CSE 464 Software Quality Assurance and Testing

Software Testing

Xusheng Xiao

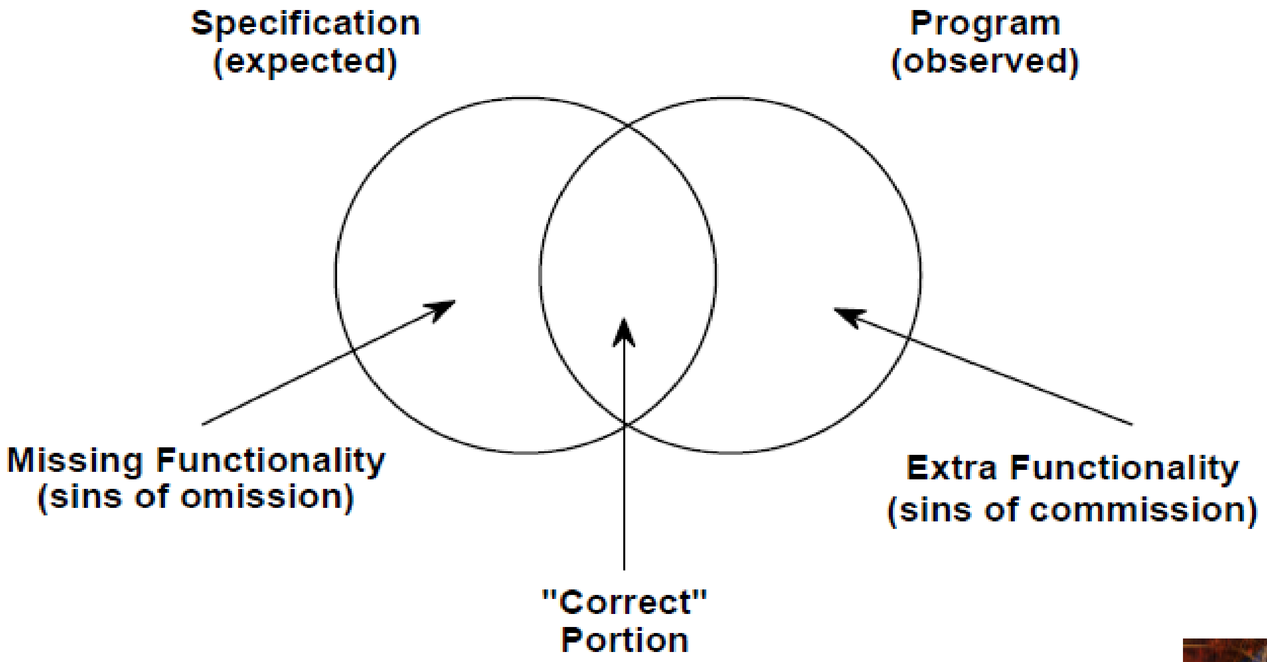
Associate Professor
School of Computing and Augmented Intelligence
Arizona State University

From the Last Lecture

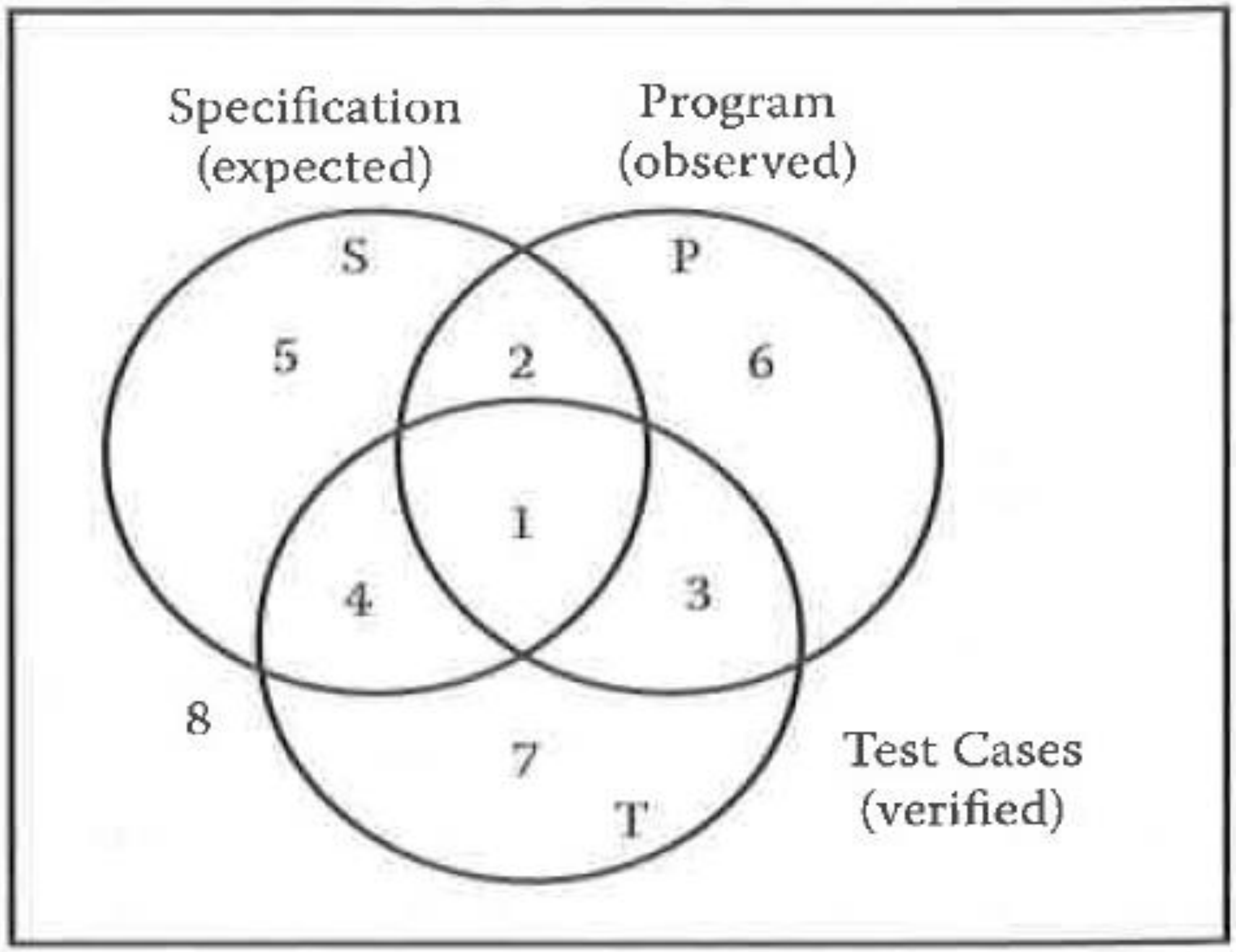
- **What is software quality?**
  - Different stakeholders have different view points about the quality
  - Various software quality frameworks such as ISO 9126, CMM and McCall's triangle of quality have being proposed
- **Quality Assurance**
  - In general, quality assurance can be enforced at three different levels (defect prevention, defect removal, and defect containment)
- **Software testing (What do we test?)**
  - Software testing is the widely practiced quality assurance activity. It comes under defect reduction.

What do we test? : Insights from a Venn diagram

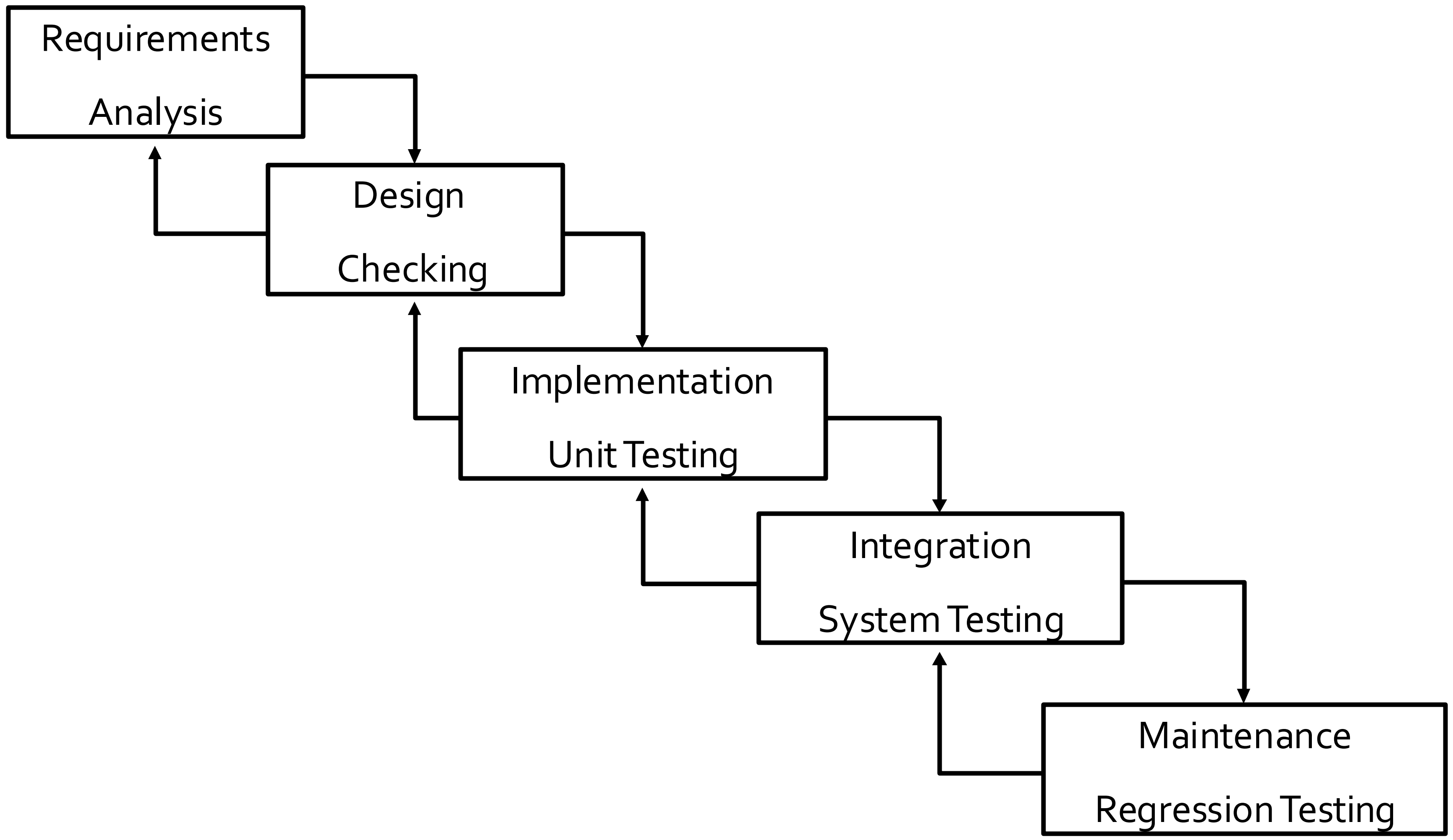
- SRS document gives the specification (expected functionality) of the software
- Actual program has the functionality implemented


Specification (expected)
Program (observed)
Missing Functionality (sins of omission)
Extra Functionality (sins of commission)
"Correct" Portion

Observing Program Behavior Including Tested Behaviors



V&V in Modified Waterfall Model



Why Conduct Testing?

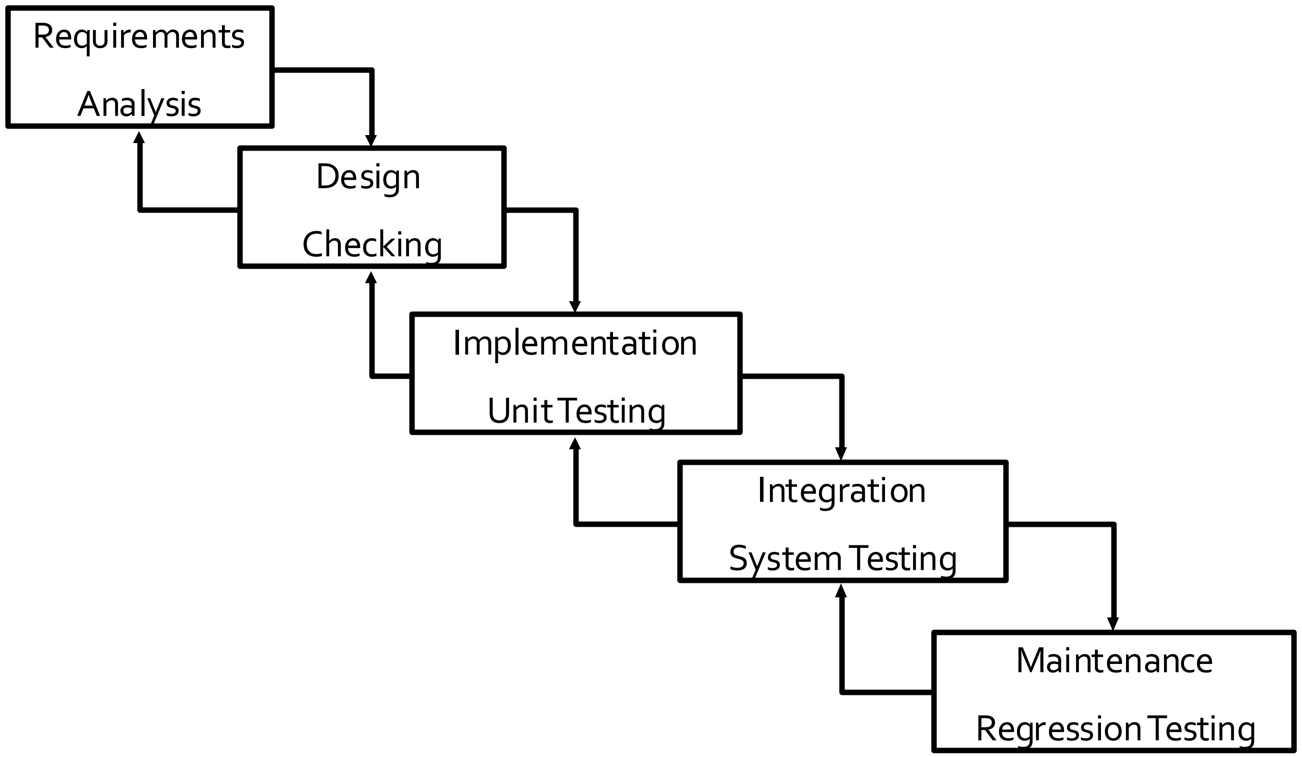
- Improve quality - find faults
- Measure quality
  - Prove there are no faults? (Is it possible?)
  - Determine whether software is ready to be released
  - Determine what to work on
- Learn the software

"Testing shows the presence, not the absence of bugs."
1972 ACM Turing Awardee: Edsger W. Dijkstra

Who Write Tests?

- Developer?
- Separate “quality assurance” group?
- User?
- Someone with a degree in “testing”?
- Someone with domain knowledge of the software?

When to Write Tests?



When to Run Tests?

- **After** you change the software under test
- **Before** you deliver the software
  - To customer
  - To next phase of project

What is a Test Case?

- Aka: test (for short)
- Run software under test with known inputs (**test inputs/data**), check results (**w/ test oracles**)
  - Tests pass or fail
- Tests can document faults
- Tests can document code

(IEEE, 2017)

Manual Test: Example

Test ID	Description	Expected Results	Actual Results
addInventorySuccessfully	Precondition: Run CoffeeMaker Click Menu option 4, "Add inventory"; Enter Coffee: 5 Milk: 3 Sugar: 7 Chocolate: 2; Return to the main menu page.	Newly added inventory displayed on the main menu page	

(Meneely et al., 2008)

Automated Test: Example

- **Doubling** the balance and then plus 10

```
public int calAmount () {
     int ret = balance * 3;
     ret = ret + 10;
     return ret;
}
```

```
void testCalAmount() {
   Account a = new Account();
   Account.setBalance(0);
   int amount = Account.calAmount();
   assertTrue(amount == 10);
}
```

Where is test input?
Where is test oracle?

Test Automation

- Tests are code/scripts
  - Created by a tool that records user actions
  - Written manually in a test scripting language

- Real projects can often have more test code than production code

- Test code is "boring"
  - Build some (complex) data values
  - Run a method or multiple methods
  - Check the result

When to Use Automated Tests vs. Manual Tests

- Automated Tests
  - When you run them over and over (Manual regression testing is BORING)
    - As you build the software incrementally
    - As you change it
  - When you want to use them as documentation (example runs)

- Manual Tests (good for exploratory, GUI testing)
  - When you are only going to run the test once
    - Should you automate the second time you run?
  - When tester doesn't know how to program
    - … or maintainer doesn't know how to program
  - When tests are too expensive to automate
    - … and maintain automated tests

What Kind of Tests?

- Programmer tests / Non-programmer tests
- Developer / Tester
- Unit tests / Integration tests / System tests
- Automated tests / Manual tests
- Regression tests / Exploratory tests

More Kinds of Tests

- Fault-based tests
  - Look for common faults
- Scenario-based tests
  - Write tests based on user stories
- Random tests (aka fuzz tests)
  - See if running random input will crash the software
  - See if output of running random input agrees with the oracle

Even More Kinds of Tests

- Performance tests
  - Measure time of each test

- Load tests
  - Test system under heavy load

- Usability tests
  - See how well people can use the software

- Security tests ...

Testing Activities

- Test Design
- Test Automation
- Test Execution
- Test Evaluation

(Ammann & Offutt, 2016)

Test ID	Description	Expected Results	Actual Results
addInventorySuccessfully	Precondition: Run CoffeeMaker Click Menu option 4, "Add inventory"; Enter Coffee: 5 Milk: 3 Sugar: 7 Chocolate: 2; Return to the main menu page.	Newly added inventory displayed on the main menu page	

(Meneely et al., 2008)

Testing Activities

- Test Design
  - Design test inputs to satisfy a test objective
- Test Automation
  - Embed test inputs into executable scripts
- Test Execution
  - Run test inputs on the software and record actual results
- Test Evaluation
  - Evaluate actual results / document expected results

Assigning Right Persons

(Ammann & Offutt, 2016)

- Test Design
  - technical knowledge of testing/programming

- Test Automation
  - knowledge of scripting

- Test Execution
  - very little knowledge

- Test Evaluation
  - domain knowledge

Mistake, Fault, Error, Failure

What does Bug mean in "Bug Report"?

Developer makes a mistake.

Fault (defect, bug) introduced in the software.

Fault remains undetected during testing

i.e., run test inputs & check outputs

The program fails (based on test oracles) during execution

i.e., it behaves unexpectedly

Error: difference between (1) computed, observed, or measured value/condition &
(2) true, specified, or theoretically correct value/condition

Fault & Failure Model: PIE

(Voas, 1992)

Three conditions necessary/sufficient for a fault $\longrightarrow$ a failure

1. Execution
   - The **fault-containing location(s)** in the software is executed
2. Infection
   - State of the software is infected, i.e., causing **error**
3. Propagation
   - Infected state is propagated to cause some state/output of the software to be observed (e.g., with **test oracle**)

Revisit Fault, Error, or Failure

- Should not allow withdrawal when there is a balance of 100 or less

```
public boolean doWithdraw(int amount) {
     if (Balance < 100)
       return false;
     else
       return withDraw(amount);
}
```

```
void testWithDraw() {
   Account a = new Account();
   Account.setBalance(100);
   boolean success = Account.doWithdraw(10);
   assertTrue(!success);
}
```

Conversion 1: Tester ⇔ Test Manager

- Test Manager: Looks like the code under test is not achieving 100% statement coverage. Please work hard to achieve 100% or close to 100% statement coverage as much as possible.
- Tester: Hmm… boss, our goal is to detect faults. I don't think I need to spend more efforts to achieve 100% statement coverage.
- Test Manager: Well, according to the PIE model, you need to. The reason is … **[You fill in here]**

Conversion 2: Tester ⇔ Test Manager

- Tester: Boss, following your instruction, I work very hard and I have already achieved 100% statement coverage! I would like to take a vacation in Hawaii. Could you approve?
- Test Manager: Well, according to the PIE model, no! The reason is …..**[You fill in here]**

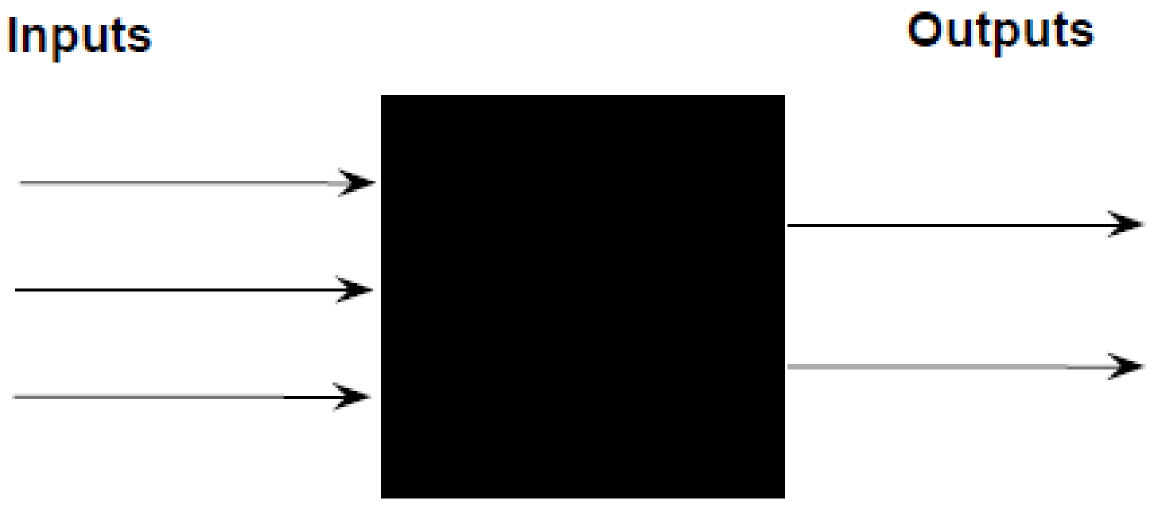
Testing & Debugging

- Testing : The process of finding inputs (and test oracles) that cause the software to fail

- Debugging : The process of finding a fault given a failure

Functional Testing

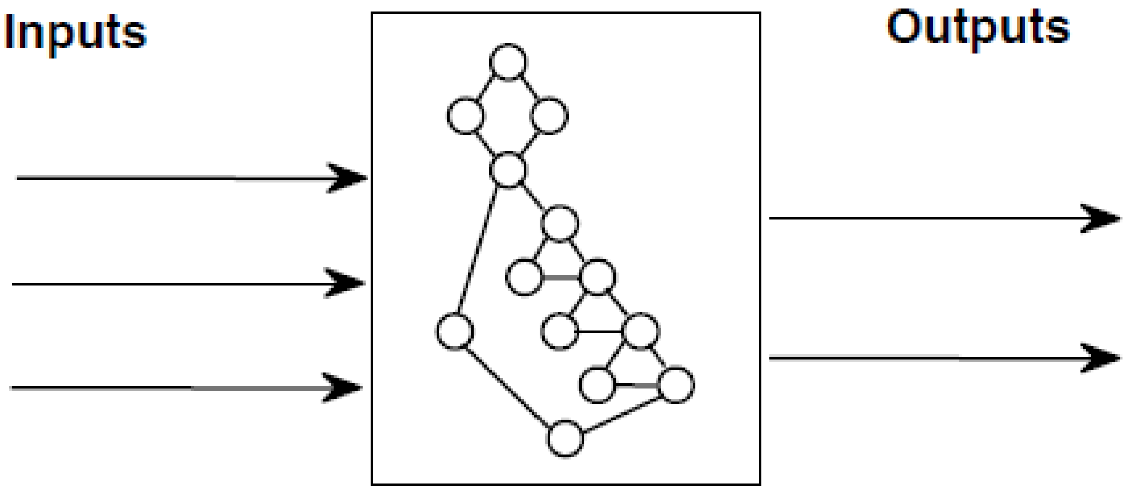
Functional Testing (Black Box)



Function is understood only in terms of it's inputs and outputs, with no knowledge of its implementation.

Structural Testing

Structural Testing (White Box)



Function is understood only in terms of its implementation.

Functional vs. Structural Testing

- They are two fundamentally different approaches in designing test cases.

- **Functional testing uses the specification** while **structural approach uses the program** in designing test cases.

- So, both approaches are needed in determining if we are building the right product and building product right

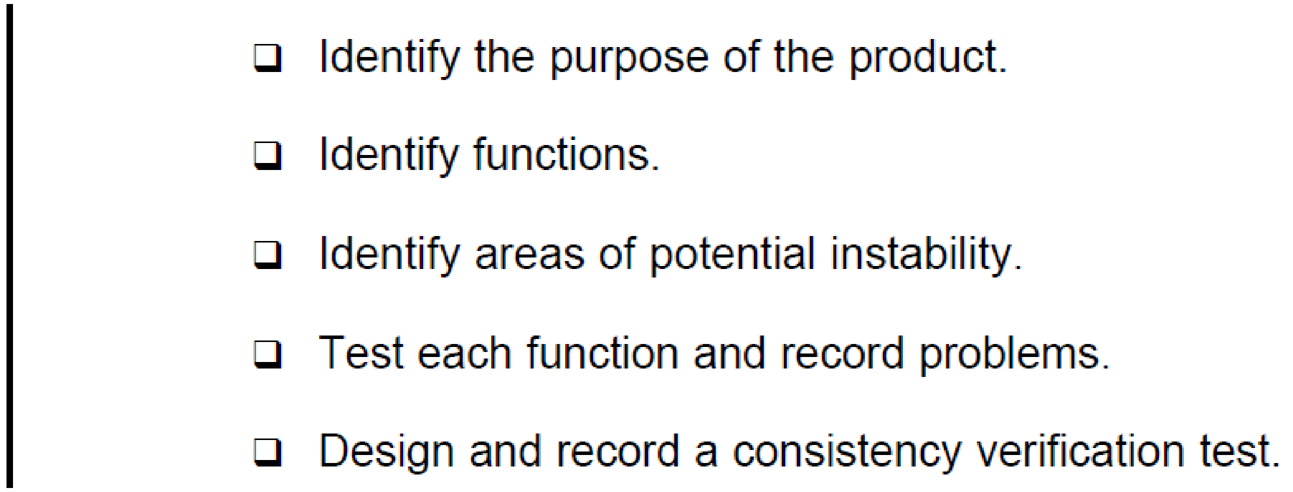
Are Functional and Structural Testing Enough?
Class Discussion

Exploratory Testing

What is exploratory testing?

Is a blackbox testing approach where, a tester first study the software system to understand its purpose (intended usage, users, execution environments, social impact ...etc) and test functionalities of the software by grouping them based in importance, similarity ...etc.

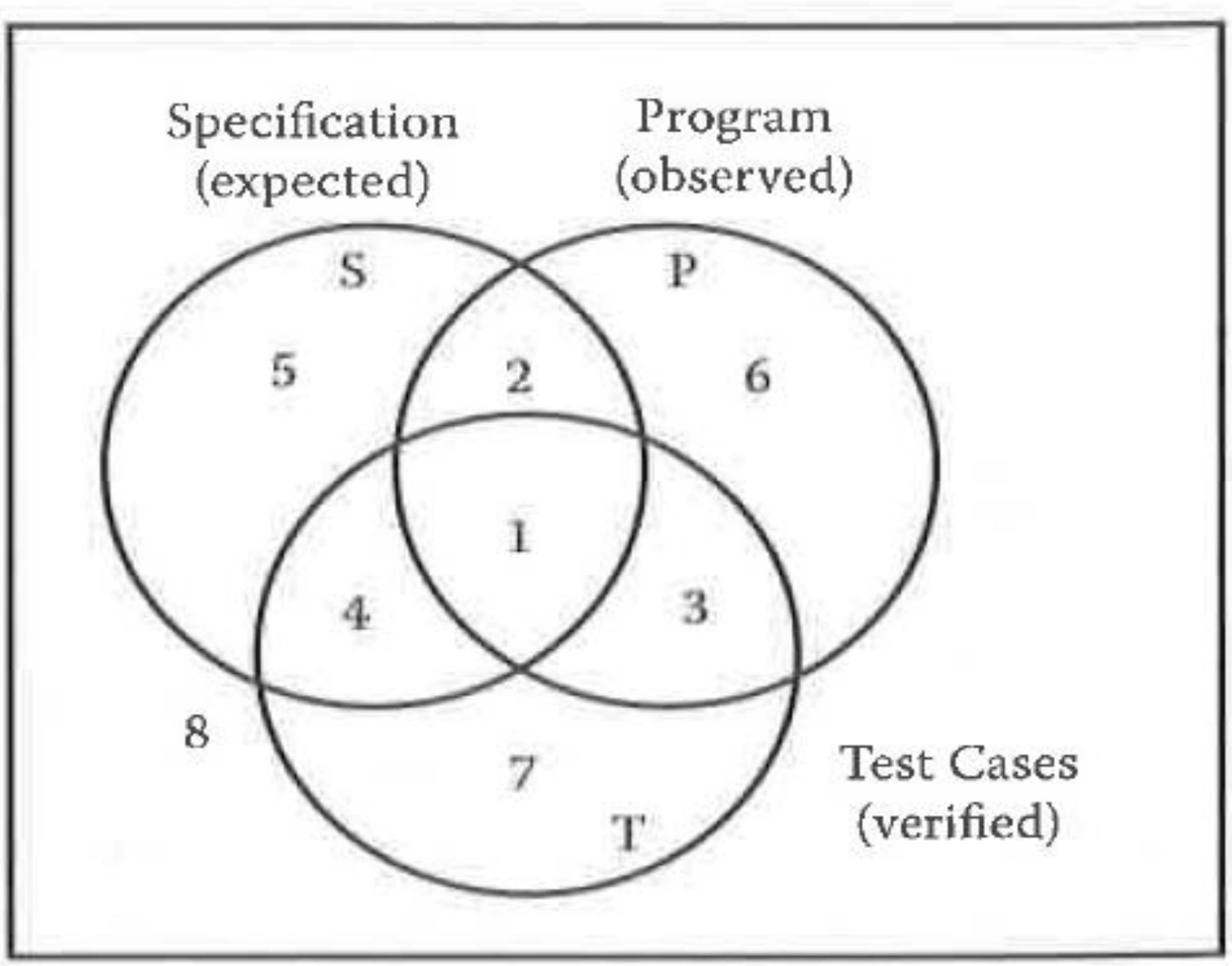
Exploratory Testing Steps



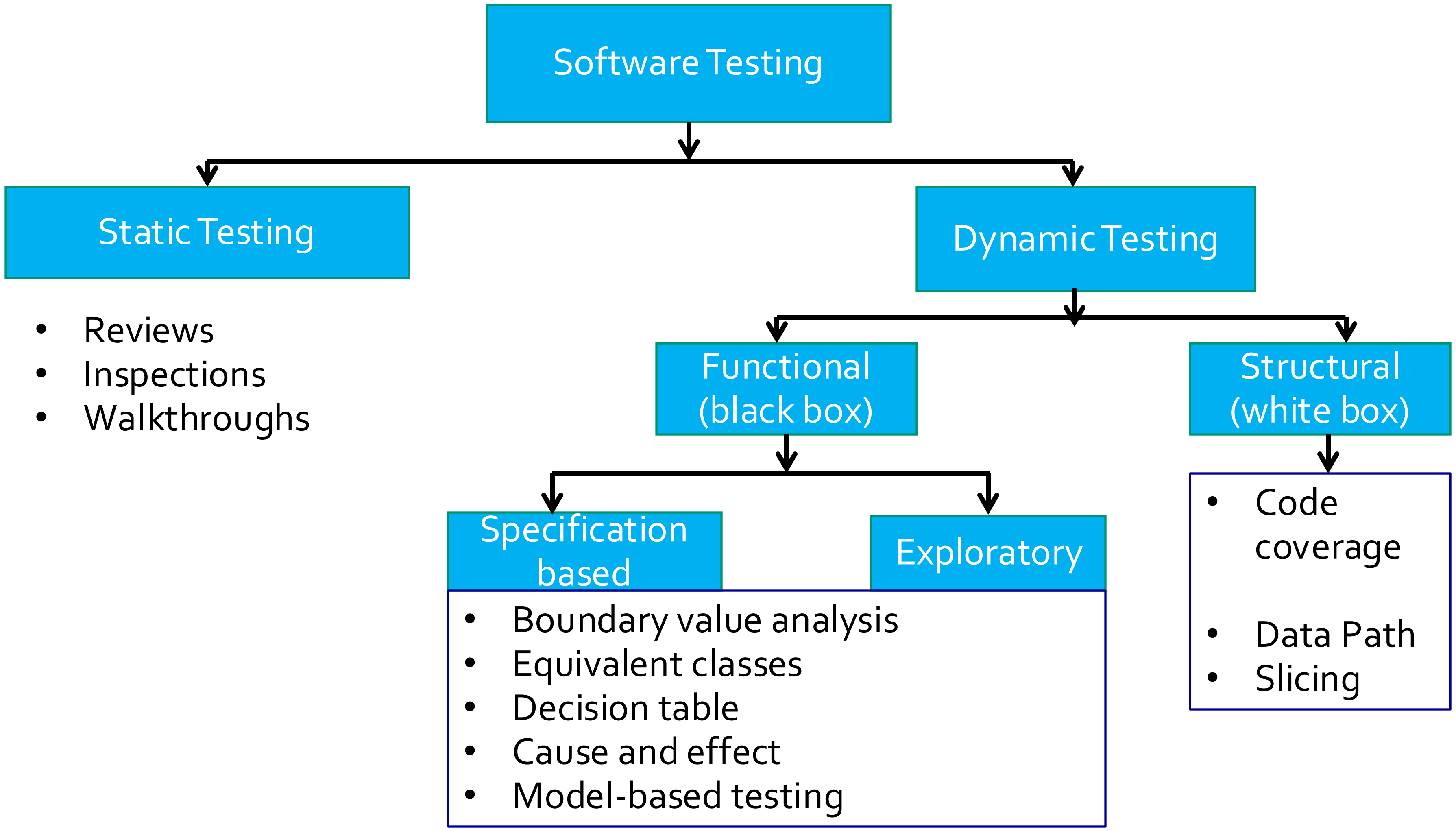
- ❑ Identify the purpose of the product.
- ❑ Identify functions.
- ❑ Identify areas of potential instability.
- ❑ Test each function and record problems.
- ❑ Design and record a consistency verification test.

http://www.satisfice.com/tools/procedure.pdf

Observing Program Behavior



Various Approaches to Software Testing

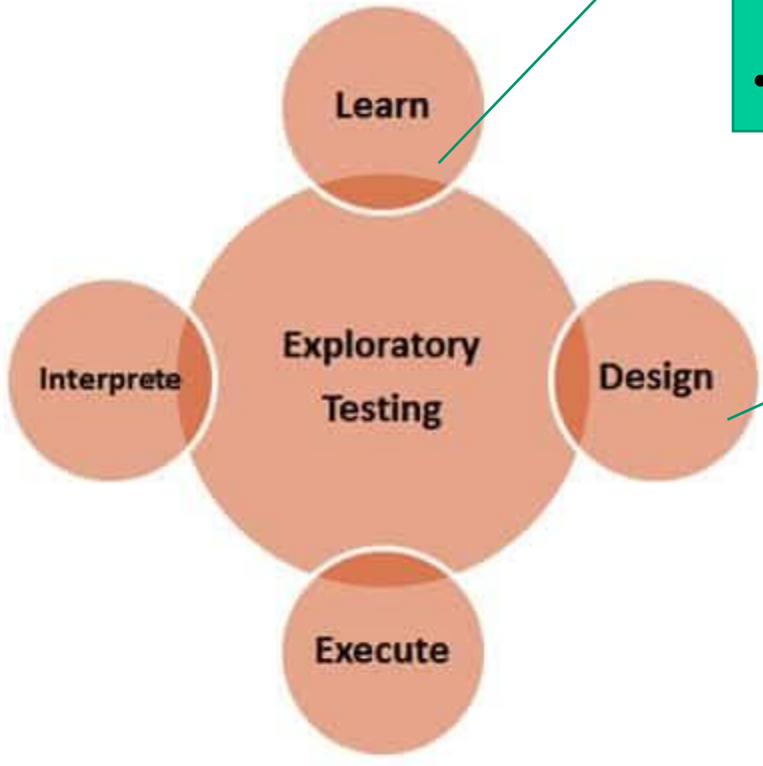


- Reviews
- Inspections
- Walkthroughs

- Boundary value analysis
- Equivalent classes
- Decision table
- Cause and effect
- Model-based testing

- Code coverage
- Data Path
- Slicing

Exploratory Testing - Approach



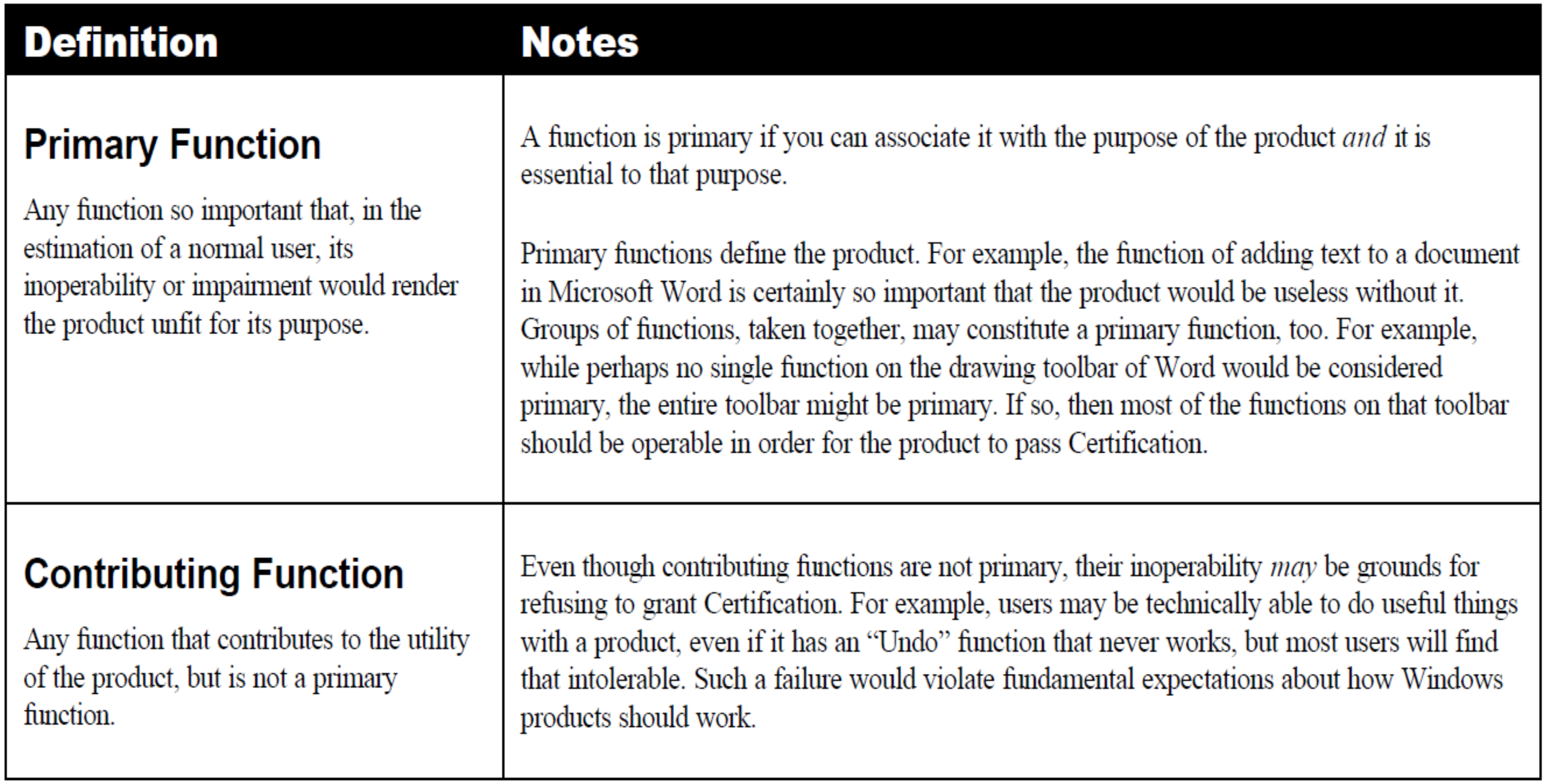
Exploratory Testing Process

- Understanding stakeholders including end users and their expectations
- Learn the system

- Think about what the system supposed to do. Also, check if its doing what it should not do

https://www.softwaretestinghelp.com/what-is-exploratory-testing/

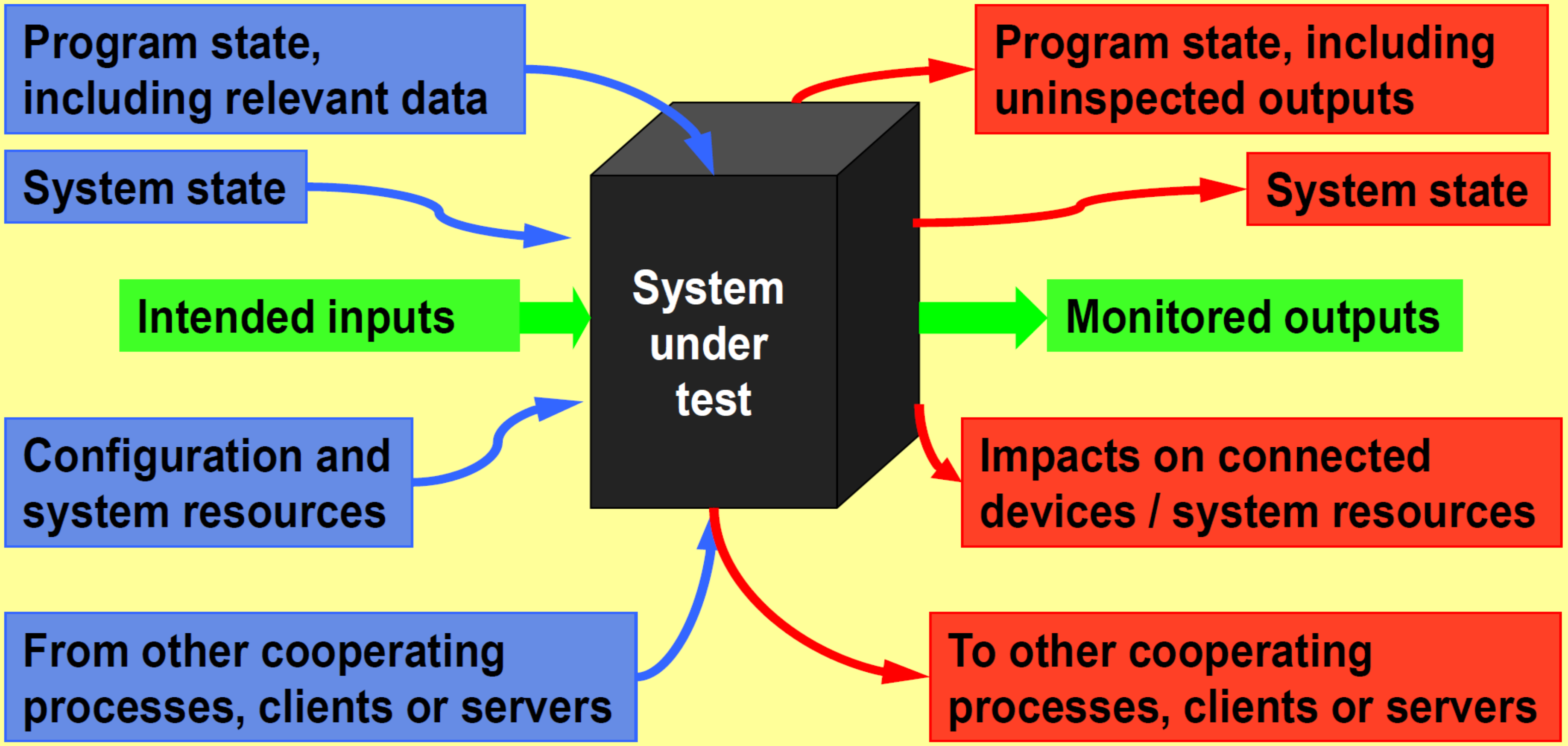
Exploratory Testing : Function Categorization



Definition	Notes
Primary Function Any function so important that, in the estimation of a normal user, its inoperability or impairment would render the product unfit for its purpose.	A function is primary if you can associate it with the purpose of the product *and* it is essential to that purpose. Primary functions define the product. For example, the function of adding text to a document in Microsoft Word is certainly so important that the product would be useless without it. Groups of functions, taken together, may constitute a primary function, too. For example, while perhaps no single function on the drawing toolbar of Word would be considered primary, the entire toolbar might be primary. If so, then most of the functions on that toolbar should be operable in order for the product to pass Certification.
Contributing Function Any function that contributes to the utility of the product, but is not a primary function.	Even though contributing functions are not primary, their inoperability *may* be grounds for refusing to grant Certification. For example, users may be technically able to do useful things with a product, even if it has an "Undo" function that never works, but most users will find that intolerable. Such a failure would violate fundamental expectations about how Windows products should work.

http://www.satisfice.com/tools/procedure.pdf

A MODEL OF A SYSTEM UNDER TEST


Program state, including relevant data
System state
Intended inputs
Configuration and system resources
From other cooperating processes, clients or servers
System under test
Program state, including uninspected outputs
System state
Monitored outputs
Impacts on connected devices / system resources
To other cooperating processes, clients or servers

-- Doug Hoffman

http://www.softwarequalitymethods.com/Slides/Orcl_Tax_Slides.PDF

Tool Support for Exploratory Testing

- Microsoft Azure Test Plans
https://docs.microsoft.com/en-us/azure/devops/test/overview?view=azure-devops#manual-testing-in-test-manager

- qTest explorer
https://www.qasymphony.com/software-testing-tools/qtest-explorer/test-execution-recorder/

- Testpad

https://ontestpad.com/library/151/structure-example-exploratory-testing

- HP Sprinter

https://www.microfocus.com/en-us/products/sprinter-manual-software-testing/overview

- .... more

Writing Test Cases : Anatomy of a Typical Test Case

IEEE Standard 610 (1990) defines *test case* as follows:
"(1) A set of test **inputs**, **execution conditions**, and **expected results** developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement.
"(2) (IEEE Std 829-1983) **Documentation specifying inputs, predicted results, and a set of execution conditions for a test item**."

Test case ID

Purpose

Preconditions

Inputs

Expected outputs

Post conditions

Execution History

Date	Result	Version	Run By

http://www.kaner.com/pdfs/GoodTest.pdf

Characteristics of a Good Test Engineer : 10 Engineering Competences

1. Analytical Problem Solving
2. Customer- focused Innovation
3. Technical excellence
4. Project/Time Management
5. Passion for quality
6. Strategic insight
7. Confidence
8. Impact and Influence
9. Cross-Boundary Collaboration
10. Interpersonal Awareness

Reference : How We Test Software at Microsoft by Alan Page , Ken Johnston , Bj Rollison

Functional Testing (Specified Behavioral Testing) Approaches

- Equivalent classes and Boundary Value Analysis
- Use cases and Decision Table
- State based testing
- Domain Testing
- Exploratory Testing
- ....Many more

Characteristics of a good test case [1]

- **It has a reasonable probability of finding a defect:** Work backward and think about possible ways of failing a particular program
- **Its not redundant :** No two test cases try to catch the same defect
- **It's the best of its bread :** Most suitable data set for a particular test

Thank You !

Questions ?