Software Testing: The Big Picture

Dr. Paul West

Department of Computer Science
College of Charleston

January 9, 2014

Engineering Process

- Sophisticated tools
  - amplify capabilities
  - but do not remove human error
- Engineering disciplines pair
  - construction activities with
  - activities that check intermediate and final products
- Software engineering is no exception: construction of high quality software requires construction and verification activities

Verification and Design Activities

- Verification and design activities take various forms
  - Suited to highly repetitive construction of non-critical items for mass markets
  - Highly customized or highly critical products.
- Appropriate verification activities(tools) depend on
  - Engineering discipline
  - Construction process
  - Final product
  - Quality requirements

Peculiarities of Software

Software has some characteristics that make V&V particularly difficult:

- Many different quality requirements
- Evolving (and deteriorating) structure
- Inherent non-linearity
- Uneven distribution of fault

Example

If an elevator can safely carry a load of 1000 kg, it can also safely carry any smaller load; If a procedure correctly sorts a set of 256 elements, it may fail on a set of 255 or 53 or 12 elements, as well as on 257 or 1023.

Impact of New Technologies

- Advanced development technologies can reduce the frequency of some classes of errors
- But do not eliminate errors
- New development approaches can introduce new kinds of faults

Examples

- Deadlock or race conditions for distributed software
- New problems due to the use of polymorphism, dynamic binding and private state in object-oriented software.

Approaches

- There are no fixed recipes
- Test designers must
  - Choose and schedule the right blend of techniques
    - To reach the required level of quality
    - Within cost constraints
  - Design a specific solution that suits
    - The problem
    - The requirements
    - The development environment

Basic Questions

1. When do verification and validation start? When are they complete?
2. What particular techniques should be applied during development?
3. How can we assess the readiness of a product?
4. How can we control the quality of successive releases?
5. How can the development process itself be improved?

When to Start and Complete V&V

- Start creating Tests from the beginning
  - Execution of tests is a small part of the verification and validation process
- V&V start as soon as we decide to build a software product, or even before
- V&V last far beyond the product delivery as long as the software is in use, to cope with evolution and adaptations to new conditions

Early Start: From Feasibility Study

- The feasibility study of a new project must take into account the required qualities and their impact on the overall cost
- At this stage, quality related activities include
    - Risk analysis
    - Measures needed to assess and control quality at each stage of development.
    - Assessment of the impact of new features and new quality requirements
    - Contribution of quality control activities to development cost and schedule.

Maintenance

- Maintenance activities include
  - Analysis of changes and extensions
  - Generation of new test suites for the added functionalities
  - Re-executions of tests to check for non regression of software functionalities after changes and extensions
  - Fault tracking and analysis

Techniques

No single Analysis/Testing technique can serve all purposes
The primary reasons for combining techniques are:

- Effectiveness for different classes of faults
  example: analysis instead of testing for race conditions
- Applicability at different points in a project
  example: inspection for early requirements validation
- Differences in purpose
  example: statistical testing to measure reliability
- Tradeoffs in cost and assurance
  example: expensive technique for key properties

Readiness of a Product

- A&T during development aims at revealing faults
- We cannot reveal are remove all faults
- A&T cannot last indefinitely: we want to know if products meet the quality requirements
- We must specify the required level of dependability and determine when that level has been attained.

Dependability

- Availability measures the quality of service in terms of running versus down time
- Mean time between failures (MTBF) measures the quality of the service in terms of time between failures
- Reliability indicates the fraction of all attempted operations that complete successfully

Dependability Examples

Web application

- 50 interactions terminating with a credit card charge.
- The software always operates flawlessly up to the point that a credit card is to be charged, but on half the attempts it charges the wrong amount.

What is the reliability of the system?

- If we count the fraction of individual interactions that are correctly carried out, only one operation in 100 fail: The system is 99% reliable.
- If we count entire sessions, only 50% reliable, since half the sessions result in an improper credit card charge

Assessing Dependability

- Randomly generated tests following an operational profile
- Alpha test: tests performed by users in a controlled environment, observed by the development organization
- Beta test: tests performed by real users in their own environment, performing actual tasks without interference or close monitoring

Software Quality

- Software test and analysis does not stop at the first release.
- Software products operate for many years, and undergo many changes:
  - They adapt to environment changes
  - They evolve to serve new and changing user requirements.
- Quality tasks after delivery
  - test and analysis of new and modified code
  - re-execution of system tests
  - extensive record-keeping

Process Improvement

- The same defects are encountered in project after project
- A third goal of the improving the quality process is to improve the process by
  - Identify and remove weaknesses in the development process
  - Identify and remove weaknesses in test and analysis that allow them to remain undetected

Four Step Process Improvement

1. Define the data to be collected and implementing procedures for collecting them
2. Analyze collected data to identify important fault classes
3. Analyze selected fault classes to identify weaknesses in development and quality measures
4. Adjust the quality and development process

Process Improvement Example

1. Faults that affect security were given highest priority
2. During A&T we identified several buffer overflow problems that may affect security
3. Faults were due to bad programming practice and were revealed late due to lack of analysis
4. Action plan: Modify programming discipline and environment and add specific entries to inspection checklists

Summary

- The quality process has three different goals:
  - Improving a software product
  - assessing the quality of the software product
  - improving the quality process
- We need to combine several A&T techniques through the software process
- A&T depend on organization and application domain.
- Cost-effectiveness depends on the extent to which techniques can be re-applied as the product evolves.
- Planning and monitoring are essential to evaluate and refine the quality process.

Chapter 1 Assignment

Choose and complete any two Chapter 1 Exercises (pg 13-14)
Due in the dropbox by January 23, 2014 2359