Test Design Techniques

Objective

- **Duration:** *2.5 Hours*
- **Purpose:** *How to design test cases from given software models using the following test design techniques*
- **Audience:** *Testers*

Training Agenda

1. Equivalent Partitioning
2. Boundary Value Analysis
3. Decision Table
4. State Transition
5. Use case

Black Box Test - Test Techniques

- Test techniques for Black Box Test (Specification-Based) can be:
  - Equivalent Partitioning
  - Boundary Value
  - Decision Table
  - State Transition
  - Use case

Equivalent Partitioning

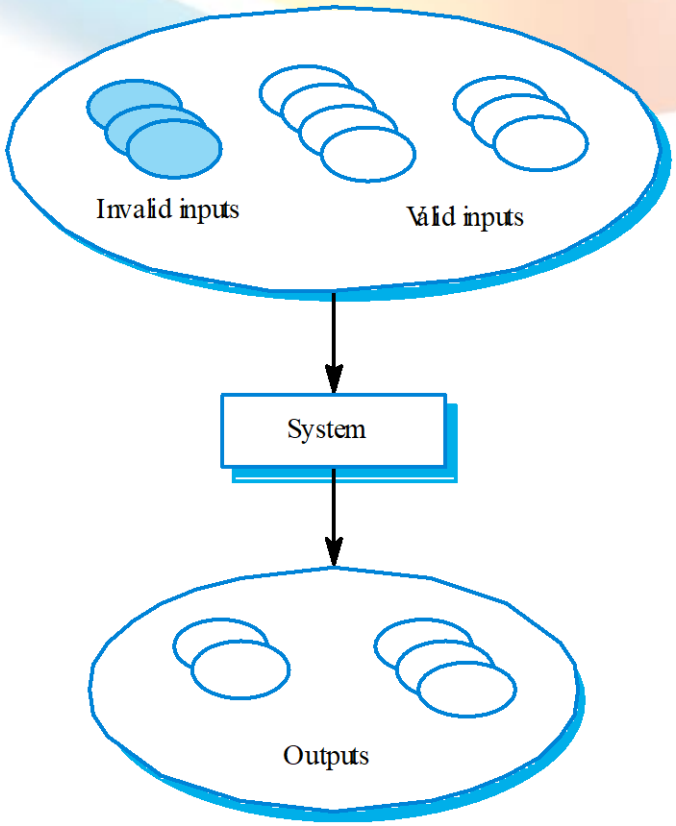
- The idea behind the technique is to divide (i.e. to partition) a set of test conditions into groups or sets that can be considered the same (i.e. the system should handle them equivalently).
- The equivalence-partitioning technique then requires that we need ***test only one condition from each partition.***
- Equivalence partitions (or classes) can be found for both ***valid data and invalid data***.
- Partitions can also be identified for outputs, internal values, time-related values (e.g. before or after an event) and for interface parameters (e.g. during integration testing), ***not only for inputs***.

Equivalent Partitioning
Identify Equivalent Classes

- Take each input condition described in the specification and derive at least two equivalence classes for it.
  - One class that satisfies the condition – the valid class.
  - Second class that doesn't satisfy the condition – the invalid class

Equivalent Partitioning
Identify Equivalent Classes

Equivalent Classes



Equivalent Partitioning Example

- A candidate is given an exam of 40 questions, should get 26 marks to pass (65%), and get more than 80% for get reward.

>> At least, how many test cases to cover all ***valid*** and ***invalid*** cases?

42 cases (-1, 0, 1, … 40, 41)

5 cases (-1, 11, 21, 31, 41)

I think it's easy… but I don't know!

5 cases (-5, 15, 30, 35, 45)

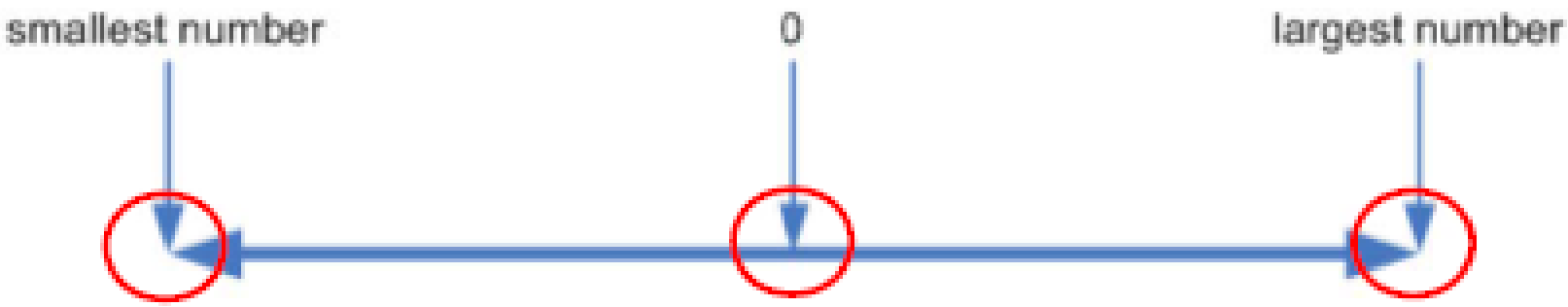
Equivalent Partitioning Example (cont.)

Answer: 5 cases

SHE'S RIGHT

5 cases (-5, 15, 30, 35, 45)

Boundary Value Analysis



Boundary Value Analysis

- Boundary Value Analysis (BVA) is based on testing at the ***boundaries between partitions*** (the maximum and minimum values of partitions).
- This technique is often considered as an ***extension of equivalence partitioning***.
- The maximum and minimum values of a partition are its boundary values
- Have both ***valid*** boundaries (in the valid partitions) and ***invalid*** boundaries (in the invalid partitions).

Boundary Value Analysis

- Boundary value analysis can be applied at all test levels. It is relatively easy to apply and its defect finding capability is high; detailed specifications are helpful
- Boundary values may also be used for test data selection.

Boundary Value Analysis Example

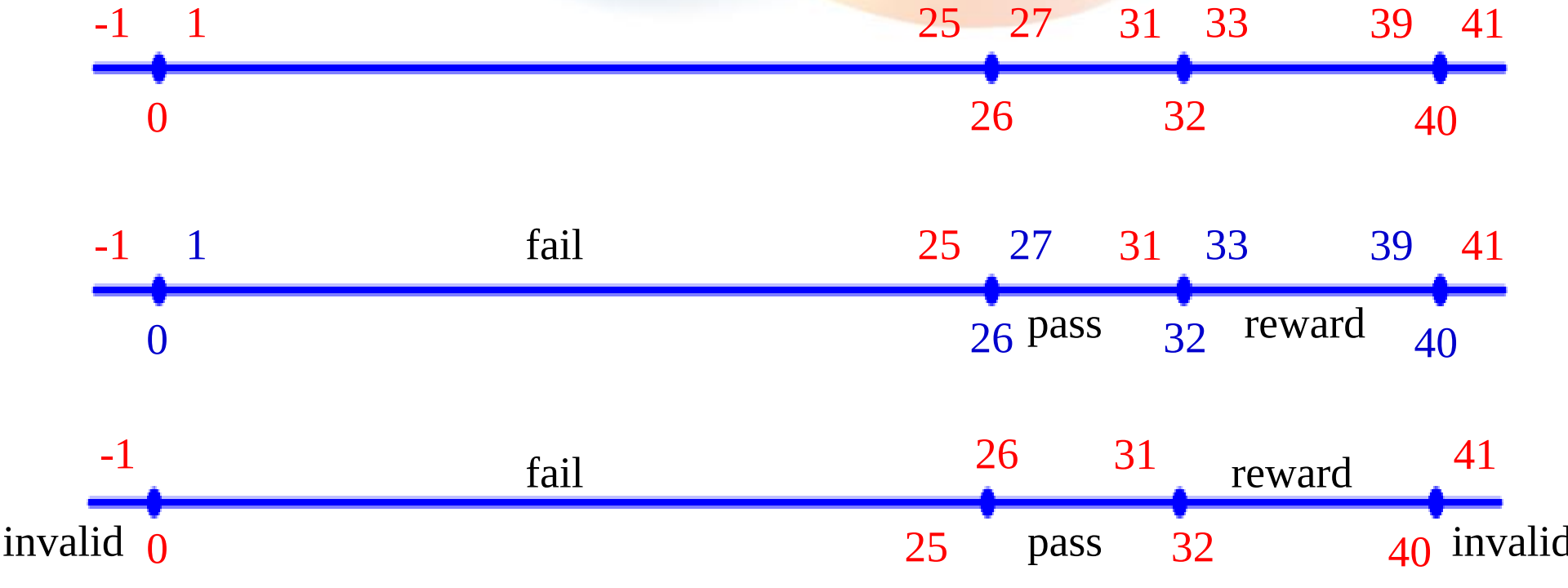
- A candidate is given an exam of 40 questions, should get 26 marks to pass (65%), and get more than 80% for get reward.

>> What are boundary values?

I understand. How about you?

Boundary Value Analysis Example

Boundary Value – 1; Boundary Value; Boundary Value + 1



Boundary Value: -1, 0, 25, 26, 31, 32, 40, 41

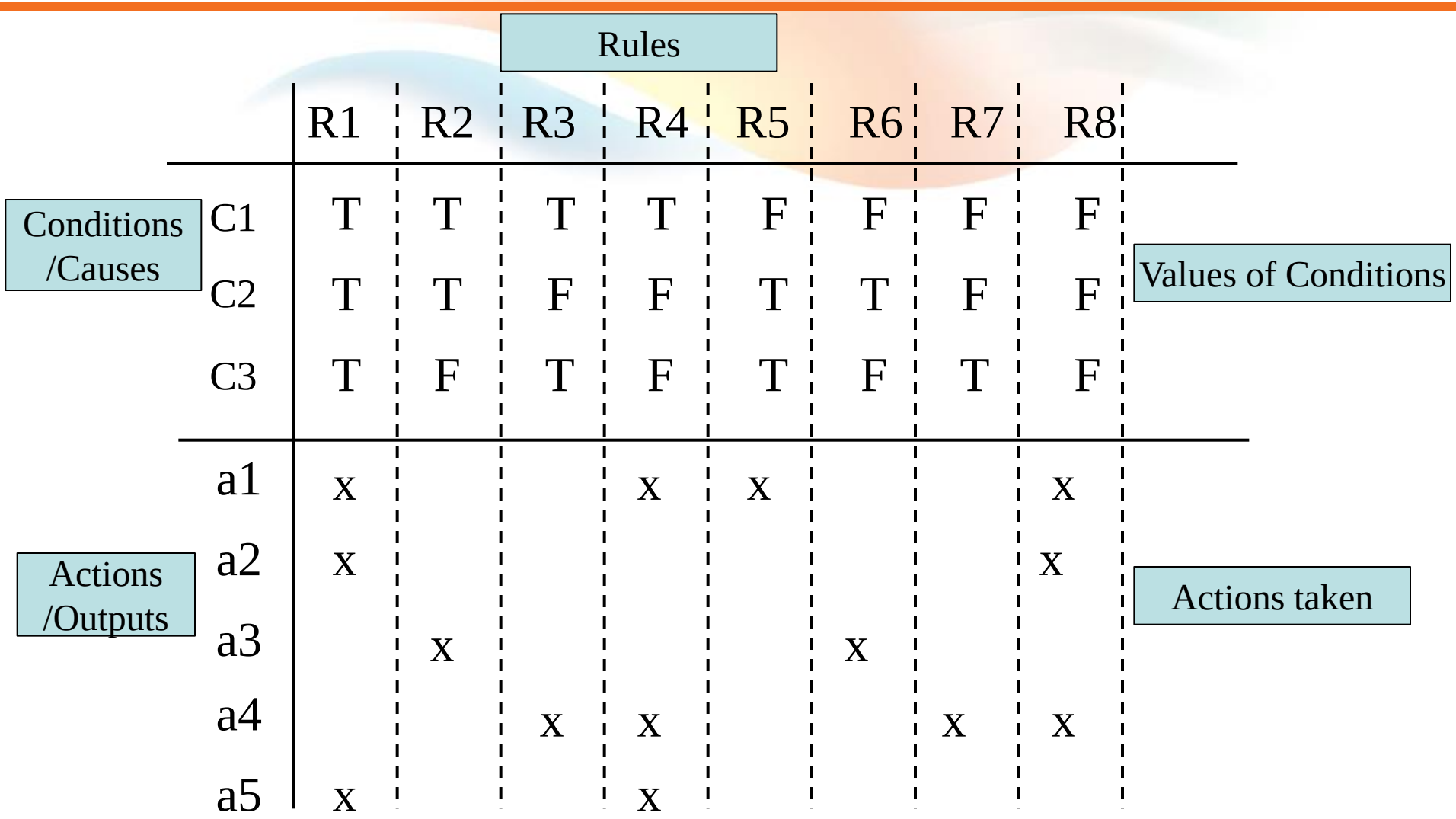
Why Both EP and BVA?

- Reasons:
  - Every boundary is in some partition, if you did only boundary value analysis you would also have tested every equivalence partition.
  - If only testing boundaries we would probably not give the users much confidence as we are using extreme values rather than normal values

Decision Table

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

Decision Table
Components of Decision Table



Decision Table Example

Example:

You want to open a credit card account, there are three conditions. First, if you are a new customer then 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). Discount amounts are added, if applicable. Note: New customer can't hold loyalty card.

=> Decision table:

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 15%			x	x				
Discount 10%					x	x		
Discount 20%			x		x		x	
Total Discount (%)	**-**	**-**	**20**	**15**	**30**	**10**	**20**	**0**

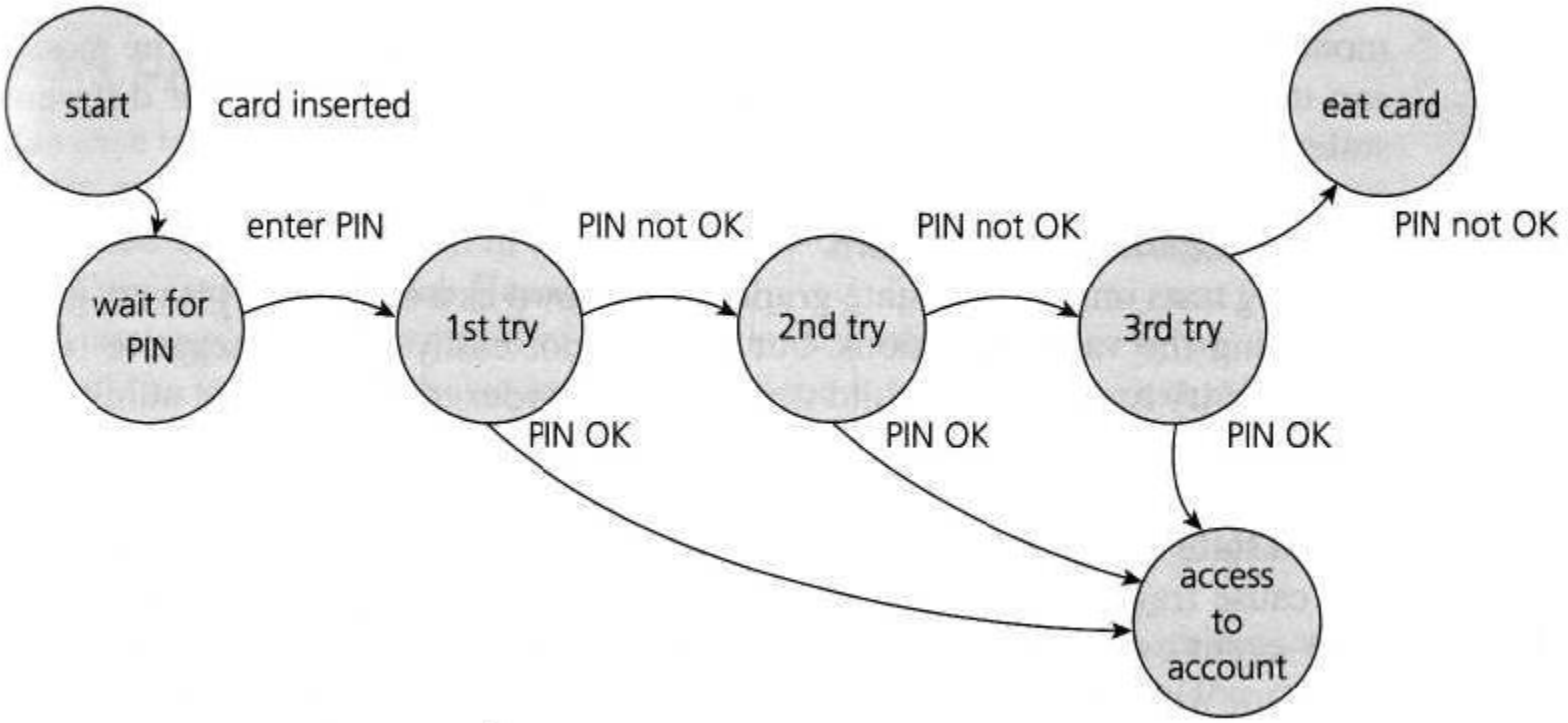
State Transition

- Allows the tester to view the software in terms of its states, transitions between states, the inputs or events that trigger state changes (transitions) and the actions which may result from those transitions.
- A ***state table*** shows the relationship between the states and inputs, and can highlight possible transitions that are ***valid***.
- A ***state diagram*** also can be used to show a finite state system.

State Transition Example

- **Example 1:** bank ATM

  State diagram:



State Transition Example

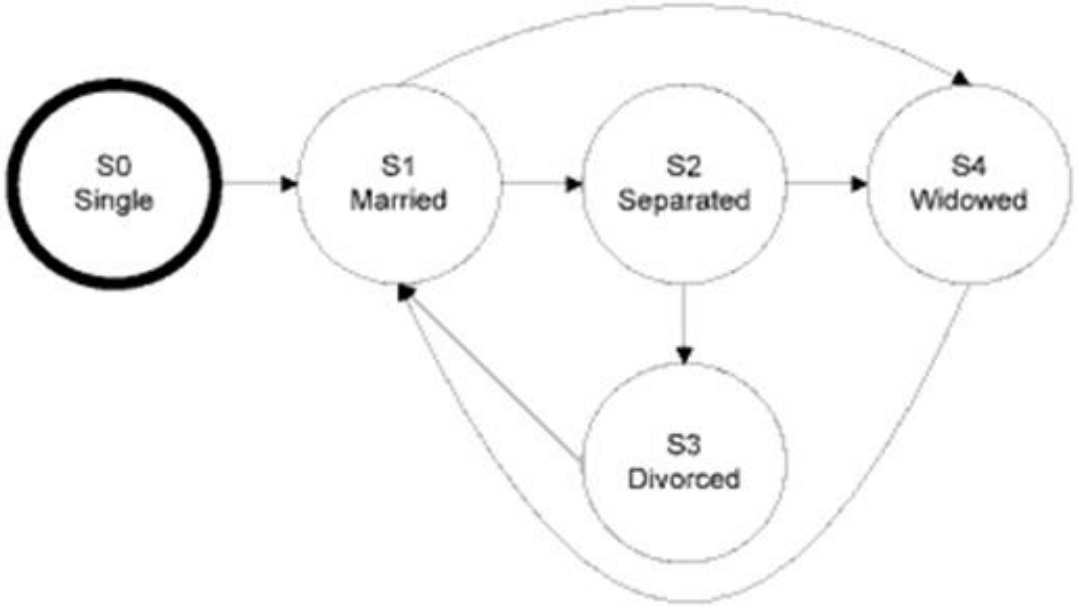
Example 1: bank ATM

- State table:

	Insert card	Valid PIN	Invalid PIN
S1) Start state	S2	–	–
S2) Wait for PIN	–	S6	S3
S3) 1st try invalid	–	S6	S4
S4) 2nd try invalid	–	S6	S5
S5) 3rd try invalid	–	–	S7
S6) Access account	–	?	?
S7) Eat card	S1 (for new card)	–	–

State Transition Example

- **Example 2:** Which test suite will check for an **invalid transition** using the diagram below?



A. S0-S1-S2-S3-S1-S4

B. S0-S1-S4-S1-S2-S3

C. S0-S1-S3-S1-S2-S1

D. S0-S1-S2-S3-S1-S2

Use case

- A use case describes interactions between actors, including users and the system, which produce a result of value to a system user.
- Each use case has preconditions, terminates with post-conditions
- A use case usually has a **mainstream (i.e. most likely) scenario**, and sometimes **alternative branches**.
- Use cases, often referred to as scenarios, are very useful for designing ***acceptance tests*** with customer/user participation.
- They also help uncover ***integration defects*** caused by the interaction and interference of different components, which individual component testing would not see.

Use case - Example

Example: ATM system include functions:

1) Login

- Insert card – input password -> go to Main screen
- User can input wrong password maximum 2 times

2) Withdraw money

- Main screen – select Withdraw - input money – receive money
- User can decide print receipt or not

3) Check Balance

- Main screen – Check Balance
- User can decide print receipt or not

4) Logout

- Any screen - press Exit – cancel process - eject card

Use case - Example

Example: Use case which Tester can consider to write Test case:

Use case	Test case	Pre-condition	User	t	Note
Withdraw money	1	Right card	Insert card	Go to Login	mainstream scenario
			Input right password	Go to Main screen	
			Select Withdraw	Go to Input Money screen	
			Enter amount money (< balance)	Process to dispense money	
				Ask user print receipt or not?	
			Confirm Yes	Eject card	
			User get card	Eject money	
			Get money	Print receipt	
			Get receipt	Go to home page	
	2	Right card	Insert card	Go to Login	alternative branches
			Input right password	Go to Main screen	
			Select Withdraw	Go to Input Money screen	
			Enter amount money (> balance)	Request user re-enter money	
			Enter amount money again (< balance)	Ask user print receipt or not?	
			Confirm No	Eject card	
			User get card	Eject money	
			Get money	Go to home page	
	...		...		
Check balance	1	Right card	Insert card	Go to Login	mainstream scenario
			Input right password	Go to Main screen	
			Select Check Balance	Check & show balance	
				Ask user print receipt or not?	
			Confirm Yes	Print receipt	
				Ask user continue or not?	
			Confirm No	Eject card	
			Get receipt	Go to home page	
	2	Right card	Insert card	Go to Login	alternative branches
			Input wrong password	Request user re-enter password 1st	
			Press exit	Eject card	
			Get card	Go to home page	
	...		...		

QUESTIONS AND ANSWERS

Thank you!!!