StaticTesting:

FeaturesofSoftwareTesting:

1. Statictestingtechniquesdonotdemonstratethat thesoftwareisoperationaloronefunction ofsoftwareis working;
2. They check the software product at each SDLC stage for conformance with the requiredspecificationsorstandards.Requirements,designspecifications,testplans,sourcecode ,user'smanuals,maintenanceproceduresaresomeofthe items thatcan bestaticallytested.
3. Statictestinghasprovedtobeacost-effectivetechniqueoferrordetection.
4. Another advantage in static testing is that a bug is found at its exact location whereas a bug foundindynamictestingprovides no indication to theexact sourcecodelocation.

TypesofStaticTesting

-->SoftwareInspections
-->Walkthroughs
-->TechnicalReviews

Inspections:

-->Inspectionprocessisanin-processmanualexaminationofanitemtodetectbugs.
-->Inspection process is carried out by a group of peers. The group of peers first inspects theproductatindividual level.After this,theydiscusspotential defectsof the product observedinaformalmeeting.
-->Itis averyformal processto verifyasoftware product.Thedocuments whichcan beinspected areSRS,SDD,codeandtestplan.
-->Inspectionprocessinvolvestheinteractionofthefollowingelements:
a) Inspectionstepsb)Rolesforparticipantsc)Itembeinginspected

InspectionTeam:

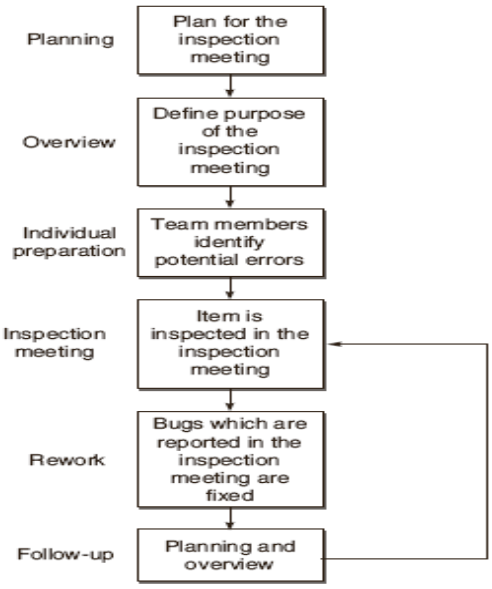
-->Author / Owner / Producer: A program error designer responsible for producing the programordocument.

-->Inspector: A peer member of the team, i.e he is not a manager or supervisor. He is not directlyrelatedto theproduct undersupervision and maybeconcerned with someotherproducts.

-->Moderator:Ateam memberwhomanages thewholeinspectionprocess. Heschedules,leads,andcontrols theinspection session.

-->Recorder: Onewhorecordsalltheresultsoftheinspection meeting.

InspectionProcess:



Planning: Duringthis phasethe followingisexecuted:
-->Theproduct tobeinspected isbeingidentified.
-->Amoderator is assigned.
-->The objective of the inspection is stated i.e whether the inspection is to be conducted for defectdetectionorsomethingelse.

Duringplanning,themoderatorperformsthe followingactivities:
--Assuresthat theproductis readyforinspection.
--Selectstheinspectionteamandassignstheirroles.
--Schedulesthemeetingvenueandtime.
--Distributestheinspectionmateriallikethe itemtobeinspected,clientlistsetc.

Overview: In this stage, the inspection team is provided with the background information forinspection. The author presents the rationale of the product, its relationship to the rest of the productsbeingdeveloped, its function andintended useandtheapproach used to develop it.

Individual Preparation: After the overview, the reviewers individually prepare themselves for theinspection process by studying the documents provided to them in the overview session. They pointout potential errors or problems found and record in a log. This log is then submitted to themoderator.Themoderatorcompilesthelogsofdifferentmembersandgivesacopyofthiscompiledlistto the author ofthe inspected item.

Inspection Meeting: Once all the initial preparation is complete, the actual inspection meeting canstart. Theinspection meeting starts with the author of the inspected item who has created it. Theauthor first discusses every issue raised by different members in the compiled log file. After thediscussion, all the members arrive at a consensus whether the issues pointed out are in fact errorsandif theyareerrors, should theybeadmittedbytheauthor.

Rework:Thesummarylist of the bugs that ariseduringthe inspection meetingneeds to bereworkedbytheauthor.Theauthorfixesallthesebugsandreportsbackto themoderator.

Follow-Up: It is the responsibility of the moderator to check that all the bugs found in the lastmeetinghavebeen resolved.Thedocumentis then approved for release.

BenefitsofInspectionProcess:

-->Bug Reduction: According to the report that through the inspection process in IBM, thenumberof bugs perthousand lines of codehas been reduced bytwo thirds.

--
>**BugPrevention:**Basedontheexperienceofpreviousinspections,analysiscanbemadeforfutureinsp ectionsorprojects,therebypreventingthebugswhich haveappearedearlier.

--
>**Productivity:**SinceallphasesofSDLCmaybeinspectedwithoutwaitingforcodedevelopmentanditsexec ution, thecostof findingbugsdecreases andincreases productivity.

-->Real-time Feedback to Software Engineers: Developers find out the type of mistakes theymake and what is the error density. Since they get this feedback in the early stage of thedevelopment,theymayimprovetheircapability.

-->Reduction in Development Resource: Inspections reduce the effort required for dynamictesting and any rework during design and code, thereby causing an overall net reduction in thedevelopmentresource.

--
>**QualityImprovement:**Thedirectconsequenceofstatictestingalsoresultsintheimprovementofqualityof thefinal product.

-->ProjectManagement
-->CheckingCouplingand Cohesion
-->LearningthroughInspection
-->ProcessImprovement

EffectivenessofInspectionProcess:

In an analysis, the inspection process was found to be effective as compared to structuraltestingbecausetheinspectionprocess alone found52%errors.So theerrordetection ratiocanbe
specifiedas:

$$\text{Errordetectionefficiency} = \frac{\text{Errorfound byan inspection}}{\text{Totalerrorsintheteambeforeinspection}} * 100$$

VariantsofInspectionProcess:

Active Design Reviews

Active Design Reviews [87] are for inspecting the design stage of SDLC. In this process, several reviews are conducted targeting a particular type of bugs and conducted by the reviewers who are experts in that area. Thus, it covers all the sections of the design document based on several small reviews instead of only one inspection. It is also based on the use of questionnaires to give the reviewers better-defined responsibilities and to make them play a more active role. The questionnaires are designed to make the reviewers take an active

stand on issues and to use the documentation. The technique has been used in the Naval Research Laboratory's (Washington) software cost reduction (SCR) project for several years with good results. The SCR project involves the experimental redevelopment of the operational flight program (OFP) for the Navy's A-7E aircraft.

These are conducted in the following stages (see Fig. 6.2):

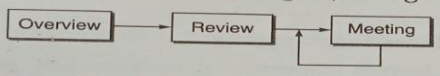


Figure 6.2 Active design reviews process

Overview A brief overview of the module being reviewed is presented. The overview explains the modular structure in case it is unfamiliar to reviewers, and shows them where the module belongs in the structure.

Review Reviewers are assigned sections of the document to be reviewed and questionnaires based on the bug type. Reviewers answer the questions in the questionnaires. They are also assigned a timeframe during which they may raise any questions they have, and a time to meet with designers after the designers have read the completed questionnaires.

Meeting The designers read the completed questionnaires and meet the reviewers to resolve any queries that the designers may have about the reviewer's answers to the questionnaires. The reviewers may be asked to defend their answers. This interaction continues until both designers and reviewers understand the issues, although an agreement on these issues may not be reached. After the review, the designers produce a new version of the documentation.

Formal Technical Asynchronous Review Method (FTArm)

In this process, the meeting phase of inspection is considered expensive and therefore, the idea is to eliminate this phase. The inspection process is carried out without having a meeting of the members. This is a type of asynchronous inspection [88] in which the inspectors never have to simultaneously meet. For this process, an online version of the document is made available to every member where they can add their comments and point out the bugs. This process consists of the following steps, as shown in Fig. 6.3.

Setup It involves choosing the members and preparing the document for an asynchronous inspection process. The document is prepared as a hypertext document.

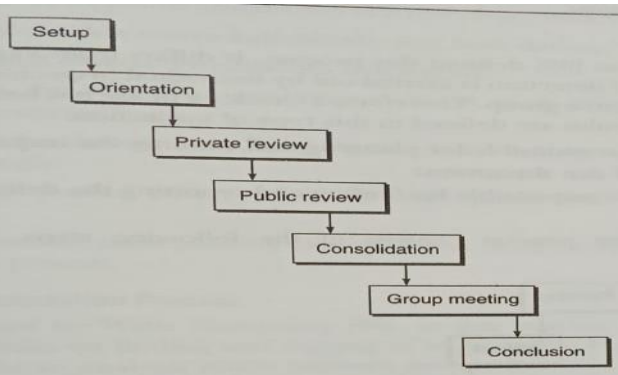


Figure 6.3 Asynchronous method

Orientation It is same as the overview step of the inspection process discussed earlier.

Private review It is same as the preparation phase discussed in the inspection process. Here, each reviewer or inspector individually gives his comments on the document being inspected. However, each inspector provides comments individually and is unable to see the other inspector's comments.

Public review In this step, all comments provided privately are made public and all inspectors are able to see each other's comments and put forward their suggestions. Based on this feedback and with consultation of the author, it is decided that there is a bug.

Consolidation In this step, the moderator analyses the result of private and public reviews and lists the findings and unresolved issues, if any.

Group meeting If required, any unresolved issues are discussed in this step. But the decision to conduct a group meeting is taken in the previous step only by the moderator.

Conclusion The final report of the inspection process along with the analysis is produced by the moderator.

Gilb Inspection

Gilb and Graham [89] defined this process. It differs from Fagan inspection in that the defect detection is carried out by individual inspectors at their own level rather than in a group. Therefore, a checking phase has been introduced. Three different roles are defined in this type of inspection:

- **Leader** is responsible for planning and running the inspection.
- **Author of the document**
- **Checker** is responsible for finding and reporting the defects in the document.

The inspection process consists of the following steps, as shown in Fig. 6.4.

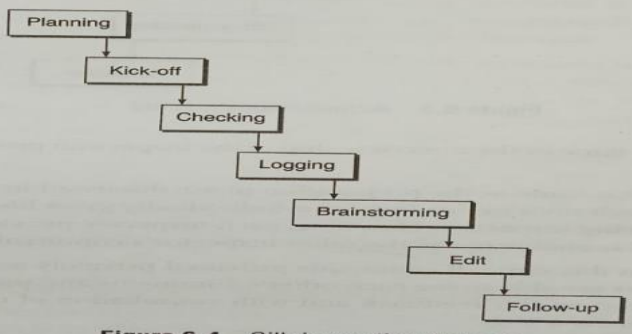


Figure 6.4 Gilb inspection process

Entry The document must pass through an entry criteria so that the inspection time is not wasted on a document which is fundamentally flawed.

Planning The leader determines the inspection participants and schedules the meeting.

Kick-off The relevant documents are distributed, participants are assigned roles and briefed about the agenda of the meeting.

Checking Each checker works individually and finds defects.

Logging Potential defects are collected and logged.

Brainstorm In this stage, process improvement suggestions are recorded based on the reported bugs.

Edit After all the defects have been reported, the author takes the list and works accordingly.

Follow-up The leader ensures that the edit phase has been executed properly.

Exit The inspection must pass the exit criteria as fixed for the completion of the inspection process.

Humphrey's Inspection Process

It was described by Watts Humphrey [90]. In this process, the preparation phase emphasizes on finding and logging of bugs, unlike Fagan inspections. It also includes an analysis phase between preparation and meeting. In the analysis phase, individual logs are analysed and combined into a single list. The steps of this process are shown in Fig. 6.5.

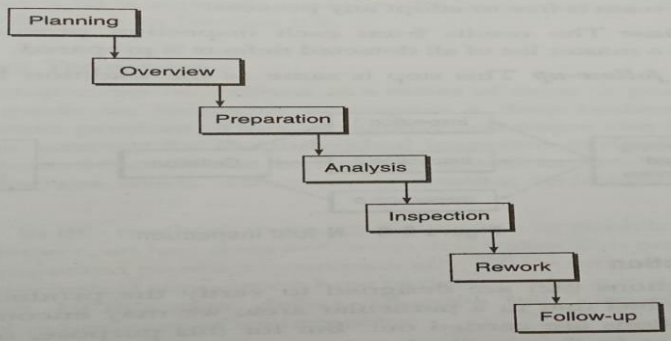


Figure 6.5 Humphrey's process

N-Fold Inspection

It is based on the idea that the effectiveness of the inspection process can be increased by replicating it [91]. If we increase the number of teams inspecting the item, the percentage of defects found may increase. But sometimes the cost of organizing multiple teams is higher as compared to the number of defects found. A proper evaluation of the situation is required. Originally, this process was used for inspecting requirement specifications, but it can also be used for any phase.

As discussed, this process consists of many independent inspection teams. This process needs a coordinator who coordinates various teams, collects and collates the inspection data received by them. For this purpose, he also meets the moderator of every inspection team. This process consists of the following stages, as shown in Fig. 6.6.

Planning and overview This is the formal planning and overview stage. In addition, it also includes the planning of how many teams will participate in the inspection process.

Inspection stages There are many inspection processes adopted by many teams. It is not necessary that every team will choose the same inspection process. The team is free to adopt any process.

Collation phase The results from each inspection process are gathered, collated, and a master list of all detected defects is prepared.

Rework and follow-up This step is same as the tradition Fagan inspection process.

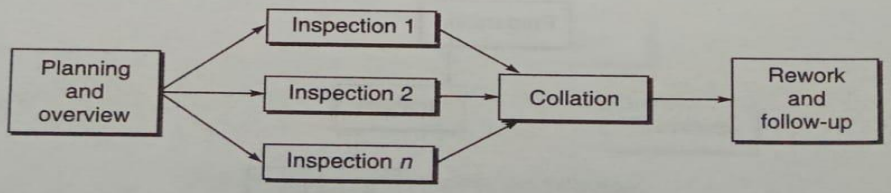


Figure 6.6 N-fold inspection

Phased Inspection

Phased inspections [92] are designed to verify the product in a particular domain. If we feel that in a particular area, we may encounter errors, then phased inspections are carried out. But for this purpose, only experts who have experience in that particular domain are called. Thus, this inspection process provides a chance to utilize human resources. In this process, inspection is divided into more than one phase. There is an ordered set of phases

and each phase is designed such that it will check a particular feature in the product. The next phase in the ordered set of phases assumes that the particular feature has been verified in the previous phase. There are two types of phases, as discussed below.

Single inspector In this phase, a rigorous checklist is used by a single inspector to verify whether the features specified are there in the item to be inspected.

Multiple inspector Here, a checklist cannot be used. It means that the item cannot be verified just with the questions mentioned in the checklist. There are many inspectors who are distributed the required documents for verification of an item. Each inspector examines this information and develops a list of questions of their own based on a particular feature. These questions are not in binary form as in single inspection. The item is then inspected individually by all the inspectors based on a self-developed checklist which is either application or domain specific. After individual checking by the inspectors, a reconciliation meeting is organized where inspectors compare their findings about the item.

Active Design Reviews (ADRs)	Several reviews are conducted targeting a particular type of bugs and conducted by the reviewers who are experts in that area.
Formal Technical Asynchronous review method (FTArm)	Inspection process is carried out without really having a meeting of the members. This is a type of asynchronous inspection in which the inspectors never have to simultaneously meet.
Gilb Inspection	Defect detection is carried out by individual inspector at his level rather than in a group.
Humphrey's Inspection Process	Preparation phase emphasizes the finding and logging of bugs, unlike Fagan inspections. It also includes an analysis phase wherein individual logs are analysed and combined into a single list.
N-Fold inspections	Inspection process's effectiveness can be increased by replicating it by having multiple inspection teams.
Phased Inspection	Phased inspections are designed to verify the product in a particular domain by experts in that domain only.
Structured Walkthrough	Described by Yourdon. Less formal and rigorous than formal inspections. Roles are coordinator, scribe, presenter, reviewers, maintenance oracle, standards bearer, user representative. Process steps are Organization, Preparation, Walkthrough, and Rework. Lacks data collection requirements of formal inspections.

ReadingTechniques:

A reading technique can be defined as a series of steps or procedures whose purpose is toguide an inspector toacquireadeepunderstandingof theinspectedsoftwareproduct.Thusreadingtechnique can be regarded as a mechanism for the individual inspector to detect defects in theinspectedproduct.Thevarious readingtechniquesare:

Ad-hocMethod:Thewordad-hoconlyrefers tothe factthat notechnical supportonhow todetectdefects in a software artifact is given them. In this case, defect detection fully depends on the skills,knowledge,andexperienceof aninspector.

Checklists:Achecklist isa list ofitems that focustheinspectors attentionon specifictopics, suchascommon defects ororganizational rules, whilereviewingasoftwaredocument.

Scenario–BasedReading: Differentmethodsdevelopedbasedonscenariobasedreadingare:

-->***Perspective based Reading: S***oftware item should be inspected from the perspective of differentstakeholders Inspectors of an inspection team have to check software quality as well as the softwarequalityfactors ofasoftwareartifactfrom differentperspectives.

-->***Usage based Reading*:** This method given is applied in design inspections. Designdocumentation is inspected based on use cases, which are documented in requirementsspecification.

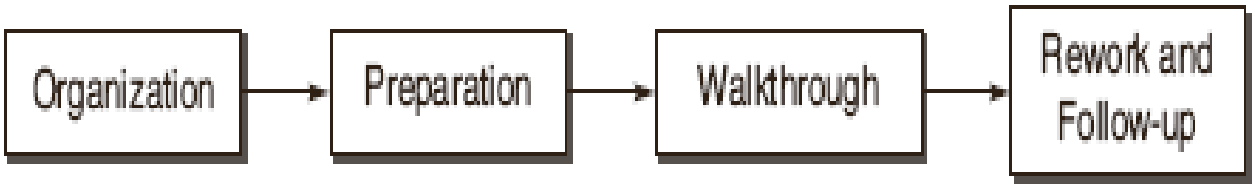
-->***Abstraction driven Reading:*** This method is designed for code inspections. In this method, aninspector reads a sequence of statements in the code and abstracts the functions these statementscompute.

-->***TaskdrivenReading:***Thismethodisalsoforcodeinspections.Inthismethod,theinspectorhasto create a data dictionary, a complete description of the logic and a cross-reference between thecodeand the specifications.

->***Function-point based Scenarios:*** This is based on scenarios for defect detection in

requirementsdocuments [103]. The scenarios, designed around function-points are known as the Function PointScenarios.AFunctionPointScenario consistsofquestionsanddirectsthefocusofaninspectortoaspecificfunction-point item within theinspected requirements document.

StructuredWalkthroughs:



-->Itisaless formalandlessrigoroustechniqueascomparedtoinspection.Theverycommontermused in the literature for static testing is Inspection but it is for very formal process. If you want togoforaless formalhavingno barsoforganized meeting,then walkthroughs area good option.

-->A review is similar to an inspection or walkthrough, except that the review team also includesmanagement.Therefore, itisconsideredahigher-leveltechniquethaninspectionorwalkthrough.

-->A technical review team is generally comprised of management-level representatives of the Userand Project Management.Review agendas should focus less on technical issues and more onoversightthanan inspection.

-->Atypical structuredwalkthrough team consists of:

--*Coordinator*:Organizes, moderatesandfollowsupthewalkthrough
--*Presenter/Developers*:Introducestheitembeinginspected.
--*Scribe/Recorder*:Notesdownthedefects
--*Reviewer/Tester*:Findsthedefectsintheitem.
--*MaintenanceOracle*:Focusesonfuturemaintenanceoftheproject.
--*StandardsBearer*:Assessesadherencetostandards
--*UserRepresentative/AccreditationAgent*:Reflectstheneedsoftheuser.

TechnicalReview:

--A review is similar to an inspection or walkthrough, except that the review team also includesmanagement.Therefore, itisconsideredahigher-leveltechniquethaninspectionorwalkthrough.

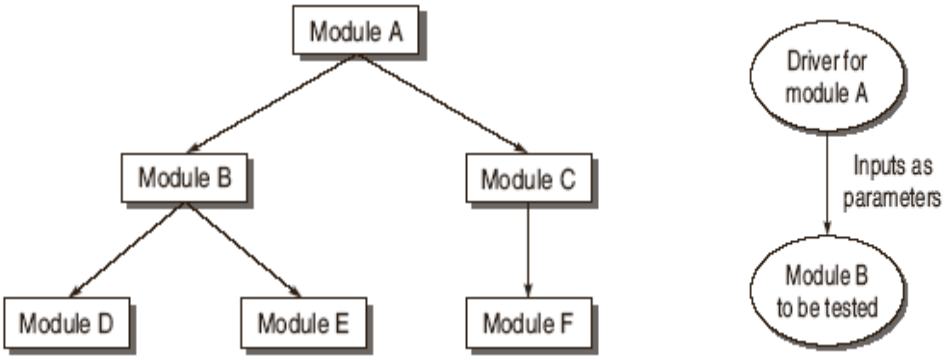
--A technical review team is generally comprised of management-level representatives of the Userand Project Management.Review agendas should focus less on technical issues and more onoversightthanan inspection.

ValidationActivities

UnitTesting:

A unit is the smallest testable part of an application like functions, classes, procedures, interfaces.Unit testing is a method by which individual units of source code are tested to determine if they arefit for use. Unit tests are basically written and executed by software developers to make sure thatcode meets its design and requirements and behaves as expected. The goal of unit testing is tosegregate each part of the program and test that the individual parts are working correctly. Thismeans that for any function or procedure when a set of inputs are given then it should return thepropervalues.

Drivers:



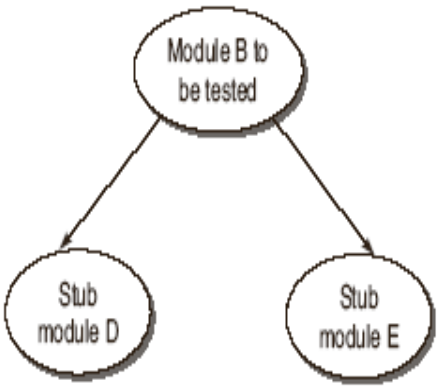
Driversare alsokindofdummymoduleswhichare knownas"callingprograms‖,whichisused when main programs are under construction. Suppose a module is to be tested, where in someinputs are to be received from another module. However this module which passes inputs to themodule tobetestedisnotreadyandunderdevelopment.Insucha situation,weneedtosimulatethe inputs required in the module to be tested. This module where the required inputs for the moduleundertest aresimulated forthe purposeofmoduleorunit testingisknownas**drivermodule**.

For example:When wehave modules B and C ready but module A which calls functions frommoduleBand Cis not readyso developer willwrite adummypiece ofcodefor moduleAwhichwillreturnvalues tomodule Band C.This dummypieceofcodeis known as driver.

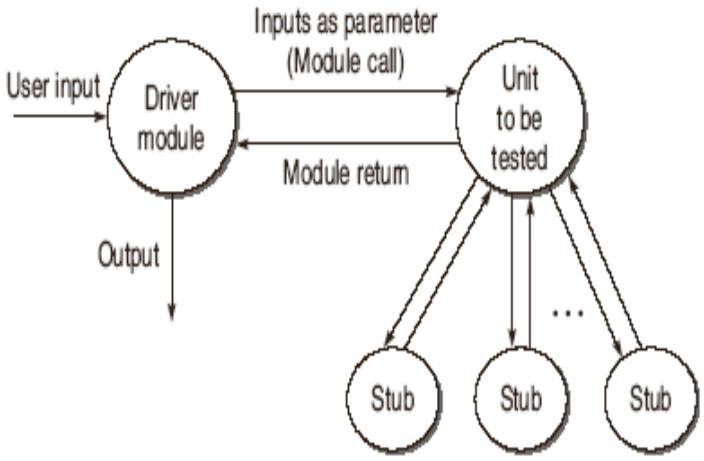
Stubs:

Stubsaredummymoduleswhich areknown as"called programs"whichisusedwhen sub programs are under construction. The module under testing may also call some other module whichisnotreadyat thetimeof testing.Therefore,these modules needtobe simulatedfortesting.In mostcases,dummymodulesinsteadofactual modules,whicharenotready,are preparedforthese subordinatemodules.Thesedummymodules arecalled **Stubs.**

Assume you have 3 modules, Module A, Module B and module C. Module A is ready and we needtotestit,butmoduleAcallsfunctionsfrom Module Band C whicharenotready, so developer willwrite a dummy module which simulates B and C and returns values to module A. This dummymodulecodeis known asstub.



BenifitsofusingStubsandDrivers:



--Stubsallowtheprogrammertocallamethodinthecodebeingdeveloped,evenifthemethoddoesnot havethe desiredbehaviouryet.
--
Byusingstubsanddriverseffectively,wecancutdownourtotaldebuggingandtestingsmallpartsofaprogramindividually, helpingus tonarrow downproblemsbeforetheyexpand.
--StubsandDriverscanalsobeaneffectivetoolfordemonstratingprogressinabusinessenvironment.

Example:

```
main()
{
        inta,b,c,sum;scanf(−%d
        %d‖,&a,&b);
        sum=calsum(a,b);diff=c
        aldiff(a,b);mul=calmul(
        a,b);
        printf(−Thesum is %d:‖,sum);
}
calsum(intx,int y)
{
        int
        d;d=x+y
        ;returnd
        ;
}
```

*Supposeifthemain()moduleotcaldiff()&calmul()moduelisnotready,thenthedriverandstubmodulecan bedesignedas:

Solution:

-->Driverfor main()Module:

```
driver_main()
{
        inta,b,c,sum;scanf(−%d
        %d‖,&a,&b);
        sum=calsum(a,b);
        printf(−Thesum is %d:‖,sum);
}
```

-->*Stubforcall_sum()module:*

```
call_sum(intx,int y)
{
        printf(–Difference  calculatingmodule‖)
        return0;
}
```

IntegrationTesting:

Once all the individual units are createdand tested, westart combiningthose–UnitTested‖ modulesandstartdoingtheintegratedtesting.So themeaningof Integration testingisquitestraightforward-Integrate/combine the unit tested module one by one and test the behaviour as a combinedunit.

The main function or goal of Integration testing is to test the interfaces between theunits/modules.Theindividualmodulesarefirsttestedinisolation.Oncethe modulesareunit tested,they are integrated one by one, till all the modules are integrated, to check the combinationalbehaviour,andvalidatewhethertherequirementsareimplemented correctlyornot.

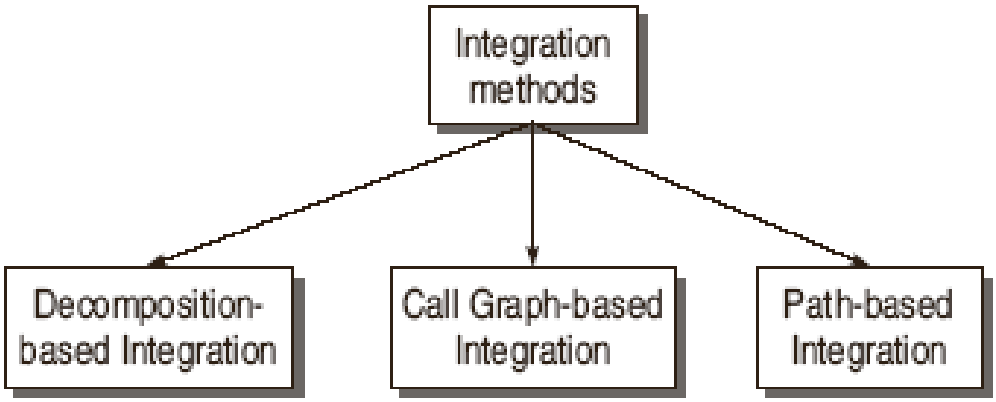
IntegrationTestingisnecessaryforthefollowingreasons:
--Itexposesinconsistencybetweenthemodulessuch asimproper callorreturnsequences.

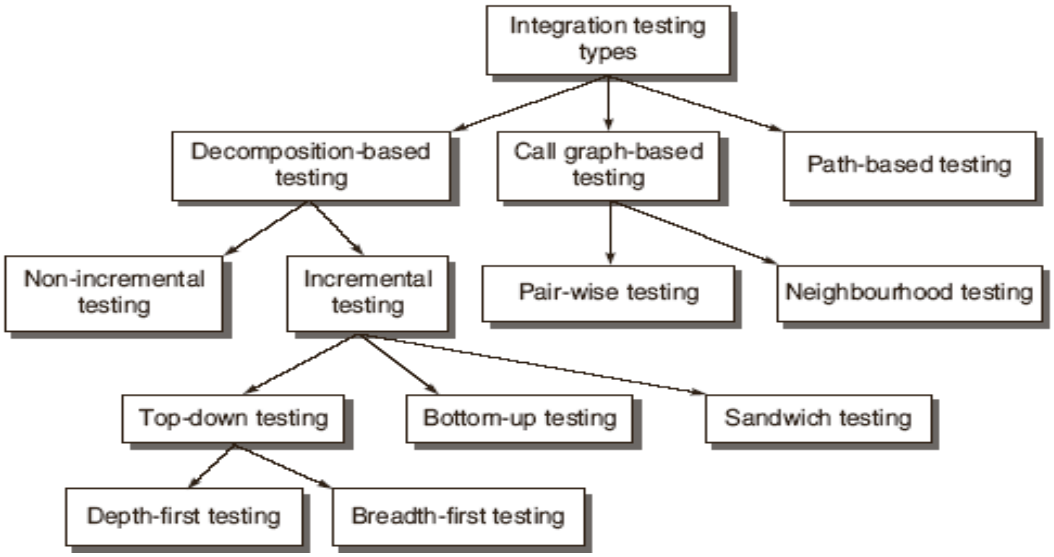
--Datacanbelostacrossaninterface.

--Onemodulewhen combined withanothermodule maynotgive thedesired result.

--Data types and their valid ranges may mismatch between the

modules.Therearethreeapproaches forintegration testing:



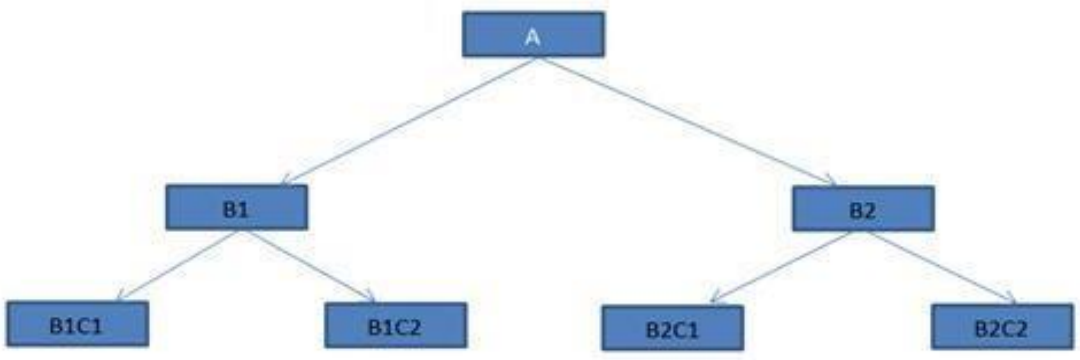


DecompositionBasedIntegration:

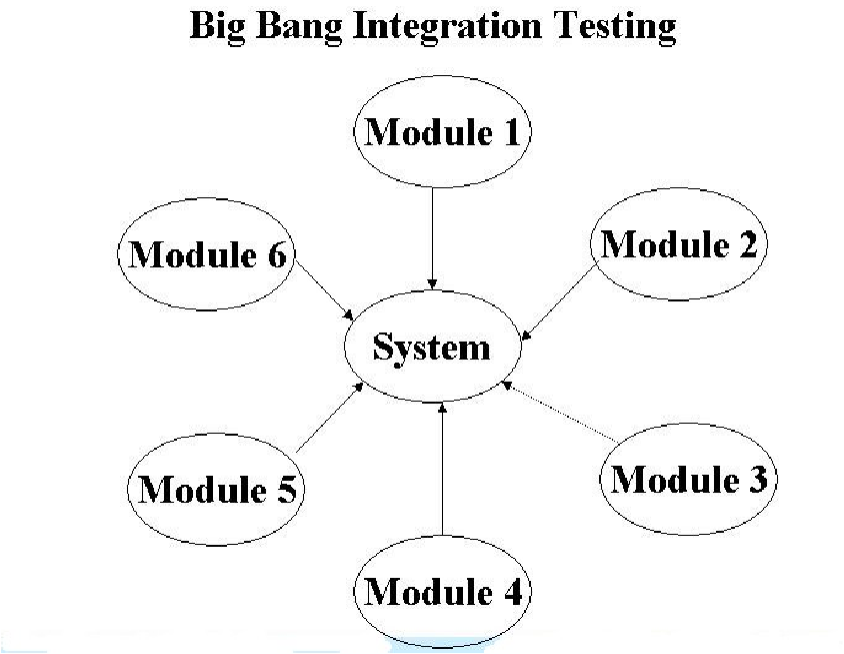
In this strategy, we do the decomposition based on the functional characteristics of the system. Afunctionalcharacteristicisdefinedbywhatthemoduledoes,thatis,actionsoractivities performedby the module. In this strategy our main goal is to test the interfaces among separately tested units.Thereare fourapproaches forthis strategy:

-->Non IncrementalIntegrationTesting/Bigbangintegration
-->IncrementalIntegrationTesting
--Top-DownIntegration
--Bottom-upIntegration
-->PracticalApproach forIntegrationTesting/Sandwichintegration.

Briefly, big-bang groups the whole system and test it in a single test phase. Top-down starts at theroot of the tree and slowly work to lower level of the tree. Bottom-up mirrors top-down, it starts atthe lower level implementation of the system and work towards the main. Sandwich is an approachthatcombines both top-down and bottom-up.



NonIncrementalIntegrationTesting/Bigbangintegration:
--Thisisoneoftheeasiest approachestoapplyinintegrationtesting.
--Herewetreatthe whole system asasubsystemandtestit inasingletestphase.
--Normally this means simply integrating all the modules, compile them all at once and then test theresultingsystem.
--Thisapproachrequireslittleresourcestoexecuteaswe donotneedtoidentifycriticalcomponents (like interactions, paths between themodules)norrequire extracodingforthe–dummy modules‖.
--Thisapproach maybe usedforverysmall systems, howeverit is stillnotrecommended becauseitisnot systematic.
--
Inlargersystems,thelowresourcesrequirementinexecutingthistestingiseasilyoffsetbytheresourcesrequ ired to locate theproblem when it occurs.



Insummary,bigbangintegrationhasthefollowingcharacteristics:
• Considersthewholesystemasasubsystem
• Testsallthemodulesinasingletestsession
• Onlyoneintegrationtestingsession

Advantages:
• Lowresourcesrequirement
• Doesnotrequireextracoding

Disadvantages:
• Notsystematic
• Hardtolocateproblems
• Hardtocreatetestcases

IncrementalIntegrationTesting:
InIncrementalintegrationtesting,thedevelopersintegratethemodulesonebyoneusingstubsor

drivers to uncover the defects. This approach is known as incremental integration testing. To thecontrary,bigbangisoneotherintegrationtestingtechnique,whereallthemodulesareintegratedinonesh ot.

IncrementalIntegrationTestingis beneficialforthefollowingreasons:

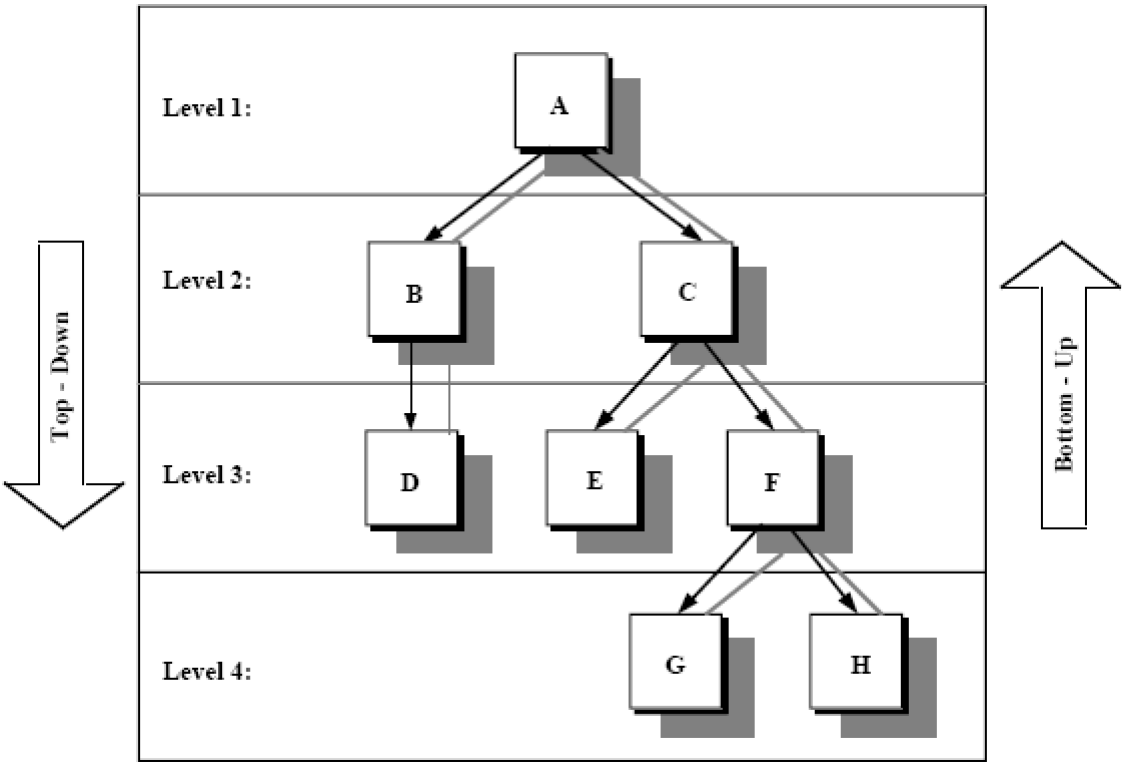
--EachModuleprovides adefinitiveroleto playinthe project/productstructure

--EachModulehasclearlydefineddependencies someof which canbeknownonlyattheruntime.

--The incremental integration testing's greater advantage is that the defects are found early in asmaller assemblywhen itis relativelyeasyto detect theroot causeof thesame.

--A disadvantage is that it can be time-consuming since stubs and drivers have to be developed forperformingthesetests.

TypesofIncrementalintegrationtesting:
-->TopDownintegrationtesting
--> Bottom-upintegrationtesting



TopDownIntegrationTesting:
--Intop-downintegration,westartatthe targetnodeatrootofthefunctionaldecomposition treeandwork toward theleaves.
--Stubsareusedtoreplacethechildrennodesattachedtothetargetnode.
--A test phase consists in replacing one of the stub modules with the real code and test the resultingsubsystem.
--Ifno problemis encounteredthen wedo thenext testphase.
--If allthechildrenwerereplacedbyrealcodeatleastonceandmeetthe requirements thenwe

movedown to thenext level.
--Now we can replace the higher level tested modules with real code and continue the integrationtesting.
--Fortop-downintegrationthenumberofintegrationtestingsessionsis:nodes−leaves+edges.Top-downintegration hasthe drawback ofrequiringstubs:
--While stubs are simpler than the real code, it is not straightforward to write them; the emulationmust be complete and realistic, that is, the test cases results ran on the stub should match with theresultson therealcode.
--Being a throw-away code, it does not reach the final product nor it will increase functionality ofthesoftwarethus it isextraprogrammingeffort withoutadirect reward.

Modulessubordinate tothe topmoduleareintegrated inthe followingtwoways:

Depth First Integration: In this type, all modules on a major control path of the design hierarchy areintegratedfirst.Inthefigureshownabove,modulesA,B,andDareintegratedfirst,nextmodulesA,C, E, FG,h and integrated.

Breadth First Integration: In this type, modules directly subordinate at each level, moving acrossthe designhierarchy horizontally are integrated first. In the figure shown above, modules B,C areintegratedfirst,nextmodulesD,E, FandatlastmodulesG, Hintegrated.

Bottom-upIntegrationTesting:
--Bottom-up integration starts at the opposite end of the functional decomposition tree, instead ofstartingat themain programwestart atthe lower-levelimplementation ofthe software.
--By moving in an opposite direction, the parent nodes are replaced by throw-away codes instead ofthechildren.
--Thesethrow-awaycodes arealso knownas drivers.
--This approach allow us to start working with simpler and lower level of the implementation,allowing us to create testing environments more easily because of the simpler outputs of thosemodules.
--Thisalsoallowsustohandletheexceptionsmoreeasily.
--Conversely,wedonothaveanearlyprototypethusthemainprogramisthelasttobetested. Ifthere is a design error then it will be just identified at a later stage, which implies high errorcorrectioncost.
--Bottom-up integration is commonly used for object-oriented systems, real-time systems andsystemswith strict performancerequirements.
--Forbottom-upintegrationthenumberofintegrationtestingsessionsis:nodes−leaves+edges.

Issue	Top-Down Testing	Bottom-Up Testing
Architectural Design	It discovers errors in high-level design, thus detects errors at an early stage.	High-level design is validated at a later stage.
System Demonstration	Since we integrate the modules from top to bottom, the high-level design slowly expands as a working system. Therefore, feasibility of the system can be demonstrated to the top management.	It may not be possible to show the feasibility of the design. However, if some modules are already built as reusable components, then it may be possible to produce some kind of demonstration.
Test Implementation	*(nodes – 1)* stubs are required for the subordinate modules.	*(nodes – leaves)* test drivers are required for super-ordinate modules to test the lower-level modules.

PracticalApproach forIntegrationTesting/Sandwichintegration:

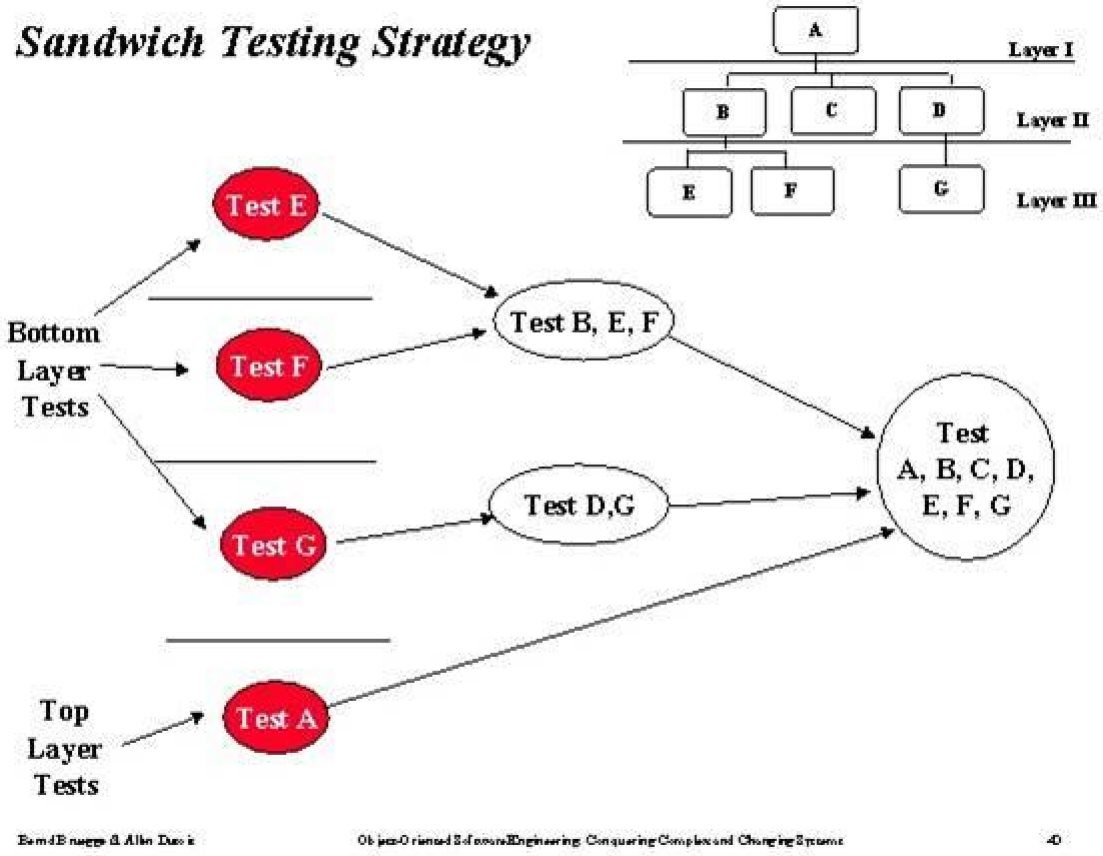
--Sandwichintegrationcombinestop-downintegrationandbottom-upintegration.

--The main concept is to maximize the advantages of top-down and bottom-up and minimizing theirweaknesses.

--Sandwich integration uses a mixed-up approach where we use stubs at the higher level of the treeanddrivers at thelower level (Figure).

--Thetestingdirectionstartsfromboth sideoftreeandconvergestothe centre,thus thetermsandwich.

--This will allow us to test both the top and bottom layers in parallel and decrease the number ofstubsand drivers required in integration testing.



Advantages:

--Topandbottomlayerscanbedoneinparallel

-- Lessstubsanddriversneeded

--Easyto constructtest cases

--Bettercoveragecontrol

--Integrationisdoneassoonacomponentisimplemented

Disadvantages:

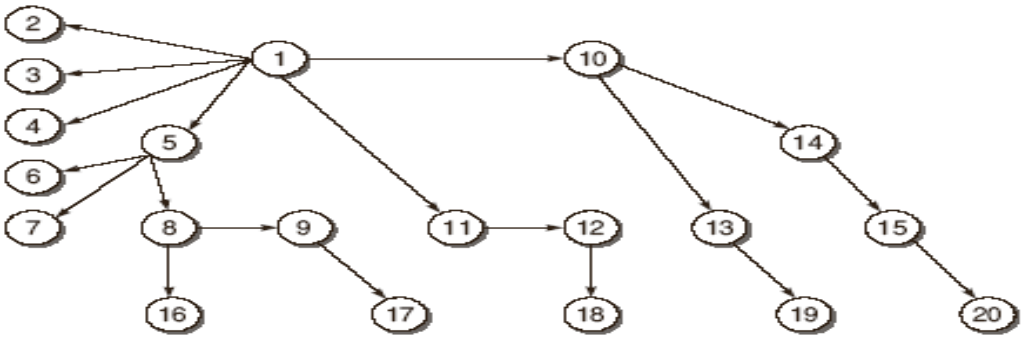
--Stillrequiresthrow-awaycodeprogramming

--Partial bigbangintegration

--Hardto isolateproblems

Call-GraphbasedIntegration:

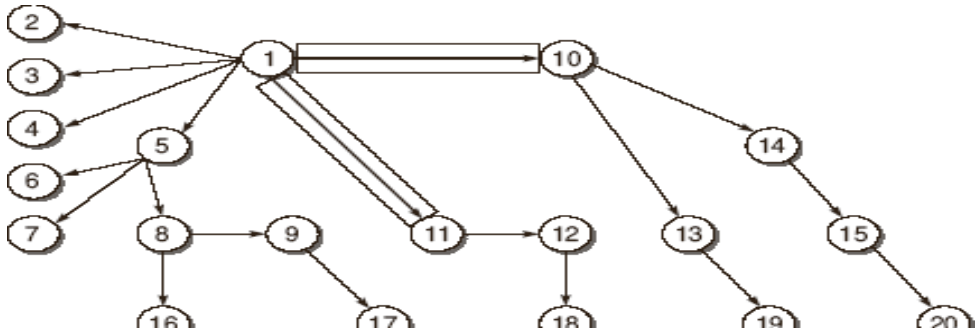
A call graph is a directed graph, where the nodes are either modules or units, and a directededge from one node to another node means one module has called another module. The call graphcanbecaptured inamatrixformwhich is known as the adjacencymatrix.



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	×	×	×	×					×	×									
2																			
3																			
4																			
5					×	×	×												
6																			
7																			
8								×							×				
9																×			
10												×	×						
11											×								
12																	×		
13																		×	
14														×					
15																			×
16																			
17																			
18																			
19																			
20																			

Therearetwotypes ofintegrationtestingbasedoncallgraph:
-->PairwiseIntegration
-->NeighbourhoodIntegration.

PairwiseIntegration:
--Inpair-wiseintegration,weeliminatetheneedofstubanddriver,byusingtherealcodeinstead.--This is similar to big bang where has problem isolation problem due to the large amount of moduleswearetestingat once.
--By pairing up the modules using the edges, we will have a number of test sessions equal to thenumberof edges that exist in thecall graph.
--Since the edges correspond to functions or procedures invocated, in a standard system, thisimplies manytest sessions.
--Forpair-wiseintegrationthenumberofintegrationtestingsessionsisthenumberofedges.



NeighbourhoodIntegration:

--Whilepair-wiseintegrationeliminatestheneedofstubanddriver,itstillrequiresmanytest cases.

--Asan attemptofimprovingfrompair-wise,neighbourhoodrequiresfewertestcases.

--Inneighbourhoodintegration, wecreateasubsystem foratestsession byhaving atarget nodeandgroupingall the nodes near it.

--Nearisdefinedasnodes thatarelinkedtothetargetnodethatisanimmediatepredecessororsuccessorofit.

--Bydoingthiswewill beable toreduceconsiderablythe amount oftest sessions required.

--The total test sessions in neighbourhood integration can be calculated as:Neighbourhood=nodes – sink nodes

=20 -10

=10

whereSinkNodeisaninstruction ina moduleatwhich executionterminates.

Node	Neighbourhoods	
	Predecessors	Successors
1	-----	2,3,4,5,10,11
5	1	6,7,8
8	5	9,16
9	8	17
10	1	13,14
11	1	12
12	11	18
13	10	19
14	10	15
15	14	20

PathBasedIntegration:

--Bymovingtopath-based integrationwewillbeapproachingintegration testingfromanewdirection.Here we will try to combine both structural and functional approach in path-baseintegration.

--

Finally,insteadoftestingtheinterfaces(whicharestructural),wewillbetestingtheinteractions(whichareb ehavioural).

--Here,whenaunitis executedcertainpathofsourcestatementsistraversed.

--When this unit calls source statements from another unit, the control is passed from the callingunitto thecalled unit.

--Forintegrationtestingwetreattheseunitcallsasanexitfollowedbyan

entry.Weneedtounderstandthefollowingdefinitionsforpath-basedintegration:

Source Node:Aprogramstatement fragmentat whichprogram execution begins or

resumes.SinkNode:Astatementfragment atwhich program execution terminates

Module executionpath (MEP):Asequence of statementswithin amodulethat beginswith asourcenode,Ends with a sink nodewith no interveningsink nodes.

Message:Aprogramminglanguagemechanismbywhich one unittransferscontrol toanotherunit.Usually interpreted as subroutine / function invocations .The unit which receives the messagealwaysreturnscontrol to the messagesource.
MM-path: A module to module path. It is an interleaved sequence of module execution paths andmessages which are used to describe sequences of module execution paths that include transfers ofcontrol among separate units. MM-paths always represent feasible execution paths, and these pathscrossunit boundaries.

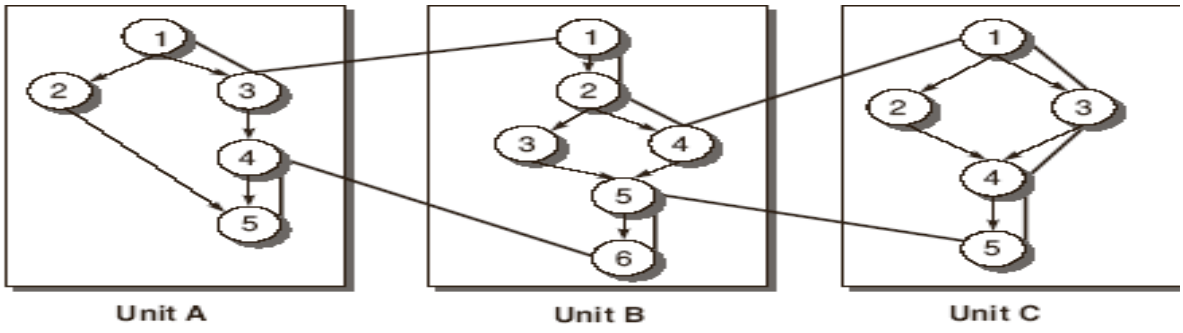


Figure 7.12 MM-path

Table 7.3 MM-path details

	Source Nodes	Sink Nodes	MEPs
Unit A	1,4	3,5	MEP(A,1) = <1,2,5> MEP(A,2) = <1,3> MEP(A,3) = <4,5>
Unit B	1,5	4,6	MEP(B,1) = <1,2,4> MEP(B,2) = <5,6> MEP(B,3) = <1,2,3,4,5,6>
Unit C	1	5	MEP(C,1) = <1,3,4,5> MEP(C,2) = <1,2,4,5>

FunctionTesting:

Functiontestingisdefinedas—theprocessofattemptingtodetectdiscrepanciesbetweenthefunctional specificationsofasoftwareanditsactualbehaviour‖.Whenanintegratedsystemistested, all its specified functions and external interfaces are tested on the software. Everyfunctionality of the system specified in the functions is tested according to its externalspecifications.Thefunctiontest must determineif eachcomponentorbusiness event:

--Performsinaccordancetothespecifications
--Responds correctlyto all conditions thatmaybepresented byincomingevents / data,
--Movesdata correctlyfromonebusiness eventto thenext(includingdata stores)
--Business eventsinitiatedintheorderrequiredtomeetthebusinessobjectivesofthesystem.

Aneffectivefunctiontestcyclemusthave adefinedsetofprocessesanddeliverables.Theprimaryprocesses / deliverables for requirementsbasedfunction test are:

TestPlanning:Duringplanning,thetestleaderwithassistancefromthetestteamdefinesthescope,schedule ,and deliverables forthefunction test cycle.

Partioning / Functional Decomposition: Functional decomposition of a system is the breakdown ofasystem into functionalcomponents orfunctionalareas.

RequirementDefinition:Thetestingorganizationneedsspecifiedrequirementsintheformofproperdocumentsto proceed withthe function test.

Testcase design:Atesterdesigns and implementsatest caseto validate that theproduct performs inaccordancewith therequirements.

Traceability matrix formation: Test cases need to be traced / mapped back to the appropriaterequirement. A function coverage matrix is prepared. This matrix is a table, listing specificfunctions to be tested, the priority for testing each function, and test cases that contain tests for eachfunction.

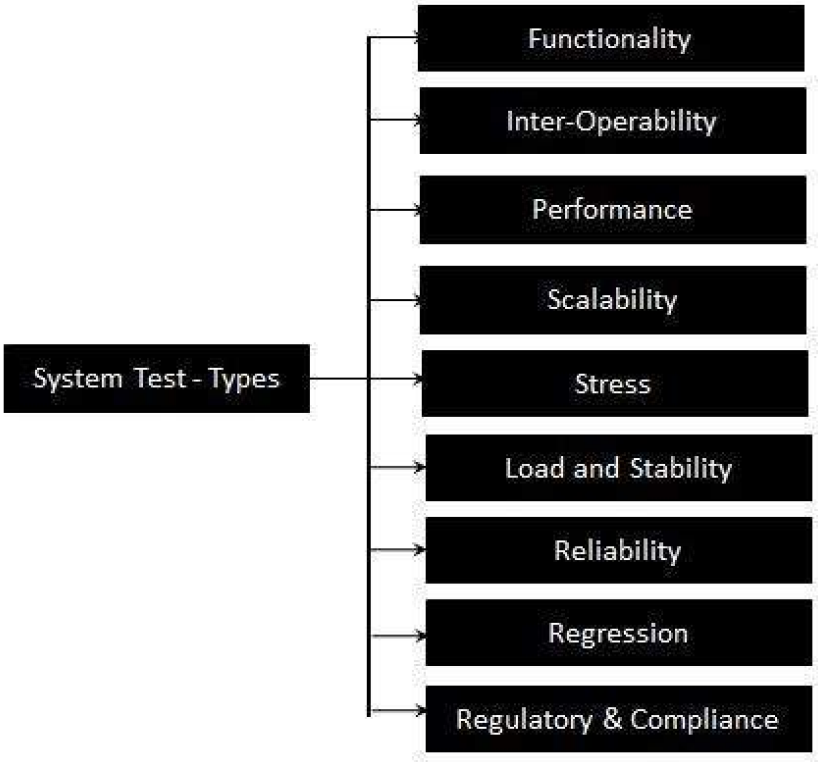
Functions/Features	**Priority**	**TestCases**
F1	3	T2,T4,T6
F2	1	T1.T3,T5

Testcaseexecution:Asinallthephasesoftesting,anappropriatesetoftestcasesneedtobeexecuteda nd the results of thosetest casesrecorded.

SystemTesting:

--System testing is the type of testing to check the behaviour of a complete and fully integratedsoftwareproduct based on thesoftware requirements specification (SRS)document.
--Themainfocus ofthis testingis toevaluateBusiness/ Functional/End-user requirements.
--Thisisblack boxtype oftestingwhereexternal workingofthesoftwareisevaluated withthehelpofrequirement documents&itistotallybased onUsers point ofview.
--Forthistypeoftestingdonotrequired knowledgeofinternaldesignorstructureor code.
--Thistestingistobecarriedoutonlyafter System IntegrationTestingiscompletedwherebothFunctional&Non-Functional requirementsareverified.
--In the integration testing testers are concentrated on finding bugs/defects on integrated modules.But in the Software System Testing testers are concentrated on finding bugs/defects based onsoftwareapplication behaviour,softwaredesign andexpectationofenduser.

CategoriesofSystemTesting:



RecoveryTesting:
Recovery is just like the exception handling feature of a programming language. It is a type of non-functional testing. Recovery testing is done in order to check how fast and better the application canrecover after it has gone through any type of crash or hardware failure etc. Recovery testing is theforcedfailureof thesoftwarein avarietyofwaysto verifythatrecoveryisproperlyperformed.
ThusRecoveryTestingis―theactivityoftestinghowwellthesoftwareisabletorecoverfromcrashes,hard ware failures, andothersimilarproblems‖.

SomeexamplesofRecoverytestingare:
--When an application isreceivingdatafromanetwork, unplugtheconnectingcable.After some time,plugthecablebackinandanalyzetheapplication'sabilitytocontinue receivingdatafromthepointat which thenetwork connection was broken.
--Restart the system while a browser has a definite number of sessions and check whether thebrowseris able torecover all of them or not.

*"Biezer" proposes that testers should work on the following areas during recovery testing:Restart:*Testersmustensurethatalltransactionshavebeenreconstructedcorrectlyandthatal ldevicesarein proper states.

Switchover: Recovery can also be done if there are standby components and in case of failure of onecomponent,the standbytakes over thecontrol.

SecurityTesting:
--Itisatypeofnon-functionaltesting.
--Securitytestingisbasicallyatypeofsoftwaretestingthat'sdonetocheck whetherthe applicationorthe product is securedornot.
--Itcheckstoseeiftheapplicationisvulnerable toattacks,ifanyonehackthesystemorloginto theapplicationwithout anyauthorization.
--It is a process to determine that an information system protects data and maintains functionality asintended.
--The security testing is performed to check whether there is any information leakage in the sensebyencryptingtheapplicationorusingwiderangeofsoftware's andhardware's andfirewalletc.
--Softwaresecurityis aboutmakingsoftwarebehavein thepresenceofa malicious attack.

TypesofSecurityRequirements:
--SecurityRequirementsshould beassociated with eachfunctionalrequirement.
--In addition to security concerns that are directly related to particular requirements, a softwareprojecthas securityissues that areglobal in nature.

Howto performsecuritytesting:
Testers must use a risk based approach, grounded in both the systems architectural realityandtheattackersmindset,togaugesoftwaresecurityadequately.
Byidentifyingrisksandpotentialloss associated with those risks in the system and creating tests driven by those risks, the tester canproperlyfocus onareas of codein whichan attackis likelyto succeed.

ElementsofSecurityTesting:
--Confidentiality
--Integrity

--Authentication
--Availability
--Authorization
--Non-repudiation.

PerformanceTesting:

--Softwareperformancetestingisameans of qualityassurance(QA).
--It involves testing software applications to ensure they will perform well under their expectedworkload.
--Featuresand Functionalitysupported byasoftwaresystem isnot theonlyconcern.Asoftwareapplication'sperformancelikeitsresponse time, domatter.
--Thegoalofperformancetestingisnottofindbugsbuttoeliminateperformancebottlenecks
--Performance testing is done to provide stakeholders with information about their applicationregardingspeed, stabilityand scalability.
--Moreimportantly,performancetestinguncoverswhatneedstobeimprovedbeforetheproductgoesto market.
--Withoutperformancetesting,softwareislikelytosuffer fromissuessuchas:runningslowwhileseveral users use it simultaneously, inconsistencies across different operating systems and poorusability.
--Performance testing will determine whether or not their software meets speed, scalability andstabilityrequirements under expected workloads.
--Applications sent to market with poor performance metrics due to non existent or poorperformancetestingarelikelytogainabadreputationand failtomeetexpectedsalesgoals.
--Also, mission critical applications like space launch programs or life saving medical equipmentsshould beperformancetested to ensurethat theyrun foralongperiod oftime without deviations.

Thefollowingtasksmust bedoneforthis thing:
--Develophigh levelplanincludingrequirements,resources,timeliness, andmilestones.
--Developadetailed performancetestplan.
--Specifytest data needed.
--Execute testsprobablyrepeatedlyonorder tosewhetheranyunaccountedfactormight affecttheresults.

LoadTesting:

--Loadtestingisa typeofnon-functionaltesting.
--A load test is type of software testing which is conducted to understand the behaviour of theapplicationunder aspecificexpected load.
--
Loadtestingisperformedtodetermineasystem’sbehaviourunderbothnormalandatpeakcondition s.
--It helps to identify the maximum operating capacity of an application as well as any bottlenecksanddeterminewhichelementiscausingdegradation.E.g.If thenumberof usersareincreasedthenhowmuch CPU,memorywill beconsumed, whatis thenetworkand bandwidthresponse time.
--
Loadtestingcanbedoneundercontrolledlabconditionstocomparethecapabilitiesofdifferentsystemsor to accuratelymeasurethecapabilities ofasinglesystem.
--Loadtestinginvolvessimulatingreal-lifeuserloadforthetargetapplication.Ithelpsyoudeterminehowyour applicationbehaveswhenmultipleusershitsitsimultaneously.
- Load testing differs from stress testing, which evaluates the extent to which a system keepsworking when subjected to extreme work loads or when some of its hardware or software has beencompromised.

--The primary goal of load testing is to define the maximum amount of work a system can handlewithoutsignificant performancedegradation.
Examplesofloadtestinginclude:
- Downloadingaseriesoflargefiles fromtheinternet.

- Runningmultipleapplicationsonacomputerorserversimultaneously.
- Assigningmanyjobs toaprinterina queue.
- Subjectingaservertoalarge amountoftraffic.
- Writingandreadingdatatoandfromaharddiskcontinuously.

StressTesting:

- Itisatypeofnon-functionaltesting.
- Itinvolvestestingbeyondnormaloperational capacity,oftentoabreakingpoint,inordertoobservethe results.
- Itis a form ofsoftwaretestingthat isusedto determinethestabilityof a given system.
- Itputs greateremphasis onrobustness,availability, anderrorhandlingunderaheavyload,ratherthanon what wouldbe considered correctbehaviour under normal circumstances.
- Thegoals ofsuch testsmaybetoensurethesoftwaredoesnotcrash inconditions ofinsufficientcomputationalresources(such as memoryor disk space).
- Thus –*StressTesting tries to breakthe system undertest by overwhelming itsresources in order to find thecircumstances underwhich itwillcrash*‖
- Theareasthatmaybestressedinasystemare:InputTransactions,DiskSpace,Output,Communications,Interaction with users.

UsabilityTesting:

--Usabilitytestingis an essential elementof qualityassurance.
--Itisthemeasureofaproduct'spotentialtoaccomplishthegoalsoftheuser.
--Usability testing is a method by which users of a product are asked to perform certain tasks in anefforttomeasuretheproduct'sease-of-use,tasktime,andtheuser'sperceptionoftheexperience. --This look as a unique usability practice because it provides direct input on how real users use thesystem.
--Usabilitytestingmeasureshuman-usableproductsto fulfiltheusers purpose.
--The item which takes benefit from usability testing are web sites or web applications, documents,computerinterfaces,consumerproducts, and devices.
--Usability testing processes the usability of a particular object or group of objects, where commonhuman-computerinteraction studies tryto formulate universal principles.

Whattheuserwantsor exceptsfromthesystemcan bedetermined usingseveral wayslike:
--AreaExperts,
--Groupmeetings
--Surveys
--Analysesimilarproducts

Usabilitycharacteristics againstwhichtestingisconductedare:
--EaseofUse
--Interfacesteps
--ResponseTime
--HelpSystem
--ErrorMessages

Compatability/Conversion/ConfigurationTesting:

--Compatibilityis anon-functionaltestingtoensurecustomersatisfaction.
--
Itistodeterminewhetheryoursoftwareapplicationorproductisproficientenoughtorunindifferentbrowsers,database,hardware,operatingsystem,mobiledevicesandnetworks.
--Application could also impact due to different versions, resolution, internet speed andconfigurationetc.Henceit'simportanttotesttheapplicationinallpossiblemannerstoreducefailuresandovercomefrom embarrassmentsof bug'sleakage.
--AsaNon-functional tests, Compatibilitytestingis to endorsethat theapplication runs properlyin

differentbrowsers,versions,OSandnetworkssuccessfully.
--Compatibility test should always perform on real environment instead of virtual environment.Testthecompatibilityofapplicationwithdifferentbrowsersandoperatingsystemstoguarantee100%coverage.
TypesofSoftwarecompatibilitytesting:
- Browsercompatibilitytesting
- Hardware
- Networks
- MobileDevices
- OperatingSystem
- Versions

AcceptanceTesting:
--After the system test has corrected all or most defects, the system will be delivered to the user orcustomerforacceptancetesting.
--Acceptance testing is basically done by the user or customer although other stakeholders may beinvolvedas well.
--Thegoalofacceptance testingistoestablishconfidenceinthesystem.
--Acceptancetestingismostoftenfocusedonavalidationtypetesting.
--Thus –*AcceptanceTesting is the formal testing conducted to determine whether a softwaresystem satisfies its acceptance criteria and to enable buyer to determine whether to accept the system ornot."*
--Thusacceptancetestingisdesignedto:
-Determine whetherthesoftwareis fitfor theuserto use.
-Makingusersconfidentaboutproduct
-Determinewhetherasoftwaresystemsatisfiesits acceptancecriteria.
-Enablethebuyertodeterminewhethertoacceptthesystem.

TypesofAcceptanceTesting:
-->AlphaTesting
-->BetaTesting

AlphaTesting:
--Alpha testing is one of the most common software testing strategyused in software development.Itsspeciallyused byproduct development organizations.
--Thistesttakesplaceatthedeveloper'ssite.Developersobservetheusers andnoteproblems.
--Alpha testing is testing of an application when development is about to complete. Minor designchangescan still be madeasa result ofalpha testing.
--Alpha testing is typically performed by a group that is independent of the design team, but stillwithinthecompany,e.g.in-housesoftwaretestengineers,orsoftwareQAengineers.
--Alpha testing is final testing before the software is released to the general public. **It has twophases:**
-->Alphatestingissimulated oractual operationaltestingbypotentialusers/customers oranindependenttestteam at thedevelopers'site.
-->Alpha testing is often employed for off-the-shelf software as a form of internal acceptancetesting,beforethe software goes to beta testing.

EntryCriteriaforAlpha:
--Allfeaturesarecomplete/testable
--Highbugsonprimaryplatformarefixed /verified.
--50%ofmedium bugs on primaryplatforms arefixed / verified.
--Allfeaturesaretestedonthe primaryplatforms.

--Performancehasbeenmeasured/compared
--Alphasitesarereadyforinstallation.

ExitcriteriatoAlpha:
--Getresponse/ feedbacksfromthecustomers.
--Preparereport ofanyserious bugs beingnoticed.
--Notifybug– fixingissues to developers.

BetaTesting:
--In software development, a beta test is the second phase of software testing in which a sampling oftheintended audiencetries the product out.
--Itisalsoknownasfieldtesting.Ittakesplaceat customer'ssite.Itsendsthesystemtouserswhoinstallit and useit under real-world workingconditions.
--Betaisthesecondletter oftheGreek alphabet.
--Originally,theterm*alphatest*meantthefirstphaseoftestinginasoftwaredevelopmentprocess.Thefirst phaseincludesunittesting, component testing, and systemtesting.
--Betatestingcanbeconsidered"pre-releasetesting."
--Betatestingisalsosometimesreferredtoasuseracceptancetesting(UAT)orendusertesting.
--Inthisphaseofsoftwaredevelopment,applicationsaresubjectedto realworldtestingbytheintendedaudienceforthesoftware.
--The experiences of the early users are forwarded back to the developers who make final changesbefore releasingthesoftware commercially.

EntryCriteriaforBeta:
--Positiveresponsesfrom alphasite.
--Customerbugsinalpha testinghavebeenaddressed.
--Therearenofatalerrors whichcanaffectthe functionalityofthesoftware.
--Betasites arereadyfor installation.

Exitcriteria toBeta:
--Getresponse /feedbacks fromthebetatesters.
--Preparereportofallseriousbugs.
--Notifybug–fixingissues to developers.

Regressiontesting

ProgressiveVsregressivetesting,Regressiontestability,Objectivesofregressiontesting,Whenregressiontestin gdone?,Regression testingtypes,Regression testingtechniques

ProgressiveVsRegressivetesting:
--All the test case design methods or testing technique, discussed till now are referred to asprogressivetestingor development testing.
--The purpose of regression testing is to confirm that a recent program or code change has notadverselyaffectedexistingfeatures.
--Regression testing is nothing but full or partial selection of already executed test cases which arere-executedtoensure existingfunctionalities workfine.
--Thistestingisdonetomakesurethat newcode changesshouldnothavesideeffectson theexistingfunctionalities.
--Itensuresthatoldcode still worksoncethe new codechangesaredone.

NeedofRegressionTesting:
- Changeinrequirementsandcodeismodifiedaccordingtotherequirement

- Newfeatureisadded tothesoftware
- Defectfixing
- Performanceissuefix

Definition:

Regression testing is the selective retesting of a system or component to verify thatmodifications have not caused unintended effects and that the system or component still complieswithits specifiedrequirements.

RegressionTestability:

Regression testability refers to the property of a program, modification or test suite that letsit be effectively and efficiently regression-tested. We can classify a program as regression testableif most single statement modifications to the program entail(involves) rerunning a small proportionofthe current test suite.

ObjectivesofRegressionTesting:

--***It tests to check that the bug has been addressed:*** The first objective in bug fix testing is to checkwhetherthe bug-fixinghas worked or not.

--***It finds other related bugs:*** Regression tests are necessary to validate that the system does nothaveanyrelated bugs.

--***It tests to check the effect on other parts of the program****:* It may be possible that bug-fixing hasunwantedconsequencesonotherpartsofaprogram.Therefore,itisnecessarytocheckintheinfluenceofch anges in onepart orother parts oftheprogram.

Whenregressiontestingdone?Software Maintenance:

--*CorrectiveMaintenance***:**Changesmadetocorrectasystemafterafailure hasbeenobserved.

--*AdaptiveMaintenance:*Changesmadetoachievecontinuingcompatibilitywiththetargetenvironemntor other systems.

--*PerfectiveMaintenance:* Changesmadetoimproveoradd capabilities.

--*Preventive Maintenance: C*hanges made to increase robustness, maintainability, portability, andotherfeatures.

RapidIterativeDevelopment:Theextremeprogrammingapproachrequiresthatatestbedevelopedfor each classand thatthis test be re-run everytimetheclass changes.

CompatabilityAssesment and Benchmarking: Some test suites designed to be run on a widerangeofplatformsandapplicationstoestablishconformancewithastandardortoevaluatetimeandspac eperformance.

RegressionTestingTypes:

Bug-fixRegression: Thistestingisperformedafterabughasbeenreportedandfixed.

Side-EffectRegression/StabilityRegression:Itinvolvesretestingasubstantialpartoftheproduct. The goal is to prove that the changes has no detrimental effect on something that wasearlierinorder.

RegressionTestingTechniques:

Therearedifferenttechniquesforregressiontesting.Theyare:

--

>**Regressiontestselectiontechnique**:Thistechniqueattemptstoreducethetimerequiredtoretestamodifie d program byselectingsomesubset ofthe existingtest suite.

-->**Testcaseprioritizationtechnique:**Regressiontestprioritizationattemptstoreorderaregression

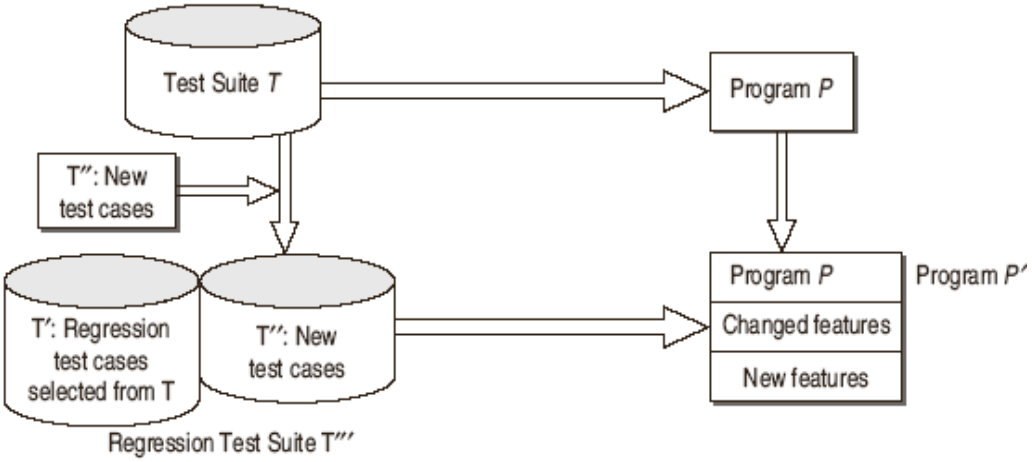
test suite so that those tests with the highest priority according to some established criteria, areexecuted earlier in the regression testing process rather than those lower priority. There are twotypesof prioritization:

(a) General Test Case Prioritization: For a given program P and test suits T, we prioritize the testcases in T that will be useful over a succession of subsequent modified versions of P, without anyknowledgeof themodified version.

*(b) Version-SpecificTest case Prioritization:*We prioritize thetestcases inT, when PismodifiedtoP',with theknowledgeofthe changesmadein P.

-->Test Suite Reduction Technique: It reduces testing costs by permanently eliminating redundanttestcases form test suitesin terms ofcodesoffunctionalitiesexercised.

SelectiveRetestTechnique:



Selective retest technique attempts to reduce the cost of testing by identifying the portions of P'(modified version of Program) that must be exercised but the regression test suite. Following are thecharacteristicfeatures ofthe selectiveretest technique:
-->It minimizestheresourcesrequiredtoregressiontestanewversion.
--> It is achievedbyminimizingthenumber oftestcases applied tothenew version.
-->Itanalysestherelationshipbetweenthetestcasesandthesoftwareelementstheycover.
-->Itusestheinformationabout changestoselecttestcases.

StepsinSelectiveretesttechnique:
1. SelectT'subsetofT,a setoftestcasestoexecuteonP'.
2. TestP'withT',establishingcorrectnessofP'withrespecttoT'.
3. Ifnecessary, createT'‖, asetofnewfunctional testcasesfor P'.
4. TestP'withT', establishingcorrectnessofP'withrespecttoT".
5. CreateT"'. anewtestsuiteandtestexecutionprofileforP',fromT,T'andT".

StrategyforTestCaseSelection:

For large software systems, there may be thousands of test cases available in its test suite. When achangeis introducedinto the systemfor next version, rerunning all thetest cases isa costly andtime consuming task. Therefore a need for selecting a subset of test cases from the original test suiteis necessary. But the use of multiple criteria should increase the code coverage. So, an effective testcaseselection strategyisto be designedbased on thecodecoverage.

SelectioncriteriabasedonCode:
-->Faultrevealingtestcases
-->ModificationrevealingTestcases.
-->ModificationtraversingTestcases.

RegressionTestSelectionTechniques:

MinimizationTechniques:Minimization-basedregressiontestselectiontechniquesattempttoselect minimal sets of test cases from T thatyield coverage of modified or affected portions of P.For example, this technique uses systems of linear equations to express relationships between testcasesandbasic blocks(single-entry,single-exit sequencesofstatementsina procedure). Thetechnique uses a 0-1 integer programming algorithm to identify a subset T' of T that ensures thatevery segment that is statically reachable from a modified segment is exercised by at least one testcaseinT9 thatalso exercises the modified segment.

Dataflow Techniques:Dataflow-coverage-basedregressiontestselectiontechniquesselecttestcases that exercise data interactions that have been affected by modifications. For example, thetechniquerequires that every definition-use pair that is deleted from P, new in P', or modified for P'be tested. The technique selects every test case in T that, when executed on P, exercised deleted ormodifieddefinition-use pairs, orexecutedastatement containingamodifiedpredicate.

Safe Techniques: Most regression test selection techniques—minimization and dataflow techniquesamong them—are not designed to be safe. Techniques that are not safe can fail to select a test casethat would have revealed a fault in the modified program. In contrast, when an explicit set of safetyconditions can be satisfied, safe regression test selection techniques guarantee that the selectedsubset,T',contains all testcases in theoriginaltestsuiteTthatcan reveal faults inP'

Ad Hoc/Random Techniques: When time constraints prohibit the use of a retest-all approach, butnotestselectiontoolisavailable,developersoftenselecttestcasesbasedon–hunches,‖orloose associations of test cases with functionality. Another simple approachisto randomly select apredeterminednumberoftestcases fromT.

Retest-All Technique: The retest-all technique simply reuses all existing test cases. To test P', thetechnique effectively–selects‖all test casesinT.

EvaluatingRegressionTestSelectionTechnique:

Inclusiveness: Let M be a regression test selection technique. Inclusiveness measures the extent towhich M chooses modification revealing tests from T for inclusion in T'. We define inclusivenessrelativeto aparticular program,modified program, and test suite,as follows:

DEFINITION
Suppose T contains n tests that are modification revealing for P and P', and suppose M selects m ofthesetests.Theinclusiveness ofM relative toP, P',andTis
1) the percentage given by the expression
(100(m/n))if n# 0 or2)100%if n=0.

For example, if T contains 50 tests of which eight are modification-revealing for P and P', and Mselects two of these eight tests, then M is 25% inclusive relative to P, P', and T. If T contains nomodification-revealing tests then every test selection technique is 100% inclusive relative to P, P'',andT.

Precision: Let M be a regression test selection technique. Precision measures the extent to which Momitstests that arenon modification-revealing. Wedefineprecisionrelativeto aparticularprogram,

modifiedprogram,andtestsuite,asfollows:

DEFINITION
SupposeTcontainsntests！thatarenonmodification-revealingforPandP'andsupposeMomitsmofthesetests.Theprecision ofMrelative to P,P: andTis
1) the percentage given by the expression (100(m/n))
if2)100%if n =0. n#0,or
For example, if T contains 50 tests of which 44 are non modification-revealing for P and P', and Momits 33 of these 44 tests, then M is 75% precise relative to P, P', and T. If T contains no non-modification-revealingtests,theneverytestselectiontechniqueis100%preciserelativetoP,P',andT.

Efficiency: We measure the efficiency of regression test selection techniques in terms of their spaceand time requirements. Where time is concerned, a test selection technique is more economical thanthe retest-all technique if the cost of selecting T' is less than the cost of running the tests in T-T'Space efficiency primarily depends on the test history and program analysis information a techniquemust store. Thus, both space and time efficiency depend on the size of the test suite that a techniqueselects,and on thecomputational cost of that technique.

RegressionTestPrioritization:

Theregressiontestprioritizationapproachisdifferentascomparedtoselectiveretesttechniques. Regression testprioritization attemptstoreordera regressiontestsuite sothat thosetests with the highest priority, according to some established criterion, are executed earlier in theregressiontestingprocess thanthose with alowerpriority.

Thesteps forthis approachare:
1. SelectT'fromT, asetoftestcasestoexecuteonP'.
2. ProduceT'$_P$,a permutationofT', suchthatT'$_P$willhavea better rateoffault detectionthanT'.
3. TestP'withT'$_P$inorderto establish thecorrectness of P'wrtT'$_P$.
4. IfnecessarycreateT",asetofnew functionalorstructuraltestsforP'.
5. TestP'withT"inordertoestablishthecorrectnessofP'wrtT".
6. CreateT"',anewtestsuiteforP',fromT,T'$_P$andT".