California State University, Sacramento
Computer Science Department

CSC 131 : Computer Software Engineering
Fall 2022

Lecture # 9
Software Testing

V & V

Software Verification Process is a process for determining whether the software products of an activity **fulfill the requirements** or **conditions imposed** on them in the previous activities.

Software Validation Process is a process for determining whether the **requirements and the final, as-built system or software product fulfills its specific intended use**.

Verification Techniques

- Reviews
- Formal inspection
- Walkthrough

SACRAMENTO STATE
Redefine the Possible

Validation Techniques

- Unit Testing – Done by developers
- Integration testing - by Testers
- System Testing – By Testers
- User Acceptance Testing- By users

Software Testing and Reliability

- Software Testing is a critical part of the software development process.
- One single fault can breakdown the whole system.
- Testing is very expensive and time consuming.

Software Testing Challenges

- Input space is very large !
- Input interactions
- Input sequencing
- When to stop testing
- Design an efficient test cases

SACRAMENTO STATE
Redefine the Possible

Purpose of Testing

- Defect Detection:
  Problem/challenge: Testing only suggests the presences of faults not their absence.

- Reliability Estimation:
  Problem/challenge : Input distribution used for selecting test cases may be flawed.

Software Testing

What is testing?

- ❑ Testing is a critical & important part of the Software Quality Assurance (SQA).
- ❑ Consists of a set of activities that have to be planned and conducted systematically (testing strategy).
- ❑ Includes low level tests (to verify implementation) and high level tests (to validate against the specification requirements document).

Testing Objective

- Testing is a process of executing a program with the intent of finding defects.

- Testing cannot prove that there are no more errors — it can only show that defects are present!

SACRAMENTO STATE
Redefine the Possible

Testing Principles

- All tests should be traceable to customer requirements.
- Tests should be planned long before testing begins.
- Testing should begin in the small and progress to larger components.
- Testing is more much more effective when conducted early in SDLC.

Software Testing Process -Four Phases-

- Modeling the software and the software environment.
- Generating and designing test cases.
- Automating and executing test cases.
- Measuring testing progress.

Modeling The Software Environment

- Testers must identify and simulate interfaces that a software system uses.

- Enumerate the inputs that can cross each interface.

Generating and Designing Test Cases

- Test cases are infinite (VERY LARGE SET)
- Only a subset is selected

SACRAMENTO STATE
Redefine the Possible

Automating and Executing Test Cases

- ❑ Automate test cases/scenarios
- ❑ Test case evaluation

Measuring Testing Progress

- Determining when to stop testing is complex.
- Pass/Fail data is collected and analyzed to quantitatively answer the question of "when to stop testing"
- # of defects found
- Types of defects found
- Some questions to answer by Testers:
- Have I tested for common errors?
- Have I exercised all of the source code?
- Have I forced all internal data to be initialized and used?
- Have I found all seeded errors?
- Have I tested the critical components?

SACRAMENTO STATE
Redefine the Possible

[Continued…]

Measuring Testing Progress

Three issues:

- **Quality estimation:** Entails measuring the reliability or mean time to failure of the application under test.
- **Process assessment:** Is the measurement of how well the development and debugging process is progressing toward a reliable product.
- **Stopping criteria:** Are there measures that give testers an insight into the completeness of testing

Metrics to Measure When To Stop Testing

- ❑ Number of defects
- ❑ Types & severity of defects
- ❑ Where defects found (components)
- ❑ Common defects (language dependent and application dependent)
- ❑ Code coverage
- ❑ Seeding defects

SACRAMENTO STATE
Redefine the Possible

Black Box Testing

- No knowledge of how code and how it written
  - focuses on input/output of each component or call
- Based on requirements and functionality, not code
- Emphasis on parameters, inputs/outputs

White-box

- Written with knowledge of the implementation of the code under test.
  - focuses on internal states of objects and code
  - focuses on trying to cover all code paths/statements
  - requires internal knowledge of the component

Questions?

SACRAMENTO STATE
Redefine the Possible