Test Design Techniques

March 2016, 2018

softserve

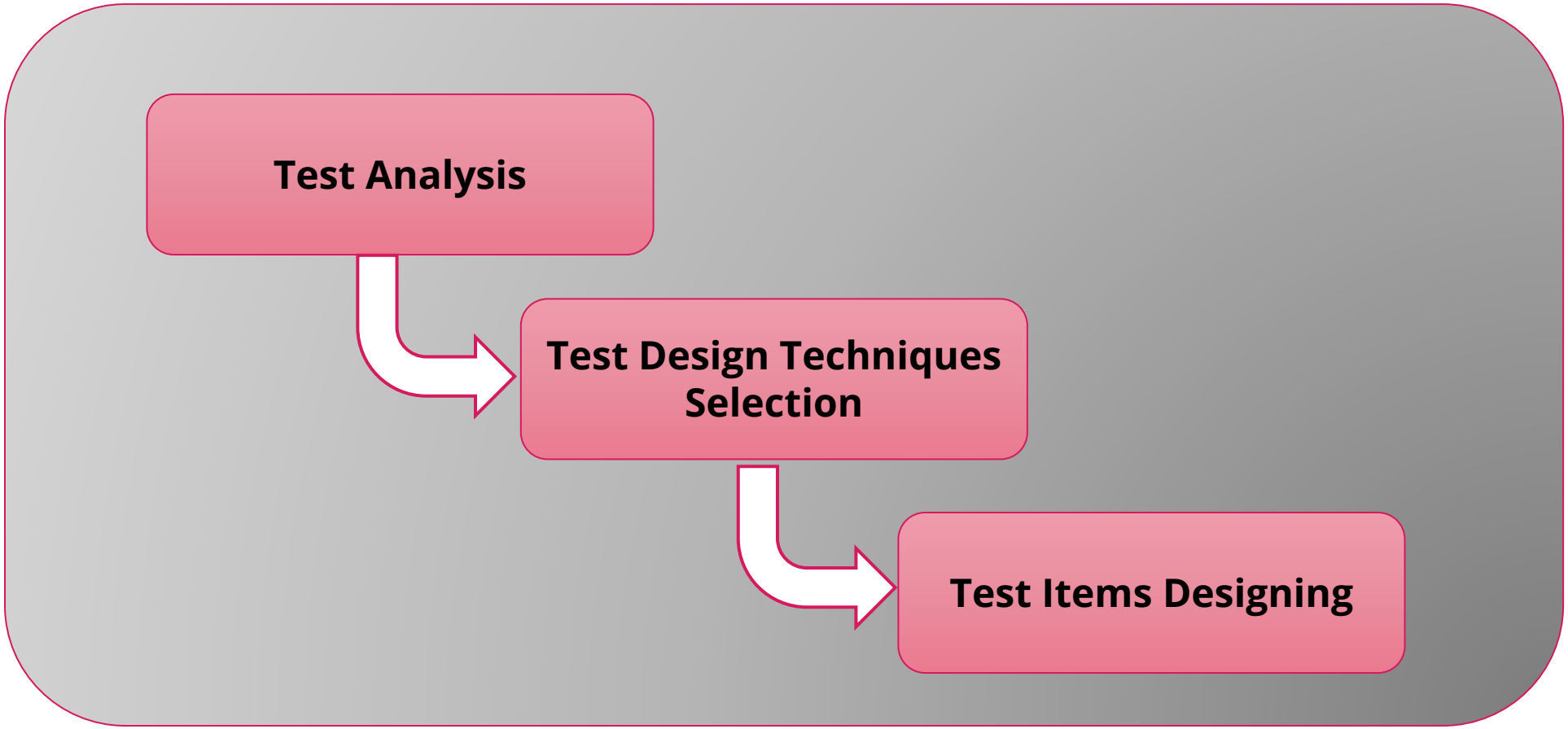
Agenda

- *Static and Dynamic Testing*
- *Test Techniques*
  - ✔ *Black-box*
  - ✔ *White-box*
  - ✔ *Experience-based*
- *Choosing a Test Design Technique*

Static and Dynamic Testing

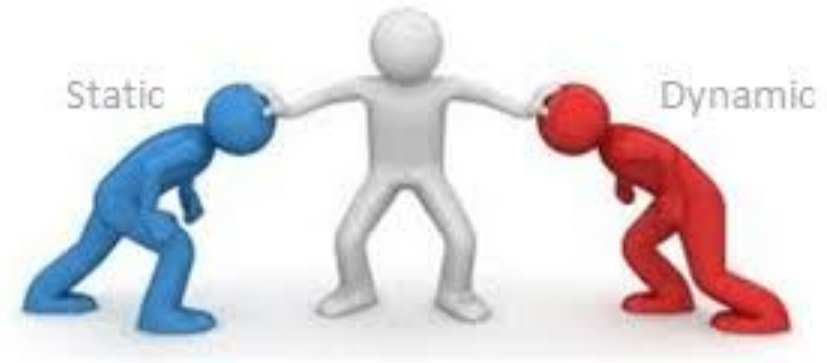
softserve

Trainings' Content



softserve

Static and Dynamic Testing

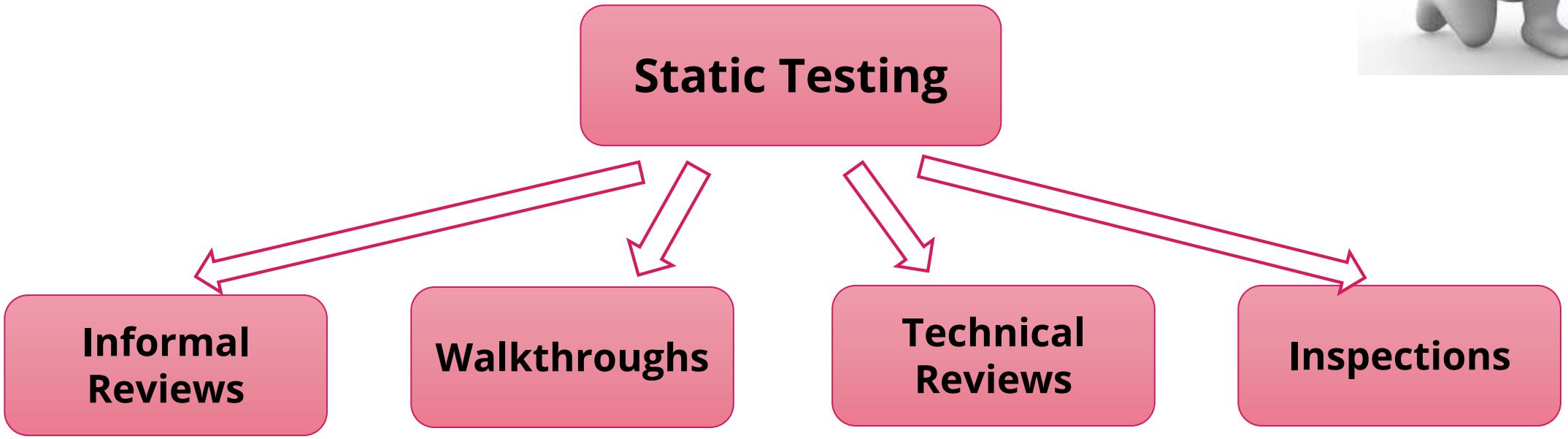


Categories

Static: Static testing test software without executing it

Dynamic: Testing that involves the execution of the software of a component or system

Static Testing



Benefits of Static Testing

Benefits of static testing may include:

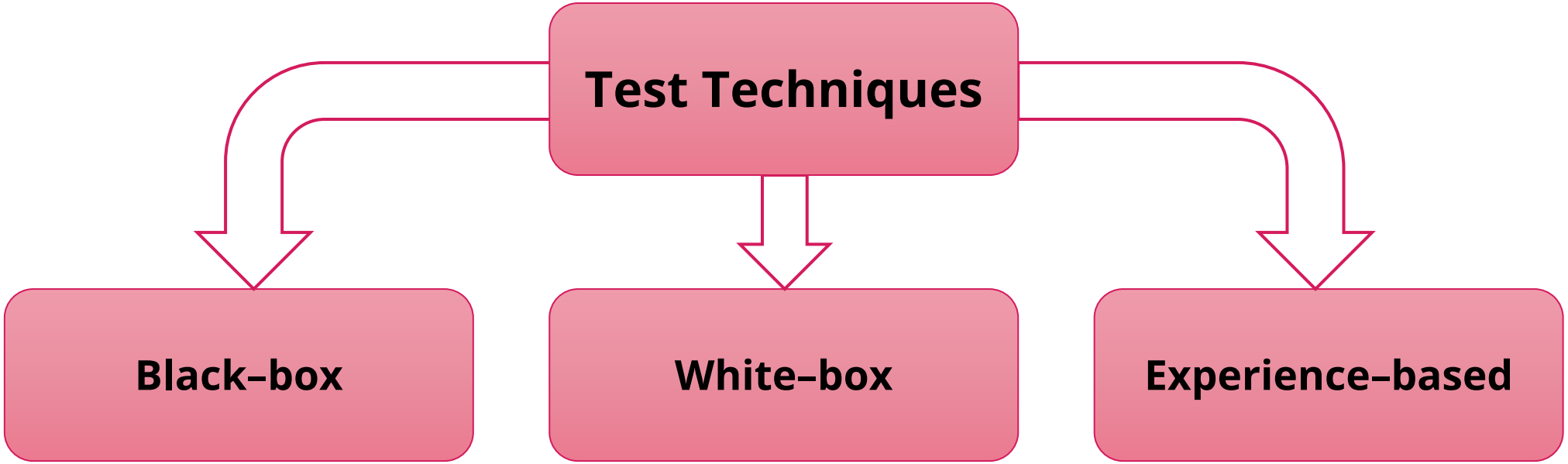
- Detecting and correcting defects more efficiently, and prior to dynamic test execution
- Identifying defects which are not easily found by dynamic testing
- Preventing defects in design or coding by uncovering inconsistencies, ambiguities, contradictions, omissions, inaccuracies, and redundancies in requirements
- Increasing development productivity
- Reducing cost and time
- Improving communication between team members in the course of participating in reviews

Test Techniques

softserve

Test Techniques

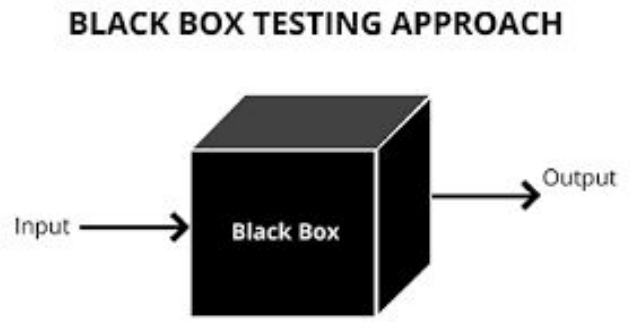
The purpose of a test technique, including those discussed in this section, is to help in identifying test conditions, test cases, and test data.



Black–box Test Techniques

softserve

Black-box Test Techniques

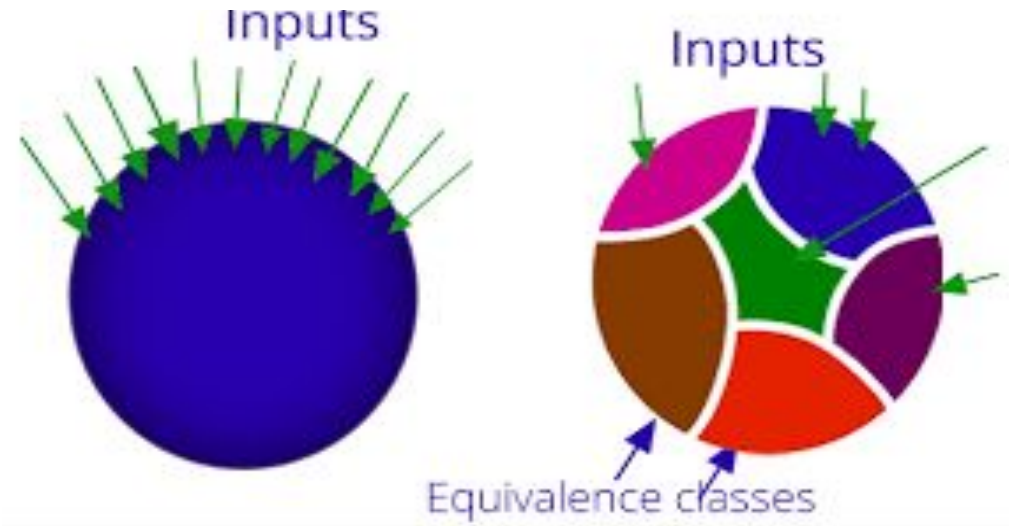


Black-box

- Equivalence Partitioning
- Boundary Values Analysis
- State Transition
- Decision Tables
- Use Case Testing

Equivalence Partitioning

- **Equivalence partitioning (EP)** – A black-box test design technique in which test cases are designed to execute representatives from equivalence partitions.
- **Idea:** 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. In principle test cases are designed to cover each partition at least once.



Equivalence Partitioning. Example

- **Example:** Bank represents new deposit program for corporate clients. According to the program client has ability to get different %, based on amount of deposited money. Minimum which can be deposited in $1, maximum is – $999. If client deposits less than $500 it will have 5% of interests. In case the amount of deposited money is $500 and higher, then client gets on 10% of interests more.

	Invalid	Valid for 5% discount	Valid for 15% discount	Invalid
Class	$0	$1 - $499	$500 - $999	>=$1000
EP	0	240	800	3500

Invalid class: please don't forget about negative values and special symbols here (e.g. "-100$", "100₴", "100#", "#$%^&", etc.)

softserve

Equivalence Partitioning: Test Conditions

	Invalid	Valid for 5% discount	Valid for 15% discount	Invalid
Class	$0	$1 - $499	$500 - $999	>=$1000
EP	0	240	800	3500

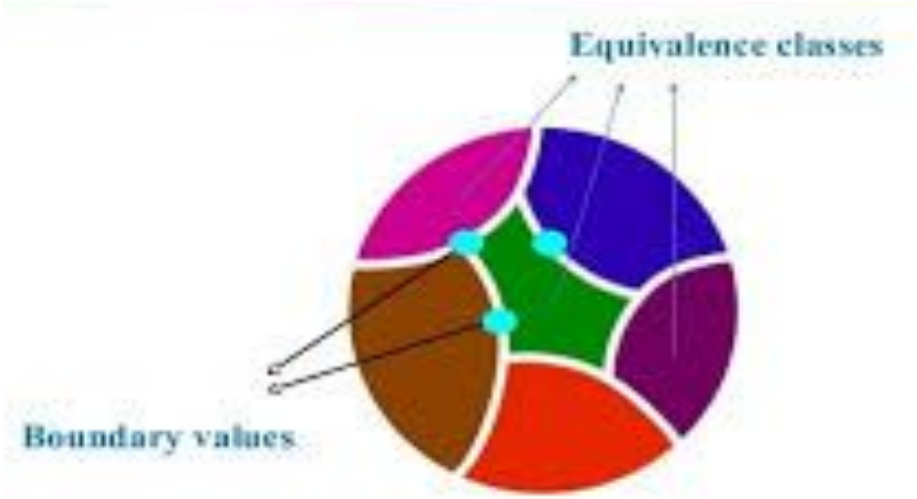
Invalid class: negative values and special

#	Condition	Expected Result
1	Put value `0$` into input field	Error message appears "You have entered an incorrect value for the amount of deposited money".
2	Put value `240$` into input field	Deposit is opened with 5% of interests.
3	Put value `800$` into input field	Deposit is opened with 15% of interests.
4	Put value `3500$` into input field	Error message appears "You have entered an incorrect value for the amount of deposited money".
5	Put value `-+=#$%^` into input field	Error message appears "You have entered an incorrect value for the amount of deposited money".

Boundary Values Analysis

- **Boundary value analysis (BVA):** A black box test design technique in which test cases are designed based on boundary values.

  BVA is an extension of equivalence partitioning, but can only be used when the partition is ordered, consisting of numeric or sequential data. The minimum and maximum values (or first and last values) of a partition are its boundary values
- **Idea:** Divide test conditions into sets and test the boundaries between these sets. Tests should be written to cover each boundary value.



BVA. Example

	Invalid	Valid for 5% discount		Valid for 15% discount		Invalid
Class	$0	$1 - $499		$500 - $999		>=$1000
BVA	0	1	499	500	999	1000

#	Condition	Expected Result
1	Put value `0$` into input field	Error message appears "You have entered an incorrect value for the amount of deposited money".
2	Put value `1$` into input field	Deposit is opened with 5% of interests.
3	Put value `499$` into input field	Deposit is opened with 5% of interests.
4	Put value `500$` into input field	Deposit is opened with 15% of interests.
5	Put value `999$` into input field	Deposit is opened with 15% of interests.
6	Put value `1000$` into input field	Error message appears "You have entered an incorrect value for the amount of deposited money".

Example: Bank represents new deposit program for corporate clients. According to the program client has ability to get different %, based on amount of deposited money. Minimum which can be deposited in $1, maximum is – $999. If client deposits less than $500 it will have 5% of interests. In case the amount of deposited money is $500 and higher, then client gets on 10% of interests more.

softserve

EP and BVA: Test Items

#	Test Items	Test data
1	Verify that deposit is opened with 5% of interests if client deposits less than $500	1. Any number from 1 to 499 (e.g. 240) 2. 1 3. 499
2	Verify that deposit is opened with 15% of interests if client deposits from $500 to $999	1. Any number from 500 to 999 (e.g. 723) 2. 500 3. 999
3	Verify that error message "You have entered an incorrect value for the amount of deposited money“ appears if client enters incorrect data.	1. Any number < 0 (e.g. -5) 2. 0 3. Any number > 999 (e.g. 1248) 4. 1000 5. Decimal number (e.g. 156.34) 6. Decimal number (e.g. 34,98) 7. Alphabetic characters 8. Special characters
4	Verify that error message appears if client leaves ‘Deposit’ field empty.	

Decision Table Testing

- **Decision table** – A table showing combinations of inputs and/or stimuli (causes) with their associated outputs and/or actions (effects), which can be used to design test cases.

- **Idea:** Divide test conditions into constraints, which could get positive or negative meanings, and rules which identify output based on values of conditions. While analyzing each possible variant of positive and negative meanings identify output or set of outputs for each variant based on the rules. Only combinations of these positive and negative meanings, which uniquely identify decisions that are made, should be covered by tests.

Decision Table. Example

Example: If you hold an 'over 60s' rail card, you get a 34% discount on whatever ticket you buy. If you hold family rail card and you are traveling with a child (under 16), you can get a 50% discount on any ticket. If you are traveling with a child (under 16), but do not have family rail card, you can get a 10% discount. You can use only one type of rail card.

- 'over 60s' rail card – 34%
- family rail card and traveling with a child – 50%
- traveling with a child, but do not have family rail card – 10%
- only one type of rail card can be used

softserve

Decision Table. Example

Causes (inputs)	R1	R2	R3	R4	R5	R6	R7	R8
Over 60s rail card?	Y	Y	Y	Y	N	N	N	N
Family rail card?	Y	Y	N	N	Y	Y	N	N
Child also traveling?	Y	N	Y	N	Y	N	Y	N
Effects (Outputs)								
Discount (%)	50	34	34	34	50	0	10	0
Message*	+	+						

- 'over 60s' rail card – 34%
- family rail card and traveling with a child – 50%
- traveling with a child, but do not have family rail card – 10%
- only one type of rail card can be used

The rationalized table with a fewer columns and thus will result in fewer test cases:

softserve

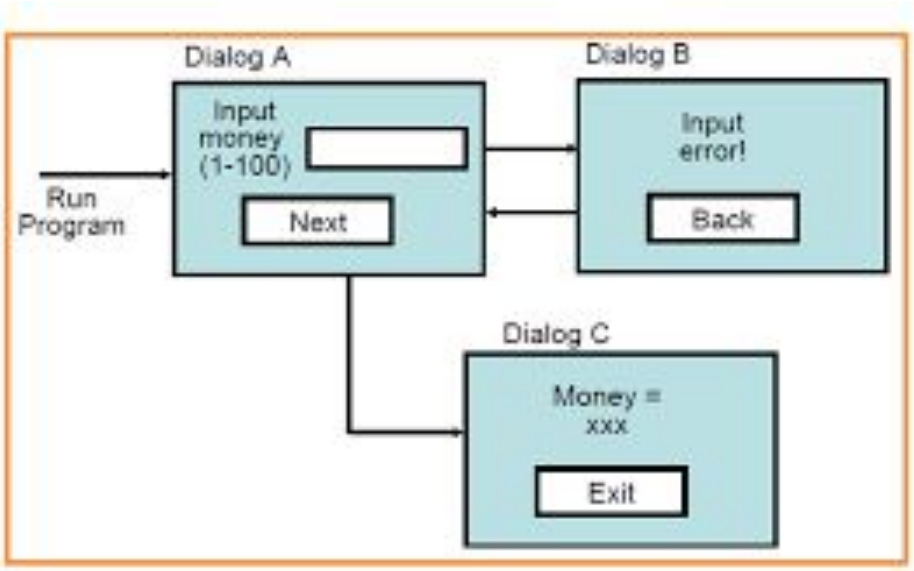
Decision Table Example

Causes (inputs)	R1	R2	R3, R4	R5	R6, R8	R7
Over 60s rail card?	Y	Y	Y	N	N	N
Family rail card?	Y	Y	N	Y	**Y/N**	N
Child also traveling?	Y	N	**Y/N**	Y	N	Y
Effects (Outputs)						
Discount (%)	50	34	34	50	0	10
Errors	+	+				

#	Condition	Outcome
1	A person who has over 60s rail & family rail card and also traveling with child under 12	50% discount will be given for both tickets
2	A person having over 60s rail card & family rail card & is traveling alone	34% discount on the ticket
3	A person having over 60s rail card only & traveling with/without his child	34% discount given to the person having over 60s rail card & no discount will be given to his child
4	A person having family rail card only & traveling with child	50% discount will be given for both tickets
5	A person having no rail card & is traveling alone	No discount (0%)
6	A person having no rail card traveling with child under 16.	10% discount will be given for both tickets

State Transition Testing

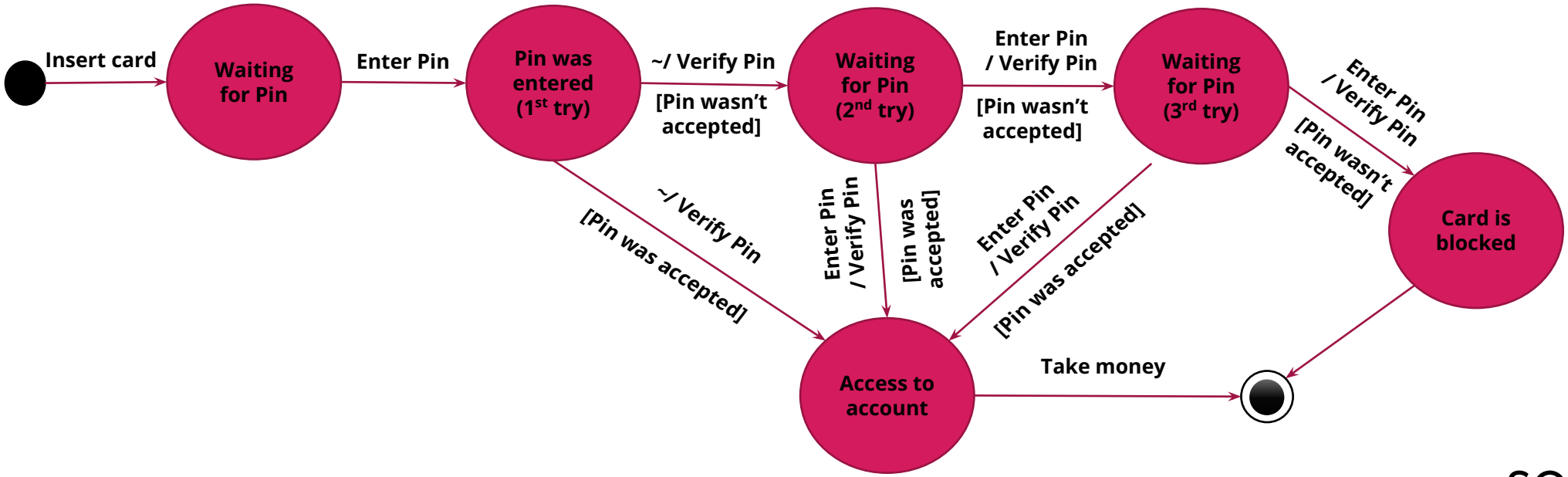
- **State transition testing** – A black box test design technique in which test cases are designed to execute valid and invalid state transitions.
  State transition – A transition between two states of a component or system.
- **Idea:** Design diagram that shows the events that cause a change from one state to another. Tests should cover each path starting from the longest state combination.



softserve

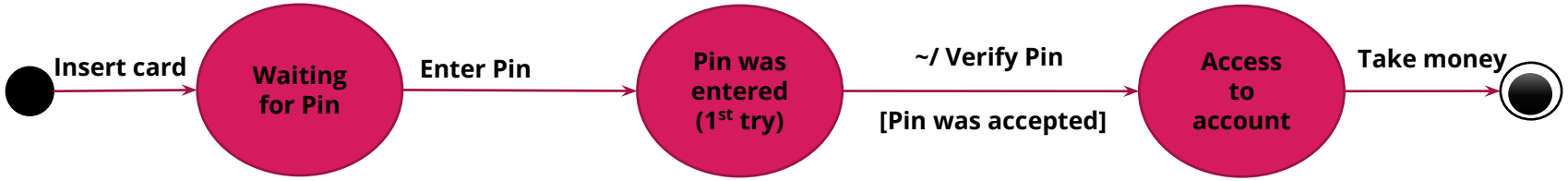
State Transition Testing. Example

- **Example:** Client of the bank would like to take money from bank account using cash machine. To get money he should enter valid Personal Identity Number (PIN). In case of 3 invalid tries, cash machine eats the card.



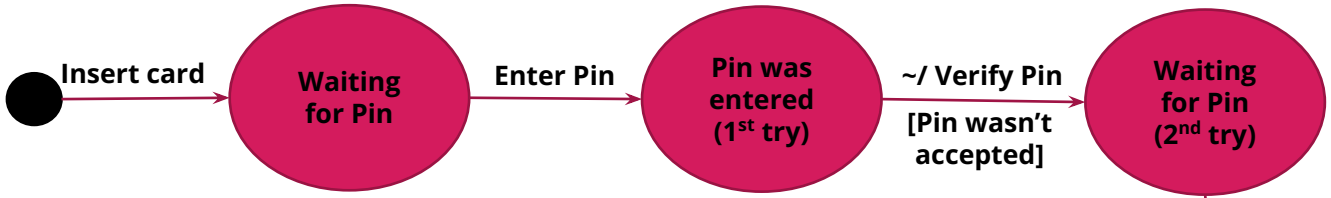
softserve

State Transition Testing. Example

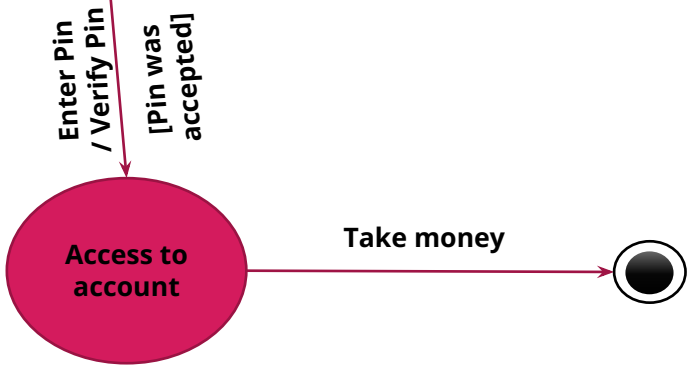


1. Verify that user will get money if he enters correct Pin in the first attempt

---

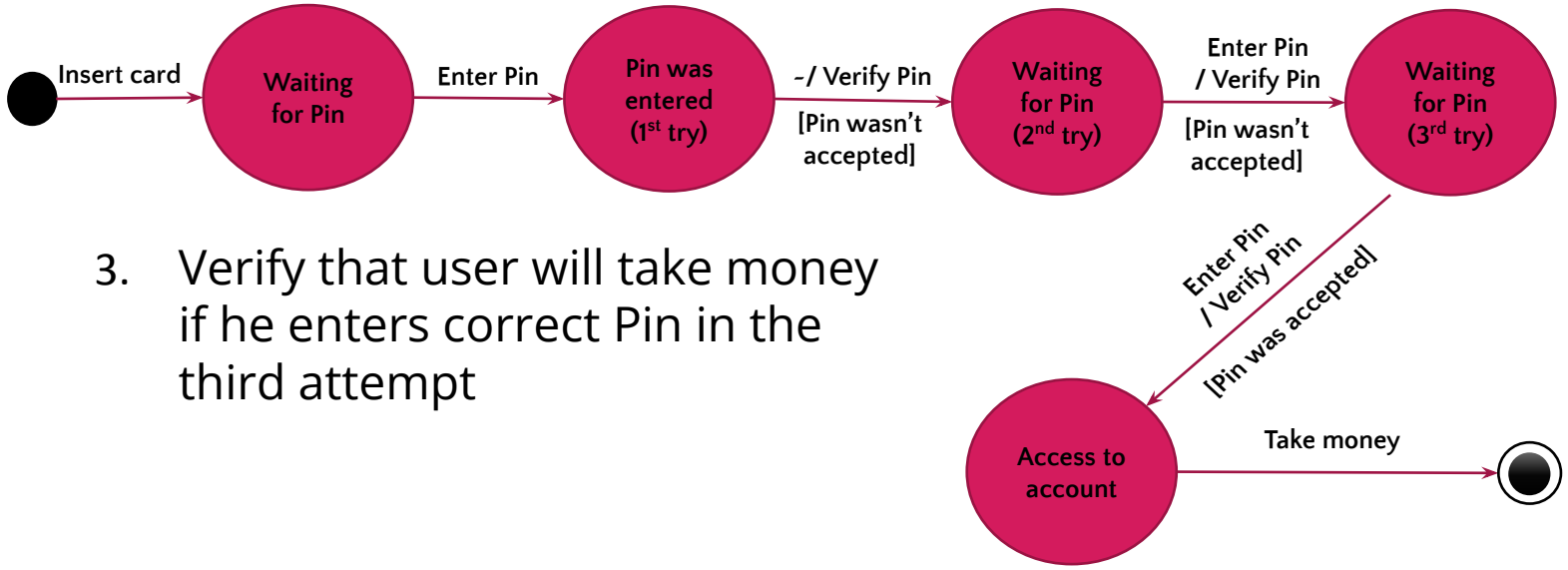


2. Verify that user will take money if he enters correct Pin in the second attempt

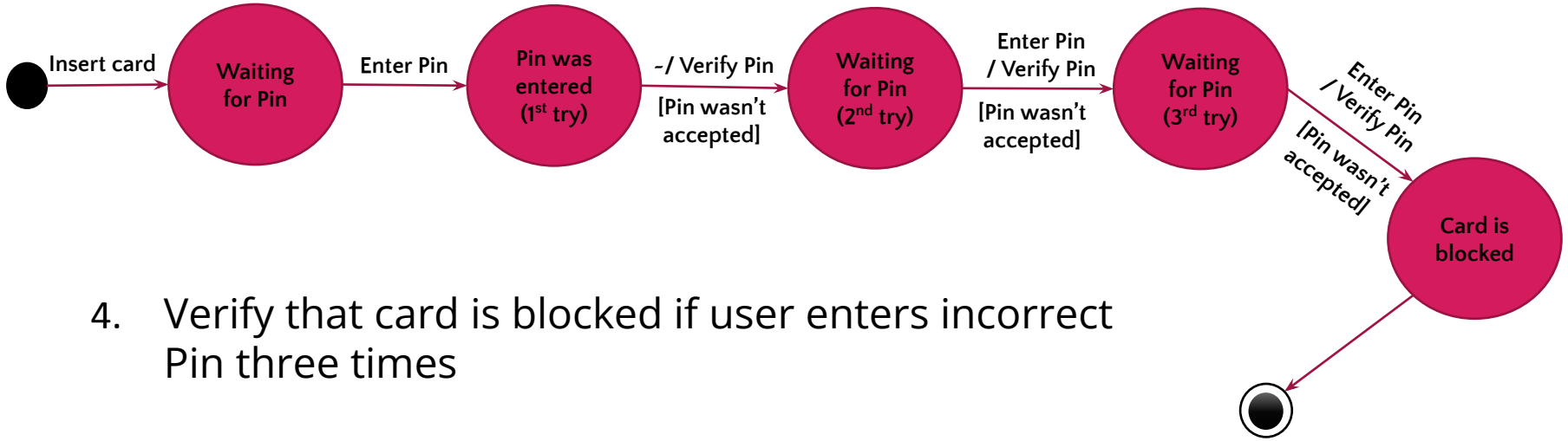


softserve

State Transition Testing. Example



3. Verify that user will take money if he enters correct Pin in the third attempt



4. Verify that card is blocked if user enters incorrect Pin three times

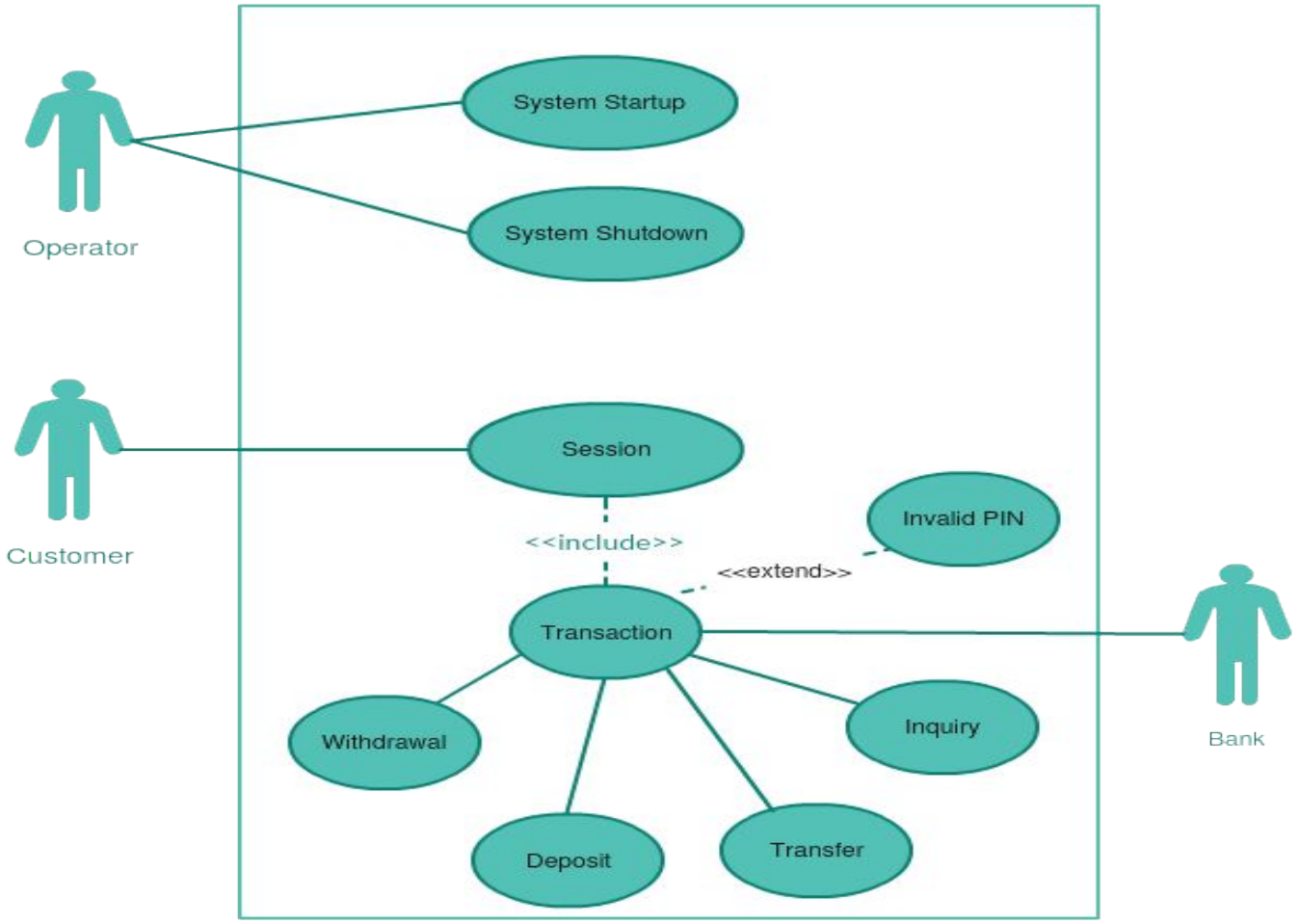
softserve

Use Case Testing

Use Case testing - is a technique that helps us identify test cases that exercise the whole system on a transaction by transaction basis from start to finish.

- Use cases describe the process flows through a system based on its most likely use
- This makes the test cases derived from use cases particularly good for finding defects in the real-world use of the system
- Each use case usually has a mainstream (or most likely) scenario and sometimes additional alternative branches (covering, for example, special cases or exceptional conditions)
- Each use case must specify any preconditions that need to be met for the use case to work
- Use cases must also specify post conditions that are observable results and a description of the final state of the system after the use case has been executed successfully

Use Cases for Simple ATM System

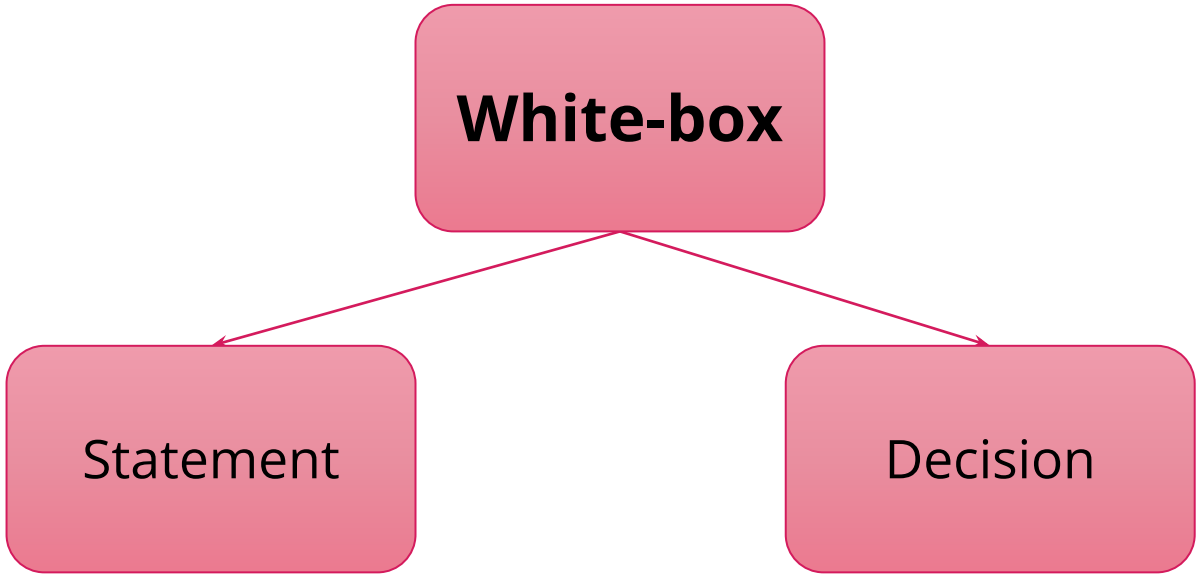


softserve

White-box Test Techniques

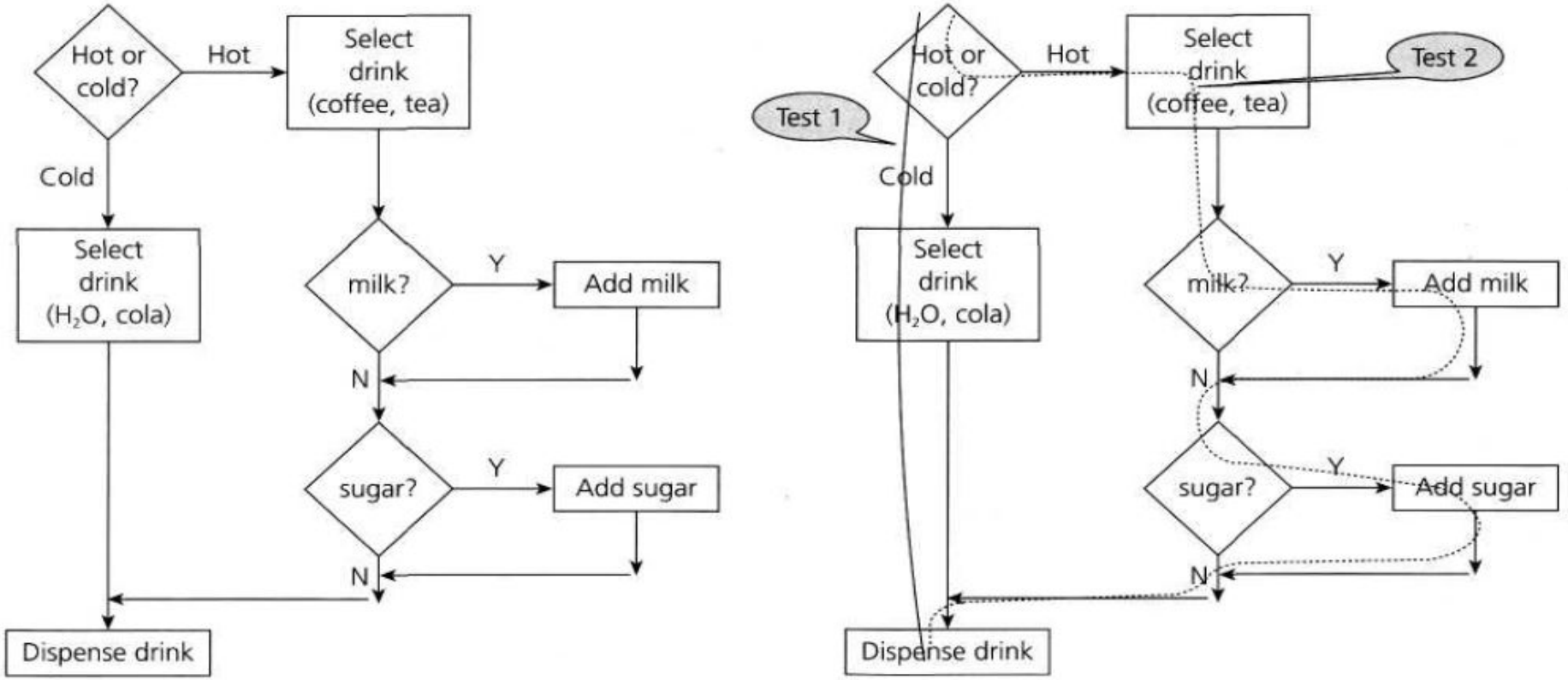
softserve

White-box Test Techniques



Statement Testing and Coverage*

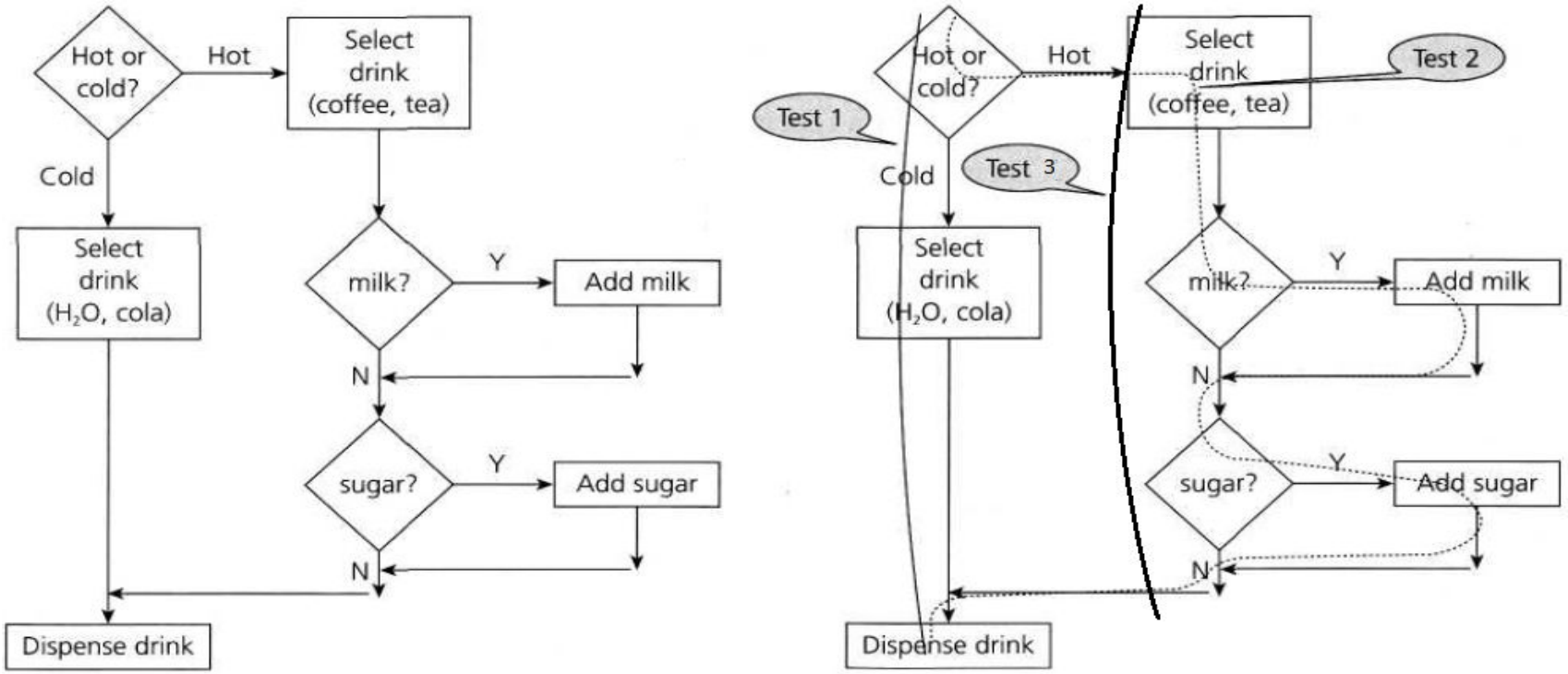
- **Statement** – an entity in a programming language, which is typically the smallest indivisible unit of execution.
- **Example:**



softserve

Decision Testing and Coverage*

- **Decision** is an IF statement, a loop control statement (e.g. DO-WHILE or REPEAT-UNTIL), or a CASE statement, where there are two or more possible exits or outcomes from the statement.
- **Example:**



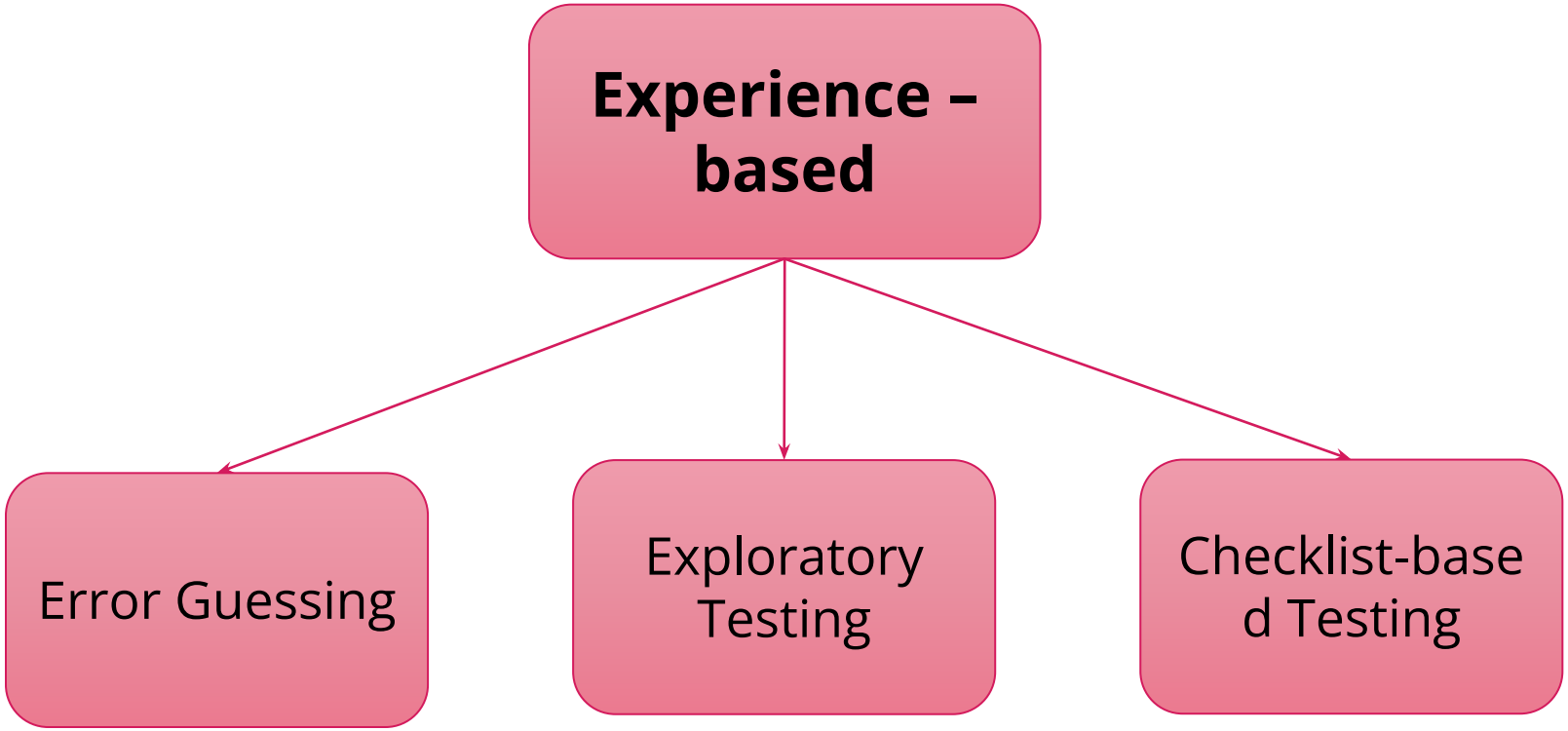
softserve

Experience-based

Test Techniques

softserve

Experience-based Test Techniques



Choosing

A Test Design Technique

softserve

Choosing A Test Design Technique

- **The internal factors that influence the decision about which technique to use are:**
  - Tester knowledge and experience
  - Expected defects
  - Test objectives
  - Documentation
  - Life cycle model
- **The external factors that influence the decision about which technique to use are:**
  - Risks
  - Customer and contractual requirements
  - System type
  - Regulatory requirements
  - Time and budget

Choosing A Test Design Technique

- **Which technique is best? This is the wrong question!**

  Each technique is good for certain things, and not as good for other things. Some techniques are more applicable to certain situations and test levels, others are applicable to all test levels.

softserve

Revision History

Version	Date	Remark	Author
v.1	March, 2016		M. Harasym
v.2	October, 2018	Update according to new ISTQB Standard	V. Ryazhska

Thank you

softserve