Software Test and Analysis in a Nutshell

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.

How software is different?

- Software is soft and intangible
- There are no physical laws underlying software behaviour
- Software are never wears out
  - traditional reliability measures don’t apply
- The specification for software continuously changes

Peculiarities of software

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

- Many different quality requirements
  - Question: what are software qualities?
- 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

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.

Long lasting: beyond 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

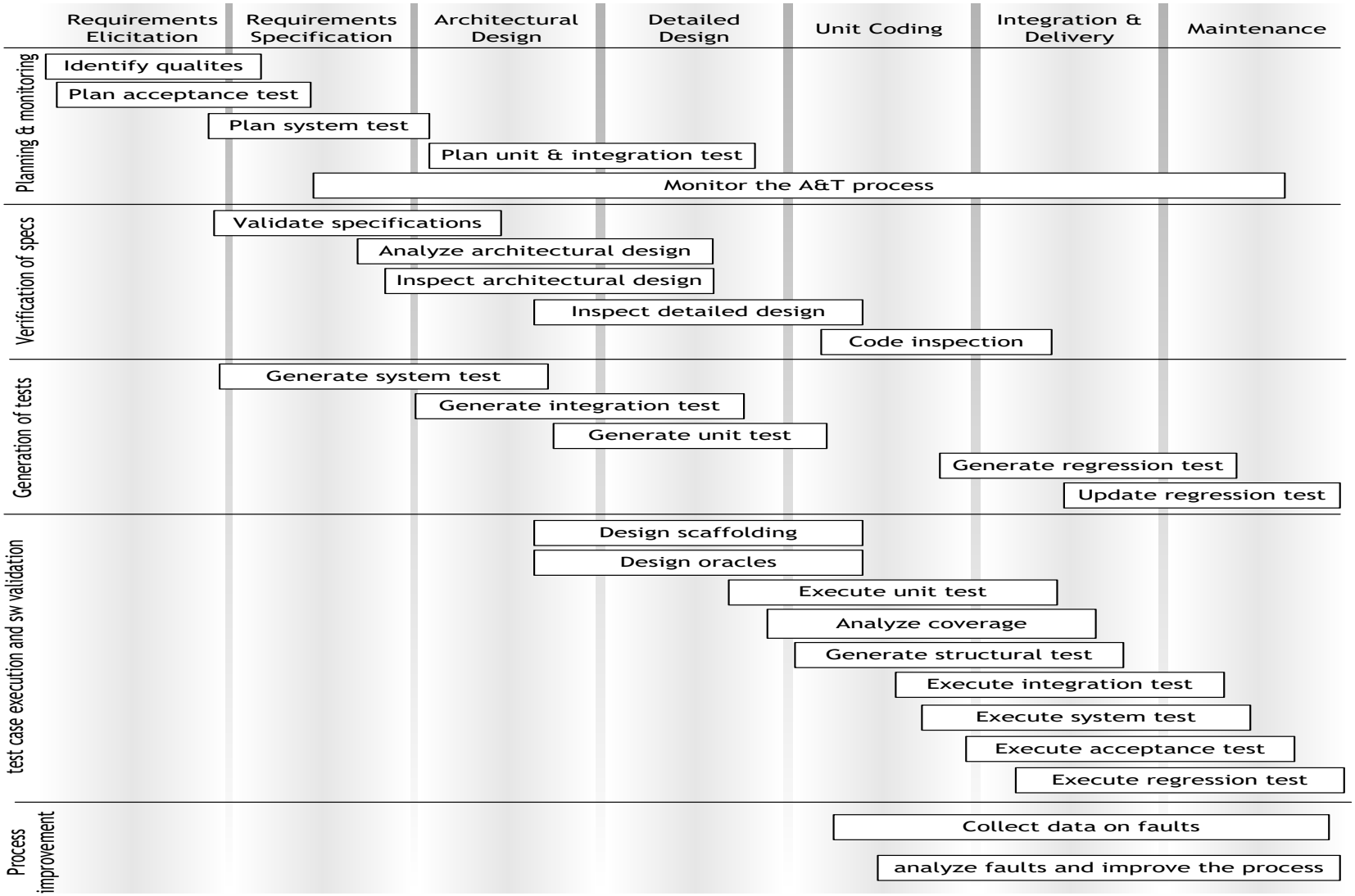
2: What particular techniques should be applied during development?

No single A&T 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

Staging A&T techniques



Example of unit testing

JUnit Testcases in Java: Simple JUnit 4.x Tutorial Hello World Example

```java
public class CrunchifyJunitTest {

    @Test
    public void testingCrunchifyAddition() {
        assertEquals("Here is test for Addition Result: ", 300, addition(27, 3));
    }

    @Test
    public void testingHelloWorld() {
        assertEquals("Here is test for Hello World String: ", "Hello -- World", helloWorld());
    }

    public int addition(int x, int y) {
        return x + y;
    }

    public String helloWorld() {
        String helloWorld = "Hello +" + " World";
        return helloWorld;
    }
```

crunchify.com

Source: https://crunchify.com/simple-junit-4-tutorial-hello-world-example/

3: How can we assess the readiness of a product?

- A&T during development aim at revealing faults
- We cannot 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

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

4: How can we control the quality of successive releases?

- 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

Examples of bug/issue tracking systems

- https://issues.jboss.org/projects/JIRA/issues/JIRA-483?filter=allopenissues
- https://bugzilla.mozilla.org/buglist.cgi?product=Firefox&component=Bookmarks%20%26%20History&resolution=---

5: How can the development process itself be improved?

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

A four step process to improve fault analysis and process

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

An example of process improvement

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.

Exit quiz

1. Software testing activities should start
   A. as soon as the code is written
   B. during the design stage
   C. when the requirements have been formally documented
   D. as soon as possible in the development life cycle
2. Faults found by users are due to:
   A. Poor quality software
   B. Poor software and poor testing
   C. bad luck
   D. insufficient time for testing
3. What is the main reason for testing software before releasing it?
   A. to show that system will work after release
   B. to decide when the software is of sufficient quality to release
   C. to find as many bugs as possible before release
   D. to give information for a risk based decision about release

Source: https://www.proprofs.com/quiz-school/story.php?title=software-testing-practice-test1