DANSE Software Quality Assurance

Tom Swain
Software Quality Research Laboratory
University of Tennessee
Department of Computer Science
DANSE Objectives

- Extensible open source framework
- Modular interoperability
- Reusable distributed components
- Legacy code migration path
- Broad scientific applicability
- HPC compatibility
- Broad user expertise range
Quality Challenges

- Large, distributed development team
- Wide range of software engineering experience
- Long term maintenance by SNS
- Straightforward extensibility for power users
- Reliable, intuitive operation for novice users
SQRL Tasks

- Software Quality Engineering Workshops for Each Science Subgroup
- Documentation of SNS Interface Requirements
- SNS Integration Plan
- Integration Test Planning
Essential Process Elements

1) Requirements Capture
2) Architecture Specification
3) Behavior Specification
4) Code Generation
5) Certification
6) User Guide Publication
7) Configuration Management
8) Change Tracking
Primary Work Product Relationships

1a. Derive Specification
1b. Invent Architecture
3a. Implement Structure
3b. Implement Function
4. Model Usage & Create Oracle
2. Map

Preliminary Requirements

Architecture Specification

Behavior Specification

User Guide

Code

Test Plan

5. Describe Usage

Copyright © 2006 Software Quality Research Laboratory
Software Production is Incremental

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

- Organize requirements into small functional increments
- For each increment
  - Elaborate requirements to expand behavior and architecture specifications
  - Code and integrate new functionality
  - Certify newly integrated system conformance with the cumulative specification
  - Update User Guide if increment will be released for beta or general use
  - Revisit requirements and increment plan
Requirements

- Use cases tagged for **traceability**.
- Initial requirements assumed to be incomplete, inconsistent, and possibly incorrect. (So don’t obsess about them)
DANSE Architecture Specifications

- **Framework** – interfaces, services, and rules for well-behaved components
- **Each Component** – class diagrams and API design
- **Each Application** – target environment, participating components, and their relationships
Behavior Specification: The General Case

1) Establish **system boundary**.
2) Define human/software/hardware **interfaces**.
3) Itemize **stimuli**.
4) Itemize **responses**.
5) Map **every stimulus sequence** to a response.

The discovery and documentation of derived requirements is a natural and desirable part of the specification process.
Map Code Specified in Behavior Specification to Architecture

Stimulus Gathering

Combination
- combination : string(idl)
- index
+ checkKey()
+ clear()

KeypadHandler
+ getKey() : int

TripInterrupt
+ trip()

SAController
- armed : boolean(idl)
- alarm : AlarmStatus
- code : CodeStatus
+ set()
+ trip()
+ clear()
+ badDigit()
+ goodDigit()

CodeStatus
«enumeration»
+ none
+ 1_OK
+ 2_OK
+ error

AlarmStatus
«enumeration»
+ On
+ Off

State Machine Implementation

State Variable Implementation

Response Generation

OutputHandler
+ alarmOn()
+ alarmOff()
+ lightOn()
+ lightOff()
Certification: Operational Definition

- Evidence that approved software engineering practices were used to produce the software
- Results of inspection and tests provide confidence that the as-built software will perform its intended function
DANSE Testing Strategy

- Dedicated Test Leaders for Framework and Each Science Team
- Statistical Testing to Certify Each Increment
- Increment Release Based on Quantitative Assessment of Test Results
Work Products Supporting Certification

- Peer Review Findings and Resolution
  - Specifications
  - Code
  - Test Plan
- Test Results and Resolution of Failures
- Release Decision Meeting Minutes
User Guide

Content
- How to install and/or configure the system
- How to initiate and control all functions
- How to interpret output

Format
- HTML for use with browser
Configuration/Release Control
Key Practices

- Document/Source control tool (Subversion)
- Centrally managed release tagging
- Repeatable automated build procedure
- Formal release procedure
- Change tracking (Trac)
- Audit trail
Configuration Items

- Software Behavior Specification
- Software Architecture Specification
- Source Code
- Software Test Plan
- Test Record
- User Guide
Change Request Processing

TL = Test Leader     CM = Configuration Manager
Roles of Trac and Subversion in the Increment Development Sequence

**Subversion Time Line**
- Move Tags for In-scope Features and Fixes
- Move Tags for Defect Corrections

**Trac Time Line**
- Tickets Pace Feature Additions
- Tickets Drive Defect Correction

**Feature Tickets**
- Resolved/Closed
- Request Resolved
- Ticket Verification

**Milestone Tickets**
- Closed/Deferred

**Defect Tickets**
- Resolved/Closed
- Request Resolved
- Ticket Verification

**Milestone Tickets**
- Closed/Deferred

Copyright © 2006 Software Quality Research Laboratory
Process Summary

- Rigorous specification of code derived from informal requirements
  - Architecture
  - Behavior
- Certification
  - Independent Work Product Review
  - Quantitative Testing
- Configuration management
  - Centralized build/release control
  - Comprehensive change tracking