Testing (revisited) & Release

V & V

- *Verification* refers to the set of tasks that ensure that software correctly implements a specific function.
- *Validation* refers to a different set of tasks that ensure that the software that has been built is traceable to customer requirements. Boehm [Boe81] states this another way:
  - *Verification:* "Are we building the product right?"
  - *Validation:* "Are we building the right product?"

Testing Phases

- Unit Testing
  - Developer tests individual modules
  - Usually glass box
- Integration testing
  - Put modules together, try to get them working together
  - Integration testing is complete when the different pieces are able to work together
- System testing
  - Black-box testing of entire deliverable against specs
- ***Acceptance testing***
  - Testing against user needs, often by the user

Inspecting compared to testing

- Both testing and inspection rely on different aspects of human intelligence.
- Testing can find defects whose consequences are obvious but which are buried in complex code.
- Inspecting can find defects that relate to maintainability or efficiency.
- The chances of mistakes are reduced if both activities are performed.

Testing or inspecting, which comes first?

- It is important to inspect software *before* extensively testing it.
- The reason for this is that inspecting allows you to quickly get rid of many defects.
- If you test first, and inspectors recommend that redesign is needed, the testing work has been wasted.
  - There is a growing consensus that it is most efficient to inspect software before *a*ny testing is done.
- Even before developer testing

The test-fix-test cycle

- When a failure occurs during testing:
  - Each failure report is entered into a failure tracking system.
  - It is then screened and assigned a priority.
  - Low-priority failures might be put on a *known bugs list* that is included with the software's *release notes*.
  - Some failure reports might be merged if they appear to result from the same defects.
  - Somebody is assigned to investigate a failure.
  - That person tracks down the defect and fixes it.
  - Finally a new version of the system is created, ready to be tested again.

The ripple effect

- There is a high probability that the efforts to remove the defects may have actually added new defects
  - The maintainer tries to fix problems without fully understanding the ramifications of the changes
  - The maintainer makes ordinary human errors
  - The system *regresses* into a more and more failure-prone state

Regression testing

- It tends to be far too expensive to re-run every single test case every time a change is made to software.
- Hence only a subset of the previously-successful test cases is actually re-run.
- This process is called *regression testing.*
  - The tests that are re-run are called regression tests.
- Regression test cases are carefully selected to cover as much of the system as possible.

- **The "law of conservation of bugs":**
  - ***The number of bugs remaining in a large system is proportional to the number of bugs already fixed***

Deciding when to stop testing

- All of the level 1 (“critical”) test cases must have been successfully executed.
- Certain pre-defined percentages of level 2 and level 3 test cases must have been executed successfully.
- The targets must have been achieved and are maintained for at least two cycles of ‘builds’.
  - A *build* involves compiling and integrating all the components.
  - Failure rates can fluctuate from build to build as:
    - Different sets of regression tests are run.
    - New defects are introduced.

Who Tests the Software?

developer

Understands the system
but, will test "gently"
and, is driven by "delivery"

independent tester

Must learn about the system,
but, will attempt to break it
and, is driven by quality

The roles of people involved in testing

- The first pass of unit and integration testing is called *developer testing*.
  - Preliminary testing performed by the software developers who do the design.
- *Independent testing* may be performed by separate group.
  - They do not have a vested interest in seeing as many test cases pass as possible.
  - They develop specific expertise in how to do good testing, and how to use testing tools.

Test planning

- Decide on overall test strategy
  - What type of integration
  - Whether to automate system tests
  - Whether there is an independent test team
- Decide on the coverage strategy for system tests
  - Compute the number of test cases needed
- Identify the test cases and implement them
  - The set of test cases constitutes a “test suite”
  - May categorize into critical, important, optional tests (level 1, 2, 3)
- Identify a subset of the tests as regression tests

Testing performed by users and clients

- *Alpha testing*
  - Performed by the user or client, but under the supervision of the software development team.
- *Beta testing*
  - Performed by the user or client in a normal work environment.
  - Recruited from the potential user population.
  - An *open beta release* is the release of low-quality software to the general population.
- *Acceptance testing*
  - Performed by users and customers.

Packaging for Delivery

- Software we deliver to the user must include
  - Executable in a convenient format e.g. EXE, JAR file
  - Release notes
  - User documentation: instructions on usage
    - Tutorials, user manuals, "getting started" instructions
  - Installation instructions
- May create "installables"
  - Compressed packages e.g. zip files, tar files
  - Scripts that automate installation procedures

Cross-team Testing

- In this project, we will do cross-team testing
  - Each team will serve as “independent test team” for another team
- Package your R2 software for delivery to other team
  - Create a JAR file, README with execution instructions, requirements doc and acceptance test doc
  - Designate someone in your team to serve as support contact
- Run acceptance test cases on received software, check for pass/fail
  - If failed, note down what inputs were and how it failed
- Perform any other testing you consider necessary
  - Can add missed test cases to acceptance test doc
  - Can include any other comments at bottom of acceptance test doc
- Deliver cross-team test report to other team
- Objective: To test the software thoroughly, help other team produce better software