HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Software Quality Assurance
Đảm bảo chất lượng phần mềm

Lecture 4: Black Box Approach for Test Case Designing

Contents
Những nội dung chính

- Motivation of Blackbox Testing
- Equivalence class partitioning
- Boundary value analysis
- Decision Table
- State Transition Testing

4.1. Blackbox Testing

What is Black box testing

- One of testing techniques allowing to **design good test cases**
- Black box testing is a strategy in which testing is based on the requirements and specifications
- Verify the output without inspecting the internal workings
- We do not know:
  - How the box handles errors
  - Whether the inputs are executing all paths of code

Black Box vs. White Box

- Black Box: **functional testing**
  - Unit test
  - Integration test
  - System test
  - Programmers & Test Engineers & Quality Assurance Engineers
- White Box: **structural testing**
  - Unit test
  - Integration test
  - Programmers & Test Engineers

- External/user view:
  - Check conformance with specification
- Abstraction from details:
  - Source code not needed
- Scales up:
  - Different techniques at different levels of granularity

USE

- Internal/developer view:
  - Allows tester to be confident about test coverage
- Based on control or data flow:
  - Easier debugging
- Does not scale up:
  - Mostly applicable at unit and integration testing levels

BOTH!

Black box testing techniques

- Equivalence class partitioning
- Boundary value analysis
- Decision Table
- State Transition Testing

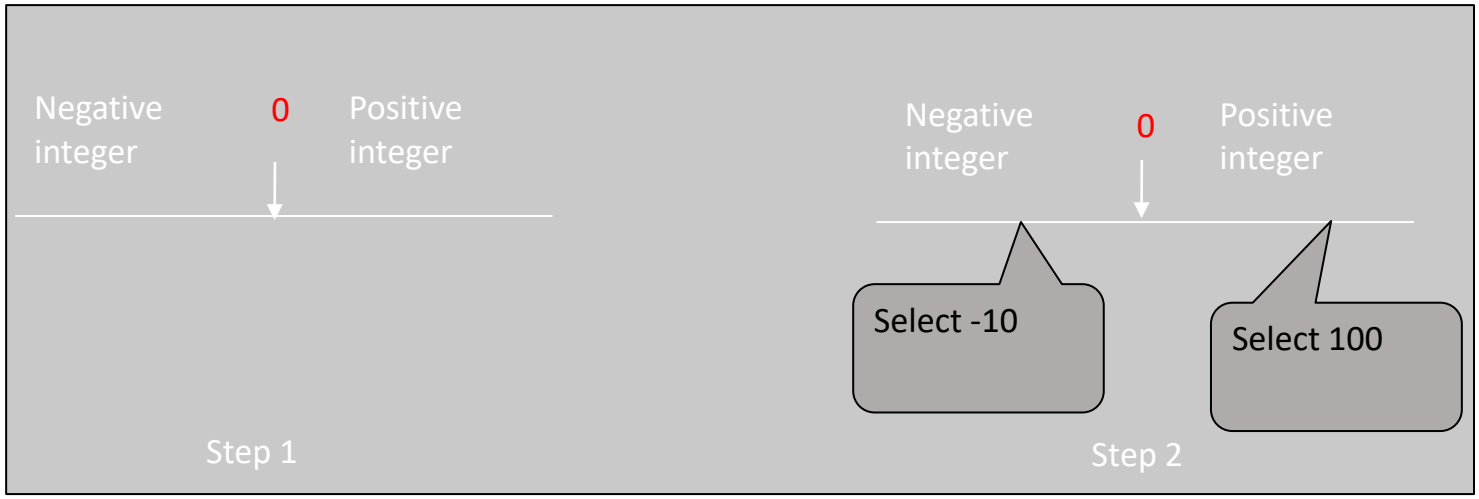
4.2. Equivalence class partitioning (ECP)

Equivalence Class Partitioning
Phân chia lớp tương đương

- A method testing that divides the input domain into **classes of data**
- An Equivalence Class is
  - Consist of a set of data that acts the same way
  - Any data value within class is equivalent
- 2 classes
  - Valid class
  - Invalid class

Equivalent Class Testing
Kiểm thử lớp tương đương

- **Step 1**: Identify all equivalent classes from the requirement and specification of the function
- **Step 2:** Pick up at least 1 value in each equivalence class to create test case



Example: Insurance System

- Specification
  - System shall reject over-age insurance applicants
  - Reject male insurance applicants over the age of 80 years on day of application
  - Reject female insurance applicants over the age of 85 years on day of application
- Identify equivalent classes?

Answer

- Input: Gender & Age
- Output: accept/reject

Classes	Test Cases
C1: Input: Males over 80	T1: male, 83, reject
C2: Input: Males 80 or under	T2: male, 56, accept
C3: Input: Females 85 or under	T3: female, 83, accept
C4: Input: Females over 85	T4: female, 87, reject

- What's about the invalid data?

Equivalence Class Testing Guidelines
Một số chú ý khi sử dụng phân lớp tương đương

If input condition

- is a **range**, e.g., x = [0..9]
- is an **ordered list** of values, e.g., owner = {1, 2, 3, 4}

- ==> one valid and two invalid classes are defined
  - is a set, e.g., vehicle = {car, motorcycle, truck}
  - is a "must be" condition (boolean), e.g., "first character of the identifier must be a letter"
- ==> one valid and one invalid class are defined
  - is anything else
- ==> partition further

Exercise 1: Program to determine employability

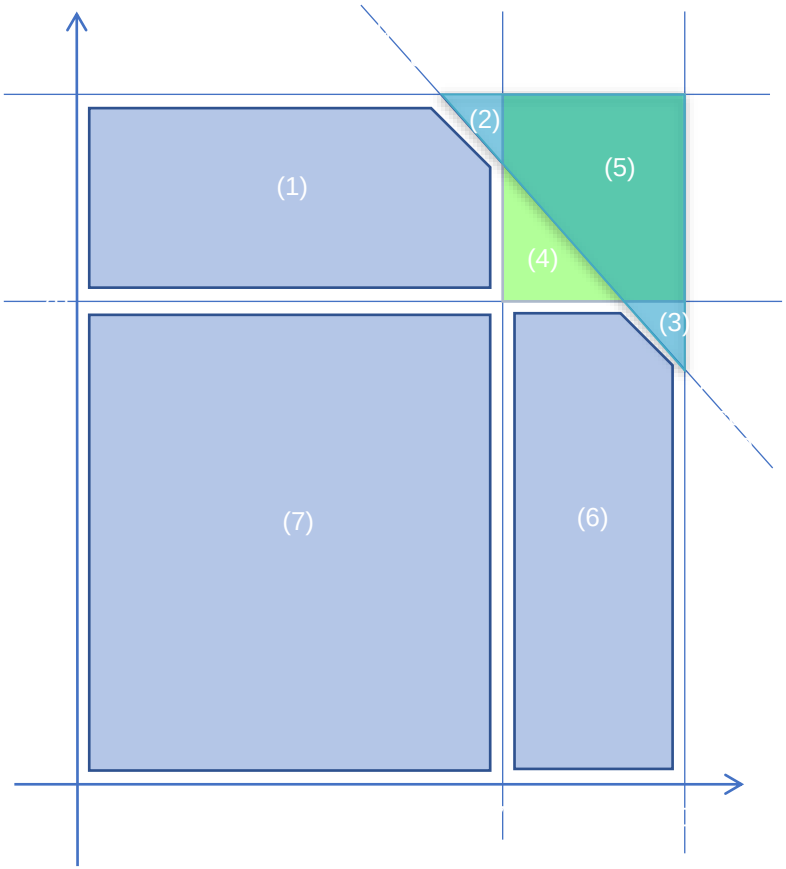
Age	Employment Status
0-15	Don't hire
16-17	Can hire part-time
18-55	Can hire full-time
56-99	Don't hire

How many equivalent classes? How many test case should be designed?

Exercise 2: Examination Judgment Program

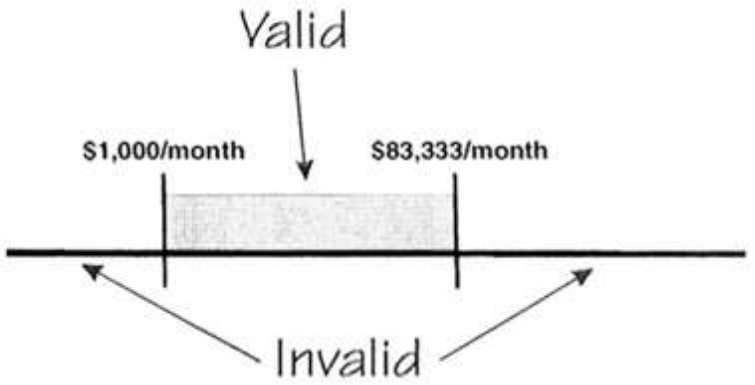
- Specification
  - Two subjects: Maths and Physics
  - Student passed if:
    - Scores of both mathematics and physics are greater than or equal to 70 out of 100
    - average of mathematics and physics is greater than or equal to 80 out of 100
  - Failed if otherwise

Exercise 2



Score	Math.	Physics	Result
(1)	55	85	Failed
(2)	67	97	Passed
(3)	96	68	Passed
(4)	77	80	Passed
(5)	85	92	Passed
(6)	79	58	Failed
(7)	52	58	Failed

Advantage of Equivalence Partitioning



Reduce the number of test case

Ensure **each class** is tested

Disadvantages of Equivalence Partitioning

- The identification of equivalence classes relies on the experiences of tester
- Not consider boundary values

4.3. Boundary Value Analysis (BVA)

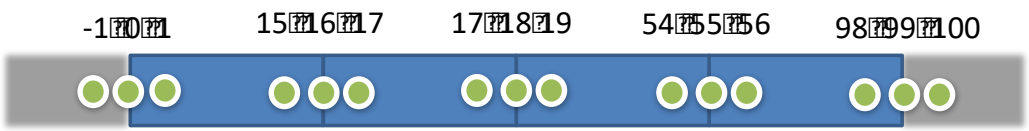
Boundary Value Analysis
Phân tích giá trị biên

- A supplementary method for the equivalence partitioning method
- Allow to select test cases to represent each side of the class boundaries
- Steps:
  - Identify the equivalence classes
  - Identify the boundary of each class
  - Create test case for each boundary value by choosing one point **on** the boundary, one point just **below** the boundary, and one point just **above** the boundary

Example

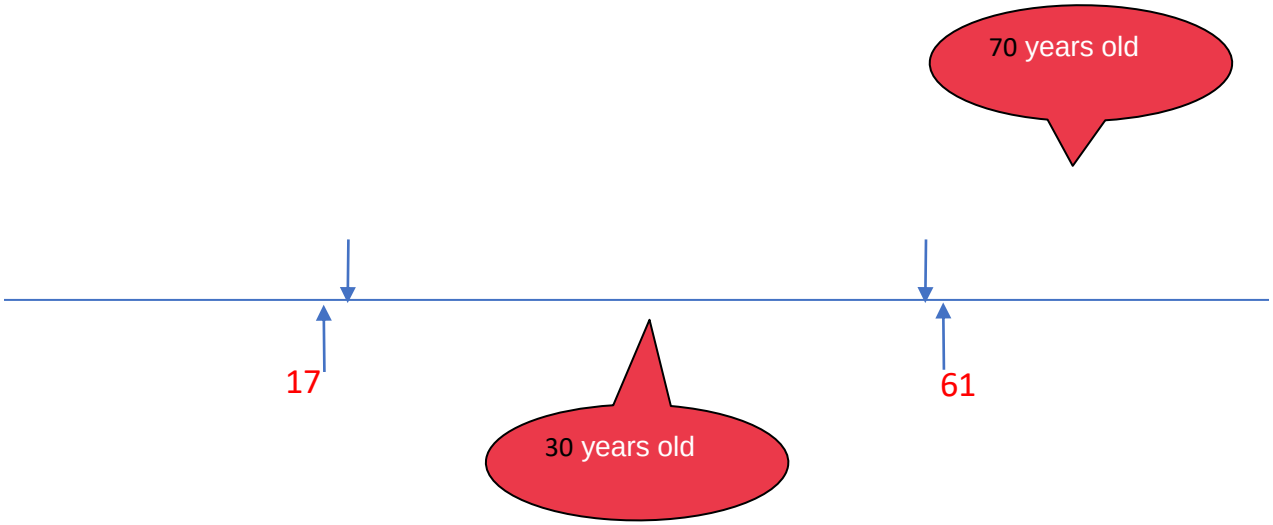
- Employability Module
  - 6 equivalence classes (4 valid classes)
  - For each boundary value, pick up 3 values **on/below/upper**
- 15 Test cases

Age	Employment status
0-15	Don't hire
16-17	Can hire part-time
18-54	Can hire full-time
55-99	Don't hire



Example: Human's age

- Another way to choose values
  - One point on the boundary
  - One point beside the boundary in **invalid class**



Advantage of Boundary Value Analysis

- Easy to understand and control
- Allow to complete the equivalence class method
- Achieve quality test cases

Limitations

- Does not work well with Boolean variables
- Very complex if variables are not independent
- Most suited to systems in which much of the input data takes on values within ranges or within sets

Example: The nextDate function

- Problem statement:
  - nextDate is a function of three variables: month(M), day(D), year(Y). It returns the date of the day after the input date. The month, day and year variables have integer values subject to these conditions:
  - C1: 1 <= M <= 12
  - C2: 1 <= D <= 31
  - C3: 1850 <= Y <= 2050
- How many equivalent classes there are?

Equivalent Classes

Valid ECs	
M1 = {month: 1 <= month <= 12}	M2 = {month: month < 1}
D1={day:1<=day<=31}	M2 = {month: month > 12}
Y1 = {year: 1850 <= year <= 2050}	D2={day:day<1}

Can we define better equivalence classes?

Equivalent Classes

- Which months have 30 days? 31 days?
- February has 28 or 29 days depend on whether year is leap year or not

Valid ECs
M1 = {month has 30 days} M2 = {month has 31 days} M3 = {February}
D1={1<=day<=27}, D2={day=28} D3= {day = 29}, D4= {day = 30}, D5 {day = 31}
Y1 = {year: year is a non-leap year} Y2 = {year: year is a leap year}

Design Test cases

Case ID	Month	Day	Year	Expected Output
C1	-1	15	1902	Value of month not in range
C2				
C3				
C4				
…				

There are too many combinations of M, D and Y ➔ a lot of test case

4.4. Decision Table Technique

Decision Table
Bảng quyết định

- Is an exellent tool to capture certain kinds of system requirements and to document internal system design.
- Often used to record complex business rules that a system must implement
- In addition, they can serve as a guide to creating test cases

General Form

		Rule 1	Rule 2	...	Rule p
Conditions	Condition 1				
	Condition 2				
	...				
	Condition n				
Actions	Action 1				
	Action 2				
	...				
	Action m				

Example: Car Insurance Discount

- Specification
  - If married and number of employed years >=3 then discount 70%
  - If married and number of employed years < 3 then
    - If good student, discount 50%
    - Otherwise, 20%
  - If not married and number of employed years >=3, discount 60%
  - In case of not married and number of employed years < 3
    - If good student, discount 40%
    - Otherwise 0%

Answer

		Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6	Rule 7	Rule 8
C	Married?	Yes	Yes	Yes	Yes	No	No	No	No
	# years employed	>=3	>=3	<3	<3	>=3	>=3	<3	<3
	Good student?	Yes	No	Yes	No	Yes	No	yes	No
A	Discount	70	70	50	20	60	60	40	0

8 test cases

Answer

		Rule 1-2	Rule 3	Rule 4	Rule 5-6	Rule 7	Rule 8
C	Married?	Yes	Yes	Yes	No	No	No
	# years employed	>= 3	<3	<3	>= 3	<3	<3
	Good student?	-	Yes	No	-	Yes	No
A	Discount	70	50	20	60	40	0

6 test cases

Some terms
Một số thuật ngữ

		Rule 1-2	Rule 3	Rule 4	Rule 5-6	Rule 7	Rule 8
C	Married?	Yes	Yes	Yes	No	No	No
	# years employed	>=3	<3	<3	>=3	<3	<3
	Good student?	-	Yes	No	-	Yes	No
A	Discount	70	50	20	60	40	0

Limited entry table – conditions are binary only

Extended entry table – conditions may take on multiple values

Don't-care-entry – condition is irrelevant or not applicable; any value is OK

Decision tables are **declarative** – order between entries does not matter

Using Table Decision for Designing Test Cases Sử dụng bảng quyết định để thiết kế ca kiểm thử

- Rules becomes Test Cases
- Conditions become inputs
- Actions become expected results

		Test Case 1	Test Case 2	...	Test Case p
Inputs	Condition 1				
	Condition 2				
	...				
	Condition n				
Expected Results	Action 1				
	Action 2				
	...				
	Action m				

Exercise 1: Triangle Program

- The program accepts three integers a,b,c as inputs
- These integers are interpreted as representing the lengths of sides of a triangle
- These variables a,b,c must satisfy the following conditions
  - a < b + c, b < a + c, c < a + b
- The program must determine whether the inputs are not a triangle, an isosceles or an equilateral triangle

Decision table

		Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6	Rule 7	Rule 8	Rule 9
C	a,b,c forms a triangle									
	a=b									
	a=c									
	b=c									
A	NaT									
	Isosceles									
	Scalene									
	Equilateral									

Decision Table

		Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6	Rule 7	Rule 8	Rule 9
C	a,b,c forms a triangle	N	Y	Y	Y	Y	Y	Y	Y	Y
	a=b	_	Y	Y	Y	Y	N	N	N	N
	a=c	_	Y	Y	N	N	Y	Y	N	N
	b=c	_	Y	N	Y	N	Y	N	Y	N
A	NaT									
	Isosceles									
	Scalene									
	Equilateral									

Decision Table

		Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6	Rule 7	Rule 8	Rule 9
C	a,b,c forms a triangle	N	Y	Y	Y	Y	Y	Y	Y	Y
	a=b	_	Y	Y	Y	Y	N	N	N	N
	a=c	_	Y	Y	N	N	Y	Y	N	N
	b=c	_	Y	N	Y	N	Y	N	Y	N
A	NaT	X		Impossible	Impossible		Impossible			
	Isosceles					X		X	X	
	Scalene									X
	Equilateral		X							

Design Test Case

		1	2	3	4	5	6	7	8	9	10	11
C	a<b+c											
	b<c+a											
	c<b+a											
	a=b											
	a=c											
	b=c											
A	NaT											
	Isosceles											
	Scalene											
	Equilateral											

a	b	c	Exp

Design Test Case

		1	2	3	4	5	6	7	8	9	10	11
C	a<b+c	N										
	b<c+a	-										
	a<b+a	-										
	a=b	-										
	a=c	-										
	b=c	-										
A	NaT	X										
	Isosceles											
	Scalene											
	Equilateral											

a	b	c	Exp

Design Test Case

		1	2	3	4	5	6	7	8	9	10	11
C	a<b+c	N										
	b<c+a	-										
	c<b+a	-										
	a=b	-										
	a=c	-										
	b=c	-										
A	NaT	X										
	Isosceles											
	Scalene											
	Equilateral											

a	b	c	Exp
4	2	1	NaT

Design Test Case

		1	2	3	4	5	6	7	8	9	10	11
C	a<b+c	N	Y	Y	T							
	b<c+a	-	N	Y	T							
	c<b+a	-	-	N	T							
	a=b	-	-	-	T							
	a=c	-	-	-	T							
	b=c	-	-	-	T							
A	NaT	X										
	Isosceles											
	Scalene											
	Equilateral											

a	b	c	Exp
4	2	1	NaT

Design Test Case

		1	2	3	4	5	6	7	8	9	10	11
C	a<b+c	N	Y	Y	T							
	b<c+a	-	N	Y	T							
	c<b+a	-	-	N	T							
	a=b	-	-	-	T							
	a=c	-	-	-	T							
	b=c	-	-	-	T							
A	NaT	X	X	X								
	Isosceles											
	Scalene											
	Equilateral				X							

a	b	c	Exp
4	2	1	NaT
2	4	1	NaT
1	4	2	NaT
3	3	3	Eq

Design Test Case

		1	2	3	4	5	6	7	8	9	10	11
C	a<b+c	N	Y	Y	T	T	T					
	b<c+a	-	N	Y	T	T	T					
	c<b+a	-	-	N	T	T	T					
	a=b	-	-	-	T	T	T					
	a=c	-	-	-	T	F	T					
	b=c	-	-	-	T	T	F					
A	NaT	X	X	X								
	Isosceles											
	Scalene											
	Equilateral				X							

a	b	c	Exp
4	2	1	NaT
2	4	1	NaT
1	4	2	NaT
3	3	3	Eq

Design Test Case

		1	2	3	4	5	6	7	8	9	10	11
C	a<b+c	N	Y	Y	T	T	T					
	b<c+a	-	N	Y	T	T	T					
	c<b+a	-	-	N	T	T	T					
	a=b	-	-	-	T	T	T					
	a=c	-	-	-	T	F	T					
	b=c	-	-	-	T	T	F					
A	NaT	X	X	X								
	Isosceles											
	Scalene											
	Equilateral				X							

a	b	c	Exp
4	2	1	NaT
2	4	1	NaT
1	4	2	NaT
3	3	3	Eq

Design Test Case

		1	2	3	4	5	6	7	8	9	10	11
C	a<b+c	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
	b<c+a	-	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
	c<b+a	-	-	N	Y	Y	Y	Y	Y	Y	Y	Y
	a=b	-	-	-	Y	Y	Y	Y	N	N	N	N
	a=c	-	-	-	Y	N	Y	N	Y	Y	N	N
	b=c	-	-	-	Y	Y	N	N	Y	N	Y	N
A	NaT	X	X	X								
	Isosceles							X		X	X	
	Scalene											X
	Equilateral				X							

a	b	c	Exp
4	2	1	NaT
2	4	1	NaT
1	4	2	NaT
3	3	3	Eq
5	5	3	Iso
5	3	5	Iso
3	5	5	Iso
3	4	5	Sca

Exercise 2: the nextDate function

Valid ECs
M1 = {month has 30 days} M2 = {month has 31 days} M3 = {February}
D1={1<=day<=27}, D2={day=28} D3= {day = 29}, D4= {day = 30}, D5 {day = 31}
Y1 = {year: year is a non-leap year} Y2 = {year: year is a leap year}

Decision Table for the nextDate function

		1	2	3	4					
C	month in M30									
	month in M31									
	February									
	December									
	1<=day<28									
	Day=28									
	Day=29									
	Day=30									
	Day=31									
	Leap year									
A	Increment Year									
	Increment Month									
	Increment Day									
	Reset year									
	Reset month									
	Reset day									

Decision Table for the nextDate function

		1	2	3	4					
C	month in M30	T	-	-	-					
	month in M31	-	T	-	-					
	February	-	-	T	-					
	December	-	-	-	T					
	1<=day<28	T	T	-	T					
	Day=28	-	-	T	-					
	Day=29	-	-	-	-					
	Day=30	-	-	-	-					
	Day=31	-	-	-	-					
	Leap year	F	F	F	F					
A	Increment Year									
	Increment Month			X						
	Increment Day	X	X		X					
	Reset year									
	Reset month									
	Reset day			X						

Can we simplify the conditions?

Decision Table for the nextDate function

		1	2	3	4					
C	month									
	day									
	year									
A	Increment Year									
	Increment Month									
	Increment Day									
	Reset year									
	Reset month									
	Reset day									
	Error									

M31 = {1,3,5,7,8,10}; M30 = {4,6,9,11}; M2 = {2}; M12 = {12}; D27={1,2,...,27};
D28={28},D29={29},D30={30},D31={31} YN = *not a leap year*; YL = *leap year*

Decision Table for the nextDate function

		1	2	3	4		5	6	7	8	9			
C	month	M30	M30	M30	M31	M31	M12	M12	M2	M2	M2	M2	M2	M2
	day	D27 D28 D29	D30	D31	D27 D28 D29 D30	D31	D27 D28 D29 D30	D31	D30 D31	D27	D28	D28	D29	D29
	year	-	-	-	-	-	-	-	-	-	YL	YN	YL	YN
A	Increment Year							X						
	Increment Month		X			X						X	X	
	Increment Day	X			X		X			X	X			
	Reset year													
	Reset month		X			X		X						
	Reset day							X				X	X	
	Error			X					X					X

Design Test Cases

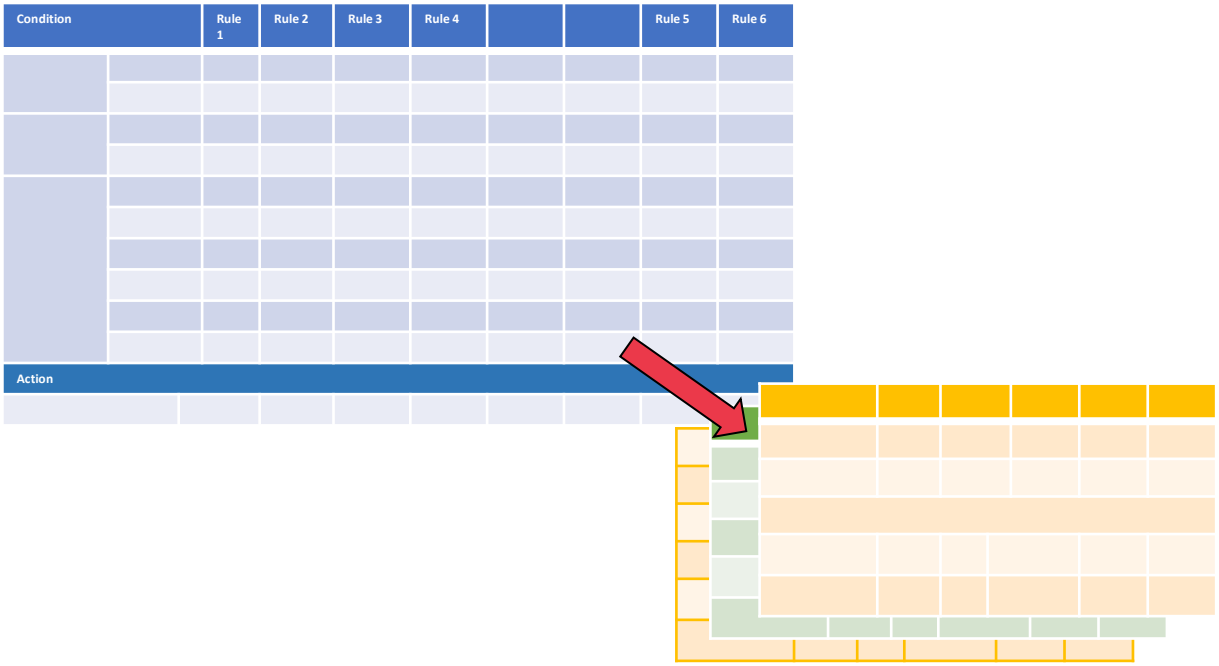
Case ID	Month	Day	Year	Expected Output
1-3	April	15	2001	April 16, 2001
4	April	30	2001	May 1, 2001
5	April	31	2001	Error
6-9	January	15	2001	January 16, 2001
10	January	31	2001	February 1, 2001
11-14	December	15	2001	December 16, 2001
15	December	31	2001	January 1, 2002
16	February	15	2001	February 16, 2001
17	February	28	2004	February 29, 2004
18	February	28	2001	March 1, 2001
19	February	29	2004	March 1, 2004
20	February	29	2001	Error
21, 22	February	30	2001	Error

Advantages

- This technique is useful for applications characterized by any of the following:
  - Prominent if-then-else logic
  - Logical relationships among input variables
  - Calculations involving subsets of the input variables
  - Cause-and-effect relationships between inputs and outputs

Disadvantages

- Decision table may become very large if we have many conditions
- We need to factor large tables into smaller ones to remove redundancy



Exercise 3: Road charge

- A program that calculates road user charges for foreign heavy vehicles
- Inputs:
  - distance that the driver are intending to drive on a road
  - the emission category of the vehicle (0 or 1 or 2)
  - the number of axles of the truck
- Driver can pay for day or week
- Chargeable period for day is 00.00-24.00. For example, if the journey begins at 21h30 on one day and finishes at 02.20 next day, payment for two days is required

Exercise 3: Tarif Tables

Toll Tariff 2014

Max axles	**3**	**3**	**3**	**4**	**4**	**4**
Emission Class	**0**	**1**	**2 or cleaner**	**0**	**1**	**2 or cleaner**
1 day	67	67	67	67	67	67
1 week	220	194	169	347	313	279

Exercise 4: Bus tarif

- A program allows to calculate automatically the monthly bus tarif
- Inputs: age and distance
- Age
  - 0-3 years old: infant charge
  - 4-14 years old: child charge
  - 15-59 years old: adult charge
  - 60-99 years old: silver charge
  - Do not treat at the age of 100 or more
- Distance: <=10 or >10 km

	Distance <=10 km	Distance > 10 km
Infant charge	0	0
Child charge	100	130
Adult charge	200	250
Silver charge	160	200

Exercise 4: Bus tarif (in EUR)

	Distance <=10 km	Distance > 10 km
Infant charge	0	0
Child charge	100	130
Adult charge	200	250
Silver charge	160	200

Exercise 4:

- With input distance, list test cases which using Equivalent and Boundary (if possible)
- With input age, list test cases which using Equivalent and Boundary if possible
- List test cases combination of distance and age by Equivalent method
- List test cases combination of distance and age by Boundary method
- Create decision table base on input distance and age

Exercise 5: Register Form

- A register form allows to create a username and a password and add the account to the database
- Specification:
  - Username: from 3 to 15 characters, no space and special characters including #,$,@
  - Password: should be 8 digits, first character is not zero
- Using Equivalent Class and Boundary Value Analysis to design test cases

4.5. State Transition Testing

Basic ideas

- Applied when an application gives a different output for the same input, depending on what has happened in the earlier state
- The system can be in a (finite) number of different states and the transitions from one state to another are determined by rules
- Have to determine states, events, actions and transitions that should be tested

Basic Terms
Thuật ngữ cơ bản

- The states that the system may occupy
  - ex: a document is closed or opened
- The transitions from one state to another
- The events that cause a transition
  - ex: the action "closing" a document cause the transition form the state "opened" to "closed"
- The actions that result from a transition
  - ex: an error message, a warning …

Design Test Cases

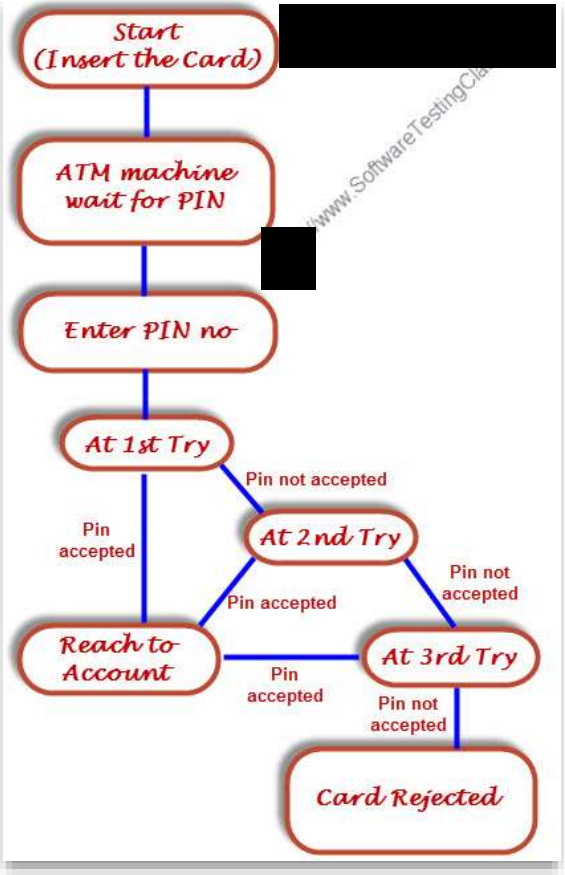
- Create a set of test cases such that **all states** are visited at least once. May miss important transitions.
- Create a set of test cases such that **all states** are triggered at least once. May miss both states and transitions
- Create a set of test cases such that **all transitions** are exercised at least once. Subsumes (includes) all-states and all-events
  - Stronger: cover all possible pairs of transitions (1-switch coverage)
  - Even stronger: cover all possible triplets of transitions (2-switch coverage)
- Create a set of test cases such that **all paths** are executed at least once. Subsumes all others. Can be infeasible - consider loops

Example 1: ATM machine

- Go to the ATM
- Insert card – Verify PIN
- Withdraw money
  - Successful if sufficient balance
  - Refused if insufficient balance
- User can try to type PIN 3 times – If not valid, card is rejected
- How many states the system has?
- Draw the state-transition diagram

State Transition Diagram

- S1: Start State
- S2: Wait for PIN
- S3: 1st try invalid
- S4: 2nd try invalid
- S5: 3rd try invalid
- S6: Card rejected
- S7: Access account



All states coverage

- TC1: Start State – Valid PIN – Access Account
- TC2: Start State – 1st try invalid – 2nd try invalid – 3rd try invalid – Card Rejected

	Insert Card	Valid PIN	InValid PIN
Start State	S2	–	–
Wait for PIN	–	S6	S3
1st try invalid	–	S6	S4
2nd try invalid	–	S6	S5
3rd try invalid	–	–	S7
Access Account	–	?	?
Card not excepted	S1 (for new card)	–	–

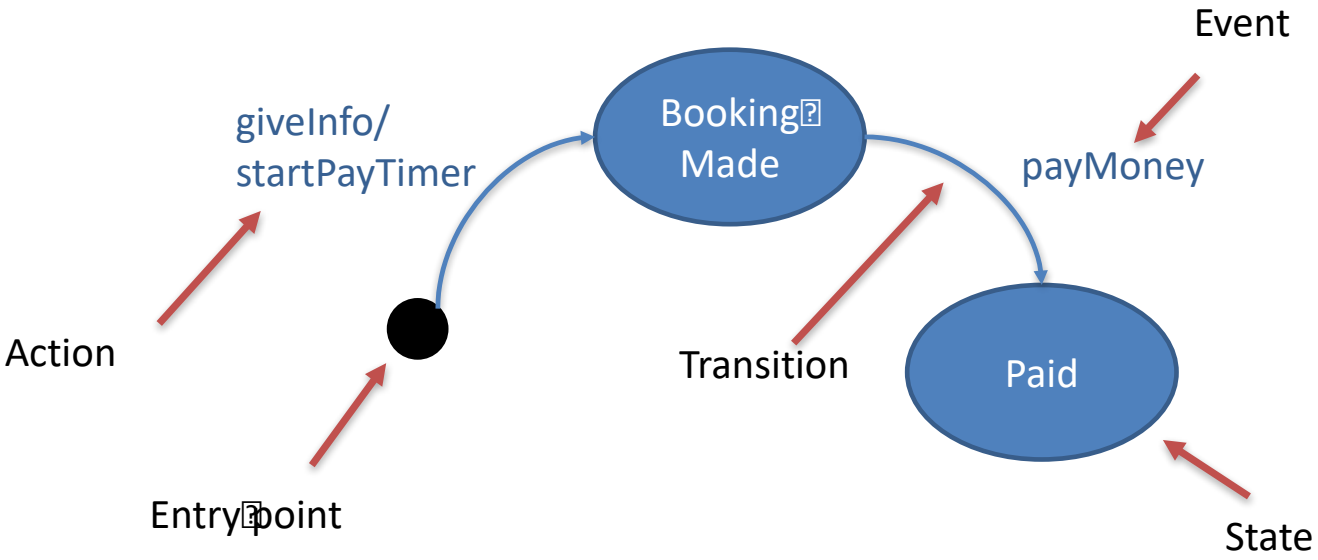
All transitions coverage

- Which test cases will coverage all transitions of the system?

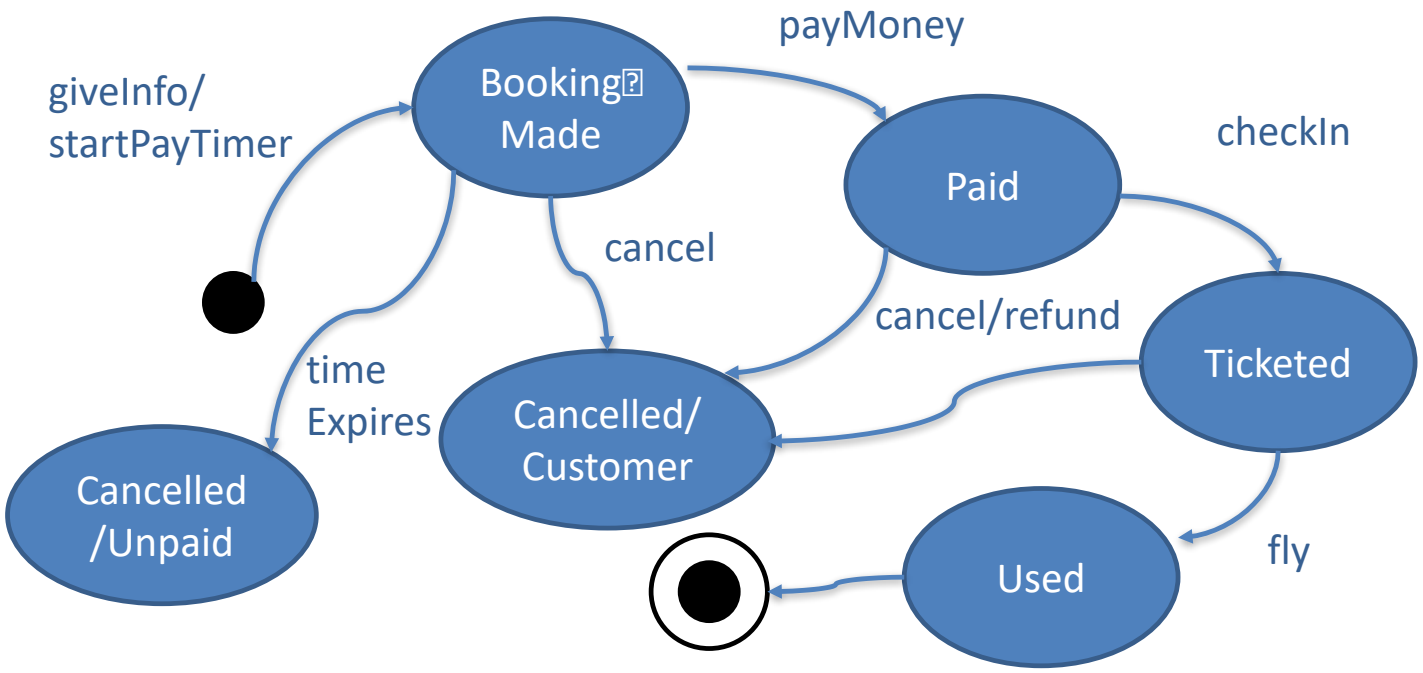
	Insert Card	Valid PIN	InValid PIN
Start State	S2	–	–
Wait for PIN	–	S6	S3
1st try invalid	–	S6	S4
2nd try invalid	–	S6	S5
3rd try invalid	–	–	S7
Access Account	–	?	?
Card not excepted	S1 (for new card)	–	–

Example 2: Flight Booking System

- The customer provides some information and makes a booking.
- He then has a certain amount of time to make the reservation.



Flight Booking System Complete



Design Test Case for Flight Booking System

- Draw a table of states and events
- Identify test cases to satisfy all states coverage
- Identify test cases to satisfy all transitions coverage

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Thank you for your attention!!!