SE-3002
SOFTWARE QUALITY ENGINEERING

RUBAB JAFFAR
RUBAB.JAFFAR@NU.EDU.PK

Part II-Software Testing

Functional Testing

Lecture # 19, 20, 21
11, 12, 13 Oct

TODAY'S OUTLINE

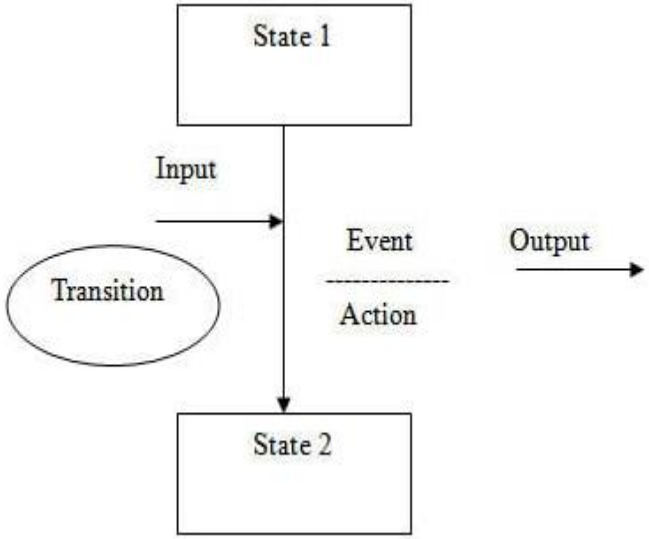
- State Transition Testing
- Domain Analysis Testing
- Use- case Testing

STATE TRANSITION TESTING

- Write test cases from given software models using state transition diagrams/tables.
- Many systems have a number of stable states and transitions from one stable state to another stable state. These machines are called "finite state machines".
- State transition testing is based on the fact that the same action will give different results depending on the initial state for this action.
- The actions that lead from one state to another are called transitions.

STATE TRANSITION TESTING

- This technique is based on the following steps:
- identify the different stable states of the system, including temporary states, including the initial and final states;
- for each state, identify the transactions, events, conditions, and actions that can be executed in this state;
- model the system as a graph (or drawing) or as a table, in order to use it;
- for each combination of state, event, and condition, verify the actions and resulting states.

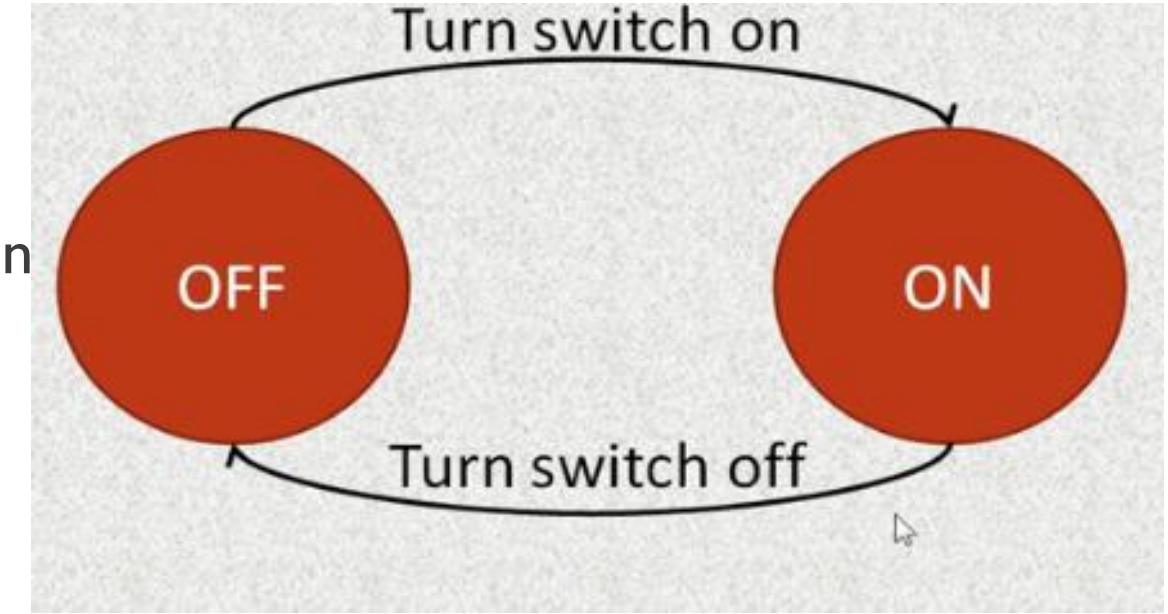


STATE TRANSITION TESTING

- A state is represented as a circle or oval
- A transition from one state to another is represented by an arrow going from the initial state to the resulting state (which can be the same as the initial state), with a description of the action executed,
- The initial state is identified by an arrow to that state that comes from a point outside the graph
- The representation of the final state is obtained by an arrow that leaves that state to reach a circle

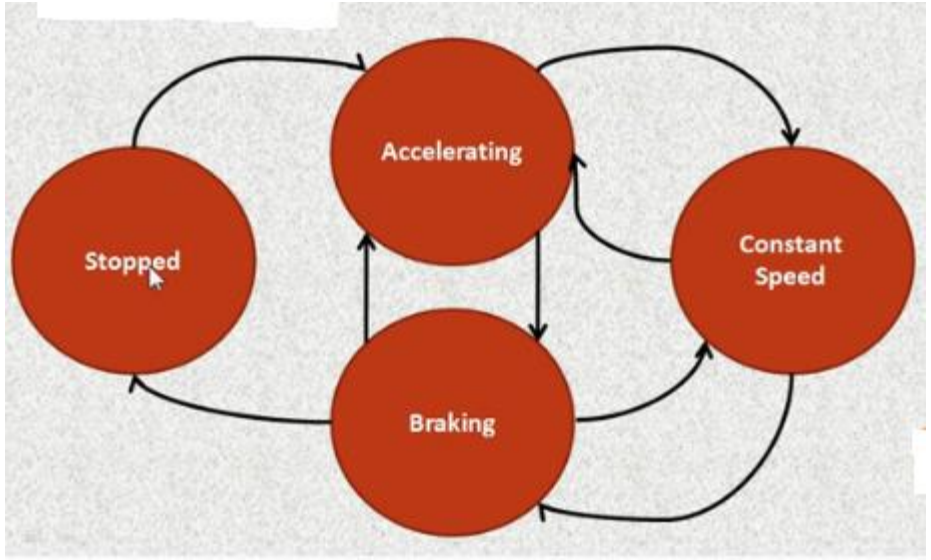
EXAMPLE: LIGHT SWITCH

- States are On> Off> On



EXAMPLE 2: CAR STATES

- States are stopped, accelerating, constant speed and decelerating



EXAMPLE: CAR STATE TABLE

States	Inputs	Next State
Stopped	Press gas pedal more	Accelerating
Accelerating	Keep gas pedal constant	Constant Speed
Accelerating	Switch to brake pedal	Braking
Constant Speed	Press gas pedal more	Accelerating
Constant Speed	Switch to brake pedal	Braking
Braking	Switch to gas pedal; press gas pedal more	Accelerating
Braking	Switch to gas pedal	Constant Speed
Braking	Keep braking	Stopped
Stopped	?	Constant Speed
Stopped	?	Braking
Accelerating	?	Stopped
Constant Speed	?	Stopped

EXAMPLE: CAR TEST CASES

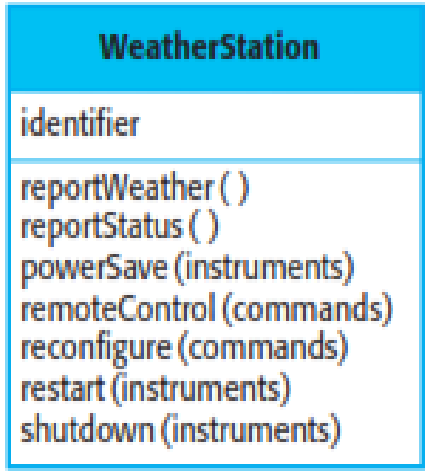
- Test case 1– Stopped > Accelerating > Constant Speed > Braking > Stopped
- Test case 2– Stopped > Accelerating > Braking > Accelerating > Braking > Stopped
- Test case 3 – Stopped > Accelerating > Constant Speed > Accelerating > Constant Speed > Braking > Constant Speed > Braking > Stopped

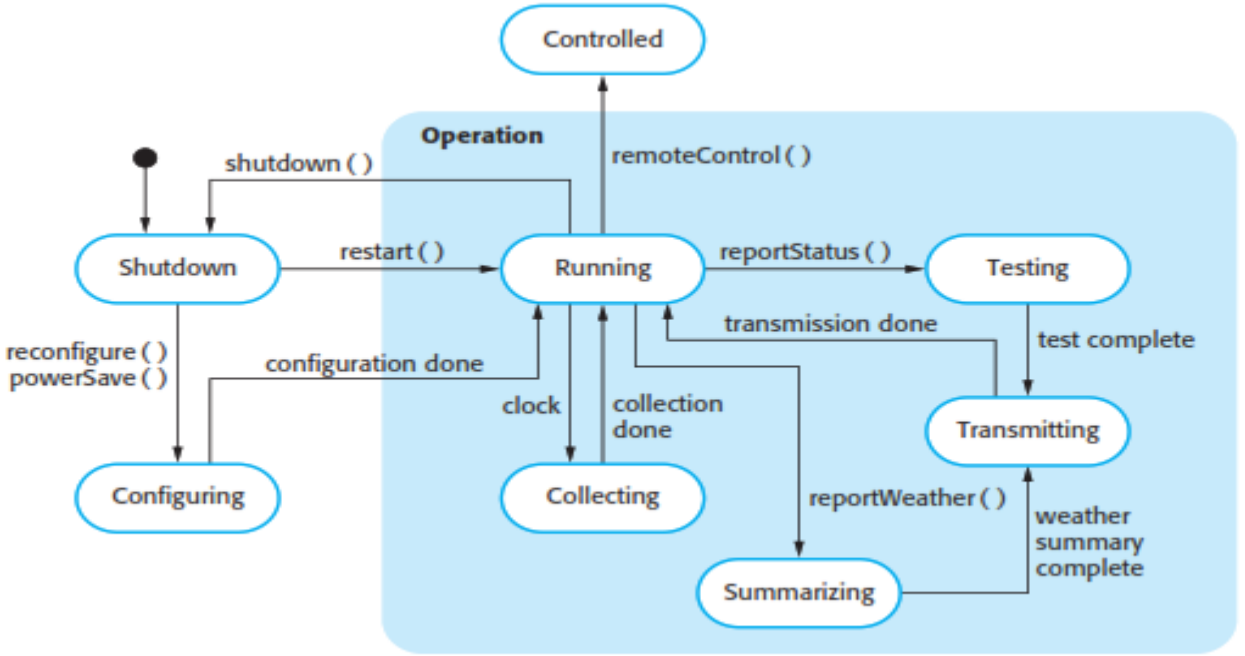
States	Next State
Stopped	Accelerating
Accelerating	Constant Speed
Accelerating	Braking
Constant Speed	Accelerating
Constant Speed	Braking
Braking	Accelerating
Braking	Constant Speed
Braking	Stopped

STATE TRANSITION TESTING

- A graphical representation of states and transitions quickly becomes cluttered and difficult to visualize if the graph has too many states or transitions. It is then necessary to group many different sub-states and their transitions in separate graphs showing the operation of the sub-system, with its inputs and outputs.
- To identify potentially forgotten actions, it is possible to explore the system by trying all possible actions at all the possible states.
- Identification of the valid states and transitions allows identification of invalid or non-authorized transitions from that state.
- It is thus possible to verify that the possible actions from that state do not allow execution of invalid transactions.
- Identification of invalid transitions and actions must be envisaged to ensure that such transitions and states are indeed impossible.
- Analysis of valid and invalid transitions, from all possible states of a system enables the design of test conditions, and sometimes identification of transitions that were not anticipated.

WEATHER STATION STATE DIAGRAM





Shutdown → Running → Shutdown

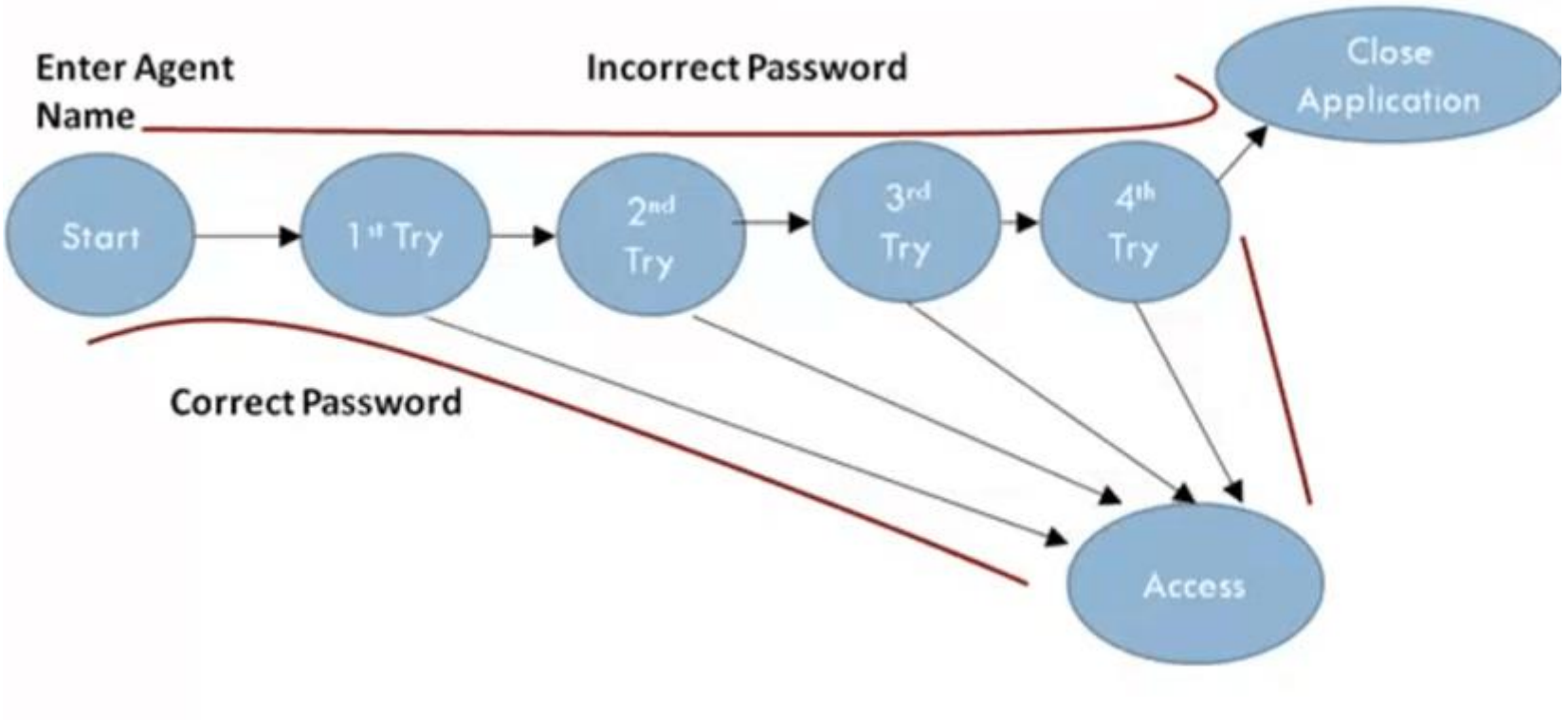
Configuring → Running → Testing → Transmitting → Running

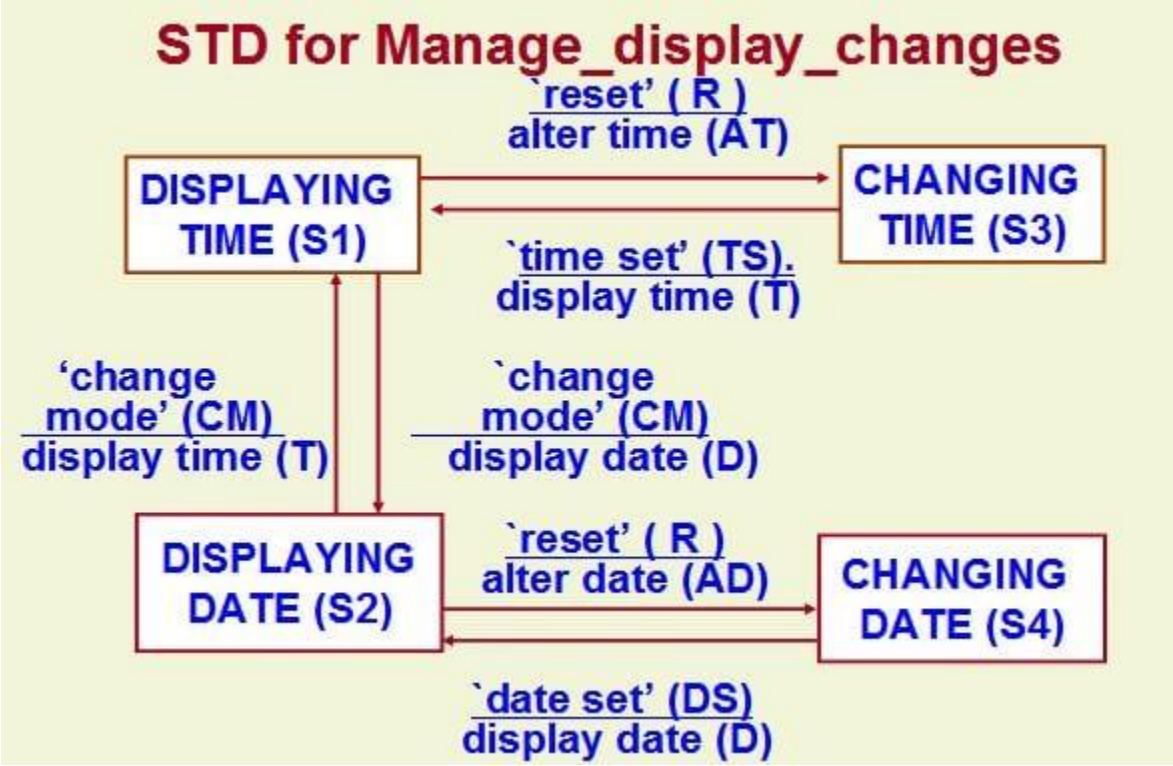
Running → Collecting → Running → Summarizing → Transmitting → Running

LIMITATIONS AND ASSUMPTIONS

- The main assumptions with state transition testing are that we have identified all states and all transitions.
- Any possible "un-identified" state or transition can lead to potential defects being left for the customer to find.

EXAMPLE


Enter Agent Name
Incorrect Password
Close Application
Start
1st Try
2nd Try
3rd Try
4th Try
Correct Password
Access


STD for Manage_display_changes
`reset' (R)
alter time (AT)
DISPLAYING TIME (S1)
CHANGING TIME (S3)
`time set' (TS).
display time (T)
'change mode' (CM)
display time (T)
`change mode' (CM)
display date (D)
DISPLAYING DATE (S2)
`reset' (R)
alter date (AD)
CHANGING DATE (S4)
`date set' (DS)
display date (D)

DOMAIN ANALYSIS TESTING

- A software testing technique in which application is tested by providing input data and verification of relevant output.
- In domain testing, testing takes place with the minimum number of input data so that the application does not allow invalid and out of range data and evaluate the expected range of output.
- Domain testing ensures that an application does not have input data outside the mentioned valid range. This testing process is used to verify the application for its ability to react or operate for different variety of input values.

DOMAIN ANALYSIS TESTING

- In domain testing, we divide a domain into sub-domains (equivalence classes) and then test using values from each subdomain.
- For example, if a website (domain) has been given for testing, we will be dividing the website into small portions (subdomain) for the ease of testing.
- Domain might involve testing of any one input variable or combination of input variables. Practitioners often study the simplest cases of domain testing less than two other names, "boundary testing" and "equivalence class analysis."
- Boundary Testing– Boundary value analysis (BVA) is based on testing at the boundaries between partitions. We will be testing both the valid and invalid input values in the partition/classes.
- Equivalence Class testing– The idea behind this 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), hence 'equivalence partitioning.'

STRATEGY OF DOMAIN TESTING

1. Domain Selection

- Input variables that need to be assigned, and the proper result has to be verified.

2. Group the Input Data into Classes

- A similar type of input data is partitioned into subsets. There are two types of partitioning, Equivalence class partitioning & Boundary value analysis (BVA).

3. Input Data of the Classes for Testing

- The boundary values should be considered as the data for testing. Boundaries represent the equivalence classes more likely to find an error than the other class members. A data in between the range is the best representative of an equivalence class.

4. Verification of Output Data

- When input data is assigned to application concerning that output data verified. Output data should be invalid and specified range.

EXAMPLES OF DOMAIN TESTING

- Consider a games exhibition for Children, 6 competitions are laid out, and tickets have to be given according to the age and gender input. The ticketing is one of the modules to be tested in for the whole functionality of Games exhibition.
- According to the scenario, we got six scenarios based on the age and the competitions:
- Age >5 and <10, Boy should participate in Storytelling.
- Age >5 and <10 , girl should participate in Drawing Competition.
- Age >10 and <15, Boy should participate in Quiz.
- Age >10 and <15 , girl should participate in Essay writing.
- Age<5, both boys and girls should participate in Rhymes Competition.
- Age >15, both boys and girls should participate in Poetry competition.
- The input will be Age and Gender and hence the ticket for the competition will be issued.

EQUIVALENCE CLASSES

- For the above example, we are partitioning the values into a subset or the subset. We are partitioning the age into the below classes :
- **Class 1:** Children with age group 5 to 10
- **Class 2 :** Children with age group less than 5
- **Class 3:** Children with age group age 10 to 15
- **Class 4:** Children with age group greater than 15.

BVA FOR EQUIVALENCE CLASSES

- For example for the scenario #1:
- **Class 1:** Children with age group 5 to 10 (Age >5 and <=10)
- **Boundary values:**
- Values should be Equal to or lesser than 10. Hence, age 10 should be included in this class.
- Values should be greater than 5. Hence, age 5 should not be included in this class.
- Values should be Equal to or lesser than 10. Hence, age 11 should not be included in this class.
- Values should be greater than 5. Hence, age 6 should be included in this class.

DOMAIN TESTING STRUCTURE

- Identify the potentially interesting variables.
- Create and identify boundary values and equivalence class values as above.
- Identify secondary dimensions and analyze each in a classical way. (In the above example, Gender is the secondary dimension).
- Identify and test variables that hold results (output variables).
- Evaluate how the program uses the value of this variable.
- Identify additional potentially-related variables for combination testing.
- Summarize your analysis with a risk/equivalence table.

TESTCASES

Scenario	Boundary values to be taken	Equivalence partitioning values
Boy – Age >5 and <=10	Input age = 6 Input age = 5 Input age = 11 Input age = 10	Input age = 8
Girl – Age >5 and <=10	Input age = 6 Input age = 5 Input age = 11 Input age = 10	Input age = 8
Boy – Age >10 and <=15	Input age = 11 Input age = 10 Input age = 15 Input age = 16	Input age = 13
Girl – Age >10 and <=15	Input age = 11 Input age = 10 Input age = 15 Input age = 16	Input age = 13
Age<=5	Input age = 4 Input age = 5	Input age = 3
Age >15	Input age = 15 Input age = 16	Input age = 25

USE CASES TESTING

- Use Case Testing is a functional black box testing technique that helps testers to identify test scenarios that exercise the whole system on each transaction basis from start to finish.
- Use cases are a representation of the expected behavior of a system as it should respond to the requests of users external to the system.
- The level of modeling detail can vary for use cases, which usually involves transactions from initiation until termination, and includes other factors besides the system, such as sub-systems or other software applications.
- The sum of all the use cases is used to describe the way the system or software works.
- This is the case in agile development methods when specifications are provided in the form of use cases or user stories.

CASH WITHDRAWAL – USE CASE SPECIFICATION

- **1 Brief Description**
- This use case describes how the Bank Customer uses the ATM to withdraw money to his/her bank account.
- **2 Actors**
- **2.1 Bank Customer**
- **2.2 Bank**
- **3 Preconditions**
- There is an active network connection to the Bank.
- The ATM has cash available.

4 BASIC FLOW OF EVENTS

1. The use case begins when Bank Customer inserts their Bank Card.
2. Use Case:Validate User is performed.
3. The ATM displays the different alternatives that are available on this unit. In this case the Bank Customer always selects "Withdraw Cash".
4. The ATM prompts for an account.
5. The Bank Customer selects an account.
6. The ATM prompts for an amount.
7. The Bank Customer enters an amount.
8. Card ID, PIN, amount and account is sent to Bank as a transaction. The Bank Consortium replies with a go/no go reply telling if the transaction is ok.
9. Then money is dispensed.
10. The Bank Card is returned.
11. The receipt is printed.
12. The use case ends successfully.

5 ALTERNATIVE FLOWS

- 2.1 If in step 2 of the basic flow Bank Customer the use case:Validate User does not complete this successfully, then the use case ends with return card.
- 7.1 If in step 7 in the basic flow, the Bank Customer enters an amount that exceeds the withdrawal limit, then the ATM shall display a warning message, and ask the Bank Customer to reenter the amount.
- 7.2 If in step 7 in the basic flow, the Bank Customer enters an amount that exceeds the account balance, then the ATM shall display a warning message that the amount entered is more than the account balance and return the card.
- 7.3 The use case resumes at step 7
- 9.1 If in step 9 of the basic flow the money is not removed from the machine within 15 seconds, then the ATM shall issue a warning sound and display the message "Please remove cash".
- 9.2 If there is still no response from the Bank Customer within 15 seconds the ATM will re-tract the money and note the failure in the log.
- 9.3 the use case end with a failure condition.

6 POST-CONDITIONS

- The user has received their cash and the internal logs have been updated.

TESTCASES

	A	B	C
1	**TEST CASE(S)**	**Steps**	**Expected Results**
2	*Test Case:*	Cash withdrawal1 - Normal workflow	
3	*1)*	Insert debit card.	The System will prompt for the PIN.
4	2)	Enter the valid PIN.	The System will display the option to "Withdraw Cash".
5	3)	Select the option to "Withdraw Cash".	The System will prompt for an amount.
6	4)	Enter amount as $100 (valid amount).	The System will dispense the cash amount. The System will return the card.

TESTCASES

Test Case:	Cash withdrawal2 - Invalid PIN	
1)	Repeat Step 1) of Cash withdrawal1	
2)	Enter an invalid PIN.	The System will return the card.

TESTCASES

Test Case:	Cash withdrawal3 - Invalid amount	
1)	Repeat Step 1 thru 4) of Cash withdrawal1 with $101 (invalid amount).	The System will display an error message and prompt for another amount.
2)	Repeat Step 4) of Cash withdrawal1 with $0 (invalid amount).	The System will display an error message and prompt for another amount.
3)	Repeat Step 4) of Cash withdrawal1 with $1 (valid amount).	The System will dispense the cash amount. The System will return the card.

Test Case:	Cash withdrawal4 - Transaction declined	
1)	Repeat Step 1 thru 4) of Cash withdrawal1 with $amount > account balance (invalid amount).	The System will return the card.

EXAMPLE

- Consider a scenario where a user is buying an Item from an Online Shopping Site. The user will First Login to the system and start performing a Search. The user will select one or more items shown in the search results and he will add them to the cart.
- After all this, he will check out.

That is all