Software Quality Group of New England

How to Build Quality into Software in

6 Easy Steps

Joseph Zec
Associate Director – Technology QA & Compliance
Shire
Joseph.Zec@gmail.com
March 8, 2017

1

Agenda

"It's harder than you might think to squander millions of dollars, but a flawed software development process is a tool well suited to the job."

- Software Development Life Cycles
- Quality activities defined and modeled
- Quality activities throughout the Life Cycle
  - Requirements
  - Design
  - Coding
  - Testing
  - Release
- Life Cycle Controls

2

Introduction

A robust software development process produces robust software.

- A life cycle identifies:
  - Sequencing and dependencies
  - Roles and responsibilities
  - Controls
- A life cycle produces predictable activities and outputs

3

Software Development Life Cycles

Common structure of life cycles

- Stages
  - Development
  - Maintenance/Sustaining
  - Retirement
- Development stage usually divided into phases
  - Concept/Planning/Requirements
  - Design
  - Coding/Construction
  - Testing
  - Release

4

A Definition of Quality

A conclusion that software has the attribute known as "Quality" is highly dependent upon comprehensive software testing, inspections, analyses, and other verification tasks performed at each stage of the software development life cycle.

5

Development tasks

Every SDLC contains common software development tasks

- Requirements
- Design
- Coding
- Testing
- Release

6

A bit more on testing

Testing is only one part of the software quality equation!

- Testing is an example of a "verification methodology"
- Different verification methodologies are applicable at different stages of the SDLC

7

Verification Methodologies

Static verification techniques are executed without running the code

- Reviews
  - Design review
  - Code review
- Inspections
- Analyses
  - Risk analysis
  - Traceability analysis

8

Verification Methodologies

Dynamic verification techniques are executed while the code is running

- Testing
  - Unit testing
  - User acceptance testing
- Code coverage analysis

9

Development and Verification Married

Requirements	Requirements review, Risk analysis, Trace analysis
Design	Design review, Risk analysis Trace analysis
Coding	Code review, Unit test, Integration test, Trace analysis
Test development	Test review, Trace analysis, Code coverage analysis
Test execution	System test, User acceptance test
Release	Installation test, System inspection

10

Requirements Phase

Verification activities associated with requirements

- Review
- Risk analysis
  - What types and severities of risk are associated with each requirement?
  - What do we do about them?
- Traceability analysis
  - Traceability from user to software requirements
  - Traceability from software requirements to the rest of the system

11

Requirements Review

A good requirement…

- specifies "what" not "how"
- concisely describes a single behavior
- is self-contained
- is complete
- is consistent with other requirements
- is unambiguous
- is verifiable
- is non-negative
- is traceable (non-compound, unique ID)
- is written in formal "requirements-speak"
- is quantified where possible
- is feasible
- is reviewed and approved

12

Requirements Risk Analysis

Factors to account for in risk analysis:

- What types of risks exist?
  - Human safety, customer satisfaction, business, etc.
- What impacts could they have?
  - Catastrophic, major, moderate, minor, etc.
- How often could they occur?
  - Frequently, seldom, rarely, etc.
- How can we mitigate their effects?
  - New requirements, design, etc.

13

Requirements Traceability Analysis

Traces user requirements to software requirements

- Ensures that every user requirement that is implemented in software is accounted for
- Ensures that every software requirement is justified

14

Design Phase

Verification activities associated with design

- Review
- Risk analysis
  - Each design decision can add risk, modify risk, or control risk
- Traceability analysis
  - Traceability from software requirements to design
  - Traceability from design to the rest of the system

15

Design Review

A good design…

- correctly implements software requirements
- complies with existing design standards
- complies with project plans
- is maintainable
- is documented
- is approved
- is complete when nothing else can be taken away

16

Design Risk Analysis

Factors to account for in design risk analysis:

- How do design decisions affect existing risk:
  - increase existing risk?
  - reduce existing risk?
  - add new risk?
- Design is most often used as a risk control measure
  - Implement risk treatments identified during requirements risk analysis

17

Design Traceability Analysis

Traces software requirements to design

- Ensures that every software requirement is accounted for
- Ensures that every design element is justified

18

Coding Phase

Verification activities associated with coding

- Review
- Testing
  - Unit test
  - Integration test
- Traceability analysis
  - Traceability from design to source code
  - Traceability from source code to unit and integration tests

19

Code Review

First a random question…

Does anyone NOT know what a Klingon is?

20

Klingon SQA

"I have challenged the entire Quality Assurance team to a Bat-Leh contest! They will not concern us again."

21

Code Review

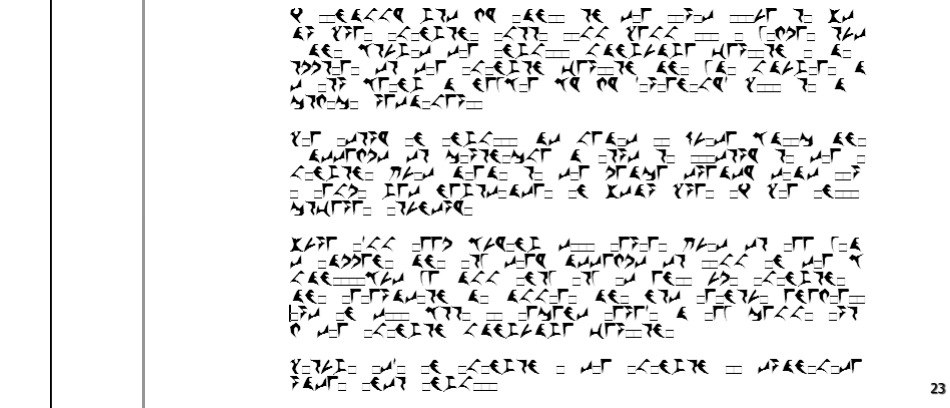
Good source code…

- correctly implements the design
- complies with existing coding standards
  - Naming conventions
  - Presentation standards
  - Coding conventions
- complies with project plans
- is maintainable
- is documented (commented)

22

Code Standards

Example of Klingon source code comments



23

Code Standards

"A TRUE Klingon warrior does not comment his code."

24

Code Standards

A Klingon programmer who does not apply coding standards may find himself out of a job

25

Code Standards

- Naming conventions
  - Rules for naming variables and constants
  - Rules for naming functions, procedures, methods, and their parameters
  - Rules for naming data structures
  - Rules for source code filenames
- Develop your own standard or rely on an industry-standard convention
  - Hungarian Notation
  - CamelCase

26

Naming Conventions

"Klingon function calls do not have 'parameters' – they have 'arguments' – and they ALWAYS WIN THEM."

Code Standards

- Presentation standards
  - Rules for consistent indentation
  - Rules for commenting
    - In-line comments
    - Module/function headers
  - Rules that foster clarity
- Need to balance standardization with individual style

28

Presentation Standards

"Indentation?! I will show you how to indent when I indent your skull!"

29

Code Standards

- Coding conventions
  - Complexity management
    - Nesting
  - Exception handling
- Should encourage robust coding techniques
  - Always have a default at the end of a CASE statement
- Should discourage dangerous coding techniques
  - Do not use GOTO
  - Do not use dynamic memory allocation
  - Do not perform pointer arithmetic
- Should allow for exceptions to coding conventions
  - Justified, reviewed, and approved

30

Coding Conventions

"You question the worthiness of my code? I should kill you where you stand!"

31

Unit and Integration Test

- Developer debugs the code
  - Can use documented test cases or not
  - Level of formality can suit the situation
- Don't debug code that hasn't been reviewed yet
- Number of levels of integration testing can be adjusted to meet the complexity of the code

32

Unit and Integration Test

"Debugging? Klingons do not debug. Our software does not coddle the weak."

33

Code Traceability Analysis

- Traces design to source code
  - Ensures that every design element is accounted for
  - Ensures that all source code is justified
- Traces source code to unit tests
  - Ensures that every unit is debugged

34

Test Development Phase

Verification activities associated with test development

- Review
- Traceability analysis
  - Traceability from software requirements to test cases
- Code coverage analysis
  - Measuring the comprehensiveness of the test cases

35

Test Development Explored

Good practices for test development

- Requirements-based test development
- Traceability analysis to assess coverage
- Enhance coverage by including various test types
- Measure coverage via code coverage analysis
- Review tests with right audience

36

Test Case Review

A good test case…

- specifies an unambiguous expected test outcome
- has a high probability of exposing an error
- examines both the usual and unusual case
  - Error and alarm conditions
  - Startup and shutdown
  - Potential operator errors
  - Maximum and minimum ranges of allowed values
  - Stress conditions
- produces documentation that permits an independent confirmation of the pass/fail status
- is traceable (unique ID)
- is reusable
- is developed by someone other than the developer
- is reviewed and approved

37

Test Case Traceability Analysis

Traces software requirements to test cases

- Ensures that every software requirement has at least one test

38

Test Case Coverage Analysis

- Test case coverage analysis is only possible when you have access to source code that has been instrumented by a coverage analyzer tool
- This is an extremely powerful method but is technically challenging and labor and time intensive
- There are many ways of measuring test case coverage

39

Test Case Coverage Analysis

- Statement coverage
- Decision (branch) coverage
- Condition coverage
- Multi-condition coverage
- Loop coverage
- Path coverage
- Data flow coverage

40

Test Execution Phase

Verification activities associated with test execution

- System and user acceptance testing
- Producing auditable documentation of test results
- Logging and resolving issues

41

Test Execution Explored

Good practices for test execution

- Document test results carefully
- Encourage exploratory testing
- Utilize a robust issue management process and tool

42

Test Case Execution

"By filing this bug report you have challenged the honor of my family. Prepare to die!"

43

Release Phase

Verification activities associated with software release

- Installation testing
- System inspection
- User training
- Logging and resolving issues

44

Release Phase

"What is this talk of 'release'? Klingons do not make software 'releases'. Our software **escapes**, leaving a bloody trail of designers and quality assurance people in its wake."

"Our users will know fear and cower before our software! Ship it! Ship it and let them flee like the dogs they are!"

45

Life Cycle Controls

Typical life cycle controls

- Milestones
- Change control
- Approvals

46

Life Cycle Controls

Milestones

- Phase end reviews
  - AKA "Quality Gates" or "Project Reviews"
  - Held at the end of each phase of the SDLC
- Requirements or design freeze
  - Helps control scope creep and covert design changes
- Establishing a system baseline
  - Makes changes more difficult

47

Life Cycle Controls

Change Control

- Timing of the introduction of formal change control is important
- Includes review and approval
- Includes independence of review
- Includes impact and risk assessments

48

Life Cycle Controls

Approvals

- Formal approval can be applied to deliverables and decisions
- Control of the SDLC should be in proportion to a project's complexity
- For more complex projects, require more approvals at higher levels in the organization

49

Thank you!

Questions?

50