Topic #1 – Software Black-Box Testing

Instructor: Jerry Gao, Ph.D., Professor
San Jose State University

Topic #1 – Software Black-Box Testing

- What Is Black-Box Testing?
- Black-Box Testing Focuses
- Why Is Black-Box Testing Important?
- Who Does Black-Box Testing?
- A Black-Box Testing Example
- Black-Box Testing Coverage

TOPIC #1 – SOFTWARE BLACK-BOX TESTING

What is software black-box testing?

Black-box testing, also known as requirement-based testing

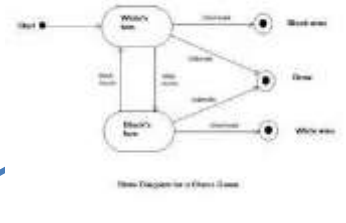
Definition: Software black-box testing refers to test design based on program requirements specifications to validate software functions, external behaviors, and external visible QoS requirements.

What do you need for black-box testing?

- Black-box testing models and test criteria
- Black-box test design and generation methods
- Software requirements specification and product specification

Black-Box Testing Focuses

System QoS Requirements



System Behaviors

Software

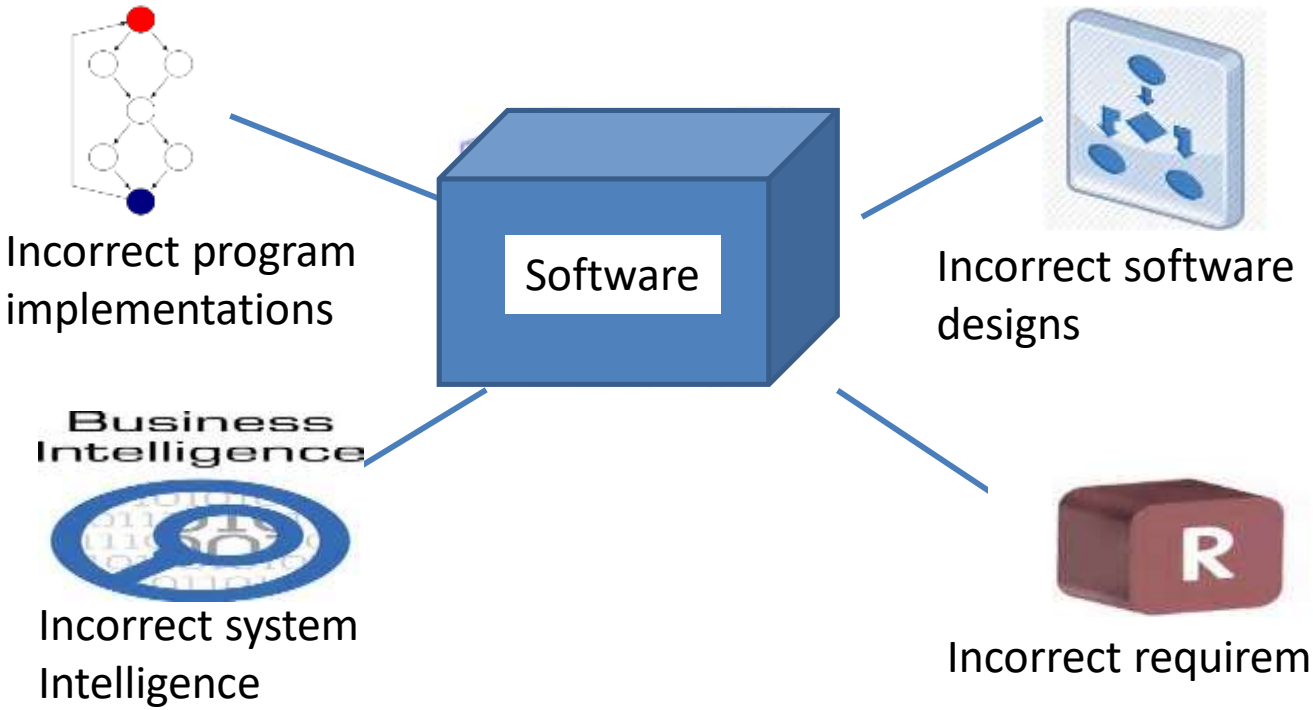
System Functions

Software GUI

- Check system QoS Requirements
- Software GUI
- System Functions
- System Behaviors

Why Is Black-Box Testing Important?

Answer: - Assure the quality of software functions, behaviors and QoS parameters.
- Achieve adequate system requirement validation criteria.



Who Does Black-Box Testing?

Several engineer groups perform black-box program testing:

- Function validation engineers
  – Perform function testing
- System test engineers
  – Conduct system testing for QoS parameters
- Quality assurance engineers
  – Performance system testing based on QA standards
- System users – Acceptance testing

TOPIC #1 – SOFTWARE BLACK-BOX TESTING

A Black-Box Testing Example - Triangle Analyzer

Program specification:

Input: 3 numbers separated by commas or spaces

Processing:

Determine if three numbers make a valid triangle; if not, print message NOT A TRIANGLE.

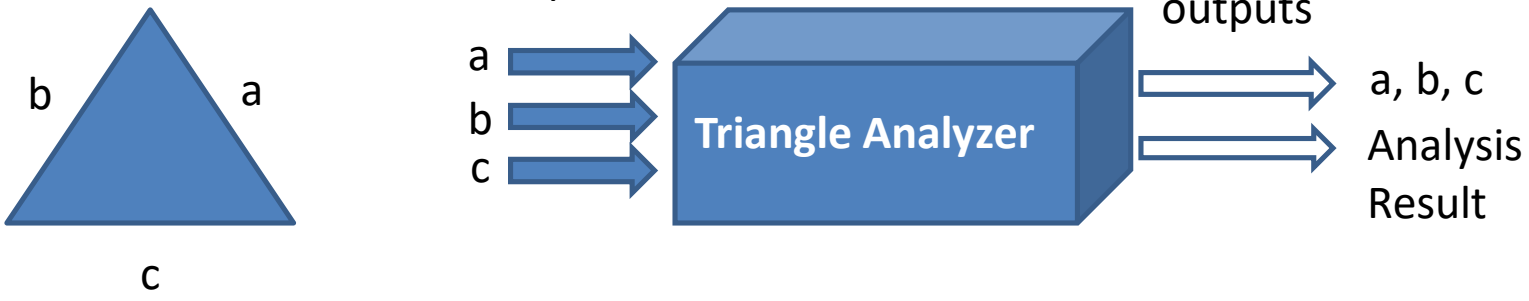
If it is a triangle, classify it according to the length of the sides as scalene (no sides equal), isosceles (two sides equal), or equilateral (all sides equal).

If it is a triangle, classify it according to the largest angle as acute (less than 90 degree), obtuse (greater than 90 degree), or right (exactly 90 degree).

Output: One line listing the three numbers provided as input and the classification or the not a triangle message.

TOPIC #1 – SOFTWARE BLACK-BOX TESTING

A Black-Box Testing Example - Triangle Analyzer

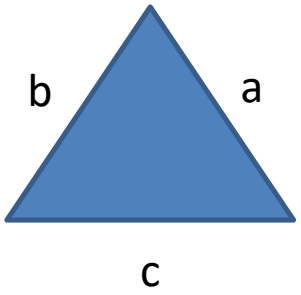


Output: One line listing the three numbers provided as input and the classification or the not a triangle message.

Test Example:	Inputs	Outputs	
	3,4,5	3,4,5	Scalene Right
	6,1,6	6,1,6	Isosceles Acute
	5,1,2	5,1,2	Not a triangle

TOPIC #1 – SOFTWARE BLACK-BOX TESTING

A Black-Box Testing Example - Triangle Analyzer

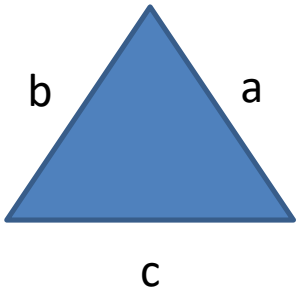


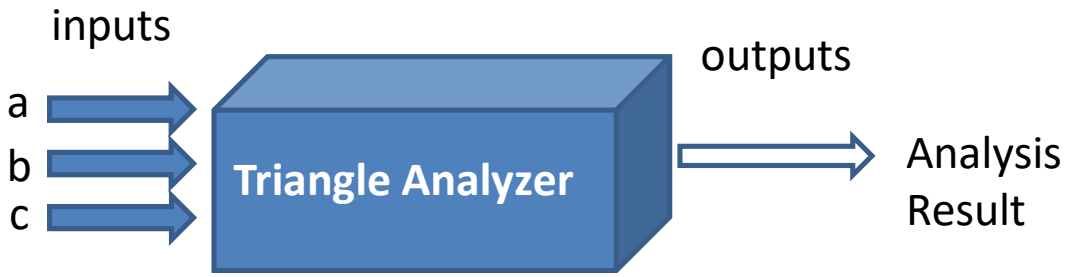
Test Set #1

Inputs	Expected Results	
4,4,4	4,4,4	Equilateral acute
6,5,3	6,5,3	Scalene acute
5,6,10	5,6,10	Scalene obtuse
3,4,5	3,4,5	Scalene right
6,1,6	6,1,6	Isosceles acute
7,4,4	7,4,4	Isosceles obtuse
1,2,2	1,2,2	Isosceles right

TOPIC #1 – SOFTWARE BLACK-BOX TESTING

A Black-Box Testing Example - Triangle Analyzer





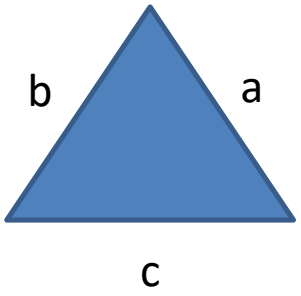
Test Set #2

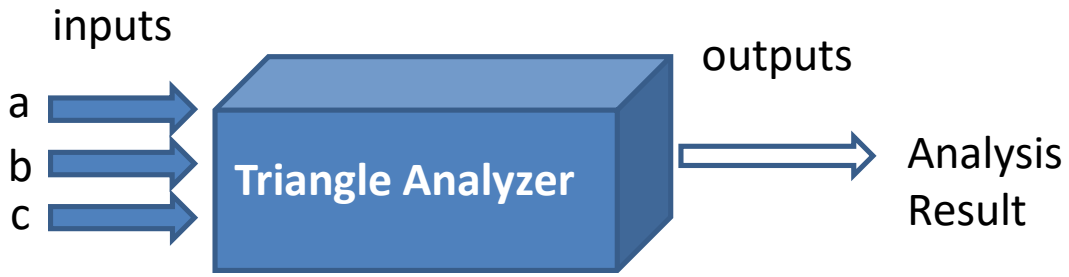
Test cases for special inputs and invalid formats:

Inputs	Descriptions
3,4,5,6	Four sides
646	Three-digit single number
3,,4,5	Two commas
3 4,5	Missing comma
3.14.6,4,5	Two decimal points
4,6	Two sides
5,5,A	Character as a side
6,-4,6	Negative number as a side
-3,-3,-3	All negative numbers
	Empty input

TOPIC #1 – SOFTWARE BLACK-BOX TESTING

A Black-Box Testing Example - Triangle Analyzer





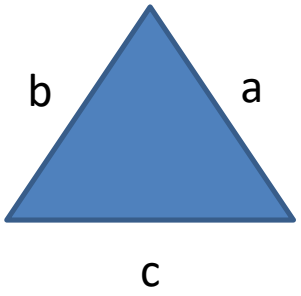
Test Set #3

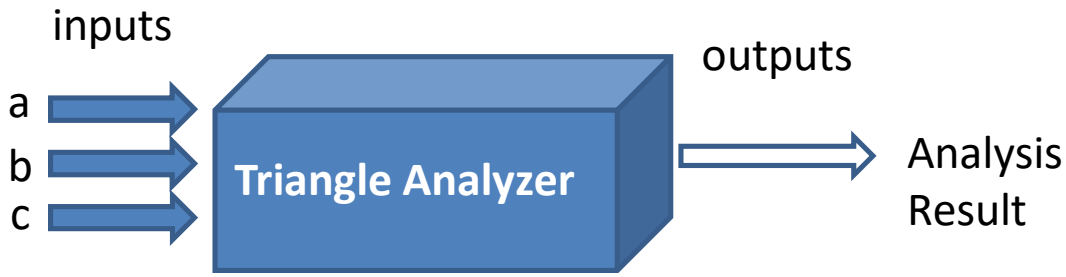
Boundary conditions for legitimate triangles:

Inputs	Descriptions
1,1,2	Makes a straight line, not a triangle
0,0,0	Makes a point, not a triangle
4,0,3	A zero side, not a triangle
1,2,3.00001	Close to a triangle but still not a triangle
9170,9168,3	Very small angle (scalene, acute)
.0001,.0001,.0001	Very small triangle (equilateral, acute)
83127168,74326166,96652988	Very large triangle, scalene, obtuse

TOPIC #1 – SOFTWARE BLACK-BOX TESTING

A Black-Box Testing Example - Triangle Analyzer





Test Set #3

Boundary conditions for sides classification:

Inputs	Descriptions
3.0000001,3,3	Very close to equilateral (isosceles, acute)
2.999999,4,5	Very close to isosceles (scalene, acute)

Boundary conditions for angles classification:

Inputs	Descriptions
3,4,5.000000001	Near right triangle (scalene, obtuse)
1,1,1.41141414141414	Near right triangle (isosceles, acute)

Software Testing Principles

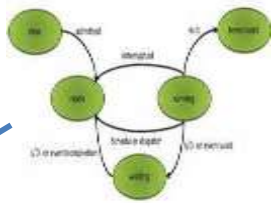
A set of testing principles listed below:

- All tests should be traceable to customer requirements.
- Tests should be planned long before testing begins.
- Testing should begin “in the small” and progress toward testing “in the large”.
- Exhaustive testing is not possible.
- Testing should be conducted by an independent third party.

Black-Box Software Testing Coverage

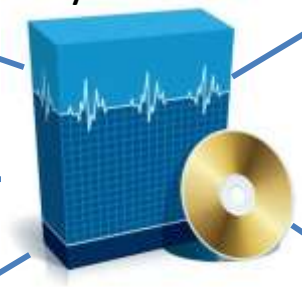
System Function Requirement Coverage

Under-test System



System Behavior Coverage

System QoS Parameter Coverage



Business Roles & Intelligence Coverage

System GUI Coverage

Topic #2 –Equivalence Partitioning Test Method

Instructor: Jerry Gao, Ph.D., Professor
San Jose State University

Topic #2 – Equivalence Partitioning Test Method

What is Equivalence Partitioning(EP) Testing?

Why Do We Need the EP Test Method?

How to Use the EP Test Method?

Equivalence Partitioning Test Examples

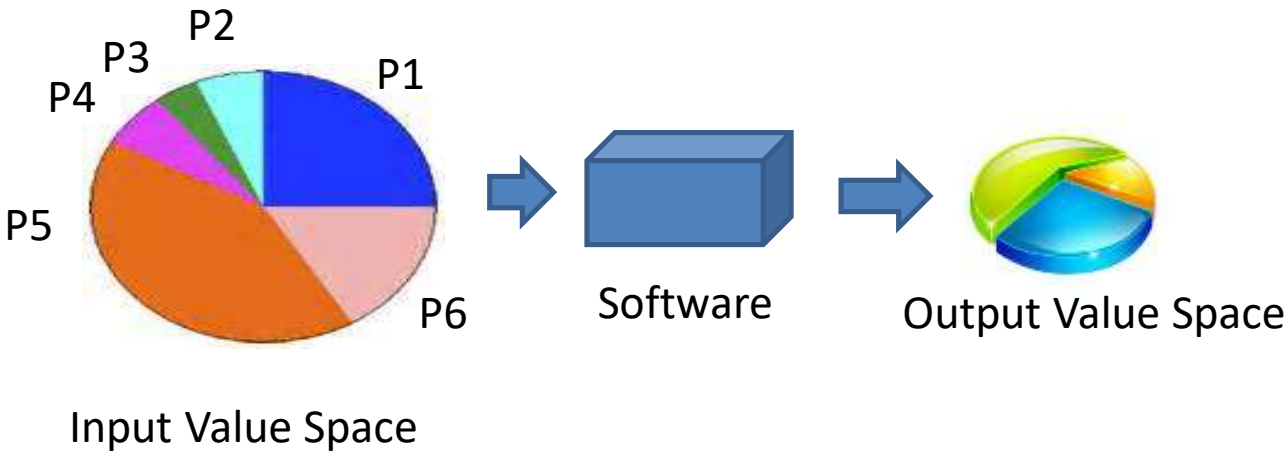
Equivalence Partitioning Summary

TOPIC #2 – EQUIVALENCE PARTITIONING TEST METHOD

What is software equivalence partitioning (EP) method?

Definition: Equivalence Partitioning (EP) is a specification-based test technique. The idea behind this is to divide the input value space for a under-test program into a set of partitions (or classes). Each partition only needs one test input vector to cover it even though there are many others.

Equivalence partitions are also known as equivalence classes.



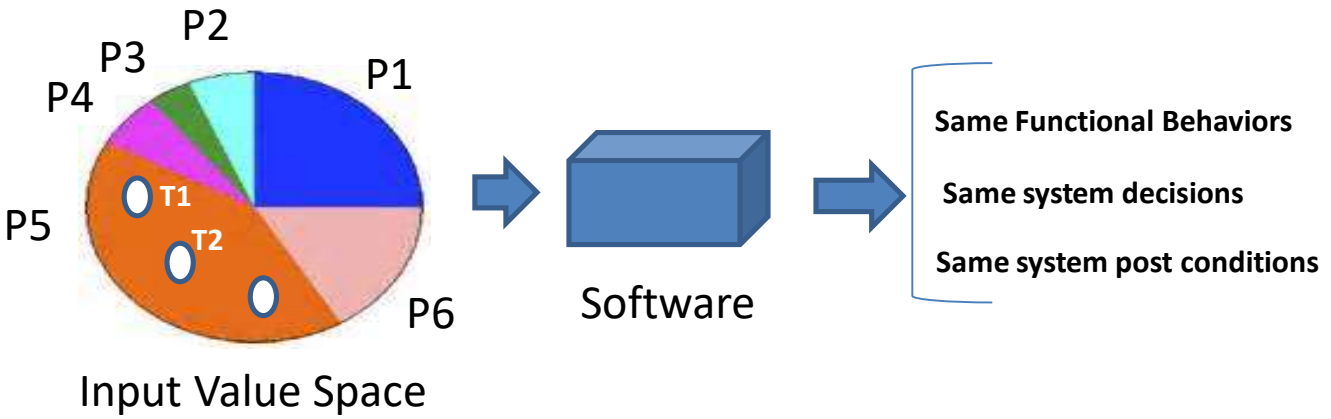
TOPIC #2 – EQUIVALENCE PARTITIONING TEST METHOD

What is software equivalence partitioning (EP) method?

Assumptions of the EP method:

- An under-test software will demonstrate the same functional behavior for all tests of an EQ partition. Therefore, only one test is needed for each partition.

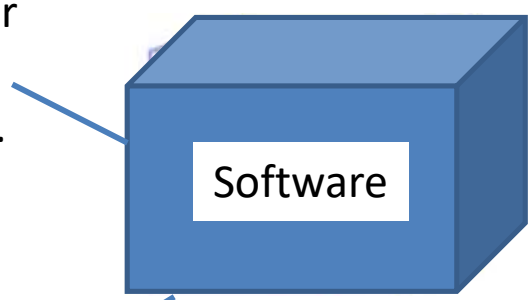
- For any test of an EQ partition (known as P), if an under-test software will demonstrate a failure (or an error), then the other tests of P will lead the software to produce an similar failure.



Why Do We Need the EP Method?

Answer: - Testing cost reduction by reducing tests for each input data.
- Achieve adequate test coverage criteria for EP classes

Exhaustive testing for
every input
Value is not possible.



Too many possible
Input values for each
Input data

Equivalence Partitioning Class Examples

- Different data value ranges of input parameters
- Option values for input data, such as dates, times, country names, ete.
- Different groups of invalid inputs for parameters
- Different groups of equivalent input/output events, such as phone signals
- Equivalent operating environments for an under-test system
- Equivalent classes of a third-party system‘s outputs as inputs
- Number of records in a database (or other equivalent objects
- Equivalent classes of input data values from users
- Equivalent resource types of a system

How to Use Equivalence Partitioning

- For the input value space of an under-test system, please do the following:

Step #1: Identify equivalent input value classes

- All EP classes must be independent, and not overlapping
- All EP classes must cover all of input value space
- EP classes must cover both invalid input values and valid input values

Step #2: Select one test from each EP class, and identify its expected output

TOPIC #2 – EQUIVALENCE PARTITIONING TEST METHOD

An Equivalence Partitioning Example – A Banking Account

For a savings Bank account,

- 3% rate of interest is credited if its balance is in the range of $0 to $100
- 5% rate of interest is credited if its balance is in the range of over $100 to $1000
- 7% rate of interest is credited if its balance is is over $1000 and above

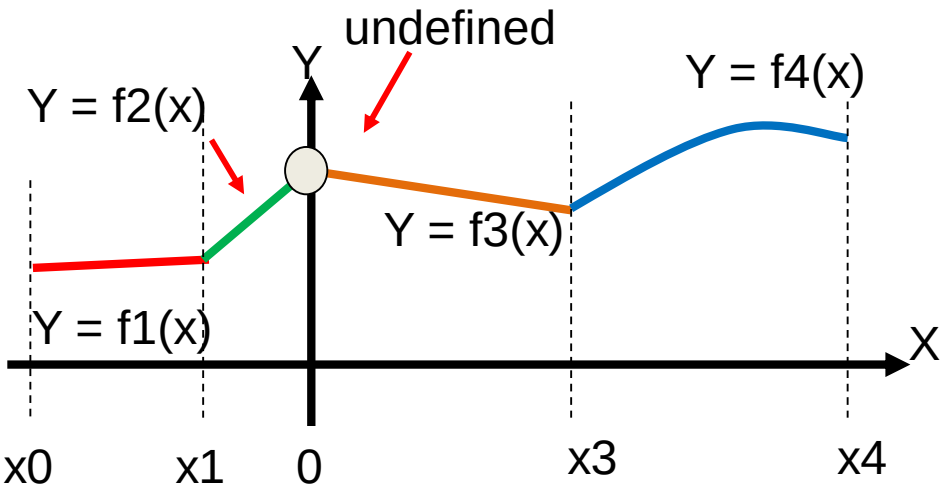
Please identify your EQ Partition classes?

	Partition 1	Partition 2		Partition 3		Partition 4
Account Balance	Invalid partition	Valid (for 3% interest)		Valid (for 5%)		Valid (for 7%)
	-$0.01	$0.00	$100.00	$100.01	$999.99	$1000.00

An Equivalence Partitioning Example

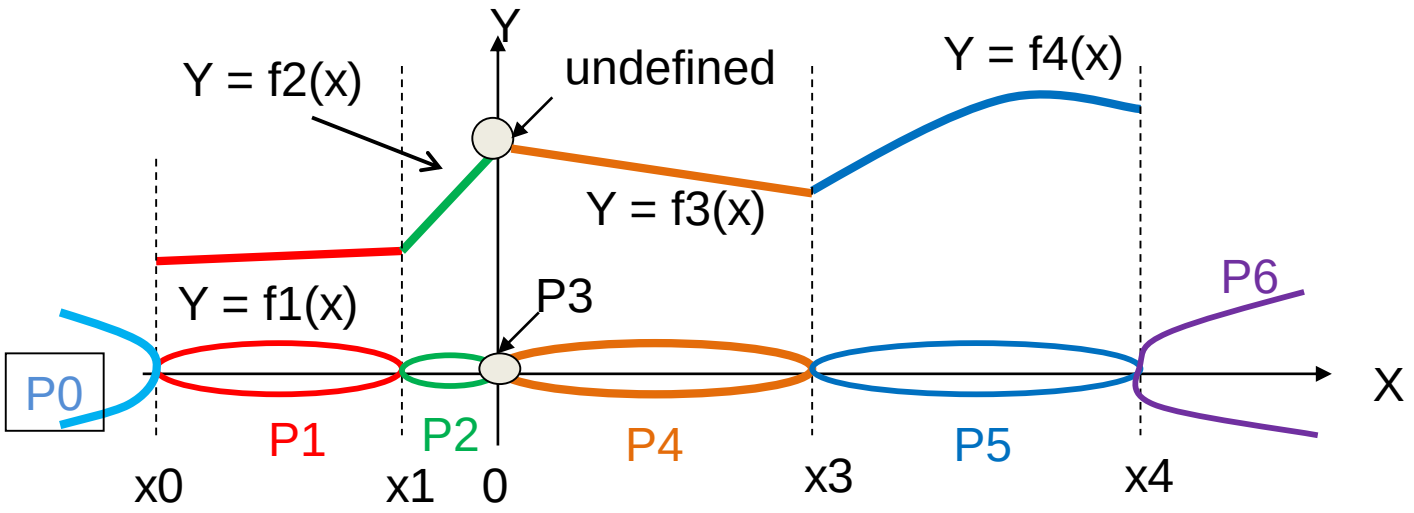
A software component provides a set of application functions.
These functions generate a corresponding Y value based on the value of input data X.
Each function is called and executed when X value is fall into a specific data range.

$$Y(x) = \begin{cases} f1(x) & x \in [x0, x1] \\ f2(x) & x \in (x1, 0) \\ \text{Undefined} & x = 0 \\ f3(x) & x \in (0, x3] \\ f4(x) & x \in (x3, x4] \end{cases}$$



TOPIC #2 – EQUIVALENCE PARTITIONING TEST METHOD

An Equivalence Partitioning Example

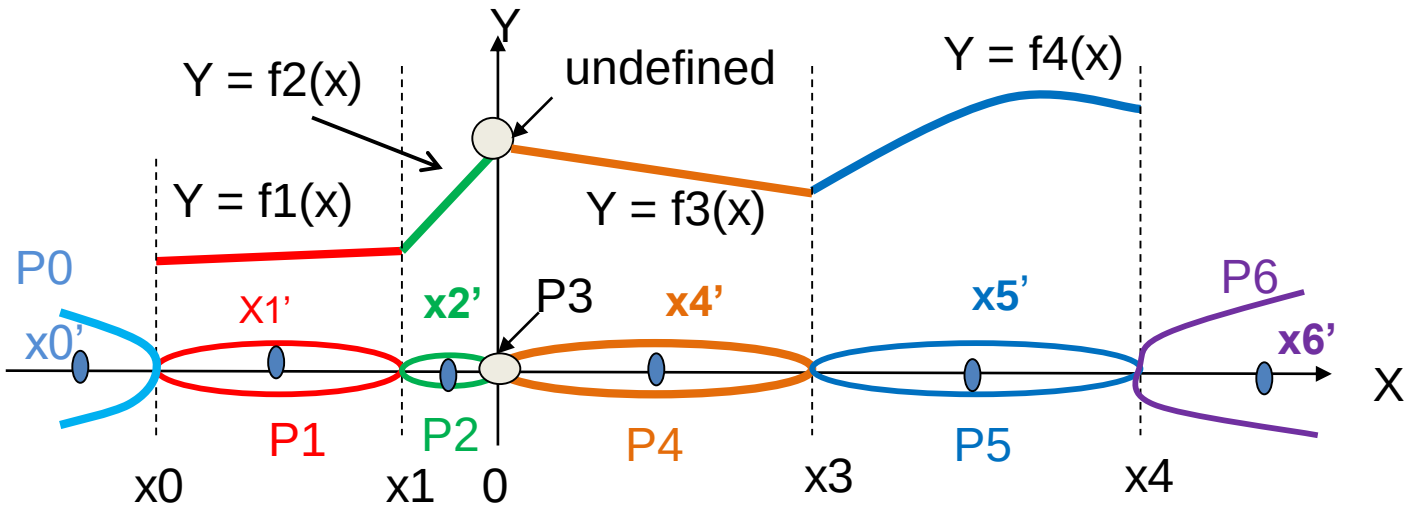


EQ Partitions:

P0: x < x0 or x in (x0, Very Small No.)
P1: x in [x0, x1]
P2: x in (x1, 0)
P3: x = 0
P4: x in (0, x3]
P5: x in (x3, x4]
P6: x > x4or x in (x4, Very Larger No.)

TOPIC #2 – EQUIVALENCE PARTITIONING TEST METHOD

An Equivalence Partitioning Example



Test Cases for EQ Partitions:

x= x1',	y = f1(x1') = y1'
x = x2',	y = f2(x2') = y2'
x = 0,	y = undefined
x = x4',	y = f3(x4') = y3'
x = x5',	y = f4(x5') = y4'
x = x0',	y = Outside value boundary
x = x6',	y = Outside value boundary

Jerry Gao Ph.D.

Equivalence Partitioning Summary

Advantage:

- Simple to use
- Cost reduction for software testing by selecting one test for each EP class

Test Coverage:

Using Equivalence Partitioning, we can achieve EP class test coverage:

For each EP class (or partition), there must be at least one test.

Limitations:

Very difficult to apply when under-test software has many Input data parameters, and multi-dimension input value spaces.

Challenges:

- Lack of step-by-step tips to identify EP classes for a software
- Difficult to deal with a complicated input data set

Topic #3 – Boundary Value Testing Method

Instructor: Jerry Gao, Ph.D., Professor
San Jose State University

Topic #3 – Boundary Value Testing Method

What is Boundary Value Testing Method?

Why Do We Need Boundary Value Testing?

How to Use Boundary Value Testing?

Boundary Value Testing Examples

Boundary Value Testing Summary

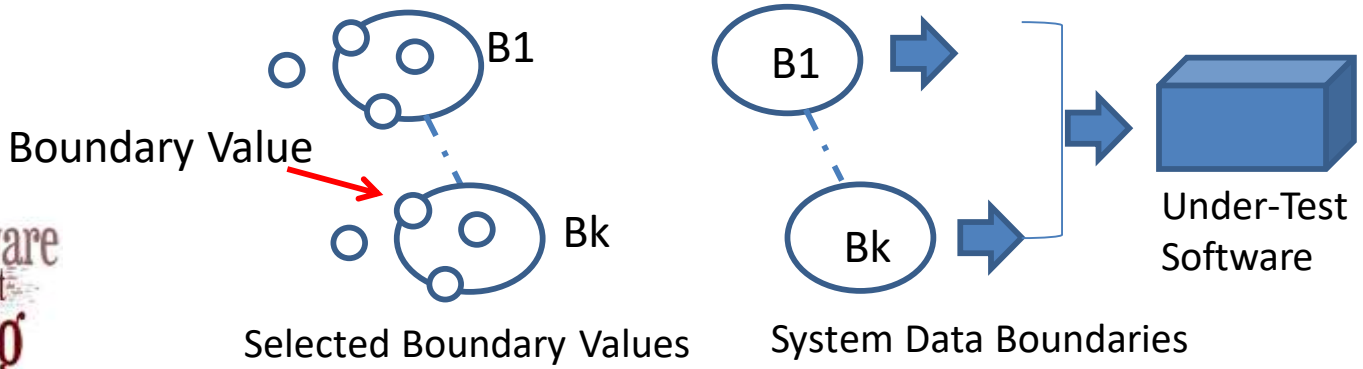
TOPIC #3 – BOUNDARY VALUE TEST METHOD

What is boundary value testing method?

Boundary value testing is also known as boundary value analysis (BVA).
It is one of the best-known functional test techniques in black-box testing.

Definition:
Boundary value analysis is a software testing technique in which tests are designed to validate software functions and behaviors by focusing on the representative boundary values for each system boundary.

Traditional boundary value analysis focuses on input or output values at the extreme boundary conditions of independent physical variables.

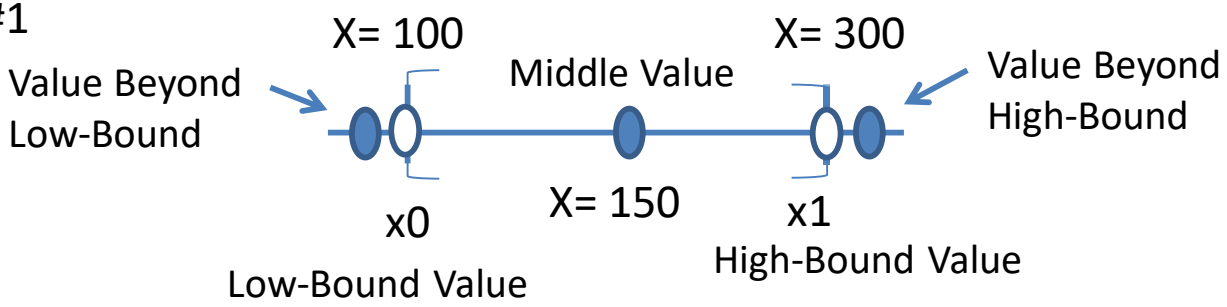


Different Types of Boundary in Software

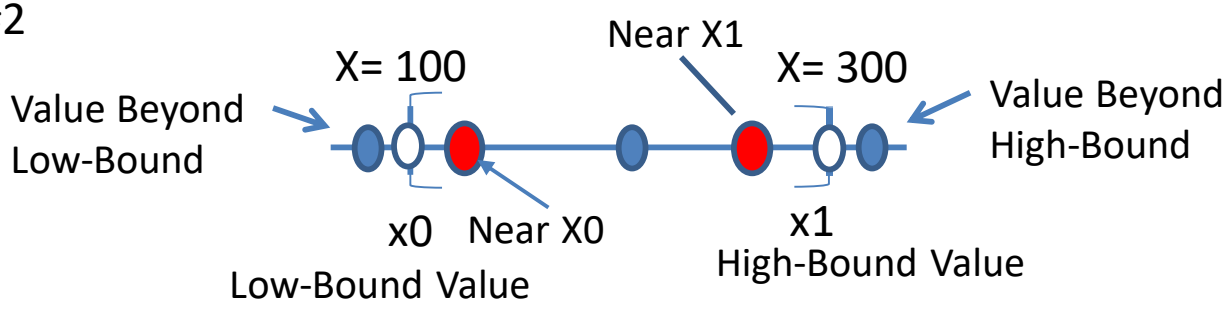
One-dimension input data boundary:

- Boundary: X in [100, 300] Boundary Value Examples: X = 100, X =300

Approach #1

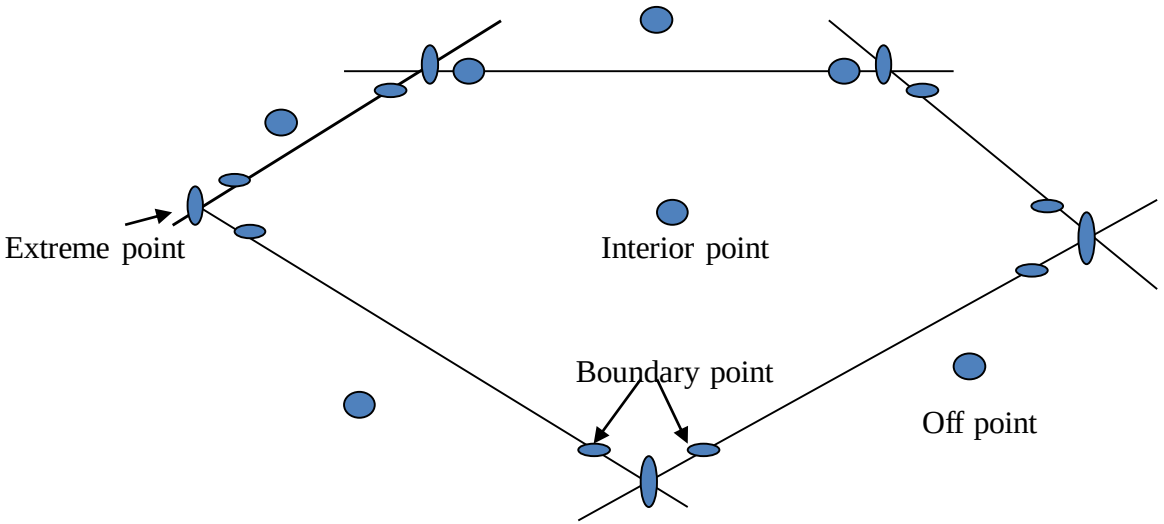


Approach #2



Different Types of Boundary in Software

Two-dimension input data boundary:



Two-dimension Domain Boundary

Why Do We Need the Boundary Value Analysis?

Answer:

- Historical evidence demonstrates errors tend to occur near the extreme boundaries of input and/or output values of physical variables.

- Achieve adequate boundary value test coverage criteria

- Reduce the test complexity by only focusing on boundary values of the under-test system.

System Boundary Examples

- Different data value ranges of input parameters
- Total counting parameters
- Aggregated data sizes: such as tree nodes, number of nodes in a linked list
- 2-dimensional boundary values
- 3-dimensional boundary values
- Condition boundaries of a system
- Resource boundaries for a system

How to Use the Boundary Value Test Method?

Step #1: Identify all of system boundaries

Step #2: For each boundary (i.e. B), identify representative boundary values

- Select representative boundary values on B.
- Select representative values beyond each lower bound of B.
- Select representative values beyond each higher bound of B.

Step #3: Define tests for identified boundary values

TOPIC #3 – BOUNDARY VALUE TEST METHOD

A Boundary Value Testing Example

Example 2 for Boundary Value Analysis :

A name text box for users to allows them to enter 1 to 30 characters.

Input Data	Beyond Low-Bound	Low-Bound	Middle Length	High-Bound	Beyond High-Bound
User Name	Text Length = 0	Text Length = 1	Text Length = 15	Text Length = 30	Text Length = 31
Boundary Values	BV1: String with 0 length	BV2: String with 1 char	BV3: String with 15 chars	BV4: String with 30 chars	BV4: String with 31 chars

TOPIC #3 – BOUNDARY VALUE TEST METHOD

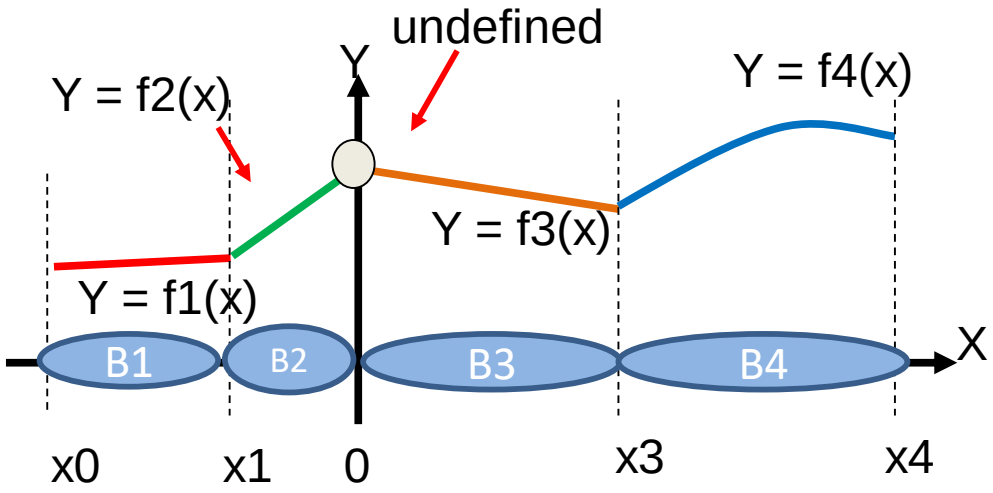
A Boundary Value Testing Example

A software component provides a set of application functions.
These functions generate a corresponding Y value based on the value of input data X.
Each function is called and executed when X value is fall into a specific data range.

$$Y(x) = \begin{cases} f1(x) & x \text{ in } [x0, x1] \\ f2(x) & x \text{ in } (x1, 0) \\ \text{Undefined} & x = 0 \\ f3(x) & x \text{ in } (0, x3] \\ f4(x) & x \text{ in } (x3, x4] \end{cases}$$

Existing Boundaries:

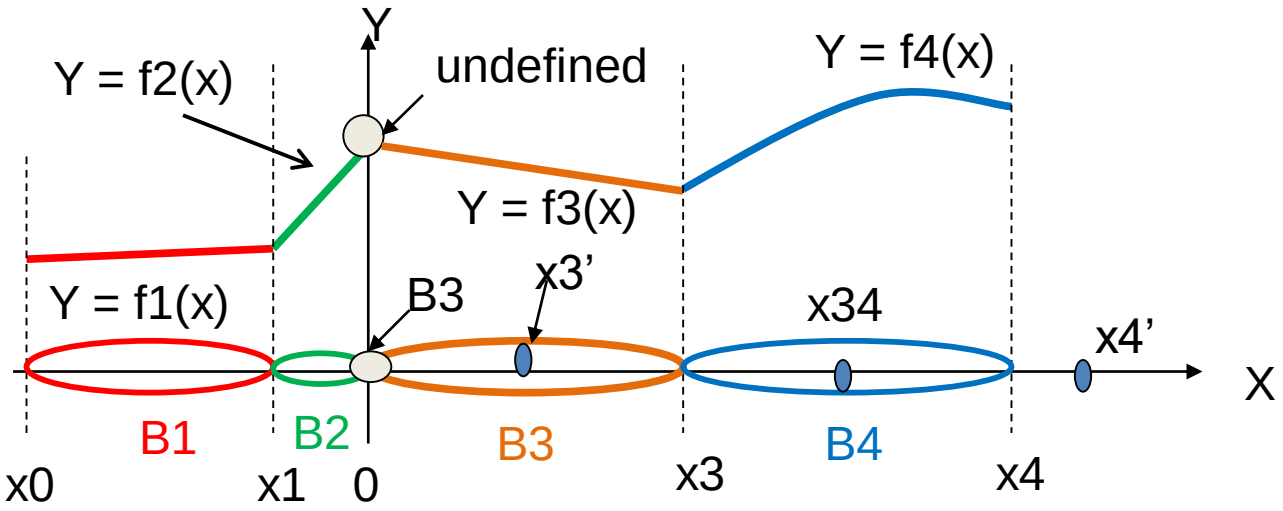
B1: x in [x0, x1]
B2: x in (x1, 0)
B3: x in (0, x3)
B4: x in [x3, x4]



Jerry Gao, Ph.D.

TOPIC #3 – BOUNDARY VALUE TEST METHOD

A Boundary Value Testing Example



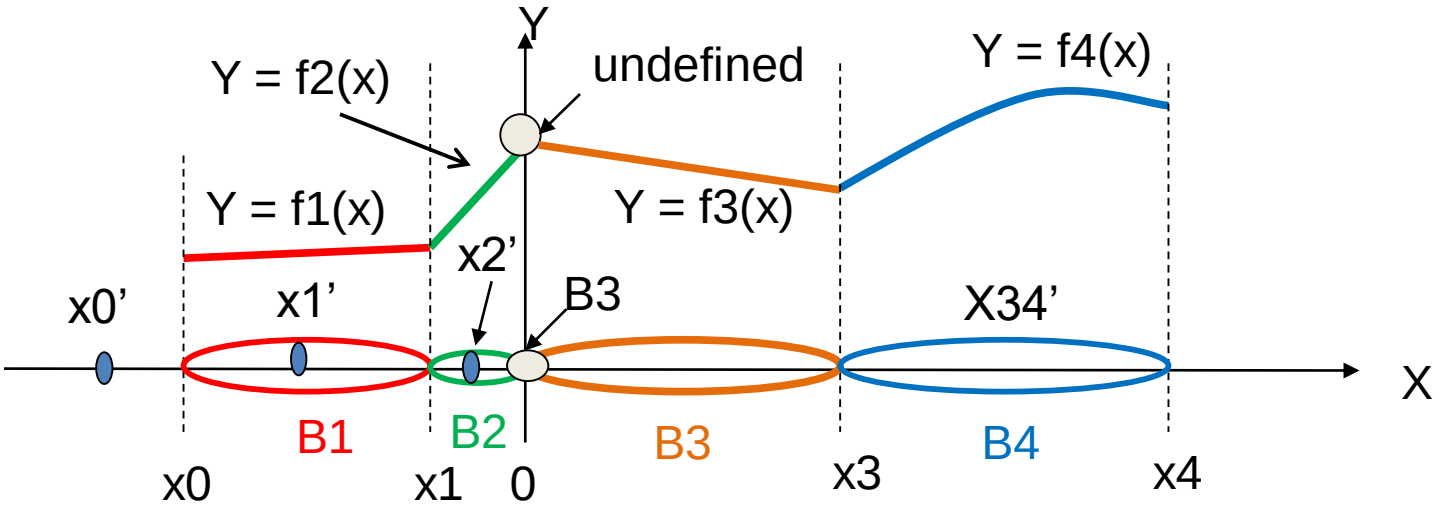
Test Cases for Boundary #4:

x = x3', y = f3(x3'),	check y =?
x = x3, y = f4(x3),	check y =?
x = x34, y = f4(x34),	check y=?
x = x4, y = f4(x),	check y=?
x =x4', y = out of boundary	

Jerry Gao, Ph.D.

TOPIC #3 – BOUNDARY VALUE TEST METHOD

A Boundary Value Testing Example



Test Cases for Boundary #1:

x = x0',	y = Out of boundary	check y =?
x = x0,	y = f1(x0),	check y =?
x = x1',	y = f1(x1'),	check y=?
x = x1,	y = f1(x1),	check y=?
x =x2',	y = f2(x2')	check y=?

TOPIC #3 – BOUNDARY VALUE TEST METHOD

Boundary Value Testing Summary

Advantage:

- Simple and easy to understand and use

Test Coverage:

- The boundary value method assures the boundary value test coverage.
- For each value boundary, there must be adequate boundary value tests to cover it.

Limitations:

Only effective to detect boundary value related software errors.

Challenges:

- It is difficult to identify all boundaries for a software when the given requirements are incomplete.

Topic #4 – Decision Table Testing Method

Instructor: Jerry Gao, Ph.D., Professor
San Jose State University

Topic #4 – Decision Table Testing Method

What is Decision Table Testing Method?

Why Do We Need Decision Table Testing?

How to Use Decision Table Testing?

Decision Table Testing Examples

Decision Table Testing Summary

TOPIC #4 – DECISION TABLE TEST METHOD

What is decision table testing method?

A **decision table** is a good way to deal with combinations of things (e.g. inputs). This technique is sometimes also referred to as a 'cause-effect' table.

Definition:
Decision table testing is black box test design technique to determine the test scenarios for complex business logic. In decision table testing the test cases are designed to execute the combinations of inputs and/or stimuli (causes) from the decision table.

A decision table is a table of rows and columns separated into four quadrants .

Decision Stub	**Rules Stub**
Action Stub	**Entries Stub**

Importance of Decision Table Test Method

- It helps testers to search the effects of combinations of different inputs and other software states that must correctly implement business rules.

- Provides a regular way of stating complex business rules, that's helpful for developers as well as for testers.

- It is a structured exercise to prepare requirements when dealing with complex business rules.

TOPIC #4– DECISION TABLE TEST METHOD

Number of Rules

- Each condition generally has two possible outcomes either **YES** or **NO**
- Total number of rules is equal to

  2 ^ no. of conditions

- For example, if there are four conditions then, there will be sixteen possible rules.

Creating a Decision Table

Steps on how to create a simple decision table using the Triangle Problem.

1. Step One – List All Stub Conditions

In this example we take three inputs, and from those inputs we perform conditional checks to calculate if it's a triangle, if so then what type of triangle it is.

2. Step Two – Calculate the Number of Possible Combinations (Rules)

So in our table we have 4 condition stubs and we are developing a limited entry decision table so we use the following formula:

Number of Rules = 2 (power) Number of Condition stubs, So therefore

Number of Rules = 2^4 = 16

So we have 16 possible combinations in our decision table.

Creating a Decision Table

Place all the combinations in the decision table

Rules

		R1	R1	R1	R1	R1	R1	R1	R1	R1
Conditions :	C1: <a, b,c > forms a triangle?	F	T	T	T	T	T	T	T	T
	C3: a = b?	–	T	T	T	T	F	F	F	F
	C4: a = c?	–	T	T	F	F	T	T	F	F
	C5: b = c?	–	T	F	T	F	T	F	T	F
Actions :	A1: Not a Triangle	X								
	A2: Scalene									X
	A3: Isosceles					X		X	X	
	A4: Equilateral		X							
	A5: Impossible			**X**	**X**		**X**			

Entries

TOPIC #4 – DECISION TABLE TEST METHOD

Few more conditions added to the decision table.

Rules

Conditions :	C1-1: a < b+c?	F	T	T	T	T	T	T	T	T	T	T
	C1-2: b < a+c?	–	F	T	T	T	T	T	T	T	T	T
	C1-3: c < a+b?	–	–	F	T	T	T	T	T	T	T	T
	C2: a = b?	–	–	–	T	T	T	T	F	F	F	F
	C3: a = c?	–	–	–	T	T	F	F	T	T	F	F
	C4: b = c?	–	–	–	T	F	T	F	T	F	T	F
Actions:	A1: Not a Triangle	X	X	X								
	A2: Scalene											X
	A3: Isosceles							X		X	X	
	A4: Equilateral				X							
	A5: Impossible					X	X		X			

Entries

Test cases from Decision Table

Case ID	a	b	c	Expected Output
DT1	4	1	2	Not a Triangle
DT2	1	4	2	Not a Triangle
DT3	1	2	4	Not a Triangle
DT4	5	5	5	Equilateral
DT5	???	???	???	Impossible
DT6	???	???	???	Impossible
DT7	2	2	3	Isosceles
DT8	???	???	???	Impossible
DT9	2	3	2	Isosceles
DT10	3	2	2	Isosceles
DT11	3	4	5	Scalene

Creating a Decision Table

Example 2 : Credit Card

If you are a new customer and you want to open a credit card account then there are three conditions first you will get a 15% discount on all your purchases today, second if you are an existing customer and you hold a loyalty card, you get a 10% discount and third if you have a coupon, you can get 20% off today (but it can't be used with the 'new customer' discount).

Question :

a. Create a decision table for the above scenario with the list of possible conditions and actions.
b. Calculate the number of possible rules for the decision table.
c. Derive test cases from the table constructed.

Creating a Decision Table

Example 2 : Credit Card

Conditions	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6	Rule 7	Rule 8
New customer (15%)	T	T	T	T	F	F	F	F
Loyalty card (10%)	T	T	F	F	T	T	F	F
Coupon (20%)	T	F	T	F	T	F	T	F
Actions								
Discount (%)	X	X	20	15	30	10	20	0

TOPIC #4 – DECISION TABLE TEST METHOD

Decision Table Testing Summary

Advantage:

•Easy to understand
•Map nicely to a set of business rules
•Applied to real problems
• Able to process both numerical and categorical data

Test Coverage:

- The decision table value method assures the decision table test coverage.
- For each rule in the table, there is a test case derived.

Limitations:

Limited to one output attribute
Decision tree algorithms are unstable
Trees created from numeric datasets can be complex.

Challenges:

- It is difficult to identify all the conditions in a given scenario.

Topic #5 – Scenario Testing Method

Instructor: Jerry Gao, Ph.D., Professor
San Jose State University

TOPIC #4 – SCENARIO TESTING METHOD

- What is Scenario Testing?
- Why Is Scenario Testing Important?
- How to Perform Scenario Testing?
- Scenario Testing Examples
- Scenario Testing Summary

TOPIC #4 – SCENARIO TESTING METHOD

What is scenario-based testing?

Scenario testing is done by creating test scenarios which replicate the end users usage scenarios based on the given system requirements and use cases.

A test scenario can be a independent test case or a series of test cases that makes a test suit. Test scenario is just a story which explains the usage of the software by any end user.

In scenario testing, testers needs to communicate with system users, clients, stakeholders, and developers to come up the user scenarios first, then create test scenarios.

In scenario testing the testers put themselves in the end users shoes and figure out the real world scenarios or consider the use cases which can be performed on the software by end users.

TOPIC #4 – SCENARIO TESTING METHOD

Importance of Scenario Testing

- Since scenario testing focus on system usage scenarios, it helps engineers to detect many system usage problems/bugs.
- Scenario testing is very useful to check user-oriented system transactions, for example, online banking transactions.
- Because scenario testing requires engineers focus on system usage contexts, it is also very effective to discover system context problems on a customer site.
- Scenario testing could be useful to end-to-end system functions and user-oriented constraints and workflow operations.

TOPIC #4 – SCENARIO TESTING METHOD

How to Perform Scenario Testing?

Scenario Testing Steps:

Step #1 – Identify use cases/user operation scenarios

Step #2 - Define test cases based on the identified scenarios

Step #3 - Execute scenario tests

Identify system usage scenarios based on system use cases and user operations by using the following ways:

1. Story lines
2. State transitions
3. Business verticals
4. Implementation story from customers

You can apply the following approach:

Use-case and role-based scenarios : This method focuses on how a user uses the system with different roles and environment.

TOPIC #4 – SCENARIO TESTING METHOD

Scenario Creation Procedure

Below template can be used to formulate the scenario

#	Step Description
1	Find all actors (roles played by persons/external systems) interacting with the system
2	Find all (relevant system external) events
3	Determine inputs, results and output of the system
4	Determine system boundaries
5	Create coarse overview scenarios (instance or type scenarios on business process or task level)
6	Prioritize scenarios according to importance, assure that the scenarios cover system functionality
7	Create a step-by-step description of events and actions for each scenario (task level)
8	Create an overview diagram and a dependency chart (see section 3.3)
9	Have users review and comment on the scenarios and diagrams
10	Extend scenarios by refining the scenario description, break down tasks to single working steps
11	Model alternative flows of actions, specify exceptions and how to react to exceptions
12	Factor out abstract scenarios (sequences of interactions appearing in more than one scenario)
13	Include non-functional (performance) requirements and qualities in scenarios
14	Revise the overview diagram and dependency chart
15	Have users check and validate the scenarios (Formal reviews)

Scenario Testing Example

Example 1: Authentication Scenario in ATM

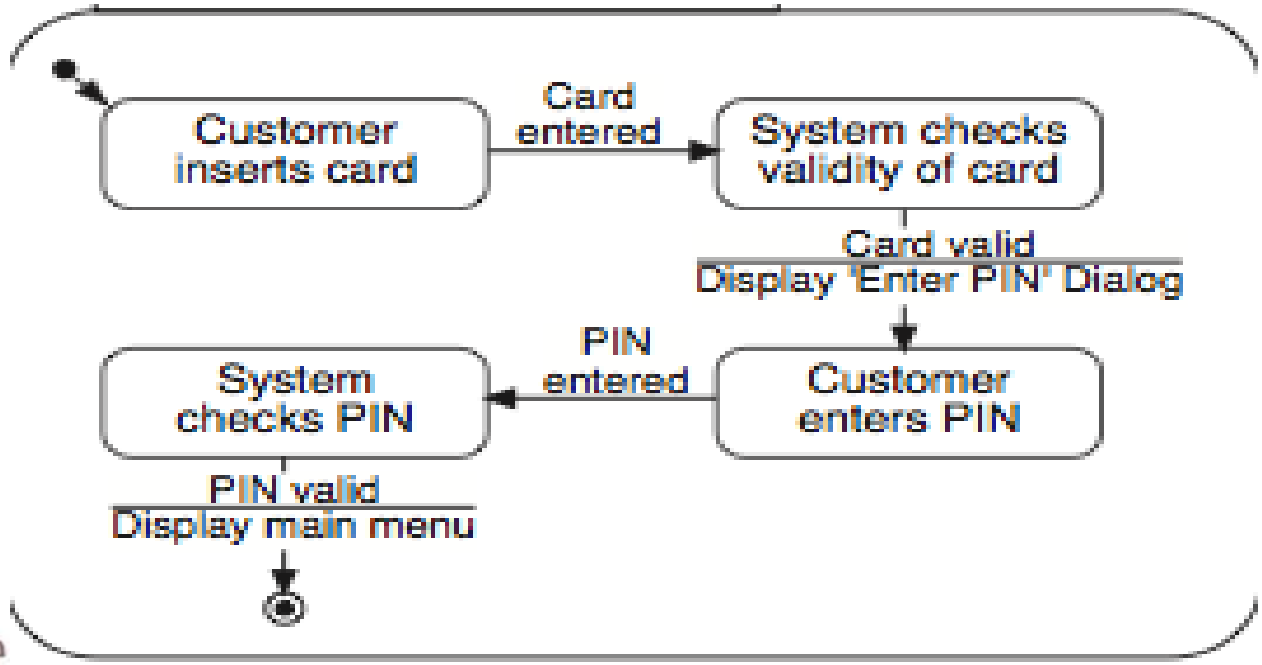
a. The Authentication scenario reads
Precondition: The ATM is operational, card being inserted.

1. The customer inserts the card
2. The system checks the card's validity
3. The system displays the "Enter PIN" Dialog
4. The customer enters his PIN
5. The system checks the PIN
6. The system displays the main menu

TOPIC #4 – SCENARIO TESTING METHOD

Scenario Testing Example

The authentication scenario depicted in flow chart.



Scenario Testing Example

Test case derived from the scenario:

Test preparation:		ATM operational, card and PIN (1234) have been issued, card is being inserted	
ID	*State*	*Input/User actions/ Conditions*	*Expected output*
1.1	Card sensed	Card can be read, card valid, valid PIN (1234) entered in time	Main menu displayed
1.2	Card sensed	Card can be read, card valid, invalid PIN (1245) entered in time (first try)	Retry message displayed
1.3	Retry msg	Invalid PIN (123) entered in time, second try	Retry message
1.4	Retry msg	Invalid PIN (1234567) entered in time, third try	Card retained, user informed
…	…	…	…

Scenario Testing Summary

Advantage:

- Helps to find bugs at the application level from user perspectives
- Easy to understand and apply based on user-centered understanding
- Reduces the chances of repeatability
- Understand the complexity of the application easily

Test Coverage:

- Scenario-based test coverage based on usage scenarios

Limitations:

- Scenario testing only useful for system level testing
- Depending on engineers' understanding of the under-test system

Challenges:

- How to achieve the completed coverage for all of system scenarios?

Topic #4 – Category Partition Testing Method

Instructor: Jerry Gao, Ph.D., Professor
San Jose State University

Topic #4 – Decision Table Testing Method

What is Category Partition Testing?

Category Partition Main Characteristics

How to Perform Category Partition Methods

Category Partition Testing Examples

Category Partition Testing Summary

TOPIC #4 – CATEGORY PARTITION TEST METHOD

What is category partition testing method?

Category Partition Method **[CPM**] is a specification based testing technique helping the testers create test cases by refining the functional specification of a program into test requirements.

The idea behind **category partition** testing is to divide the input domains of a component into N different disjoint partitions and select one value from each input domain to create a test case.

This method emphasizes both the specification **coverage** and **error detection** aspects of testing

TOPIC #4 – CATEGORY PARTITION TEST METHOD

Main Characteristics

- The test specification is concise and uniform representation of test information for a function
- The partition can be easily modified and gives the tester a logical way to control the volume the tests.
- The generator tool provides an automated way to produce thorough tests which avoids impossible or undesirable tests.

TOPIC #4 – CATEGORY PARTITION TEST METHOD

Category Partition Systematic Method

According to T.J Ostrand and M.J.Balcer , he proposed a systematic method consisting of the following steps :

- ✓Decompose function specifications into functional units.
- ✓Identify parameters and environment conditions.
- ✓Find categories of information.
- ✓Partition each category into choices.
- ✓Write test specification for each unit.
- ✓Produce test frames.
- ✓Generate test cases.

Partition Testing Example

Example 1 : Test Specification for ATM – PIN Number

Partition:

PIN

Wrong PIN [property mismatch]
Correct PIN [property match]
Withdraw amount
Multiple of 20 [if match]
[property correct]
Less than 20 [if match]
[property wrong]
Greater than 20 but not multiple of 20
[if match]
[property wrong]

An Example of using Category Partition Method

Example 2:
Test a command-line program that supports “find” operation as follows:

Command: find
Syntax: find <pattern> <file>

Function Specifications:
The find command is used to locate one or more instances of a given pattern in a text file.

- All lines in the file that contain the pattern are written to standard output.
- A line containing the pattern is written only once, regardless of the no. of times the pattern occurs in it.
- The pattern is any sequence of characters whose length does not exceed the maximum length of a line in the file.
- To include a blank in the pattern, the entire pattern must be enclosed in quotes (“). To include a quotation mark in the pattern, two quotes in a row (“ ”) must be used.

An Example of using Category Partition Method

Examples:

- find john myfile
  - displays lines in the file myfile which contain *john*
- find “john smith” myfile
  - display lines in the file myfile which contains john smith.
- find “john” ” smith” myfile
  - display lines in the file which contains john” smith.

When file is considered as a parameter, we need to consider the following:- no. of occurrences of the pattern in the file.- no. of occurrences of the pattern in a line that contains it.- maximum line length in the file

An Example of using Category Partition Method

Test specification for Find command:

Parameters :

1. **Pattern Size:**
   - ✓ empty
   - ✓ single character
   - ✓ many character
   - ✓ longer than any line in the file

2. **Quoting:**
   - ✓ Pattern is quoted
   - ✓ Pattern is not quoted
   - ✓ Pattern is improperly quoted

TOPIC #4 – CATEGORY PARTITION TEST METHOD

An Example of using Category Partition Method

Test specification for Find command:

Parameters:

1. **Embedded blanks:**
   - ✓ No embedded blank
   - ✓ One embedded blank
   - ✓ Several embedded blanks

2. **Embedded quotes:**
   - ✓ No embedded quotes
   - ✓ One embedded quotes
   - ✓ Several embedded quotes

Parameters:

File name:

- ✓ Good File name
- ✓ No File name
- ✓ Omitted

Environment:

(only for the pattern)

File access environment:

- ✓ File not accessible
- ✓ File can't read
- ✓ File can't open

An Example of using Category Partition Method

Parameters:

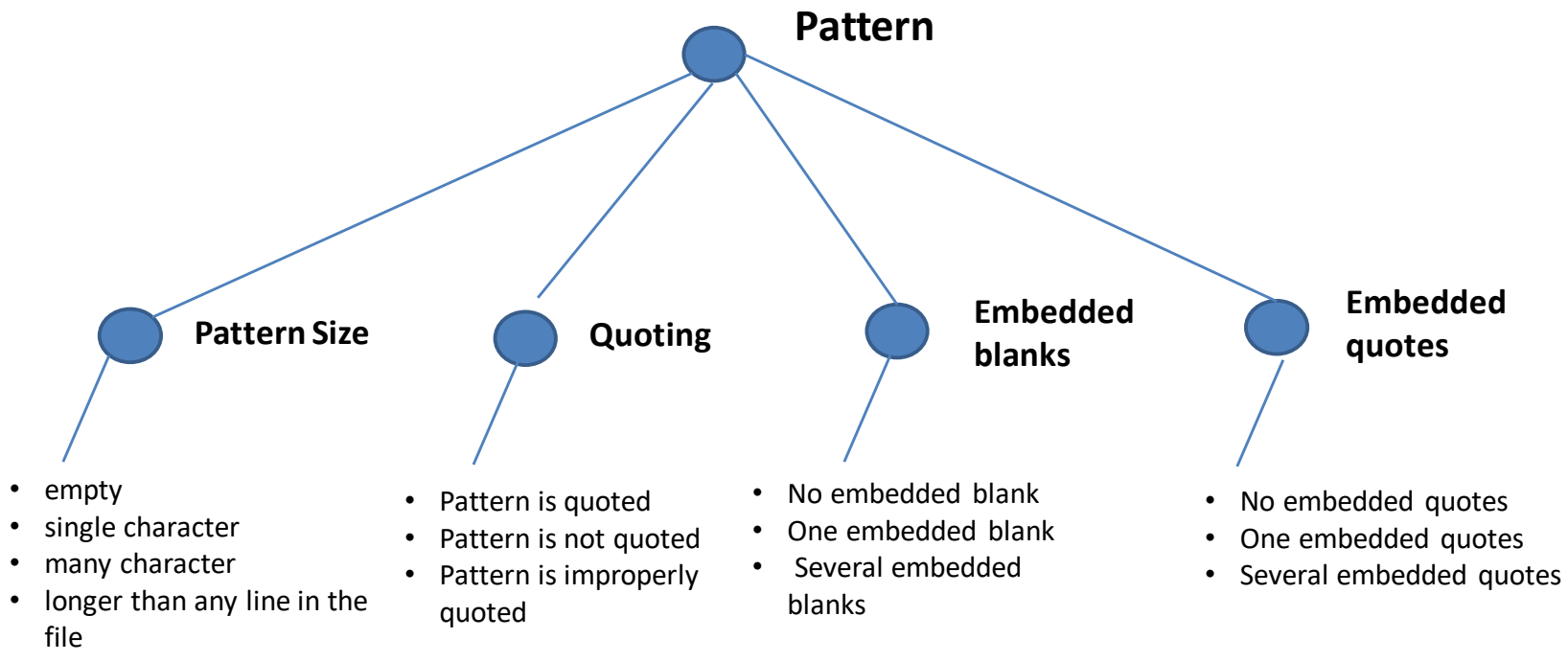
Environment: (only for the pattern)

1. **No. of occurrences of pattern in the file:**
   - ✓ None
   - ✓ Exactly one
   - ✓ More than one

2. **Pattern occurrences on target line:**
   - ✓ One
   - ✓ None
   - ✓ More than one

TOPIC #4 – CATEGORY PARTITION TEST METHOD

An Example of using Category Partition Method



TOPIC #4 – CATEGORY PARTITION TEST METHOD

An Example of using Category Partition Method

Example 2 :

- Specification: The program prompts the user for a positive integer in the range 1 to 20 and then for a string of characters of that length. The program then prompts for a character and returns the position in the string at which the character was first found or a message indicating that the character was not present in the string. The user has the option to search for more characters.

TOPIC #4 – CATEGORY PARTITION TEST METHOD

An Example of using Category Partition Method

Example 2:

Parameters and Categories

- Three parameters: integer x (length), the string a, and the character c
- For x the categories are "in-range" (1-20) or "out-of-range"
- Categories for a: minimal, maximal, intermediate length
- Categories for c: character appears at the beginning, middle, end of string, or does not occur in the string

An Example of using Category Partition Method

Example 2:

Formal Test Specifications

x:		
1)	0	[error]
2)	1	[property stringok, length1]
3)	2-19	[property stringok, midlength]
4)	20	[property stringok, length20]
5)	21	[error]
a:		
1)	Length 1	[if stringok and length1]
2)	Length 2-19	[if stringok and midlength]
3)	Length 20	[if stringok and length20]
c:		
1)	At first position in string	[if stringok]
2)	At last position in string	[if stringok and not length1]
3)	In middle of string	[if stringok and not length1]
4)	Not in string	[if stringok]

An Example of using Category Partition Method

Example 2:

Test Frames and Cases

x 1	x = 0
x 2a1c1	x = 1, a = 'A', c = 'A'
x 2a1c4	x = 1, a = 'A', c = 'B'
x 3a2c1	x = 7, a = 'ABCDEFG', c = 'A'
x 3a2c2	x = 7, a = 'ABCDEFG', c = 'G'
x 3a2c3	x = 7, a = 'ABCDEFG', c = 'D'
x 3a2c4	x = 7, a = 'ABCDEFG', c = 'X'
x 4a3c1	x = 20, a = 'ABCDEFGHIJKLMNOPQRST', c = 'A'
x 4a3c2	x = 20, a = 'ABCDEFGHIJKLMNOPQRST', c = 'T'
x 4a3c3	x = 20, a = 'ABCDEFGHIJKLMNOPQRST', c = 'J'
x 4a3c4	x = 20, a = 'ABCDEFGHIJKLMNOPQRST', c = 'X'
x 5	x = 21

TOPIC #4 – CATEGORY PARTITION TEST METHOD

Category Partition Testing Summary

Advantage:

- The tester can modify the test specification whenever necessary
- Reduce the number of test cases.
- Provides logical way to control the volume of tests.
- Language or implementation independent

Test Coverage:

- Each of the categorized partition has a test case derived from it.
- Test frames are generated which consist of maximum combination of choices in the category that is being partitioned.

Limitations:

- Lack of systematic methods to partition input domains of a component for the given non-formal function specification.
- Does not unearth bugs due to incorrect specifications.

Challenges:

- Identifying the parameters and environments, conditions and categories requires experienced tester.