

DANSE Software Quality Assurance

Tom Swain

Software Quality Research Laboratory

University of Tennessee
Department of Computer Science

SQA WBS

3.2.4.1 Software Quality Engineering

3.2.4.2 SNS Interface Requirements

3.2.4.3 SNS Integration Plan

3.2.4.4 Software Testing

3.4.2.1

Software Quality Engineering

- Write Software Quality Engineering Guidelines
 - Done
- Present 4 Software Quality Engineering Workshops, focused on Specific DANSE Tasks
 - To be scheduled
 - General preparation done

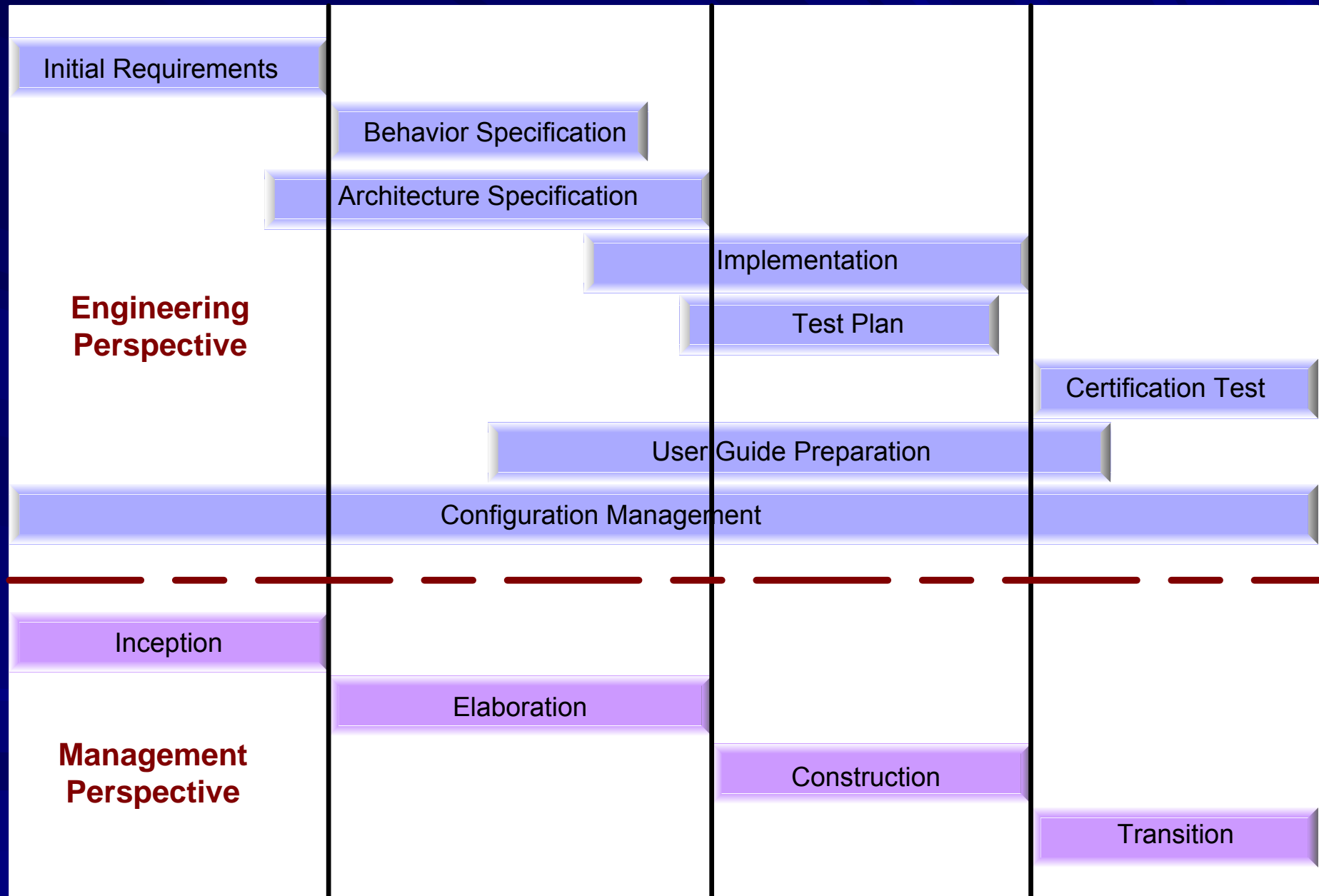
Software Quality Engineering Responsibilities

- ❑ Define development process procedures and standards (see <http://danse.us/trac/sqa>)
- ❑ Recommend methods and tools to support productivity and process compliance
- ❑ Perform periodic audits

Process Summary

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

Essential Process Elements



Work Product Completion Criteria

- ❑ Planning Review
 - Initial Requirements
- ❑ Peer Review
 - Behavior Specification
 - Architecture Specification
- ❑ Code Review
 - Software Implementation
- ❑ Certification Test
 - Software Release

Certification Protocol

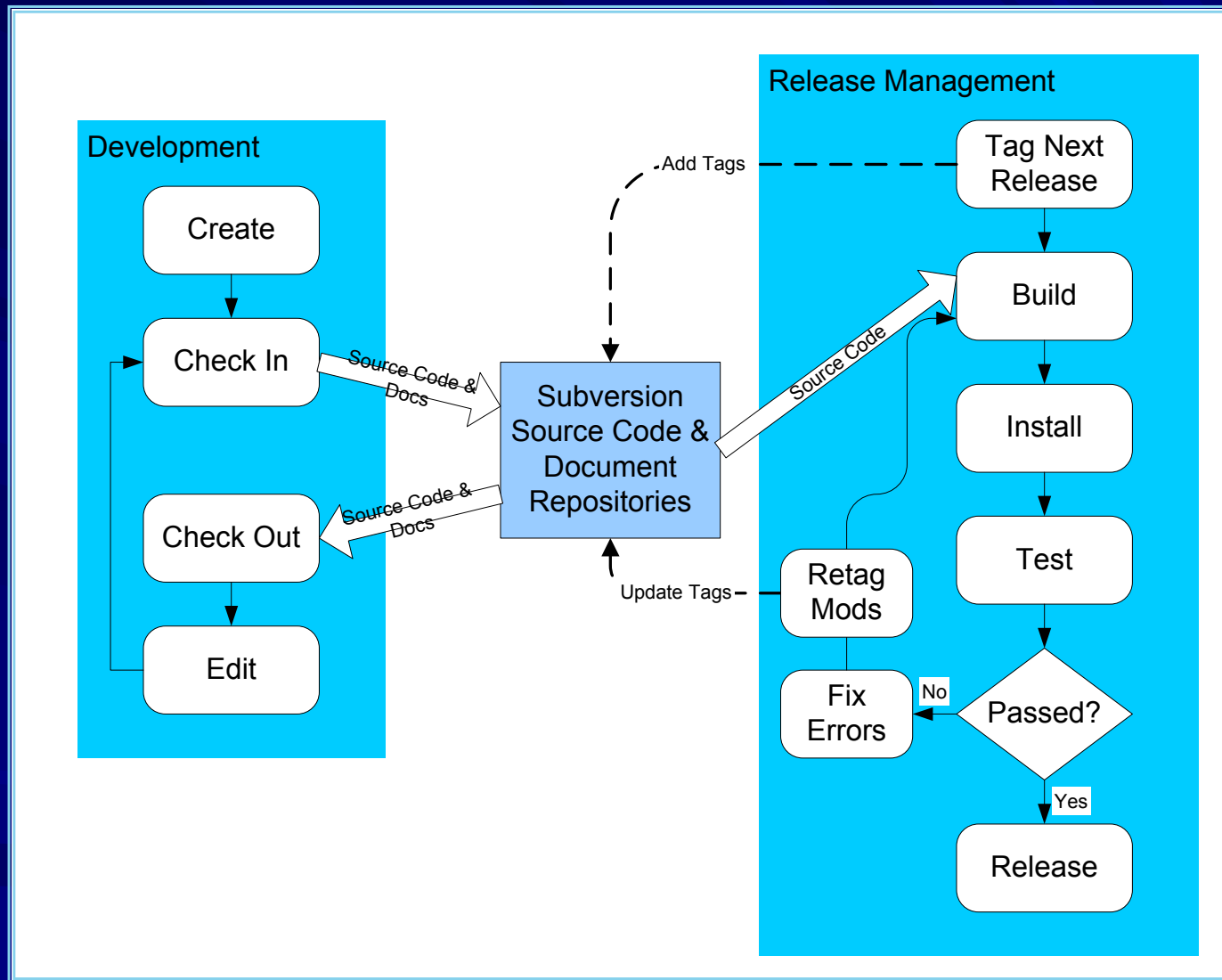
- ❑ Architecture and Behavior Specs updated to as-built design – peer review action items resolved
- ❑ Code inspection complete – action items resolved
- ❑ Certification testing complete – defect correction for all failures resolved
- ❑ Beta test complete* – reported issues resolved

*optional for all except final increment

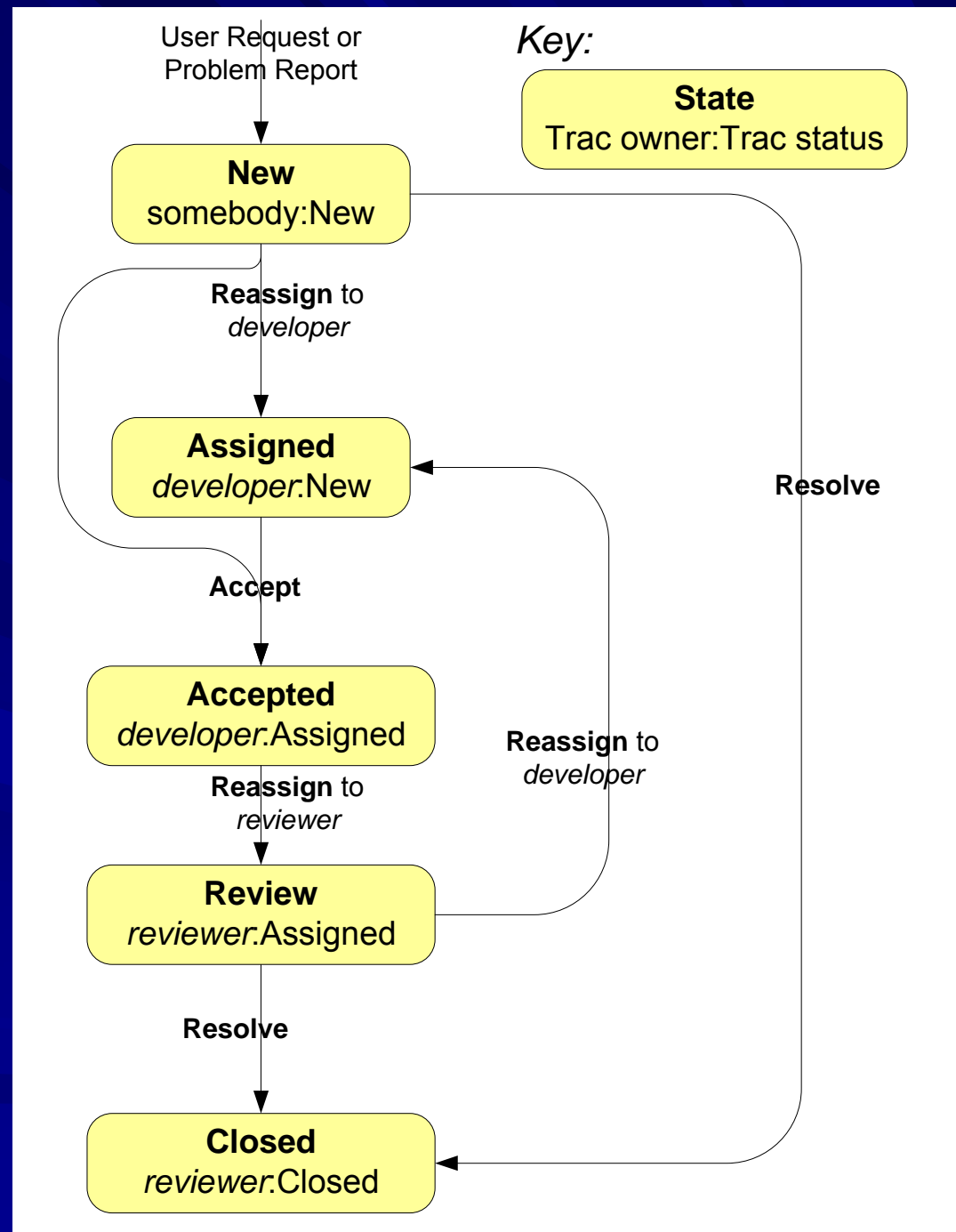
Software Quality Engineering Responsibilities

- ❑ Define development process procedures and standards (see <http://danse.us/trac/sqa>)
- ❑ **Recommend methods and tools to support productivity and process compliance**
- ❑ Perform periodic audits

Software Configuration Management

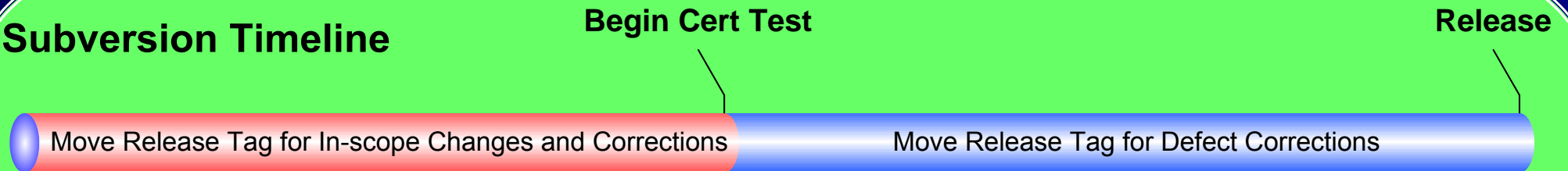


Trac Ticket Lifecycle

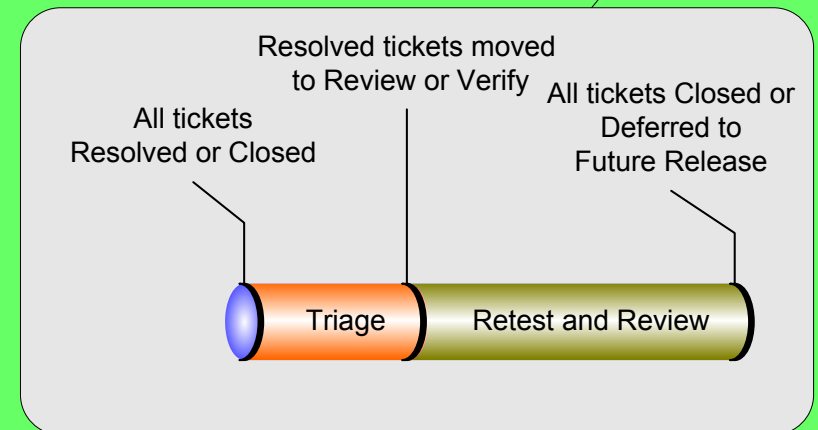
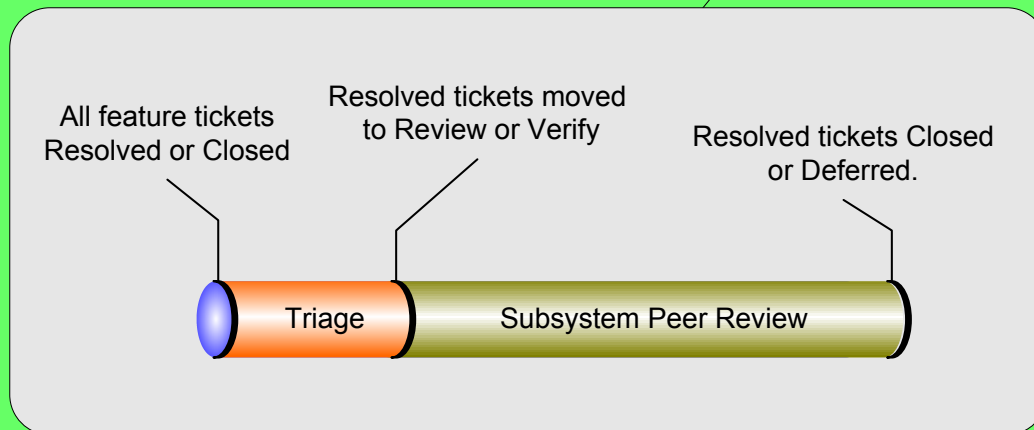
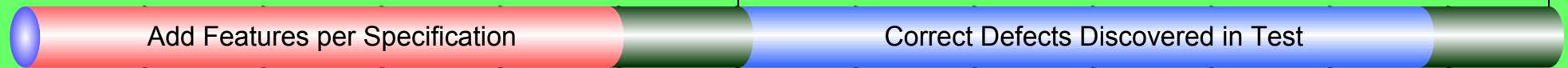


Roles of Trac and Subversion in the Development Sequence

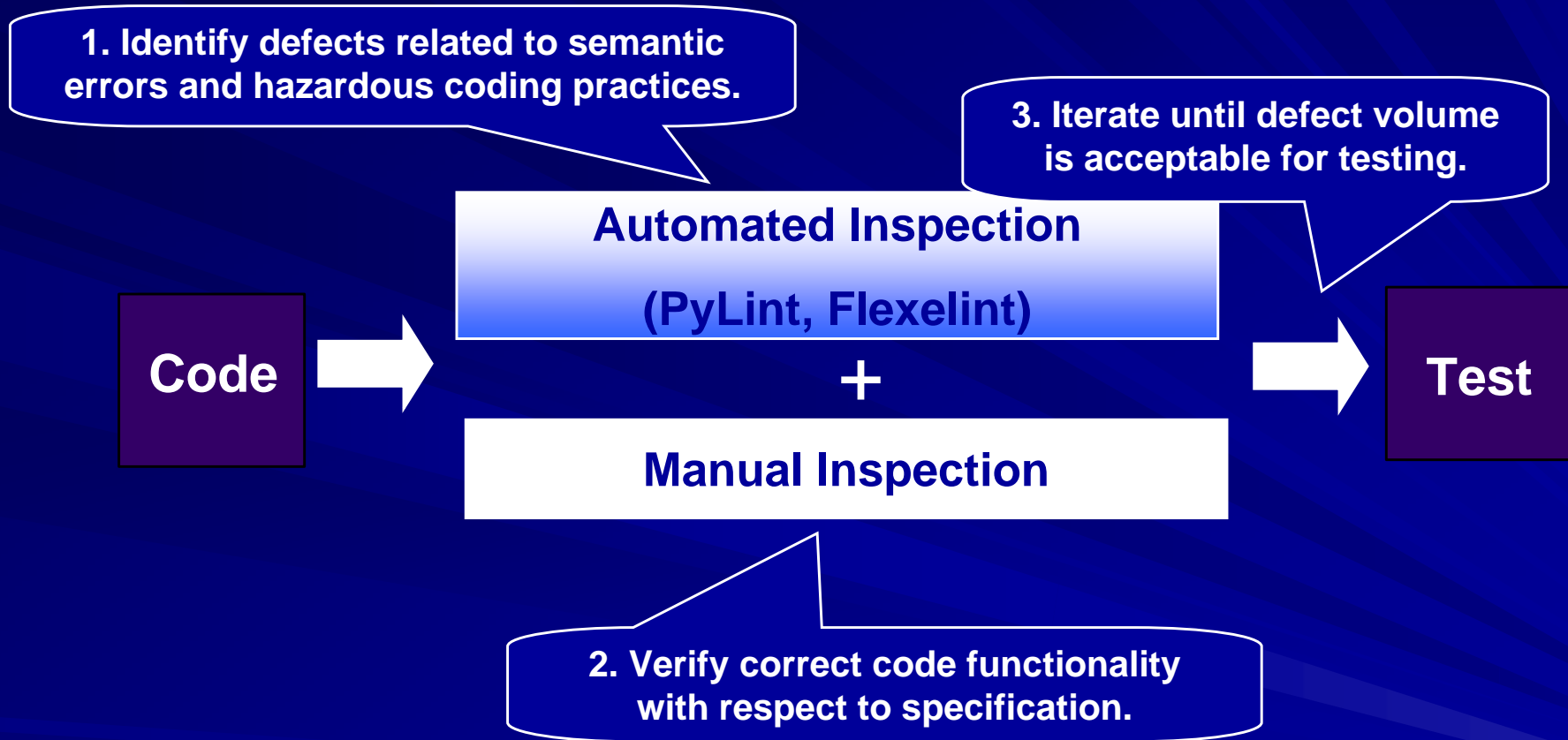
Subversion Timeline



Trac Timeline



Manual and Automated Code Inspection



Software Quality Engineering Responsibilities

- ❑ Define development process procedures and standards (see <http://danse.us/trac/sqa>)
- ❑ Recommend methods and tools to support productivity and process compliance
- ❑ **Perform periodic audits**

Quick-look Audit Results: Areas for Improvement

- ❑ Requirements Definition
- ❑ Behavior Specification
- ❑ Architecture Specification
- ❑ Code Annotation (Doxygen)
- ❑ Certification Test Planning
- ❑ Design Review

Toward Better Requirements

(Use Cases) U (Functional Requirements)

- ❑ Express as stimulus/condition/response action statement

“When the user does X while Y is true, then the system output is Z.”

- ❑ Collect all in a single work product
- ❑ Itemize and tag for traceability

Toward Better Requirements

Supplementary Requirements

- ❑ **Nonfunctional** requirements only
- ❑ Should be **measurable**
- ❑ If not measurable, then **objectively verifiable**
- ❑ Otherwise, don't bother

Behavior Specification (currently non-existent)

- 1) Establish system boundary.
- 2) Define human/software/hardware interfaces.
- 3) Itemize stimuli.
- 4) Itemize responses.
- 5) Define response for every stimulus sequence.

Architecture Specifications

(currently too implicit and incomplete)

- ❑ Define **components** and their **responsibilities**
- ❑ Specify **relationships** among components
- ❑ Specify intra-system and external **interfaces**
- ❑ Ensure **unidirectional dependency hierarchy**
- ❑ Specify assumptions regarding **platform/environment**

DANSE Architecture Specification Recommendation

- ❑ Overview class diagram(s) with summary description of each class/component on Doxygen main page
- ❑ Detailed interface specifications embedded in code as Doxygen annotation

Specification for Functions & Static Processes (using Doxygen)

- 1) For arguments and return values, specify **type, definition, units, valid range, and default value**
- 2) Partition input space into domains
- 3) Specify response function for **EVERY** domain

Required Testing

(missing in some planning documents)

❑ Certification Test

- ❑ Model system
- ❑ Generate test cases
- ❑ Define test oracle(s)
- ❑ Execute test
- ❑ Evaluate results

❑ Beta Test

- ❑ Plan
- ❑ User Guide
- ❑ User training
- ❑ User feedback
- ❑ Completion criteria

Design Review Essentials

- ❑ Reviewable work product checked into SVN
- ❑ Notify reviewers 2+ days before review
- ❑ Review formats
 1. “Walk-thru” meeting
 2. Issue resolution meeting
 3. E-review by individuals
- ❑ Reviewers confirm correctness and traceability
- ❑ Record each action item as Trac ticket
- ❑ Resolve action items (tickets)
- ❑ Update work product in SVN

Software Engineering Workshop Topics

Engineering Diffraction Focus

CNMS Room L382, Wed, Jan 24

- Better and Easier Requirements

- Systematic transition:

Requirements → Behavior Spec → Code

- Architecture spec standards

- Model-based Testing How-to

Quality Challenges

- ❑ New, large, distributed development team
- ❑ Wide range of software engineering experience
- ❑ Expectations comparable to industrial product line
 - ❑ Reliable, intuitive operation for novice users
 - ❑ Straightforward extensibility for power users
 - ❑ Long term maintenance by SNS

