Software Testing Interviewing Questions

Chapter 1 – Software Testing Basics

- PDCA (Plan, Do, Check, Act)
- Box Testing
  - Black Box – solely on requirements and specifications
  - White Box – internal paths, code structures, and implementation of software
  - Grey Box – Look at code to understand implementation, then test via black box
- Defect v.s. Failure
  - Defect found in-house
  - Failure – client found
- Defect Categories
  - Wrong – variance from requirement
  - Missing – requirement that isn't coded
  - Extra – requirement missing from specification
- Verification v.s. Validation
  - Verification – review without executing the process
  - Validation – checking product with actual execution
- Does increase in testing always improve project?
  - No, 20% of test plan is critical
- Defining Testing Policy
  - Definition – one unique definition for testing
  - How to achieve – testing committee/test plans?
  - Evaluate – how to derive metrics of defects (phase/programmer). How has testing added value
  - Standards – what is the standard of good enough
- Should testing be done after build/execution phases?
  - Traditional (requirement → Design → Code & Build → Testing → Maintenance)
  - Modern
    - Requirement → Design → Build & Execution
    - Test → Installation → Maintenance
- More Defects in Design or Coding
  - Design!! (requirements, then bad architecture & technical decisions)
  - 60% defects occur in design, 40% in coding
- End User Input for proper testing
  - Acceptance test plan prior to production
  - Requirement documents signed by client

    - Risky sections of project, obstacles to use
    - Proper data for testing
    - Test scenarios
- Latent and masked defects
    - Latent – existing defect that has not yet caused a failure due to conditions not met
    - Masked – existing defect that hasn't yet caused a failure due to another defect prevented code execution
- Defects not removed in initial stages can be up to 20 times more costly to fix in maintenance phase
- Workbench concept
    - Documenting how an activity must be performed (phases, steps, tasks)
    - Workbench
        - Input – input/entrance criteria
        - Execute – transform input to expected output
        - Check – assure output meets desired result
        - Production output – exit criteria of workbench
        - Rework – once fix implemented, go to execute
    - Phases

	Input	Execute	Check	Output
Requirement	Customer requirements	Write requirement doc	Addresses all needs?	Requirement Doc
Design	Requirement Doc	Write Technical Doc	Technically correct?	Technical Doc
Execution	Technical Doc	Implementation & coding		Implementation & Source code
Testing	Source code	Test cases		Test results
Deployment	Source code/test results			Production
Maintenance	Deployment results, Change Req		Regression Testing	New release

- Alpha(development) and Beta (Customer location)
- Defect leads to other defects based on their dependency/inheritance of object
- Usability testing
    - Give client prototype or mock-up, generally only the UI portion of project
- Strategies for Rollout to End Users
    - Pilot – limited deployment to a control set and is usually considered Beta Test
    - Gradual Implementation – deploy entire product while working on a finished product. Must maintain multiple versions

  - Phased Implementation – Rollout to all users incrementally as features are completed
  - Parallel Implementation – existing and new applications run simultaneously, but need duplicated hardware
- Requirement Traceability
  - Testing begins at requirement phase
  - Ensures proper coverage
  - Easily identify project debt as defects found
  - Identify risk with current state of software
- Pilot v.s. Beta Testing
  - Pilot product installed at live locations(Entering Real Data)
  - Beta testing not done with real data (Acceptance Test Plan)
- Rick Analysis During Testing

Features	Probability of Failure	Impact	Priority
Add a user	Low	Low	1
Check user prefs	Low	Low	1
Login User	Low	High	2
Add new invoice	High	High	4
Print invoice	Med	High	3
Concerns			
Maintainability	Low	Low	1
Security	High	High	4
Performance	High	Low	3
	Low/Med/High	**High/Low**	**1-4**

  - List Features/Concerns
  - Rate Defect probabilities
  - Impact Rating
  - Compute Risk(Probabilty * Impact)
  - Review
- Acceptance Plan Preparation
  - Requirement document
  - Input form the customer
  - Project Plan (project manager)
  - User Manual
- Environment reality and Test Phases
  - Latter Test Phases need more reality
- Inspections and Walkthroughs
  - Walkthrough is informal
  - Inspection is formal and documented
- Confirmation v.s. regression

  - Confirmation verifies fix works
  - Regression verifies nothing else broke as a result
- Coverage v.s. Techniques
  - Statement coverage – source code completed covered
  - Decision c overage – ensures Booleans covered
  - Path coverage – verifies that every possible route through code is covered
- Coverage Tools
  - Run simultaneously with testing and reports what was tested and results
- Configuration Management
  - Detailed recording and updating of infor for hardware and software components(requirements, design, test cases)
  - Knowing the state of the test system and expected outputs
- Baseline concept
- Different test Plan documents
  - Project Test Plan – resource utilization, testing strategies, estimation, risk, priorities
  - Acceptance Test Plan – verify user requirements are met
  - System Test Plan – main testing, functionality, load, performance, reliability
  - Integration Testing – tying software and hardware components together and verifying interoperability
  - Unit testing – developer level to check individual module in test

	Project TP	
Integration TP	Central TP	Acceptance TP
Unit TP		System TP

	Central				
Requirement		Acceptance			
Design			System		
Execution				Integration	Unit
Testing					
Deployment					
Maintenance					

- Inventories, Analysis & Design for Testing Projects, Calibration
  - Test objectives
    - Policies
    - Error Checking
    - Features
    - Speed
  - Inventory
    - List of things to be test for an objective
    - Add new Policy

        - Change/Add Address
        - Delete a Customer
    - Tracking Matrix
        - Inventory ← Mapping → Test Case
        - Mapping is Calibrating inventory to test cases
- Black Box or White Box Testing first?
    - Black Box first
    - White Box only after Design is fairly stable
    - Requirement Doc, Design Doc, Project Plan → Black Box → White Box
- Cohabiting Software
    - Software that can reside on same PC and affect project (and vice versa)
- Impact Rating used (Minor/Major/Critical)
- Test Log = data produced from test case(s)
- SDLC
    - Entry Criteria → Estimation Doc
        - SDLC
            - Exit Criteria
                - Acceptance Doc
    - WaterFall
        - Big Bang Waterfall
            - Requirement
            - Design
            - Build
            - Test
            - Deliver
        - Phased Waterfall
            - Project divided and developed in parallel, gluing things together at the end. Problems arise with lack of coordination
    - Iterative
        - Incremental – work divided into chunks like the Phase Waterfall, but one team can work on one OR many chunks
        - Spiral – Prototype and refining cycle, waterfall is repeated for each cycle
    - Evolutionary
        - Produce minimal features using process, then evolve software with updated customer requirements
    - V-model
        - Emphasizes importance of early testing so that each Stage is refined immediately after introduction
- Testing phases

    - Unit – Component Design Features fully covered, usually done by a developer by creating fake components that mimic what finished components due in order to test the logic
    - Integration – tests all components and their interoperability/relationships
    - System – System Specification driven and tests the entire system as a whole.
        - Performance
        - Volume
        - Stress
        - Documentation
        - Robustness
    - Acceptance testing
        - Checks system against requirements, done by customer NOT developer
- Who does testing?
    - Isolated test Team
    - Outsource
    - Inside
    - Developers as testers
    - QA/QC Team

Chapter 2 – Testing Techniques

- Equivalence Partitionign
    - Eliminating duplicate test cases for maximum efficiency
    - Test <, >, and =
    - All redundancies should be eliminated
- States and Transitions
    - State = result of previous input
    - Transition = actions that cause change in state
    - Each state or transition can generate test cases
    - Combine state and transition for complete coverage since using only one in isolation can leave gaps in coverage
- Random/Monkey testing
    - Not realistic
    - Redundant
    - Time spent analyzing results
    - Cannot recreate the test if data isn't recorded
- Negative/Positive testings
    - Negative – invalid input = errors
    - Positive – valid input with some expectation on output
- Exploratory Testing (Adhoc)
    - Unplanned, unstructured, impulsive or intuitive journey with intent to find bugs

  - Simultaneous learning, test design, and test execution
  - Any testing doen tot eh extent that the tester proactively controls the design of the tests as those tests are performed and uses info gained while testing to design better tests.
  - NOT random
  - Learning → Design → Execution
- Semi-Random
  - Random testing that removes duplicates
  - Random test case → Equivalent Partitionign → Semi-Random Test Case
- Orthogonal Arrays / Pair-wise defect
  - 2D Table where if any 2 columns chosen, all values will appear
- Decision Tables
  - Lists all inputs (rows) and outputs(columns)
- Severity Ratings
  - Catastrophic –
  - Severe
  - Moderate
  - Mild

Chapter 3: Software Process

- Software Process

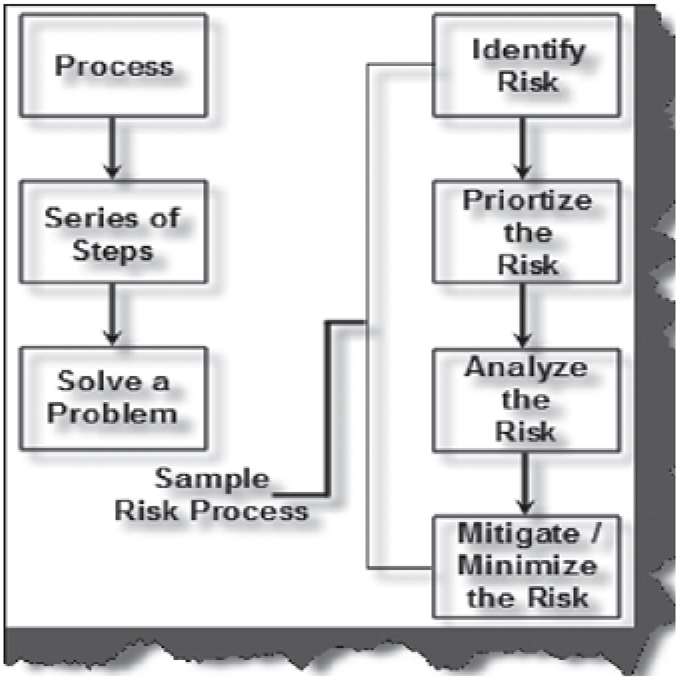


FIGURE 66 Software process

- Different cost elements involved in implementing a process
  - Salary, Consultant, Training Costs, Tools
- Model
  - Best practices followed in an industry to solve issues an problems
- Maturity Level – level of performance expected from an organization
- Process areas in CMMI

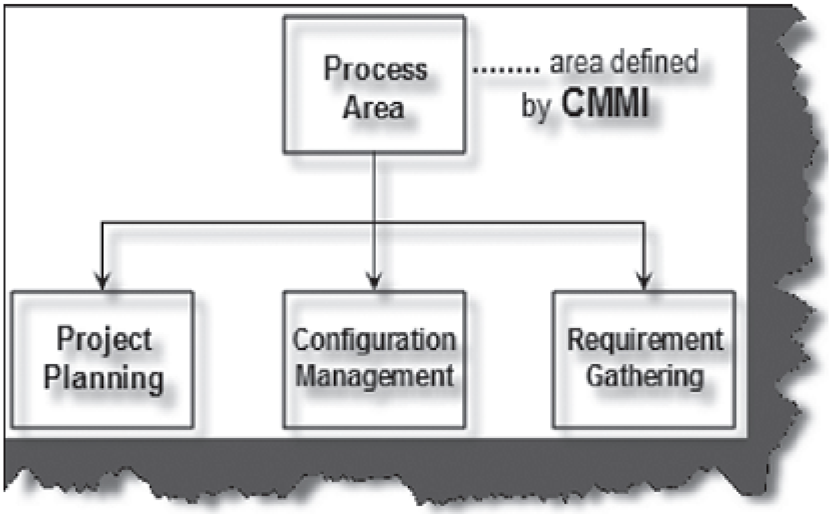


FIGURE 70 Process areas in action

- Tailoring – where process is done in a custom way, NOT bypassed

Chapter 4 - CMMI

- CMMI (Capability Maturity Model Integration): Process improvement approach that provides companies with the essential elements of an effective process.
    - Systems engineering
    - Software engineering
    - Integrated Product and Process Development (IPPD)
    - Software acquisition
- Implementation – performing a task in a process area
- Institutionalization – output of performing a process again and again
- Models
    - Staged (Maturity Level)

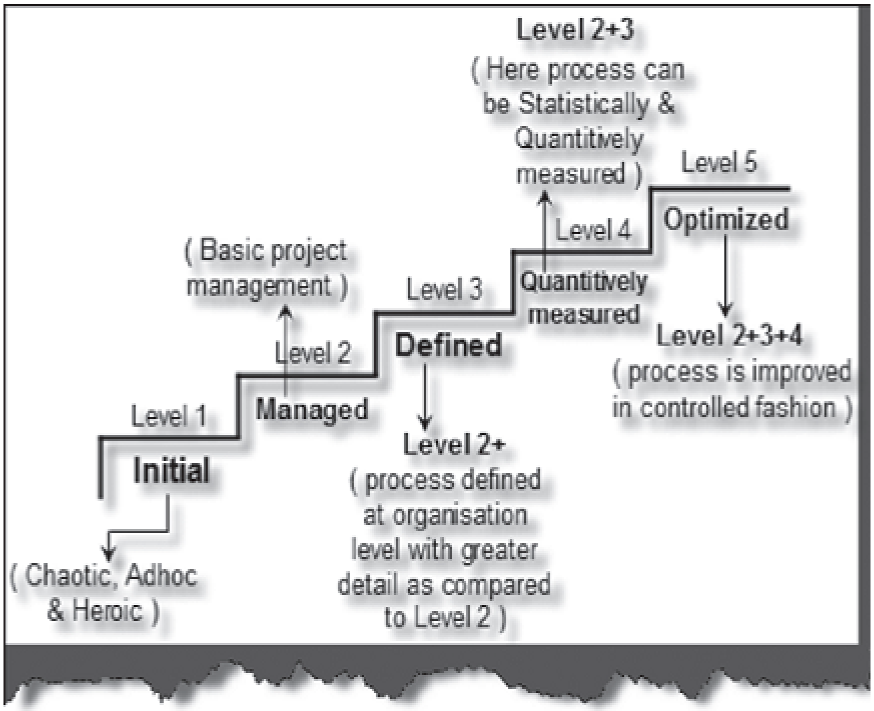

Level 2+3
(Here process can be Statistically & Quantitively measured)
Level 5
Optimized
Level 4
Quantitively measured
Level 2+3+4
(process is improved in controlled fashion)
(Basic project management)
Level 3
Defined
Level 2
Managed
Level 2+
(process defined at organisation level with greater detail as compared to Level 2)
Level 1
Initial
(Chaotic, Adhoc & Heroic)

FIGURE 79 Maturity level in staged model

    - Continuous

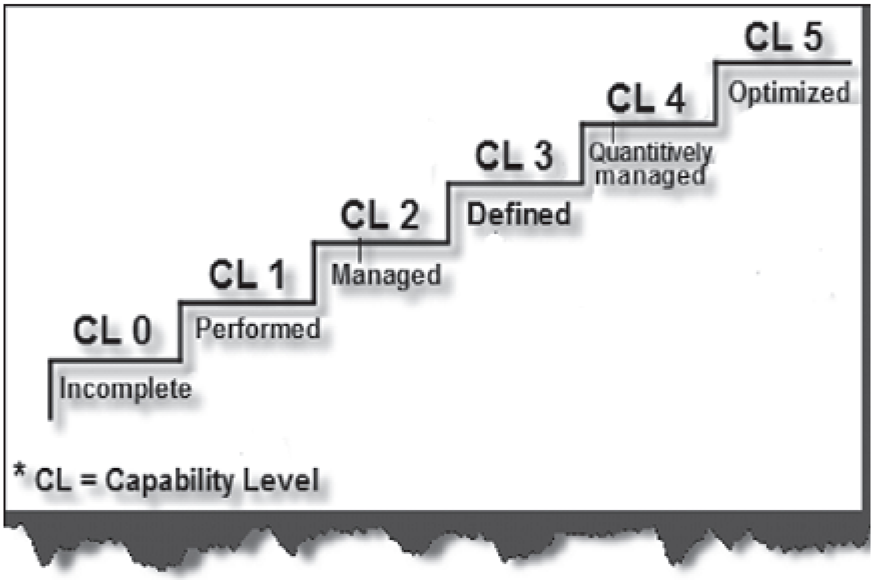


FIGURE 80 Capability levels in a continuous model

- CMMI (25 processes)
  - Process management
  - Project Management
  - Engineering
  - Support

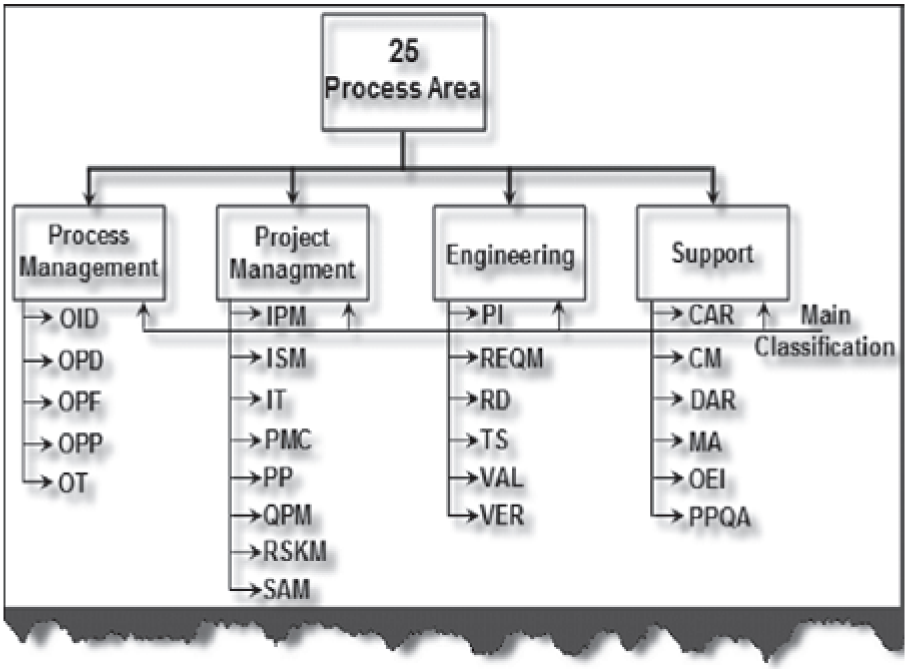
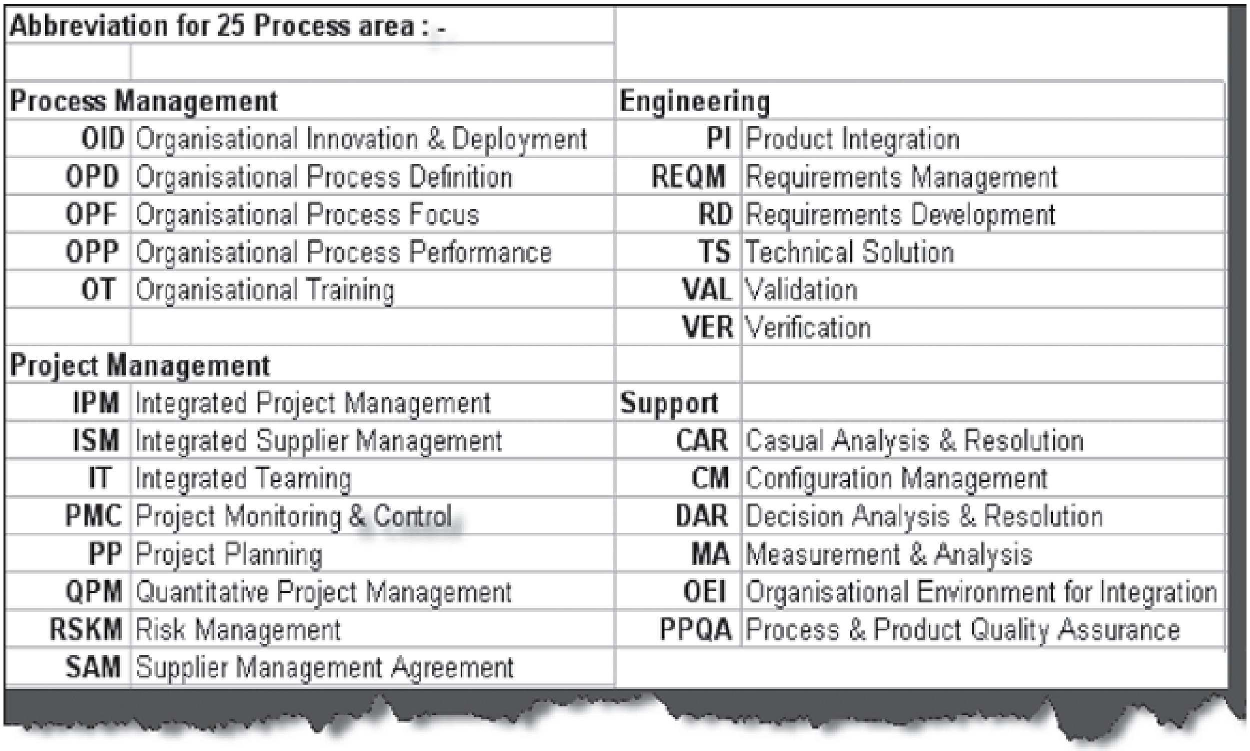


FIGURE 81 The 25 Process areas



Abbreviation for 25 Process area : -			
Process Management		Engineering	
OID	Organisational Innovation & Deployment	PI	Product Integration
OPD	Organisational Process Definition	REQM	Requirements Management
OPF	Organisational Process Focus	RD	Requirements Development
OPP	Organisational Process Performance	TS	Technical Solution
OT	Organisational Training	VAL	Validation
		VER	Verification
Project Management			
IPM	Integrated Project Management	Support	
ISM	Integrated Supplier Management	CAR	Casual Analysis & Resolution
IT	Integrated Teaming	CM	Configuration Management
PMC	Project Monitoring & Control	DAR	Decision Analysis & Resolution
PP	Project Planning	MA	Measurement & Analysis
QPM	Quantitative Project Management	OEI	Organisational Environment for Integration
RSKM	Risk Management	PPQA	Process & Product Quality Assurance
SAM	Supplier Management Agreement		

FIGURE 82 Abbreviations of all the process areas

Chapter 5 – Six Sigma

- Six Sigma – statistical measurement of variation in a process
    - 3.4 Defect per Million Opportunities
    - DMAIC – Improves, DMADV – Defines

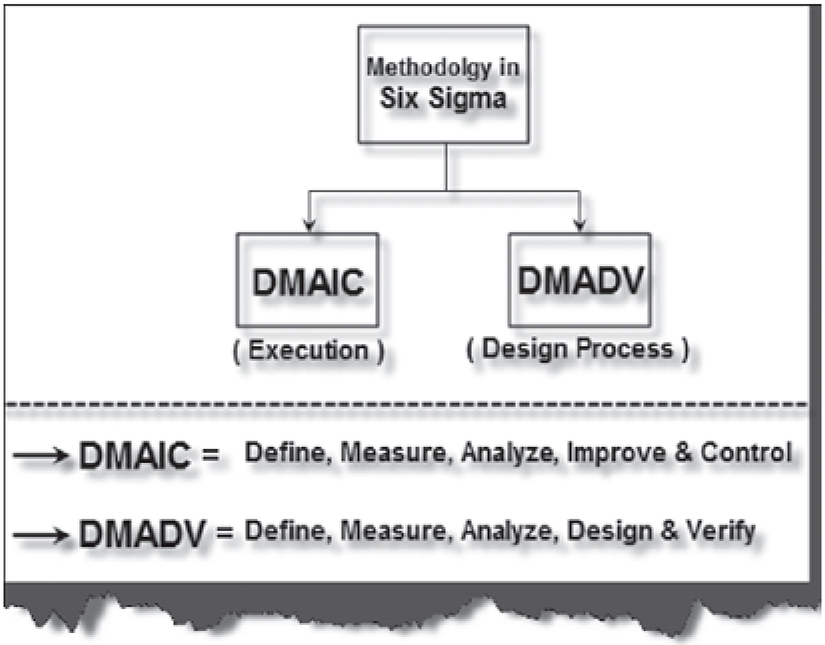

Methodolgy in Six Sigma
DMAIC
(Execution)
DMADV
(Design Process)
DMAIC = Define, Measure, Analyze, Improve & Control
DMADV = Define, Measure, Analyze, Design & Verify

FIGURE 97 Methodology in Six Sigma

- 5 key players
    - Executive – deciders, funders
    - Champions – Sr. Mgmnt works with business mgrs
    - Master black belts – technical masters, coach/mentor
    - Black belts – team leaders – discover variations and IMPLEMENT SS
    - Green belts – part-time assistants

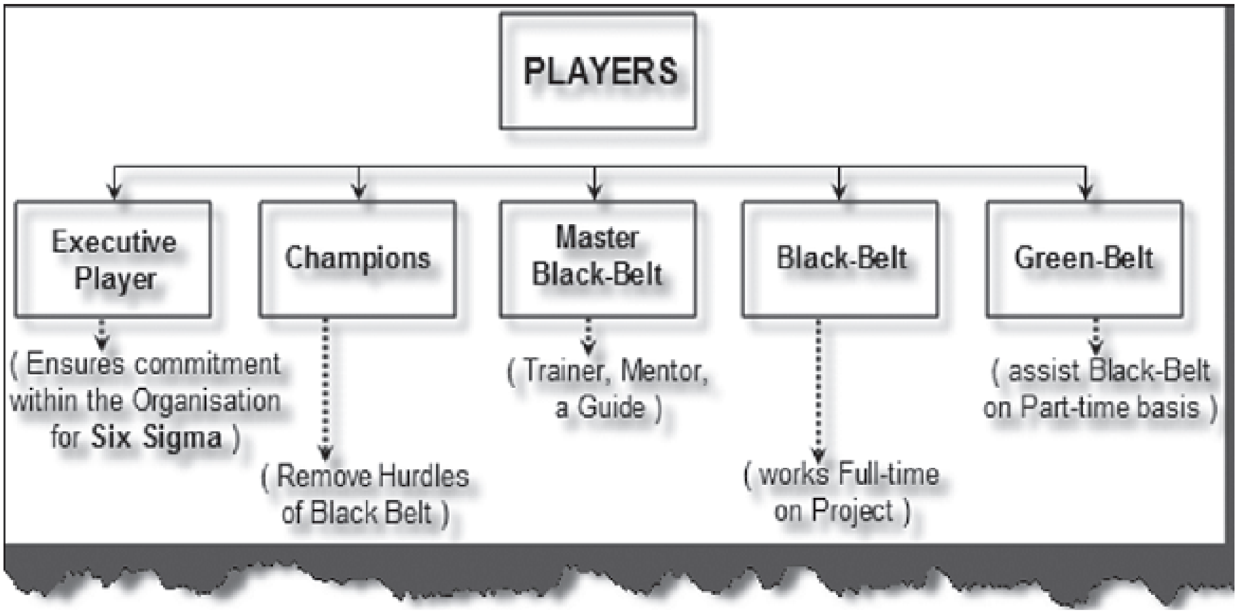

PLAYERS
Executive Player
(Ensures commitment within the Organisation for Six Sigma)
Champions
(Remove Hurdles of Black Belt)
Master Black-Belt
(Trainer, Mentor, a Guide)
Black-Belt
(works Full-time on Project)
Green-Belt
(assist Black-Belt on Part-time basis)

FIGURE 99 Six Sigma key players

- Variation Types in Six Sigma
    - Mean - Averaging
    - Median – Mid-Point in the data
    - Range – spread of values: high/low
    - Mode – most frequently appearing value

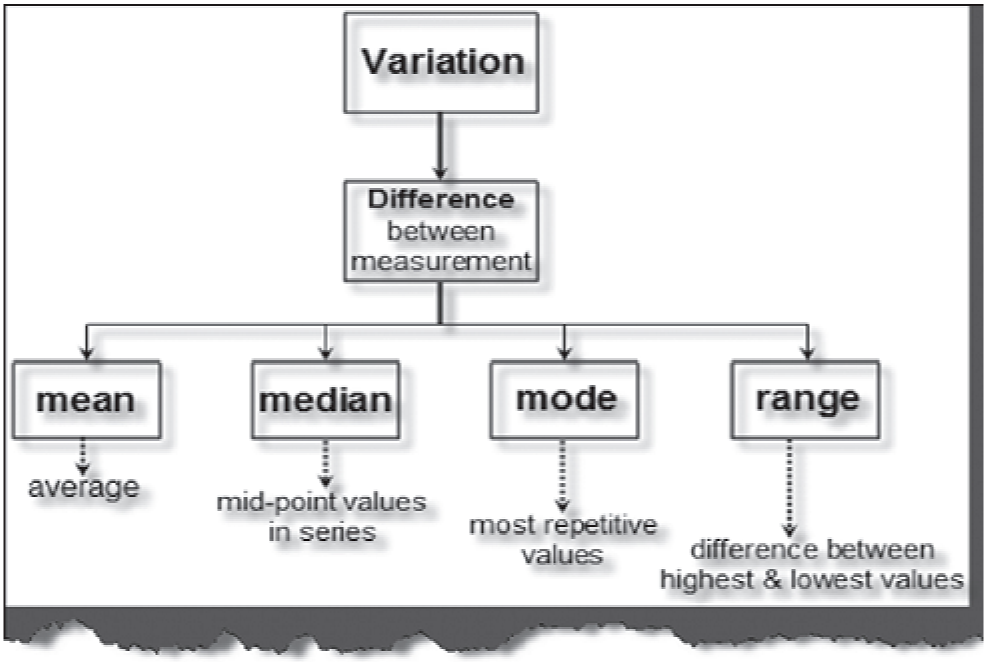
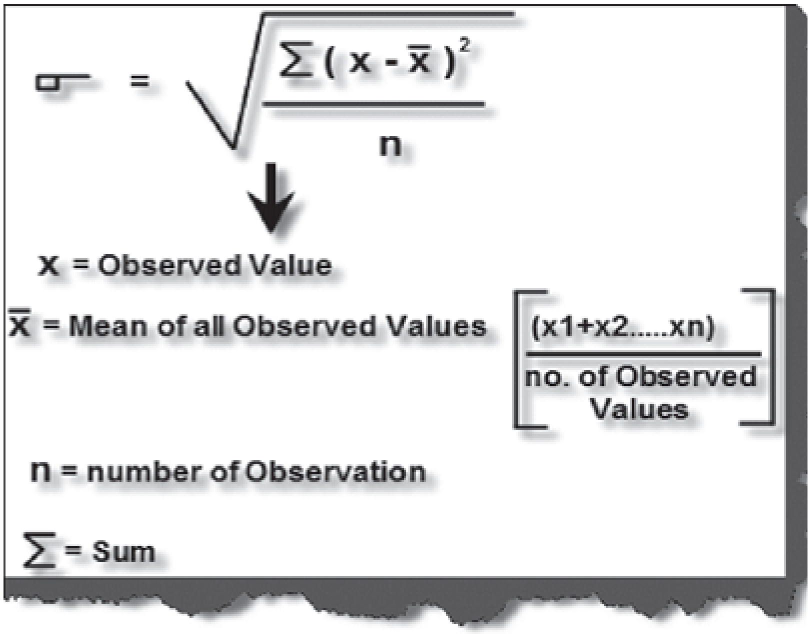

Variation
Difference between measurement
mean
median
mode
range
average
mid-point values in series
most repetitive values
difference between highest & lowest values

FIGURE 100 Different variations in Six Sigma

- Standard Deviation



$$\sigma = \sqrt{\frac{\sum(x-\bar{x})^2}{n}}$$

x = Observed Value

$\bar{x}$ = Mean of all Observed Values $\left[\frac{(x1+x2.....xn)}{\text{no. of Observed Values}}\right]$

n = number of Observation

$\sum$ = Sum

FIGURE 105 **Standard deviation formula**

- Standard Deviation Calculations
  - Get Mean
  - SUM(x – xMean)^2 for X1, X2, X3
  - Divide by number of Observations
  - Square Root
- Fish Bone/Ishikawa Diagram

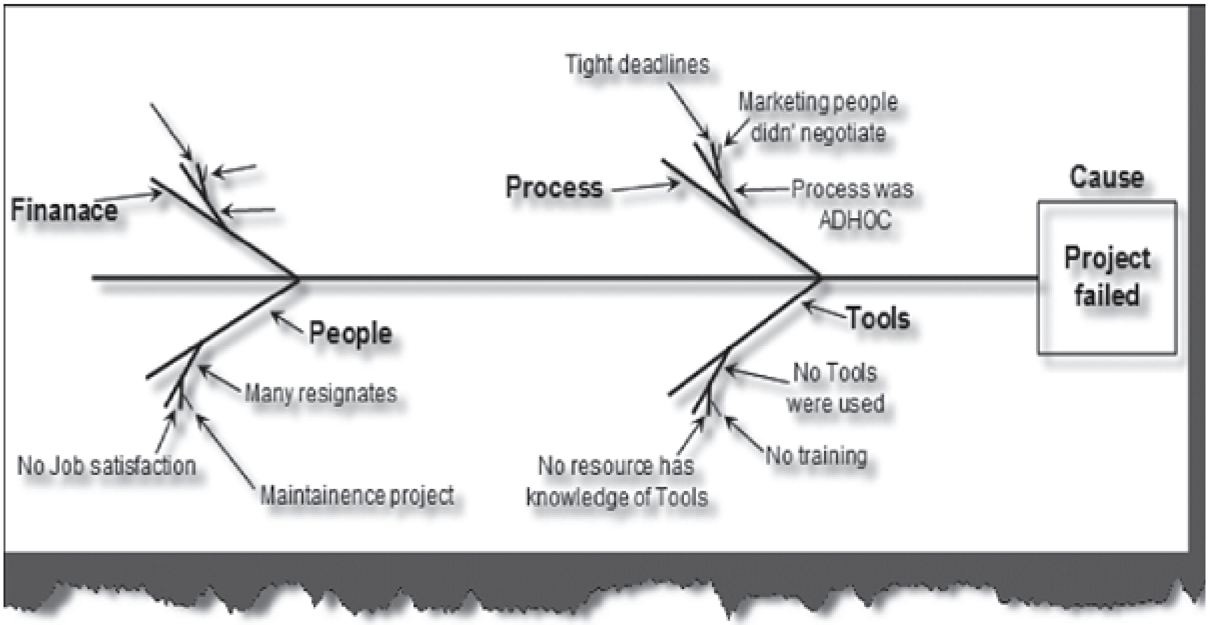


FIGURE 110 **Fish bone/Ishikawa diagram**

Chapter 6 - Metrics

- Measures & Metrics

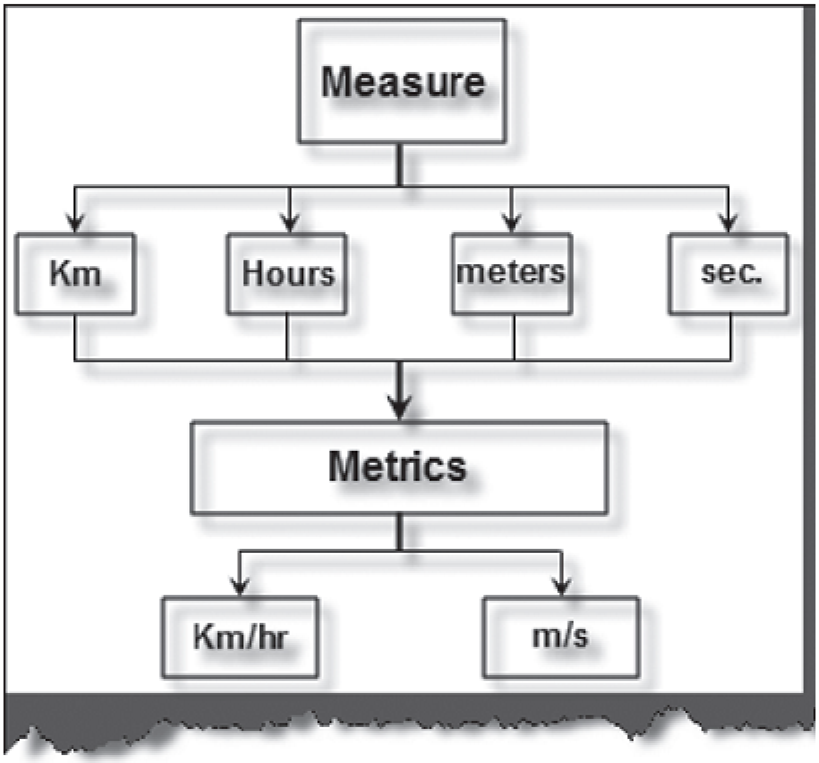
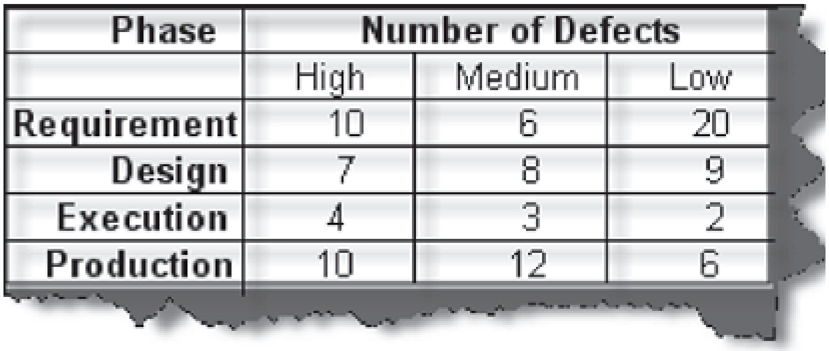


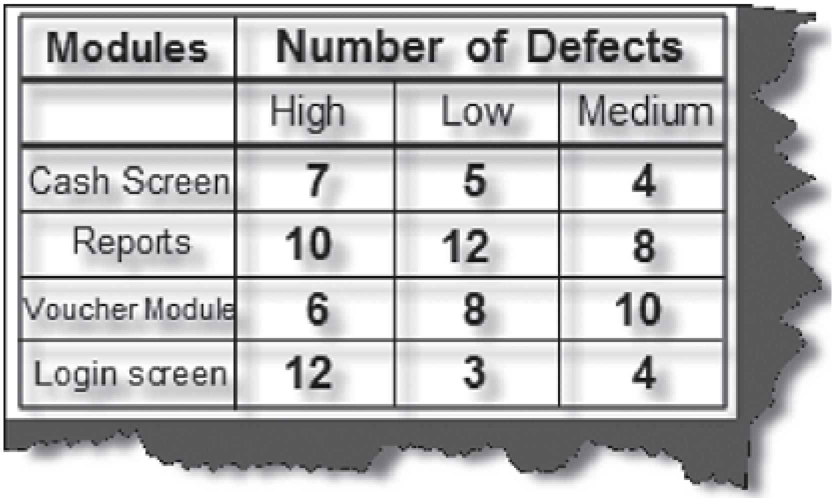
FIGURE 111 **Measure and metrics**

- Number of Defects are measured by WEIGHTING, usually by 3 or 5 categories



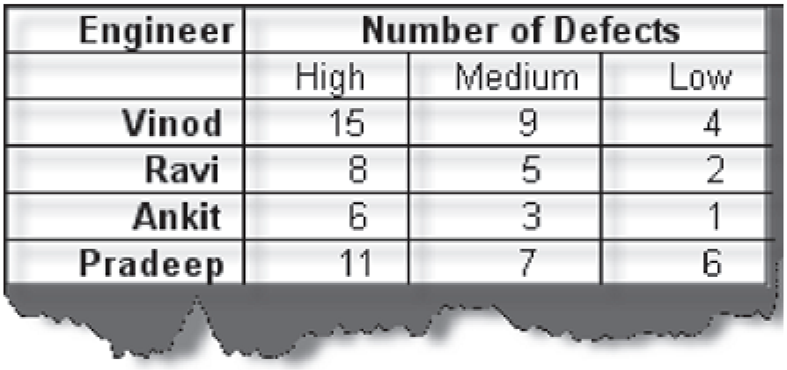
Phase	Number of Defects		
	High	Medium	Low
Requirement	10	6	20
Design	7	8	9
Execution	4	3	2
Production	10	12	6

FIGURE 112 **Number of defects phase-wise**



Modules	Number of Defects		
	High	Low	Medium
Cash Screen	7	5	4
Reports	10	12	8
Voucher Module	6	8	10
Login screen	12	3	4

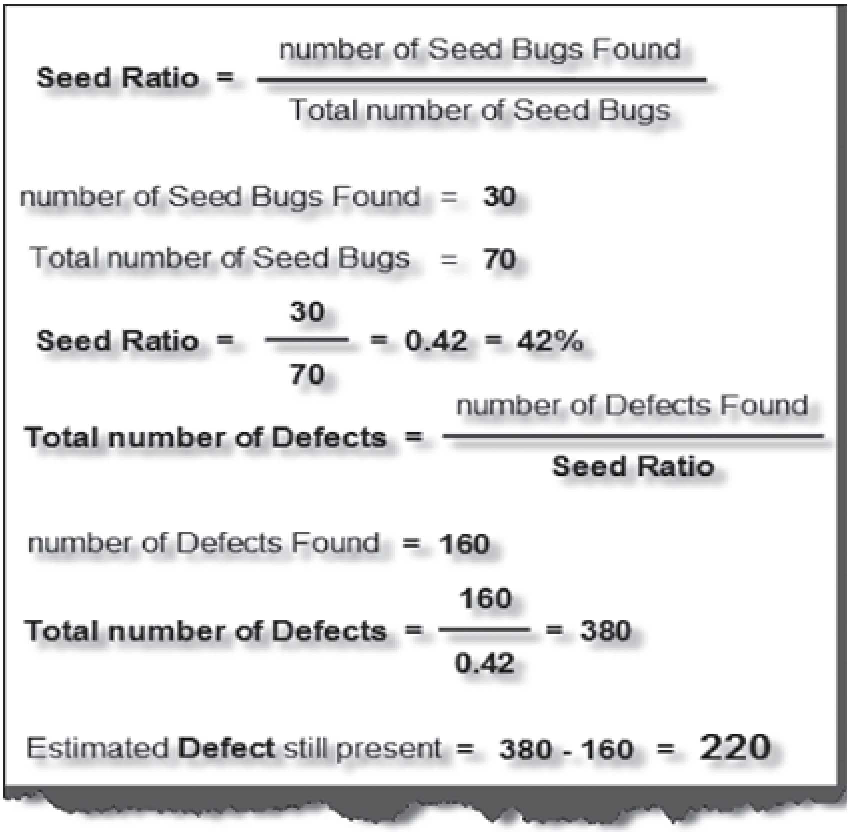
FIGURE 113 Number of defects module-wise



Engineer	Number of Defects		
	High	Medium	Low
Vinod	15	9	4
Ravi	8	5	2
Ankit	6	3	1
Pradeep	11	7	6

FIGURE 114 Number of defects

- Production Bugs – good measure of performance, but latency involved
- Defect seeding
  - How many found
  - How many not found
  - How many unseeded defects found



$$\text{Seed Ratio} = \frac{\text{number of Seed Bugs Found}}{\text{Total number of Seed Bugs}}$$

number of Seed Bugs Found = **30**

Total number of Seed Bugs = **70**

$$\text{Seed Ratio} = \frac{30}{70} = 0.42 = 42\%$$

$$\text{Total number of Defects} = \frac{\text{number of Defects Found}}{\text{Seed Ratio}}$$

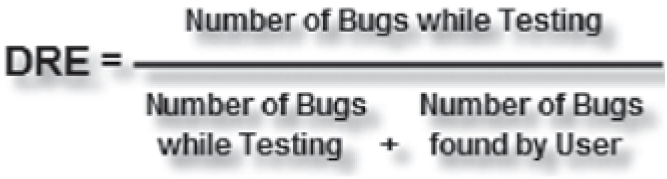
number of Defects Found = **160**

$$\text{Total number of Defects} = \frac{160}{0.42} = 380$$

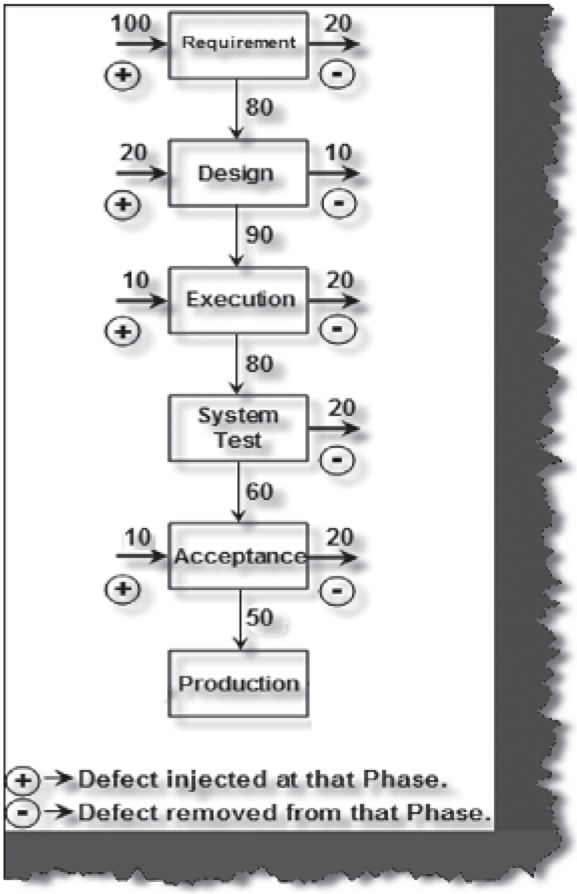
Estimated **Defect** still present = **380 - 160** = **220**

FIGURE 116 Seed calculation

- DRE – Defect Removal Efficiency
  - Take into account severity and history of customer found defects
  - Can be used on a ‘per phase’ basis to analyze efficiency of any given phase



$$\text{DRE} = \frac{\text{Number of Bugs while Testing}}{\text{Number of Bugs while Testing} + \text{Number of Bugs found by User}}$$



JRE 118 Defect injected and removed per phase

- Measure Test Effectiveness

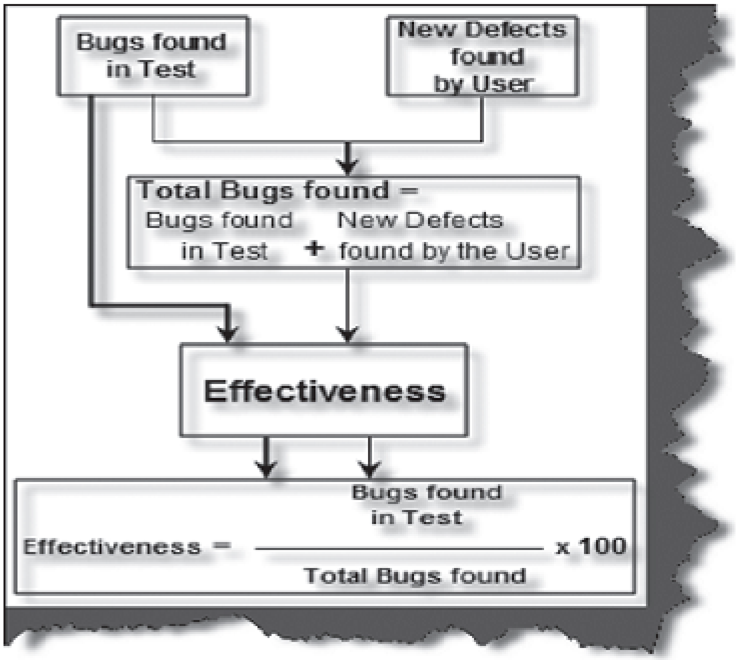
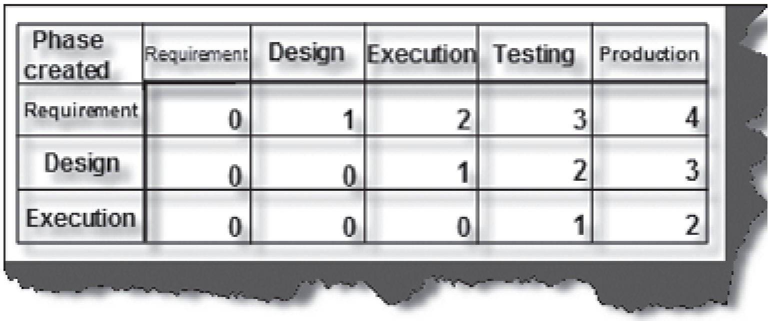

Bugs found in Test
New Defects found by User
Total Bugs found = Bugs found in Test + New Defects found by the User
Effectiveness
Effectiveness = Bugs found in Test / Total Bugs found x 100

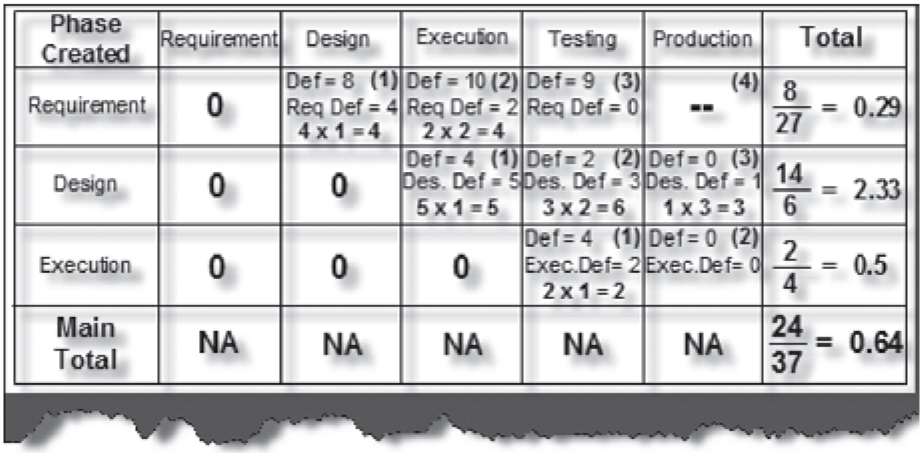
FIGURE 122 Measure test effectiveness

- Defect Age and Defect Spoilage



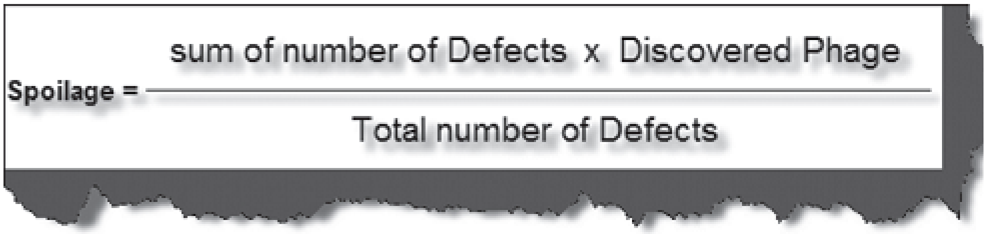
Phase created	Requirement	Design	Execution	Testing	Production
Requirement	0	1	2	3	4
Design	0	0	1	2	3
Execution	0	0	0	1	2

FIGURE 123 Scale of defect age



Phase Created	Requirement	Design	Execution	Testing	Production	Total
Requirement	0	Def = 8 (1) Req Def = 4 4 x 1 = 4	Def = 10 (2) Req Def = 2 2 x 2 = 4	Def = 9 (3) Req Def = 0	-- (4)	$\frac{8}{27}$ = 0.29
Design	0	0	Def = 4 (1) Des. Def = 5 5 x 1 = 5	Def = 2 (2) Des. Def = 3 3 x 2 = 6	Def = 0 (3) Des. Def = 1 1 x 3 = 3	$\frac{14}{6}$ = 2.33
Execution	0	0	0	Def = 4 (1) Exec.Def= 2 2 x 1 = 2	Def = 0 (2) Exec.Def= 0	$\frac{2}{4}$ = 0.5
Main Total	NA	NA	NA	NA	NA	$\frac{24}{37}$ = 0.64

FIGURE124 Defect spoilage



$$\text{Spoilage} = \frac{\text{sum of number of Defects x Discovered Phage}}{\text{Total number of Defects}}$$

FIGURE 125 Spoilage formula

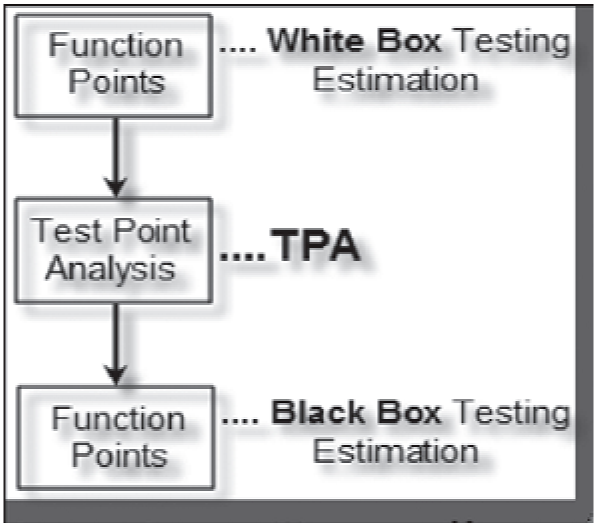
Chapter 7 – Automated Testing

- Does Automated replace Manual Testing
  - NO
    - Unstable Software
    - Blue Moon Scripts
    - Code and Document review
  - YES
    - Repetitive Tasks
    - Regression
    - Smoke, Load, Performance
    - Code coverage
    - White box testing
- Tools Employed
  - ATGX – Proprietary framework
  - QuickTest wVB – Record/Playback, Scripted, and Manual Test tool
  - TFS - Team Foundation Server
    - Mainly Manual Testing
- Load Testing for Websites
  - Use capture tool for request/response times
  - Use timeouts for simulated latency where needed
  - Create ‘virtual users’ to simulate simultaneous usage
- Data-Driven testing – data read fromdatabase/excel or csv
- Table-Driven testing –

Chapter 8 – Testing Estimation

- Different ways of doing black box testing
  - Top down according to budget
  - WBS (Work Breakdown Structure)
  - Guess and gut feeling
  - Early project data
  - TPA (Test Point Analysis)

- TPA (Test point analysis)
    - Black box testing ONLY
    - IFPUG (International Function Point User Group)
        - Break system down into main sections
        - Break sections into functions
        - Analyze and time functions
        - Add times


Function Points
.... White Box Testing Estimation
Test Point Analysis
....TPA
Function Points
.... Black Box Testing Estimation

- FPA
    - Define boundaries
        - Internal
        - External
            - 3rd party applications/interfaces
            - Can you make changes to the component, then yes
- The Elementary Process OR Static/Dynamic Elementary Process
    - Smallest unit of activity meaningful to the end user
    - EP must be self-contained and leave the application in a consistent state.
- Dynamic EP - Moves data from an internal application boundary to an external boundary, or vice-versa
    - Input data screen to application
    - Transactions exported in export files in XML , etc
    - Display reports which can come from an external application boundary and an internal application boundary
- Static EP – internal to the application
    - Maintaining customer data
- FPA Function Points
    - ILF – Internal Logical Files
        - Logically related data from user's POV

    - Reside in internal boundary
- EIF – External Interface Files
    - Logically related from the user's POV
    - Reside in the external boundary
    - Used only for reference purpose, not maintained by internal applications; rather external applications
- RET – Record Element Type
    - Sub-group element data of ILF or EIF
    - No sub-group of ILF then count the ILF itself as one RET
    - Group or RETs within ILF or logically related. Most likely with a parent-child relationship.
- DET - Data Element Types
    - Each DET should be user recognizable.
    - AutoIncrement ID, NOT DET
    - Non-recursive fields in ILF
    - Supplier ID doesn't count as DET, but foreign keys (relationship keys) do.
- FTR – File Type Reference
    - File or data referenced by a transaction
        - FTR should be ILF or EIF. Count each read during the process
        - If EP is maintained as an ILF then count that as an FTR.
        - By Default, one FTR in any EP
- EI – External Input
    - Dynamic elementary processes ni which data is received from the extranl application boundary.
    - UI to Internal Application
    - EIs may maintain the ILF of the application, but it's not a compulsory rule.
    - User screens usually EI as it passes data from user to internal application
    - Import batch screen, also EI
- External Inquiry (EQ)
    - Dynamic elementary process in which result data is retrieved from one or more ILF or EIF.
-