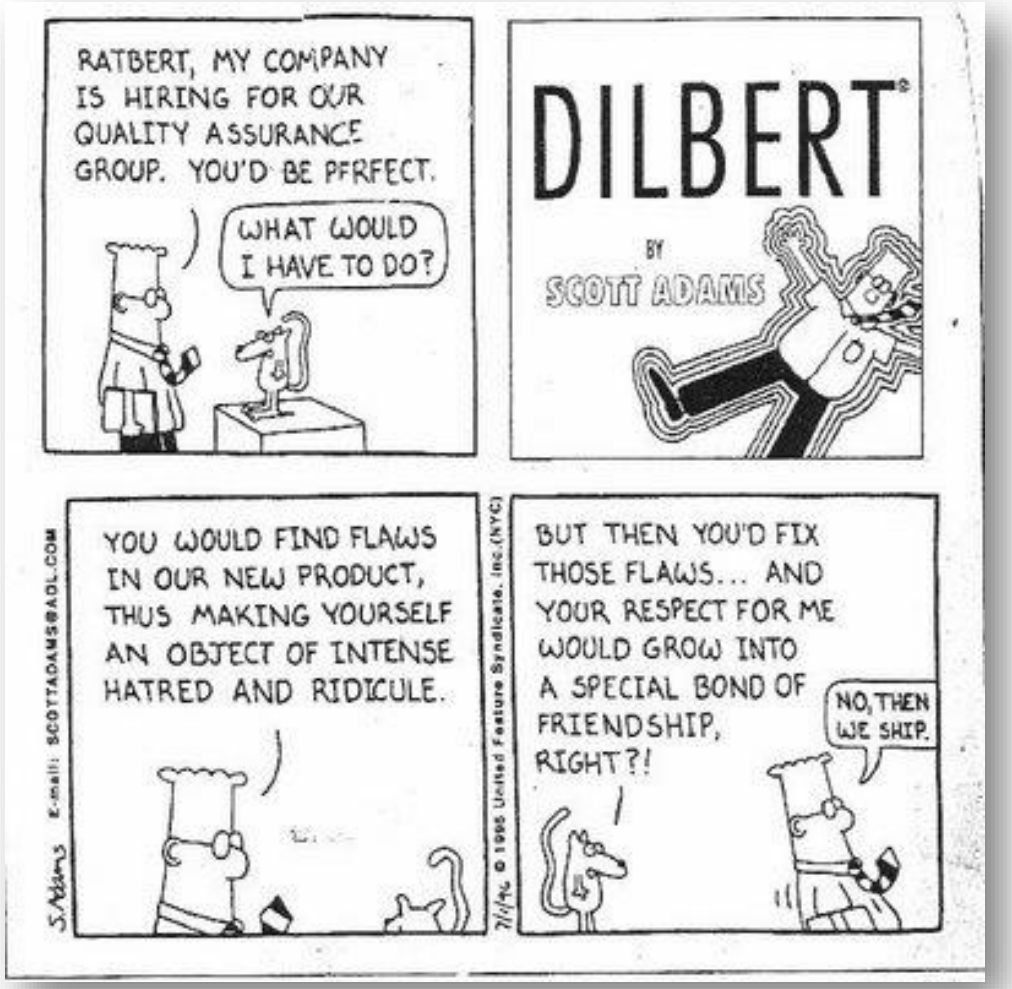
Introduction to Systematic Testing I



Introduction to Systematic Testing I

Outline

- Today we begin a long, deep look at **software testing**
- We begin with:
  - Definitions: what is software testing?
  - Role of specifications
  - Levels of testing: unit, integration, system, acceptance

What is Testing?

Testing

- Testing is the process of executing software in a controlled manner, to answer the question:

  "does the software behave as specified?"

- Implies that we have a specification (or possibly the tests are it)
- (or) Implies that we have some property we wish to test for independently of the specification
  - e.g., *"all paths in the code are reachable (no dead code)"*
- Testing is often associated with the words validation and verification

Verification vs. Validation

Verification

- Verification is the checking or testing of software (or anything else) for conformance and consistency with a given specification – answers the question:

"Are we doing the job right?"

Validation

- Validation is the process of checking that what has been specified is what the user actually wanted – answers the question:

"Are we doing the right job?"

Verification vs. Validation

Verification

- Testing is most useful in verification – checking software (or anything else) for conformance and consistency with a given specification,
- However, testing is just one part of it analysis – inspection and measurement are also important

Validation

- Checking that what has been specified is what the user actually wanted usually involves meetings, reviews and discussions
- Testing is *less* useful in validation, although it can have a role

Debugging vs. Testing

Debugging is not Testing!

Debugging:
the process of analyzing and locating bugs when the software does not behave as expected.

Testing:
the process of methodically searching for and exposing bugs (not just fixing those that happen to show up) – much more comprehensive.

- Debugging therefore supports testing, but cannot replace it
- However, no amount of testing* is guaranteed to find all bugs

* except possibly exhaustive testing (where practical)

What is Systematic Testing?

Systematic Testing

- Systematic testing is an explicit discipline or procedure (a system) for:
  - choosing and creating test cases
  - executing the tests and documenting the results
  - evaluating the results, possibly automatically
  - deciding when we are done (enough testing)
- Because in general it is impossible to ever test completely, systematic methods choose a particular point of view, and test only from that point of view (the test criterion)
  - e.g., test only that every decision (if statement) can be executed either way

The Role of Specification

The Need for Specification

- Validation and verification activities, such as testing, cannot be meaningful unless we have a specification for the software
- The software we are building could be a single module or class, or could be an entire system
- Depending on the size of the project and the development methods, specifications can range from a single page to a complex hierarchy of interrelated documents

Levels of Specifications

- There are usually at least **three levels** of software specification documents in large systems:
  1. Functional specifications (or requirements) give a precise description of the required behaviour of the system – *what the software should do, not how it should do it – may also describe constraints on how this can be achieved*
  2. Design specifications describe the architecture of the design to implement the functional specification – *the components of the software and how they are to relate to one another*
  3. Detailed design specifications describe how each component of the architecture is to be implemented – *down to the individual code units*

Levels of Testing

- Given the hierarchy of specifications, it is usual to structure testing into **three** (or more) corresponding levels:
  3. Unit testing addresses the verification that individual components meet their detailed design specification
  2. Integration testing verifies that the groups of units corresponding to architectural elements of the design specification can be integrated to work as a whole
  1. System testing verifies that the integrated total product has the functionality specified in the functional specification
- To these levels it is usual to add the additional test level:
  0. Acceptance testing, in which the actual customers validate that the software meets their real intentions as well as what has been functionally specified, and accept the result

Tests as Goals

An Integral Task

- Once each level of specification is written, the next step is to write the tests for that level
  (Test-driven development speeds this by making the tests themselves the specification)
- It is important that the tests be designed without knowledge of the software implementation
  (e.g., in Test-driven development, before implementation)
- Otherwise we are tempted to simply test the software for what it actually does, not what it should do

Using Tests

Evaluating Tests

- Within each level of testing, once the tests have been applied, test results are evaluated
- If a problem is encountered, then either:
  a) the tests are revised and applied again, if the tests are wrong, or
  b) the software is fixed and the tests are applied again, if the software is wrong
- In either case, the tests are applied again, and so on, until no more problems are found.
- Only when no more problems are found does development proceed to the next level of testing

Test Evolution

Tests Don't Die!

- Testing does not end when the software is accepted by the customer
- Tests must be repeated, modified and extended to insure that no existing functionality has been broken, and that any new functionality is implemented according to the revised specifications and design
- Maintenance of the tests for a system is a major part of the effort to maintain and evolve a software system while retaining a high level of quality
- In order to make this continual testing practical, automation plays a large role in software testing methods

Summary

Summary

- Testing addresses primarily the verification that software meets it specifications – without some kind of specification, we cannot test
- Testing is done at several levels, corresponding the the levels of functional, design, and detailed design specifications in reverse order
- Testing remains for the life of the software system

References

- Sommerville ch. 22 (22.1)

Next Time

- More on Systematic Testing