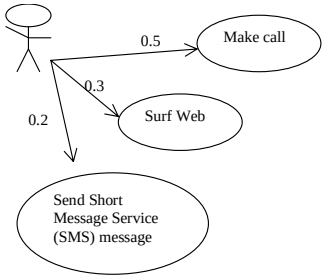
Test points - 2

- Clusters of components
  - Interactions between components
- Increments
  - end user functionality
  - Interactions with existing increments
- Complete developed systems
  - comparison to requirements model
- Deployed systems
  - interaction with other installed applications

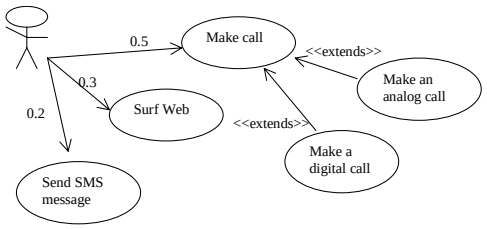
Test asset dependency structure



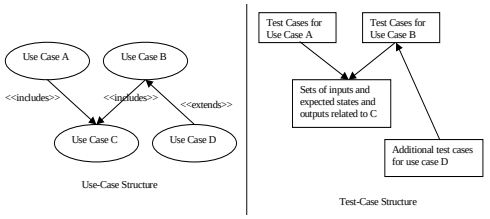
Use cases



Use cases and variations



Use case/test case parallels



Exercise

- Think about your own development process.
  - Make a list of the major steps.
  - Identify the testing activities for each development step.
  - Identify those development steps for which there is no testing activity.
    - What types of defects could be escaping from these steps?

Reusable test assets

- Knowledge
  - what can go wrong
  - techniques
  - patterns
- Models
  - document templates
  - specifications that appear multiple times
- Code
  - test case implementations

25

IEEE Test Plan - 1

- Test plan for each phase of testing
- The plan is used to allocate resources and as input to scheduling
- The core asset developers create general test plans for the various points in the process
- Product developers specialize each general plan to fit their specific product

26

IEEE Test Plan - 2

- **Introduction**
- **Test Items**
- **Tested Features**
- **Features Not Tested (per cycle)**
- **Testing Strategy and Approach**
  - Syntax
  - Description of Functionality
  - Arguments for tests
  - Expected Output
  - Specific Exclusions
  - Dependencies
  - Test Case Success/Failure Criteria

27

IEEE Test Plan - 3

- **Pass/Fail Criteria for the Complete Test Cycle**
- **Entrance Criteria/Exit Criteria**
- **Test Suspension Criteria and Resumption Requirements**
- **Test Deliverables/Status Communications Vehicles**
- **Testing Tasks**
- **Hardware and Software Requirements**
- **Problem Determination and Correction Responsibilities**
- **Staffing and Training Needs/Assignments**
- **Test Schedules**
- **Risks and Contingencies**
- **Approvals**

Test case

- <pre-conditions, input stimulus, expected results>
- pre-conditions
  - need 35,000 transactions to be realistic
- input stimulus
  - run the “balance sheet” report
  - Includes test data
- expected results
  - must independently verify what the result should be
  - trace through 35,000 transactions?
- reuse of this test case will be VERY cost effective

Test report

- Test reports are more important in some domains than others
- Serve as legal proof of meeting obligations
- Present results
  - Individual results
  - Summarized results
- Analysis of test effectiveness

Other testing definitions

- Test technique
  - technical processes for test activities
  - example - combinatorial test design
- Test coverage
  - a measure of how much of the test item has been examined by the tests
  - example - one test per use case

Exercise

- Create a list of the test documents that your current project creates
- Are there documents that you produce that have not been discussed? Describe their usefulness.
- Are there documents that we have discussed that your project does not use? Is the information captured elsewhere? What might you be missing because of not having this document?

Product Line Assets

- Architecture
  - product line - sufficiently general to describe an entire set of products
  - product - sufficiently specific to describe an individual product
- Points of Variability
  - place in the architecture where one product is allowed to be different from other products
- Regions of Commonality
  - place in the architecture where all products must be the same

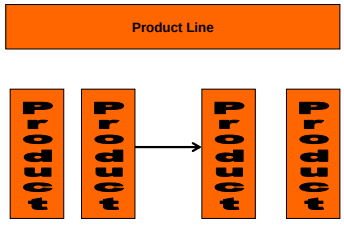
Product Line Assets

- Architecture
  - reusable test patterns
- Points of Variability
  - Standard interfaces/variable implementation
- Regions of Commonality
  - Best reuse across products

Basic Reuse Principles

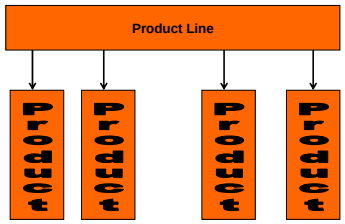
- Encapsulation
  - Packaging for portability
- Information Hiding
  - Separate what something does from how it does it
- Abstraction
  - Removing detail to focus on essentials

Horizontal Reuse



- **Product assets are reused in other products**
- **Components may be reused as is from one product to another**
- **Level of abstraction stays the same**
- **Test cases will be reused as is**

Vertical Reuse



- Product line assets are reused in the individual products
- Product line assets include varying levels of abstraction
- Abstract assets are specialized for use in products

Outline

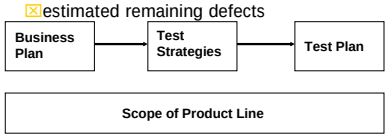
- Overview
- Definitions
- Activities & Assets
- An example company

Activities by Company Levels

- Organizational Management
  - test strategies
  - test standards
- Technical Management
  - test process
  - assign roles
  - construct test plan
- Software Engineering
  - construct test infrastructure
  - execute test plan

Organizational Management Activities

- Strategic level
  - organize for reuse of assets
  - establish strategies
  - integrate development and testing
- Set company-level standards
  - test coverage
  - estimated remaining defects



Scope of Product Line

Organizational Management Activities

- Encapsulation
  - business plan provides a single basis for strategies
- Information Hiding
  - strategies define what is to be achieved not how to achieve it
  - reliability, safety, ...
- Abstraction
  - separates test plans and strategies into levels of responsibilities

Exercise

- What test strategies are used in your domain currently?
  - Levels of coverage that are required
  - Method of sampling
- How might these be affected by a product line approach?

Technical Management Activities

- Establish processes
  - test process for each level
- Assign roles in testing processes
  - define roles and associate w/activities
- Construct test plan
  - test activity for each development step

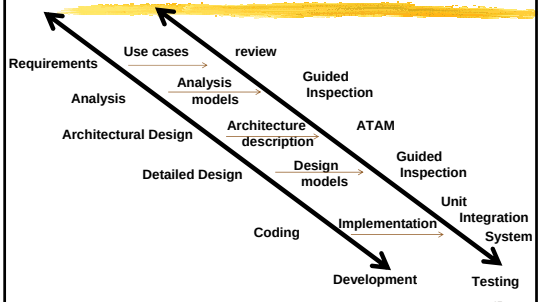
43

Technical Management Activities

- Encapsulation
  - System test plans are created on a per use case basis
  - all required information is bundled together
- Information Hiding
  - Test plans are specified but not implemented
- Abstraction
  - The test assets are defined in levels of abstraction
  - The levels correspond to the levels in the design of the software

44

Process Interactions

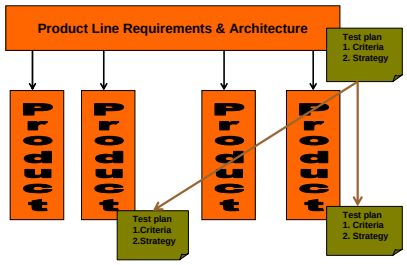


45

Creating the test plan

- The plan describes how test assets will be propagated from the product line to the product teams
- The plan assigns responsibilities to personnel for creation and maintenance
- The core asset test plan for a particular phase is modified for a specific product to reflect specific variations – “how much security is needed for this product”

Test Plan Reuse



Identifying Test Risks

- The resources required for testing may increase if the testability of the specifications is low.
- The test coverage may be lower than is acceptable if the test case selection strategy is not adequate.
- The test results may not be useful if the correct answers are not clearly specified.

Exercise

- What test activities are done at the product line level in your project?
- What test activities are done at the product level in your project?
- What risks do you see in your testing process?

49

Software Engineering Activities

- Construct test infrastructure
  - acquire tools that complement developers' tools
  - construct test software
  - mine for test assets
- Execute test plan
  - Static testing (reviews) of analysis/design deliverables
  - Dynamic testing of code deliverables

50

Software Engineering Activities

- Encapsulation
  - associate each test asset with a production asset
  - test case is associated with the product it tests
- Information Hiding
  - test interface vs test implementation
- Abstraction
  - follow good design practice
  - test cases are scripts written in a language that supports levels of abstraction

51

Product line perspective

- Test over complete range of specification since it will be used in many contexts
  - Product line test coverage levels should be higher than at the product level
  - To maintain the same level of aggregate quality coverage must be increased
- Provide test suites for regression testing of assets in a product context
  - Use structures appropriate to the development technology in use

Product perspective

- What has changed from the product line specification
  - more specific requirements
  - additional requirements
- What has changed from the product line implementation
  - modified components
  - new components

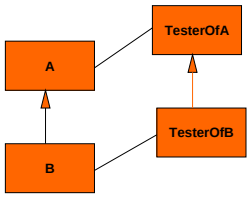
Architecture

- The product line architecture is verified using a guided inspection process
- The product architectures are also verified using guided inspection sessions
- The test cases for the guided inspection are reused from the product line to product inspection sessions

Architecture TradeOff Analysis Method (ATAM℠)

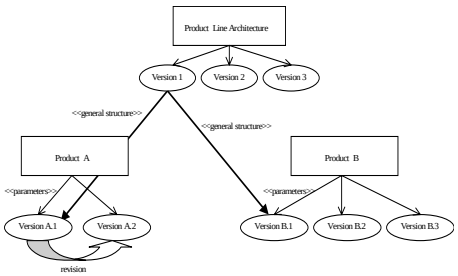
- "Test" the architecture
- Use scenarios from the use cases
- Test for architectural qualities such as
  - extensibility
  - maintainability
- Guided Inspection is a lighter-weight technique for the same purpose

"Inheritance" reuse

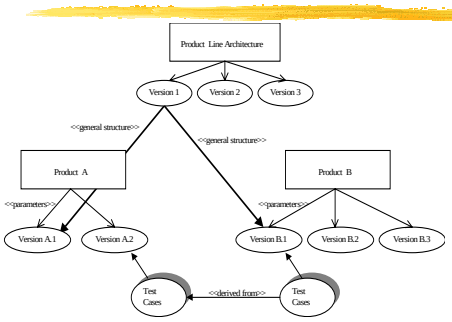


- **Test cases created for A are also viable test cases for B**
- **Test cases for a component are encapsulated in a single component**
- **This inheritance may be:**
  - **class inheritance in a programming language**
  - **it may be a specialization relationship in a language without inheritance**
  - **it may be an interface implementation**

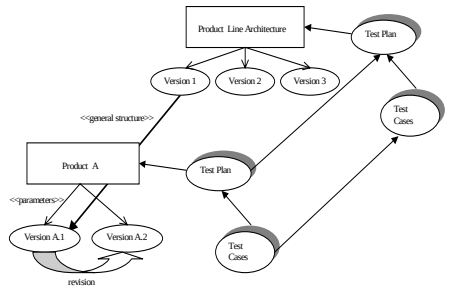
Associating Test Cases



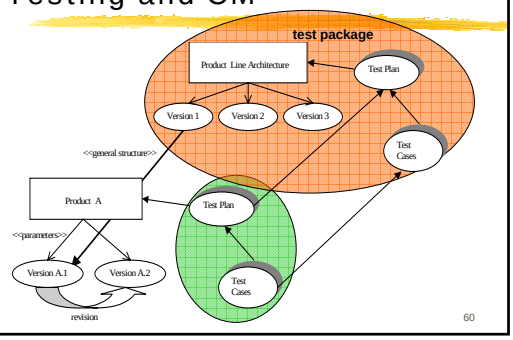
Reuse across products and versions



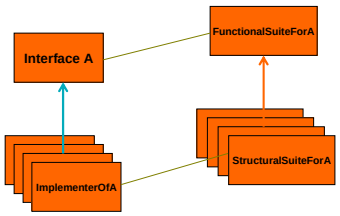
Associations among assets



Testing and CM



Interface reuse



- Functional test cases created for interface A are viable test cases for all implementations of A
- Test cases for a component are encapsulated in a single component
- Structural test cases are defined at the implementation level

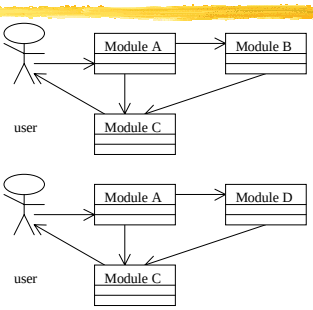
Points of Variability

- A point of variability is a point in the architecture where products can be different from other products and the product line
- A point of variability is defined by an interface that specifies the services that must be provided by the implementer
- A test suite is defined based on the interface
- Functional tests are passed from product line level to the product level

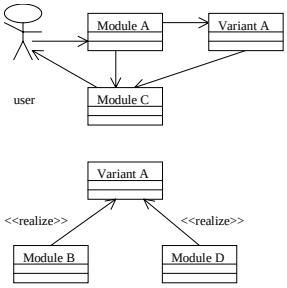
Regions of Commonality

- A region of commonality is that portion of the architecture that is fixed and remains the same in all products
- These regions are noted only because of their interface with the variable components in the architecture
- The product line implementation of this functionality should be tested in detail once and then only tested for interactions at the product level

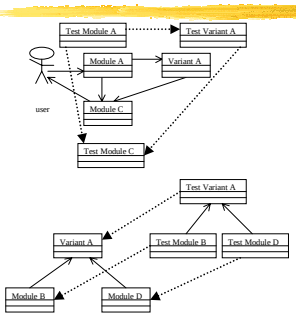
Variants shown in UML



Modeling Variability in UML



Associating variations with test cases



Interactions among variants

- Too many combinations to test them all
- Test plan should call for sampling
- Orthogonal Array Testing is a combinatorial technique to test all two way interactions as well as additional isolated interactions

Orthogonal Array Testing System (OATS)

$Factor_1$	$Factor_2$	$Factor_3$
1	1	3
2	1	2
3	1	1
1	2	2
2	2	1
3	2	3
1	3	1
2	3	3
3	3	2

Mapped arrays of variants

Handset Manufacturer	Protocol Set	Coverage Area
A	GPRS	international
B	GPRS	national
C	GPRS	local
A	EGPRS	national
B	EGPRS	local
C	EGPRS	international
A	UMTS	local
B	UMTS	international
C	UMTS	national

Exercise

- List the testing tools currently used by your project.
- How will they contribute to the product line effort?
- How do they need to be modified or improved to better support your product line project?

70

Outline

- Overview
- Definitions
- Activities & Assets
- An example company

71

Example company - ProtocolsRUs

- Builds components that are protocol engines for telecommunication software
- Uses a product line approach to produce highly reliable components
- Business plan calls for individual products to take less than 90 days once product line is established

72

ProtocolsRUs - Test Strategy

- Protocols are constrained by very specific standards and have well-defined states
- The high quality demanded by protocol-based applications requires thorough testing
- Strategies:
  - test from earliest stages of development
  - test sufficiently to achieve 99% reliability

73

ProtocolsRUs - Test Process & Plan

- Process
  - In addition to the process-based tests, each of ProtocolsRUs' components must also pass a protocol conformance test
- Plan
  - The plan describes how test assets will be propagated from the product line to the product teams
  - The plan assigns responsibilities to personnel for creation and maintenance

74

ProtocolsRUs – Test Approaches

- For state-based products, a state-based testing approach provides a natural correspondence
- Test cases walk the protocol component along specific paths
- Coverage is measured as the percentage of possible paths covered

75

ProtocolsRUs – Reuse

- Multiple implementations of each protocol are to be assembled to satisfy different deployment environments
- The functional test suite for each protocol will be reused on all protocol implementations
- A core asset test plan will be modified for each different implementation of a protocol to reflect the QoS attributes required by each

ProtocolsRUs - Test Results & Reports

- Test results and reports are used as the basis for ProtocolsRUs reliability advertising
- Actual reliability measurements are only taken on the final component products
- Each constituent component is also measured as it is developed
- These sub-product measurements are used to estimate the final reliability of the protocol component

Summary Exercise

- Make a list of changes to your testing process that you might recommend.
- Identify whether those changes occur at the product line or product levels.
- What are the trade-offs with each?

Conclusions

- Reuse must occur at all three company levels
- Test plans are templated and “inherited” from the the product line to the individual products
- Product line test cases provide regression test cases for the products
- Test reports provide feedback both to the product team and the product line team

79

References

- Clements, Paul; Kazman, Rick; Klein, Mark. Evaluating Software Architectures: Methods and Case Studies, Addison-Wesley, 2002.
- John D. McGregor Testing a Software Product Line, CMU/SEI-2001-TR-022, Software Engineering Institute
- John D. McGregor and David A. Sykes. A Practical Guide to Testing Object-Oriented Software, Addison-Wesley, 2001.

- ATAM is a service mark of the Software Engineering Institute

80