05 Specification-based test design

PowerPoint by Wanida Khamprapai

Test design techniques

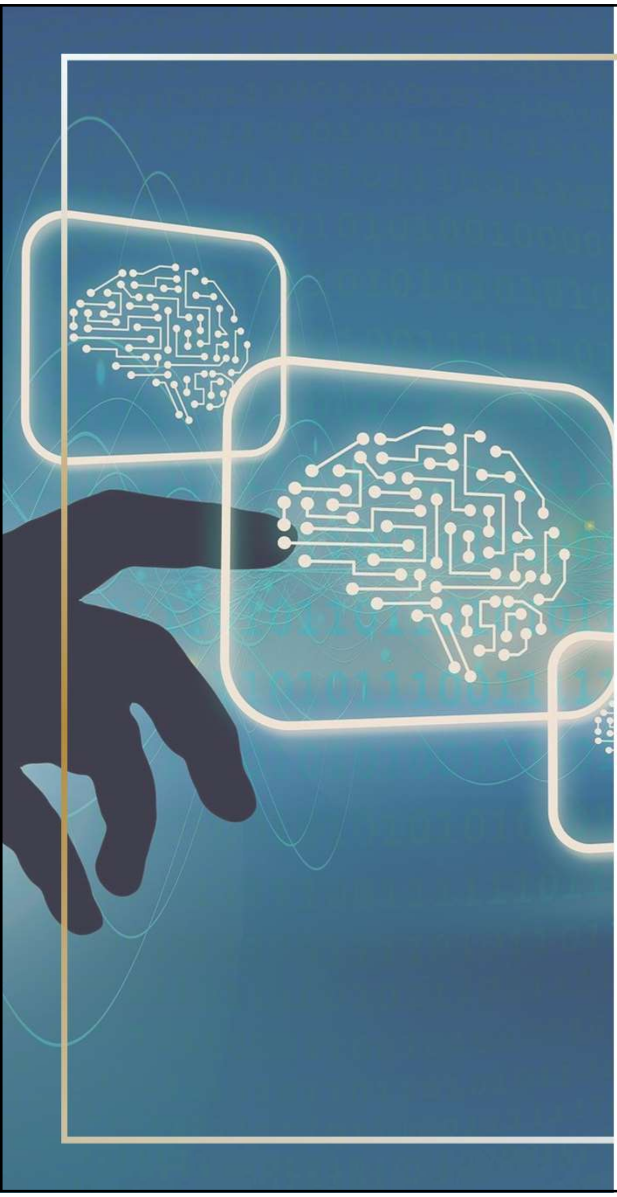
Goal: Select test cases based on test objectives.

Specification-based testing

- SUT: Black-box testing
- Only specification is known
- Testing specified functionality

Structure-based testing

- SUT: White-box testing
- internal structure known
- Testing based on internal behavior



Specification-based testing

Techniques help identify tests that are most likely to find defects with limited test cases. Techniques for designing tests based on specifications are created using the application's function requirements or business rules.

Following techniques are used for input validations done as part of field validation as described earlier.

- Equivalence partitioning
- Boundary value analysis

Following technique is primarily used to see how the system is processing different input combinations

- Cause & effect graphing/decision table following techniques are used to do overall functional testing and verify the system's behavior
- State transition
- Use case based

Specification-based testing

In addition to Applying various techniques based on input types, one can guess the possibility of defects based on experience and intuition on various other aspects or explore the functionality of new applications and evaluate those functionalities based on experience. These techniques are known as

- Error guessing
- Exploratory testing

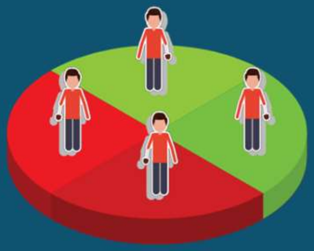
Since the goal of testing is to determine what the application should do (positive testing) and what it should not do (negative testing), it is important that at least one value from each positive class (valid values) and one value from each negative class (invalid value) be used as inputs for testing.

Equivalence partitioning

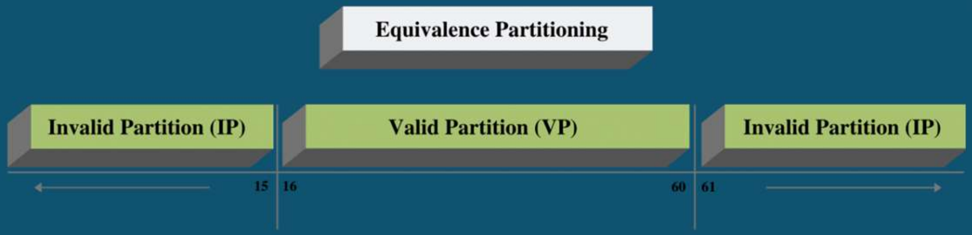
It is a common-sense approach to testing, as most developers/testers use it informally without realizing it. As per this technique, the input data of a software unit is divided into partitions/classes/sets of equivalent data from which test cases can be derived. All the data within the class or partition are equivalent means the system is expected to behave in the same manner for all the data of the partition. Hence, test cases are designed to cover each partition at least once.

Equivalence class or partition

A portion of an input or output domain for which the behavior of a component or system is assumed to be the same, based on the specification.



Equivalence Partitioning



Equivalence partitioning

Mandatory field

For every mandatory field, you will have two classes,

1) Null value
2) Some value

Length checking

For length checking you will have two classes,

1) All values whose length is less than the maximum length allowed
2) All values whose length is more than the maximum length allowed

Type checking

For every type checking, you will have two classes,

1) All values with valid type
2) All values with invalid type

Range checking

You can have 3 or more classes.

1) All values are less than the minimum allowable value
2) All allowable values between minimum and maximum value
3) All values are more than the maximum allowable value

Equivalence partitioning

Example: Travel date (Mandatory field)

Travel date could be next day till one month from next day (Ticket can be booked only up to one month in advance)

Valid class: In the reange day

Invalid class: NULL, <= Aug 10 2023, > Sep 10 2023

Input/test data: Aug 10, 2023
NULL
Jan 15 2023

Equivalence partitioning

Example: การตัดเกรดในรายวิชา Not mandatory field don't need test null

เกรดที่นิสิตได้รับขึ้นอยู่กับคะแนนของนิสิต ตามเกณฑ์ต่อไปนี้

คะแนน < 50	ได้รับเกรด E
คะแนน >= 50 และ < 60	ได้รับเกรด D
คะแนน >= 60 และ < 70	ได้รับเกรด C
คะแนน >= 70 และ < 80	ได้รับเกรด B
คะแนน >= 80	ได้รับเกรด A

Valid class: > 50 , >= 50 - < 60, >= 60 - < 70, >= 70 - < 80, >= 80

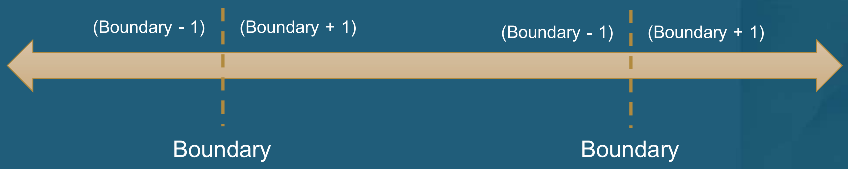
Invalid class: Character, NULL

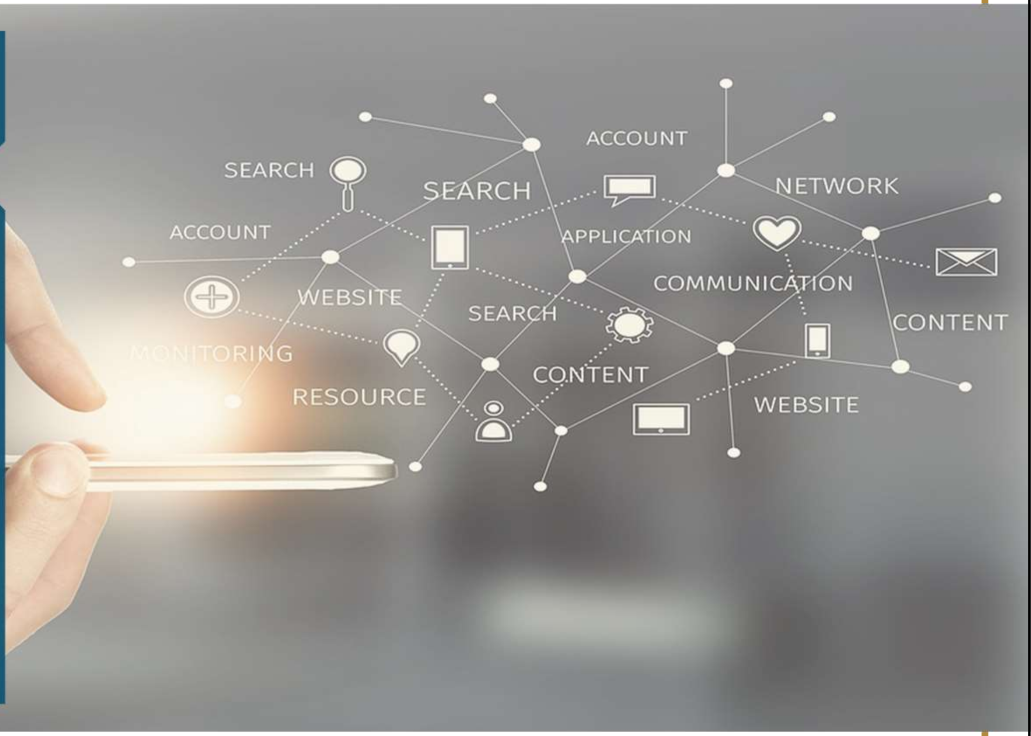
Input/test data: 45, 52, 65, 75, 90, AB, NULL

Test at maximum or minimum values
Select 3 value per boundary

Boundary value analysis

- A black-box test design technique in which test cases are designed based on boundary values.
- Test cases or values at the boundary of each input includes the values at the boundary, below the boundary, and above the boundary.





Boundary value analysis

Example: Income of applicant

The input variable is an applicant's income. The valid range is 15,000 to 40,000 Baht.

Valid class: 15000 - 40000 ……………………………………………………………………

Invalid class: < 15000, > 40000 ……………………………………………………………………

Input/test data: 14999, 15000, 15001 ……………………………………………………………………
39999, 40000, 40001

Boundary value analysis

Example: การตัดเกรดในรายวิชา

เกรดที่นิสิตได้รับขึ้นอยู่กับคะแนนของนิสิต ตามเกณฑ์ต่อไปนี้

คะแนน < 50	ได้รับเกรด E
คะแนน >= 50 และ < 60	ได้รับเกรด D
คะแนน >= 60 และ < 70	ได้รับเกรด C
คะแนน >= 70 และ < 80	ได้รับเกรด B
คะแนน >= 80	ได้รับเกรด A

Valid class: ………………………………………………………………….

Invalid class: Invalid can spread partition like a-z, A-Z ……………………………………………………………..

Input/test data: …..49, 50, 51……………………………………………………

59, 60, 61

69, 70, 71

79, 80, 81

NULL

AB

Decision table

Equivalence partitioning and boundary value analysis techniques are important when you need to select values for validating a single input field. Many times, these techniques are also used to check if there are business rules generating different outputs for different valid inputs (e.g., Examination result) based on a single variable.

However, many times, business decisions are taken based on different combinations of inputs given in more than one field. The processing, calculation, or output could be different for different valid values entered in those fields. Such rules cannot be tested based on equivalence partitioning or boundary values analysis.

Decision table

Decision table testing is a software testing technique used to test system behavior for different input combinations. This is a systematic approach where the different input combinations and their corresponding system behavior (Output) are captured in a tabular form. That is why it is also called as a **Cause-Effect** table where Cause and effects are captured for better test coverage.

A Decision table is a tabular representation of inputs versus rules/cases/test conditions. It is a very effective tool used for both complex software testing and requirements management. A decision table helps to check all possible combinations of conditions for testing and testers can also identify missed conditions easily. The conditions are indicated as True (T) and False (F) values.

Decision table

Types of decision tables

The decision tables are categorized into two types, and these are given below:

Limited Entry

In the limited entry decision tables, the condition entries are restricted to binary values (True and False).

Extended Entry

The condition entries have more than two values. The decision tables use multiple conditions where a condition may have many possibilities instead of only ‘true’ and ‘false’ are known as extended entry decision tables.

Decision table

For applying the decision table, the steps outlined below must be followed.

1. Identify the causes and effects from the specification
2. Prepare decision table
    a) Identify and write down all conditions (of true or false nature)
    b) Identify all the combinations of true and false for these conditions and write as rules
3. Identify the correct outcome (effects) for each combination (rule) and write

Causes	Rule1	Rule2	…	RuleN
Effects				

4. Combine the rules if the alternatives do not make any difference or remove any rules which are not going to be possible.
5. Prepare test cases based on the decision table

Decision table

Example: A movie ticket booking

For a movie ticket booking If (booking is done 5 days in advance or the number of seats booked is 4 or more) and you are a registered member, you get a 10% discount.

Causes	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6	Rule 7	Rule 8
C1: Registered member	T	T	T	T	F	F	F	F
C2: Tickets booked 5 days in advance	T	T	F	F	T	T	F	F
C3: Number of seats booked >= 4	T	F	T	F	T	F	T	F
Effects								
E1: Get 10% discount?	✓	✓	✓	✗	✗	✗	✗	✗
Final discount %	10%	10%	10%	0%	0%	0%	0%	0%

Note: for limited entry decision tables if there are n conditions then there will be 2^n rules. So, if there are 3 conditions, there will $2^3 = 2*2*2 = 8$ rules.

Decision table

Example: A movie ticket booking (Cont.)

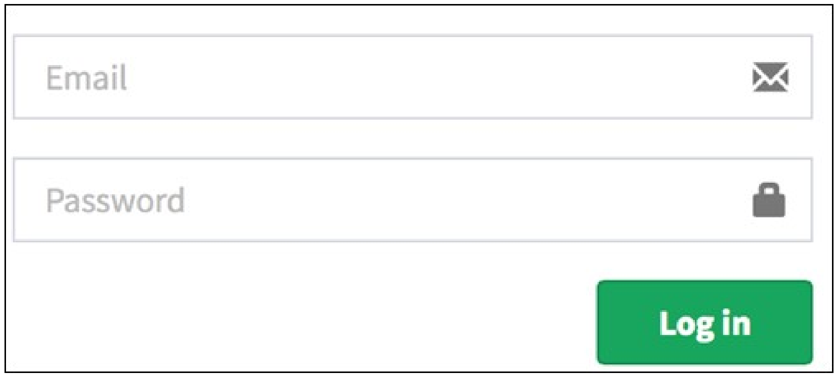
Rule	Test case	Input/test data	Expected result
1	Registered member and ticket booked 5 days in advance and seats >= 4	Registered member, Movie date = August 12, 2023, Number of seats = 5	Discount = 10%
2	Registered member and ticket booked more than 5 days in advance but seats < 4	Registered member, Movie date = August 15, 2023, Number of seats = 2	Discount = 10%
3			
4			
5			
6			
7			
8			

Note: Today is August 7, 2023.

Decision table

Example: Login page

If the user provides the correct username and password the user will be redirected to the homepage. If any of the input is wrong, an error message will be displayed.



Decision table

Example: การตัดเกรดในรายวิชา

เกรดที่นิสิตได้รับขึ้นอยู่กับคะแนนของนิสิต ตามเกณฑ์ต่อไปนี้

คะแนน < 50	ได้รับเกรด E
คะแนน >= 50 และ < 60	ได้รับเกรด D
คะแนน >= 60 และ < 70	ได้รับเกรด C
คะแนน >= 70 และ < 80	ได้รับเกรด B
คะแนน >= 80	ได้รับเกรด A

และนิสิตต้องเข้าเรียนอย่างน้อย 80% ของจำนวนคาบเรียนทั้งหมดในภาคการศึกษา

Questions & Answers