Software Test and Analysis in a Nutshell
Learning objectives

• View the “big picture” of software quality in the context of a software development project and organization:
• Introduce the range of software verification and validation activities
• Provide a rationale for selecting and combining them within a software development process.
Engineering processes

• 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 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 faults

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.
Variety of 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
Five 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?
1: When do V&V start? When are they complete?

- Test is not a (late) phase of software development
  - 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
## Staging A&T techniques

<table>
<thead>
<tr>
<th>Requirements Elicitation</th>
<th>Requirements Specification</th>
<th>Architectural Design</th>
<th>Detailed Design</th>
<th>Unit Coding</th>
<th>Integration &amp; Delivery</th>
<th>Maintenance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identify qualities</td>
<td>Plan acceptance test</td>
<td>Plan system test</td>
<td>Plan unit &amp; integration test</td>
<td>Monitor the A&amp;T process</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Validate specifications</td>
<td>Analyze architectural design</td>
<td>Inspect architectural design</td>
<td>Inspect detailed design</td>
<td>Code inspection</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Generate system test</td>
<td>Generate integration test</td>
<td>Generate unit test</td>
<td>Generate regression test</td>
<td>Update regression test</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design scaffolding</td>
<td>Design oracles</td>
<td>Execute unit test</td>
<td>Analyze coverage</td>
<td>Generate structural test</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Execute integration test</td>
<td>Execute system test</td>
<td>Execute acceptance test</td>
<td>Execute regression test</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Collect data on faults</td>
<td>analyze faults and improve the process</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3: How can we assess the readiness of a product?

- A&T during development aim at revealing faults
- We cannot reveal or 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.
Different measures of 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.
Example of different dependability measures

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.
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.
A Framework for Testing and Analysis
Learning objectives

• Introduce dimensions and tradeoff between test and analysis activities
• Distinguish validation from verification activities
• Understand limitations and possibilities of test and analysis
Verification and validation

• Validation:
  does the software system meets the user’s real needs?
  \textit{are we building the right software?}

• Verification:
  does the software system meets the requirements specifications?
  \textit{are we building the software right?}
Validation and Verification

Validation
Includes usability testing, user feedback

Verification
Includes testing, inspections, static analysis

Actual Requirements → SW Specs → System
Verification or validation depends on the specification

Example: elevator response

Unverifiable (but validatable) spec: ... if a user presses a request button at floor i, an available elevator must arrive at floor i soon...

Verifiable spec: ... if a user presses a request button at floor i, an available elevator must arrive at floor i within 30 seconds...
Validation and Verification Activities

Actual Needs and Constraints -> User Acceptance (alpha, beta test) -> Delivered Package

System Specifications

Subsystem Design/Specs

Unit/Component Specs

System Test

Analysis / Review

Integration Test

Analysis / Review

Module Test

User review of external behavior as it is determined or becomes visible

Review

validation

verification

(c) 2007 Mauro Pezzè & Michal Young
You can’t always get what you want

Correctness properties are undecidable
the halting problem can be embedded in almost every property of interest
Getting what you need ...

- optimistic inaccuracy: we may accept some programs that do not possess the property (i.e., it may not detect all violations).
  - testing
- pessimistic inaccuracy: it is not guaranteed to accept a program even if the program does possess the property being analyzed
  - automated program analysis techniques
- simplified properties: reduce the degree of freedom for simplifying the property to check
Summary

• Most interesting properties are undecidable, thus in general we cannot count on tools that work without human intervention.

• Assessing program qualities comprises two complementary sets of activities: validation (does the software do what it is supposed to do?) and verification (does the system behave as specified?)

• There is no single technique for all purposes: test designers need to select a suitable combination of techniques.