NARSIMHA REDDY ENGINEERING COLLEGE
UGC AUTONOMOUS INSTITUTION
Maisammaguda (V), Kompally - 500100, Secunderabad, Telangana State, India
Your roots to success....
UGC - Autonomous Institute
Accredited by NBA & NAAC with 'A' Grade
Approved by AICTE
Permanently affiliated to JNTUH

School of Computer Science

SOFTWARE ENGINEERIN PPTS

FACULTY: Dr.VenkateswaruluNaik

CS3102PC:SOFTWAREENGINEERING

III YEARB.TECH. CSEI -SEM (R21)

UNIT-I

INTRODUCTIONTOSOFTWAREENGINEERING

Software:

- collectionofintegratedprograms.
- carefully-organized instructions that provide desired features, function, andperformance,whenexecuted.
- Computerprogramandrelateddocumentationsuch asrequirements, designmodelsandusermanuals.

CharacteristicsofSoftware:

- Software is developed orengineered;itisnotmanufacturedinthe classicalsense.
- Softwaredoes not “wearout”(notsusceptibletoenvironmenteffects).
- Reusabilityofcomponents.

Engineering:

- application of scientific and practical knowledge to invent, design, build,maintain,andimproveframeworks,processes,etc.

SoftwareEngineering:

- Thesystematic,disciplinedquantifiableapproachtothedevelopment,operationandmaintenanceofsoftware.
- includes activitiestomanagetheproject,developtools,methodsandtheoriesthatsupportthesoftwareproduction.
- providesastandardproceduretodesignanddevelopasoftware.

EVOLVINGROLEOFSOFTWARE:

- Softwaretakesdualrole.
- **product**:
  - Itdeliversthecomputingpotentialembodiedbycomputer Hardwareorbya network ofcomputers.
- **vehicle**:
  - Itprovidessystemfunctionality(e.g.,payrollsystem)
  - Itcontrolsothersoftware(e.g.,anoperatingsystem)
  - Ithelpsbuildothersoftware(e.g.,softwaretools)

Evolution:

- **1970sand1980s**
- **1990sbegan**
- **Mid-1990s**
- **Later1990s**
- **2000sprogressed**
- Todayahugesoftwareindustryhasbecomeadominantfactorintheeconomiesoftheindustrialized world.

THECHANGINGNATURE OFSOFTWARE:

- Systemsoftware
- Applicationsoftware
- Engineering/scientificsoftware
- Embeddedsoftware
- Product-linesoftware
- Web-applications
- Artificialintelligencesoftware.

Systemsoftware:

- collectionofprogramswritten toserviceotherprograms.
- *E.g.*compilers, editorsand filemanagementutilities.

Applicationsoftware:

- standaloneprogramsthatsolveaspecificbusinessneed.
- *E.g.*point-of-saletransaction processing,real-timemanufacturingprocesscontrol.

Engineering/Scientificsoftware:

- fromastronomytovolcanology
- *E.g.*computeraideddesign,systemsimulationandotherinteractiveapplications.

Embedded software:

- resideswithin aproductorsystem
- *E.g.*Digitalfunctionsin automobile, dashboarddisplays,braking systemsetc.

Product-linesoftware:

- provideaspecificcapabilityforuseby manydifferentcustomers
- *E.g.*Word processing,spreadsheets,computergraphics,multimedia,entertainment,databasemanagement, personalandbusinessfinancialapplications

Web-applications:

- evolvingintosophisticatedcomputingenvironmentsthatnotonlyprovidestandalonefeatures,computingfunctions,andcontenttotheenduser,butalsoareintegratedwithcorporate databasesand businessapplications.

Artificialintelligencesoftware:

- makesuseofnonnumericalalgorithmstosolvecomplexproblemsthatarenotamenabletocomputationorstraightforwardanalysis.
- *E.g.*robotics,expertsystems,patternrecognition,artificialneuralnetworks,theoremproving,andgame playing.

Newchallengesonthehorizon:

1. Ubiquitouscomputing
2. Netsourcing

NRCM

SOFTWAREMYTHS

Managementmyths:

- Manageswithsoftwareresponsibility,likemanagersinmostdisciplines,areoftenunderpressure tomaintainbudgets,keepschedulesfromslipping,and improvequality.

Customermyths:

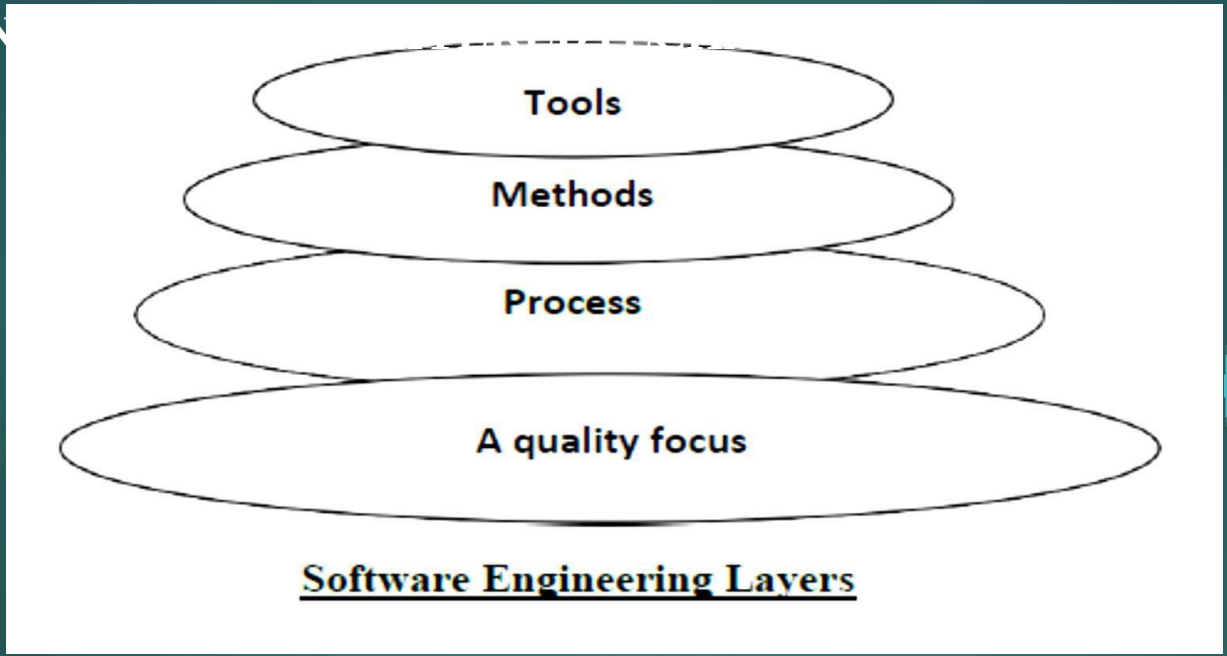
- Thecustomerbelievesmythsaboutsoftwarebecausesoftwaremanagersandpractitionersdolittletocor rectmisinformation.Mythsleadtofalseexpectationsandultimately,dissatisfactionwiththe developer.

Practitioner’smyths:

- Mythsthatarestillbelievedbysoftwarepractitioners:duringtheearlydaysofsoftware,programmingw asviewedasanartfromoldwaysandattitudesdiehard.

AGENERICVIEWOFPROCESS

SOFTWARE EN

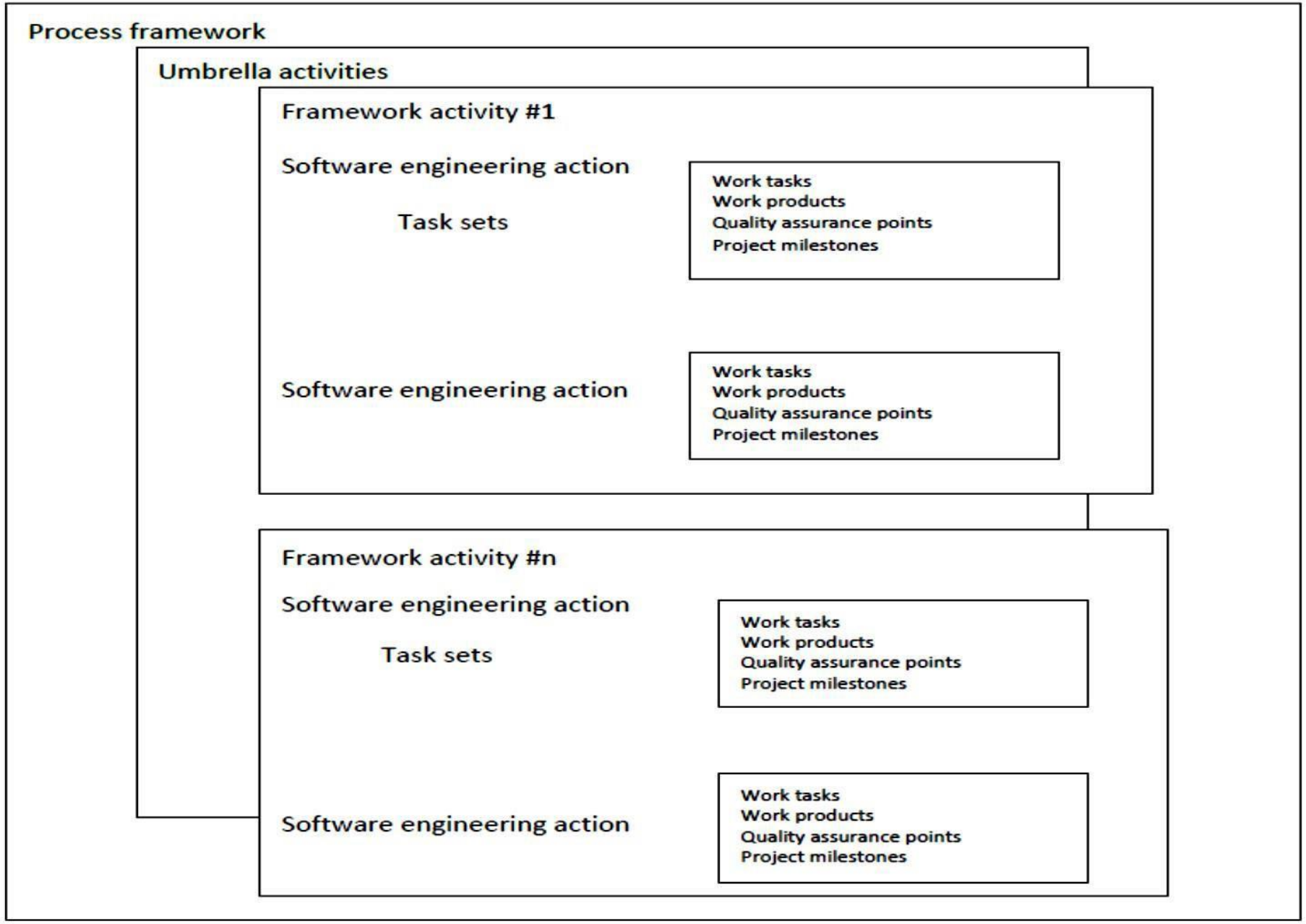


Software Engineering Layers

- The bedrockthatsupportssoftwareengineering is a qualityfocus.
- Thefoundationforsoftwareengineeringis theprocess layer.
- Software engineeringmethods relyona set ofbasicprinciplesthatgovernarea ofthe technologyandincludemodelingactivities.
- Software engineeringtoolsprovideautomatedorsemi-automatedsupport fortheprocessandthemethods.

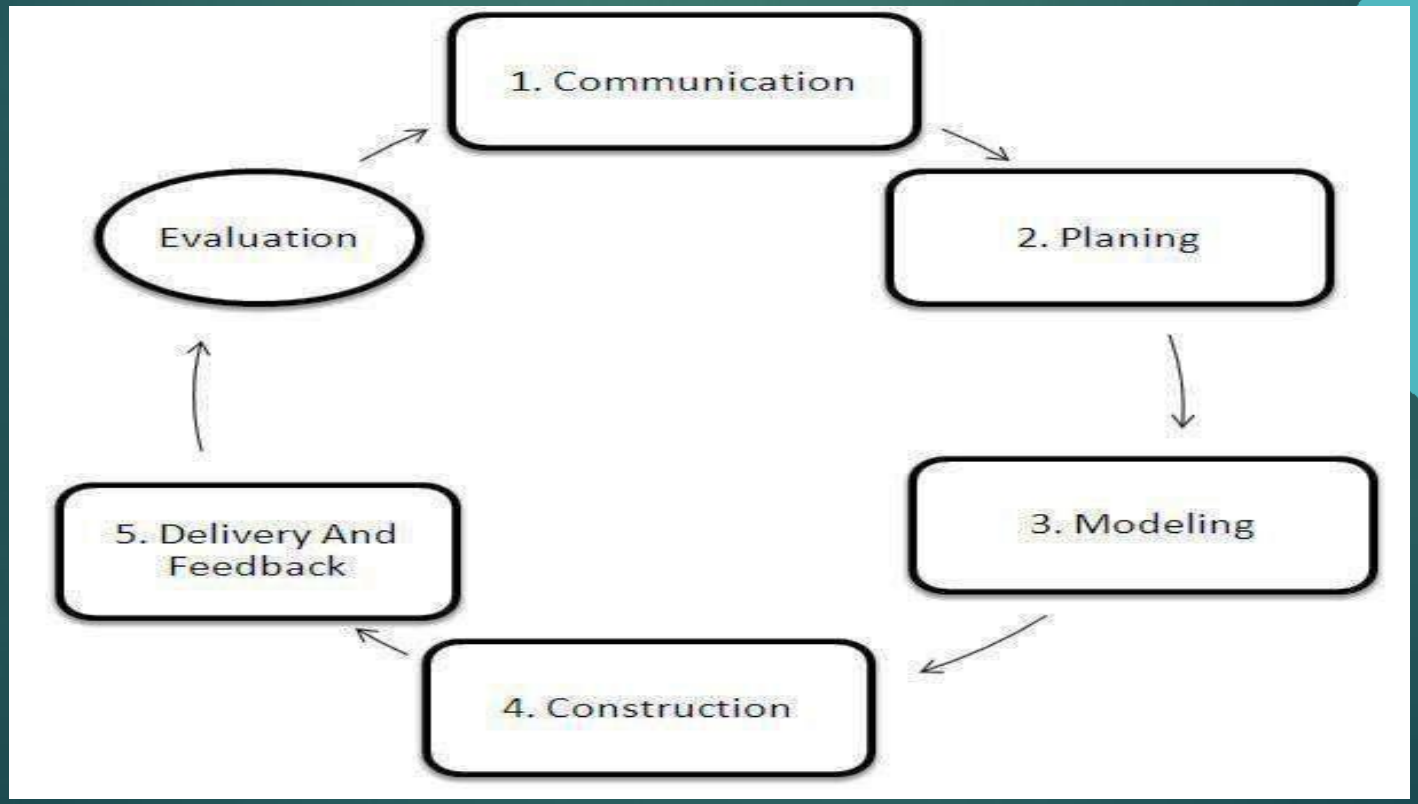
APROCESSFRAMEWORK:

- Aprocessdefineswhoisdoingwhat,when,andhowto reachacertaingoal.
- AProcessFramework establishesthe foundationfora complete softwareprocess
- identifiesasmallnumberof**frameworkactivities**
- also,setof**umbrellaactivities**
- Eachframeworkactivityhassetofs/wengineeringactions.
- Eachs/wengineeringaction(e.g., design)has
  - collectionofrelatedtasks(calledtasksets):
    - worktasks
    - workproducts(deliverables)
    - qualityassurancepoints
    - projectmilestones.


Process framework
Umbrella activities
Framework activity #1
Software engineering action
Task sets
Work tasks
Work products
Quality assurance points
Project milestones
Software engineering action
Work tasks
Work products
Quality assurance points
Project milestones
Framework activity #n
Software engineering action
Task sets
Work tasks
Work products
Quality assurance points
Project milestones
Software engineering action
Work tasks
Work products
Quality assurance points
Project milestones

GenericProcessFramework:

- Itisapplicabletothevastmajorityofsoftwareprojects.
- These5genericframeworkactivitiescanbeusedduringthedevelopmentofsmallprograms,thecreationoflargewebapplications,andfortheengineeringoflarge,complexcomputer-basedsystems.



Communication:

- involvescommunicationandcollaborationwiththecustomerandencompassesrequirementsgatheringandotherrelated activities.

Planning:

- establishesaplanforthesoftwareengineeringworkthatfollows.Itdescribesthetechnicaltaskstobeconducted, theriskthatarelikely,theresourcesthatwillberequired,theworkproductstobeproduced, anda workschedule.

Modeling:

- encompassesthecreationofmodelsthatallowthedeveloperandcustomertobetterunderstandsoftwarerequirements
- Themodelingactivityis composedof2softwareengineeringactions-analysisand design.
  - Analysisencompasses a setofworktasks.
  - Design encompassesworktasksthatcreateadesign model.

Construction:

- combinescore generationandthe testingthatisrequiredtouncoverthe errorsinthecode.

Deployment:

- Thesoftwareis deliveredto the customerwhoevaluatesthedeliveredproductand providesfeedback

your tools to success

Thefollowingare thesetof**UmbrellaActivities.Softwareprojecttracking and control**

- assessprogressagainsttheprojectplan and takenecessary action to maintainschedule.

RiskManagement

- assessesrisks thatmayeffectthe outcomeofthe projector thequalityofthe product.

SoftwareQualityAssurance

- definesand conductstheactivitiesrequiredtoensuresoftwarequality.

FormalTechnicalReviews

- assessessoftwareengineeringwork productsinan efforttouncoverandremoveerrors.

Measurement

- defineandcollectsprocess,projectandproductmeasuresthatassisttheteamindeliveringsoftwarethatneeds customer’sneeds

Softwareconfiguration management

- managestheeffectsofchangethroughoutthesoftwareprocess.

Reusabilitymanagement

- definescriteriaforwork productreuseand establishesmechanisms to achievereusablecomponents.

WorkProductpreparationandproduction


NRCM
your tools to success

THECAPABILITYMATURITYMODELINTEGRATION(CMMI):

- TheCMMIrepresentsaprocessmeta-modelin two differentways:
  - Asacontinuousmodel
  - Asastaged model.

Eachprocess area isformallyassessedandis ratedaccording tothe followingcapabilitylevels.

- **Level0:Incomplete**
- **Level1:Performed**
- **Level2:Managed**
- **Level3:Defined**
- **Level4:Quantitativelymanaged**
- **Level5:Optimized**

TheCMMIdefineseachprocessareaintermsof“specificgoals”andthe“specificpractices”requiredtoachieve these goals.

The specific goals(SG)andtheassociated**specificpractices(SP)**definedforprojectplanning are

SG1Establishestimates

- SP1.1Estimatethescopeoftheproject
- SP1.2 Establish estimates ofwork productandtaskattributes
- SP1.3Defineprojectlifecycle
- SP1.4 Determineestimatesof effortandcost

SG2Develop aProjectPlan

- SP2.1Establishthebudgetandschedule
- SP2.2Identifyprojectrisks
- SP2.3Planfordatamanagement
- SP2.4Planforneeded knowledgeandskills
- SP2.5Planstakeholderinvolvement
- SP2.6Establishtheprojectplan

SG3Obtaincommitmenttothe plan

- SP3.1Reviewplansthataffecttheproject
- SP3.2Reconcileworkandresourcelevels
- SP3.3Obtainplancommitment

- Inadditiontospecificgoalsandpractices,theCMMIalsodefinesasetoffivegenericgoalsandrelated practicesforeachprocessarea.
- Each ofthefivegenericgoalscorrespondstooneofthefivecapabilitylevels.

GG 1Achieve specificgoals

- GP1.1Performbasepractices

GG2 Institutionalizeamanagedprocess

- GP2.1Establish andorganizationalpolicy
- GP2.2Plantheprocess
- GP2.3Provideresources
- GP2.4Assignresponsibility
- GP2.5Trainpeople
- GP2.6Manageconfigurations
- GP2.7Identifyand involverelevantstakeholders
- GP2.8Monitorand controltheprocess
- GP2.9Objectivelyevaluateadherence
- GP2.10 Reviewstatus withhigherlevelmanagement

GG3 Institutionalizeadefinedprocess

- GP3.1Establish adefinedprocess
- GP3.2Collectimprovementinformation

GG 4 Institutionalizeaquantitativelymanaged process

- GP4.1 Establishquantitativeobjectivesforthe process
- GP4.2Stabilizesub processperformance

GG5 Institutionalizeandoptimizingprocess

- GP5.1Ensurecontinuousprocessimprovement
- GP5.2 Correctrootcausesofproblems

PROCESSPATTERNS:

- Thesoftwareprocesscanbedefinedasacollectionpatternsthatdefinesasetofactivities,actions,worktasks,workproductsand/orrelatedbehaviorsrequiredtodevelopcomputersoftware.
- provides us with a template - a consistent method for describing an importantcharacteristicofthe software process.

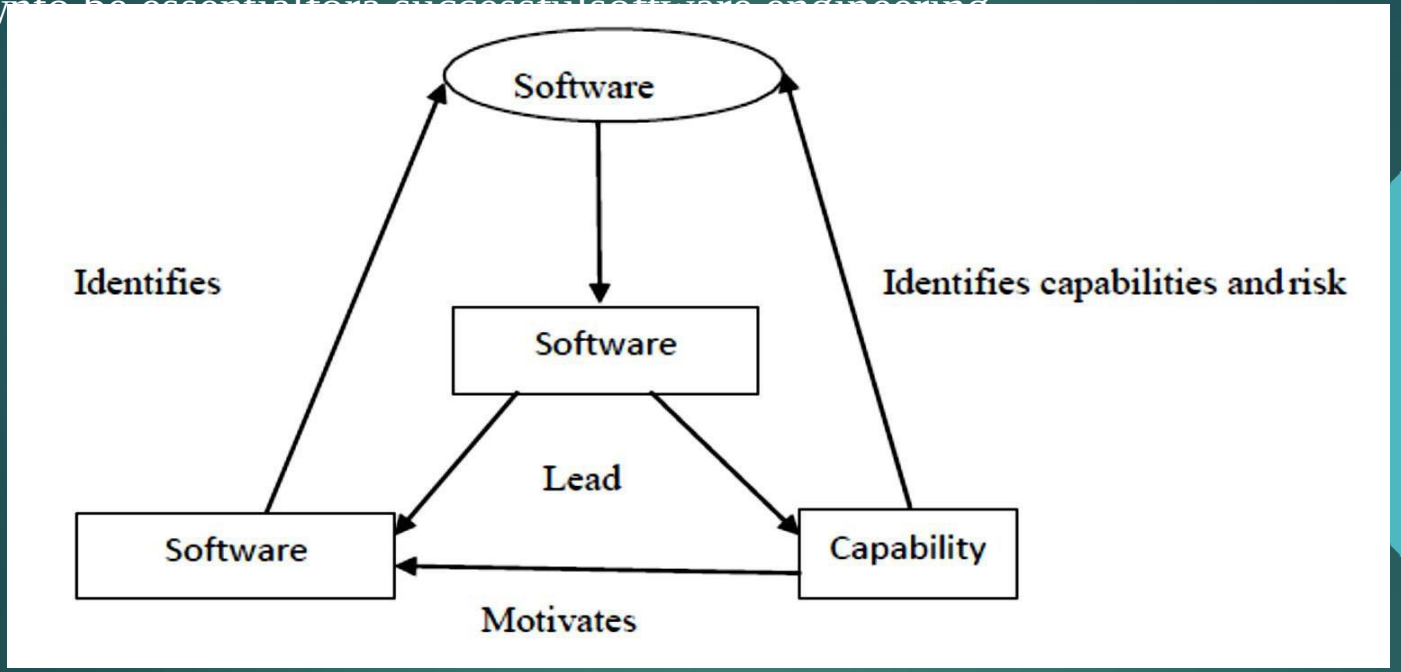
Apatternmightbeusedto describe a completeprocess andataskwithin a frameworkactivity.

- PatternName
- Intent
- Type:3types
  - **Taskpatterns**define asoftwareengineeringactionor worktaskthatispart oftheprocessandrelevanttosuccessfulsoftwareengineeringpractice.
    - *Example:*Requirement Gathering
  - **Stage Patterns**definea framework activityforthe process.Thispatternincorporatesmultipletaskpatternsthatarerelevanttothe stage.
    - *Example:*Communication
  - **Phase patterns**define the sequence offrameworkactivities thatoccurwiththeprocess,evenwhenthe overallflow ofactivitiesis iterativeinnature.
    - *Example:*Spiralmodelorprototyping.
- InitialContext
- Problem
- Solution
- ResultingContext


NRCM
your tools to success

PROCESSASSESSMENT

- the processitselfshouldbeassessedtobe essentialtoensurethatitmeets a set ofbasicprocesscriteriathathave beenshown



- Numberofdifferentapproachestosoftwareprocessassessmenthavebeenproposed,
  - StandardsCMMIAssessmentMethodforProcessImprovement(SCAMPI)
  - CMMBasedAppraisalforInternalProcessImprovement(CBAIPI)
  - SPICE(ISO/IEC15504)
  - ISO9001:2000forSoftware.

PERSONALANDTEAMPROCESSMODELS:

- Eachsoftwareengineerwouldcreateaprocessthatbestfitshisorherneeds,andatthesametimemeetsthebroader needsoftheteamandtheorganization.

Personalsoftwareprocess(PSP)

- emphasizespersonalmeasurementofboththeworkproductthatisproducedandtheresultantqualityofthework product.
- ThePSPprocessmodeldefinesfiveframeworkactivities
  - **Planning:**Thisactivityisolatesrequirementsand allthemetricsarerecorded onworksheetsorte mplates.Finally,developmenttasksare identified andaprojectscheduleis created.
  - **Highleveldesign:**Externalspecificationsforeachcomponenttobeconstructedaredevelopedandacomponentdes ignis created.Allissuesare recordedand tracked.
  - **Highleveldesignreview:**Formalverification methodsare applied touncovererrorsin thedesign.
  - **Development:**Thecomponentleveldesignisrefinedandreviewed.Codeisgenerated,reviewed,compiled,

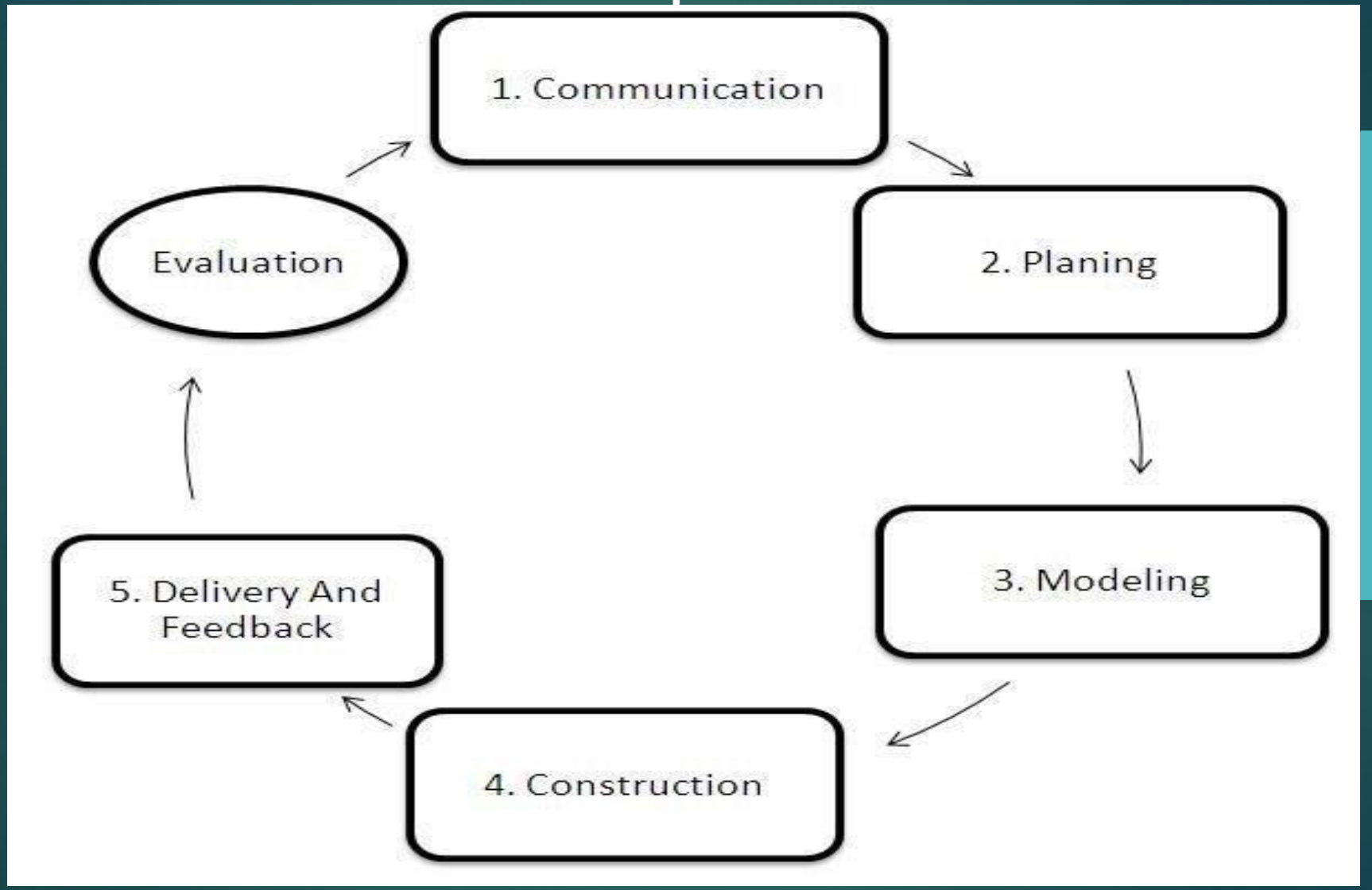
Teamsoftwareprocess(TSP)

- ThegoalofTSPistobuilda“self-directedprojectteamthatorganizesitselftoproducehigh-qualitysoftware.
- TSPdefinesthefollowingframework activities:
  - **launch**
  - **high-leveldesign**
  - **implementation**
  - **integrationandtest**
  - **postmortem.**

PROCESSMODELS:

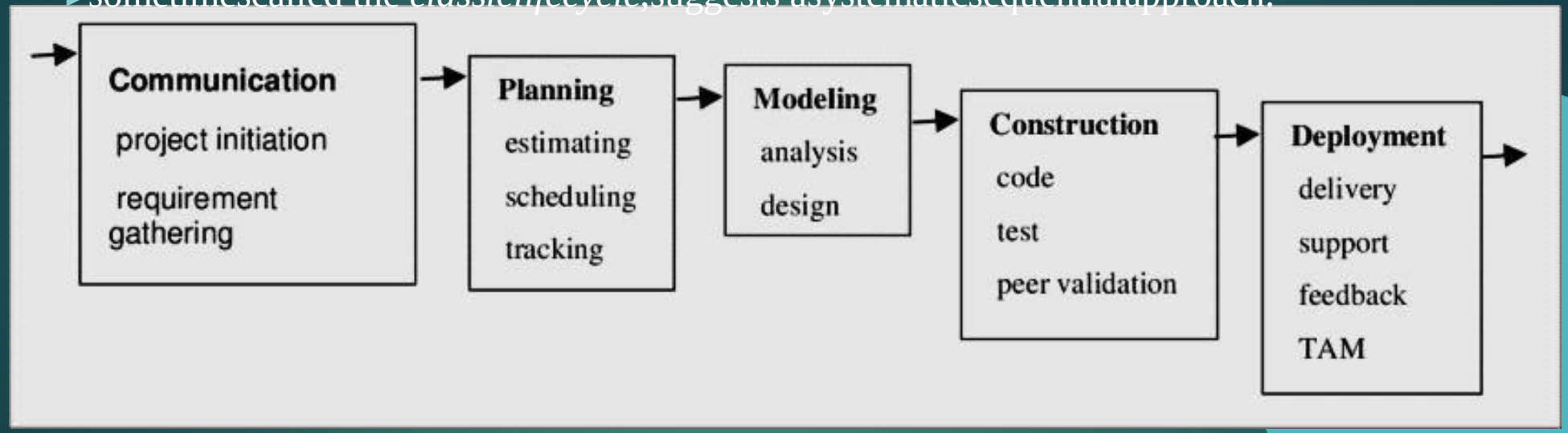
- **Prescriptiveprocessmodels**defineasetofactivities,actions,tasks,milestones,andworkproductsthatare required toengineerhigh-qualitysoftware.
- SoftwareDevelopmentLifeCycle(SDLC)isaprocessusedbythesoftwareindustrytodesign,develop andtesthighqualitysoftware.

SDLCphases


1. Communication
2. Planing
3. Modeling
4. Construction
5. Delivery And Feedback
Evaluation

THEWATERFALLMODEL:

- sometimescalled the *classiclifecycle*,suggests asystematicsequentialapproach.



Adv:

- usefulprocessmodelwhererequirementsarefixedandworkistoproceedtocompleteinalinearmanner.

Disadv:

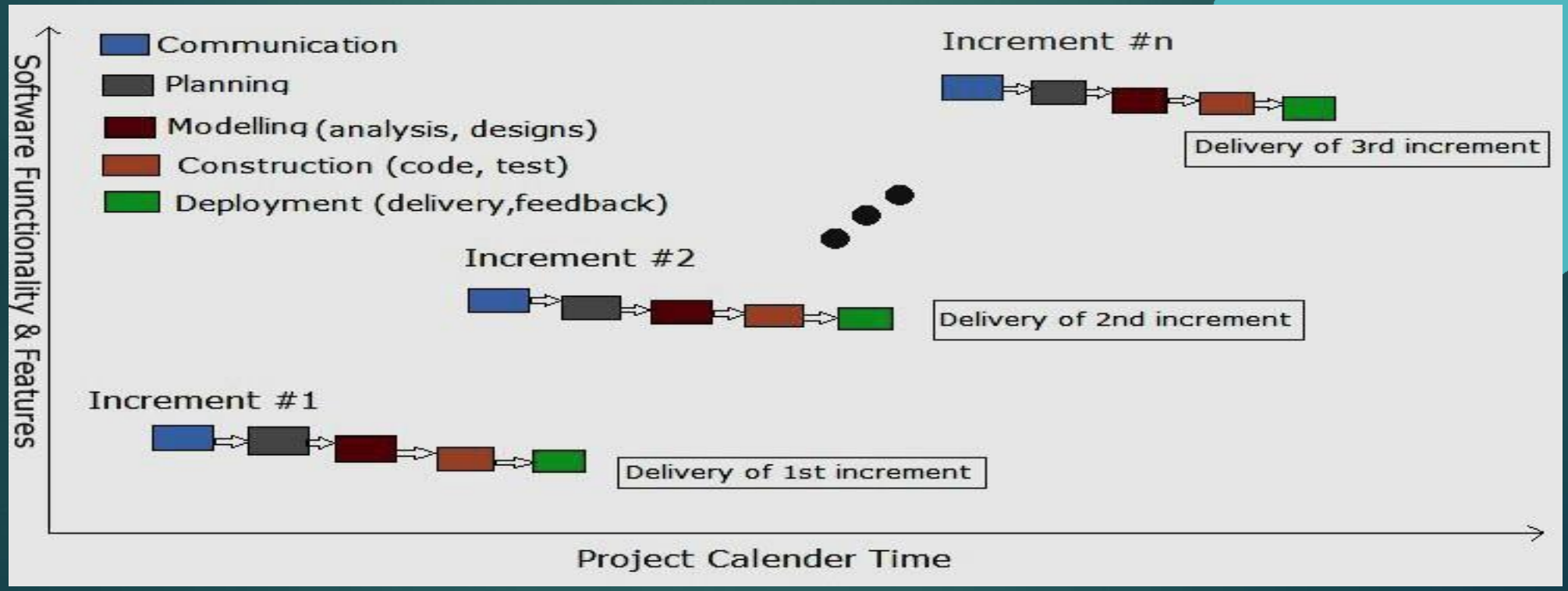
- changescan causeconfusion astheprojectteamproceeds.
- Itisoftendifficultforthecustomertostateallrequirementsexplicitly.
- Aworking versionoftheprogramswillnotbe availableuntillateintheprojecttime-span.

INCREMENTALPROCESSMODELS:

- Theincrementalmodel
- TheRAD model

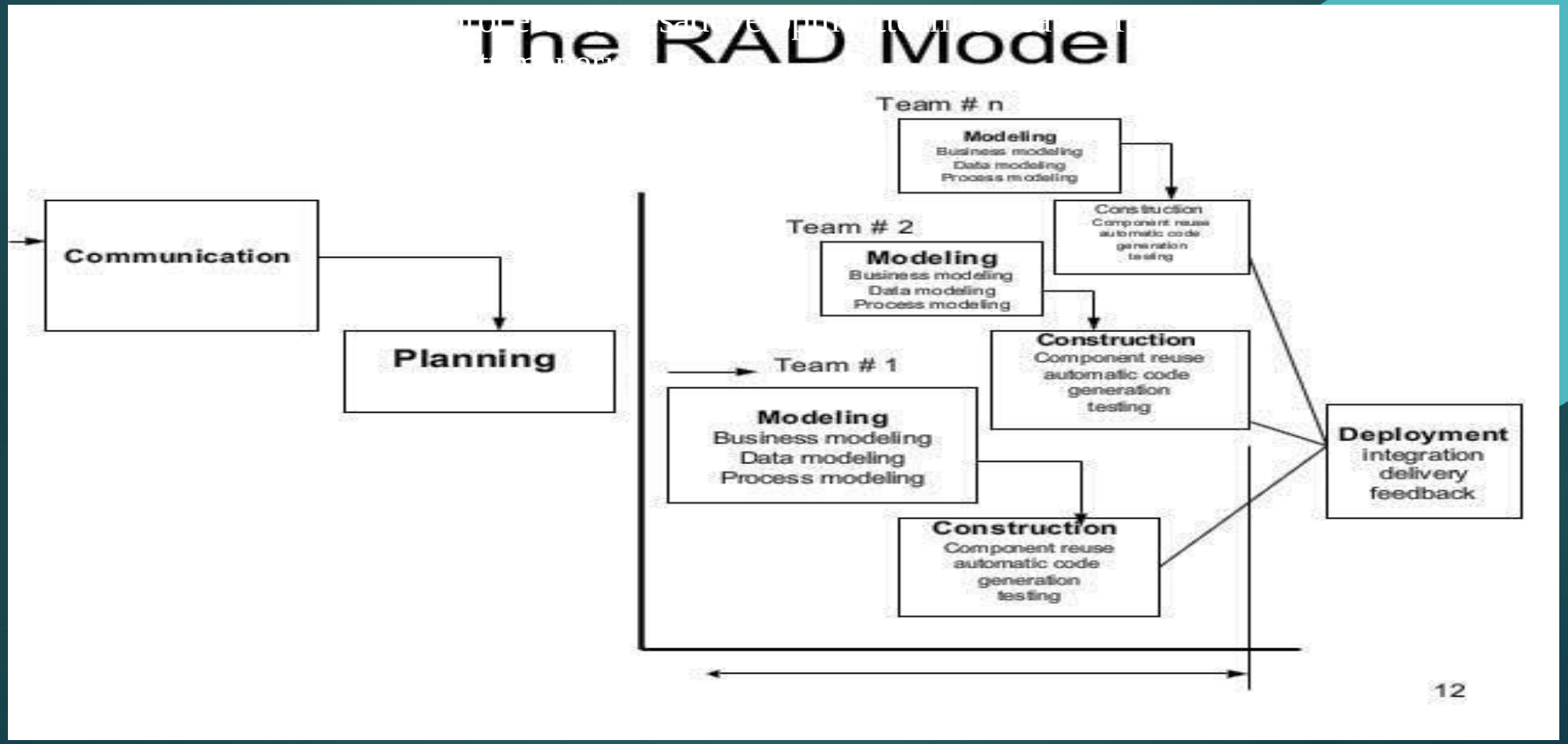
THEINCREMENTALMODEL:

- Incrementaldevelopmentis particularlyusefulwhenstaffing is unavailablefor a completeimplementationbythe businessdeadlinethathasbeenestablished fortheproject.



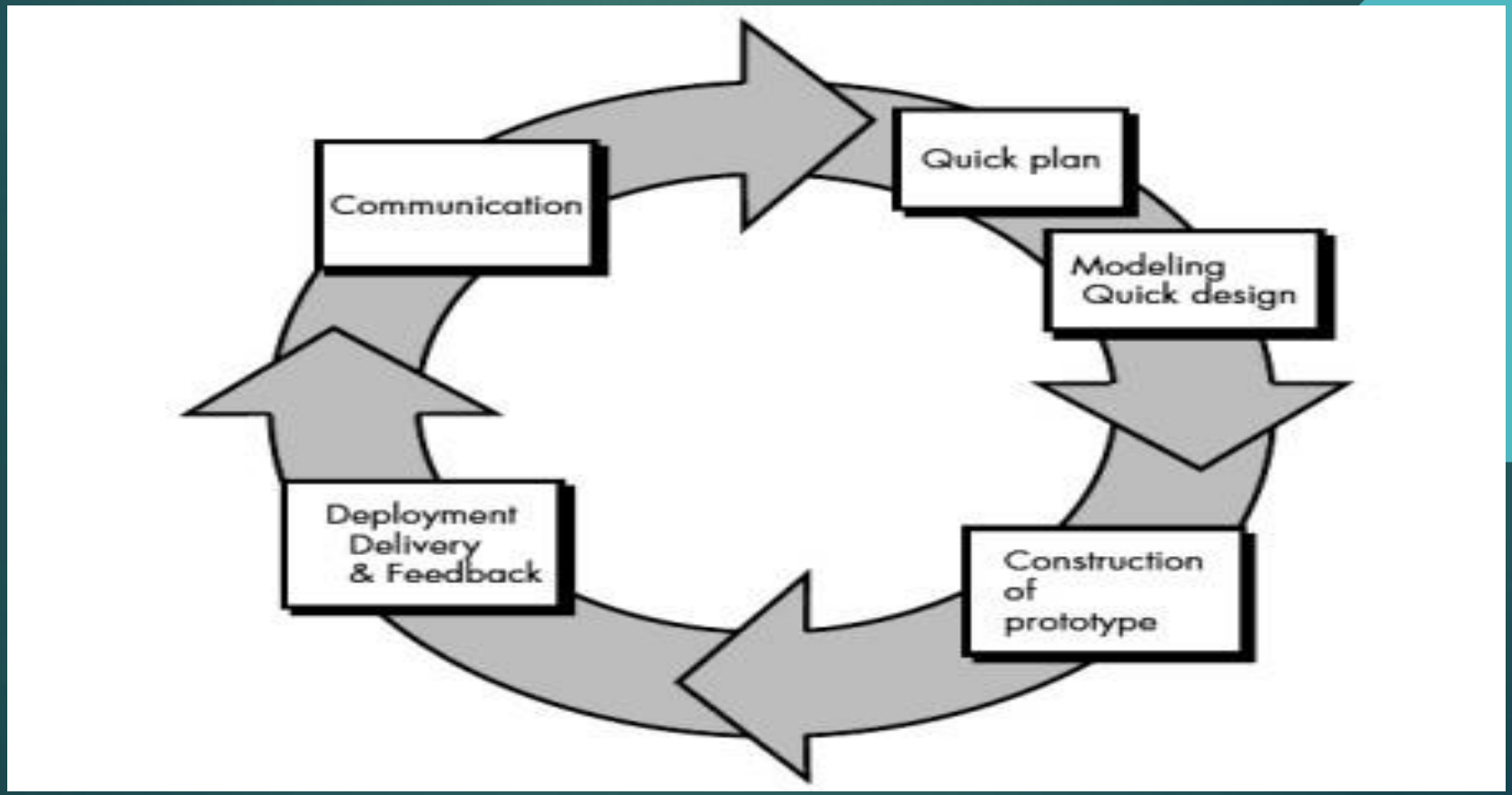
THERAD MODEL:

- a“high-speed”adaptionofthewaterfallmodel,inwhichrapiddevelopmentisachievedbyusinga componentbaseconstructionapproach.
- Ifrequirementsarewellunderstoodand projectscope



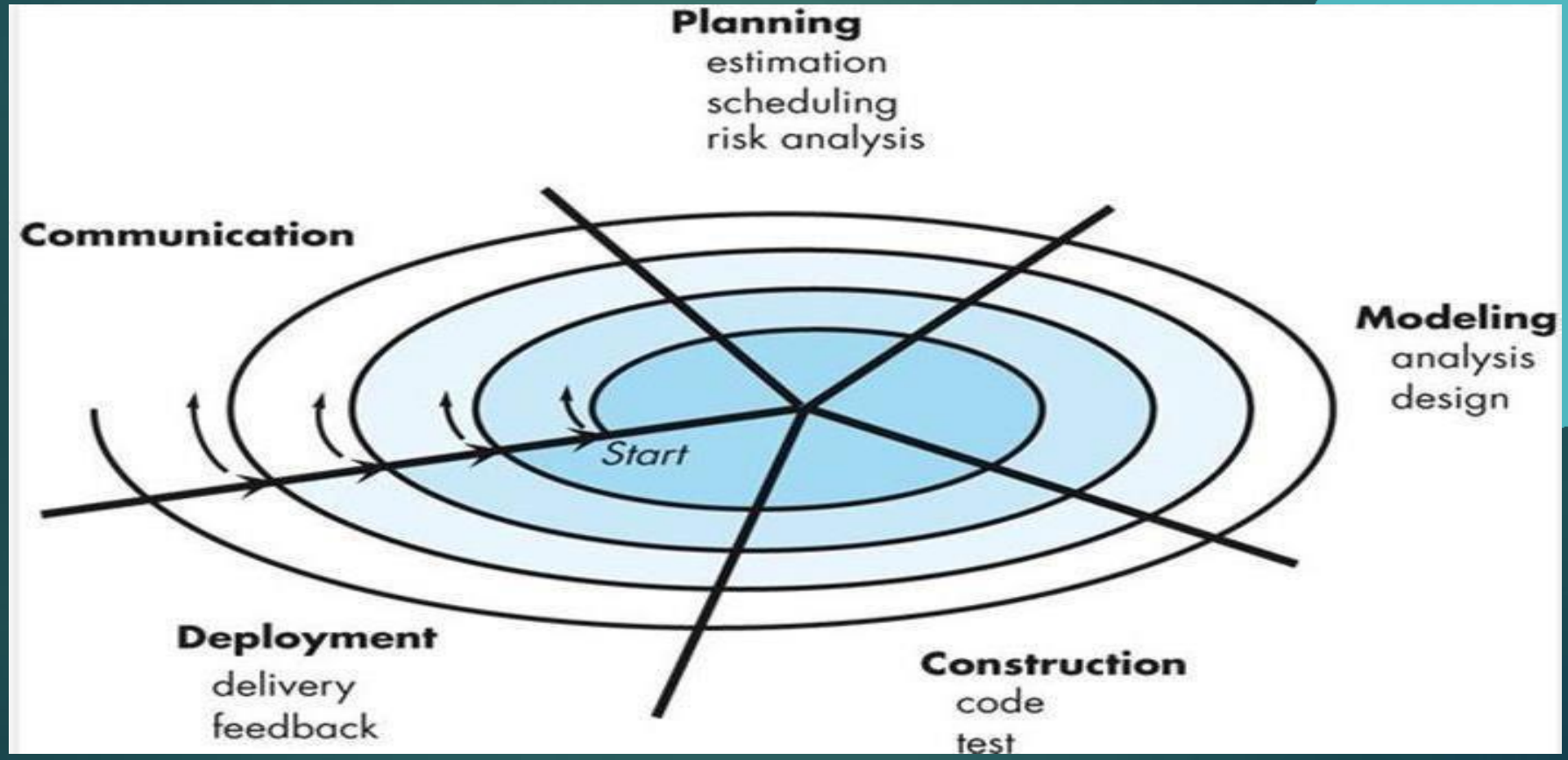
EVOLUTIONARYPROCESSMODELS:PROTOTYPING:

- can beimplementedwithinthecontextofanyoneoftheprocessmodel.
- Prototypingiterationisplannedquicklyandmodelingoccurs.Thequickdesignleadstotheconstructionofa prototype.The prototypeis deployed andthen evaluatedbythe customer/user.



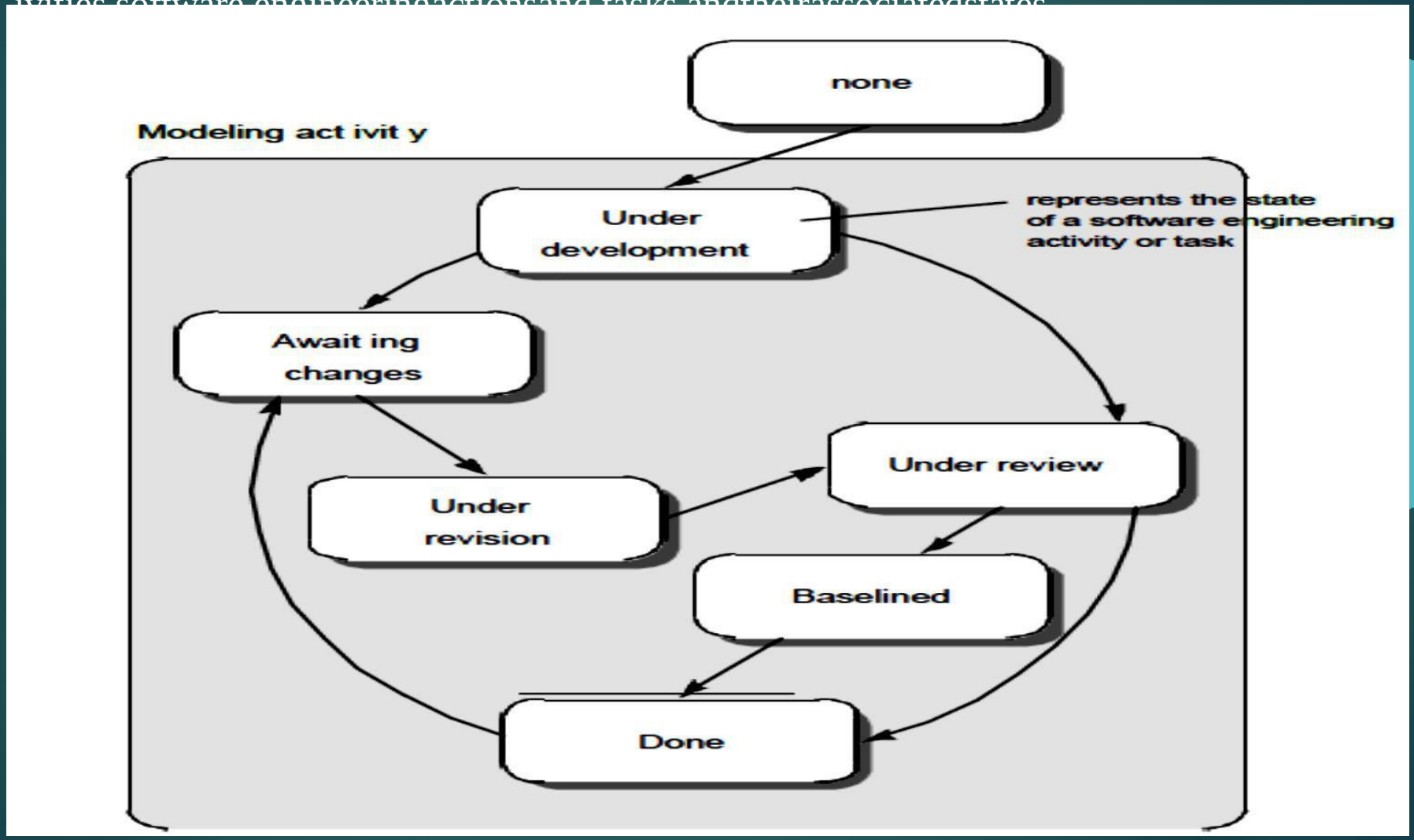
THESPIRALMODEL

- couplestheiterativenatureof prototypingwiththecontrolledandsystematicaspectsofthewaterfallmodel.
- adaptedtoapplythroughouttheentirelifecycleofanapplication,fromconceptdevelopmenttomaintenance.



THECONCURRENTDEVELOPMENTMODEL:

- sometimescalled*concurrentengineering*,canberepresentedschematicallyasaseriesoframeworkact ivities,softwareengineeringactionsand tasks,andtheirassociatedstates.

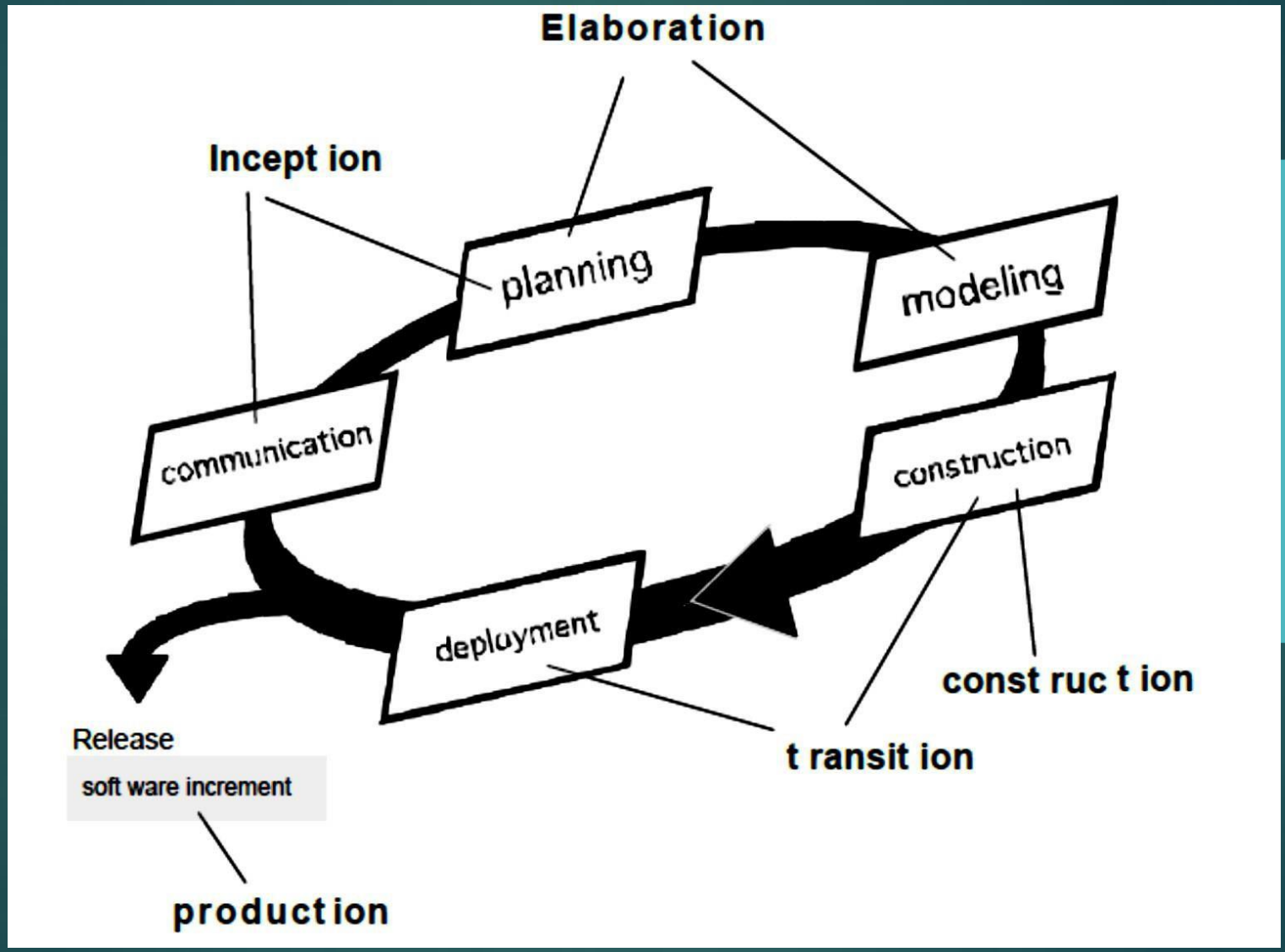


THEUNIFIEDPROCESS:

- implementsmanyofthebestprinciplesofagilesoftwaredevelopment.
- recognizestheimportanceofcustomercommunicationandstreamlinedmethodsfordescribingthe customer'sviewofa system.

PHASESOFTHEUNIFIEDPROCESS:

- The***inception***phase -encompasses bothcustomercommunicationandplanningactivities.
- The***elaboration***phase-encompassesthecustomercommunicationandmodelingactivitiesofthe genericprocessmodel.
- The***construction***phase-identicaltotheconstructionactivitydefinedforthegenericsoftwareprocess.
- The***transition***phase-encompassesthelatterstagesofthegenericconstructionactivityandthefirstpartofthe genericdeploymentactivity.
- The***production*** phase -coincideswiththedeploymentactivityofthe genericprocess.


Elaboration
Incept ion
planning
modeling
communication
construction
deployment
const ruc t ion
Release
soft ware increment
t ransit ion
product ion

NARASIMHAREDDYENGINEERINGCOLLEGE

FACULTY:Dr.VenkateswaruluNaik

CS3102PC:SOFTWAREENGINEERING

III YEARB.TECH. CSE I-SEM (R20)

- **SoftwareRequirements:**Functionalandnon-functionalrequirements,userrequirements,systemrequirements,interfacespecification,thesoftwarerequirements document.

- **Requirementsengineeringprocess:**Feasibilitystudies,requirementselicitationandanalysis,requirementsvalidation,requirementsmanagement.

- **Systemmodels:**Contextmodels,behavioralmodels,datamodels,objectmodels,structuredmethods.

SOFTWAREREQUIREMENTS

Softwarerequirementsarenecessary

- Tointroducetheconceptsofuserand systemrequirements
- Todescribefunctionalandnon-functionalrequirements
- To explainhowsoftwarerequirementsmaybeorganized

Whatis arequirement?

- thedescriptionoftheservicesprovidedbythesystemanditsoperationalconstraints
- Itmayrangefromahigh-levelabstractstatementofaserviceorofasystemconstrainttoadetailed mathematicalfunctionalspecification

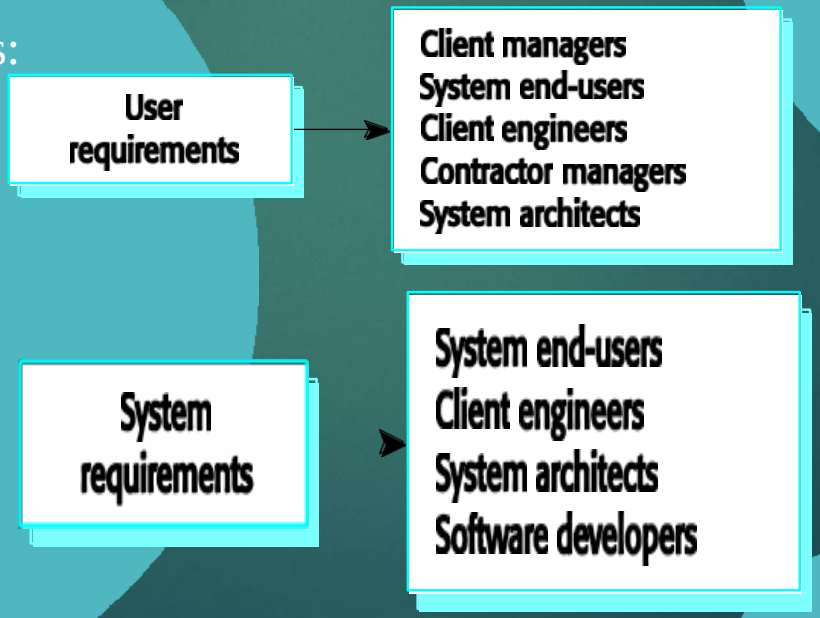
Requirementsengineering:

- The processoffindingout,analysingdocumentingandcheckingtheseservices andconstraints.
- Therequirementsthemselvesarethedescriptionsofthesystemservicesandconstraintsthataregeneratedduringtherequirementsengineeringprocess

Typesofrequirement:

- **Userrequirements**
  - Statementsinnaturallanguageplusdiagramsoftheservicesthesystemprovides anditsoperationalconstraints.Writtenforcustomers.
- **Systemrequirements**
  - Astructureddocumentsettingoutdetaileddescriptionsofthesystem's functions, services andoperationalconstraints.
  - Defineswhatshouldbeimplementedsomaybepartofa contractbetween clientandcontractor.

Requirementsreaders:



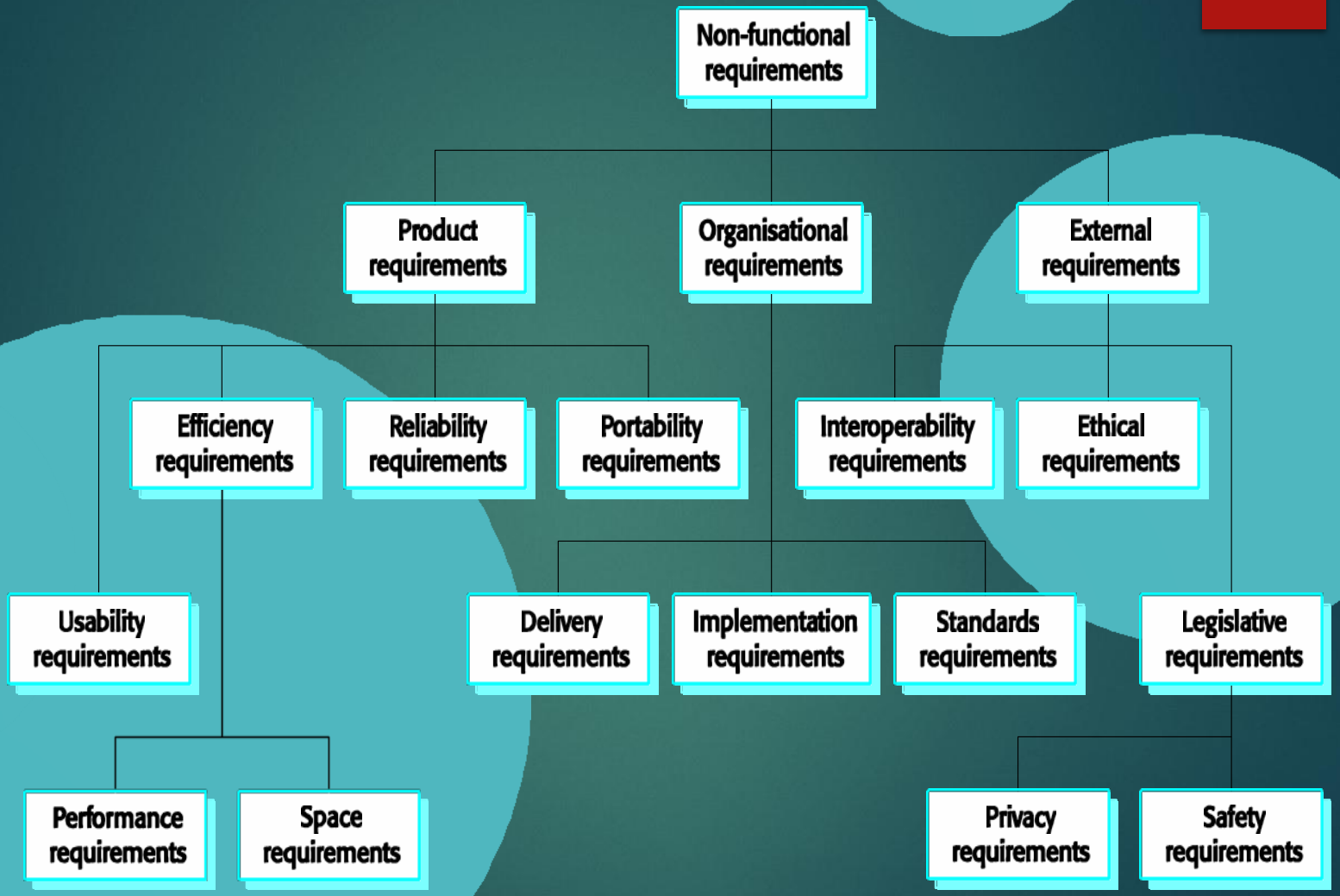
Functionalandnon-functionalrequirements:

- **Functionalrequirements**
  - Statementsofservicesthesystemshouldprovidehowthesystemshouldreacttoparticularinputsandhowthesystemshouldbehaveinparticularsituations.

  Examplesoffunctionalrequirements

  - Theusershallbeabletosearcheitherallofftheinitialsetofdatabasesorselectasubsetfromit.
  - Thesystemshallprovideappropriateviewersfortheusertoreaddocumentsinthedocumentstore.
  - Everyordershallbeallocatedauniqueidentifier(ORDER_ID)whichtheusershallbeable to copytotheaccount'spermanentstoragearea.
- **Non-functional requirements**
  - Constraintsontheservicesorfunctionsofferedbythesystemsuchastimingconstraints,constraintson the developmentprocess,standards,etc.
  - Non-functional requirements may be more critical than functionalrequirements.Ifthese are notmet,the systemis useless.

Non-functionalrequirementtypes:



Requirementsmeasures:

Property	Measure
Speed	Processedtransactions/secondUser/Event responsetimeScreenrefreshtime
Size	MBytes NumberofROM chips
Easeof use	Trainingtime Numberofhelpframes
Reliability	Mean timetofailure Probabilityof unavailabilityRate offailureoccurrenceAvailability
Robustness	Timeto restartafter failurePercentageofeventscausingfailure Probabilityof data corruptiononfailure
Portability	Percentageoftargetdependentstatements Numberoftargetsystems

- **Domainrequirements**
  - Requirementsthatcomefromtheapplicationdomainofthesystemandthatreflectcharacteristicsofthatdomain.

USERREQUIREMENTS

Userrequirementsaredefinedusingnaturallanguage,tablesanddiagramsasthes ecanbeunderstoodbyallusers.

Problemswithnaturallanguage

- Lackofclarity
  - Precision isdifficultwithoutmakingthedocumentdifficultto read.
- Requirementsconfusion
  - Functional and non-functionalrequirementstendtobemixed-up.
- Requirementsamalgamation
  - Severaldifferentrequirementsmaybeexpressedtogether.
- **Guidelinesforwritingrequirements**
  - Inventastandardformatanduseitforallrequirements.
  - Use language in a consistent way. Use shall for mandatoryrequirements,should fordesirablerequirements.
  - Use texthighlightingto identifykeypartsofthe requirement.
  - Avoidtheuseofcomputerjargon.

SYSTEMREQUIREMENTS

- Moredetailedspecificationsofsystemfunctions,servicesandconstraintsthanuserrequirements.
- Theyare intended to beabasisfordesigningthesystem.
- Theymaybeincorporatedintothesystemcontract.
- Systemrequirementsmaybedefinedorillustratedusingsystemmodels.

ProblemswithNL(naturallanguage)specification

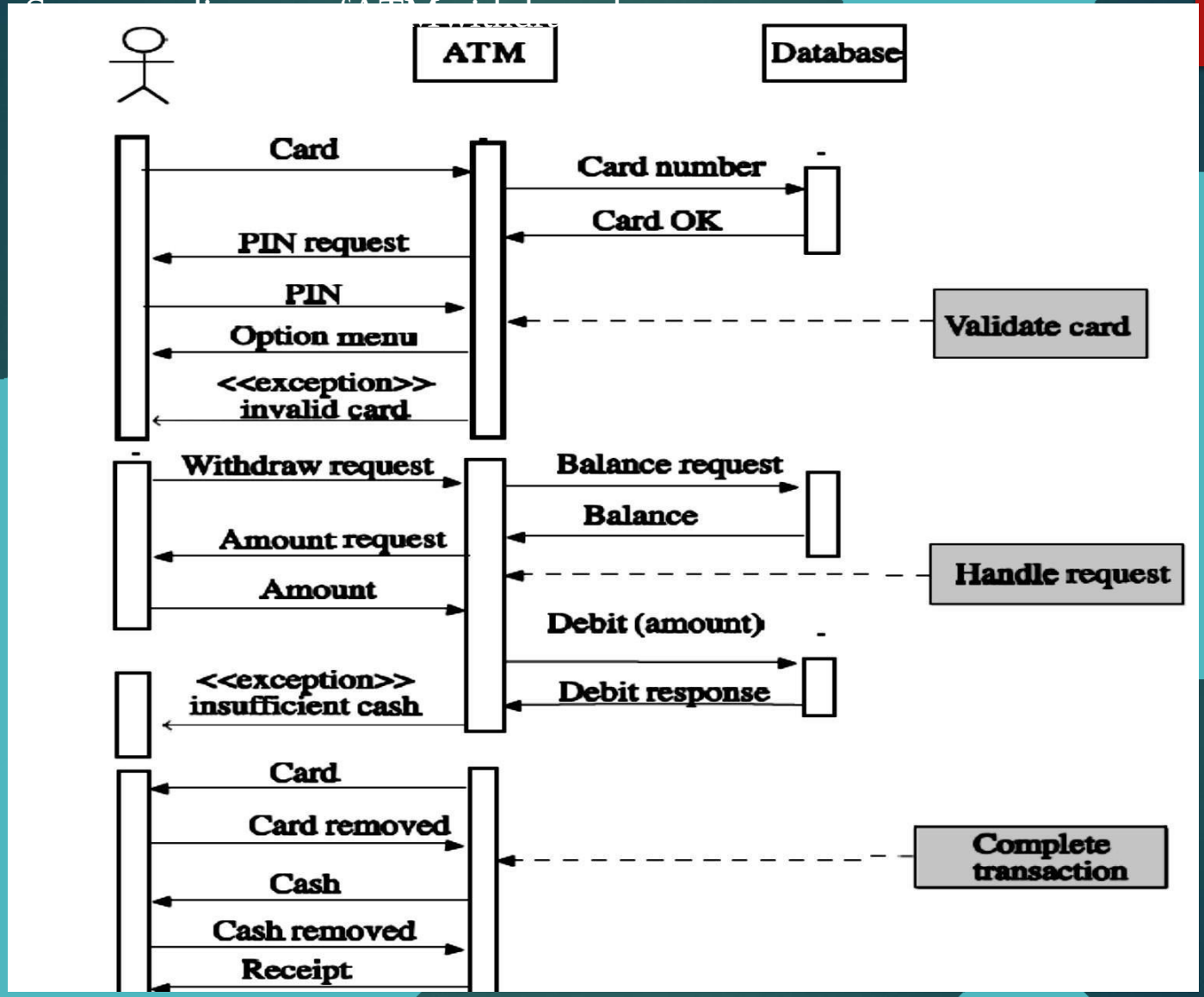
- Ambiguity
  - Thereadersandwritersoftherequirementmustinterpretthesamewords inthesameway.NLisnaturallyambiguoussothisisverydifficult.
- Over-flexibility
  - Thesamethingmaybesaidinanumberofdifferentwaysinthespecification.
- Lackofmodularization.
  - NLstructuresareinadequatetostructuresystemrequirements.

Alternativesto NLspecification:

- **Structuredlanguagespecifications**
  - All requirementsarewritteninastandardway.
  - Theterminologyusedinthedescriptionmaybelimited.
  - Theadvantageisthatthemostoftheexpressivenessofnaturallanguageis maintainedbutadegreeofuniformityisimposedonthe specification.

- **Form-basedspecifications**
  - Definitionofthefunctionorentity.
  - Descriptionofinputsand where theycome from.
  - Descriptionofoutputsandwhere theygoto.
  - Indicationofotherentitiesrequired.
  - Preandpostconditions(ifappropriate).
  - Thesideeffects(ifany)ofthefunction.

- **Tabularspecification**
  - Usedtosupplementnaturallanguage.
  - Particularlyusefulwhenyouhavetodefineanumberofpossiblealternative courses ofaction.
- **Graphicalmodels**
  - Graphicalmodels are mostusefulwhenyouneedtoshowhowstatechanges or whereyou need todescribe a sequence ofactions.
- **Sequencediagrams**
  - Theseshowthe sequenceofeventsthat takeplaceduringsomeuserinteraction withasystem.
  - Youreadthemfromtoptobottomtoseetheorderoftheactionsthattake place.
  - CashwithdrawalfromanATM
    - Validatecard;
    - Handlerequest;
    - Completetransaction.

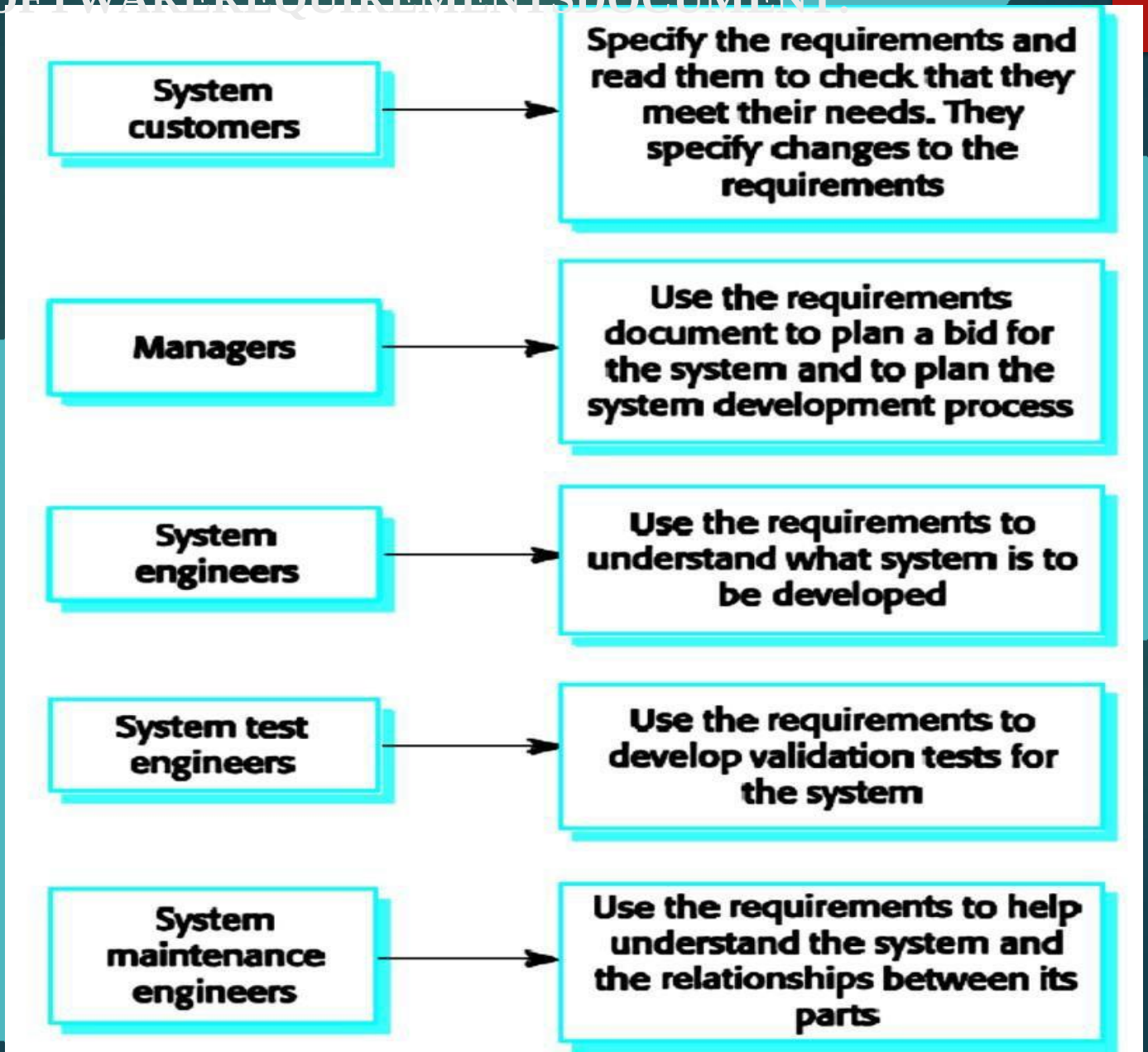

ATM
Database
Card
Card number
Card OK
PIN request
PIN
Option menu
<<exception>>
invalid card
Validate card
Withdraw request
Balance request
Balance
Amount request
Amount
Handle request
Debit (amount)
<<exception>>
insufficient cash
Debit response
Card
Card removed
Cash
Cash removed
Receipt
Complete
transaction

INTERFACESPECIFICATION

- Most systems must operate with other systems and the operatinginterfacesmust bespecifiedas part ofthe requirements.

- Threetypesofinterfacemayhavetobedefined
  - ***Proceduralinterfaces***whereexistingprogramsorsub-systemsofferarangeofservicesthatareaccessedbycallinginterfaceprocedures.TheseinterfacesaresometimescalledApplicationProgrammingInterfaces(APIs)
  - ***Datastructuresthatareexchanged***thatarepassedfromonesub-systemtoanother.Graphicaldatamodelsarethebestnotationsforthistypeofdescription
  - ***Datarepresentations***thathavebeenestablishedforanexistingsub-system
- Formalnotationsareaneffectivetechniqueforinterface specification.

THESOFTWAREREQUIREMENTSDOCUMENT:



- **IEEErequirementsstandard**definesagenericstructureforarequirementsdocumentthatmustbe instantiatedforeach specificsystem.

1. Introduction.

i)Purpose ofthe requirementsdocument ii)Scopeofthe project

iii)Definitions, acronymsandabbreviations iv)References

v)Overviewofthe remainderofthe document

2. Generaldescription.

i)Productperspective ii)Productfunctions

iii)Usercharacteristics iv)Generalconstraints

v)Assumptionsand dependencies

3. Specificrequirementscoverfunctional,non-functionalandinterfacerequirements.Therequirementsmaydocumentexternalinterfaces,describesystemfunctionalityand performance,specifylogicaldatabaserequirements,designconstraints, emergentsystemproperties andqualitycharacteristics.

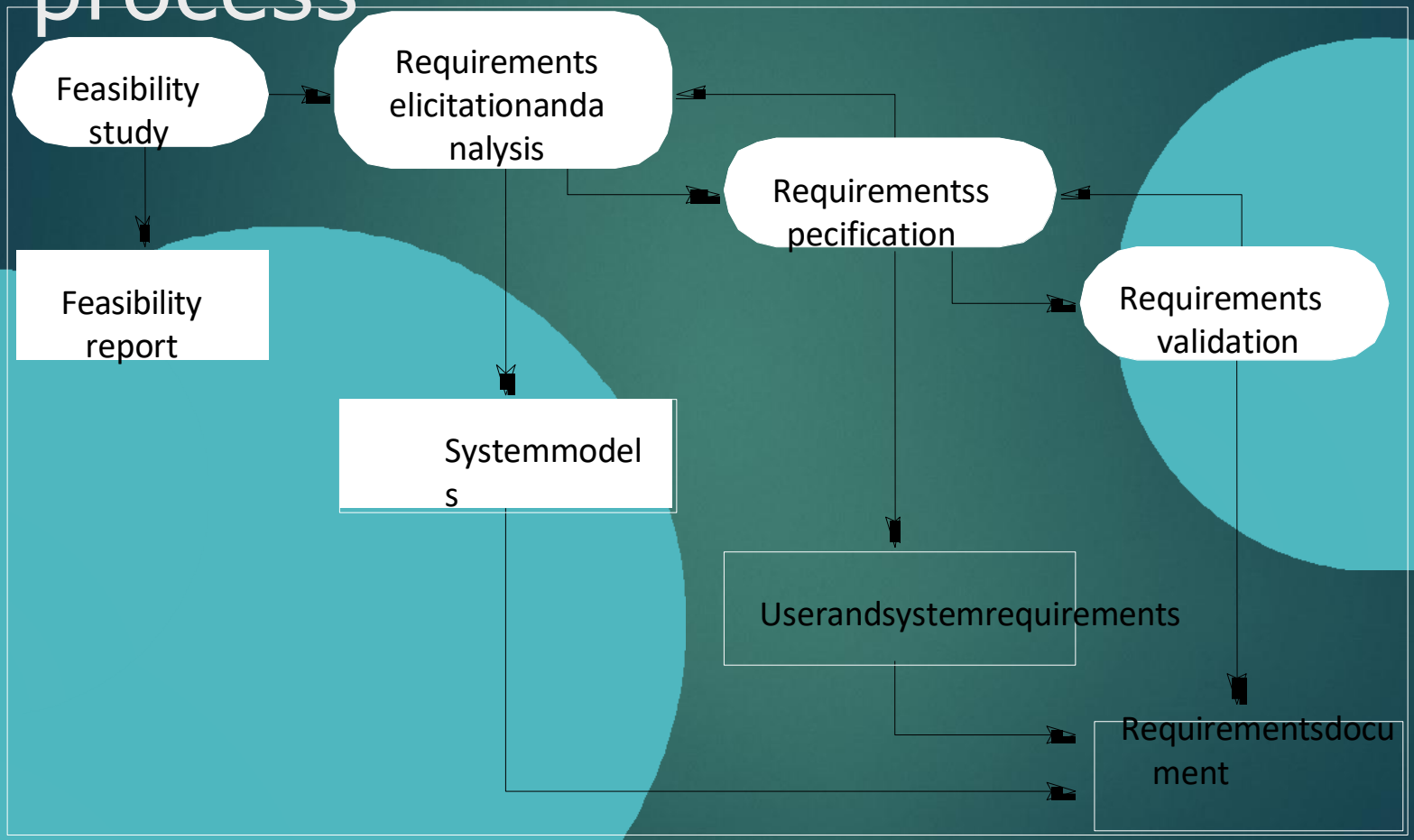
4. Appendices.

NRCM

REQUIREMENTSENGINEERINGPROCESS

- Tocreateandmaintainasystemrequirementdocument
- Theoverallprocessincludesfourhighlevelrequirementsen gineeringsub-processes:
- 1.Feasibilitystudy
  --
  Concernedwithassessingwhetherthesystemisusefultothebusiness
- 2.Elicitationandanalysis
  --Discoveringrequirements
- 3.Specifications
  --Convertingtherequirementsintoastandardform
- 4.Validation
  --Checkingthattherequirementsactuallydefinethe system thatthecustomerwants

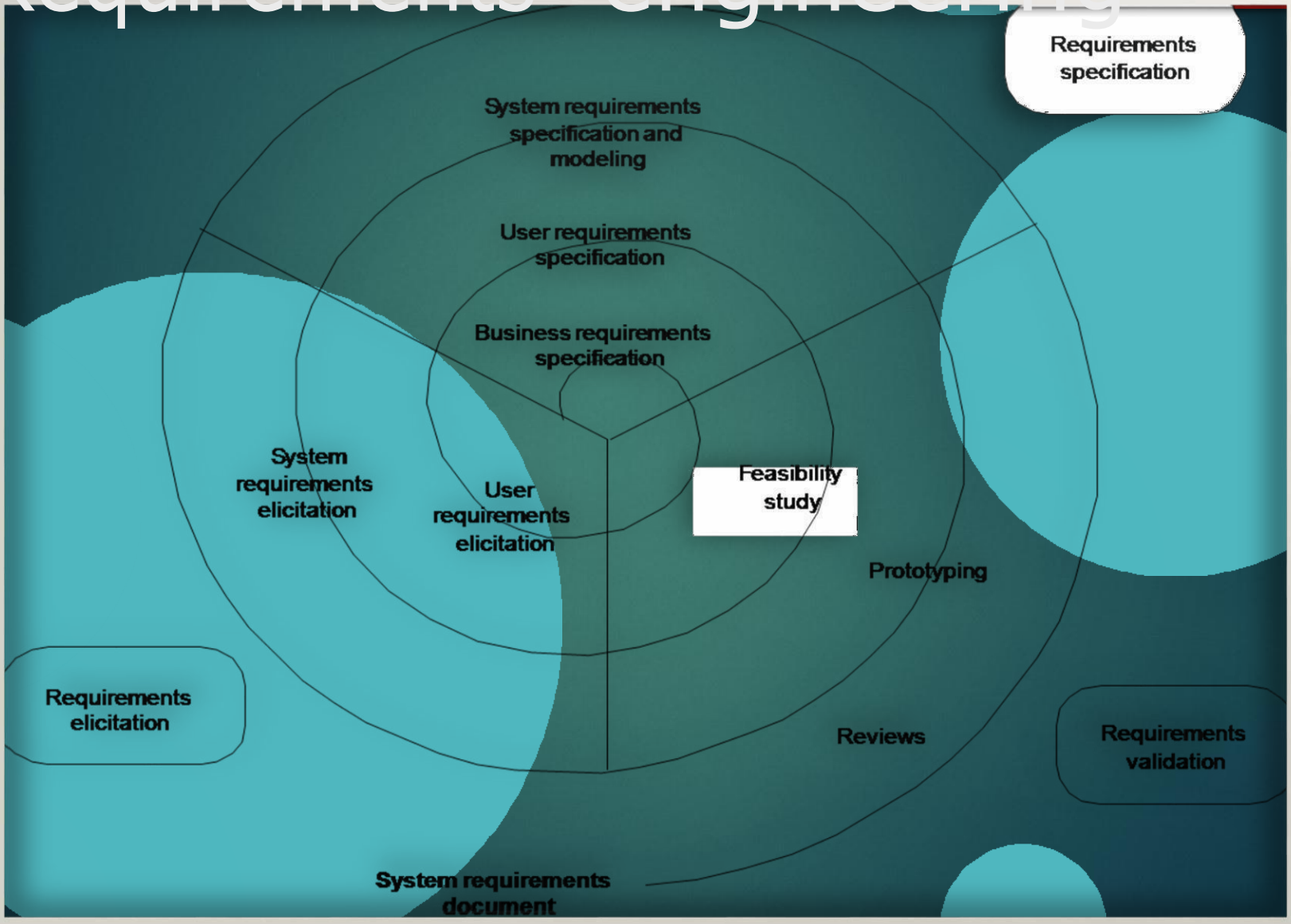
Therequirementsengineering process



SPIRALREPRESENTATIONOFREQUIREMENTSENGINEERINGPROCESS

- Processrepresentedasthreestageactivity
- Activitiesareorganizedasaniterativeprocessaroundaspiral.
- Earlyintheprocess,mosteffortwillbespentonunderstandinghigh-levelbusinessandtheuserrequirement.
- Laterintheouterrings,moreeffortwillbedevotedtosystemrequirementsengineeringandsystemmodeling
- Threelevelprocessconsistsof:
  - 1.Requirementselicitation
  - 2.Requirementsspecification
  - 3.Requirementsvalidation

Requirements engineering



FEASIBILITY STUDIES

- Startingpointoftherequirementsengineeringprocess
- Input:Setofpreliminarybusinessrequirements,anoutinedescriptionofthesystemandhowthesystemisintendedtosupportbusinessprocesses
- Output:Feasibilityreportthatrecommendswhetherornotitisworthcarryingoutfurther
- Feasibilityreportanswersanumberofquestions:
- 1.Doesthesystemcontributetotheoverallobjective
- 2.Canthesystembeimplementedusingthecurrenttechnologyandwithingivencostandschedule
- 2.Canthesystembeintegratedwithothersystemwhichare alreadyinplace.

REQUIREMENTS ELICITATIONANALYSIS

- Involvesanumberofpeopleinanorganization
- Stakeholderdefinition

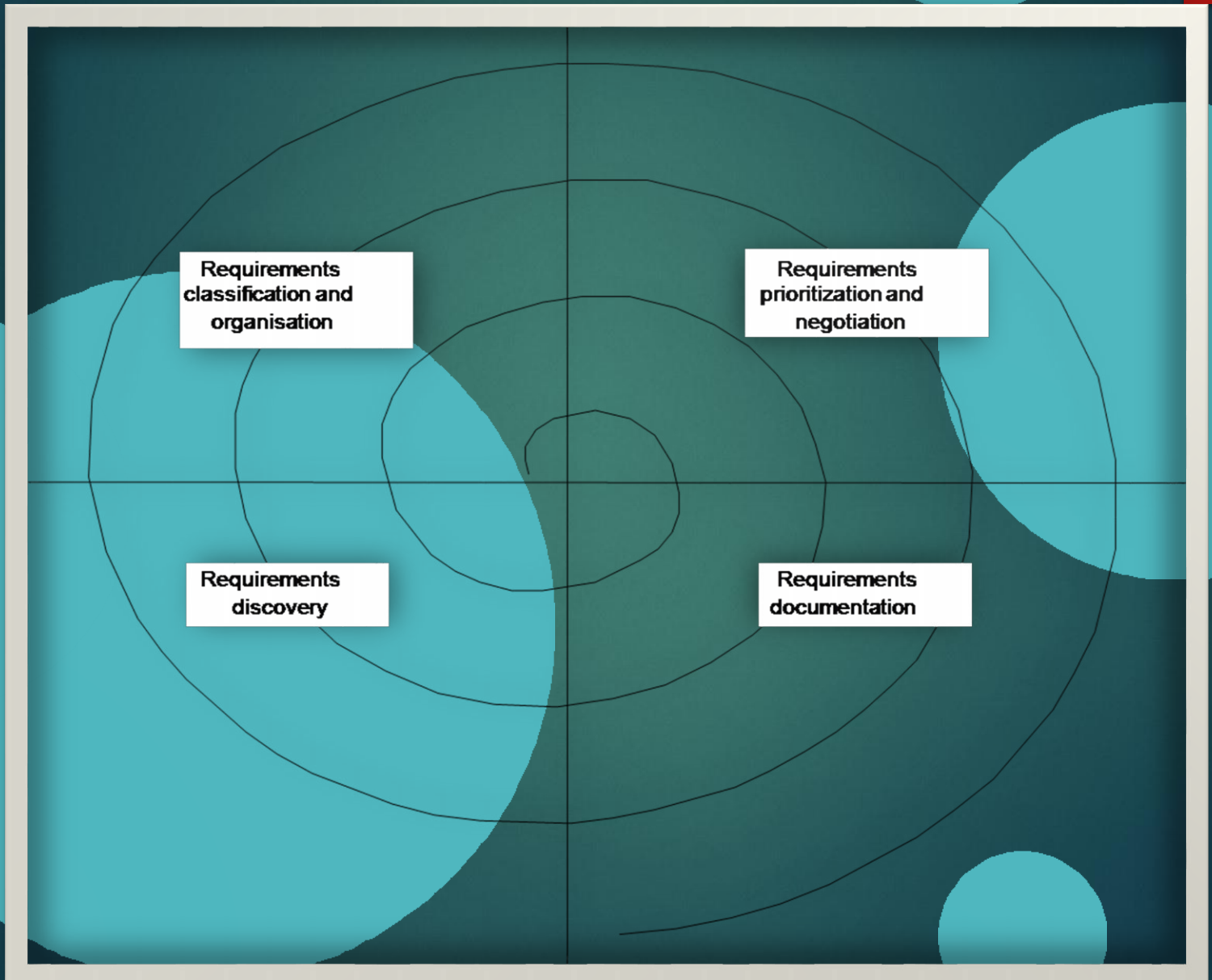
  Referstoanypersonorgroup whowillbeaffectedbythesystemdirectlyor indirectlyi.e. Endusers,Engineers, businessmanagers,domainexperts.
- Reasonswhyelicitingisdifficult
  - 1.Stakeholderoftendon’tknowwhattheywantfromthecomputersystem.
  - 2.Stakeholderexpressionofrequirementsinnaturallanguageissometimesdifficulttounderstand.
  - 3.Differentstakeholdersexpressrequirementsdifferently
  - 4.Influencesofpoliticalfactors
- Changeinrequirementsduetodynamicenvironments.

REQUIREMENTSELICITATIONPROCESS

Processactivities

- 1.RequirementDiscovery
  - Interactionwithstakeholdertocollecttheirrequirementsincluding domainanddocumentation
- 2.Requirementsclassificationandorganization
  - Coherentclusteringofrequirementsfromunstructuredcollectionofrequirements
- 3.Requirementsprioritizationandnegotiation
  - Assigningpriorityrequirements
  - Resolvesconflictingrequirementsthroughnegotiation
- 4.Requirementsdocumentation
  - Requirementsbedocumentedandplacedinthenextroundofspiral

ThespiralrepresentationofRequirementsEngineering



REQUIEMENTSDISCOVERYTECHNIQUES

1. Viewpoints

--Basedontheviewpointsexpressedbythestakeholder

--
Recognizesmultipleperspectivesandprovidesaframeworkfordiscoveringconflictsintherequirementsproposedbydifferentstakeholders

ThreeGenerictypesofviewpoints1.Interactorviewpoint

--Representspeopleorothersystemthatinteractdirectlywiththesystem

2.Indirectviewpoint

--Stakeholderswhoinfluencetherequirements,butdon’tusethesystem

3.Domainviewpoint

--Requirementsdomaincharacteristicsand constraintsthatinfluencetherequirements.

REQUIREMENTSDISCOVERYTECHNIQUES

2. Interviewing

--Putsquestionstostakeholdersaboutthesystem thattheyuseandthesystemtobedeveloped.

--Requirementsarederivedfromtheanswers

- Twotypesofinterview
  - Closedinterviewswherethestakeholdersanswerapre-definedsetofquestions.
  - Openinterviewsdiscussarangeofissueswiththestakeholdersforbetterunderstandingtheirneeds.
- Effectiveinterviewers

  a) Open-minded:nopre-conceivedideas

  b) Prompter:prompttheintervieweetostartdiscussionwithaquestionoraproposal

REQUIREMENTSDISCOVERYTECHNIQUES

3. Scenarios

--Easiertorelatetoreallifeexamplesthantoabstractdescription

--

Startswithanoutlineoftheinteractionandduringelicitation,detailsareaddedtocreateacompletedescriptionofthatinteraction

--Scenarioincludes:

1. Descriptionatthestartofthescenario
2. Descriptionofnormalflowoftheevent
3. Descriptionofwhatcangowrongandhowthisishandled4.Information aboutotheractivitiesparalleltothescenario5.Descriptionofthesystemstatewhenthescenariofinishes

LIBSYS scenario

- **Initialassumption**:TheuserhasloggedontotheLIBSYSsystemandhaslocatedthejournalcontainingthecopyofthearticle.

- **Normal**:Theuserselectsthearticletobecopied.Heorsheisthenpromptedbythesystemtoeitherprovidesubscriberinformationforthejournalortoindicatehowtheywillpayforthearticle.Alternativepaymentmethodsarebycreditcardorbyquotinganorganisationalaccountnumber.

- TheuseristhenaskedtofillinacopyrightformthatmaintainsdetailsofthetransactionandtheythensubmitthistotheLIBSYSsystem.

- Thecopyrightformischeckedand,ifOK,thePDFversionofthearticleisdownloadedtotheLIBSYSworkingareaontheuser'scomputerandtheuserisinformedthatitisavailable.Theuserisaskedtoselectaprinterandacopyofthearticleisprinted

LIBSYS scenario

- **Whatcangowrong**:Theusermayfailtofillinthecopyrightformcorrectly.Inthiscase,theformshouldbere-presentedtotheuserfor correction.Iftheresubmittedformisstillincorrectthentheuser’srequestforthearticleisrejected.
- Thepaymentmayberejectedbythesystem.Theuser’srequestforthearticleisrejected.
- Thearticledownloadmayfail.Retryuntilsuccessfulortheuserterminatesthesession..
- **Otheractivities**:Simultaneousdownloadsofotherarticles.
- **Systemstateoncompletion**:Userisloggedon.Thedownloadedarticlehasbeendeletedfrom LIBSYSworkspaceifithasbeenflaggedasprint-only.

REQUIREMENTSDISCOVERYTECHNIQUES

4. Usecases

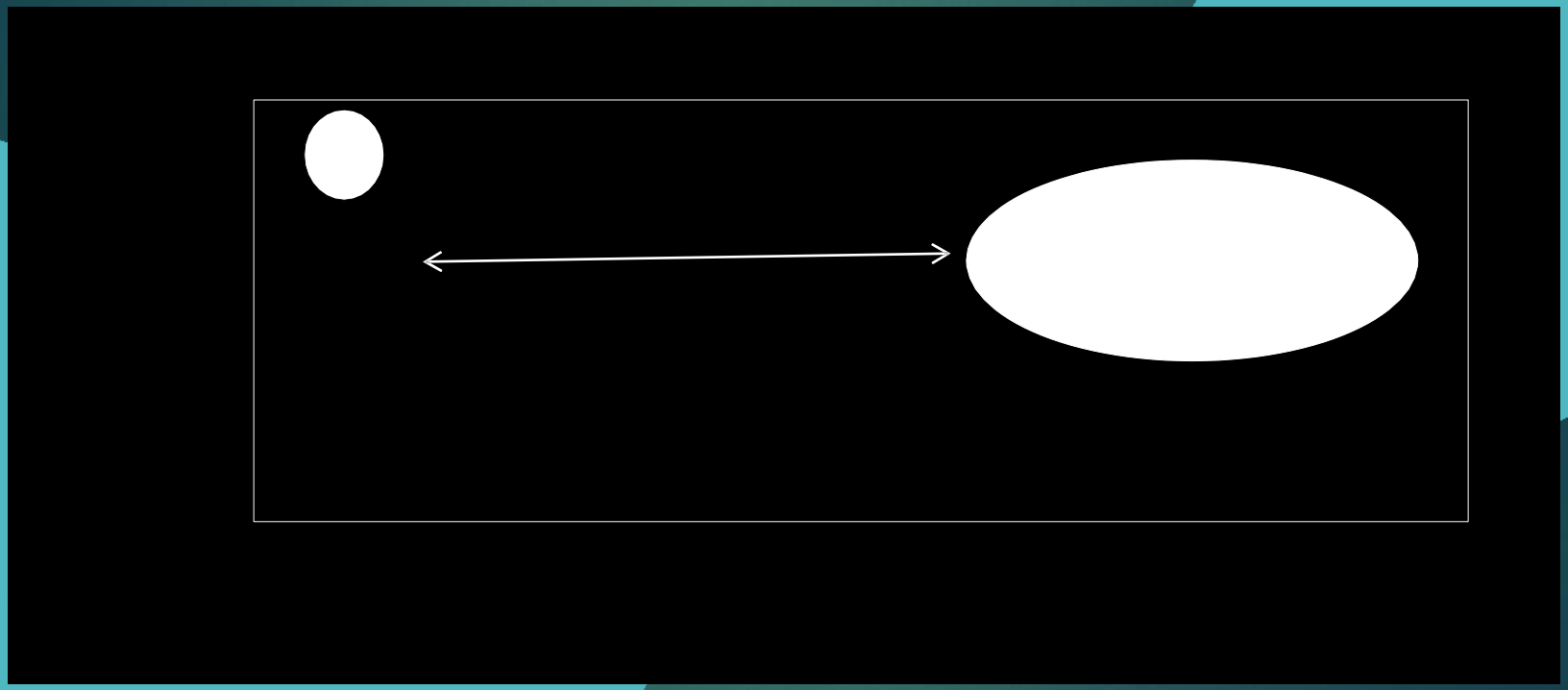
-- scenariobased techniqueforrequirementelicitation

--Afundamentalfeatureof UML, notationfordescribingobject-orientedsystemmodels

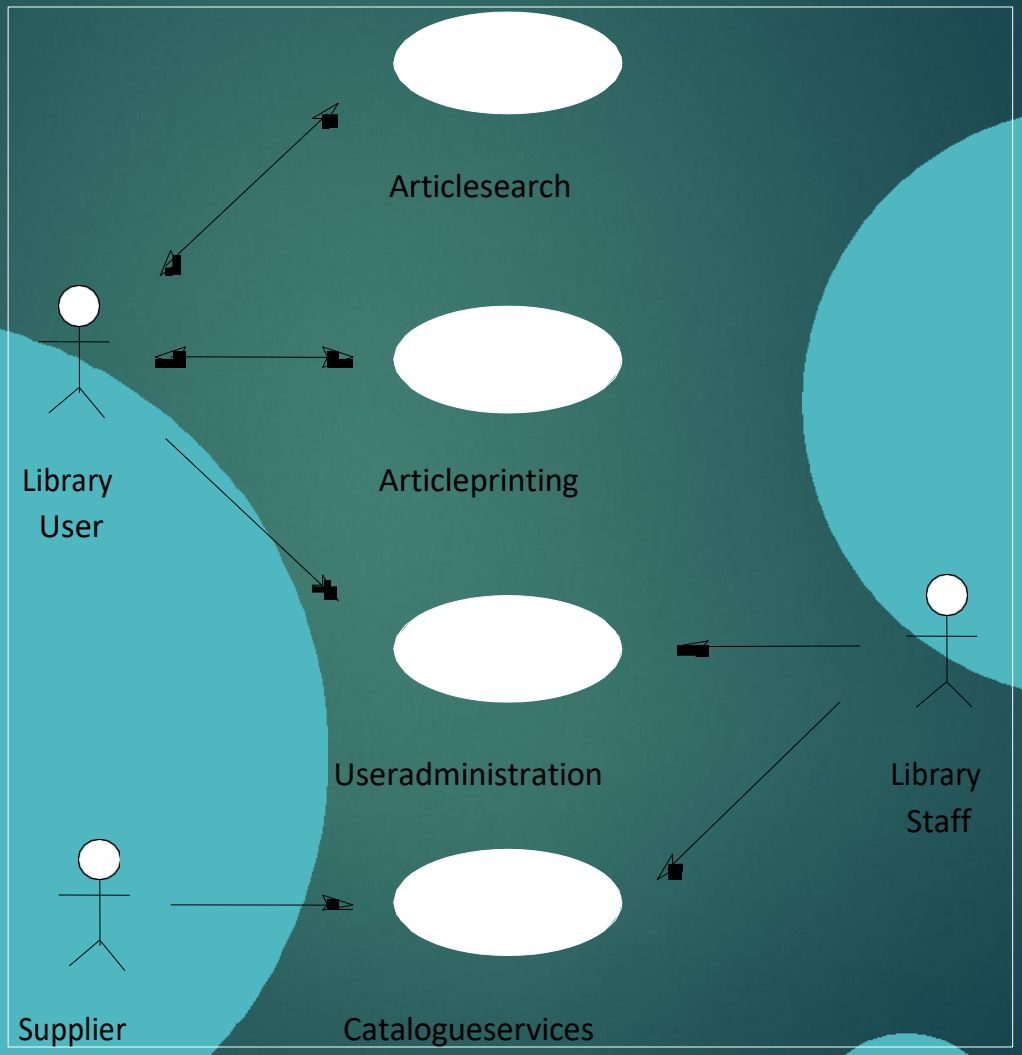
--Identifiesatypeof interactionand the actorsinvolved

--Sequence diagramsareused toaddinformationtoaUsecase

Articleprintinguse-case



LIBSYS use cases



REQUIREMENTSVALIDATION

- Concernedwithshowingthattherequirementsdefine the systemthatthe customerwants.
- Importantbecauseerrorsinrequirements canleadtoextensivereworkcost
- Validation
  checks1.Validity
  checks

  --Verificationthatthesystemperformstheintendedfunctionbytheuser

  2.Consistencycheck

  --Requirementsshould notconflict

  3. Completenesschecks

  --Includesrequirementswhichdefineallfunctionsand constraints intended bythesystemuser

  4. Realismchecks

  --Ensuresthatthe requirementscanbe actuallyimplemented

  5. Verifiability

  --Testabletoavoiddisputesbetween customerand developer.

VALIDATIONTECHNIQUES

1.REQUIREMENTSREVIEWS

--Reviewerscheckthefollowing:

(a) Verifiability:Testable

(b) Comprehensibility

(c) Traceability

(d) Adaptability

2.PROTOTYPING

3.TEST-CASEGENERATION

Requirementsmanagement

Requirementsarelikelytochangeforlargesoftwaresystemsand assuchrequirementsmanagement processisrequiredtohandlechanges.

Reasonsforrequirementschanges

(a) DiverseUserscommunitywhereusershavedifferentreq uirementsandpriorities

(b) Systemcustomersandendusersaredifferent

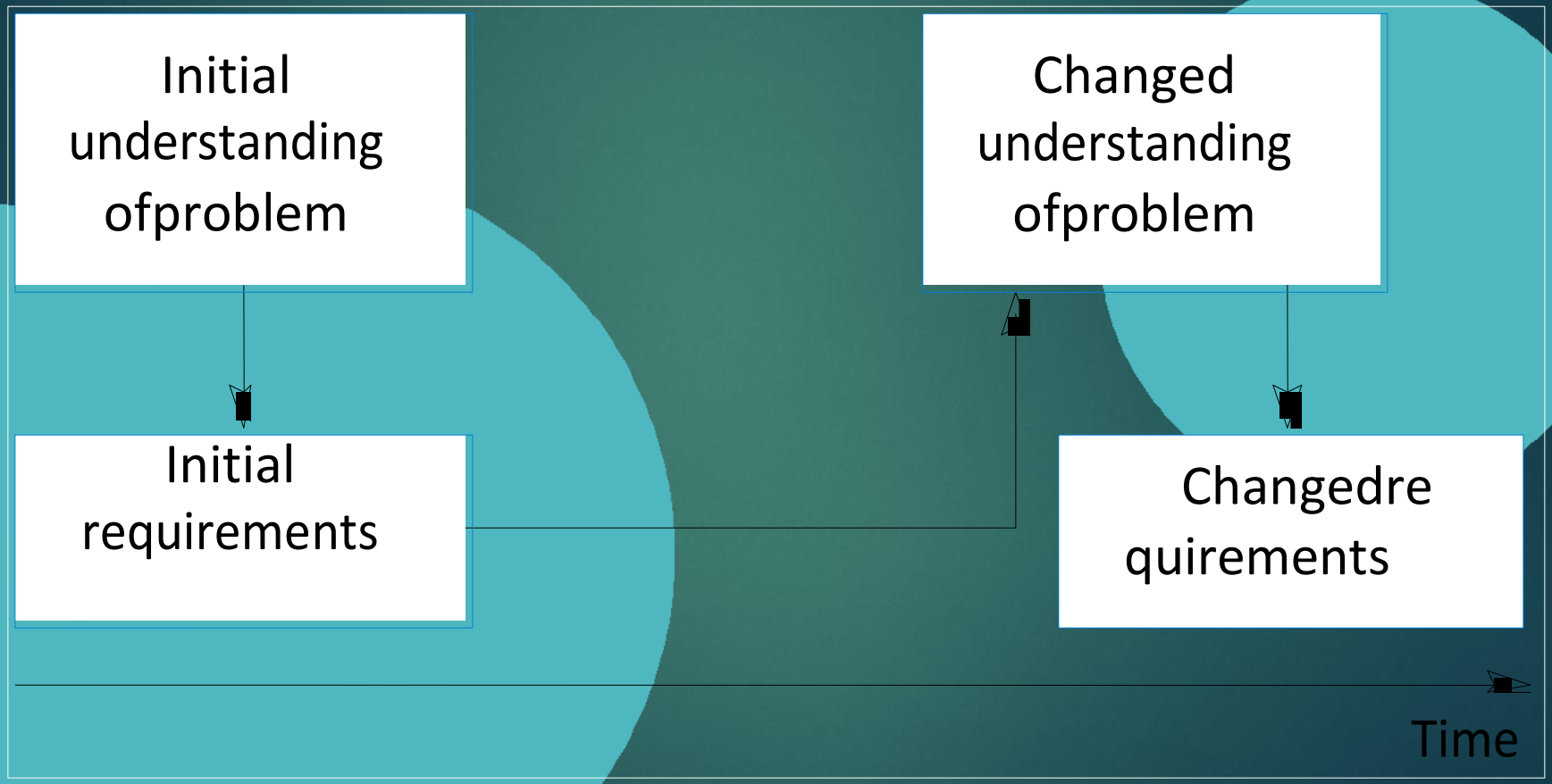
(c) Changeinthebusinessandtechnicalenvironmentafterinst allation

Twoclassesofrequirements

(a) Enduringrequirements:Relativelystablerequirements

(b) Volatilerequirements:Likelytochangeduringsystemde velopmentprocessorduringoperation

Requirementsevolution



Requirements management planning

Anessentialfirststageinrequirementmanagementprocess Planningprocessconsistsofthefollowing

1. Requirementsidentification

-- Eachrequirementmusthaveuniquetagforcrossreferenceandtraceability

2. Changemanagementprocess

-- Setofactivitiesthatassesstheimpactandcostofchanges 3. Traceabilitypolicy

-- Amatrixshowinglinksbetweenrequirementsandotherelementsofsoftwaredevelopment

4. CASEtoolsupport

-- Automatictooltoimproveefficiencyofchangemanagementprocess. Automatedtoolsarerequiredforrequirementsstorage,changemanagementandtr

NRCM

Traceability

Maintainsthreetypesoftraceabilityinformation.

1.Source traceability

--Linkstherequirementsto the stakeholders

2. Requirementstraceability

--

Linksdependentrequirementswithintherequirements document

3. Designtraceability

-- Linksfromtherequirementstothe designmodule

Atraceabilitymatrix

Req. id	1.1	1.2	1.3	2.1	2.2	2.3	3.1	3.2
1.1		D	R					
1.2			D			D		D
1.3	R			R				
2.1			R		D			D
2.2								D
2.3		R		D				
3.1								R
3.2							R	

Requirementschange management

Consistsofthreeprincipal stages:

1. Problemanalysisandchangespecification

   --Processstartswith aspecificchange proposalandanalysed toverifythatit isvalid

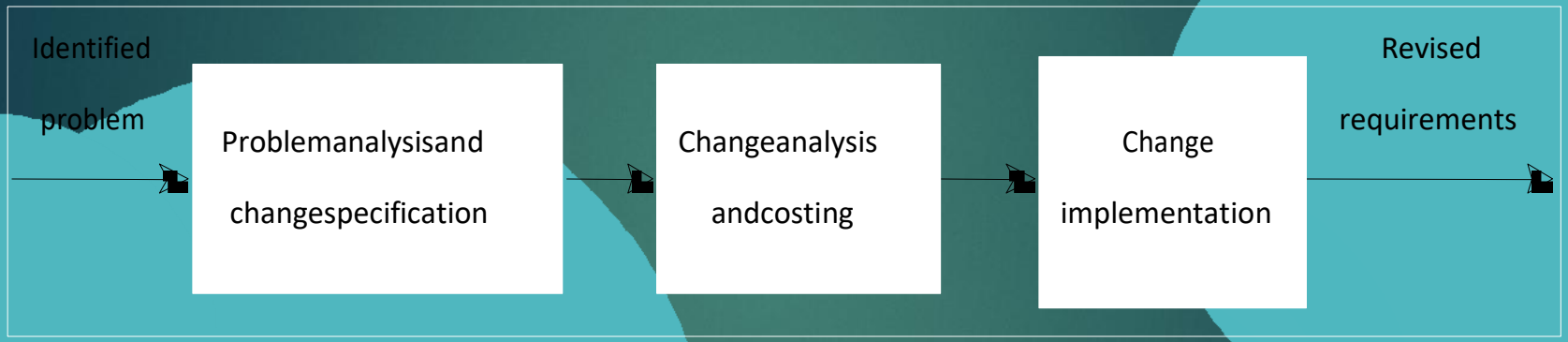
2. Changeanalysisandcosting

   --Impactanalysisinterms ofcost,time andrisks

3. Changeimplementation

   --Carryingoutthechangesin requirementsdocument,systemdesignanditsimplementation

Change management



SYSTEMMODELS

- Used in analysisprocess to develop understandingof theexistingsystemor newsystem.
- Excludedetails
- An abstraction ofthe

systemTypesofsystemmodels

1.Contextmodels

2. Behaviouralmodels

3.Datamodels4.Object models5.Structuredmodels

CONTEXT MODELS

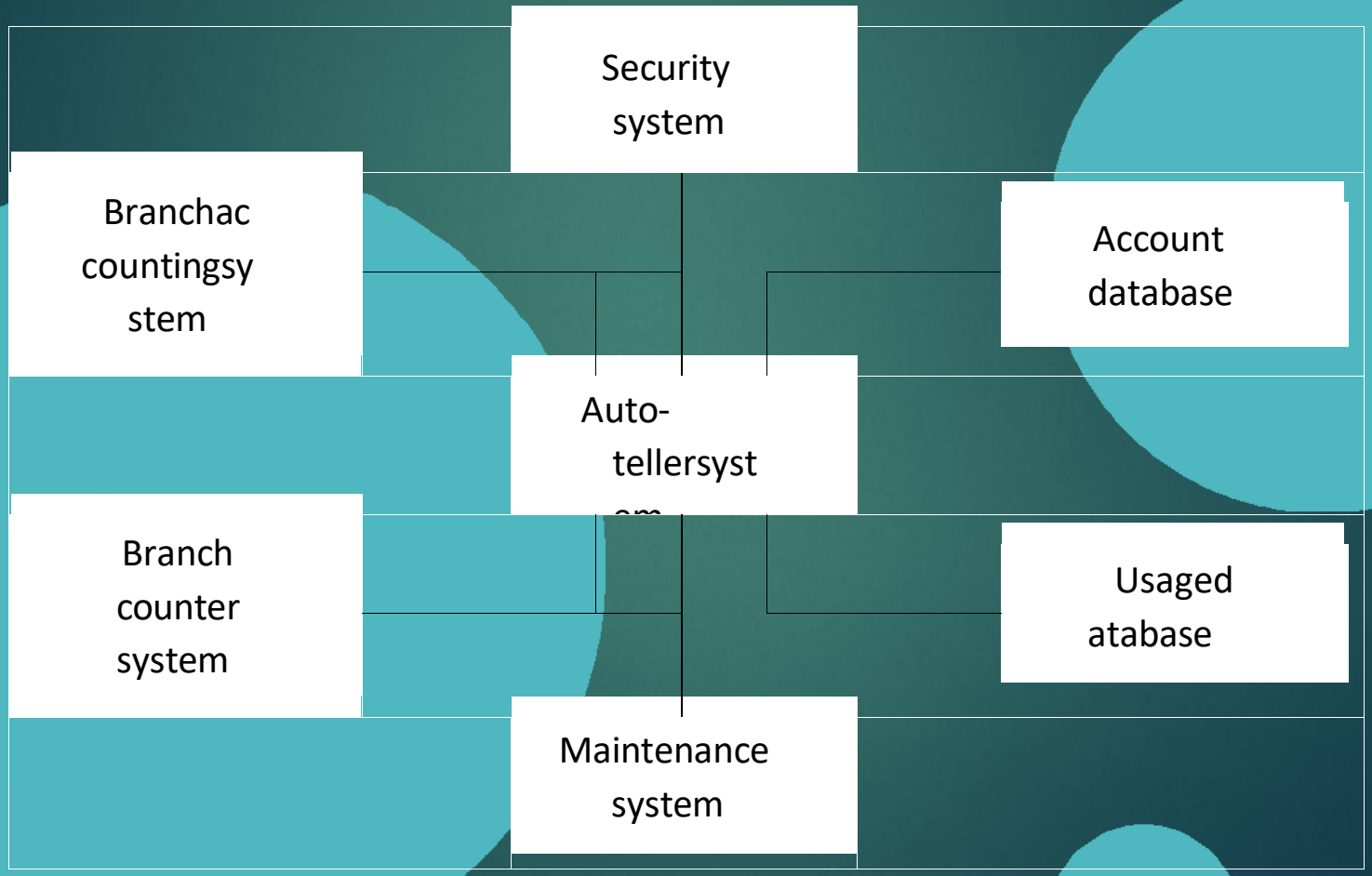
Atypeofarchitecturalmodel

- Consistsofsub-systemsthatmake upanentiresystem
- Firststep:Toidentifythesubsystem
- Representthehighlevelarchitecturalmodelassimpleblock diagram
- Depicteachsubsystemanamedrectangle
- Linesbetweenrectanglesindicateassociationsbetweensub systems

Disadvantages

- Concernedwithsystemenvironmentonly,doesn'ttakeintoa ccountothersystems,whichmaytakedataorgivedatatothe model

ThecontextofanATM system



Behavioral models

Describethe overallbehaviourof asystem.

Twotypesofbehaviouralmodel

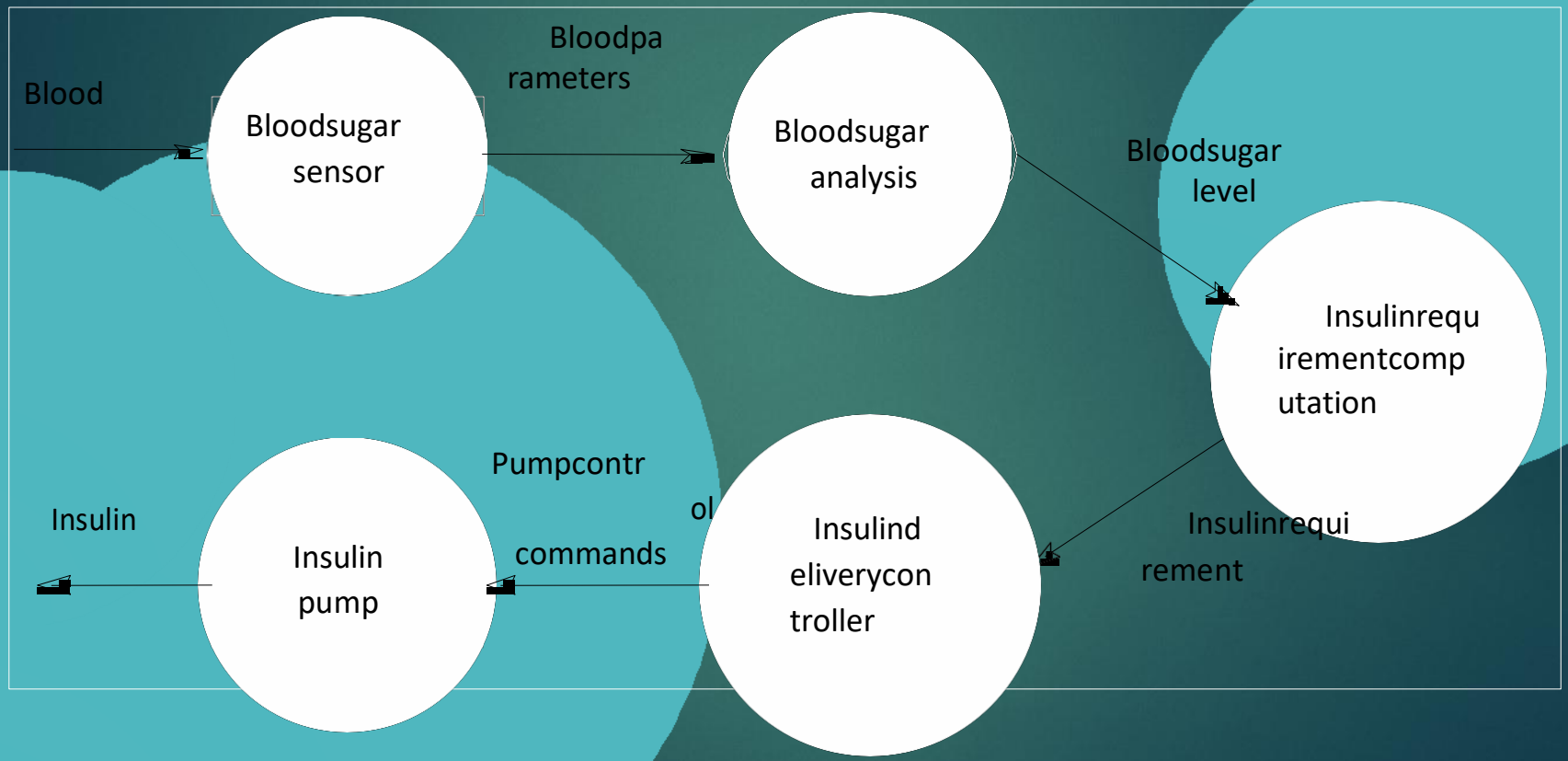
1. DataFlowmodels
2. Statemachinemodels

- Dataflowmodels
  - Concentrate ontheflowofdataandfunctionaltransformationonthatdata
  - Showtheprocessingofdataanditsflowthroughasequenceofprocessing steps
  - Helpanalystunderstandwhat isgoingon

Advantages

- Simpleandeasilyunderstandable
- Useful duringanalysisofrequirements

InsulinpumpDFD



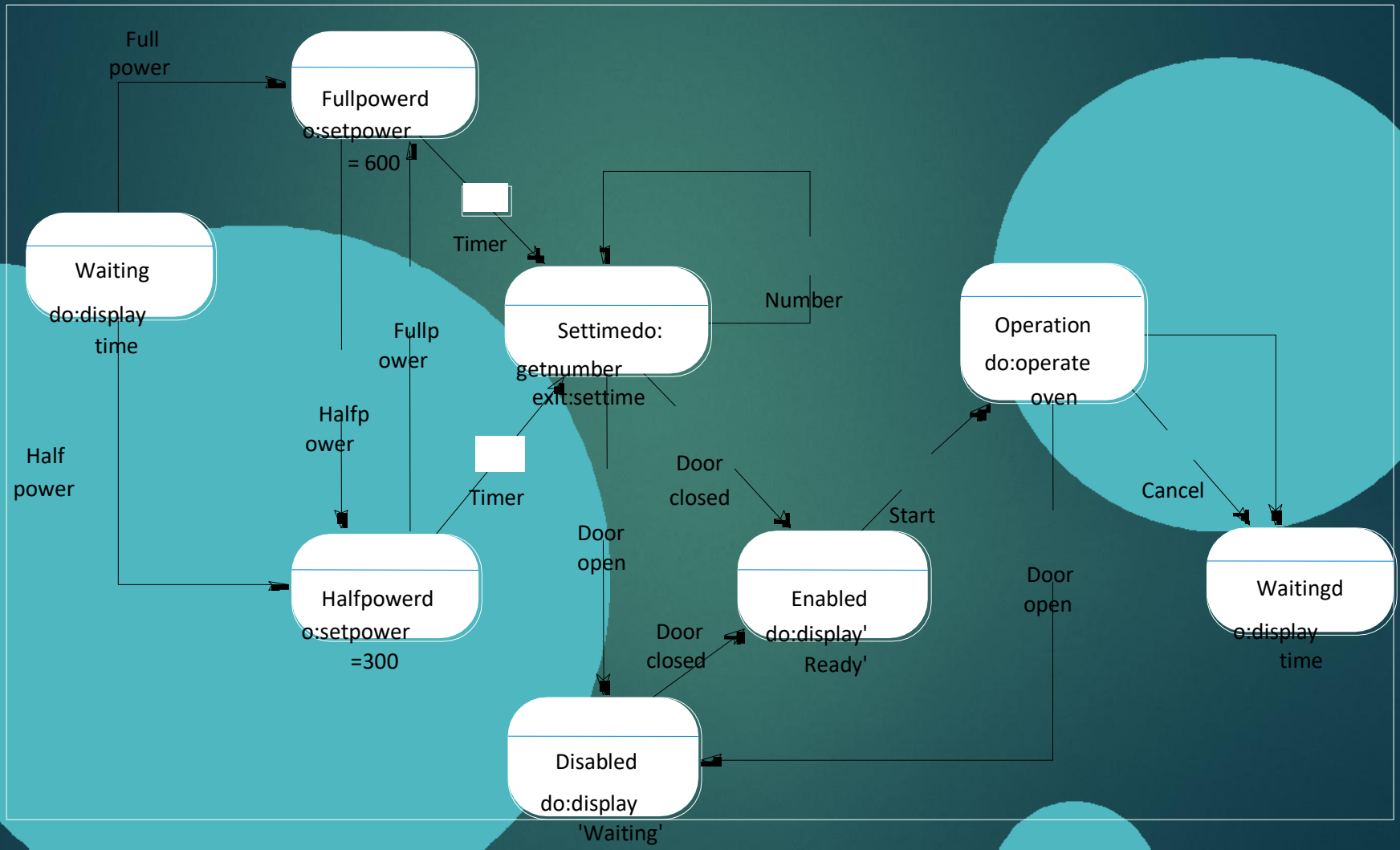
State machine models

- Describehowasystemrespondstointernal or external events
- Showssystemstatesandeventsthatcausetransitionfromonestateto another
- Doesnotshowtheflowof datawithinthesystem
- Usedfor modelingofreal timesystems
- Exp:Microwaveoven
- Assumesthatatanytime, thesystemis inone ofanumberofpossiblestates
- Stimulus triggers atransitionfromonstatetoanotherstate

Disadvantage

- Numberof possiblestatesincreasesrapidlyforlargesystemmodels

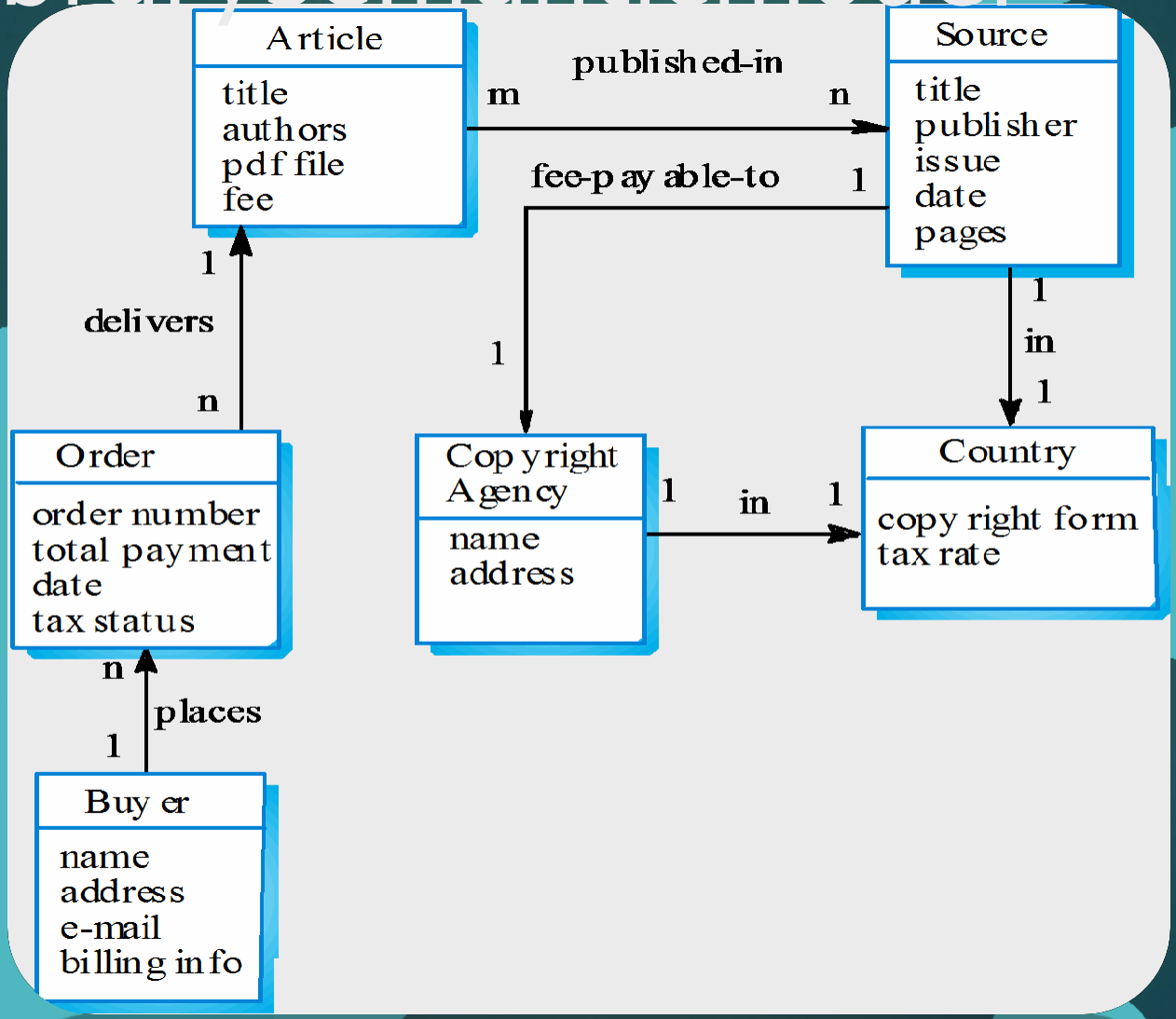
Microwave ovenmodel



DATAMODELS

- Usedtodescribethelogicalstructureofdataprocessed bythesystem.
- Anentity-relation-attributemodelsetsouttheentitiesinthesystem,there lationshipsbetweentheseentitiesandtheentityattrib utes
- Widelyusedindatabasedesign.Canreadilybeimplem entedusingrelationaldatabases.
- NospecificnotationprovidedintheUMLbutobjectsa ndassociationscanbeused.

Librarysemanticmodel



Datadictionaryentries

Name	Description	Type	Date
Article	Details of the published article that may be ordered by people using LIBSYS.	Entity	30.12.2002
authors	The names of the authors of the article who may be due a share of the fee.	Attribute	30.12.2002
Buyer	The person or organisation that orders a co py of the article.	Entity	30.12.2002
fee-payable-to	A 1:1 relationship between Article and the Copyright Agency who should be paid the copyright fee.	Relation	29.12.2002
Address (Buyer)	The address of the buyer. This is used to any paper billing information that is required.	Attribute	31.12.2002

OBJECTMODELS

- Anobjectorientedapproachis commonlyusedforinteractivesystemsdevelopment
- Expressesthe systems requirementsusing objectsand developingthe systeminanobjectorientedPLsuchasc++
- Aobjectclass:ANabstractionoveraset ofobjectsthatidentifiescommonattributes
- Objectsare instances of objectclass
- Manyobjectsmaybe createdfrom a singleclass
- Analysisprocess
  - Identifiesobjectsand objectclasses
- Objectclass inUML
  - Represented as averticallyoriented rectanglewiththreesections
    - (a)Thenameoftheobjectclassinthetopsection
    - (b)Theclassattributesin themiddlesection
    - (c)The operationsassociatedwiththeobjectclassare in lowersection.

OBJECTMODELS

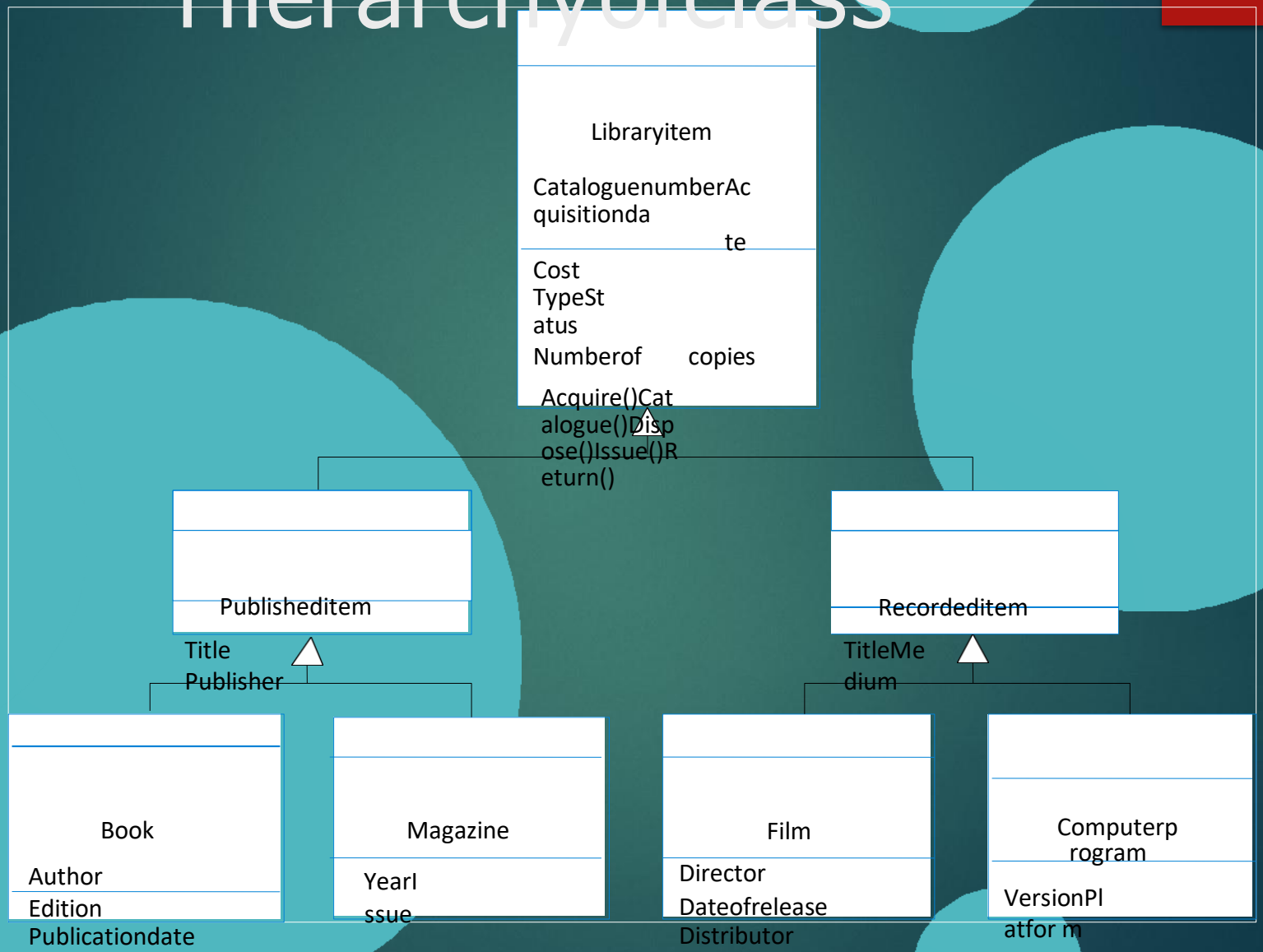
INHERITANCEMODELS

- Atypeofobjectorientedmodelwhichinvolvesinobjectclassesattributes
- Arrangesclassesintoaninheritancehierarchywiththemostgeneralobjectclassatthetopofhierarchy
- SpecializedobjectsinherittheirattributesandservicesUML Lnotation
- Inheritanceisshownupwardratherthandownward

SingleInheritance:Everyobjectclassinheritsitsattributesandoperationsfromasingleparentclass

MultipleInheritance:Aclassofseveralofseveralparents.

Hierarchyofclass



OBJECTMODELS

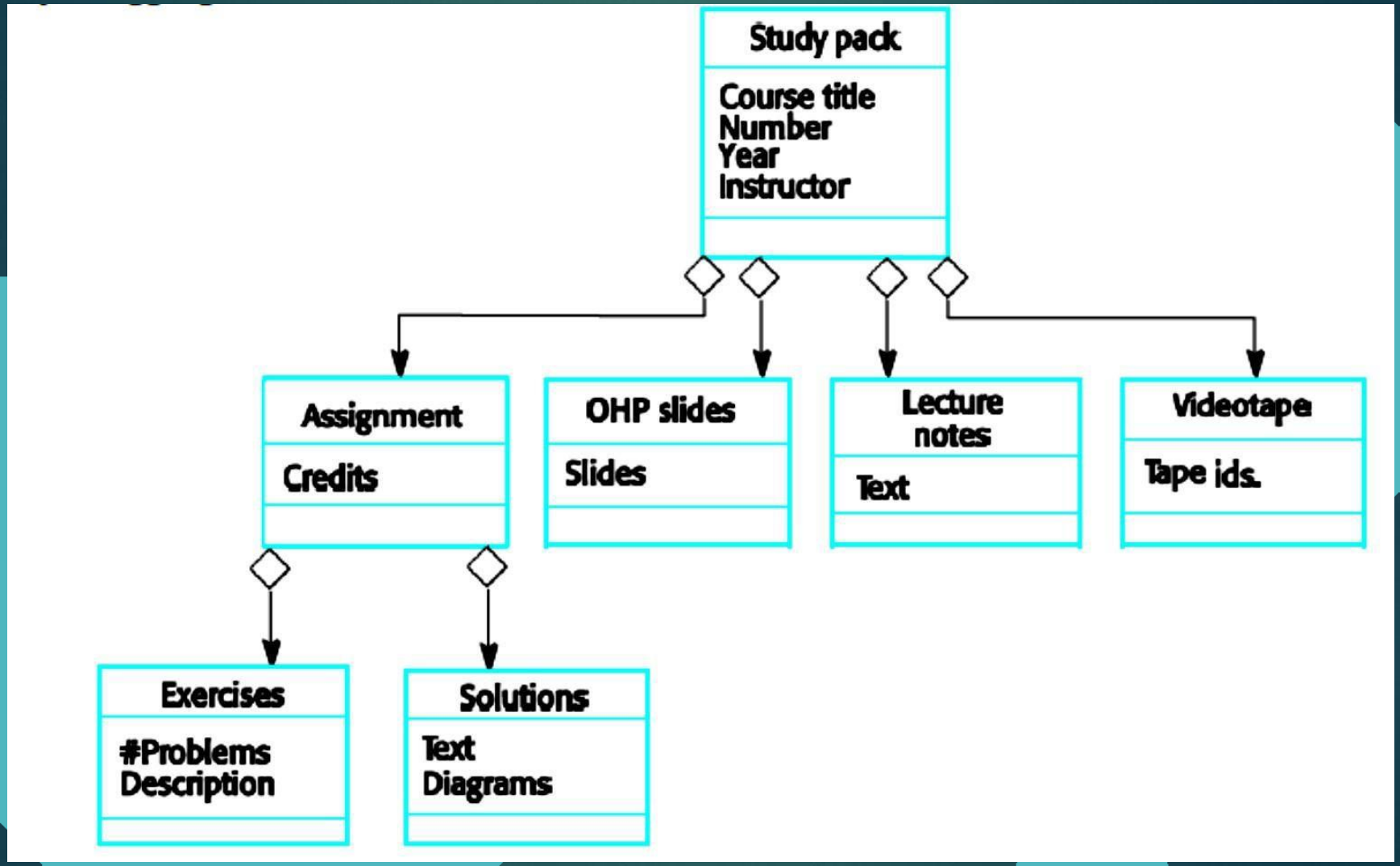
OBJECTAGGREGATION

- Someobjectsaregroupingofotherobjects
- Anaggregateofasetofotherobjects
- Theclasses representingtheseobjectsmaybemodeledusingan objectaggregationmodel
- Adiamondshapeon thesourceofthelinkrepresentsthecomposition

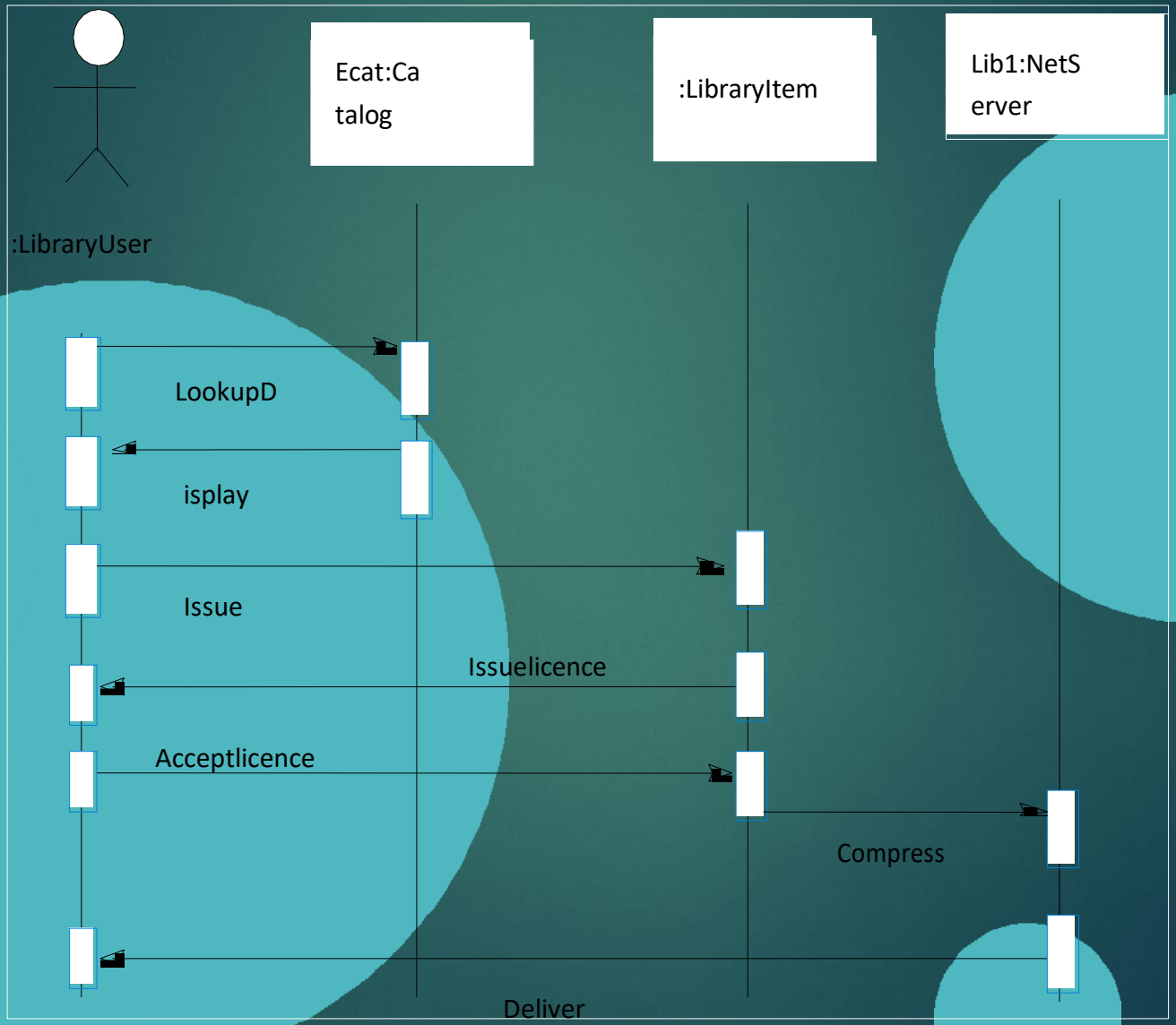
OBJECT-BEHAVIORALMODEL

- Showstheoperations providedbytheobjects
- Sequencediagram ofUMLcan beused forbehavioralmodeling

Object aggregation



OBJECTBEHAVIORALMODEL



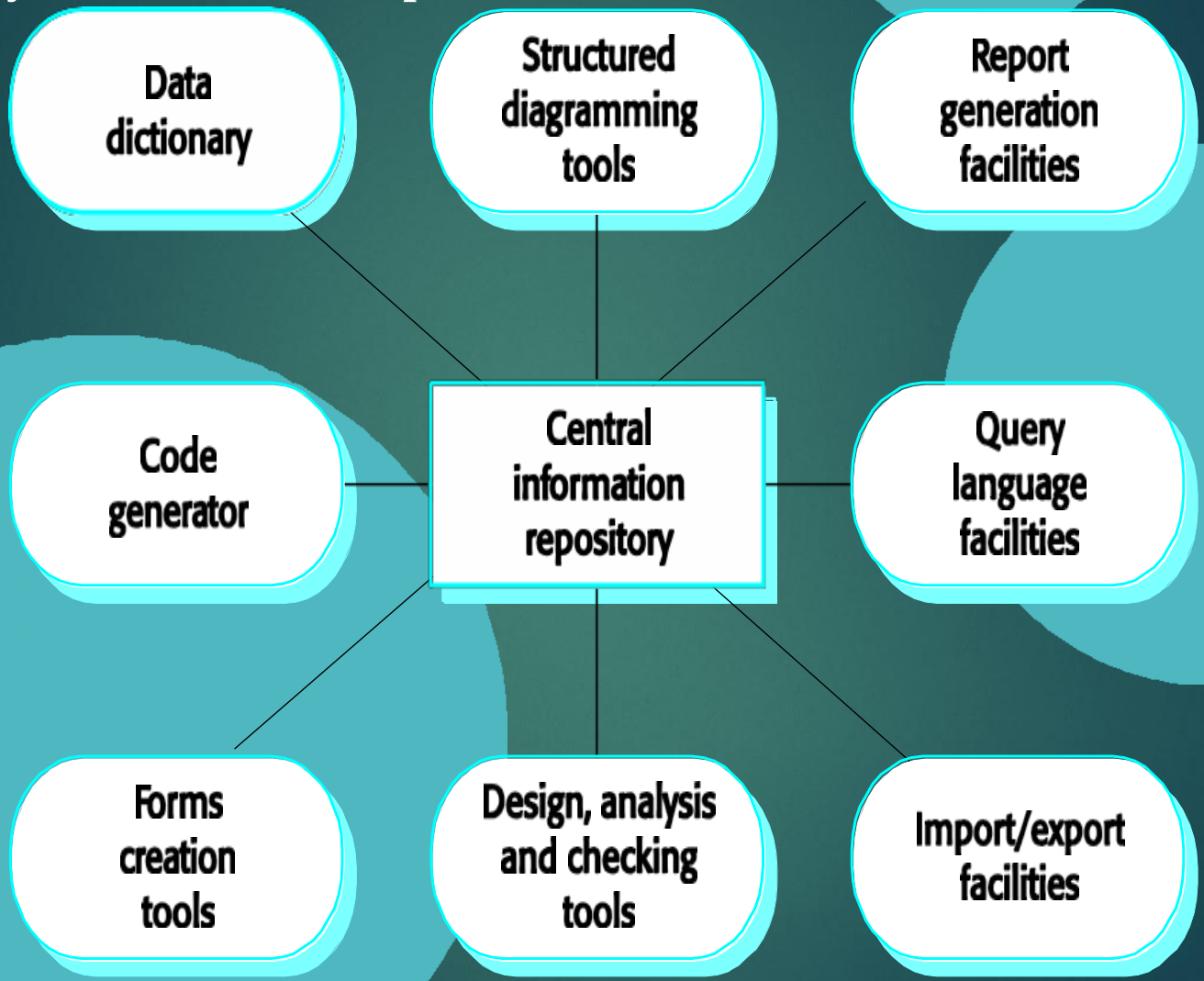
STRUCTURED METHODS:

- Structuredmethodsincorporatesystemmodellingasaninherentpart ofthemethod.
- Methodsdefine a setofmodels,aprocessforderivingthesemodelsandrules andguidelinesthatshouldapplytothemodels.
- CASEtoolssupportsystemmodellingaspartofastructuredmethod.

Methodweaknesses:

- Theydo notmodelnon-functionalsystemrequirements.
- Theydonotusuallyinclude informationaboutwhetheramethod isappropriateforagivenproblem.
- Themayproducetoo much documentation.
- The system models aresometimes toodetailedanddifficultforusers tounderstand.

Analysisworkbenchcomponents:



NARASIMHAREDDYENGINEERINGCOLLEGE

FACULTY:Dr.VenkateswaruluNaik

CS3102PC:SOFTWAREENGINEERING

III YEARB.TECH. CSE I-SEM (R21)

UNIT-III

- **Design Engineering:**DesignprocessandDesignquality,Designconcepts,thedesignmodel.

- **Creating an architecturaldesign:S**oftwarearchitecture, Datadesign,Architecturalstylesandpatterns,ArchitecturalDesign,conceptual modelofUML, basic structural modeling, classdiagrams,sequencediagrams,collaborationdiagrams,usecasediagrams,componentdiagrams.

DesignEngineering

- **Designengineering**encompassesthe**setofprinciples,concepts,and practices**thatleadstothedevelopmentofahigh-qualitysystemorproduct.

Whatisdesign?

- Designiswhatvirtuallyeveryengineerwantstodo.
- Itistheplacewherecreativityrules–customer'srequirements,businessneeds,andtechnicalconsiderationsallcometogetherintheformulationofaproductorasystem.

Whyisitimportant?

- Designallowsasoftwareengineertomodelthesystemorproductthatistobebuilt.
- Designistheplacewheresoftware**qualityisestablished.**

DESIGNPROCESSANDDESIGNQUALITY:

Softwaredesignisaniterativeprocessthroughwhichrequirementsaretranslatedintoa“blueprint”forconstructingthesoftware.

Goalsofdesign:

- Thedesignmustimplementalloftheexplicitrequirementscontainedintheanalysismodel.
- Thedesignmustbeareadable,understandableguide.
- Thedesignshouldprovideacompletepictureofthesoftware

Qualityguidelines:

- Adesignshouldexhibitanarchitecturethat,
  - **a.**hasbeencreatedusingrecognizablearchitecturalstylesorpatterns
  - **b.**iscomposedofcomponentsthatexhibitgooddesigncharacteristics and
  - **c.**canbeimplementedinanevolutionaryfashion,therebyfacilitatingimplementationandtesting.
- Adesignshouldbemodular.
- Adesignshouldcontaindistinctrepresentationofdata,architecture,interfacesandcomponents.

- leadtodatastructuresthatareappropriatefortheclassestobeimplemented.
- leadtocomponentsthatexhibitindependentfunctionalcharacteristics.
- leadtointerfacethatreducethecomplexityofconnectionsbetweencomponentsandwiththeexternalenvironment.
- shouldbederivedusingarepeatablemethodthatisdrivenbyinformationobtainedduringsoftwarerequirementsanalysis.
- shouldberepresentedusinganotationthateffectivelycommunicatesitsmeaning.

Qualityattributes: (FURPS)

- ***Functionality***
- ***Usability***
- ***Reliability***
- ***Performance***

NRCM
your tools to success

DESIGNCONCEPTS:

Fundamentalsoftwaredesignconceptsprovidethenecessaryframeworkfor“ gettingitright.”

1. Abstraction: Manylevelsofabstractionarethere,

- Atthehighestlevelofabstraction,asolutionisstatedinbroadtermsusin gthelanguageoftheproblemenvironment.
- Atlowerlevelsofabstraction,amoredetaileddescriptionofthesolution isprovided.
- A***proceduralabstraction***referstoasequenceofinstructionsthathavea specificandlimitedfunction.
- A***dataabstraction***isanamedcollectionofdatathatdescribesadataobje ct.

2. Architecture:

- theoverallstructureofthesoftwareandthewaysinwhichthatstructureprovidesconceptualintegrityforasystem
  - ***Structuredmodels***-organizedcollectionofprogramcomponents.
  - ***Frameworkmodels***-increasethelevelofdesignabstraction.
  - ***Dynamic models***-addressthebehavioralaspectsoftheprogramarchitecture,indicatinghowthestructureorsystemconfigurationmaychangeasafunctionexternalevents.
  - ***Processmodels*** -focusonthedesignofthebusinessortechnicalprocessthatthesystemmustaccommodate.
  - ***Functionalmodels***-canbeusedtorepresentthefunctionalhierarchyofasystem.

3. Patterns:

- adesignpatterndescribesadesignstructurethatsolvesaparticulardesign withinaspecificcontextandamid“forces”thatmayhaveanimpactonthe mannerinwhichthepatternisappliedandused.

- Theintentofeachdesignpatternistoprovideadescriptionthatenabl esadesignertodetermine

  1) Whetherthepatterniscapabletothecurrentwork,

  2) Whetherthepatterncanbereused,

  3) Whetherthepatterncanserveasaguidefordevelopingasimilar,butfun ctionallyorstructurallydifferentpattern.

4. Modularity

- softwareisdividedintoseparatelynamedandaddressablecomponents,sometimescalled***modules***thatareintegratedtosatisfyproblemrequirements.

5. InformationHiding:

- information contained within a module is inaccessible to othermodulesthathavenoneedforsuchinformation.

6. FunctionalIndependence:

- directoutgrowthofmodularityandtheconceptsofabstractionandinformationhiding.
- Independenceisassessedusingtwoqualitativecriteria:cohesionandcoupling.
  - *Cohesion*isanindicationoftherelativefunctionalstrengthofamodule.
  - *Coupling*isanindicationoftherelativeinterdependenceamongmodules.

7. Refinement:

- Aprogramisdevelopmentbysuccessivelyrefininglevelsofproceduraldetail.
- Refinementisactuallyaprocessofelaboration.

8. Refactoring:

- Refactoringis areorganizationtechniquethatsimplifiesthedesignof acomponentwithoutchangingitsfunctionorbehavior.

9. Designclasses:

- Thesoftwareteammustdefineasetofdesignclassesthat,
  - 1.Refinetheanalysisclassesbyprovidingdesigndetailthatwillenablethec lassestobeimplemented,and
  - 2.Createanewsetofdesignclassesthatimplementasoftwareinfrastructure tosupportthedesignsolution.

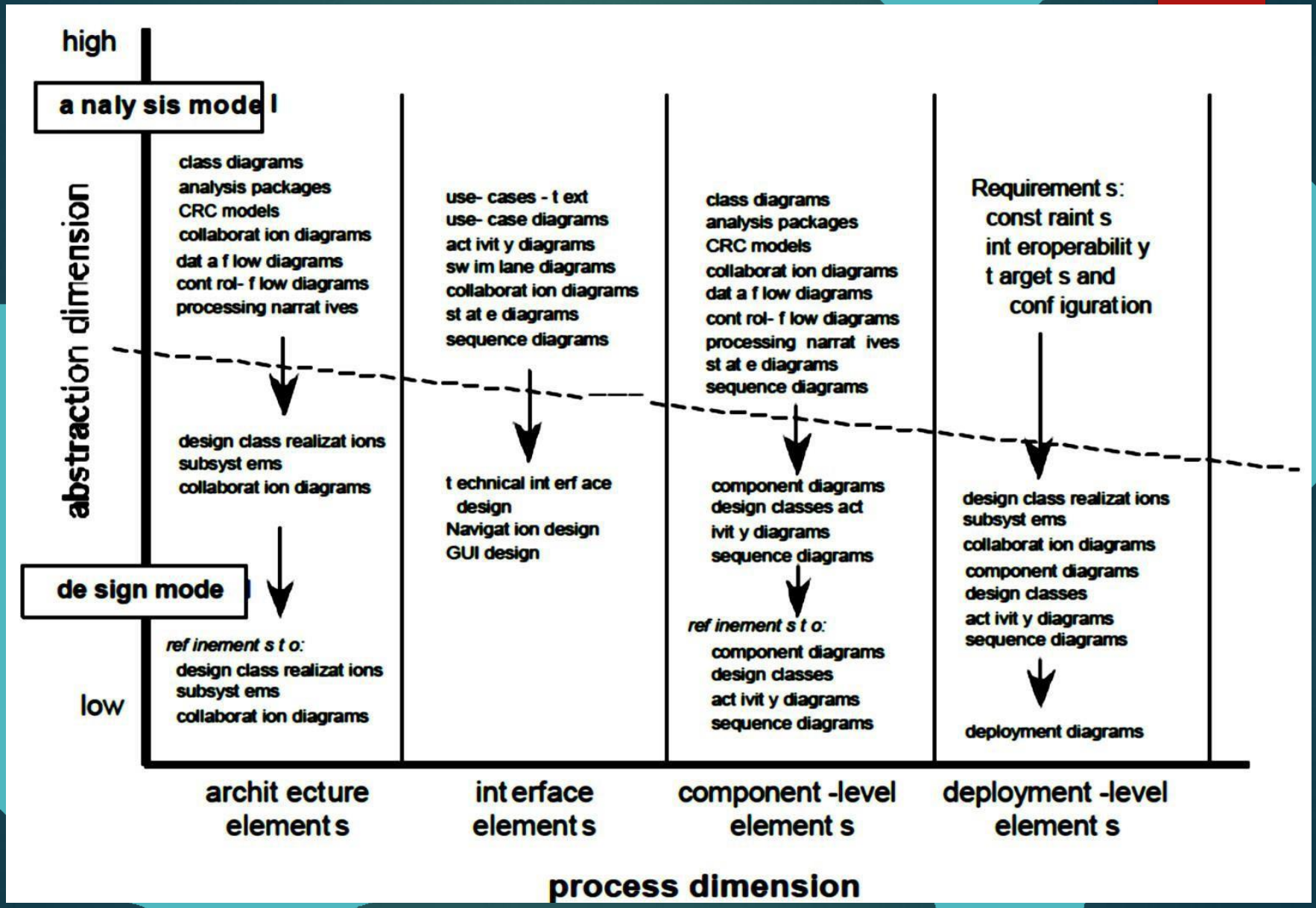
Fivedifferenttypesofdesignclasses,

- **Userinterfaceclasses:**defineallabstractionsthatarenecessaryforhumancomputerinteraction.
- **Businessdomain classes:** are often refinements of the analysisclassesdefinedearlier.
- **Process classes** implement lower – level business abstractionsrequiredtofullymanagethebusinessdomainclasses.
- **Persistentclasses**representdatastoresthatwillpersistbeyondtheexecutionofthesoftware.
- **Systemclasses**implementsoftwaremanagementandcontrolfunctionsthatenablethesystemtooperateandcommunicatewithinitscomputingenvironmentandwiththeoutsideworld.

Fourcharacteristicsofawell-formeddesignclass,

- Completeandsufficient
- Primitiveness
- Highcohesion
- Lowcoupling

THEDESIGNMODEL:



i.Datadesignelements:

- createsamodelofdataand/orinformationthatisrepresentedatahighlevelofabstraction.
- Thestructureofdatahasalwaysbeenanimportantpartofsoftwaredesign.
  - Atthe**programcomponentlevel**
  - At**theapplicationlevel**
  - Atthe**businesslevel**

ii. **Architecturaldesignelements:**

- The*architecturaldesign*forsoftwareistheequivalenttothefloorplanofahouse.Thearchitecturalmodelisderivedfromthreesources.
  - a.Informationabouttheapplicationdomainforthesoftwaretobebuilt.
  - b.Specificanalysismodelelements,and
  - c.Theavailabilityofarchitecturalpatterns.

iii. **Interfacedesignelements:**

- equivalenttoasetofdetaileddrawingsforthedoors,windows,andexternalutilitiesofahouse.
- Thereare3importantelementsofinterfacedesign:
  - a.Theuserinterface(UI);
  - b.Externalinterfacestoothersystems,devices,networks,orotherproducesorconsumersofinformation;and
  - c.Internalinterfacesbetweenvariousdesigncomponents.

iv. **Component-leveldesignelements:**

- Itisequivalenttoasetofdetaileddrawings.
- Thecomponent-leveldesignforsoftwarefullydescribestheinternaldetailofeachsoftwarecomponent.

v. **Deployment-leveldesignelements:**

- Itindicateshowsoftwarefunctionalityandsubsystemswillbeallocatedwithinthephysicalcomputingenvironmentthatwillsupportthesoftware.

ARCHITECTURALDESIGNSOFTWAR

EARCHITECTURE:

WhatIsArchitecture?

- Architectural design represents the structure of data and programcomponentsthatarerequiredtobuildacomputer-basedsystem.
- Thedesignofsoftwarearchitectureconsiderstwolevelsofthedesignpyramid
  - -datadesign
  - -architecturaldesign.

WhyIsArchitectureImportant?

- Representationsofsoftwarearchitectureareanenablerforcommunicationbetweenallparties(stakeholders)interestedinthedevelopmentofacomputer-basedsystem.
- Thearchitecturehighlightsearlydesigndecisionsthatwillhaveaprofoundimpactonallsoftwareengineeringworkthatfollowsand,asimportant,ontheultimatesuccessofthesystemasanoperationalentity.
- Architecture“constitutesarelativelysmall, intellectuallygraspablemodelofhowthesystemisstructuredandhowitscomponentsworktogether”.

DATADESIGN:

- translatesdataobjectsaspartoftheanalysismodelintodatastructuresatthesoftwarecomponentleveland,whennecessary,adatabasearchitectureattheapplicationlevel.

1. **DatadesignattheArchitecturalLevel**:

- Thechallengeforabusinesshasbeentoextractusefulinformationfromthisdataenvironment,particularlywhentheinformationdesirediscrossfunctional.

2. **DatadesignattheComponentLevel**:

- focusesontherepresentationofdatastructuresthataredirectlyaccessedbyoneormoresoftwarecomponents.

Thefollowingsetofprinciplesfordataspecification:

1. Thesystematicanalysisprinciplesappliedtofunctionandbehaviorshouldalsobeappliedtodata.
2. Alldatastructuresandtheoperationstobeperformedoneachshouldbeidentified.
3. Adatadictionaryshouldbeestablishedandusedtodefinebothdataandprogramdesign.
4. Low-leveldatadesigndecisionsshouldbedeferreduntillateinthedesignprocess.
5. Therepresentationofdatastructureshouldbeknownonlytothosemodulesthatmustmake directuseofthedatacontainedwithinthestructure.
6. Alibraryofusefuldatastructuresandtheoperationsthatmaybeappliedtothemshouldbedeveloped.
7. Asoftwaredesignandprogramminglanguageshouldsupportthespecificationandrealizationofabstractdatatypes.

ARCHITECTURALSTYLESANDPATTERNS:

- Thesoftwarethatisbuiltforcomputer-basedsystemsalsoexhibitsoneofmanyarchitecturalstyles.
- Eachstyledescribesasystemcategorythatencompasses
  - ***(1)*** Asetof***components***
  - ***(2)*** Asetof***connectors***
  - ***(3) Constraints***
  - ***(4) Semanticmodels***

An***architectural pattern,***imposesatransformationthedesign ofarchitecture.However,apatterndiffersfromastyleinanumberoffundamental ways:

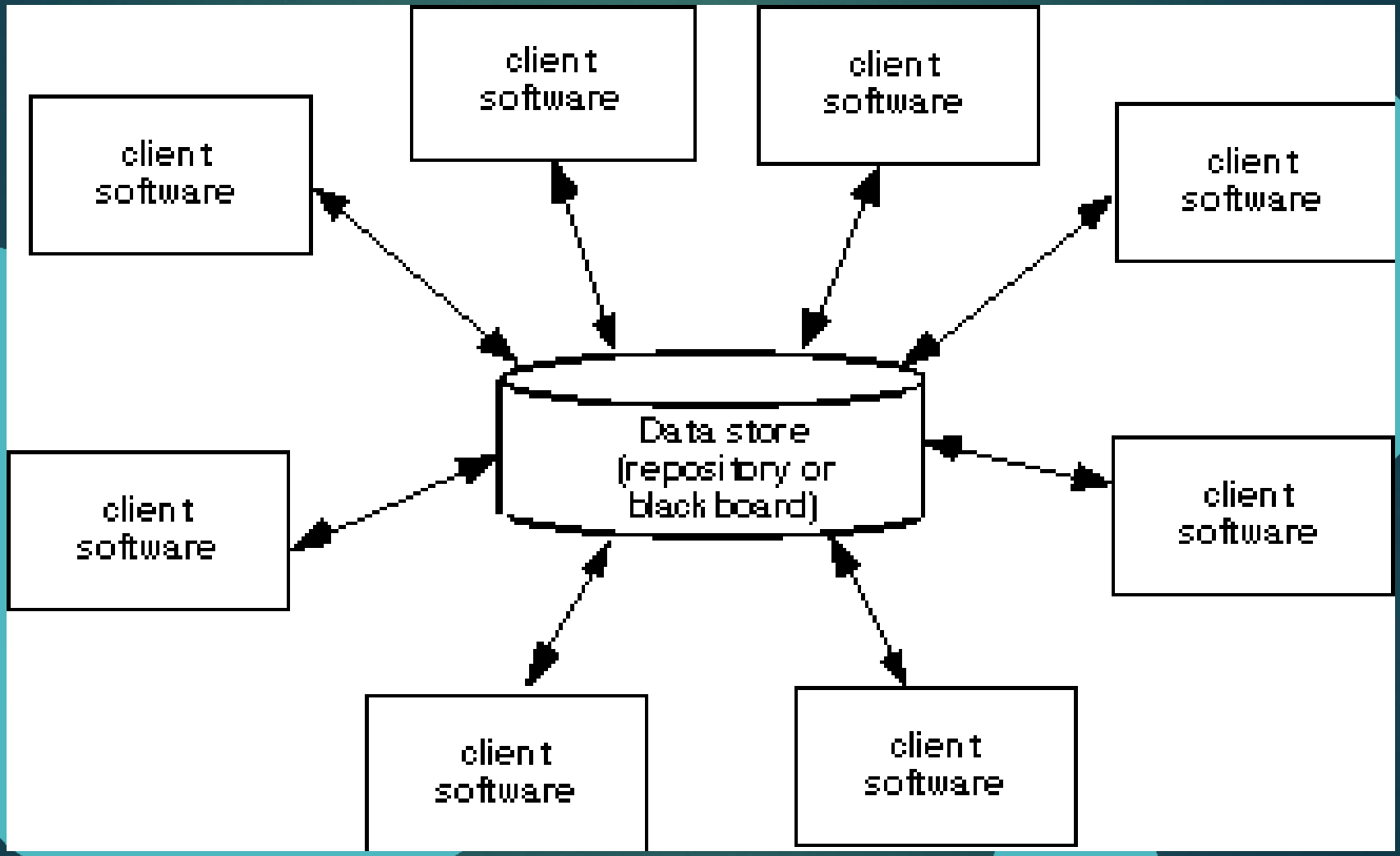
(1) Thescopeofapatternislessbroad,focusingononeaspectofthearchitect ureratherthanthearchitectureinitsentirety.

(2) Apatternimposesaruleonthearchitecture,describinghowthesoftware willhandlesomeaspectofitsfunctionalityattheinfrastructurelevel.

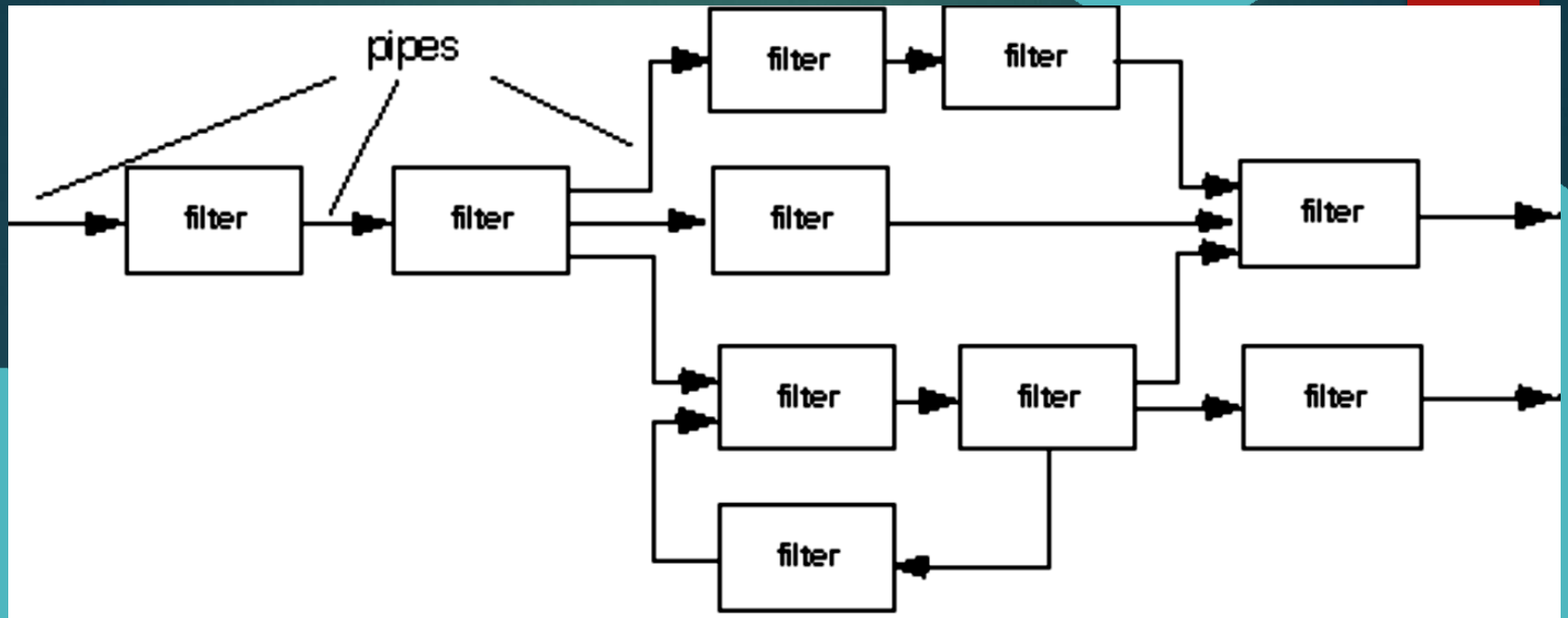
(3) Architecturalpatternstendtoaddressspecificbehavioralissueswithint hecontextofthearchitectural

1. ABriefTaxonomyofStylesandPatternsDa

ta-centeredarchitectures:

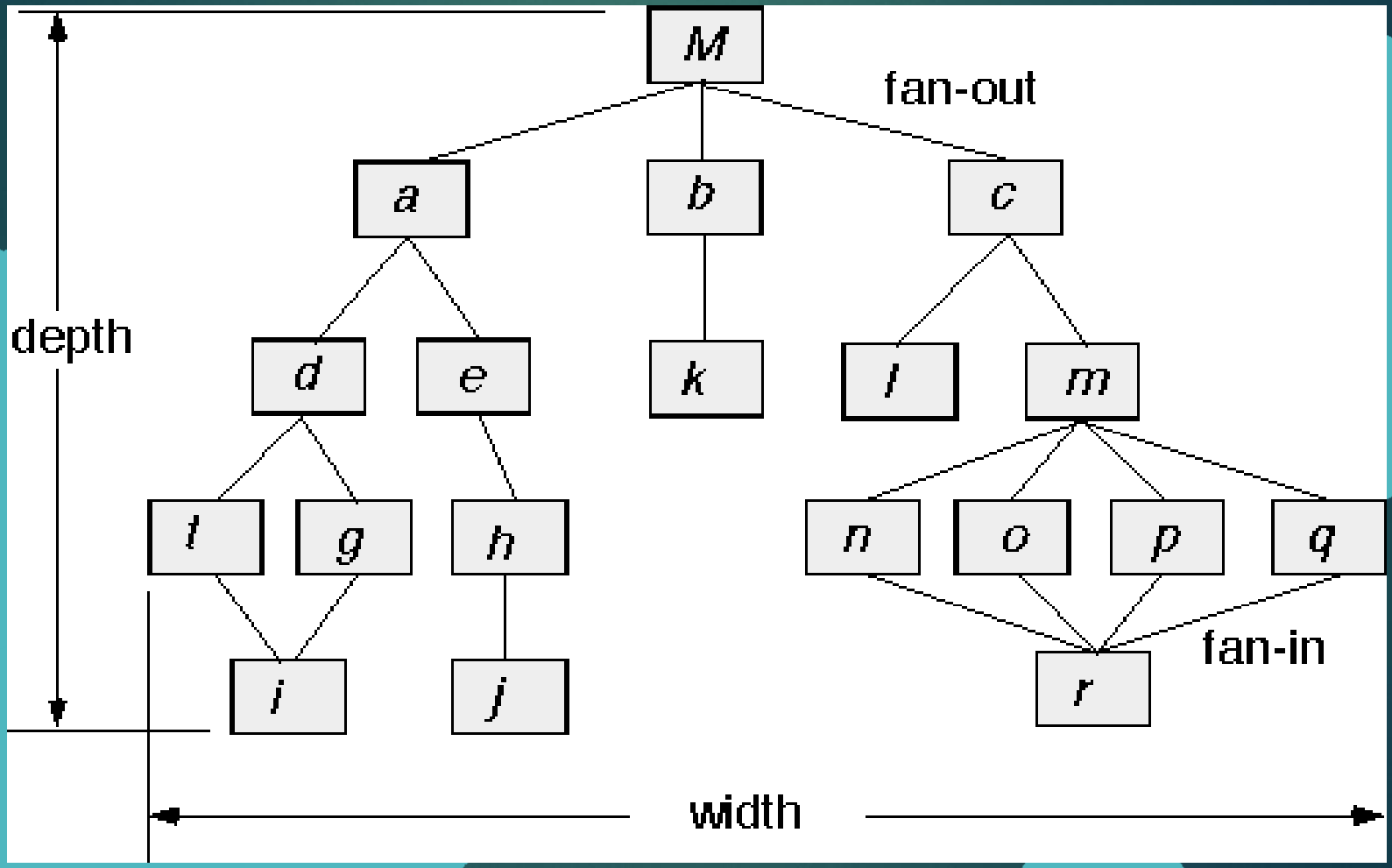


Data-flowarchitectures:



Callandreturnarchitectures:

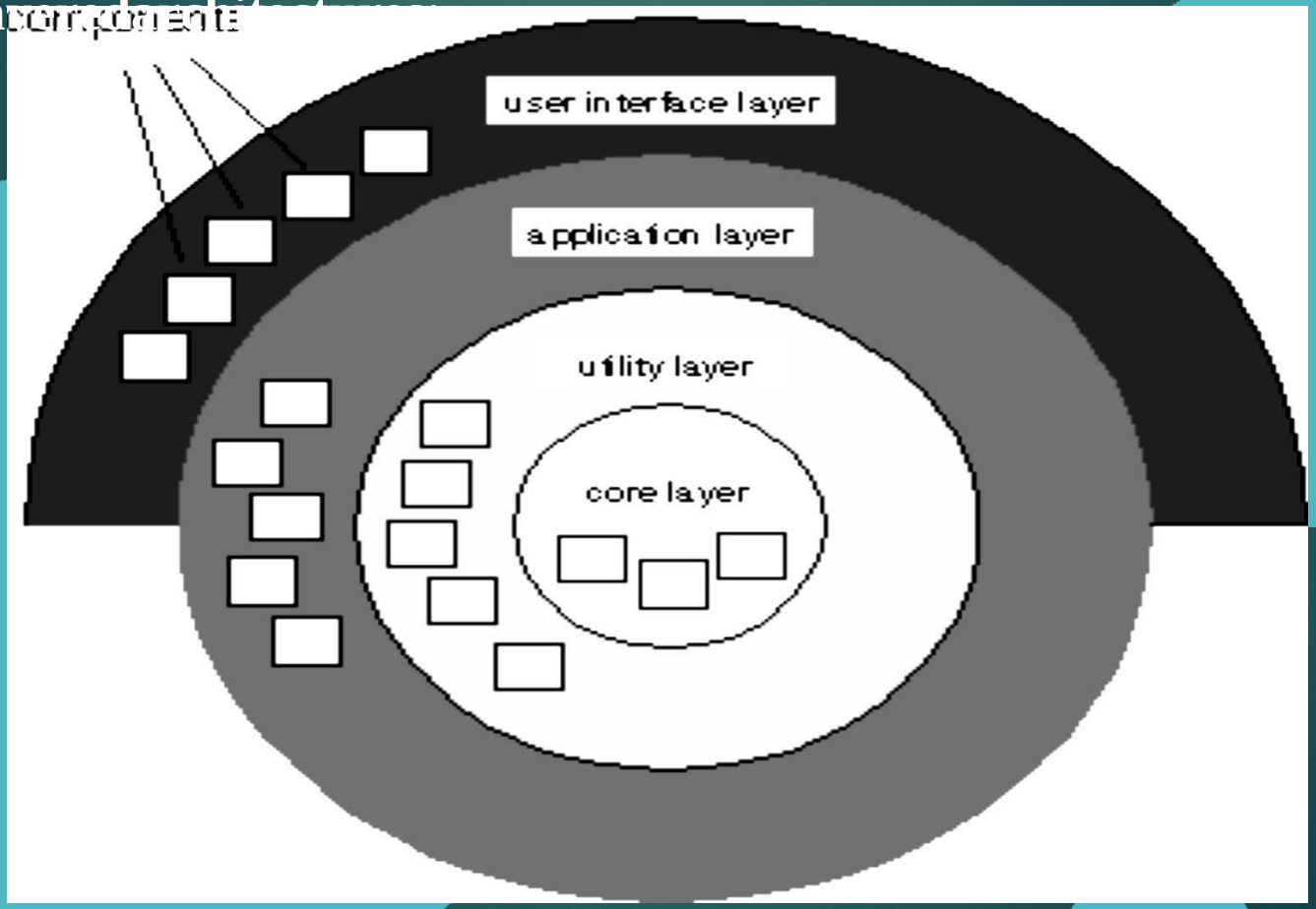
- ***Mainprogram/subprogramarchitectures.***
- ***Remoteprocedurecallarchitectures.***



Object-orientedarchitectures:

- Thecomponentsofasystemencapsulatedataandtheoperationsthatmust beappliedtomanipulatethedata.

La

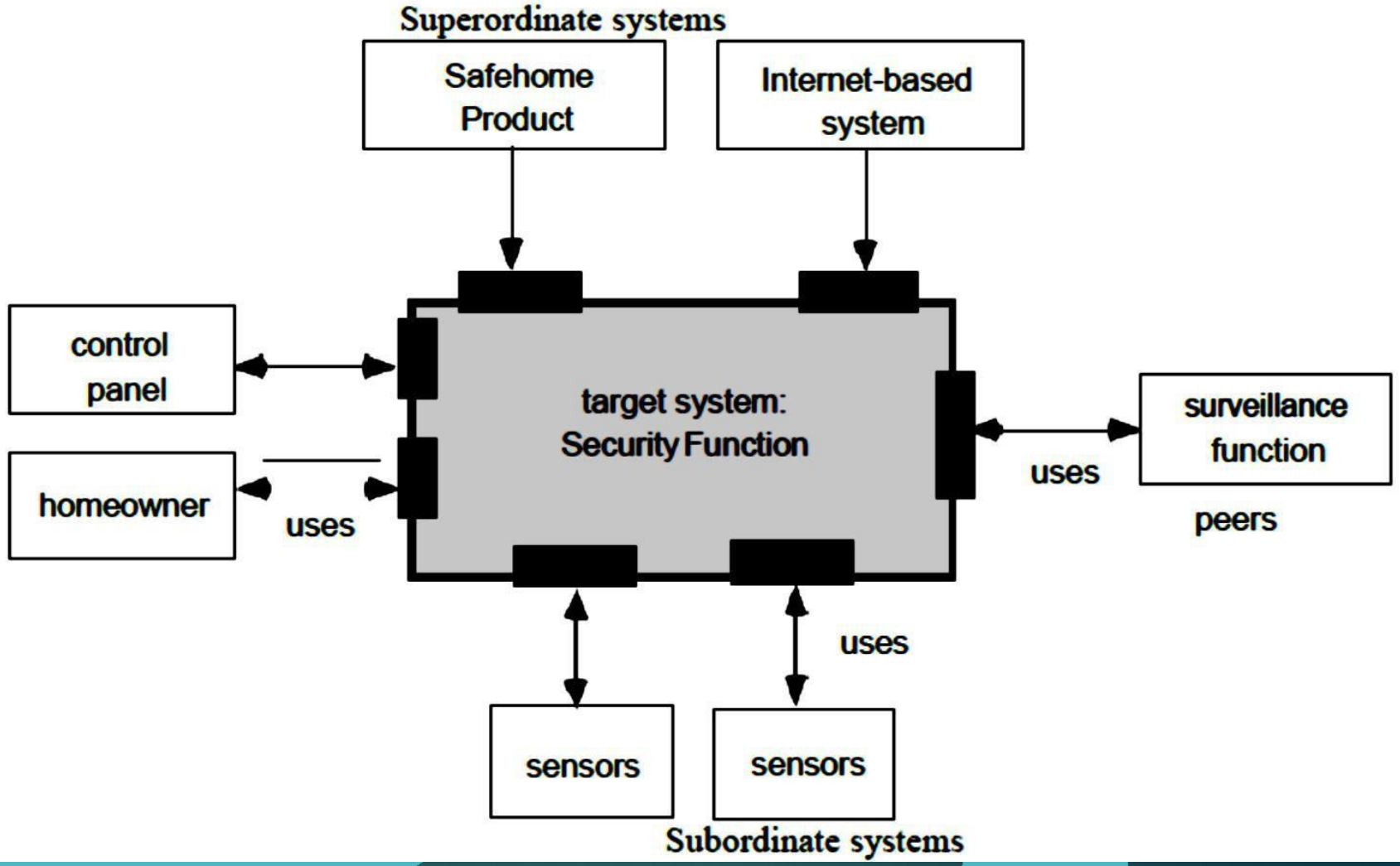


2. ArchitecturalPatterns:

- Thearchitecturalpatternsforsoftwaredefineaspecificapproachforhandlingsomebehavioralcharacteristicsofthesystem,
  - **Concurrency**
    - *operatingsystemprocessmanagement*pattern
    - *taskscheduler*pattern
  - **Persistence**
    - a***databasemanagementsystem***pattern
    - an***applicationlevelpersistence***pattern
  - **Distribution**
- **OrganizationandRefinement:**
  - itisimportanttoestablishasetofdesigncriteriathatcanbeusedtoassessanarchitecturaldesignthatisderived.

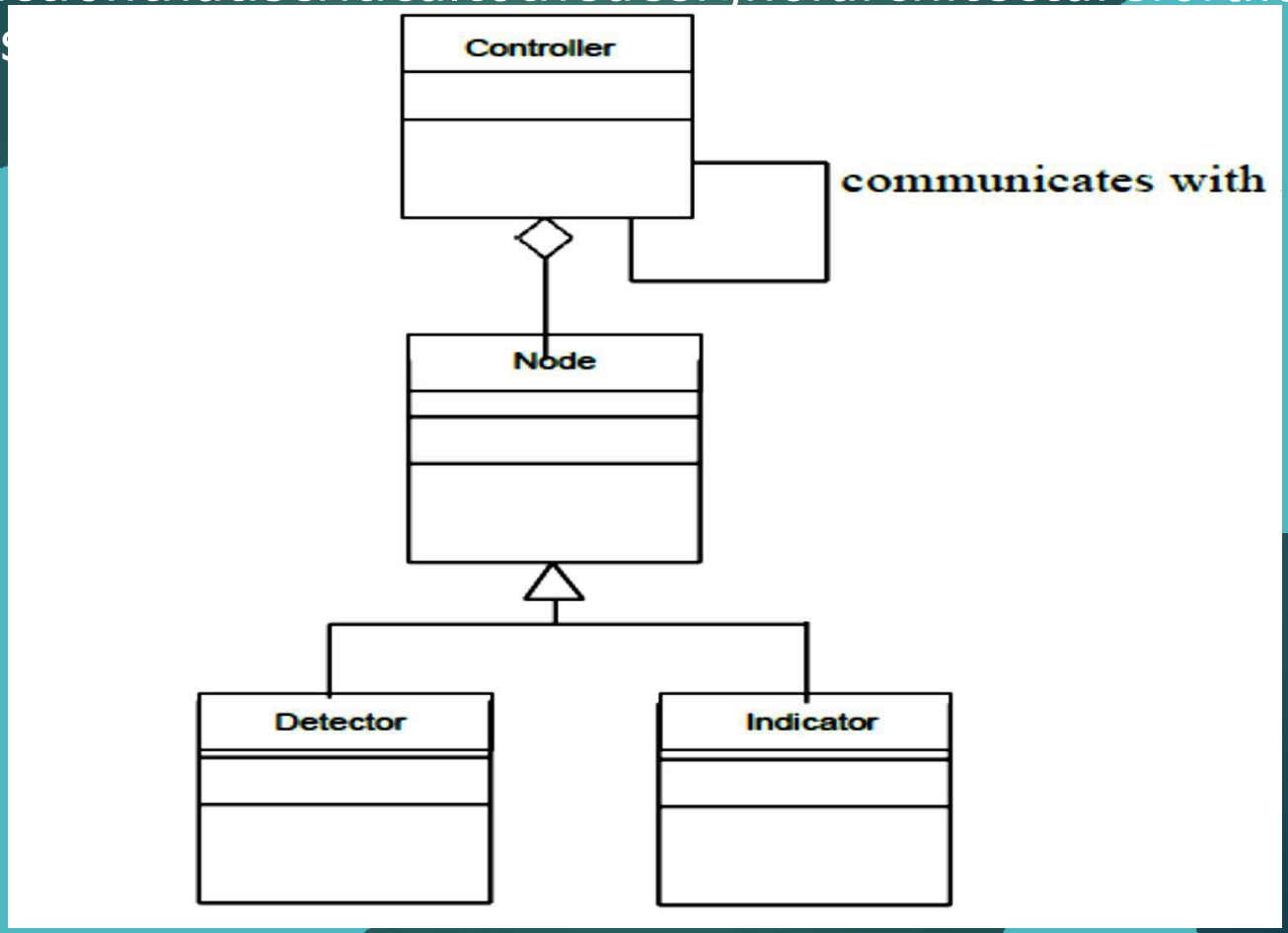
ARCHITECTURALDESIGN:

i. RepresentingtheSysteminContext:



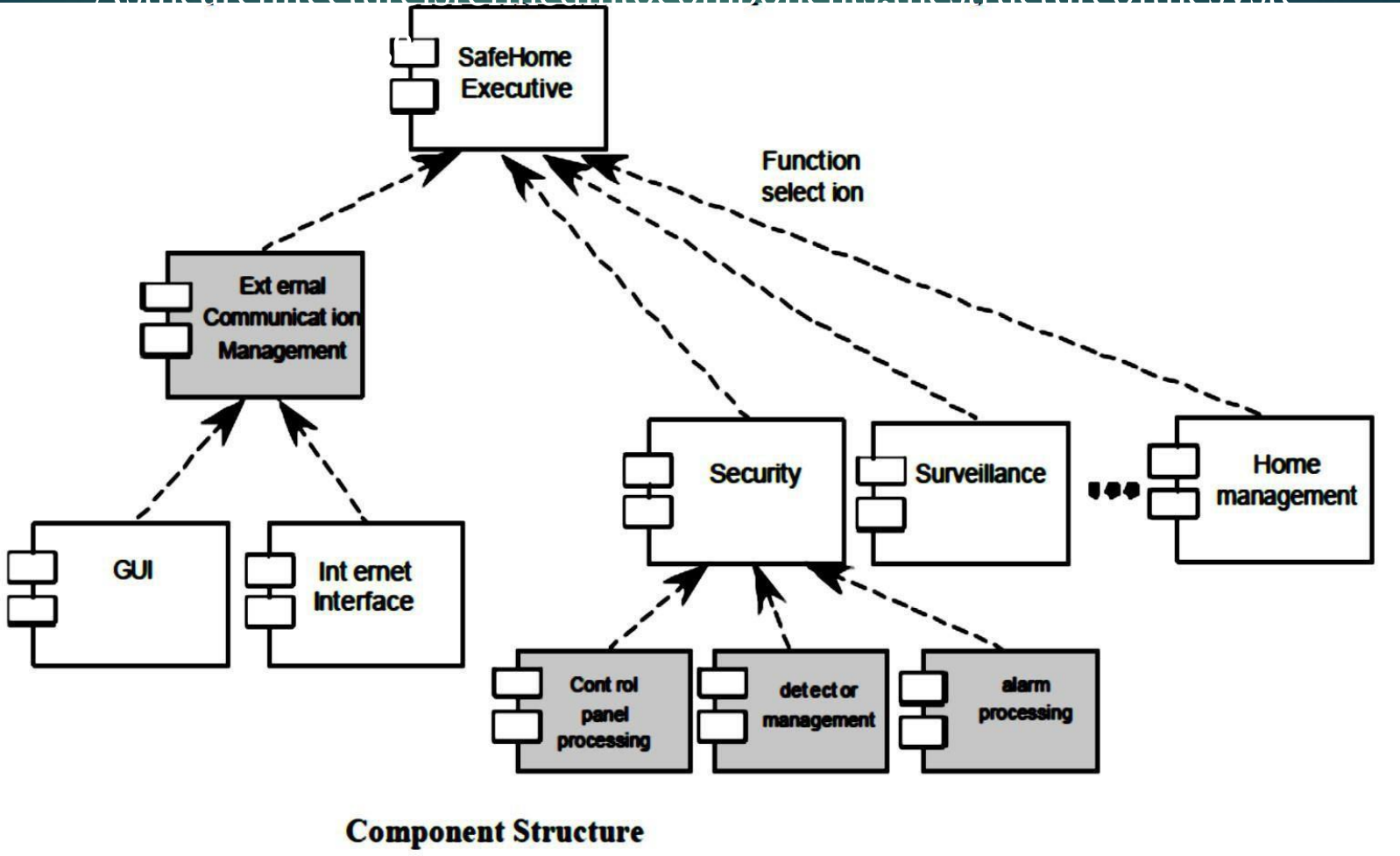
ii. DefiningArchetypes:

- Anarchetypeisaclassorpatternthatrepresentsacoreabstractionthatiscriticaltothedesignofarchitectureforthetargets



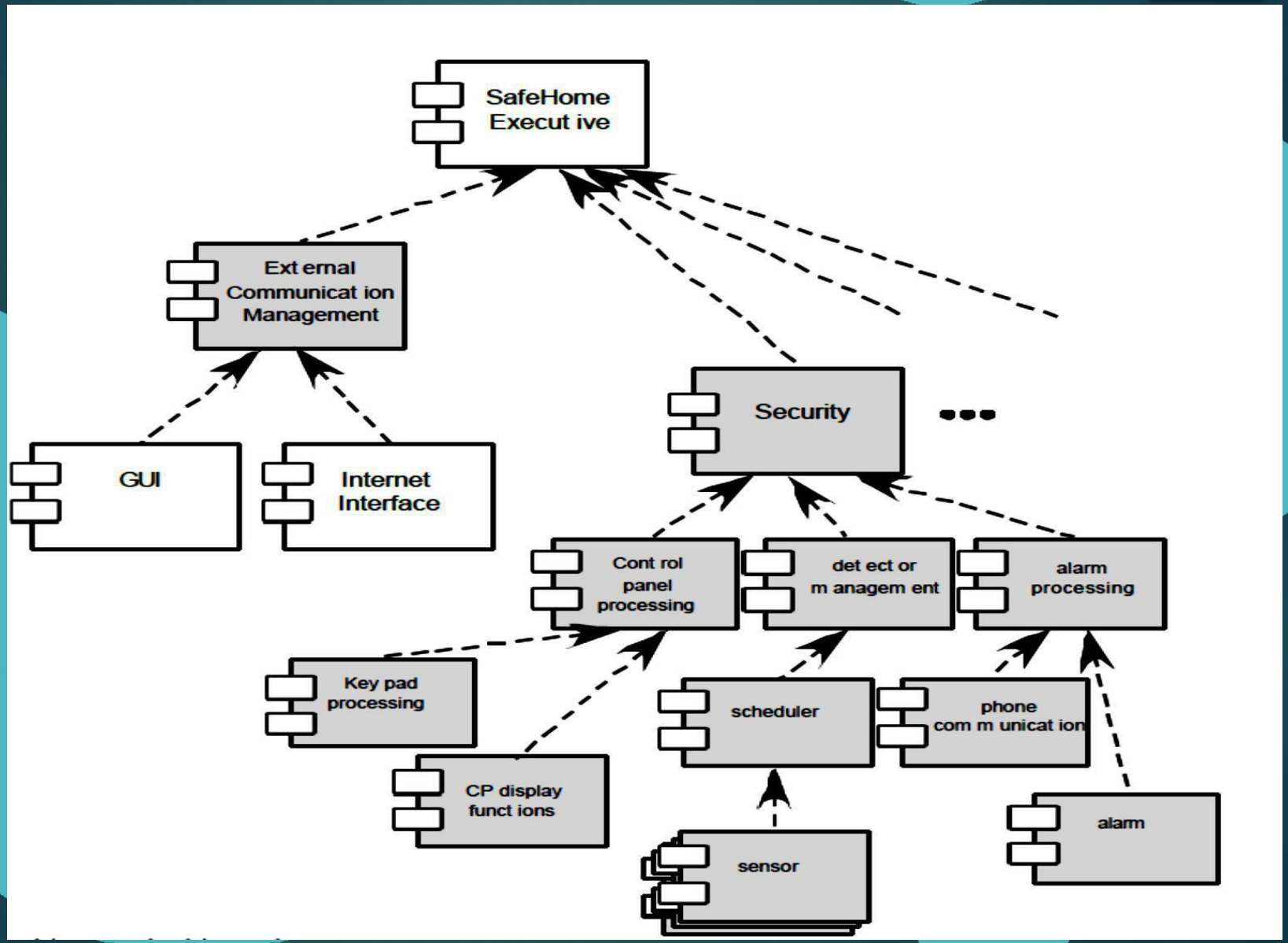
iii. RefiningtheArchitectureintoComponents:

- Asthearchitectureisrefinedintocomponents,thestructureofthesyste



Component Structure

iv. DescribingInstantiations oftheSystem:



AConceptualModelofUML:

- UMLisastandardlanguageforspecifying,visualizing,constructing,anddocumentingtheartifactsofsoftwaresystems.

Object-OrientedConcepts

- UML can be described as the successor of object-oriented (OO)analysisanddesign.
- Followingaresomefundamentalconceptsoftheobject-orientedworld,
- **Objects**−Objectsrepresentanentityandthebasicbuildingblock.
- **Class**−Classistheblueprintofanobject.
- **Abstraction**−Abstractionrepresentsthebehaviorofanrealworldentity.
- **Encapsulation**−Encapsulationisthemechanismofbindingthedatatogetherandhidingthemfromtheoutsideworld.
- **Inheritance**−Inheritanceisthemechanismofmakingnewclassesfromexistingones.
- **Polymorphism**−Itdefinesthemechanismtoexistsindifferentforms.

OOAnalysisandDesign:

- OOcanbedefinedasaninvestigationandtobemorespecific,itistheinvestigationofobjects.Designmeanscollaborationofidentifiedobjects.

- ThepurposeofOOanalysisanddesigncandescribedas−
  - Identifyingtheobjectsofasystem.
  - Identifyingtheirrelationships.
  - Makingadesign,whichcanbeconvertedtoexecutablesusingOOlanguages.

- TherearethreebasicstepswheretheOOconceptsareappliedandimplemented. Thestepscanbedefinedas,

OOAnalysis→OODesign→OOimplementationusingOOlanguages

BasicStructuralModeling:

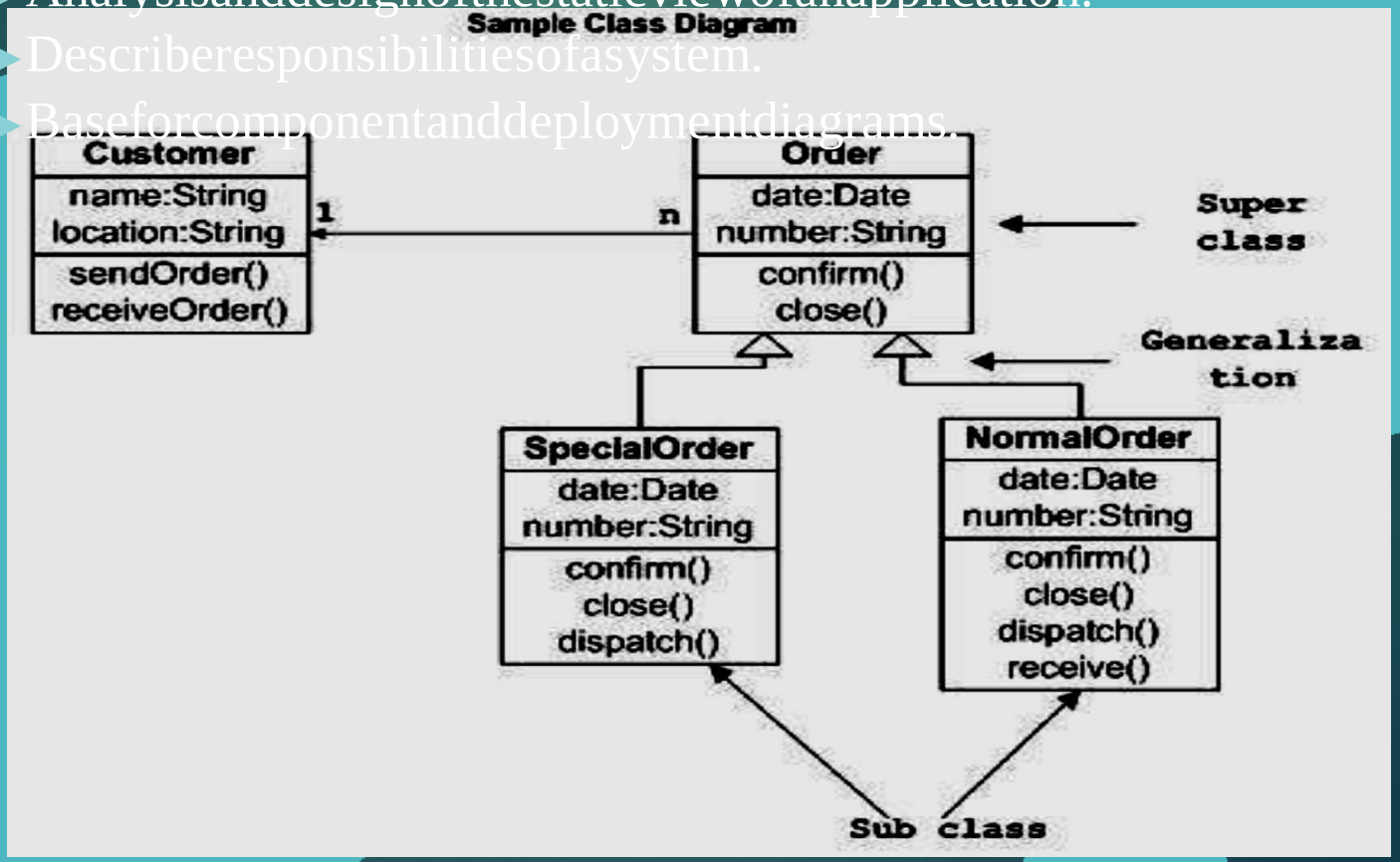
- Structuralmodelingcapturesthestaticfeaturesofasystem.Theyconsistofthefollowing−
  - Classesdiagrams
  - Objectsdiagrams
  - Deploymentdiagrams
  - Packagediagrams
  - Compositestructurediagram
  - Componentdiagram
- Structuralmodelrepresentstheframeworkforthesystemandthisframeworkistheplacewhereallothercomponentsexist.

ClassDiagram:

- Classdiagramisastaticdiagramthatshowsacollectionofclasses,interfaces, associations,collaborations,andconstraints.

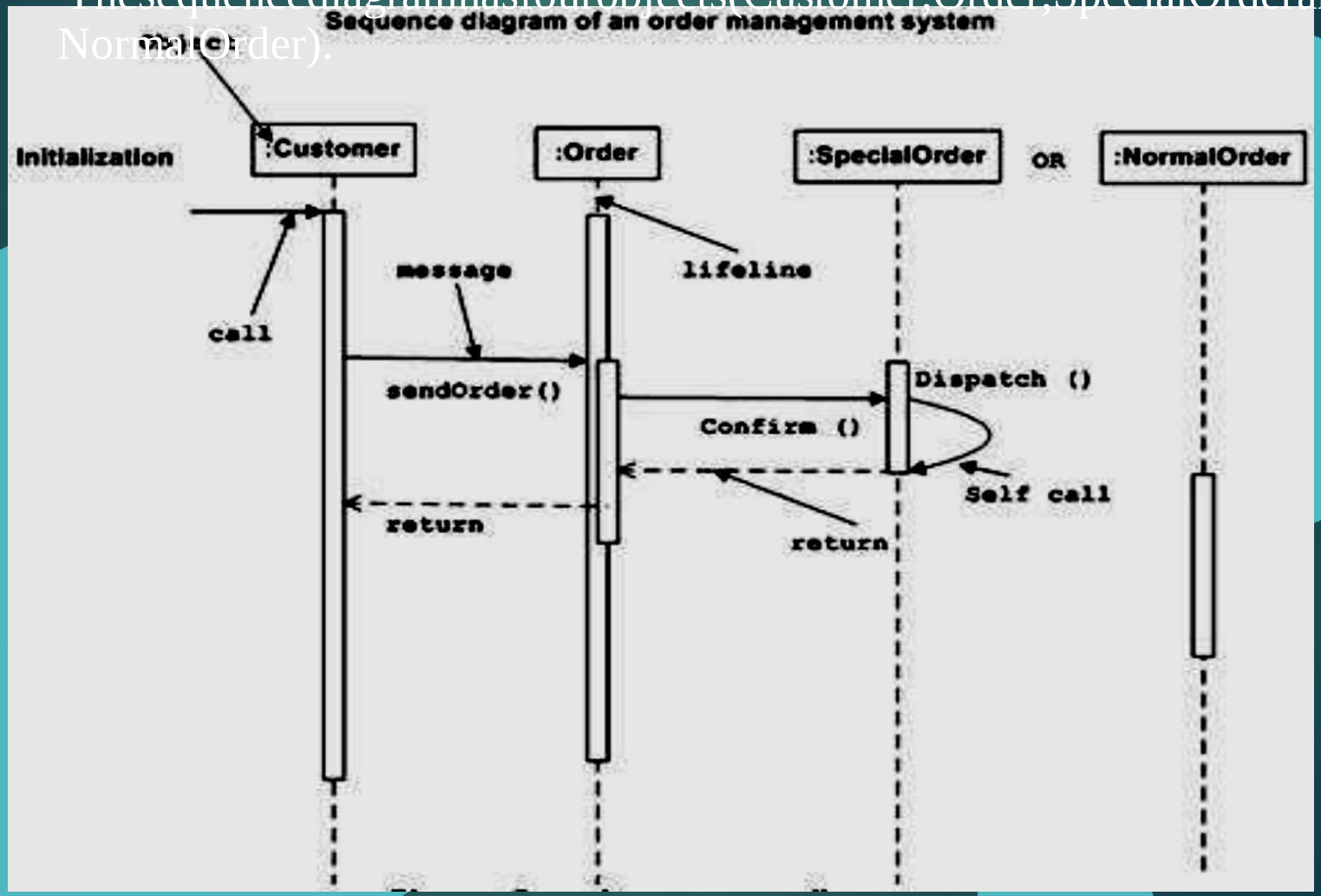
PurposeofClassDiagrams

- Analysisanddesignofthestaticviewofanapplication.
- Describeresponsibilitiesofasystem.
- Baseforcomponentanddeploymentdiagrams.

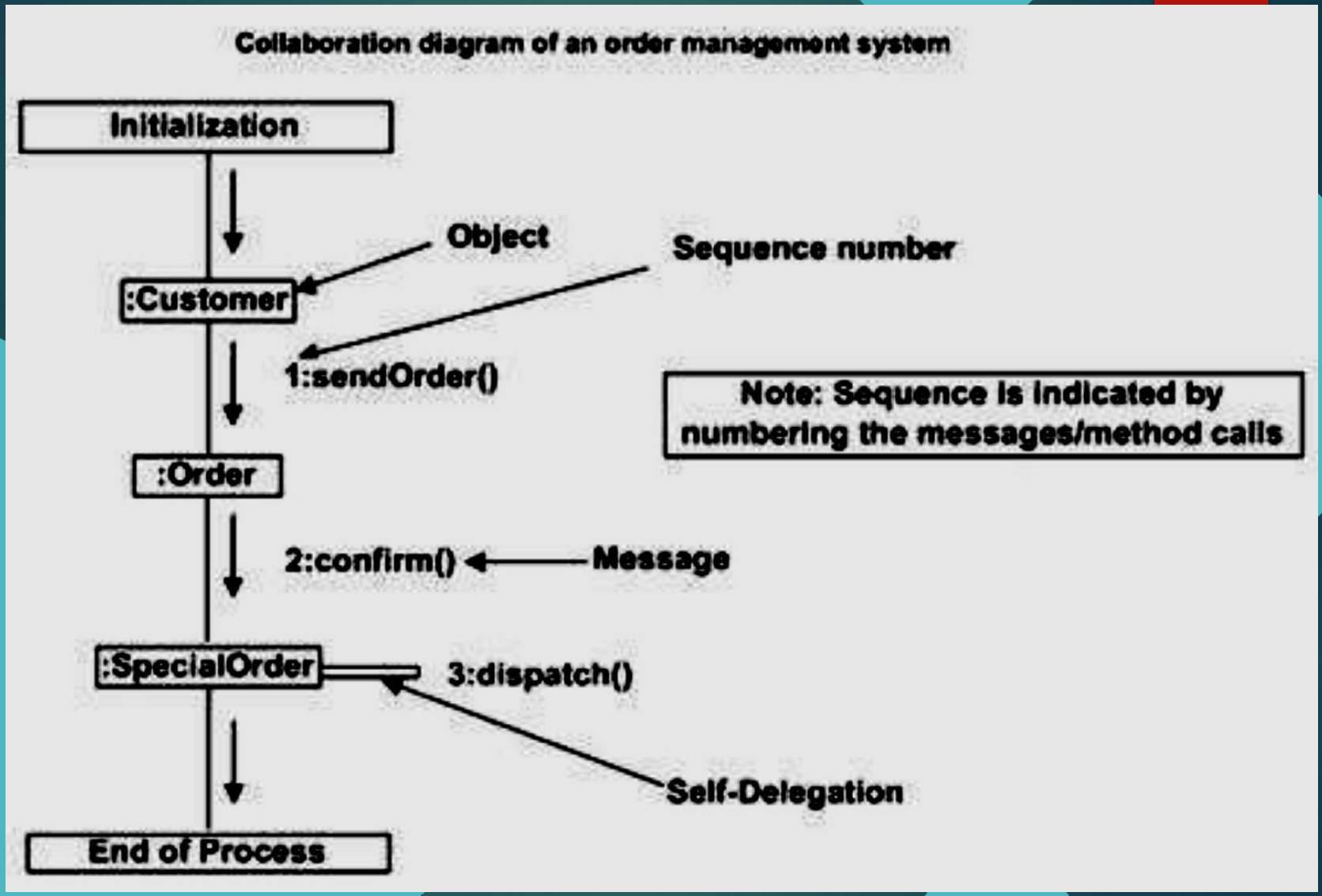


TheSequenceDiagram:

- Thesequencediagramhasfourobjects(Customer,Order,SpecialOrderand NormalOrder).



TheCollaborationDiagram:



UseCaseDiagram:

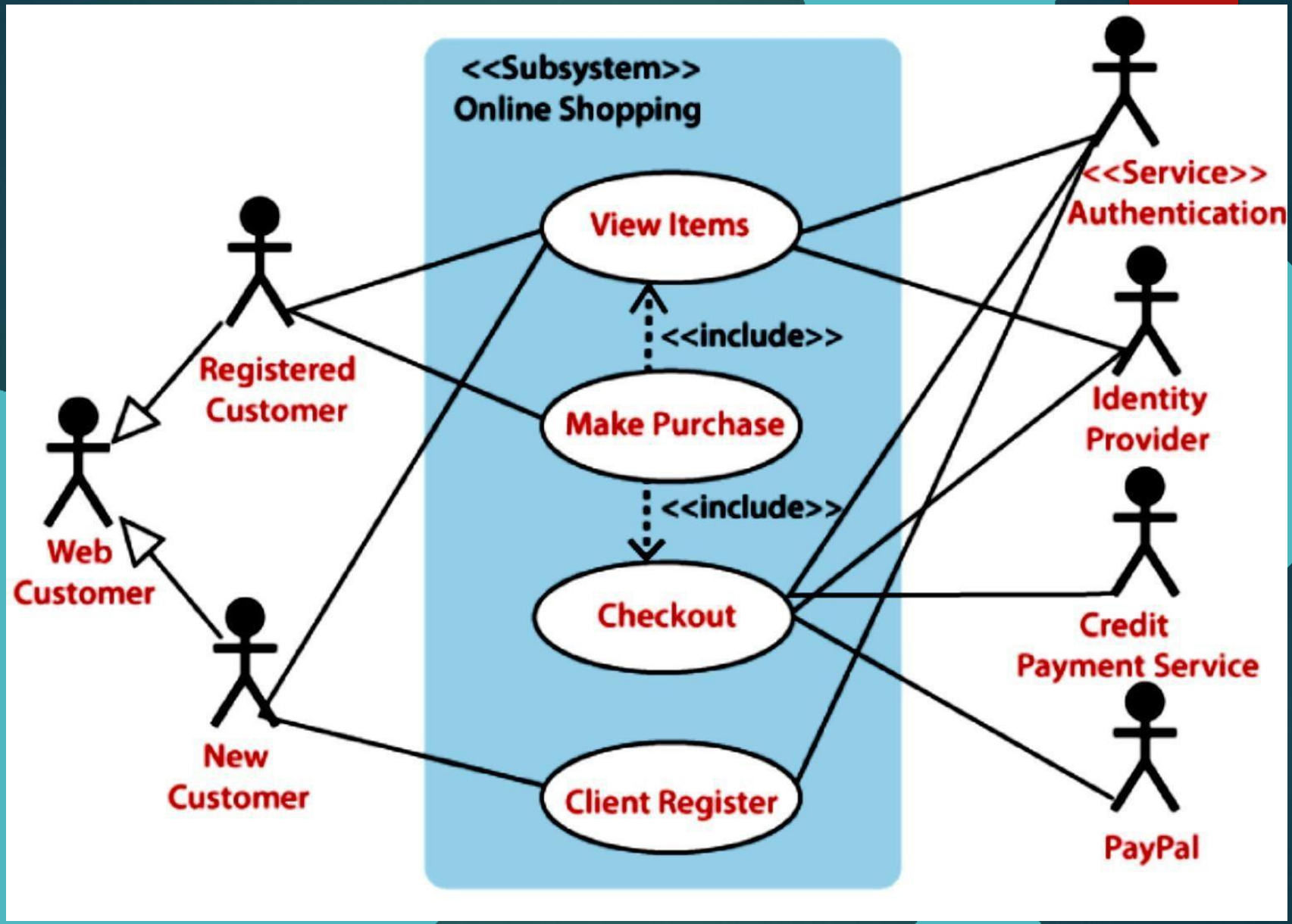
- Itisusedtorepresentthedynamicbehaviorofasystem.
- Itencapsulatesthesystem'sfunctionalitybyincorporatingusecases,actors, andtheirrelationships.

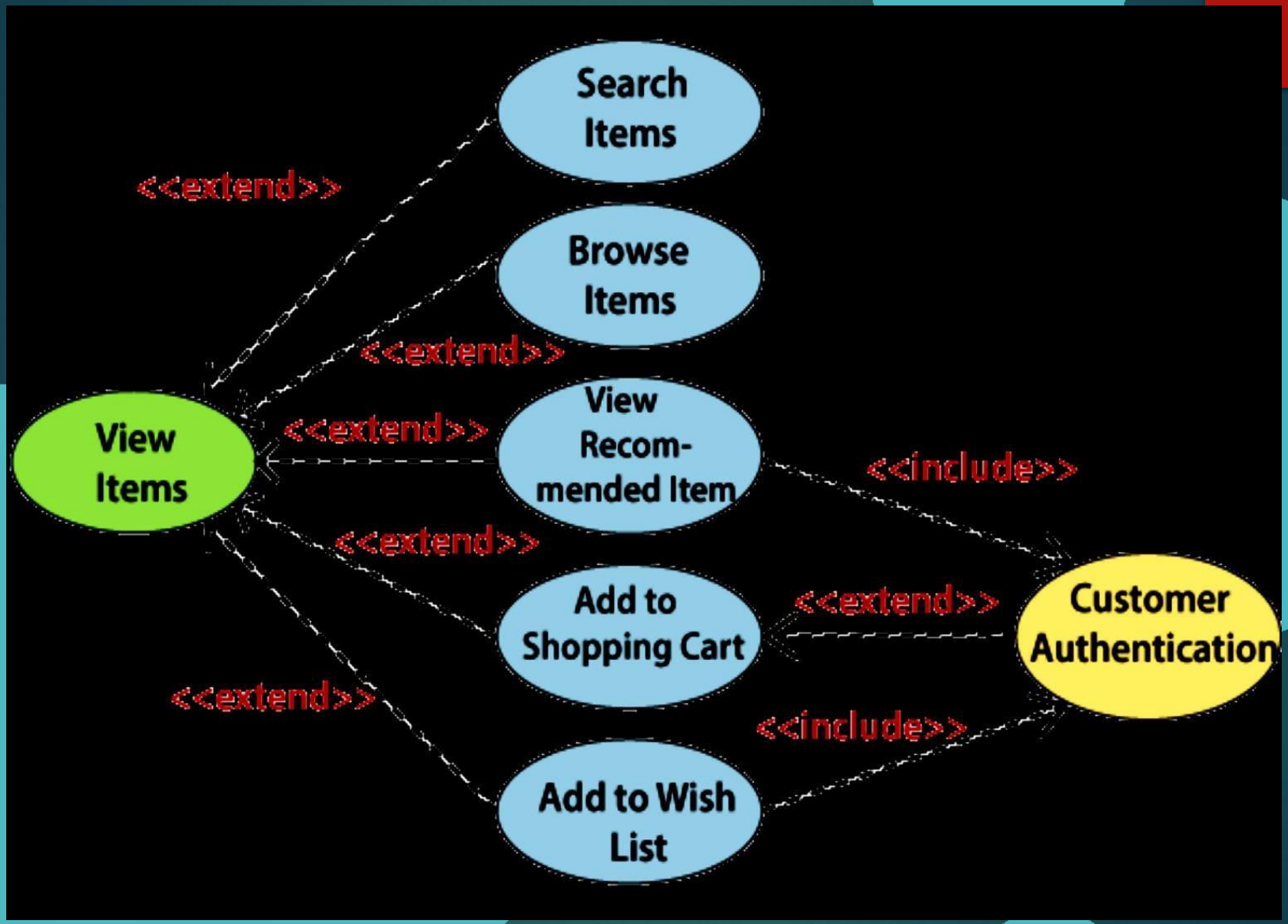
PurposeofUseCaseDiagrams

1. Itgathersthesystem'sneeds.
2. Itdepictstheexternalviewofthesystem.
3. Itrecognizestheinternalaswellasexternalfactorsthatinfluencethesystem.
4. Itrepresentstheinteractionbetweentheactors.

Somerulesthatmustbefollowedwhiledrawingausecasediagram:

1. Apertinentandmeaningfulnameshouldbeassignedtotheactororausecaseofasystem.
2. Thecommunicationofanactorwithausecasemustbedefinedinanunderstandableway.
3. Specifiednotationstobeusedasandwhenrequired.
4. Themostsignificantinteractionsshouldberepresentedamongthemultiplenoofinteractionsbetweentheusecaseandactors.


<<Subsystem>>
Online Shopping
View Items
<<include>>
Make Purchase
<<include>>
Checkout
Client Register
Registered Customer
Web Customer
New Customer
<<Service>>
Authentication
Identity Provider
Credit Payment Service
PayPal


Search Items
<<extend>>
Browse Items
<<extend>>
View Items
<<extend>>
View Recom-mended Item
<<include>>
<<extend>>
Add to Shopping Cart
<<extend>>
Customer Authentication
<<extend>>
<<include>>
Add to Wish List

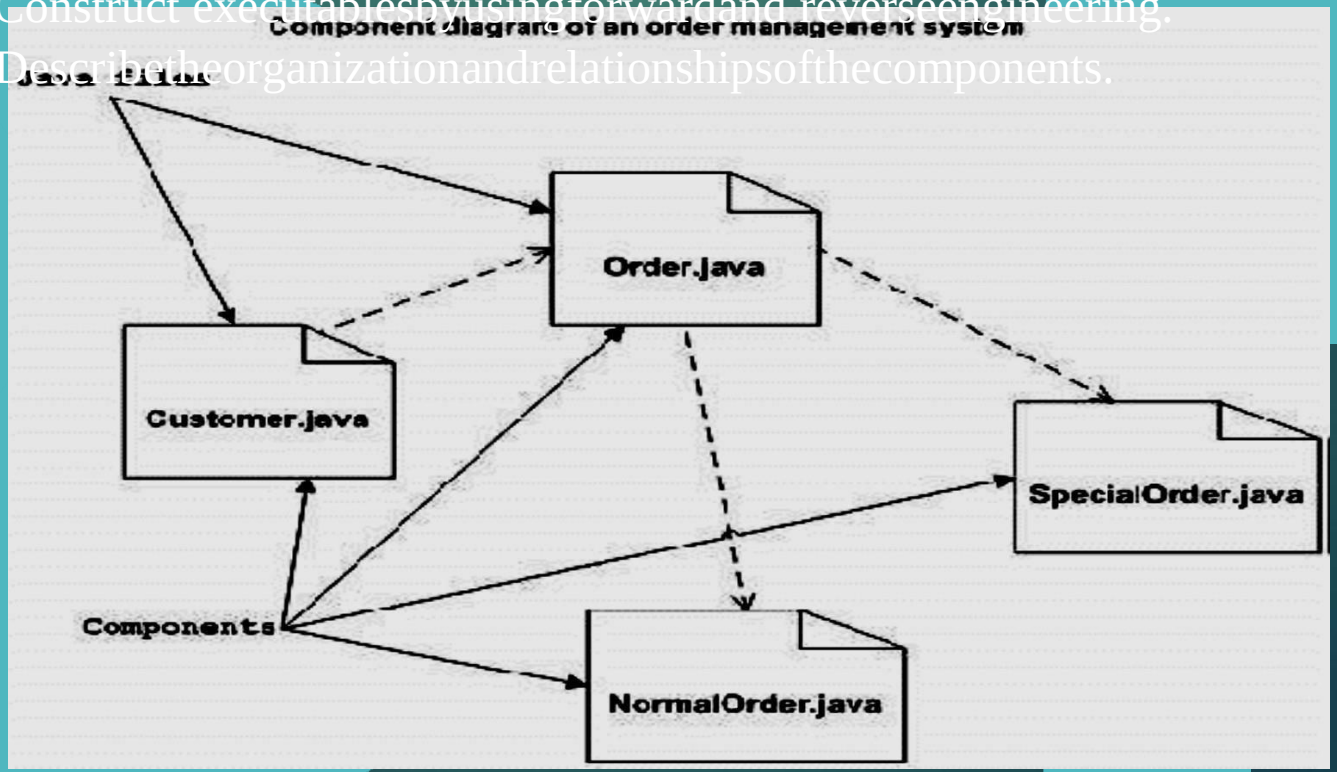
Componentdiagram:

- usedtomodelthephysicalaspectsofasystem.
-

  Physicalaspectsaretheelementssuchasexecutables,libraries,files,document s,etc.whichresideinanode.

PurposeofComponentDiagrams

- Visualizethecomponentsofasystem.
- Construct executablesbyusingforwardand reverseengineering.
- Describetheorganizationandrelationshipsofthecomponents.



NARASIMHAREDDYENGINEERINGCOLLEGE

FACULTY:Dr.VenkateswaruluNaik

CS3102PC:SOFTWAREENGINEERING

III YEARB.TECH. CSE I-SEM

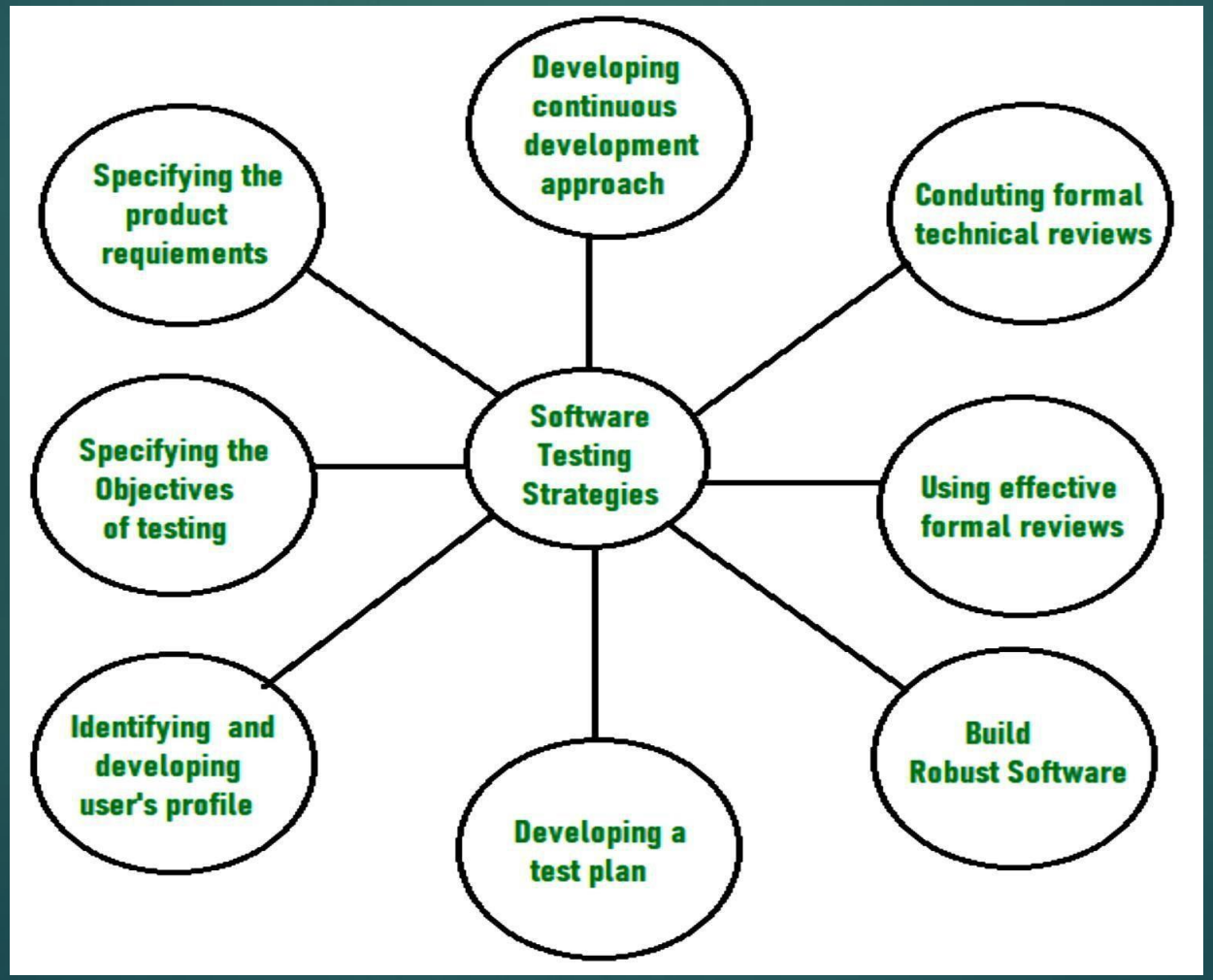
(R20)UNIT-IV

TESTINGSTRATEGIES:ASTRATEGICAPPROACHTOSOFTWARETESTING,TESTSTRATEGIESFORCONVENTIONALSOFTWARE,BLACK-BOXANDWHITE-BOXTESTING,VALIDATIONTESTING,SYSTEMTESTING,THEARTOFDEBUGGING.

PRODUCTMETRICS:SOFTWAREQUALITY,METRICSFORANALYSISMODEL,METRICSFORDESIGNMODEL,METRICSFORSOURCECODE,METRICSFORTESTING,METRICSFORMAINTENANCE.

AstrategicApproachforSoftware testing:

SoftwareTestingisatypeofinvestigationtofind outifthere isanydefault or errorpresentinthe software.



TestingStrategiesforConventionalSoftware:

- youcanwaituntilthesystemisfullyconstructedandthenconducttestsontheoverallsysteminhopesoffindingerrors.
- youcouldconducttestsonadailybasis,wheneveranypartofthesystemisconstructed.

Types:

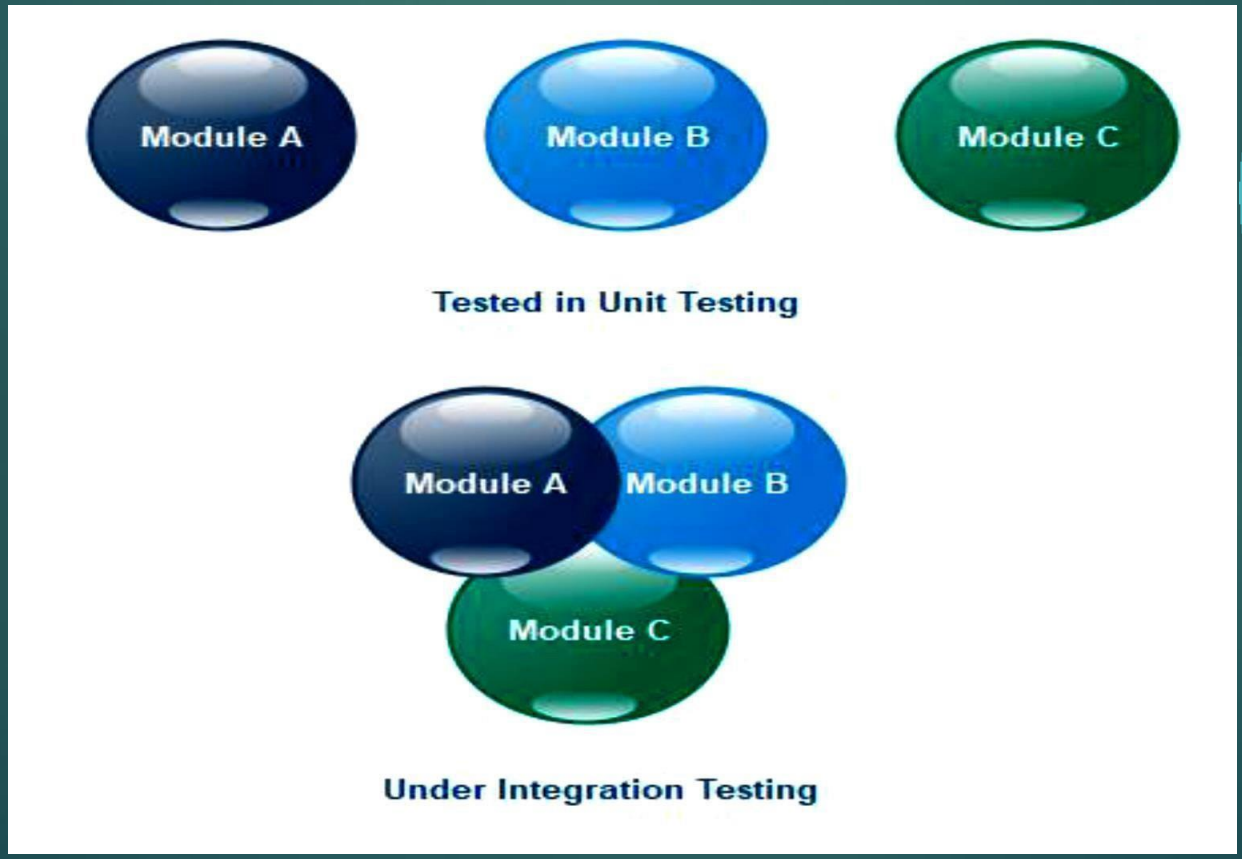
1) UnitTesting
2) IntegrationTesting
3) ValidationTestingand
4) SystemTesting

1) UnitTesting:

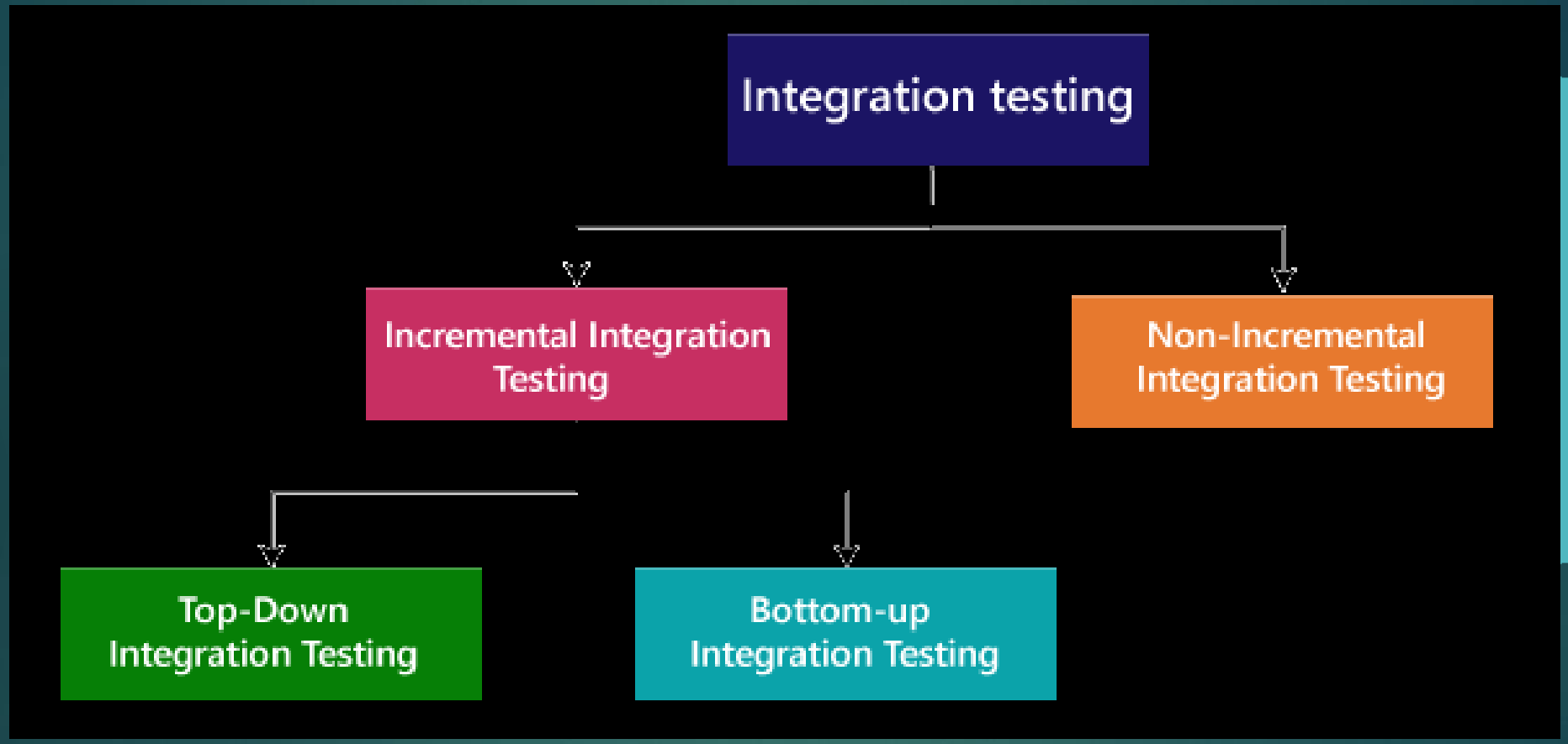
- individualunitsorcomponentsofasoftwareareretested.
- doneduringthedevelopment(codingphase)ofanapplicationbythedevelopers.
- concernedwithfunctionalcorrectnessofthestandalonemodules

2) IntegrationTesting:

- secondlevelofthesoftwaretestingprocesscomesafterunittesting.
- Thefocusoftheintegrationtestinglevelistoexposedefectsatthetime ofinteractionbetweenintegratedcomponentsorunits.



TypesofIntegrationTesting:

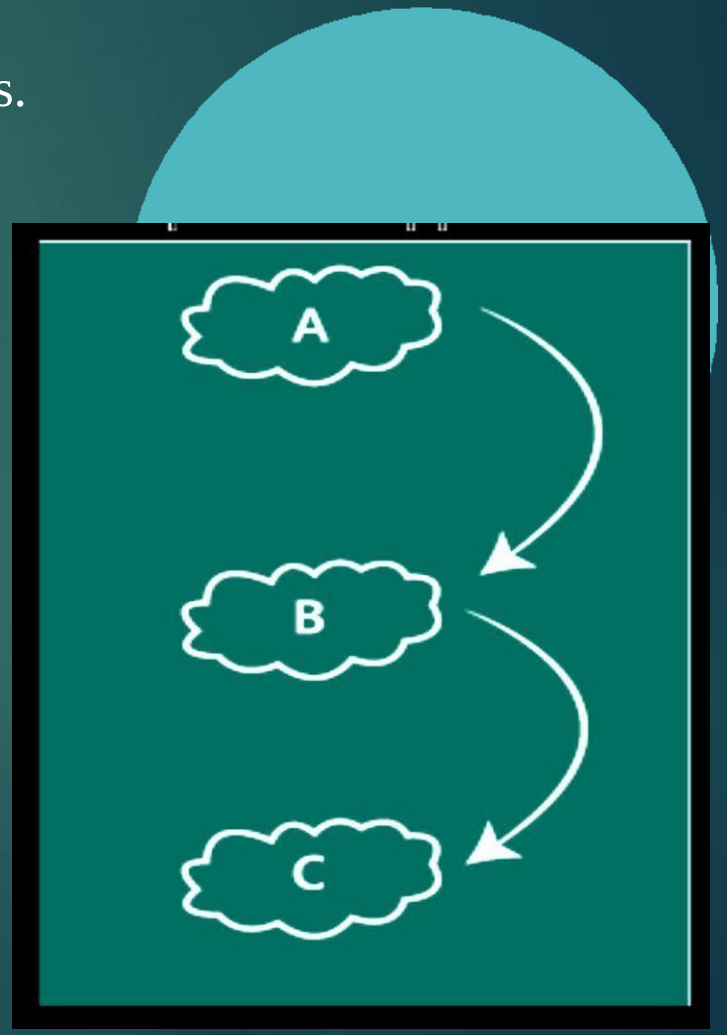


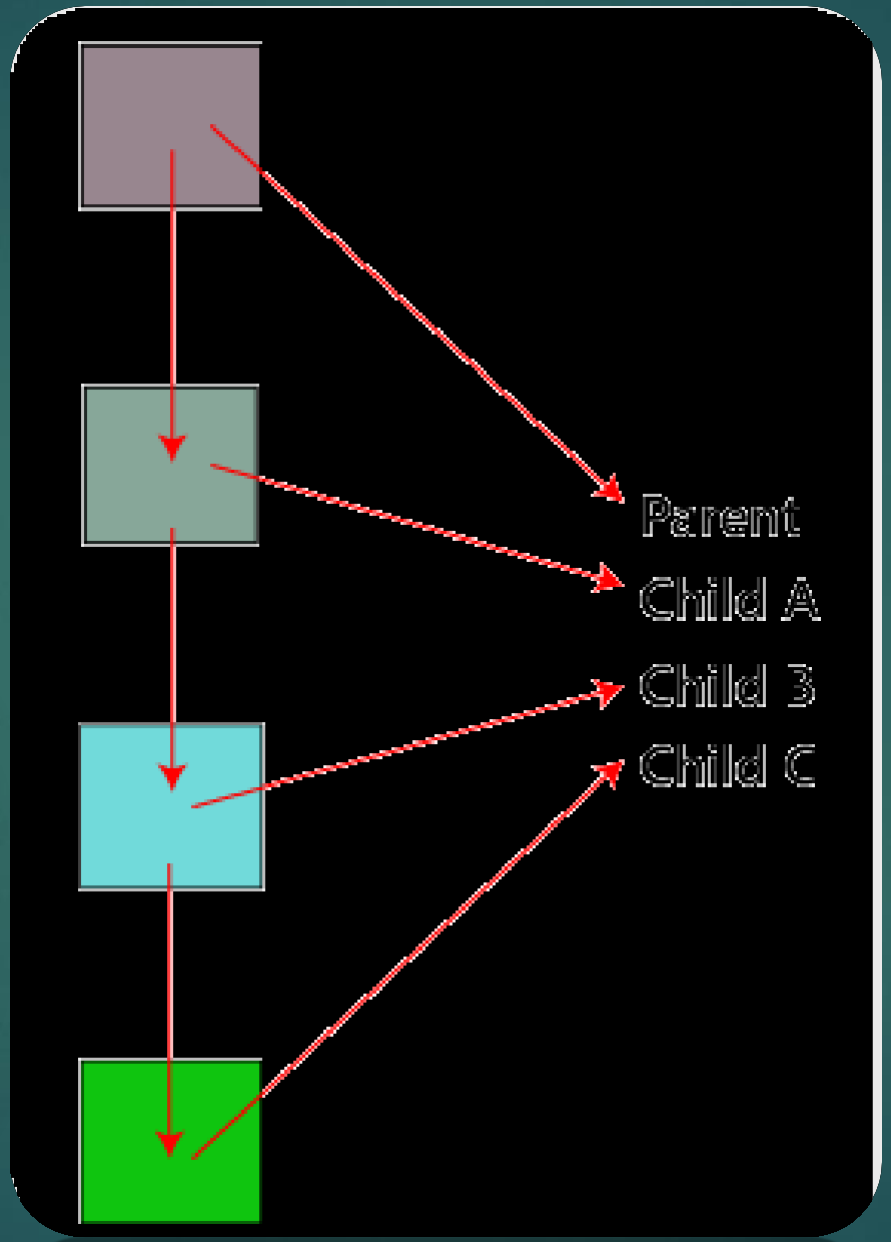
IncrementalApproach:

- modulesareaddedinascendingorderonebyoneoraccordingtoneed.
- Generally,twoor morethantwo modulesareaddedandtestedtodeterminethecorrectnessoffunctions.
- Theprocesscontinuesuntilthesuccessfultestingofallthemodules.

Top-DownApproach:

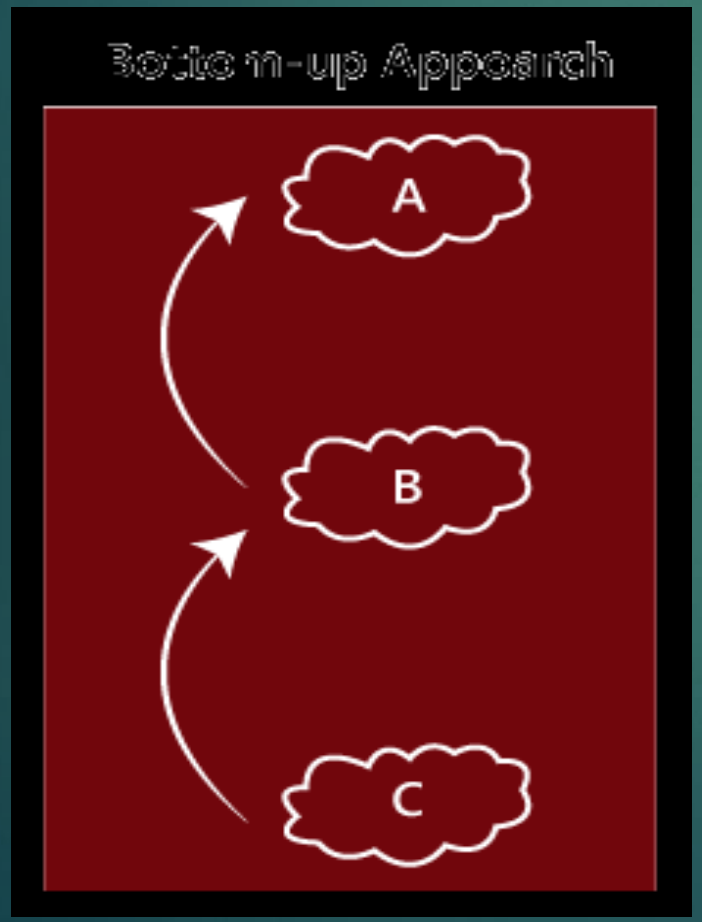
- dealswiththeprocessinwhichhigherlevelmodulesarete stedwithlowerlevelmodules.
- Majordesignflawscanbedetectedandfixedearly.
- themodulewareaddingisthe**childofpreviousone.**

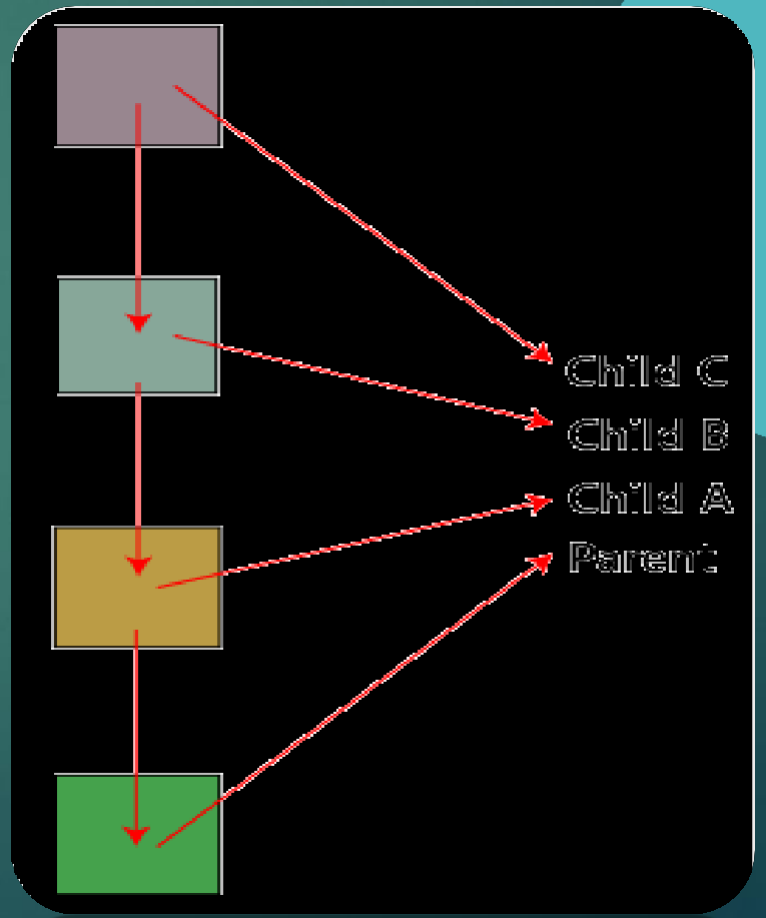



Parent
Child A
Child 3
Child C

Bottom-UpMethod:

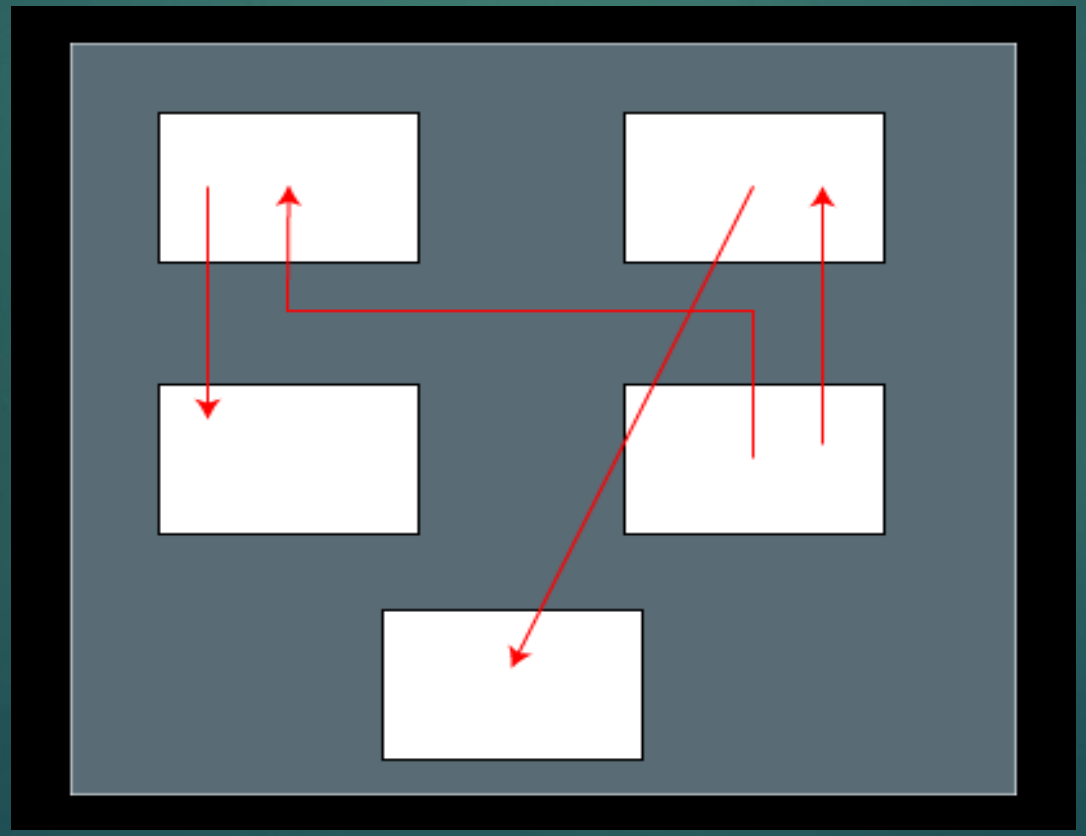
- theprocessinwhichlowerlevelmodulesaretested withhigherlevelmodules.
- Toplevelcriticalmodulesaretestedatlast.
- themoduleswearetheparentofthepreviousone.





Non-incrementalintegrationtesting:

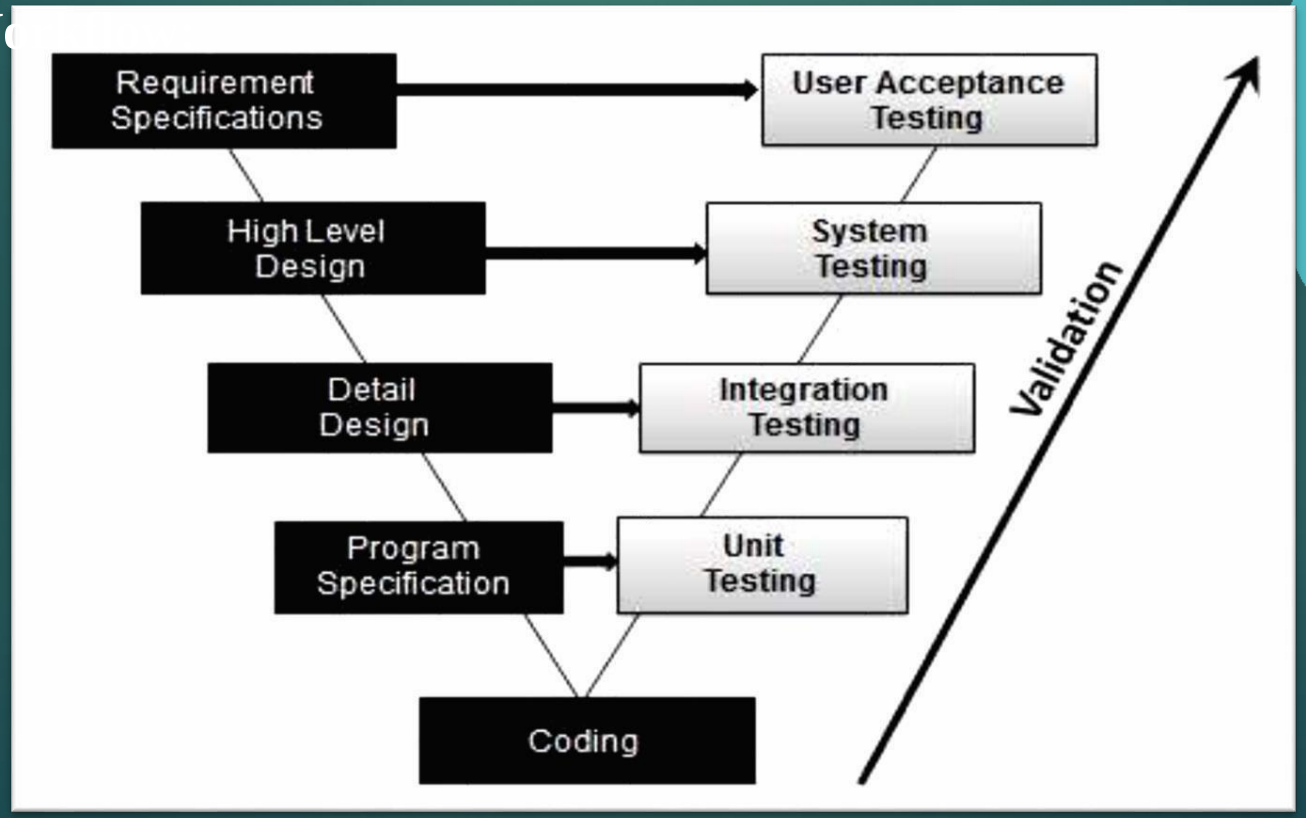
- whenthedataflowisverycomplexandwhenitisdifficulttofindwhoisaparentandwhoisachild.
- createthedatainanymodulebangonallotherexistingmodulesandcheckifthedataispresent.
- itisalsoknownastheBigbangmethod.



3.ValidationTesting:

- Testerperformsfunctionalandnon-functionaltesting.
- Validationtestingisalsoknownasdynamictesting.
-

ensuringthat**"wehavedevelopedtheproductright."Val idationTesting-W**



4. SystemTesting:

- includestestingofafullyintegratedsoftwaresystem.
- Tochecktheend-to-endflowofanapplicationorthesoftwareasauser.

SystemTestingincludesthefollowingsteps.

- Verification ofinputfunctionsof theapplication to testwhetheritis producingthe expectedoutputornot.
- Testingof integrated softwarebyincludingexternal peripherals to checktheinteractionofvariouscomponents witheachother.
- TestingofthewholesystemforEndtoEndtesting.
- Behaviortestingofthe applicationviaauser'sexperience

SoftwareTesting

Twomajorcategoriesofsoftwaretesting

- Blackboxtesting
- Whiteboxtesting

Blackboxtesting

- functionalitiesofsoftwareapplicationsaretested.
- mainlyfocusesoninputandoutputofsoftwareapplications.

TypesofBlackBoxTesting

- Functionaltesting
- Non-functionaltesting
- Regression

BlackBoxTestingTechniques

- EquivalenceClassPartitioning
- BoundaryValueAnalysis
- DecisionTableTesting

EquivalencePartitioningTesting:

- canbeappliedtoalllevelsofsoftwaretestinglikeunit,integration,system.
- dividestheinputtestdataoftheapplicationundertestintoeachpartitionatleastonceofequivalentdatafromwhichtestcasescanbederived.

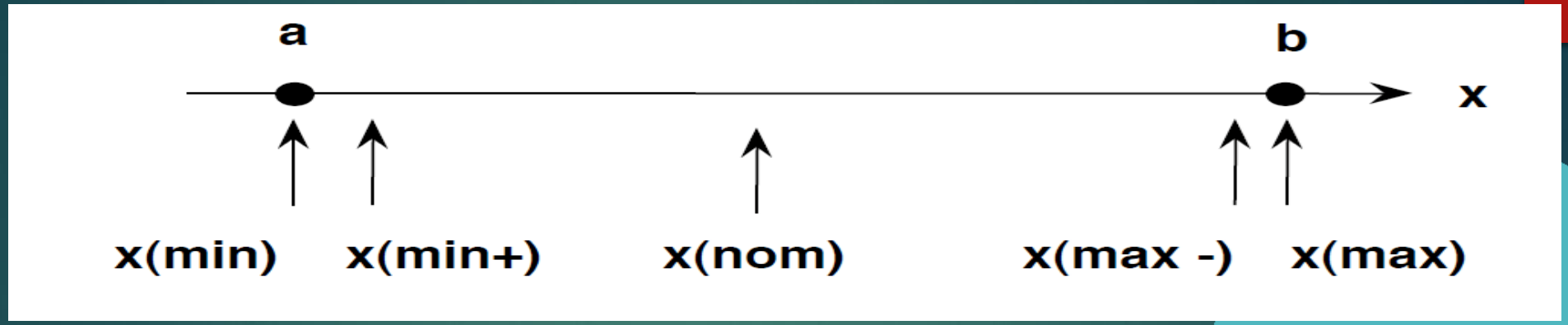
Example:

Assumethattheapplicationacceptsanintegerintherange100to999

- ValidEquivalenceClasspartition:100to999inclusive.
- Non-validEquivalenceClasspartitions:lessthan100,morethan999,decimalnumbersandalphabets/non-numericcharacters.

BoundaryValueAnalysis:

- testingbetweenextremeendsorboundariesbetweenpartitionsoftheinputvalues.
- theseextremeendslikeStart-End,Lower-Upper,Maximum-Minimum,JustInside-JustOutsidevaluesarecalledboundaryvaluesandthetestingiscalled"boundarytesting".



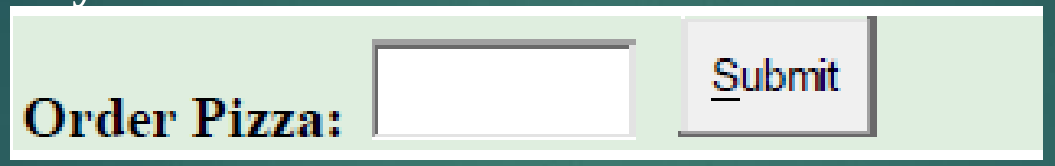
Example:

InputBoxshouldacceptttheNumber1to10Test

ScenarioDescription	**ExpectedOutcome**
BoundaryValue=0	SystemshouldNOTaccept
BoundaryValue=1	Systemshouldaccept
BoundaryValue=2	Systemshouldaccept
BoundaryValue=9	Systemshouldaccept
BoundaryValue=10	Systemshouldaccept
BoundaryValue=11	SystemshouldNOTaccept

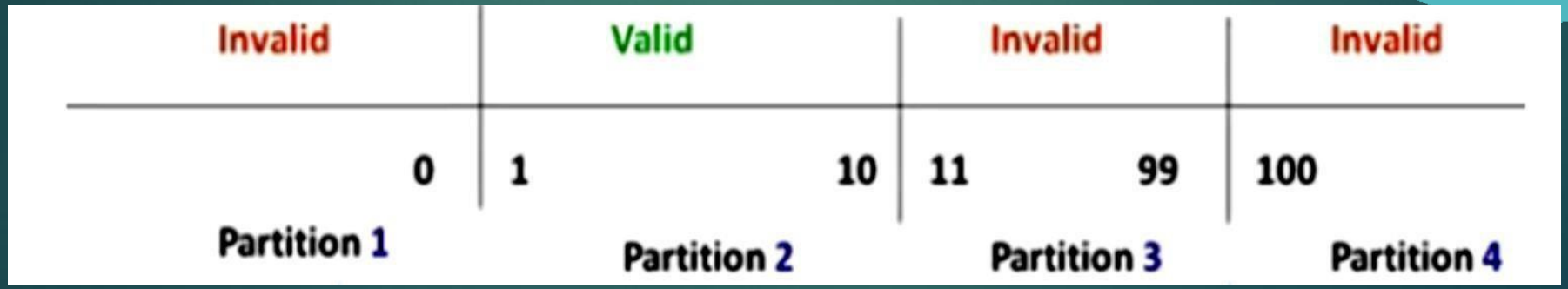
EquivalenceandBoundaryValue:

- Pizzavalues1to10isconsideredvalid.Asuccessmessageisshown.
- Whilevalue11to99areconsideredinvalidfororderandanerrormessage willappear,**"Only10Pizzacanbeordered"**

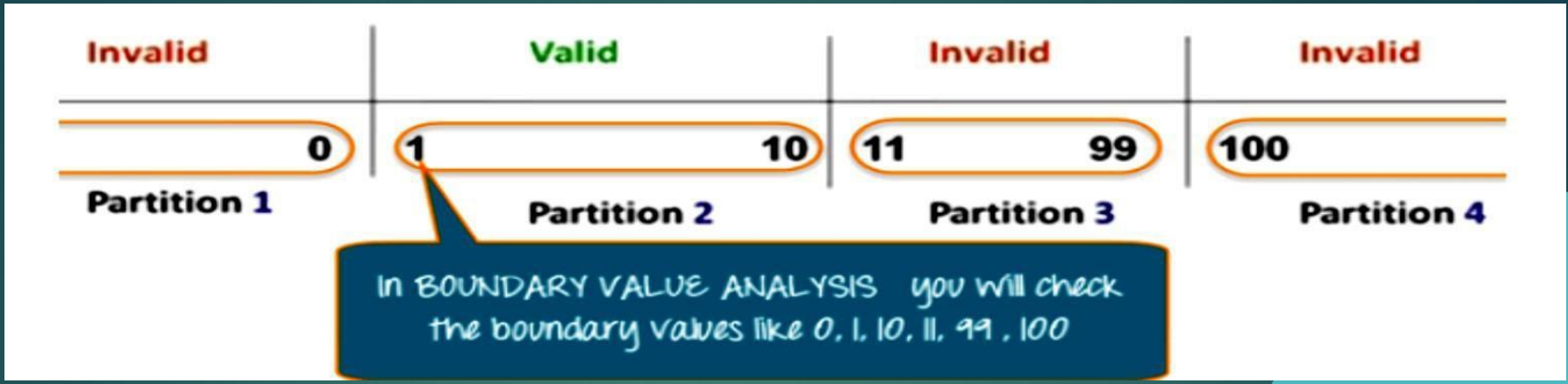


Hereisthetestcondition

1. AnyNumbergreaterthan10enteredintheOrderPizzafield(letsay11)isconsideredinvalid.
2. AnyNumberlessthan1thatis0orbelow,thenitisconsideredinvalid.
3. Numbers1to10areconsideredvalid
4. Any3DigitNumbersay-100isinvalid.



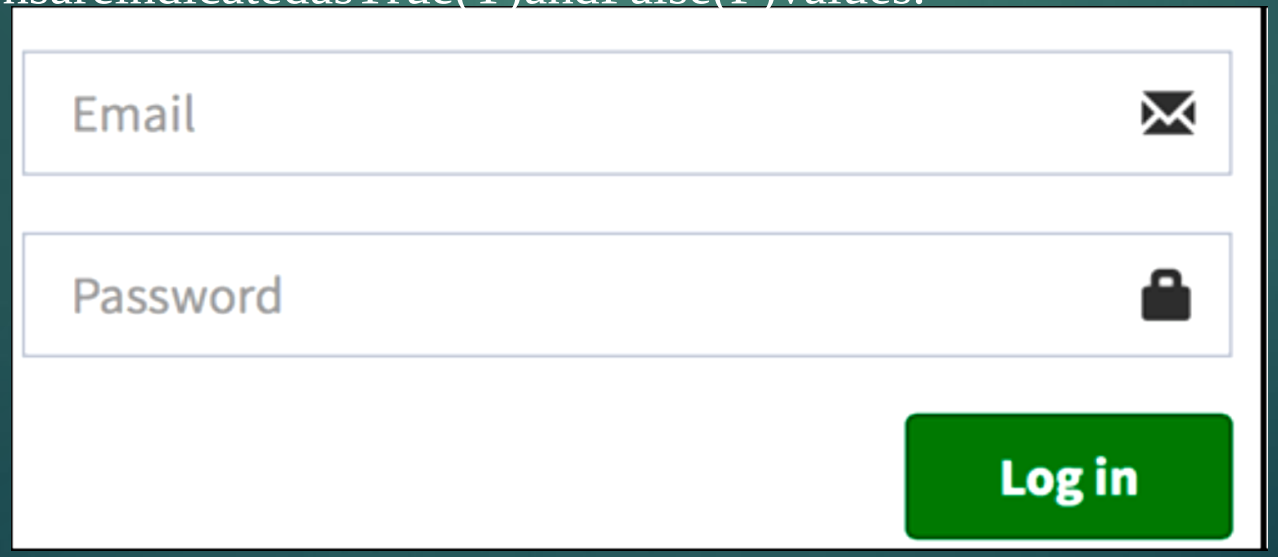
EquivalenceClassPartitioning



Boundaryvalueanalysis

DecisionTable:

- atabularrepresentationofinputsversusrules/cases/testconditions.
- TheconditionsareindicatedasTrue(T)andFalse(F)values.



Conditions	Rule1	Rule2	Rule3	Rule4
Username(T/F)	T	F	T	F
Password(T/F)	F	F	T	T
Output (E/H)	E	H	E	E

- Case 1–Usernameandpassword bothwerewrong.Theuseris shown anerror message.
- Case 2– Usernamewascorrect,butthepassword waswrong.Theuserisshown anerrormessage.
- Case 3–Usernamewaswrong, butthepassword wascorrect.Theuserisshown anerrormessage.
- Case 4– Usernameandpassword both werecorrect,andthe

NRCM
your tools to success

WhiteBoxTesting:

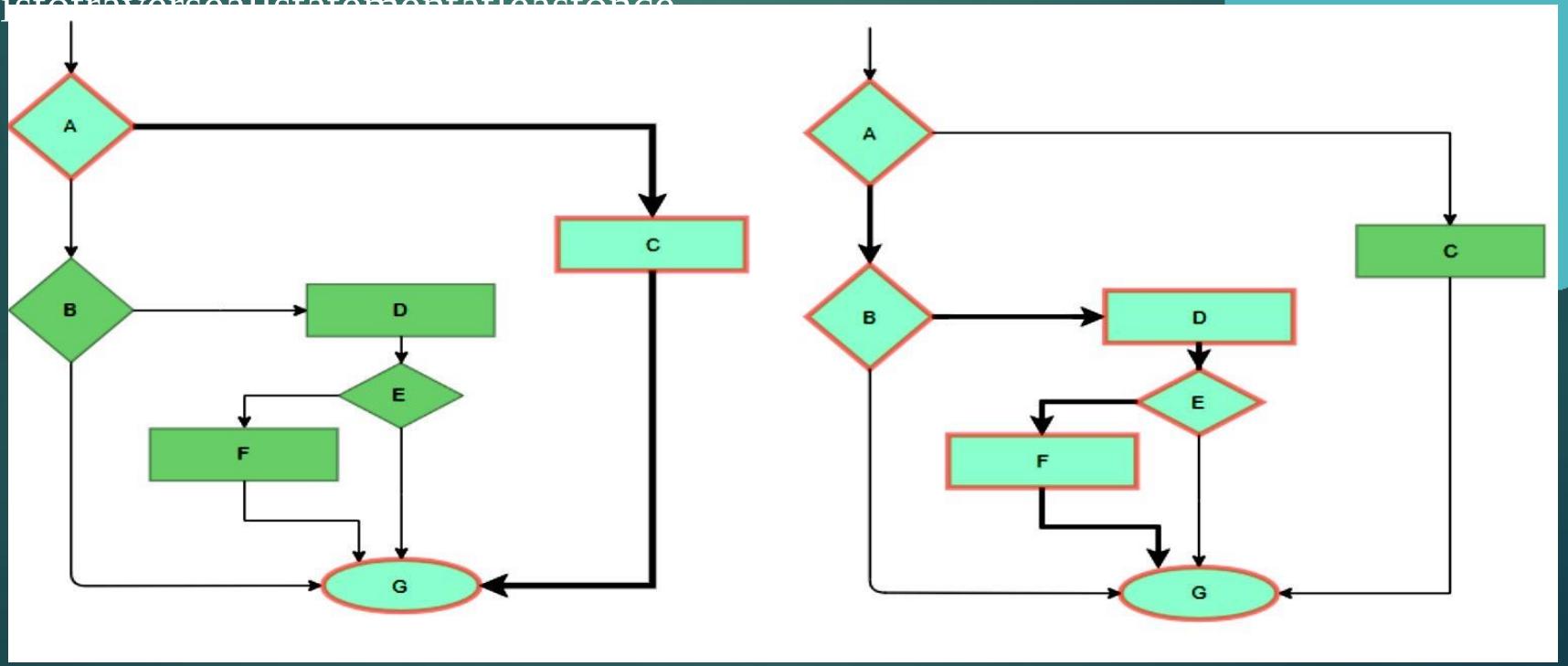
- examinestheprogramstructureandderivestestdatafromtheprogramlogic/code.

WhiteBoxTestingTechniques:

- StatementCoverage
- BranchCoverage
- PathCoverage

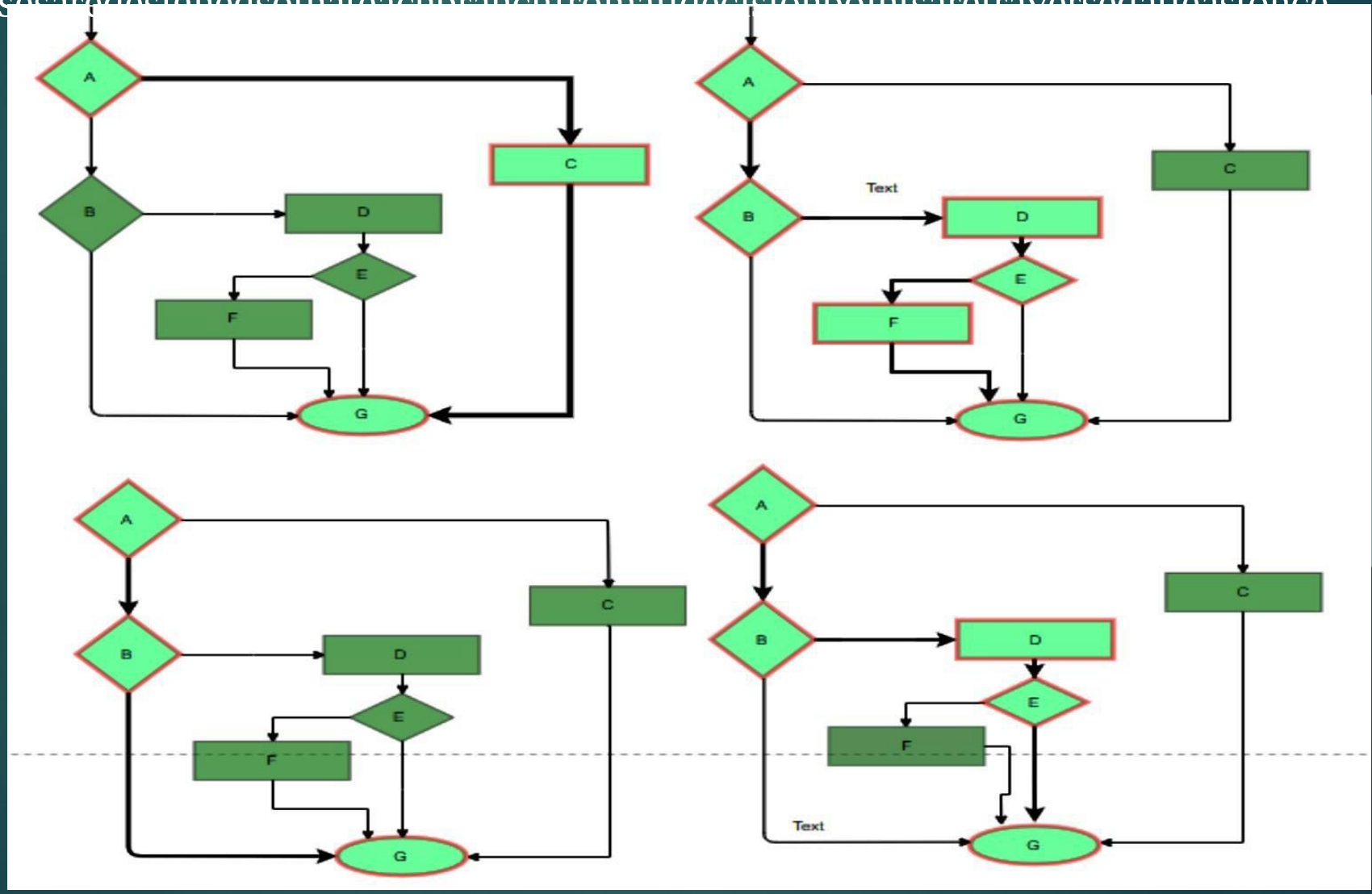
Statementcoverage:

- theaimi



BranchCoverage:

- testcasesaredesignedsothateachbranchfromalldecisionpointsaretraversedatleastonce



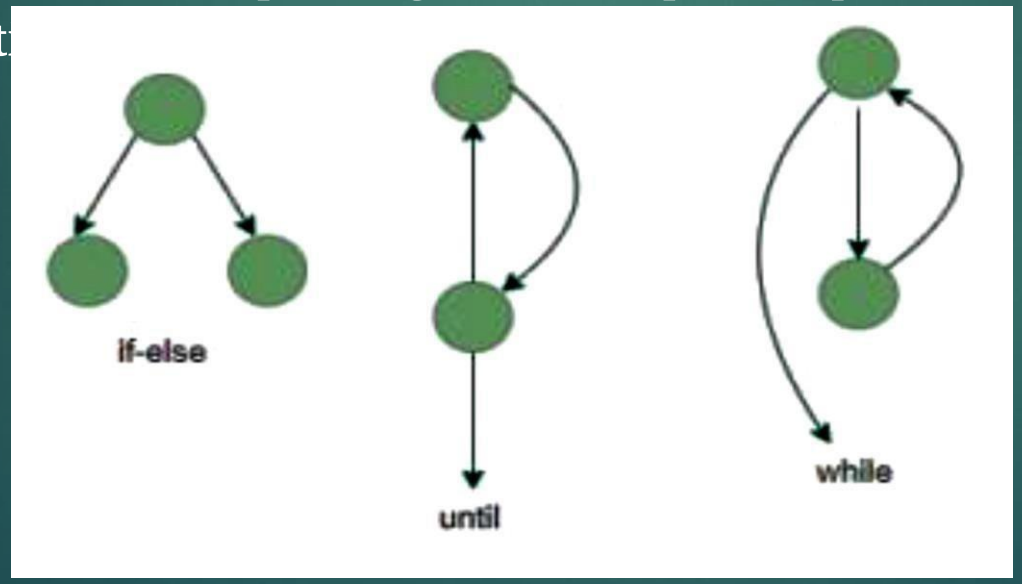
BasisPathTesting:

- controlflowgraphsaremadefromcodeorflowchartandthenCyclomaticcomplexityiscalculatedwhichdefinesthenumberofindependentpaths.

Steps:

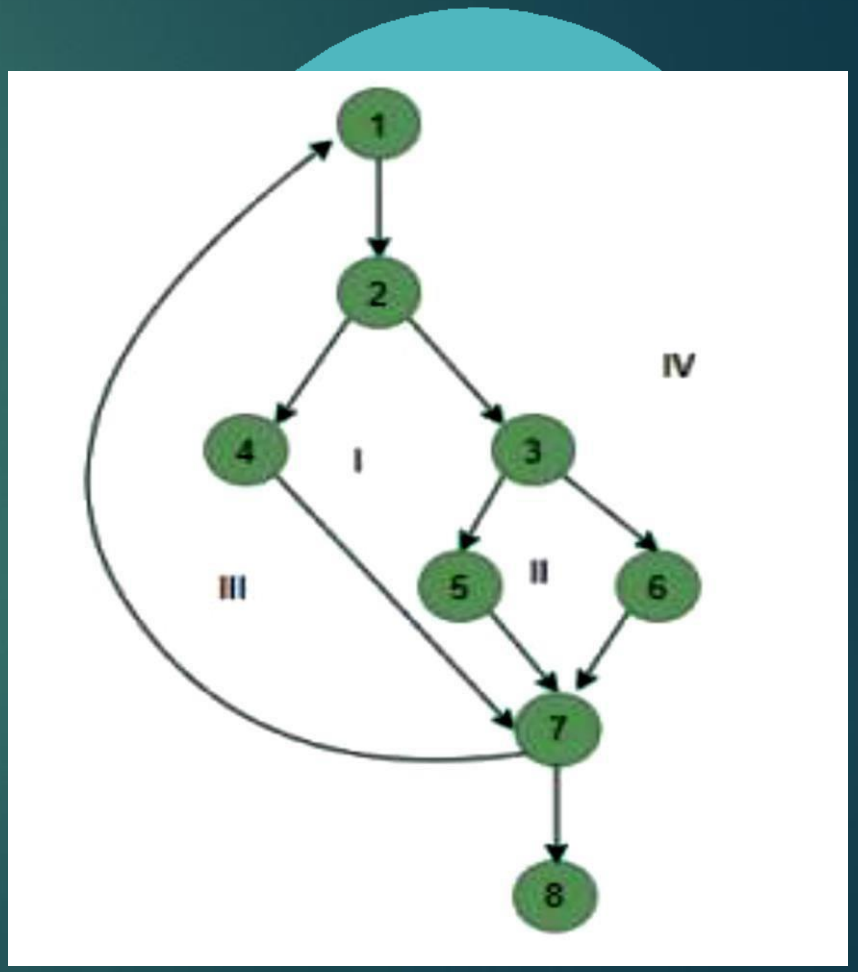
1. Makethecorrespondingcontrolflowgraph
2. Calculatethecyclomaticcomplexity
3. Findtheindependentpaths
4. Designtestcasescorrespondingtoeachindependentpath

Flowgraphnotat



CyclomaticComplexity:

- measurethelogicalcomplexityofthesoftwareand isused todefinethenumberofindependentpaths.
- ForagraphG,V(G)isitscyclomaticcomplexity.
- V(G)=P+1,wherePisthenumberofpredicate nodes inthe flowgraph
- V(G)=E–N+2, whereEis thenumberof edges NisthetotalnumberofnodesV(G)=4 , Noof independentpaths=4



LoopTesting:

- Simpleloops
- Nestedloop

SoftwareQuality:

- Softwarequalityproductisdefinedintermofitsfitness ofpurpose.
- aqualityproductdoes preciselywhat theuserswantitto do.

Severalquality methods:

- Portability
- Usability
- Reusability
- Correctness
- Maintainability

Variousmetricsformulatedforproductsinthedevelopmentprocessarelistedbelow.

- Metricsforanalysismodel
- Metricsfordesignmodel
- Metricsforsourcecode
- Metricsfortesting
- Metricsformaintenance

MetricsfortheAnalysis Model:

- usedtoexaminetheanalysismodelwiththeobjectiveofpredictingthesizeoftheresultantsystem.
- Functionpoint andlinesofcodearethecommonly usedmethodsforsizeestimation.

1. FunctionPoint(FP)Metric:

- measurethefunctionalitydelivered bythesystem,estimatetheeffort,predict thenumberoferrors,andestimatethenumberof componentsinthesystem.
- Functionpointisderivedbyusinga relationshipbetweenthecomplexityofsoftwareandtheinformationdomainvalue.

2. Lines ofCode(LOC):

- mostwidelyusedmethodsforsizeestimation.
- definedasthenumberofdeliveredlinesofcode,excludingcomments and blanklines.
- highlydependentontheprogramminglanguageusedascodewritingvariesfromoneprogramminglanguagetoanother.