UNIT-IV

TestCaseDeisgn:

--Basicallytest designis theact ofcreatingandwritingtest suitesfortestinga software.
--Testanalysisandidentifyingtestconditions givesusagenericideafortestingwhichcoversquitealargerangeof possibilities.
--Butwhen wecome tomakeatest caseweneedtobe veryspecific.In factnow weneed theexactanddetailed specific input.
--Butjusthavingsomevaluestoinputtothesystemisnotatest,ifyoudon’tknowwhatthesystemis supposed to do with the inputs, you will not be able to tell that whether your test has passed orfailed.
--TestcasescanbedocumentedasdescribedintheIEEE829StandardforTestDocumentation.
--TheIEEE829StandardforTestDocumentationconsistsofdifferentdocumentscovering:*Testplan*,*TestDesignSpecification*,*TestCaseSpecification*,*TestProcedureSpecification*,*TestItemTransmittalReport*,*TestLog*,*TestIncidentalReport*,*TestSummaryReport.*

Whydoesatestsuitegrow?

--Theremaybeunnecessarytestcasesinthetestsuiteincludingbothobsoleteandredundanttestcases.
--Forexampleachangein aprogramcauses atest casetobecomeobsolete.
--Atestcase is redundantif other test cases in test suiteprovide thesamecoverageof theprogram.
--Thusduetoobsoleteandredundanttestcases,thesizeofatestsuitecontinuestogrowunnecessarily.
--Testengineersmeasurestheextenttowhichacriterionissatisfiedintermsof*Coverage*.
--Coverageismeasuredin termsoftherequirements thatareimposed.
--PartialCoverageisdefinedasthepercentageof requirementsthataresatisfied.
--CoverageCriteriais used asa stoppingpointto decide whenaprogramis sufficientlytested.

Minimizingthetest suiteand itsbenefits:

A test suite can sometimes grow to an extent that it is nearly impossible to execute. In thiscase, it becomes necessary to minimize the test cases such that they are executed for maximumcoverage.
Thereasons whyminimization is important are:
-->Releasedateoftheproductisnear.
-->Limitedstafftoexecuteallthetestcases.
-->Limitedtestequipments orunavailabilityof testingtools

MinimizingthetestsuitehasthefollowingBenefits:
--Redundanttestcaseswillbeeliminated.
--Lower costs byreducingatest suite to aminimal subset.

--A reduction in the size of test suite decreases both the overhead of maintaining the test suite andthe number of test cases that must be rerun after changes are made to the software, thereby reducingthecost of regression testing.

TestSuitePrioritization:

Thepurposeofprioritizationistoreducethenumberoftestcasesbasedonsomecriteria,whileaimingto selectthemost appropriatetests.Thedifferentpriorities are:

Priority1: *The test cases that must be executed* otherwise there may be worse consequences for thereleaseof theproduct.

Priority2:*Thetestcases maybeexecuted, if timepermits.*

Priority3:*Thetestcase isnotimportantpriortothecurrentrelease*.Itmaybetestedshortlyafterthereleaseofthecurrent version of thesoftware.

Priority4:*Thetest case isneverimportant asitsimpact isnearlynegligible.*

Typesoftestcaseprioritization:

GeneralTestCasePrioritization:

In this prioritization, we prioritize the test cases that will be useful over a succession ofsubsequentmodifiedversionsof P(Programor Product),withoutanyknowledgeofthemodifiedversions.

Version-SpecificTestCasePrioritization:

Here, weprioritizethetestcasessuchthattheywillbeusefulonaspecificversionP'ofP.

PrioritizationTechniques:

Coverage Based Test

PrioritizationRisk Based

PrioritizationPrioritization using

Relevant

SlicesPrioritizationbasedonRequirem

ents

CoverageBasedTestPrioritization:

Total Statement based Prioritization: This prioritization orders the test cases based on the totalnumber of statements covered. It counts the number of statements covered by test cases and ordersthem in descending order. For example, if T1 covers 5 statements, T2 covers 7 statements and T3covers14 statements, then the order of prioritization isT3,T3,T1.

AdditionalStatementCoveragePrioritization:

Statement	Statement Coverage		
	Test case 1	Test case 2	Test case 3
1	X	X	X
2	X	X	X
3		X	X
4			X
5			
6		X	
7	X	X	
8	X	X	
9	X	X	

This technique iteratively selects a test case T1, that yields the greatest statement coverage, thenselectsa test casewhichcovers astatement uncovered byT1.

For example, if we consider above table, according to total statement coverage criteria, the order is2,1,3.

But additional statement coverage selects test case 2 first and next, it selects test case 3, as it covers statement 4 which has not been covered by test case 2.

So, the order is 2,3,1.

Total Branch coverage Prioritization: In this prioritization, the criteria to order is to considercondition branches in a program instead of statements. Thus it is the coverage of each possibleoutcome of a condition ina predicate. The test case which will cover maximum branch outcomeswillbeorderedfirst. For example, inthefollowingdiagram theorderwillbetestcases 1,2,3.

Branch Statements	Branch Coverage		
	Test case 1	Test case 2	Test case 3
Entry to while	X	X	X
2-true	X	X	X
2-false	X		
3-true		X	
3-false	X		

Total Fault-Exposing-Potential (FEP) Prioritization: Statement and branch coverageprioritization ignore a fact about test cases and faults. Thus the ability of a test case to uncover afaultiscalledthe***faultexposingpotential***.Toobtainanapproximationofatestcase, anapproachwasadopted usingmanual analysis.Theapproachis:

-->Given program P and test suite T, First create a set of mutants N = {n1, n2,…..nm} for P, notingwhichstatement s_j in Pcontains each mutant.

-->Next,foreachtestcaset_iɛT,executeeachmutantversion$n_k$ofPont_i ,notingwhether$t_i$killsthat mutant.

-->CalculatetheFEP(s,t)astheratio:Mutantsof$s_j$killed/Totalnumberofmutantsofs_j.

ToperformtotalFEPprioritization,giventheseFEP(s,t)values,calculatean*awardvalue*foreachtest case tiɛT, by summing the FEP(sj, ti) values for all statements sj in P. Given these awardvalues, weprioritizetest cases bysortingthem inorder ofdescendingaward value.

Statement	FEP(s,t) values		
	Test case 1	Test case 2	Test case 3
1	0.5	0.5	0.3
2	0.4	0.5	0.4
3		0.01	0.4
4		1.3	
5			
6	0.3		
7	0.6		0.1
8		0.8	0.2
9			0.6
Award values	1.8	3.11	2.0

RiskBasedPrioritization:

Risk Based technique is to prioritize the test cases based on some potential problems which mayoccurduringthe project.Thetwocomponents of risk based technique are:

*Probabilityofoccurrence/faultlikelihood:*Itindicatestheprobabilityofoccurrenceofaproblem.

Severity of impact / failure impact: If the problem has occurred, how much impact does it have onthesoftware?

Ariskanalysistableconsistsofthefollowingcolumns:ProblemID,PotentialProblem,Uncertainty factor, Severity of impact,RiskExposure. For example, the problems inthe giventablecan beprioritized in theorder ofP5, P4, P2,P3 and P1 based onthe risk exposure.

Problem ID	Potential Problem	Uncertainty Factor	Risk Impact	Risk Exposure
P_1	Specification ambiguity	2	3	6
P_2	Interface problems	5	6	30
P_3	File corruption	6	4	24
P_4	Databases not synchronized	8	7	56
P_5	Unavailability of modules for integration	9	10	90

PrioritizationusingRelevantSlices:

Execution Slice: The set of statements executed under a test case is called *Execution Slice* of theprogram.

Example:

```
S1:     read(a,b);
S2:     result;
S2:     if(a>=100||b>=200)
S3:
        result=a+b;S4:
        else
S5:             result=a-
b;S6:   returnresult;
```

Test Cases	a	b	result
1	150	250	400
2	50	200	250
3	200	50	250
4	20	20	0

DynamicSlice: Theset ofstatementsexecutedunderatestcaseandhave aneffectontheprogramoutput under that test case is called *Dynamic Slice* of the program with respect to the outputvariables.

Example:

```
S1:     read(a,b);
```

```
S2:     sum=0,I;
S2:     if(a==0)
S3:             printa;
S4:     else
S5:             printb;
S6:     else
ifS7:           {
S8:                     sum=a+b;
S9:                     I=50;
S10:                    Print(I);
S11:            }
S12:    return(sum);
```

Relevant Slice: The set of statements that were executed under a test case and did not affect theoutput, but have potential to affect the output produced by a test case is known as *Relevant Slice* ofthe program. For example, in the above code, statements S2 , S4 have the potential to affect theoutput,if modified.

PrioritizationbasedonRequirements:

HemaSrikanth et al. [136] have applied requirement engineering approach for prioritizingthe system test cases. It is known as PORT (Prioritization of Requirements for Test). They haveconsideredthefourfactors foranalyzingand measuringthecriticalityof requirements:

--*Customer-Assignedpriorityofrequirements*
--*RequirementVolatility*
--*Developer*-perceivedimplementationcomplexity
--*Faultpronenessofrequirements*

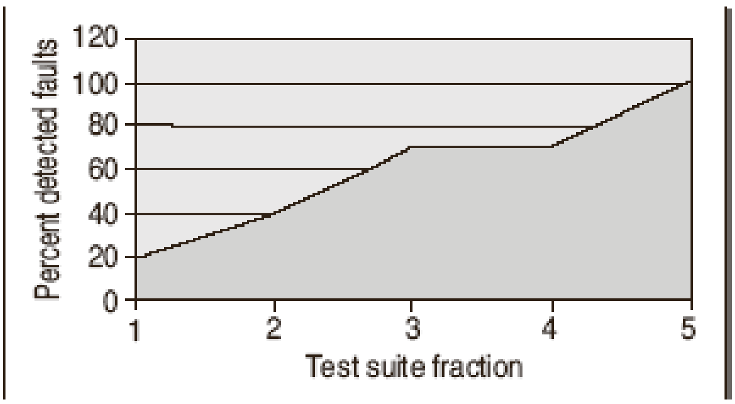
MeasuringtheeffectivenessofaPrioritizedTestSuite:

Elbaum et al. [133,134] developed APFD (Average percentage of faults detected) metric thatmeasures the weighted average of the percentage of faults detected during the execution of the testsuite.Itsvalue ranges from0 to100wherehigher valuemeans fasterfaultdetection rate.

APFDiscalculatedasgivenbelow.

APFD=1-((TF1+TF2+............... +TFm)/nm)+1/2n

where:--TFiisthepositionofthefirsttestinTestsuiteTthatexposesfaulti
--m isthe totalnumber offaults exposedinthesystemor moduleunderT
--nis thetotalnumber oftestcases inT



Forexample:					
	T1	**T2**	**T3**	**T4**	**T5**
F1			X	X	
F2	X		X		X
F3		X		X	
F4	X		X	X	
F5		X		X	X

APFD=1-((3+1+2+1+2)/ 5*5)+(1/2)*5

SoftwareQualityManagement

SoftwareQualitymetrics,SQAmodels

SoftwareQualityMetrics:

Software Quality Metrics a subset of software metrics that focus on the quality aspects of theproduct,process and project.SoftwareQualityMetrics can bedivided into 3 groups:
-->ProductQualityMetrics
-->In-ProcessQualityMetrics
-->MetricsforSoftwareMaintenance

ProductQualityMetrics:

Meantimeto failure(MTTF):MTTFmetricisanestimateoftheaverageormeantimeuntilaproduct'sfirstfailureoccurs.It iscalculatedbyusing:

MTTF=totaloperatingtimeoftheunitstested/totalnumberoffailuresencountered.

Defect Density Metrics: It measures the defects relative to the software

size.DefectDensity=Number ofDefects / Sizeof Product

Customerproblemmetrics:Thismetric measurestheproblems whichcustomersfacewhileusingtheproduct. Theproblemsmaybevaliddefects or usabilityproblems, unclear documentationetc.
Theproblems metric is usuallyexpressed in terms of Problems perUserMonth (PUM).

PUM=Totalproblemsreportedbythecustomerforatimeperiod/Totalnumberoflicensedmonthsof thesoftwareduringthe period.

Customer Satisfaction metrics: Customer satisfaction is usually measured through variousmethodsofcustomersurveysviathefivepointscale:
VerySatisfied,Satisfied,Neutral,Dissatisfied,VerySatisfied.

In-ProcessQualityMetrics:

Defect Density during testing: Higher defect rates found during testing is an indicator that thesoftwarehas experiencedhigher error injection duringits development process.

DefectArrival patternduringTesting: Thepatternofdefectarrivalsorthetimebetweenconsecutivefailuresgives moreinformation.

DefectRemovalEfficiency:

DRE=(Defects removed duringa developmentphase/Defects latentintheproduct)X100%

MetricsforSoftwareMaintenance:

Fix backlog and backlog management index: Fix backlog metric is the count of reportedproblemsthat remainopen at theend ofamonth oraweek. BMI(BacklogManagementMetric)isused to express theindex:

BMI = (Number of problems closed during the month / Number of problem arrivalsduring themonth)x100 %

Fixresponsetimeandfixresponsiveness: Whilefixingtheproblems,time-limitalsomattersaccordingto theseverityofthe problems.

PercentDelinquentfixes:Foreachfix,iftheturn-aroundtimegreatlyexceedsthe requiredresponsetime, then it is classified as delinquent.

Percent Delinquent fixes = (Number of fixes that exceeded the response time criteria byseveritylevel / Number of fixes delivered in a specifiedtime)X 100%

FixQuality: It isthemetric tomeasurethenumberoffixesthat turnoutto bedefective.

Software QualityAssurance (SQA)Models:

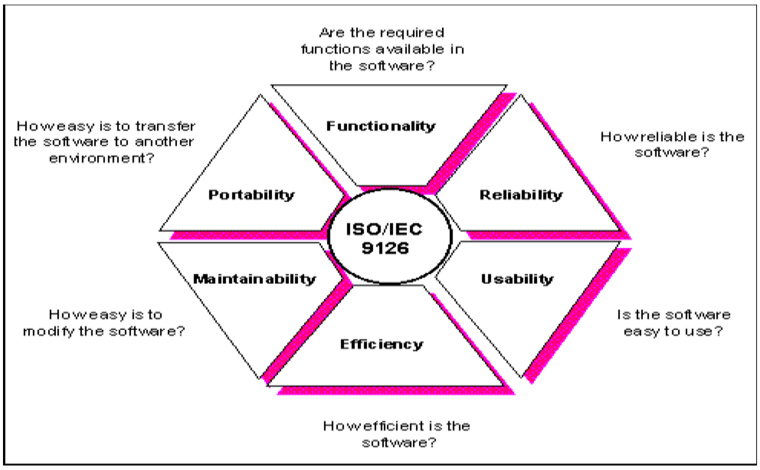
ISO9126:

--Softwaresystemsarelargeand complex withlotsofmaintenanceproblems,whileusersexpecthighquality.However,it ishardtoassessand assurequality.
--TheISO/IEC9126standardhas beendeveloped toaddress softwarequalityissues.
-- It specifies software product quality characteristics and sub-characteristics and proposes metricsfortheirevaluation.
--It is generic, and can be applied to any type of software product by being tailored to a specificpurpose.
--Itaimsateliminatinganymisunderstandingbetweencustomersanddevelopers.
--ISO 9126providesaset ofsixqualitycharacteristics:

- Functionality
- Efficiency
- Maintainability
- Reliability
- Usability
- Portability.

--Theproposednew editionof ISO/IEC9126 willbedividedintothreeparts:
ISO/IEC 9126-1: Information technology - Software quality characteristics and metrics - Part 1:Qualitycharacteristics and subcharacteristics.
ISO/IEC 9126-2: Information technology - Software quality characteristics and metrics - Part 2:Externalmetrics

ISO/IEC 9126-3: Information technology - Software quality characteristics and metrics - Part 3:Internalmetrics.



ISO9126definesthefollowingguidelines forimplementationofthismodel:
Quality requirements definition: First comes the quality requirements definition, which takes asinputasetofstatedorimpliedneeds,relevanttechnicaldocumentationand the ISOStandarditselfandproducesaqualityrequirement specification.

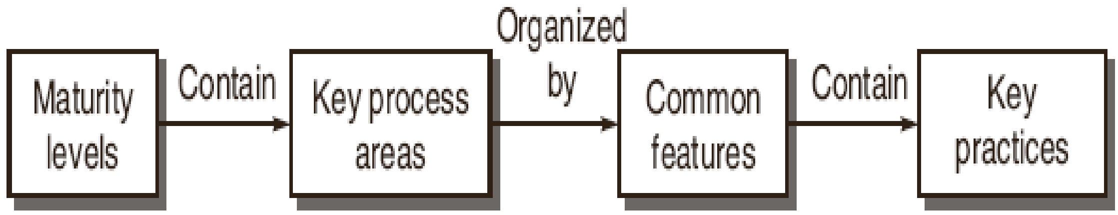
Evaluation preparation: The second stage is that of evaluation preparation, which involves theselection of appropriate metrics, a rating level definition and the definition of assessment criteria.Metrics, in ISO 9126, typically give rise to quantifiable measures mapped on to scales. The ratinglevels definition determines what ranges of values on those scales count as satisfactory orunsatisfactory. The assessment criteria definition involves preparing a procedure for summarizingtheresults of theevaluation.

Evaluation procedure: The final stage is the evaluation procedure, which is refined into threesteps: measurement, rating and assessment. In measurement, the selected metrics are applied to thesoftware product and values on the scales of the metrics obtained. Subsequently, for each measuredvalue, the rating level is determined. Assessment is the final step of the software evaluation process,where a set of rated levels are summarized. The result is a summary of the quality of the softwareproduct.

CapabilityMaturityModel(CMM):
CapabilityMaturityModelisabench-markformeasuringthematurityofanorganization’ssoftwareprocess.Itisamethodologyusedtodevelopan drefineanorganization’ssoftwaredevelopment process. CMM can be used to assess an organization against a scale of five processmaturity levels based on certain Key Process Areas (KPA). It describes the maturity of the companybasedupontheprojectthecompanyisdealingwithandtheclients.Eachlevelrankstheorganization accordingtoitsstandardizationofprocessesinthesubjectarea being assessed.

Amaturity modelprovides:
--Aplacetostart
--Thebenefitofacommunity'spriorexperiences
--Acommonlanguage andasharedvision
--Aframeworkforprioritizingactions
--Awaytodefinewhatimprovementmeansforyourorganization

CMMStructure:



Maturity levels	Indicate	Process capability
Key process areas	Achieve	Goals
Common features	Address	Implementation/Institutionalization
Key practices	Describe	Infrastructure/Activities

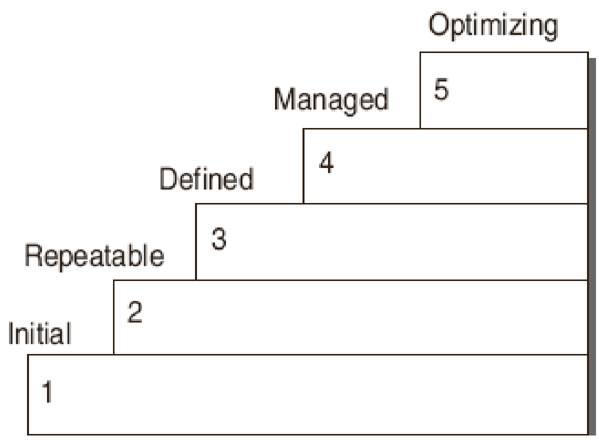
*Maturity Levels:*It is a layered framework providing a progression to the discipline needed toengagein continuous improvement

*Key ProcessAreas:*AKeyProcessArea(KPA) identifiesacluster of related activitiesthat, whenperformed collectively,achieveaset of goals consideredimportant.

*CommonFeatures:*Commonfeaturesincludepracticesthatimplementandinstitutionalizeakeyproces s area. These five types of common features include: Commitment to Perform, Ability toPerform,ActivitiesPerformed,MeasurementandAnalysis,andVerifyingImplementation.

Key Practices: The key practices describe the elements of infrastructure and practice that contributemosteffectivelytotheimplementationand institutionalizationofthekeyprocessareas.

MaturityLevels:



***Initial*:**Itisabasisforcomparisonwiththenextlevels.
Inanorganizationattheinitiallevel,conditionsarenot stableforthe development of qualitysoftware.

***Repeatable*:**Atthislevel,projectmanagementtechnologieshavebeenintroducedinacompany.

That project planning and management is based on accumulated experience and there are standardsfor produced software (these standards are documented) and there is a special quality managementgroup.

Defined: Here, standards for the processes of software development and maintenance areintroducedanddocumented(includingprojectmanagement).Duringtheintroductionofstandards,atr ansition to moreeffectivetechnologies occurs.

Managed: There are quantitative indices (for both software and process as a whole) established inthe organization. Better project management is achieved due to the decrease of digression indifferentproject indices.

Optimizing: Improvement procedures are carried out not only for existing processes, but also forevaluation of the efficiency of newly introduced innovative technologies. The main goal of anorganizationon thislevel ispermanent improvement ofexistingprocesses.

MaturityLevel	**ProcessCapability**
1	-----
2	DisciplinedProcess
3	StandardConsistentProcess
4	PredictableProcess
5	ContinuouslyImprovingProcess

KeyProcessAreas:

Table 13.2 KPAs for level 2

KPAs, at this level, focus on the project's concerns related to establishing basic project management controls.	
KPA	***Goals***
Requirement Management	Document the requirements properly. Manage the requirement changes properly.
Software Project Planning	Ensure proper project plan including estimation and listing of activities to be done.
Software Project Tracking and Oversight	Evaluate the actual performance of the project against the plans during project execution. Action, if there is deviation from plan.
Software Sub-contract Management	Selects qualified software sub-contractors. Maintain ongoing communications between prime contractor and the software sub-contractor. Track the software sub-contractor's actual results and performance against its commitments.
Software Quality Assurance	Plan the SQA activities. Ensure that there are proper processes by conducting review and audits. Take proper actions, if projects fail.
Software Configuration Management	Identify work products and documents to be controlled in the project. Control the changes in the software.

Table 13.3 KPAs for level 3

KPA	Goals
KPAs, at this level, address both project and organizational issues, as the organization establishes an infrastructure that institutionalizes effective software engineering and management processes across all projects.	
Organization Process Focus	Coordinate process development and improvement activities across the organization. Compare software processes with a process standard. Plan organization-level process development and improvement activities.
Organization Process Definition	Standard software processes are defined and documented.
Training Program	Identify training needs of various team members and implementation of training programs.
Integrated Software Management	Tailor the project from the standard defined process. Manage the project according to defined process.
Software Product Engineering	Define, integrate, and perform software engineering tasks. Keep the work products consistent especially if there are changes.
Inter-group Coordination	Ensure coordination between different groups.
Peer Reviews	Plan peer review activities. Identify and remove defects in the software work products.

Table 13.4 KPAs for level 4

KPA	Goals
KPAs, at this level, focus on establishing a quantitative understanding of both the software process and the software work products being built.	
Quantitative Process Management	Plan the quantitative process management activities. Control the performance of the project's defined software process quantitatively. The process capability of the organization's standard software process is known in quantitative terms.
Software Quality Management	Set and plan quantitative quality goals for the project. Measure the actual performance of the project with quality goals and compare them with plans.

Table 13.5 KPAs for level 5

KPAs, at this level, cover the issues that both the organization and the projects must address to implement continuous and measurable software process improvement.	
KPA	**Goals**
Defect Prevention	Plan the defect prevention activities. Identify, prioritize, and eliminate the common causes of bugs.
Technology Change Management	Plan the technology changes to be incorporated in the project, if any. Evaluate the effect of new technologies on quality and productivity.
Process Change Management	Plan the process improvement activities such that an organization-wide participation is there. Measure the change of improvement in the process.

SoftwareTotalQualityManagement(TQM):

Total quality management is a widely used philosophy and business approach that requires alldepartments in an organization to participate in continuous quality improvement efforts. A TQMworkplace values high performance and avoids, or at least minimizes, waste. Most companies,especially manufacturers in regulated environments, use TQM software (or TQM system) to helpthem instill total quality management procedures in all aspects of their operations. TQM softwarethat exists in the market today is designed based on total quality management principles that can befoundin qualitystandards andregulations.

TQM is defined as a quality-centered, customer-focused, fact-based, team-driven, senior-management-ledprocesstoachieveanorganization'sstrategicimperativethroughcontinuousprocessimprovement.

T=Total=***everyone***intheorganizationQ
=Quality= ***customersatisfaction***
M=Management=***people****and****processes***

TheelementsoftheTQMsystemcanbesummarizedasfollows:

Customer-focused*:* The customer ultimately determines the level of quality. No matter what anorganizationdoestofosterqualityimprovement—trainingemployees,integratingqualityintothedesign process, upgrading computers or software, or buying new measuring tools—the customerdetermineswhether theefforts wereworthwhile.

Process-centered**:**Afundamental partofTQM is a focus on process thinking.Aprocess is aseriesof steps that take inputs from suppliers (internal or external) and transforms them into outputs thatare delivered to customers (again, either internal or external). The steps required to carry out theprocess are defined, and performance measures are continuously monitored in order to detectunexpectedvariation.

Human side of Quality: Every organization has a unique work culture, and it is virtually impossibletoachieve excellencein itsproducts andservicesunlessagood qualityculturehasbeen fostered.
Thus, an integrated system connects business improvement elements in an attempt to continuallyimproveand exceedtheexpectationsofcustomers, employees,andotherstakeholders.

Measurement andAnalysis*:*Amajorthrust ofTQM is continual processimprovement. Continualimprovement drives an organization to be both analytical and creative in finding ways to becomemorecompetitive and moreeffectiveatmeetingstakeholderexpectations.

SixSigma:

Six Sigma is a methodology for pursuing continuous improvement in customer satisfaction andprofit.Itisamanagementphilosophyattemptingtoimproveeffectivenessandefficiency.SixSigmais a highly disciplined process that helps us focus on developing and delivering near-perfectproductsand services.

FeaturesofSixSigma:
--SixSigma'saimistoeliminatewasteandinefficiency,therebyincreasingcustomersatisfaction bydeliveringwhat thecustomeris expecting.
--SixSigmafollowsastructuredmethodology, andhasdefinedrolesfortheparticipants.
--
SixSigmaisadatadrivenmethodology,andrequiresaccuratedatacollectionfortheprocessesbeinganalyzed.
--Six Sigmaisaboutputtingresultson FinancialStatements.
--Six Sigmaisabusiness-driven,multi-dimensionalstructuredapproachfor:

- ImprovingProcesses
- LoweringDefects
- Reducingprocess variability
- Reducingcosts
- Increasingcustomersatisfaction
- Increasedprofits

The word *Sigma* is a statistical term that measures how far a given process deviates from perfection.The central idea behind Six Sigma: If you can measure how many "defects" you have in a process,you can systematically figure out how to eliminate them and get as close to "zero defects" aspossibleand specificallyit meansa failure rate of 3.4 partsper millionor 99.9997%perfect.

BenefitsofSixSigma
--Generatessustainedsuccess
--Setsaperformance goalforeveryone
--Enhancesvalueto customers
--Acceleratestherateofimprovement
--Promoteslearningandcross-pollination
--Executesstrategicchange

Sixsigmaprocesseswillproducelessthan3.4defectspermillionopportunities.Toachievethistarget,it uses amethodologyknownas DMAIC having thefollowingsteps:

--Defineopportunities,
--Measureperformance,
--Analyzeopportunity,
--Improveperformance,

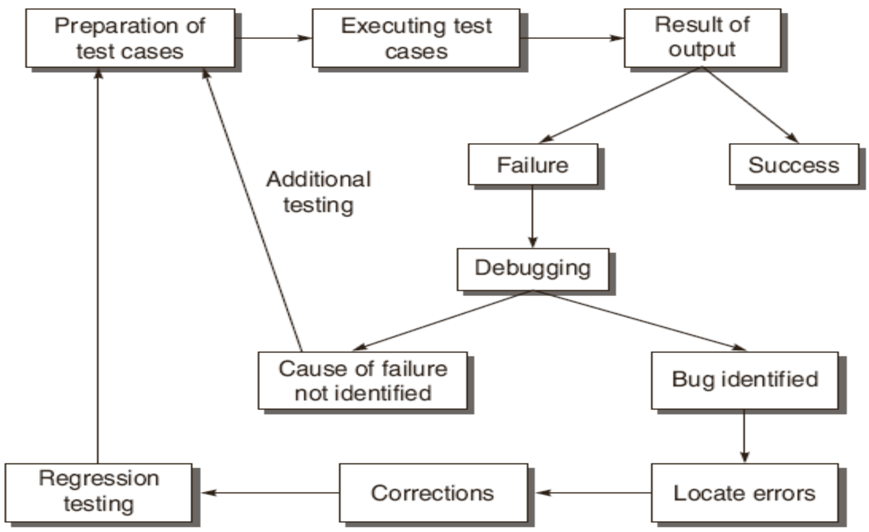
--**C**ontrolperformance

Debugging: Process,Techniques,Correctingbugs,Basicsoftestingmanagementtools,testlinkand Jira

Process:

The goal of debugging process is to determine the exact nature of failure with the help ofsymptoms identified, locate the bugs and errors, and finally correct it. The debugging process isexplainedin thefollowingsteps:

--> Check the result of the output produced by executing test cases prepared in the testing process.If theactual outputmatcheswith theexpectedoutput, itmeansthat theresultsaresuccessful.
-->Debugging is performed for the analysis of failure that has occurred, where we identify the causeof the problem and correct it. It may be possible that symptoms associated with the present failure arenotsufficient to find thebug.



DebuggingTechniques:

DebuggingwithMemoryDump:

In this technique a printout of all registers and relevant memory locations is obtained andstudied.Therelevantdataoftheprogram isobservedthroughthesememorylocationsand registersforanybugin theprogram. Followingarethe drawbacks of this method:

--Thereisdifficultyof establishingthecorrespondencebetweenstoragelocationsandthevariableinone's sourceprogram.

--It shows thestateofthe program at onlyone instant of time.

--TheStoragelocationsareoftenrepresentedinoctalorhexadecimalformats.

DebuggingwithWatchPoints:

At a particular point of execution in the program, value of variables or other actions can beverified. These particular points of execution are known as *watch points*. Debugging with *watchpoints* can beimplemented with the followingmethods:

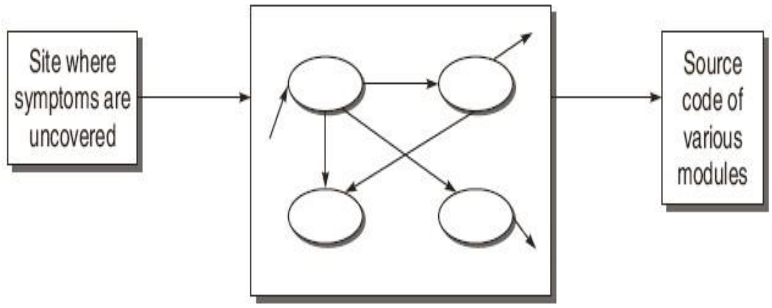
OutputStatements: Inthismethod,outputstatementscanbeusedto checkthestateof acondition oravariableatsomewatchpointintheprogram.Therefore,outputstatementsareinsertedatvariousw atch points;program iscompiled andexecuted withtheseoutputstatements.

Breakpoint Execution: Breakpoint is actually a watch point inserted at various places in theprogram. Breakpoints allow the programmer to control execution of the program and specify howand where the application will stop to allow further examination. Breakpoints have an obviousadvantageover output statements:
--Removingthebreakpointsareeasywhencomparedtooutputstatements.
--Thereisno need tocompile theprogram after insertingbreakpoint, whileitis necessaryforoutputstatement.

Breakpointscanbecategorizedas:
UnconditionalBreakpoint: Itisasimplebreakpointwhitoutanycondition tobeevaluated.
ConditionalBreakpoint:Ontheactivationofthis breakpoint,oneexpressionisevaluatedforitsbooleanvalue.
TemporaryBreakpoint:Thisbreakpointisusedonlyonceintheprogram.Whenitisset,theprogram starts running,andonceit stops,it is removed.

SingleStepping: Theideaofsinglesteppingisthattheusersahouldbeable towatchtheexecutionofthe programaftereveryexecutablestatement.

BackTracking:



An effective method for locating errors in small programs is to backtrack the incorrect resultsthrough the logic of the program until you find the point where the logic went astray. In otherwords, start at the point where the program gives the incorrect result—such as where incorrect datawere printed. At this point you deduce from the observed output what the values of the program'svariables must have been. By performing a mental reverse execution of the program from this pointand repeatedlyusingtheprocessof–if this wasthe stateof theprogram at this point, then this must havebeen the stateoftheprogram up here,‖youcan quicklypinpointtheerror.

CorrectingBugs:

Beforecorrectingtheerrors, weshouldconcentrateon the followingpoints:

(a) Highly coupled module correction can introduce many other bugs. That is why low-coupledmoduleis easyto debug.

(b) Afterrecognizingthe influenceof correctionsonthe othermodules,plan theregressiontests.

(c) Perform regression testing with every correction in the software to ensure that the correctionshavenot introduced newbugs.

DebuggingGuidelines:
--*Freshthinkingleadsto gooddebugging*
--*Don'tisolatethebugfromyourcolleagues*
--*Don'tattemptthecodemodificationsinfirstattempt*

--*Additionaltestcasesaremustifyoudon'tgetthesymptomorcluestosolvetheproblem*
--*Regressiontestingisamustafterdebugging*
--*Designshouldbereferredbeforefixingtheerror*

Mailin JIRA: UsingMailsysteminadminyoucanmailissuestoanaccountonaPOPor IMAPmailserverormessageswritten tothefilesystemgenerated byanexternalmailservice.

Events: An event describes the status, the default template and the notification scheme andworkflow transition post function associations for the event. The events are classified in two aSystemevent(JIRAdefined events)and Custom event (User defined events).

Watchlist:JIRAallowsyoutowatcha particularissue, whichtellsyou aboutthenotificationsofanyupdates relatingto that issue.

Issue Collectors: Inthe form of JIRA issues, an issue collector allows you to gather feedback onanywebsite.

DevelopmentTools:
Youcanalsoconnectyourdevelopmenttoolsto JIRAusingthisadminfunction.Youhaveto entertheURLoftheapplicationto connectwithJIRA.

AdvantagesofJIRATutorial:

--EasytouseSoftwareTestingTutorials
--Amazingtomanageand trackbugs
--Customizeaworkflow tofityourprojectneed
--Plug-insand adaptorsfor yourdeviceenvironment
--Betterreportingforbetteranalysis
--Popularalternate toQC and otherexpensive tools