SEPTEMBER

M	T	W	T	F	S	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

SQE - L01

1437 ذی الحجه

September 2016

expectation + standards

=> Quality can not be defined but it is only stakeholders expectation

=> SQE : Process par jo disciplined kaam kiya jae

Testing → Static
Testing → Dynamic → Manual
Testing → Dynamic → Automation

L02

Quality Attributes :

- Robustness : Recover hojae
- Secure : Internally (Authorization)
- Safety : External attacks (Firewall)
- Scalability : Load balancing ~~(No balance~~
- Maintainability : Easy to implement change
- Reliability : Trusted System (Correct Functionality)
  - Functionally performing well
- Responsive
- Re-usability : Whatsapp update includes previous code as well as new features.
- Usability : System should do right things the right way. (Ticket reservation example)
- Portability : Different platform par use kiya ja skta he.

FRIDAY ۱۳ ذی الحجه 16
SATURDAY ۱۴ ذی الحجه 17
SUNDAY ۱۵ ذی الحجه 18

Evening

Scanned with CamScanner

Reading Assig : ISO-9126

ذى الحجه 1437

September 2016

SEPTEMBER
M T W T F S
5 6 7 1 2 3
12 13 14 15 16 17
19 20 21 22 23 24
26 27 28 29 30

Constraints in SQA Environment

- Resource constraint {Time & Budget} (Contracted Condition)
- Client Relationship
- Teamwork : - No cooperation or coordination (Other Dev teams / Cross Organization)
- Interfacing : System data-le whom ya de whom he doosre system ko
- Non-Compliance with documentation.
- Team change while in development

Evening Quality Perspectives: Total 5

19 MONDAY ١٩ ذى الحجه Transcendental view : Everyone different has a

20 TUESDAY ٢٠ ذى الحجه other understanding of word meaning seen/not defined

→ Error is human caused. That error will generally result in fault in the system (functionality if ∅, effected). Then it is a failure.

→ Defect is collection of Error, Fault, Failure

→ Relationship is not 1-to-1

Incorrect process, data in a program

Dormant/Latent Faults : Not leading to failure in current scenario. (but may in all

Design

Correctness ka measures

No. of bugs kia hen
System ka failure rate kiya he
System usable he ya nhi

GS DEALING:

fect Prevention
tection & removal (Debugging)
Defect Containment : Bug mojood he & usko chenna nhi he

LO4

kinds of quality:

uality of Design: (Difference in quality features)

Konsion se quality features hen product

WEDNESDAY 21 ١٨ ذی الحجہ
me - THURSDAY 22 ١٩ ذی الحجہ

uality of Conformance:

How much customer is satisfied by the work
How much the product meets its specifications.

Techniques to enhance quality

Quality Assurance VS Quality Control

Assurance is yakeen dehani karana while control actual implementation of standards.

Procen-Oriented
ocuses on defect prevention
Pro-active
Plan

-> Product-Oriented
-> Defect Identification & correction
Example: Mid Exams
-> Reactive Strategy
-> How much the plan is followed

Product Quality Problems

* Testing (Activity) is overlapped b/w QA and QC

1437 ذي الحجه

September 2016

=> QC identifies defect then NOTIFIES QA, QA then again work on its processes.

Hospital. QA

- Affordable
- Professional/Experienced staff
- Employees Treatment
- Standardize recruitment process

QC

- Sanitization
- Data maintainance of recruitment

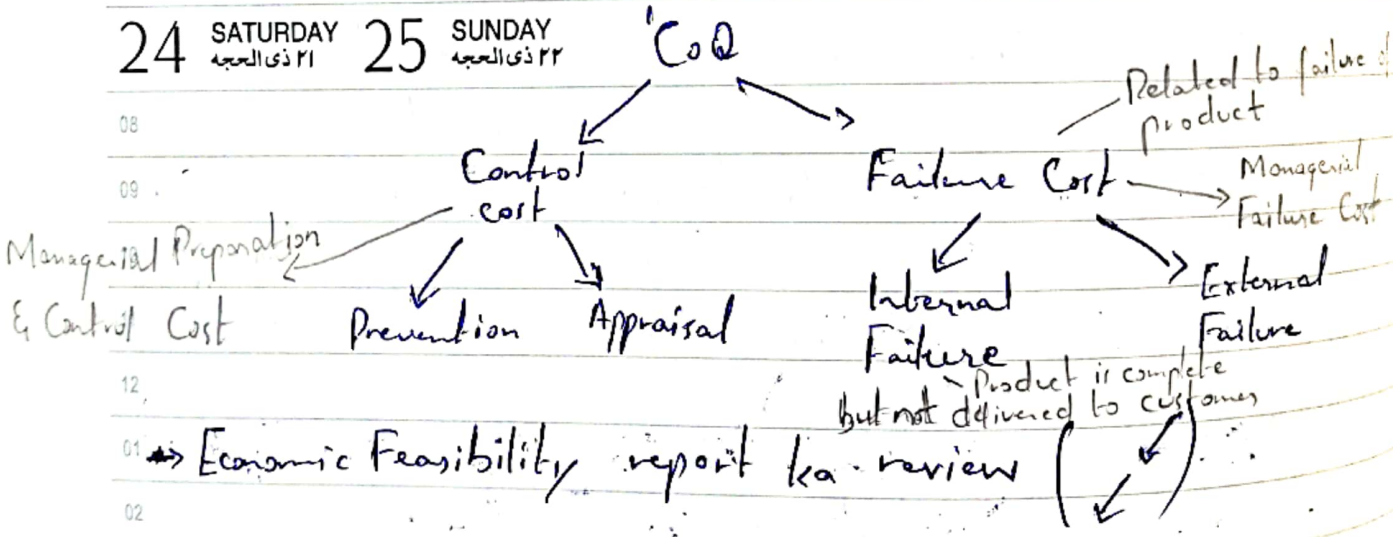
LO5

Evening - Software Cost of Quality Model.

* Cost of activities of QA and QC (Software quality)

23 FRIDAY ٢٠ ذي الحجه

24 SATURDAY ٢١ ذي الحجه 25 SUNDAY ٢٢ ذي الحجه



=> Economic Feasibility report ka review ( )

=> Employee Training Program ( )

=> SQA ko product dedi check krne kelie ( )

=> Product launch krne ke baad ( ) E.g. Laptop kharaab
ice masle (warranty par dena)

OCTOBER

M	T	W	T	F	S	S
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

Good Poor

ذی الحجہ 1437

Sep-Oct 2016

LO7

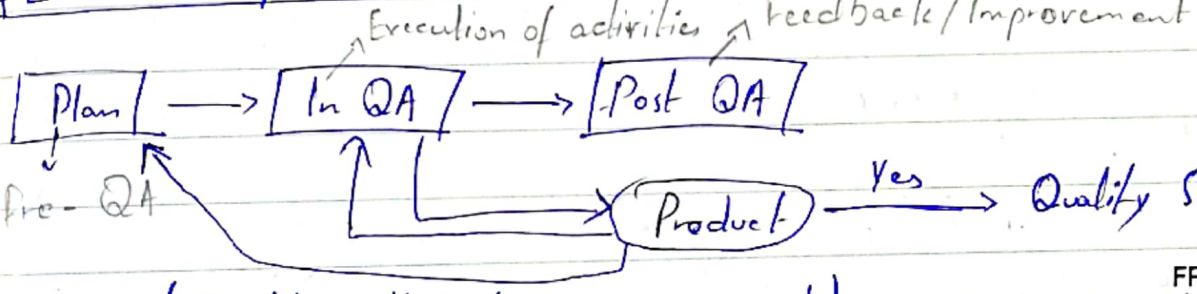
Control/CoGQ Failure/CoPQ

Cost of Conformance + Cost of Non-Conformance = Software CoQ

Project Inverse Relation Product

-> If defect level is high the cost will be :
- Recode
- Redesign

SQE Activities / SDLC:



-> Pre-QA (Quality attributes requirement)

-> In QA (Subset of Design Phase)

FRIDAY 30 ذی الحجہ ۲۷

SUNDAY 2 ذی الحجہ ۲۹ SATURDAY 1 ذی الحجہ ۲۸

Indirect: No. of bugs ko dekh kar failure rate calculate krna

-> Post QA (Analysis, Feedback and use that feedback for improvement)
- Analysis
- Feedback (Needed, Organizational Support)
- Measurement (of activities done in In QA)

QIP (Post QA Activities)

Evening

NCI NICE

QIP (① Understanding ② Assessing ③ Packaging)

1438 محرم

October 2016

Pre-QA

LO8

-> Quality Planning is a subset of Project Planning.

-> In QA : ~~Quality~~ Activities (QC) is subset of Design & Development.

-> Post QA : QIP is subset of Project Management.

Effort (Profile) :- Kis kis activities par kis process stage par kitni quality effort lag rhi he uski graphical representation.

Evening

LO9

3 MONDAY What is the need of SQA plan

4 TUESDAY -> Deliverables : Specify them such as Test Cases, the log of Test Cases etc.

Maps of development process (Gantt Chart)

OCTOBER

M	T	W	T	F	S	S
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

Revision

1438 محرم

October 2016

What is Software Quality ?

"Conformance of explicit stated functional & performance req., explicit documenting standards, implicit characteristics expected of all professional softwares."

"The degree of extent to which a system meets users expectations."

Why is it important ? · Safety · Cost (Bugs increase cost)
· Future value · Customer Satisfaction

Quality in Processes:

-> Specification : Must be correct, consistent

-> Design : Design model must conform to req.

-> Construction : Coding a/c to local standards.

-> Conformance : Application of QA activities

WEDNESDAY 5 محرم ٣

THURSDAY 6 محرم ٤

ISO-9126 : Six-top level quality characteristics (with each having sub characteristics)

-> Hierarchical Framework

-> FREMPU

Characteristics of SQA Environment Process

-> Contractual Conditions

-> Subject to - Customer-Supplier Relationship

-> Need for teamwork.

-> Cooperation with other development teams

-> Interfaces with software systems.

08 09 10 11 12 01 02 03 05 Evening 08 09 10 11 12 01 02 03 04 05 Evening

08 → Need to continue project while team changes.

09 → Need to maintain software for years.

10

11 Error : ◇ Fault : O Failure : ✡

12

01 Failure measurement : By examinig failure count, etc.

02

03 Reliability measurement : likelihood of failure to occur

04

05 CoQ Model : What are the cost related to quality?

Evening

→ Managerial Preparation and Control Cost :

7 FRIDAY → Cost of preparing project plans, periodic

8 SATURDAY 9 SUNDAY ...of project plans.

08

09 Pre-QA activities : (Planning)

10

11 In-QA activities : • Perform QA activities

12 • Deal with discovered problems

01

02

03

04

05

Evening

OCTOBER

T	W	T	F	S	S
				1	2
4	5	6	7	8	9
11	12	13	14	15	16
18	19	20	21	22	23
25	26	27	28	29	30

1438 محرم

L 10

October 2016

~~tidak~~ Verification And Validation :

- Making the right product
- Making the product right.

nch Coverage : Same as Path Testing.

FR kelie test cases likh skte hein? NO, because quality attributes vary person to person. So specific values kelie ly ~~likh~~ skte hein.
↳ verify kr

v to write verifiable NFR.

xample : System easy to use ho

Verifiable statement : User training 1 ghante ki karun to ers ko software chalana ajae

MONDAY 10 محرم ٨

TUESDAY 11 محرم ٩

lation of NFR : Quality attributes ko verifiable form me likhna.

Model :

Waterfall steps (And testing kahan se ho rhi he) rification and validation kis ke against ho rhi hei

⇒ Validation

⇒ Verification

08 L11

09 -> Different modules ke against different testing hoti he

10 -> Different type of applications kelie different strategies

11 and tools can be used.

12

01 Revision of Week 3

02 Short-Term Feedback: Identification of areas which need

03 special attention, progress tracking. etc

04

05 Long-Term Feedback: Necessary so Adjustments can be made

Evening QA strategies.

12 WEDNESDAY محرم ١٠ QIP (Quality Improvement Paradigm) (Post QA)

13 THURSDAY محرم ١١ -> 3 interconnected steps:

08 -> Understanding: the base line. So future process changes are

09 measured against baseline.

10 -> Assessing: introduce process changes (experiments, pilot)

11 and asses their impact.

12 -> Packaging: Package & all info and infuse into improvement

01 of development organization

02

03

04

05

Evening

OCTOBER

M	T	W	T	F	S	S
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

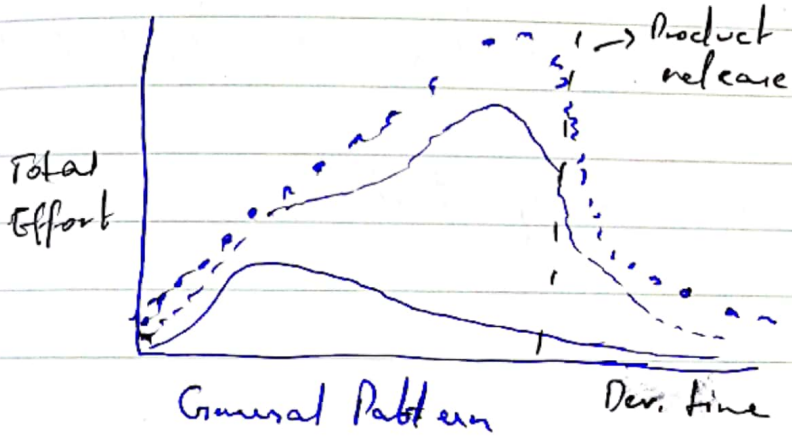
1438 محرم

October 2016

Effort Profile:

-> Human effort and computing resources equally distributed with both life cycle process characteristics and shift focus as project progresses over time.

-> Specific QA activities is dealing with defects (identify and remove) and ensuring quality



FRIDAY 14 محرم ١٢

SUNDAY 16 محرم ١٤ SATURDAY 15 محرم ١٣

-> Set quality goals by managing customer's quality expectations.

-> SQA is a process which ensures that SDLC is monitered and comply against defined standards. (ensure quality)

SQA Goals:
- Requirement quality (SQA ensure that the team has reviewed req. model)
- Design quality (Design should conform to req.)
- Code quality (Standards & maintainability)
- Quality Control Effectiveness (Resource are allocated in effective manner.

1438 محرم

October 2016

OCTOBER

M	T	W	T	F	S
31					
3	4	5	6	7	
10	11	12	13	14	
17	18	19	20	21	
24	25	26	27	28	

06 SQA Activities:

09 • Prepare SQA plan for project

10 • Participate in development of software process description.

11 • Review SE activities to verify compliance with defined process

12 • Audit work products. " "

01 • Ensures deviations are documented and handled a/c to defined procedure

02 • Records non-compliance and report it.

03

04

05

Evening

17 MONDAY
١٥ محرم

18 TUESDAY
١٦ محرم

08

09

10

11

12

01

02

03

04

05

Evening

OCTOBER

T	W	T	F	S	S
				1	2
4	5	6	7	8	9
11	12	13	14	15	16
18	19	20	21	22	23
25	26	27	28	29	30

1438 محرم

October 2016

L 12

hat we are doing (Test)

low we will do (Test Cases)

t Case Template.

Test ID, Req. Description, Pre-Condition, Post-Condition, Test Data,

How the test case, Expected output etc.

Actual Outputs, Comparison (Actual & Expected), Pass/Fail

t Suite:

-> Set of all test cases (fail or pass kitne hen)

ds of Testing:

it test (Individual module)

egration test (modules work with eachothers?)

stem test

AT (Alpha, Beta)

} Levels of Testing

WEDNESDAY 19 محرم ١٧

THURSDAY 20 محرم ١٨

ck Box Testing: (Give input to System and compare the actual output to expected output) (Real life me mobile hena)

tic Testing: Without executing the source code

amic Testing: Executing the source code

08 09 10 11 12 01 02 03 04 05 Evening

1438 محرم

October 2016 L13

Boundary Value Analysis:

1 variable ke no. of test cases : $4n + 1$

(~~variable~~ numbers of inputs)

1. min
2. jus above min
3. mid
4. just below max
5. max

⇒ Input valid range mei hoti chahie

Documenting Way

Test ID	Input data	Expected Output

21 FRIDAY محرم ١٩

22 SATURDAY محرم ٢٠ 23 SUNDAY محرم ٢١

~~The right answers x and y (4n+1) cases and all their cartesian~~

	x	y
1	100	300
2	101	300
3	200	300
4	299	300
5	300	300
6	200	200
7	200	201
8	200	399
9	200	400
	~~200~~	~~300~~

Nominal values ke saath x ke (4n+1) cases then

y kisi nominal value pr x "

"

Nominal = Average

Repeat test case hence eliminated

L14

Boolean variables par Boundary Value Analysis NAHI apply hoti.

Equivalance Class Testing :

1 Divided in minimum two : ① Valid ② Invalid

Valid and invalid can be further divided.

Example :

1st classes

① $1 \leq x \leq 100$ (Valid) (Range)

② $x > 1$ (Invalid)

③ $x > 100$ (Invalid)

If BVA not applied (3 cases : Min, Mid, Max)

MONDAY ٢٢ محرم 24

TUESDAY ٢٣ محرم 25

08 09 10 11 12 01 02 03 04 05 Evening

20K-1708

Boundary Value Analysis : $4(3)+1 = 13$

Range [1-300]

	x	y	z
1)	1	150	150
2)	2	150	150
3)	150	150	150
4)	299	150	150
5)	300	150	150
6)	150	1	150
7)	150	2	150
8)	150	299	150
9)	150	300	150
10)	150	150	1
11)	150	150	2
12)	150	150	299
13)	150	150	300

Equivalent Classes:

$I_1 = \{1 \le x \le 300 \text{ and } 1 \le y \le 300 \text{ and } 1 \le z \le 300\}$ (x, y and z are valid)

$I_2 = \{1 \le x \le 300 \text{ and } y > 1 \text{ and } z > 1\}$ (x valid, y, z, invalid)

$I_3 = \{1 \le x \le 300 \text{ and } 1 \le y \le 300 \text{ and } z > 1\}$ (x, y valid, z invalid)

1438 صفر

November 2016

Revision of Week 4

Testing: aka (Software Quality Control or SQA)
Goal: Validation & Verification

Verification : Program 'X' meets its specifications (the things right)
Validation : Are we building the Right product ? (the right things)

- Testing is expensive ($\frac{1}{3}$ or $\frac{1}{2}$ of cost of development project)

-> A good test set is :
1) covers all functional capabilities of system
2) cover all code using metric such as "Branch Coverage".

WEDNESDAY 2 صفر

Branch Coverage:
Every branch = Every 'True' and 'False' condition of the program.

THURSDAY 3 صفر

-> Testing ensures quality and increase confidence in deployment of system

What makes V & V difficult ?
- Different quality requirements
- Evolving (and Deteriorating) structure
- Uneven distribution of faults.

08 Validation & Verification:

09 - Includes usability testing, user feedback

10

11

12 - Review of specification of system, subsystem and each unit.

01 - Also UAT of defined product

02

03

04

05

Evening

- Includes testing, inspections, static analysis

- Testing of System Integration, subsystem integration ~~and unit/component testing~~.

4 FRIDAY ٣ صفر - Verification activities lead to specification during

5 SATURDAY ٤ صفر 6 SUNDAY ٥ صفر or at the end each development process

08 which is then validated to "Actual Needs and

09 Constraints".

10

11 Error: Made during code (by Human) - aka 'bug'.

12 Error/Mistake/Defect in coding is called as 'Bug'.

01 Condition to software to fail.

02 Fault: Fault is a representation of error in DFDs, ER

03 diagram, Use cases etc.

04

05 Failure: Failure is result of execution of a fault.

Evening - Dynamic in Nature (Program needs to execute)

- A fault may lead to many failures.

When does V & V start and end?
From Feasibility study beyond maintenance

How can we asses the readiness of a product?
- Specify required level of Dependability and determine whether it has been attained.

Measure of Dependability:
- Running versus Down Time
- Mean Time Between Failure (MTBF)
- Successful operations' fraction.
  ↘ run

Assessing Dependability:
- Alpha Test
- Beta Test

MONDAY ٦ صفر 7
TUESDAY ٧ صفر 8

How can development process itself be improved?
- identify and remove weaknesses in development process and test and analysis

4 Steps to Improve fault analysis and process:
1) Define Data & how to collect it
2) Analyze Data to make fault classes
3) Analyze fault classes to identify weakness
4) Adjust the process.

Test Case Template:

Before Execution:
① Purpose / Goal ② Input(s)
③ Pre-Condition(s) ④ Expected output
⑤ Written by ⑥ Date of design
⑦ Expected output(s)

After Execution:
① Output(s) ② Post-condition(s)
③ Pass/Fail ④ Reason for failure
⑤ Run by ⑥ Suggestion
⑦ Date of suggestion

UNIT TESTING:

Scaffolding: The complete additional code written for design of "stub" and "driver".

9 WEDNESDAY ٨ صفر

10 THURSDAY ٩ صفر

- To handle activities of unit A and unit B
- To handle activity of unit X

White-Box techniques are effective at Unit Testing

INTEGRATION TESTING:

Coupling: The relationship b/w combination of 2 units.
- Highly coupled (Dependency ↑)
- Loosely coupled (↓)

- A good design should have low coupling.

SYSTEM TESTING:

- Generally, functional testing techniques are used here. (structural may also be used)

- Only phase to test both Functional and NFR.

UAT:

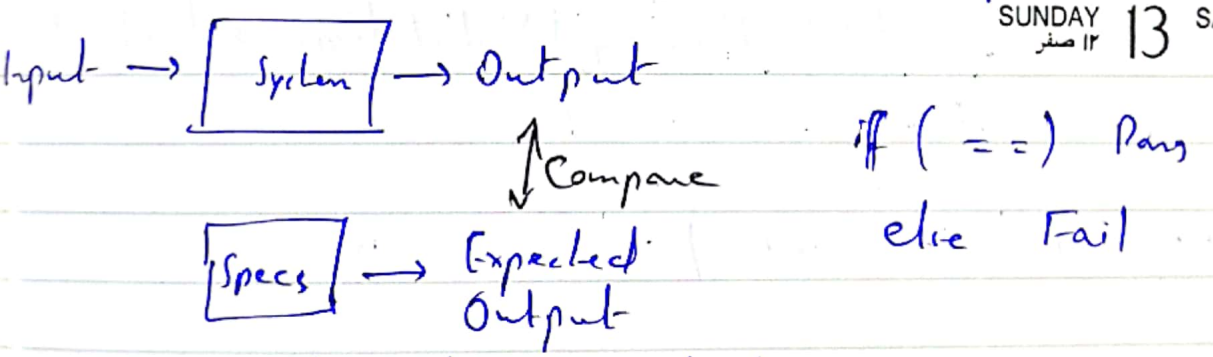
Feasible, only when ~~Base~~ product is made for a specific customer.

In Case of generalized customer base. (Alpha & Beta Testing)

Black Box Testing:

Input, get Output compare it to expected output.

FRIDAY ١٠ صفر 11
SUNDAY ١٢ صفر 13 SATURDAY ١١ صفر 12



- Need to automate this testing
- Atleast one test case per functionality.

Grey Box Testing:

White Box testing:

08 QA in testing:
09 - To enforce standards & technique to improve development
10 process.
11 - Prevent previous faults from ever occuring again.
12 - Defect Prevention technique
01 QC in testing:
02 - If failure, occurs remove it and ensure correctness
03 of removal.
04 - Concentrates of product rather than process.
05 - Defect detection and correction technique.
Evening - Done after completion of software development.

14 MONDAY ١٣ صفر Static Testing: Without Running (Verification activities)
15 TUESDAY ١٤ صفر Dynamic Testing: With Running (Validation activity)

08
09
10 Functional Testing (Black Box Testing)
11
12
01
02
03
04
05
Evening

NOVEMBER

M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

MID 2

1438 صفر

November 2016

L 16:

- BVA cannot be applied where multiple conditions are present.
- Alarm System (When alarm goes off : Multiple conditions)

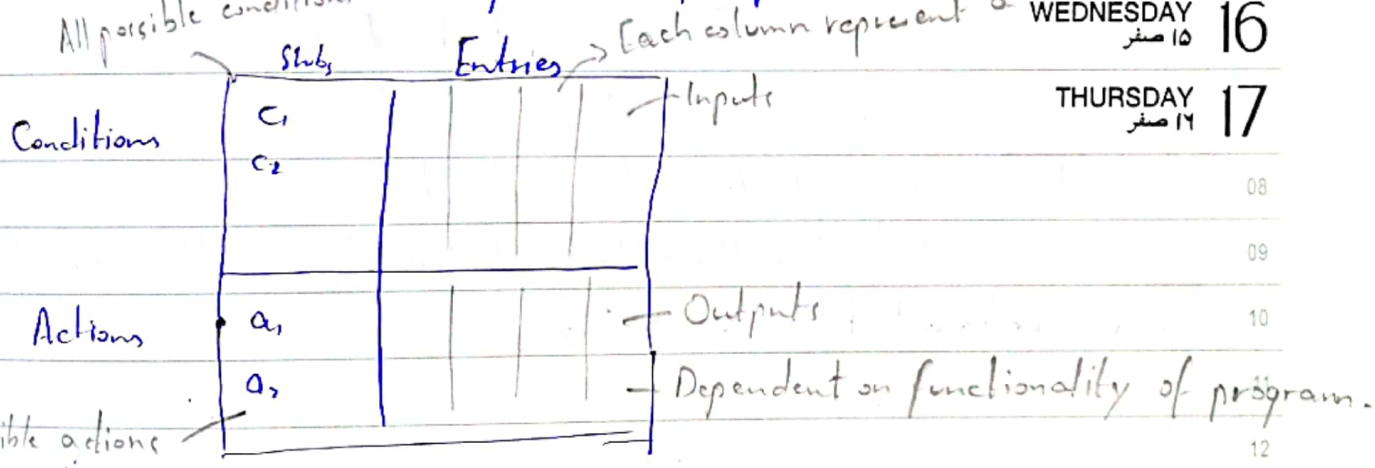
Decision Table : based Testing

1. Conditions
2. Actions (Function)
3. Rules (Combination of different condition and actions)
   - Column in Entries portion of the table

⇒ BVA and Equivalence Class Partioning (EP) both combine covers the full domain(valid and invalid)

Evening

Decision Table : When we large sets of input



WEDNESDAY 16 ١٥ صفر

THURSDAY 17 ١٦ صفر

- Pictorial views of various combinations of input conditions

Condition Stub : To determine action or set of actions

- Only (T/F) conditions (Binary), the decision table is known as Limited Entry Decision Table
- Where multiple conditions (other than T/F), is known as Extended Entry Decision Table.

JPG (T/F)	T	T	T	T	F	F	F	F
Size < (T/F)	T	T	F	F	T	T	F	F
Resolution (T/F)	T	F	F	F	T	F	T	F
Action (U/NU)	U	NU	NU	NU	NU	NU	NU	NU

Each of the column is a test case = 2^n

n = No. of Inputs / Conditions

Can be varied, but it should fully cover the Domain.

Limit on Decision Table based Testing:

Where one conditions must atleast be True

Evening E.g. TA or SLA 1 saath nhi ho sakta

Do Not Care Condition: Notation (-) Minus Sign

18 FRIDAY صفر ١٧

19 SATURDAY صفر ١٨ 20 SUNDAY صفر ١٩

Example:

T	T
-	F
-	-

Doesn't have any effect on the output - If one condition is false and makes the action false than other conditions are irrelevant (Do not Care).

Rule Count:

Condit	
Action	

← Rule Count comes here

- Comes with 'Do not Care' conditions.

If 0, do not care conditions, the Rule Count = 1

Evening Rule Count = $2^{\text{no. of do not care}}$

- Only applicable in Limited Entry Decision Table

- Total No. of rules for every decision table = Sum of Rule Counts

Applicability:- Where output is dependent on many conditions
- Large no of decisions are required to be taken
(Only)- At unit level testing (If size too large, difficult, handling)

- Impossible condition (Written in Action Stub)

L 17:
Cause Effect Testing: (Graphic Technique)
- Ye waja hogi to ye action hoga

⑥ Identify no: of causes

① Cause effect graph banana hai
② Conditions (Apply constraints) } Advantage of ~~CEGT~~ Cause effect

MONDAY ٢٠ صفر 21
TUESDAY ٢١ صفر 22

③ Decision Table banana hai
④ Test Cases generate.

Nodes represent Cause and Effect

① Identity : C_1 —(Identity)— e_1 Ye cause c_1 true hoga to effect e_1 true hoga. Nhi to else e_1 is 0

② NOT : C_1 —¬— e_1 Condition c_1 true hogi to effect e_1 nhi hoga.

③ AND : C_1, C_2 → $\wedge$ → e_1

④ OR

صفر 1438

November 2016

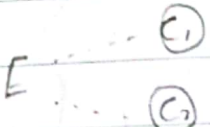
NOVEMBER

M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

08 **Use of Constraints:**

09 (1) **Exclusive**: Both c_1 and c_2 cannot be simultaneously 1.

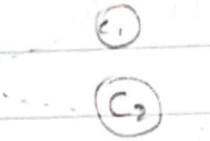
10 Either c_1 is 1 or c_2 is 1. But, both c_1 and c_2 can be 0, simultaneously.



12

01 (2) **Inclusive**: Atleast c_1 or c_2 must always be 1. But both cannot

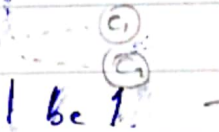
02 be 0 simultaneously.



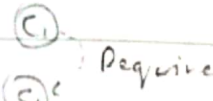
03

04 (3) **One and Only One**: Sirf 1 c_1 or c_2 me se 1 hi 1 hoga. Or

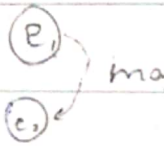
05 1 hi 0 hoga.



Evening (4) **Requires**: c_1 to be 1, c_2 must be 1.

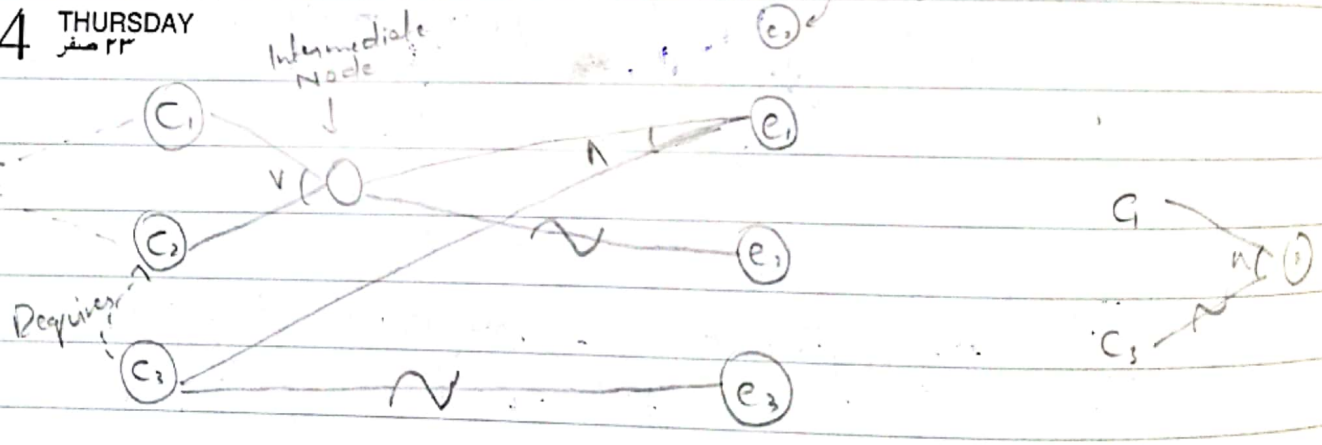


(5) **Mask**: Cause c_1 hoga to no effect ki ho mask kardega.



23 WEDNESDAY صفر ٢٢

24 THURSDAY صفر ٢٣



02 c_1 = U c_2 = M c_3 = # of children

03 e_1 = updation e_2 = error message (manipulation)

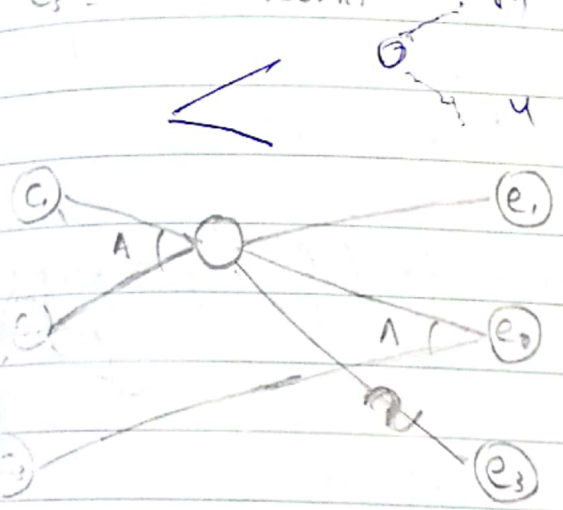
04 e_3 = error (# of children)

05

Evening

L18: c_1 = Age > 21 c_2 = Clean record c_3 = business tourist

e_1 = supply car (without premium)
e_2 = charge premium
e_3 = end message



If there are a lot of no. of combinations 2^n is very large then, we do:

Pair Wise Testing: (L19)

- Subset of combinations
- Maximum no. of test cases with minimum no. of combinations

Step 1: List down the variables involved.

Step 2: Simplify (Registration 5000 ki jaga valid OR invalid ka pair bana dia to total 2 values hogain 5000 ki jaga)

Step 3: Arrang variables

1 2	1 2	1 2 3	1 2

Arrange values involved

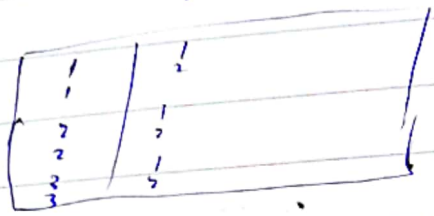
1 2 3	1/2	1/2	2

FRIDAY 25 صفر ٢٤
SUNDAY 27 صفر ٢٦
SATURDAY 26 صفر ٢٥

صفر 1438

November 2016

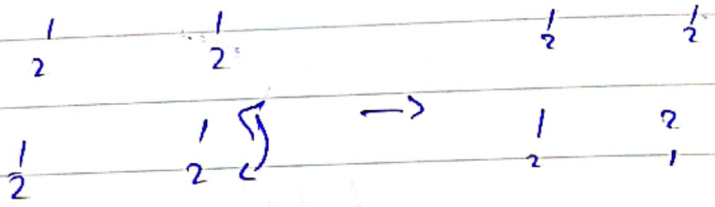
Step 4: Arrange variables to create a test suite



Largest colum ko aage wale columns ke hisaab se divide krdena

(For example: Ben wa had 2 values in the following columns so each value of 1st column will be divided into 2)

→ Re-arrange to cover all possible combinations.



28 MONDAY صفر ٢٧

29 TUESDAY صفر ٢٨ → See slide for the last column (if swapping se cover nhi horhi to add kerdi values)

L20:

State-Transition Diagram:

Action that leads from one state to another.

State-states transition depending on the trigger/actions.

State can be denoted by : S_i $i = 1, 2, 3$

Table:

State	Input (Transition)	Next State

L 22:

~~Structural Testing~~:

→ Requirements are feature and features are use cases.

→ Use Case Testing:

Use Case Testing ka base document = SDA: Use Case Document

↳ { Name, Stakeholder, Pre-Con, Post-Con, Trigger, Normal Flow, Alternate Flow

1st Use Case: Normal Flow
2nd " : Alternate Flow
3rd " : "

Evening

→ No need to re-write each step, just write "Repeat Steps (1c)"

MONDAY 5 ۴ ربیع الاول
TUESDAY 6 ۵ ربیع الاول

Test Cases	Expected Result
Normal Flow: Check out a item	
1) Login to website	Home screen is displayed
2) Search for an item	The matched item is displayed
3) User selects one or more item	Item is added to cart
Alternate Flow: Invalid Credentials	
1)	

08 L23.

09 Structural Testing:

10 - More technical than functional testing b/c here we
11 know the internal working (code) of system
12 - Kind of "White Box" testing.
01 - Base Document: Source Code, Implementation
02 4 techniques (we will learn)
03 1. Control Flow Testing
04 - Path Testing
05 ↳ Flow Chart etc. (Check each path individually)

Evening → Different variations of Control Flow Testing:

① • Statement Coverage - (All statements of program)

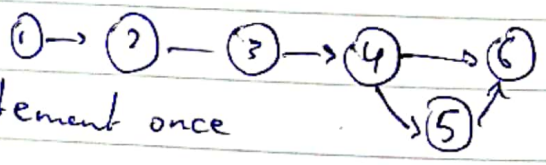
7 WEDNESDAY ② • Branch Coverage

8 THURSDAY ③ • Condition Coverage

08 ① Test Data: variable values (Input)

09 Steps: 1. Number the code (Compound Statement ko 1 number)

10 2. Control Graph

11 

→ Atleast Control covers each statement once

01 ② 1 → 2 → 3 → 4 → 5 → 6

02 1 → 2 → 3 → 4 → 6

03

04 Cyclomatic Complexity: No. of paths that exists in your program.

05

Evening $V(G) = e - n + 2P$

e = edges

n = nodes

P = If function is called in b/w

V(G) = Paths

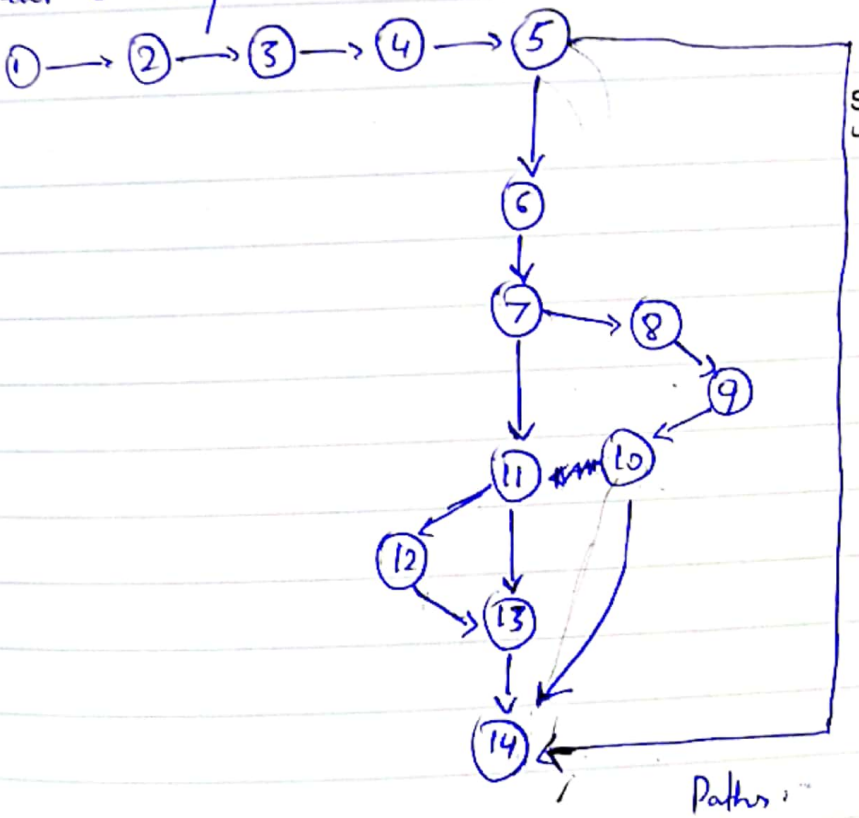
- The main path which covers all statements is not included in "path coverage" b/c every other path is covered means the main path is also covered.

(5) Condition Coverage

- if (PP _)
  - 1st is True, 2nd is False
  - Both True
  - Both False
  - 1st False, 2nd True

L24:

Statement Coverage:



Paths:

1-5, 14

1-10, 14

1-7, 11, 12, 13, 14 (11-14)

1-7, 11, 13, 14

16 - 14 + 2 => 4

Date _____

L25. → Control Graph in all Structural Testing

Data Flow Testing,

Anamolies: ① Variable is used but not defined
② Repeated declaration

Uses of variables, ③ Variable is defined but not used
④ Variable is used before even first initialization

Nodes < Definition Node : DEF (v, n) — node, variable
Usage Node : USE (v, n)

→ paths (begin, end) — defined, used

Defined : Where value is assigned to variable

=> All<d,u> paths

hi	3	5, 6, 10	3-5, 3-6, 3-10
mid	6	7, 8, 9, 10, 11, 12	6-7, 6-8, 6-9, 6-10, 6-11, 6-12
low	2	5, 6, 12	2-5, 2-6, 2-12
result	4	8, 14	4-8, 4-14

low = 0 — 3 10 5 6
high = null — (3-5)
result = -1 — (3-6)
mid. — (10-5)
(10-6)

Date

L 26.

Data Flow → Test all uses
→ Test all definition

Gray Box: Architecture knowledge

Slice Based Testing:

- Criteria will be defined (All d;u OR All uses OR All definitions)

Guidelines: (Not exact rules) (possibilities)

↳ All statements where variables are defined and redefined

↳

↳

↳

Example:

$a^1 = 3$, $b^2 = 6$, $c^3 = b^2$, $d = a^4 + b^2$, $e = a^5 + b$

Two Slices

1. a = 3
2. b = 6
5. c = a+b

S(c,5)

1. a = 3
2. b = 6
3. c = b²

S(c,3)

'c' at statement 5

→ Slicing Criteria : S(variable, line_number) = { . , . , . } (line array)

S(A,6) S(A,13) S(A,28) S(B,8) S(B,24) S(B,28) S(C,10) S(C,16)
S(C,21) S(C,28)

Date _____ Revision For Mid 2

Equivalence Class Partitioning:

I_n (Test Classes) $= 3^x$; x = no. of variables

Example: 3 variables $= 3^3 = 27$ test classes

1 Valid, 2 Invalid $= 4 + 4 + 4$

2 v, 1 iv $= 2 + 2 + 2$

3 v $= 1$

3 invalid $= 2^x - 2^3 = 8$; x is variables

$= 27$

Step 1: Note down classes

$I_1 = \{ \}$, $I_2 = \{ \}$ ....

Step 2: Generate test cases

Test Case	x	y	z	Expected Output
I_1	1	1	1	---
I_2	2	2	2	---

I_n = Input Domain

O_n = Output Domain

Total no. of Equivalence Class Testing = $I_n + O_n$

Decision-Table Based Testing:

→ When Output is dependent on multiple inputs, each set of inputs gives a different output.

→ Represents complex logical relationships.

Date

Cause-effect Testing:

Cause = Input. Effect = Output

→ Effective only for small programs

Applicability : Dependency of inputs

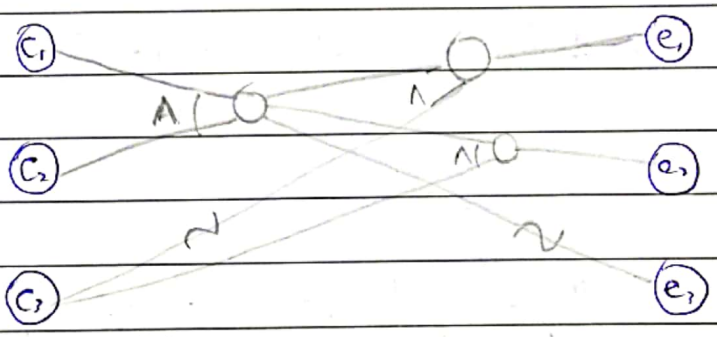
Steps: 1. ~~Make Graph~~ Identification of all cause & effects
2. Design graph
3. Apply constraints, if any
4. Design limited entry Decision Table
5. Write test cases using column of decision table

Example 2.17.

c_1 = age > 21 e_1 = supplied car

c_2 = clean driving record e_2 = car (with premium)

c_3 = Business e_3 = message



State Transition Testing:

Input document : State transition diagrams.

→ Based on fact that same action will give different results depending on initial state.

→ State ——Action = Transition——→ Next State

Date _____

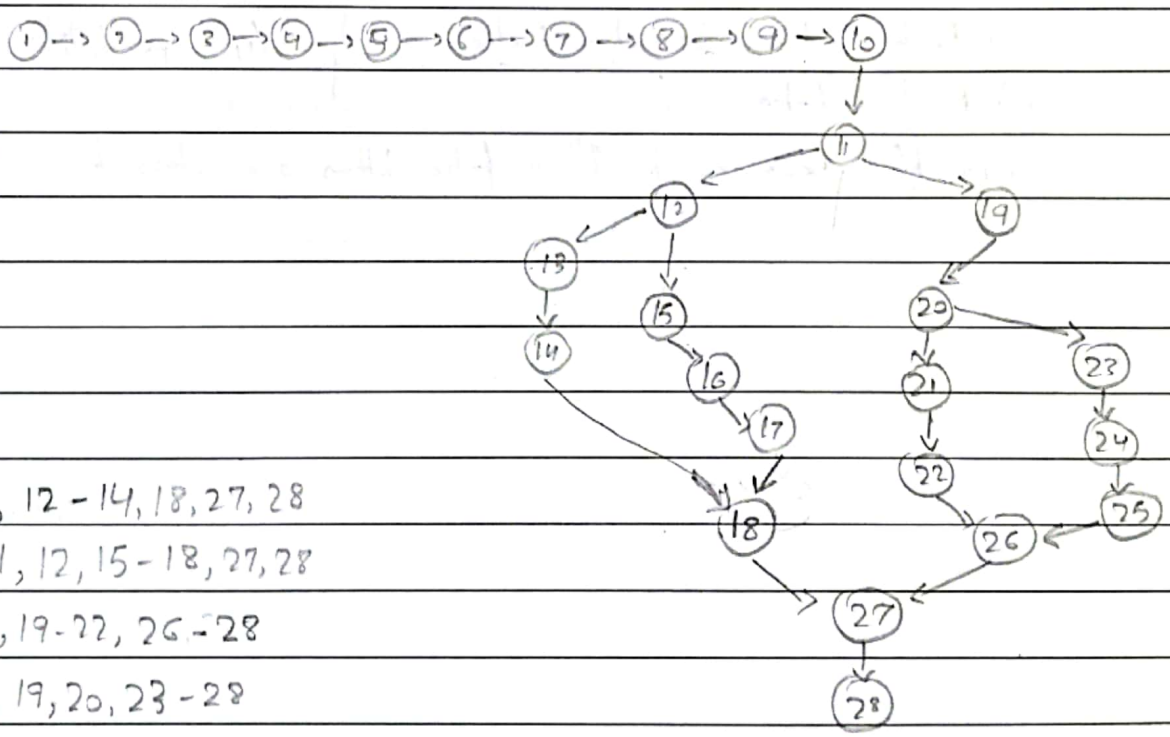
→ Drawback : Any possible "un-identified" state or transition can lead to potential defects being left.

States	Input	Next States

Control Flow Testing :

Total : Paths we get by identifying them ourselves

$V(G) = e - n + 2P$ gives us independent paths



1-11, 12-14, 18, 27, 28

1-11, 12, 15-18, 27, 28

1-11, 19-22, 26-28

1-11, 19, 20, 23-28

30 - 28 + 2 = 4

39 - 35 + 2 = 6

Date

Branch Coverage:

When we want to check every 'True' and 'False' condition of the program.

if ( ) — Data where all 3 if become true

if ( ) — and where all 3 if become False

if ( )

Condition Coverage:

Where 1 statement has 2 or more conditions

if (&&)

Possibilities:

i) Both True

ii) First is true, second is False } 3/4 possibilities

iii) First is false (

3 are enough b/c because if 1st is false other one doesn't matter

Date _____

Data Flow Testing.

Main points:

→ Statement where values are used (referenced)

→ Statement where values are assigned to variables (definition)

→ The procen is used to Detect Bugs b/c of anomalies

① Draw Program Graph → Graph

② Find all defined and used node → Table 1

③ Generate all du-path → Table 2

Limitations.

1- Testers require good knowledge of programing

2- Time Consuming 3- Costly process

Domain Analysis Testing.

→ Where application is tested by giving min. number of input and evaluating its outputs

Goal, Whether software accept inputs within acceptable range and delivers required output.

Drawback: Have to have ~~specific~~ domain specific knowledge

BVA an EP partitioning

~~How~~ First make classes I_1, I_2, I_3 etc

What boundary values to pick

Suppose if range (" > 5 and <= 10)

pick 5, 6, 10, 11 ~~I = {~~

Not in Range

Date

Equivalence partitioning value se b/w the range wali uthani
if (>5 - 10) pick 8

Scenario	BV to be taken	Equiv D value
Boy - Age >5 and <=10	Age = 5	Age = 8
	= 6	
	= 10	
	= 11	
Boy - Age >10 and <=15	= 10	
	= 11	
	= 15	
	= 16	

Scenario	values to be taken	Test Case	Expected Results
Age >5 and <=10	Gender - Boy BV Age = 6, 5, 10, 11 Eq to be taken 8	Age = 5 Age = 6 A = 10 Age = 11 Age = 8	story telling Rhymes Quiz Story Telling

Age = 5 Input = 5 Input = 3
Input = 4

Date _____ FINALS

L28: MUTATION TESTING: (Test Cases ki testing)

* If code is changed, test case are not changed.
* In this testing, we check effectiveness of the test cases.
  Checking through mutations (change)

→ How Mutants are made?
  → Mutant Operators : Jisse change aega
  → Aesa change krna he jisse code remains executable (invalid change nhi krna)
  → Versions are known as 'Mutant' of that program, i.e, $M_1, M_2 \ldots$

→ No of Change in program : If 1 Change = 1st Order Mutant
  " 2 changes = 2nd Order Mutant
  (If another change in 1st one)

→ Default/General is 1st order Mutant

$$\text{Mutation Score} = \frac{\text{No. of Mutants killed}}{\text{Total no. of mutants}}$$

(Test Suites ki base par (Not a single test case)) — $M_1 + M_2 + M_3$; $0 < \text{range} \le 1$

$M_1 + M_2 + M_3$ (Killed + Live)

If program output is changed (effected by) and test case detects it, it is called "Killed Mutant"

Live Mutant is vice versa of this:
(When mutant is not detected)
Example: $x - y$ where ye 5 is changed to $x - 5$. (No Output is effected, so no change is detected, hence Live Mutant)

Example:

Test	a	b	c	F.O
1	6	10	2	10

M_1

```
if (A < B)
    if (A > C)
        print (A)
```

M_2

Tst.	a	b	c	F.O	A.O
1	6	10	2	10	6

Change Occurs

T.

Killed Score = 1 for M_1

Project Deliverables: ① Purpose (What application) ② Features to be tested/not tested ③ Approaches (Black Box Testing) ④ Environmental Needs ⑤ When to stop } 1. TEST PLAN

Date

→ $M_1, M_2, M_3 \ldots M_n$ are independent of each other if all 1st Order Mutants

L31:

Driver : Calling Component (Implement nhi hua lekin call kar rha he)

Stub : Called component (Implement nhi hua lekin return ker rha he)

Integration Testing

- Top-Down approach (Stub Code)
- Bottom-Up approach (Driver Code)

Project Deliverables:

2- Test Case/Test Suite (For Main Features) — For every feature = 1 Test Suite

①

3- Test log Report

TID	Feature	Test Data	Expected Output	Actual Output	P/F

Regression Testing:

If change occurs, test the system again.

Slides :- Test Data → Function called → Expected/Actual Output

Date

L32. Other QA Activities (Chapter 13)

Defects Prevention Technique

-> Preventing injection of faults into the system

2 strategies:

1. Error Blocking: Pre-defined procedures removing (minimizing) chances of error.

2. Error Source Removal: For people like appraisal cost (to train them)
   -> Stopping the error at its root

-> Root Cause Analysis will not be effective when you work on unfamilian Technology

-> If we have identified we have good error blocking or error source removal strats.

SOFTWARE INSPECTION: (without executing the actual process. (like SRS, code etc))

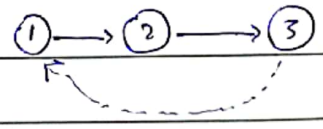
-> Advantage: It is kind of "Static Testing" which can be done before the Testing phase.

-> Drawback: Can't check Emergent Properties (NFR, performance etc)

-> Any Tangible thing can be "inspected."

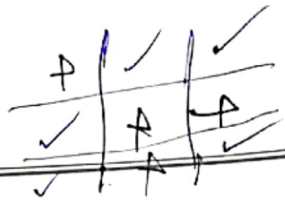
Basic 3 steps of Inspection:

1. Planning and Preparation: What to inspect and who will do it.
2. Conductance of Meeting
3. Feedback



-> Whenever a milestone is achieved Inspection can be performed (Product Plan, Test Plan, Manual etc).

Date _____



Degree of Formality:

→ Whoever is familiar with domain needs to be performing the inspection
→ Ideal case is a mix of Domain experts and other experts.

Inspection OR Collection:

Finding ~~exact~~ defects in the given Artifact

Fagan Inspection:

→ Process and Participants:

- L> Planning
- L> Overview Meeting
- L> Preparation : Individual who are going to review and identify possible defects.
- L> Inspection Meeting : Valid defects ko identify krlen jo aage feedback me bhejne hein
- L> Rework : Concerned bande ko feedback bhej dia
- L> Follow-up : Closing the inspection process by final validation.

Date

Applied on Calculation Based Systems
Insulin Pump

Formal Verification: (Before it, We need Formal Specification)
Only then formal verification can be checked.
→ Time Consuming (Least used) (Costly)
→ The most difficult among all the QA activities.
formal verification technique is defined in the chapter.

Formal Specifications. → On it inspection can be performed.
- DescriptiveSpec
- Operational Spec

→ Results generated by it needs an expert to elaborate.

Defect Containment: (Fault tolerance and failure containment)
- Example : Rollback on transaction failure
- limit defect to a local area instead of global (by some Duplication designed software)

Fault Containment Techniques: (If defect is inevitable)
→ Damage ~~occur~~ ko reduce karen

Date ________ CHAP 18

1. McCall
2. FURPS
3. ISO 9126
4. GQM
5. CMMI

→ Compare ho raken is kitna likhna he or ~~kis~~
→ which category of Quality model it falls in?

Part IV: Quantifiable Quality Improvement

Strategy for tool support:

Using existing tools

↳ Why use tools? Presentation tools, Analysis tools etc.

Data Sources: Testing + Other QA activities

Feedback doesn't have as such tools.

Quality Models category

① General Models (E-Commerce website) — 1, 2, 3

② Product-specific models (Insulin pump system) — 1, 2, 3

①

↳ 1. Overall Models.

Generally product me defect level kitna he by observing general historical models. (80% errors rise in 20% code) (Observation related), general models.

2. Segmented Models.

Jahan ~~sa~~ models ko segments me torna he.

3. Dynamic Models.

↳ then quality models ko apply krden

↳ Quality model attribute are applied on phases depending on how much effort was put in.

↳ Varies from phase to phase.

Scanned with CamScanner

Date

(2)

↳ 1. Semi-Customized Models,

→ Defect distribution on basis previously released specific product.

2. PSM (Observation Based). P = Predictive

Reasons why Quality Models can be effected based on historical models.

Software Testing Metrics

1. Process Metrics : process efficiency of SDLC.
2. Product Metrics : Quality of software product
3. Project :

Software Configuration Management

→ Preventive maintenance
→ Adaptive . . : New environment
→ Client Need change
→ Changes a thi hen

Jab bhi changes a thi hain to use record krna he, bdc if we meant Traceability (Back tracking). if we didn't achieve quality attributes we wanted.

Version Control

↳ Identify changes
↳ Analyze change
↳ If change feasible do it
↳ Document the change

Change Control , Baseline ke against compare kr the

Date

→ Configuration: Any characteristics of Hardware of Software

→ SCI (Software Config Item): Items of hardware/software/document that change.

Name: Muhammad Huzaifa Khan
Roll No: 20K-1708
Section: BSE-5B

Date

COMPARISON OF QUALITY MODELS:

Software Quality Models are a standard way of measuring a software product. New application software are planned and developed everyday which gave rise to the need of measuring that the product so built meets at least the expected standards. Some Quality are discussed below:

1. McCall:

McCall was the first quality model developed. The model is incorporated with many attributes, termed as software factors, which influence a software. It also defines a layout of the various ~~factors~~ aspects that define the software quality. The product's quality is defined in following manner.

- Product Operation
- Product Revision
- Product Transition

Product Operation is about factors ~~fact~~ like Correctness, Efficiency, Integrity, Reliability and Usability. Product Revision is about maintenance and testing of the software. It does so with quality factors of Maintainability, Flexibility and Testability. Product Transition defines the quality factors like Portability, Re-usability and Interoperability which allows software product to adapt to change of environments. in the new platform.

Date

2. FURPS:

FURPS is an acronym representing a model for classifying software quality attributes. F = Functionality, U = Usability, R = Reliability, P = Performance, S = Supportability. This model categories requirements into functional and Non-functional requirements. The functional requirements (F) are dependend on the expected input and output. The Non-functional requirements are (U, R, P, S) in which the Usability includes human factors, aesthetic, documentation of user material of training. Reliability determines frequency and severity of failure and time between failures. Supportability includes backup, requirement of design and implementation.

3. ISO 9126.

ISO 9126 is an international standard which was proposed to make sure 'quality of all software "intensive" products' such as a system Blood Insulin Injection System. This model determines the quality of software product through characteristics which are Functionality, Reliability, Usability, Efficiency, Maintainability and Portability. Each characteristic is also divided into sub-characteristics. The above mentioned characteristics can be divided into two categories.

1. Internal quality attributes.
2. External quality attributes.

Date

The internal quality attributes are those properties of the system which can be evaluated without execution. Whereas the external quality attributes are those that are evaluated by observing the system during execution.

4. GQM:

This method is a proven technique used for goal oriented measures. It consist of 3 elements:

1. Goal
2. Question
3. Metric

In this method, measurement is goal-oriented. Before this we need describe what the goals are. Then the goals can be transformed into questions ~~and metrics~~. Then questions are transformed into metrics. The GQM method consists of four phases.

1. Planning : Project plan is prepared by identifying requirements.
2. Definition : Goals, questions and metrics are defined.
3. Data Collection : As the name suggests data is collected
4. Interpretation : Answers to the questions are inputs of this phase, ~~whi~~ through which the goal's achievement is verified

Date _____

5. CMMI: (Capability Maturity Model Integration)

CMMI is a successor of CMM and is a more evolved model that incorporates best components of individual disciplines of CMM.

CMM is a reference model which needed to be evolve because CMM practices matured "specific" disciplines, which makes it difficult to integrate the said disciplines as per the requirements. That is why CMMI is used as it allows the integration of multiple disciplines as and when needed.

CMMI has two representations:

1. Staged Representation:

In this pre-defined set of process areas, to define improvement path, are used.

2. Continuous Representation:

It allows selection of "specific" process areas only.

CMMI main objectives are:

-> Fulfilling customes needs and expectations.
-> Value creation for investors.
-> Market growth
-> Improved quality of product
-> Enhanced reputation in industry.