Software Quality Assurance

Module 4

Introduction to Software Testing

Objectives

- Introduce foundation topics of software testing
- Explain test ideas
- Introduce test matrices

Module 4 Content Outline

- **➔Software testing**
  - Defining software testing
  - Testing Model
  - Classification of software testing
  - Economics of Testing
  - Vital Program Testing Guidelines
- Defining functional testing
- Test ideas
- Test idea catalogs

Definition of Software Testing

- Software testing is a process, or a series of processes, designed to make sure computer code does what it was designed to do and that it does not do anything unintended.
  - Software should be predictable and consistent, presenting no surprises to users.
- ***Testing is the process of executing a program with the intent of finding errors.***

Definition of Software Testing

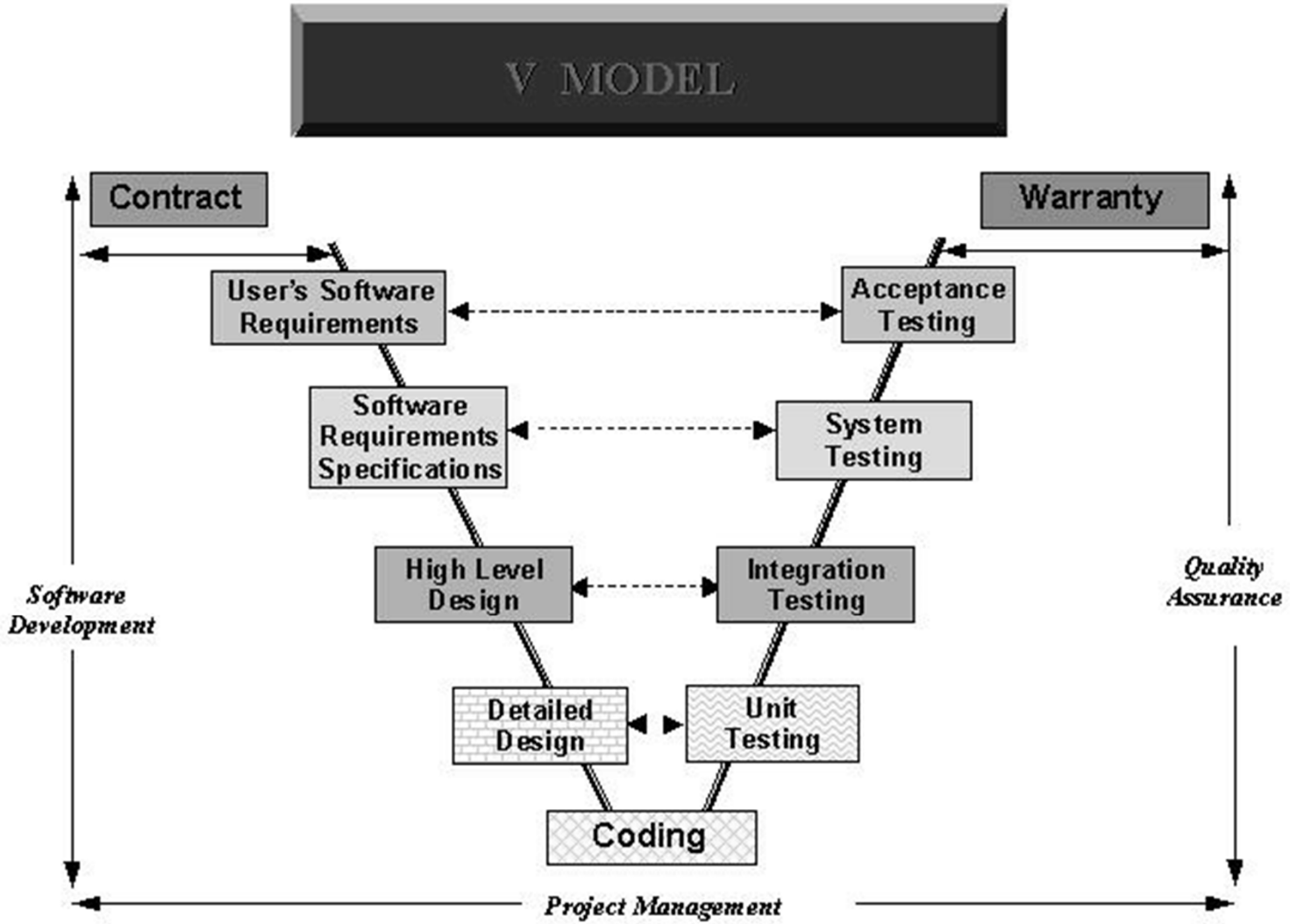
- **Software testing** is a formal process carried out by a specialized testing team in which a software unit, several integrated software units or an entire software package are examined by running the programs on a computer. All the associated tests are performed according to approved test procedures on approved test cases.

Software testing objectives

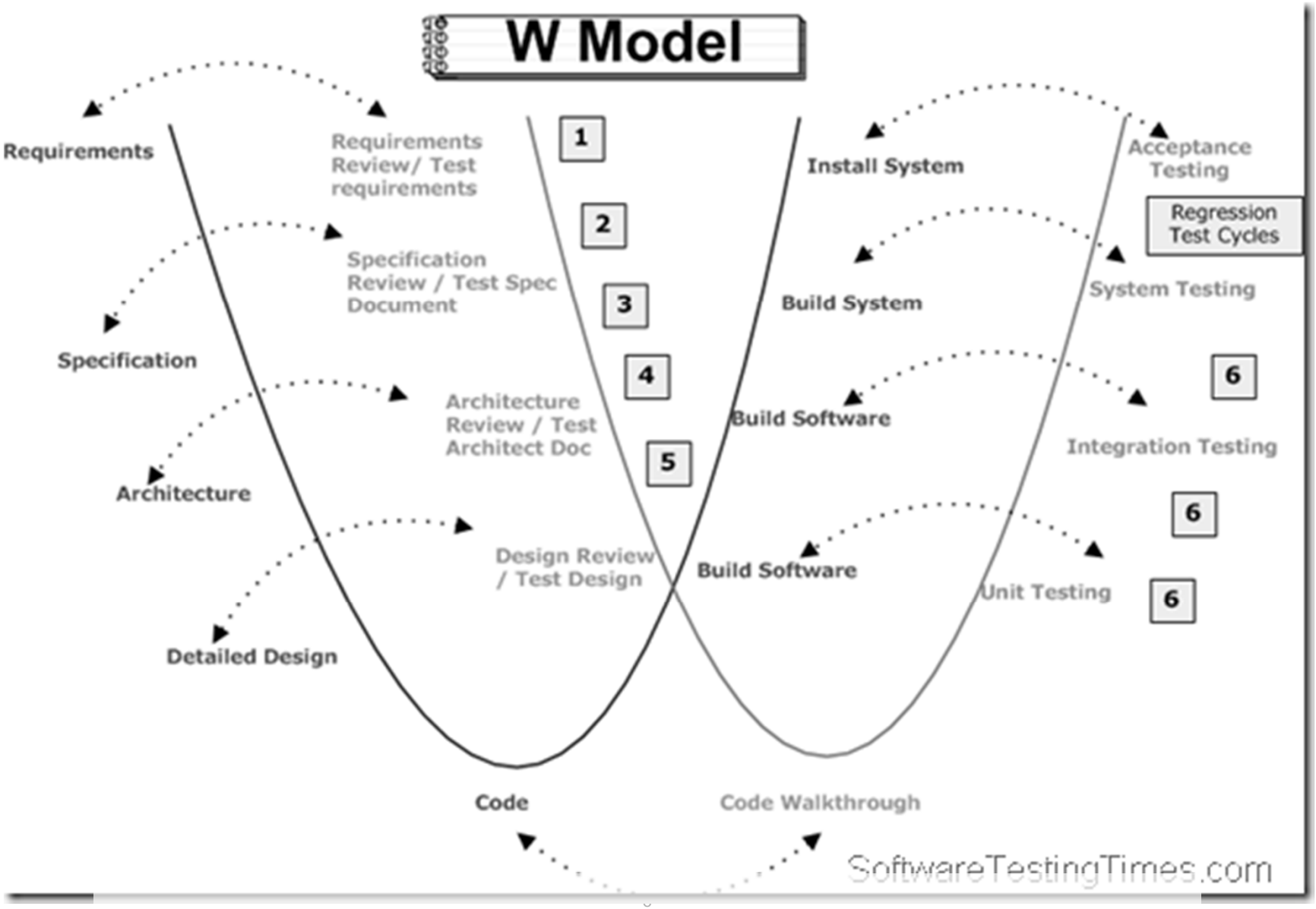
- Direct objectives
  - To identify and reveal as many errors as possible in the tested software.
  - To bring the tested software, after correction of the identified errors and retesting, to an acceptable level of quality.
  - To perform the required tests efficiently and effectively, within budgetary and scheduling limitations.
- Indirect objective
  - To compile a record of software errors for use in error prevention (by corrective and preventive actions).

V Model

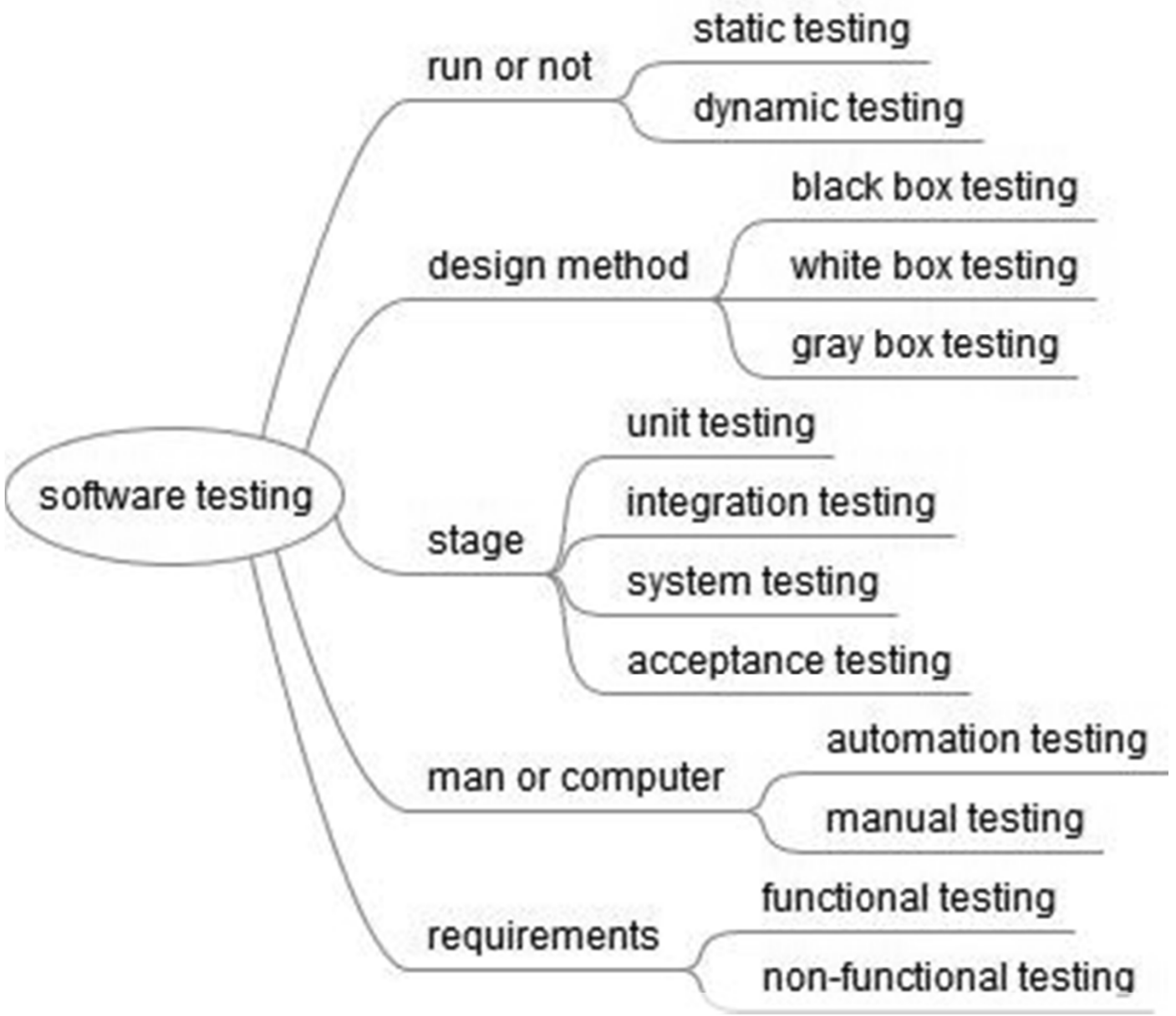
The "V" Diagram indicating this relationship is as follows:



W Model



Classification of software testing



The Economics of Testing

- Given definition of program testing, an appropriate next step is to determine whether it is possible to test a program to find all of its errors.
  - thoroughly testing program

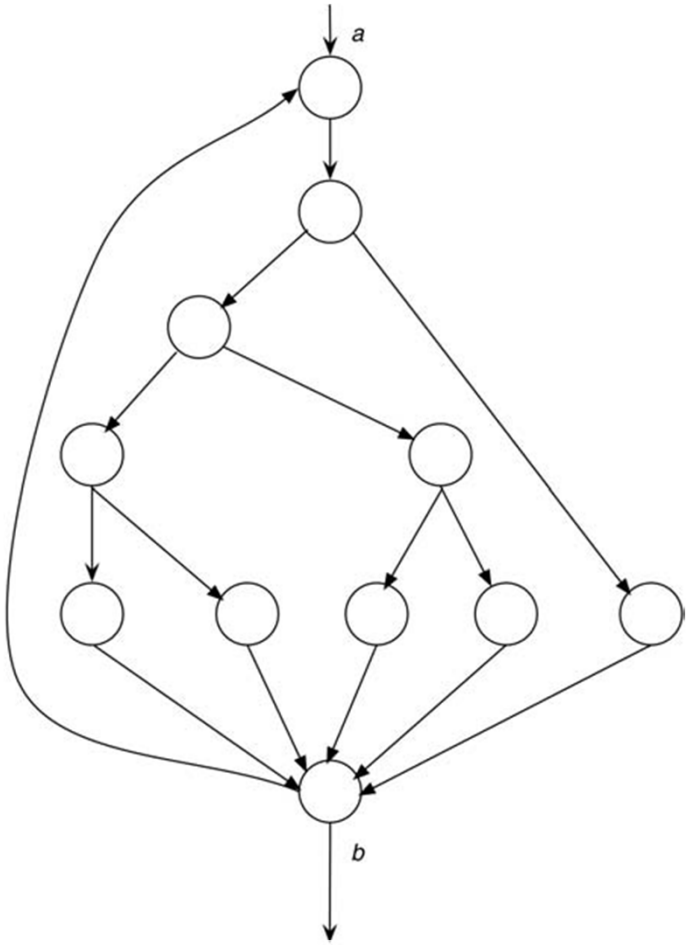
- In general, it is impractical, often impossible, to find all the errors in a program.

The Economics of Testing

- Example 1 --- black box testing
- X = sin(Y)
- X,Y – double
- This discussion shows that exhaustive input testing is impossible. You cannot test a program to guarantee that it is error free.
- Objective should be to maximize the yield on the testing investment by maximizing the number of errors found by a finite number of test cases

The Economics of Testing

- Example 2 --- white box testing
  - DO loop that iterates up to 20 times.



•5^20 ≈ 10^14
where 5 is the number of paths through the loop body
• completing a test once per second, need 3.02 million years

Exhaustive path testing, like exhaustive input testing, appears to be impractical, if not impossible.

The Economics of Testing

- Example 2 --- white box testing

- Even every path could be tested, the program still has errors.
  - An exhaustive path test in no way guarantees that a program matches its specification.
  - A program may be incorrect because of missing path.
  - An exhaustive path test might not uncover data-sensitivity errors.

Test-Case Design

- ***What subset of all possible test cases has the highest probability of detecting the most errors?***

- The recommended procedure is to develop test cases using the black-box methods and then develop supplementary test cases as necessary with white-box methods.

Vital Program Testing Guidelines

Principle Number	*Principle*
1	A necessary part of a test case is a definition of the expected output or result.
2	A programmer should avoid attempting to test his or her own program.
3	A programming organization should not test its own programs.
4	Thoroughly inspect the results of each test.
5	Test cases must be written for input conditions that are invalid and unexpected, as well as for those that are valid and expected.
6	Examining a program to see if it does not do what it is supposed to do is only half the battle; the other half is seeing whether the program does what it is not supposed to do.
7	Avoid throwaway test cases unless the program is truly a throwaway program.
8	Do not plan a testing effort under the tacit assumption that no errors will be found.
9	The probability of the existence of more errors in a section of a program is proportional to the number of errors already found in that section.
10	Testing is an extremely creative and intellectually challenging task.

Module 4 Content Outline

- Software testing
- **➔Defining functional testing**
- Test ideas
- Test idea catalogs

Functional Testing

- In this course, we adopt a common, broad current meaning for functional testing. It is
  - Black box
  - Interested in any externally visible or measurable attributes of the software other than performance.
- In functional testing, we think of the program as a collection of functions
  - We test it in terms of its inputs and outputs.

Discussion 4.1: Adv-s & Disadv-s of Functional testing

- What are the advantages of functional testing?
- What are the disadvantages of functional testing?

Module 4 Content Outline

- Software testing
- Defining functional testing

➔**Test ideas**

- Test idea catalogs

Test Ideas

- A test idea is a brief statement that identifies a test that might be useful.
- A test idea differs from a test case, in that the test idea contains no specification of the test workings, only the essence of the idea behind the test.
- Test ideas are generators for test cases: potential test cases are derived from a test ideas list.
- A key question for the tester or test analyst is which ones are the ones worth trying.

Exercise 4.2: Brainstorm Test Ideas (1/2)

- Ground Rules for Brainstorming
  - The goal is to get lots of ideas. You brainstorm together to discover categories of possible tests—good ideas that you can refine later.
  - There are more great ideas out there than you think.
  - Don’t criticize others’ contributions.
  - Jokes are OK, and are often valuable.
  - Work later, alone or in a much smaller group, to eliminate redundancy, cut bad ideas, and refine and optimize the specific tests.
  - Often, these meetings have a facilitator (who runs the meeting) and a recorder (who writes good stuff onto flipcharts). These two keep their opinions to themselves.

Exercise 4.3: Brainstorm Test Ideas (2/2)

- A field can accept integer values between 20 and 50.
- What tests should you try?

A Test Ideas List for Integer-Input Tests

- Common answers to the exercise would include:

Test	Why it's interesting	Expected result
20	Smallest valid value	Accepts it
19	Smallest -1	Reject, error msg
0	0 is always interesting	Reject, error msg
Blank	Empty field, what's it do?	Reject? Ignore?
49	Valid value	Accepts it
50	Largest valid value	Accepts it
51	Largest +1	Reject, error msg
-1	Negative number	Reject, error msg
4294967296	2^32, overflow integer?	Reject, error msg

Discussion 4.4: Where Do Test Ideas Come From?

- Where would you derive Test Ideas Lists?
  - Models
  - Specifications
  - Customer complaints
  - Brainstorm sessions among colleagues
  - Bug lists
  - Representative exemplars

Module 4 Content Outline

- Software testing
- Defining functional testing
- Test ideas
- ➔**Test idea catalogs**
  - Test matrices

Identify a Generic List of Test Ideas

- Think about the categories of values you'd consider generally applicable in an integer input field with Lower Bound (LB) and Upper Bound (UB).

Test	Why it's interesting	Expected result

"Why it is interesting" often means "what risk are we managing" or "what error we are looking for" or "what category or group of tests this test is an example of."

A Catalog of Test Ideas for Integer-Input tests

- Nothing
- Valid value
- At LB of value
- At UB of value
- At LB of value - 1
- At UB of value + 1
- Outside of LB of value
- Outside of UB of value
- 0
- Negative
- At LB number of digits or chars
- At UB number of digits or chars
- Empty field (clear the default value)
- Outside of UB number of digits or chars
- Non-digits
- Wrong data type (e.g. decimal into integer)
- Expressions
- Space
- Non-printing char (e.g., Ctrl+char)
- DOS filename reserved chars (e.g., "\ * . :")
- Upper ASCII (128-254)
- Upper case chars
- Lower case chars
- Modifiers (e.g., Ctrl, Alt, Shift-Ctrl, etc.)
- Function key (F2, F3, F4, etc.)

The Test-Ideas Catalog

- A test-ideas catalog is a list of related test ideas that are usable under many circumstances.
  - For example, the test ideas for numeric input fields can be catalogued together and used for any numeric input field.
- In many situations, these catalogs are sufficient test documentation. That is, an experienced tester can often proceed with testing directly from these without creating documented test cases.

Apply a Test Ideas Catalog Using a Test Matrix

	Lower Bound	LB-1	Upper Bound	UB+1	Zero	Spaces	Nothing	Negative
Field name								
Field name								
Field name								

Exercise 4.5: Your Own Test Ideas Lists

- Now (in class)
  - Pick a topic of interest for a test ideas list
- Homework
  - Expand into test ideas list
  - We will discuss this at next class
- Homework follow-up in class
  - Develop a matrix from these ideas

Review: Core Concepts of Software Testing

- What is Software Testing?
- What are the models of software testing?
- What is Functional testing?
- What are Test Ideas?
- Where are Test Ideas useful?
- Give some examples of a Test Ideas.
- Explain how a catalog of Test Ideas could be applied to a Test Matrix.