SAAD FINAL :)

LECTURE 1 SDLC

SDLC: Traditional methodology for developing, maintaining, and replacing information systems / set of activities conducted and planned for each development project.

Outcome: Software / Documentation about the system and how it was developed/ Training for users

1-planning phase >> needs

Project manager :creates a project work plan, staffs the project, techniques.

Feasibility Study: preliminary investigation that helps the management to take decision**/ main objective** of a feasibility study is to **acquire problem outline and scope** (not solving the problem).

formal system proposal act as decision document which includes the nature and scope of the system.

- **Economic Feasibility**: effectiveness of candidate system by using cost/benefit analysis method.(business value)
- **Technical Feasibility** :analyses and determines whether the solution can be supported by existing technology or not.(can we build it?)
- **Operational Feasibility** ::analyses whether the users will be affected, and they accept the modified (will it be used?)
- **Behavioural Feasibility:** It evaluates and estimates the user attitude or behaviour towards the development of new system.
- **Schedule Feasibility:** ensures that the project should be completed within given time constraint or schedule.

2- Analysis phase >> Requirements

Requirements Determination: Careful study of organization current procedures and the information systems used to perform organizational tasks.

Requirements Structuring: Relationships between requirements and eliminating redundancy

3-Design phase >> Convert / How will the system operate in terms of hw, sw

Logical design: functional features described independently of computer platform.

Physical design: logical specifications transformed to technology-specific details.

5-Implementation >> Code, validate, install, and support

6-Maintenance>>Systematically repair and improve information system /not a separate phase but a repetition

Business analysis :is the practice of enabling change in an enterprise by **defining needs** and **recommending solutions** that deliver value to stakeholders.

Business analysts: are responsible for discovering, synthesizing, and analyzing information ,eliciting the actual needs of stakeholders

The systems analyst :works closely with all project team members so that the team develops the right system in an effective way /understand how to apply technology/

skills: technical, business, analytical, interpersonal, management, and ethical.

The Business Analysis Core Concept Model:

- **Need**: A problem or opportunity to be addressed.
- **Change**: The act of transformation in response to a need.
- **Solution**: A specific way of satisfying one or more needs in a context. / satisfies a need by resolving a problem
- **Stakeholder**: A group or individual with a relationship with business analyst is likely to interact with directly or indirectly**./ Examples** >> customer / end-user/project manager /supplier/sponsor/tester
- **Value**: worth, importance, or usefulness (tangible =measurable =monetary component or intangible=measured indirectly like >> motivational component>>company's reputation / employee morale)
- **Context** : The circumstances that influence, are influenced by, and provide understanding of the change. **Examples** >> behaviors, beliefs, competitors, culture, demographics, goals, governments, weather

CASE tools: are used to support a wide variety of SDLC activities.

- **Diagramming tools :** enable system process, data, and control structures to be represented graphically.
- **Computer display and report generators** help prototype how systems "look and feel."
- **Analysis tools** automatically check for incomplete, inconsistent, or incorrect specifications in diagrams, forms, and reports.
- **A central repository** enables the integrated storage of specifications, diagrams, reports, and project management information.
- **Documentation generators** produce technical and user documentation in standard formats.
- **Code generators** enable the automatic generation of program

Outsourcing: Turning over responsibility of some or all of an organization's information systems applications and operations to an outside firm**./ reasons** >> cost-effective /free up internal resources/reduce time to market

LECTURE 2
ANALYSIS PHASE

Requirement : usable representation of a need.

- **Feature: that a future solution has to enable (Cloud Access)**
- **A Function :that a future solution has to Execute (Calculate Savings)**
- **A Fact :that a future solution has to enforce (regulations)**
- **A Quality :that a future solution has to exhibit (Access file in 1 second)**

1-Business requirements: statements of goals, objectives, and outcomes that describe why a change has been initiated/ executive levels define them / purpose is to ***prioritize*** and ***resource*** projects / hint : starts with to ……….

2-Stakeholder requirements: describe the needs and wants of stakeholders that must be met in order to achieve the business requirements. They may serve as a bridge between business and solution requirements./ hint : starts with as....

3-Solution requirements: describe the capabilities(functions) and qualities of a solution that meets the stakeholder requirements. 2 types:

- **Functional requirements**: describe the capabilities/features that a solution must have in terms of the behavior and information / ex: prototypes / list of functions / activity diagrams
- **Non-functional requirements**: describe conditions/performance constraints under which a solution must remain effective or qualities/ ex: performance / scalability / usability / security / portability / ay haga a5erha ity

4- Transition requirements: describe the capabilities that the solution must have and the conditions the solution must meet to facilitate transition from the current state to the future state, address topics such as ***data conversion***, ***training***, and ***business continuity***. differentiated from other req. by their ***temporary nature***. /***Not needed*** when solution is in production. / hint : old to new

Requirement Determination Tasks:

1- Prepare for the Elicitation

Mind mapping: used to articulate and capture thoughts, ideas, and information / **images**, **words**, **color**, and connected **relationships** to apply **structure** and **logic** to thoughts, ideas, and information

2- Conduct Elicitation (traditional / modern methods)

3-Approve Elicitation

4- Communicate Information : The purpose of communicating analysis Information is to ensure stakeholders have a shared understanding of analysis information./Communication of analysis information is ***bi-directional*** and ***iterative.***

5- Manage Stakeholder Collaboration :The purpose of managing stakeholder collaboration is to encourage stakeholders to work towards a common goal./***ongoing activity***.

- Stakeholder Lists
- Stakeholder Maps:

  Stakeholder Matrix: maps the level of stakeholder influence against the level of stakeholder interest.
- **Persona:** is defined as a fictional character that exemplifies the way a typical user interacts with a product.

SRS: determining the information requirements of an organization used by analysts is to prepare a precise SRS understood by user / ideal SRS : specify operational, tactical, and strategic information requirements/ be complete, Unambiguous, and Jargon-free/ solve possible disputes between users and analyst/ use graphical aids which simplify understanding and design.

LECTURE 3 ANALYSIS PHASE

2- Conduct Elicitation in Requirements Determination (Analysis)

purpose : **draw out**, **explore**, and **identify** information relevant to the change.

Traditional Methods:

•**Interviewing individuals :** Dialogue with stakeholder to obtain their requirements/ Advantage: **Easier** to **schedule** than group interviews **.**Disadvantages **Contradictions** and **inconsistencies** between interviewees**/ Follow-up** discussions are time consuming

Interview Guide is a document for developing, planning and conducting an interview.

Open-ended Questions:*cannot anticipate* all possible responses or for *which you do not know the precise question* to ask Advantage: *Previously unknown* information can *surface*. You can then *continue exploring* along unexpected lines of inquiry to reveal even more new information. Often *put the interviewees at ease* because they are able to respond in their *own words* using their *own structure.*Give interviewees more of a *sense of involvement* and *control* in the interview. Disadvantage: **The *length of time*** it can take for the questions to be answered. *Difficult to summarize*.

Closed-ended Questions: *range of answers . Advantages:* work well when the *major answers* to questions are *well known*. *do not* necessarily *require* a *large time* Can be an *easy way to begin an interview* and to determine which line of open-ended questions to pursue. you can include an "*other*" option to encourage the interviewee to add unanticipated responses *disadvantage* useful information that *does not quite fit into the defined answers may be overlooked* as the respondent tries to make a choice instead of providing his or her best answer.

- **Group interviews:** Advantage: ***effective use of time*** / Can hear ***agreements*** and ***disagreements*** at once/Opportunity for ***synergies.*** Disadvantage: ***More difficult to schedule*** than individual interviews
- **Nominal Group Technique (NGT):** A facilitated process that supports idea generation by groups. / end result would be a list of either ***problems or features*** that group members themselves had ***generated*** and ***prioritized***.
- **Survey/Questionnaire:** Is used to elicit business analysis information including information about customers, products, work practices, and attitudes—from a group of people in a structured way and in a relatively short period of time.
- **Directly observing Users: p**rovide ***more accurate information*** than ***self-reporting*** /c***ause*** people to ***change*** their ***normal operating behavior***/ not continuous (comment) : u receive only a snapshot image of person / very time consuming
- **Analyzing procedures and other documents:**
  1) **Written work procedures**: is a business document that formally describes work processes, provides useful information regarding system functionality and logic / how a job is performed / data and information used and created
  2) **Business form** :Explicitly indicate data flow in or out of a system/ document that contains useful information regarding data organizations and possible screen layouts.
  3) **Report:** Enables the analyst to work backwards from the report to the data that generated it
  4) **Description of current information system**

Modern Methods:

1-Joint Application Design (JAD): Intensive group-oriented requirements determination technique / Team members meet in isolation for an extended period of time.

•**Session Leader**: facilitates group process/ End Result:**Documentation** detailing **existing system/ Features** of **proposed system**

Scribe: record session activities

2-Case tools

3-Prototyping : Is used to elicit and validate stakeholder needs through an iterative process that creates a model or design of requirements / converts requirements to working version of system.

LECTURE 4 SCRUM

PDCA: plan do check act
Agile: iterative development vs **Waterfall**: linear development
Scrum: is a framework that helps teams work together / set of meetings, tools, and roles that work in concert to help teams structure and manage their work./ subset of Agile

Scrum Pillars:

- Transparency : current status is visible to team and stalkholders
- Inspection: bring all to closer solution
- Adaptation: business can be unpredictable and change / must be prepared to make trade-offs

Scrum Team (5 to 11 members):
Scrum Master: facilitate the process / resolve impediments(عائق) / focus the team / serves the team
Product Owner: represent the customer / prioritize sprints / accepts work / approves releases / participates in all meetings
Scrum Backlog: sprint plan that team constructs/ finalized during sprint planning / pts of user stories = team velocity
Sprint: the product's development iteration

Scrum Events:-

- **Sprint planning meeting** : team plans what is hoped to be accomplished and completed during the upcoming sprint / 1-2 hrs / takes place on day 1
- **Daily Stand-up. (DSU):** team's mini status meeting / <=15 mins / occurs everyday / each team mem. Answers 3 questions
- **Sprint Review** : team runs and demonstrates current product to themselves / product packaged and running / formal time to check the product and validate / 1 to 2 hrs ./ last day or 2 days of sprint
- **Sprint Retrospective.:** (START STOP CONTINUE) doing/ last day or 2 days of sprint

Definition of Done (DoD): checklist of items that the team , product owner and stalkholder agree must be completed before the User stories can be consider done / similar to exit criteria (testing) /examples:acceptance criteria satisfied / unit test run / compliance tasks completed
Tools: jira/click-up / notion

Product Management: (Business /Customer /Technology)
Building the right product(Product owner /research) VS Building the product right (architecture /technology)

Product Roles

- **Project Manager**: Creates a project plan, tracks progress, ensures that the project is completed on time and within budget, and manages the resources involved in the project, such as people, time, and money. / owns the product
- **Product Owner:** Is responsible for the product backlog, prioritizes the backlog items, ensures that the most important features are delivered first, and works with the product manager to gather and prioritize customer feedback / owns set of feature requirements
- **Product Manager**: Defines the product vision, sets the product goals, creates the product roadmap, gathers and analyzes customer feedback, and works with engineers, designers, and other stakeholders to bring the product to market. /. Owns project from beginning to end

Competitive Analysis

Benefits: Trendspotting / updating table stakes / right price? / keeping up with joneses
Types:

- **Customer analysis**: understanding your competitors' target customers, their needs, and their pain points.
- **Product analysis**: evaluating your competitors' products or services, their features, pricing, and marketing.
- **Marketing analysis**: assessing your competitors' marketing strategies, such as their advertising, public relations, and sales efforts.
- **SWOT analysis:** identifying the(strengths>>where a company excels and has a competitive advantage over its peers., weaknesses>>where a business is at a competitive disadvantage relative to its peers., opportunities>>external factors that represent potential growth or improvement areas for a business., and threats >>external forces that represent risks)facing your competitors.

Feature Prioritization:

LECTURE 5
PRODUCT MANAGEMENT

- **Kano Model**.:potential features through the lens of the delight a feature provides to customers versus the potential investment you make to improve the feature/function

- **Value versus Complexity Quadrant**: evaluate every opportunity based on its business value and its relative complexity to implement.

- **Weighted Scoring**: using a scoring method to rank your major features,

- **Rice Scoring Model:** a decision-making tool used by product managers to prioritize features.
  RICE stands for Reach, Impact, Confidence, and Effort , RICE Score = (Reach * Impact * Confidence) / Effort
  Reach: This is an estimation of the number of people or users who will be affected by the feature in a time period.
  Impact: This is an estimate of how much the project will contribute to the user's satisfaction, minimal (0.25), low(0.5), medium (1), high (2), or massive (3).
  Confidence: Confidence is a measure of how certain you are about the estimates for Reach, Impact, and Effort, and is usually expressed as a percentage. For example, you could use 100% to indicate high, 80% for medium, and 50% for low confidence.
  Effort : This is an estimation of the total amount of work required to complete the project

- **Eisenhower Matrix**: help you prioritize a list of tasks or agenda items by first categorizing those items according to their urgency and importance / four-box square with an x-axis labelled Urgent and Not Urgent, and the y-axis labelled Important and Not Important.
- **Story Mapping**: Arranging user stories to create a more holistic view of how they fit into the overall user experience./ done on a wall (or floor) using sticky notes
- **MoSCoW Model** : popular prioritization technique for managing requirements

LECTURE 6 DESIGN PHASE

System modelling: process of developing abstract models of a system, with each model presenting a different perspective of that system.

•**Benefits :** Help project teams **communication / Explore** potential designs / **Validate** the architectural design of the software / Be **independent** of particular programming

Unified Modelling Language (UML) :standard language for specifying, modeling, visualizing, constructing, and documenting the software systems.

System perspectives

- **external** perspective>>model the context or environment of the system **(context models)**
- **interaction** perspective>>model the interactions between a system and its environment, or between the components of a system **(use case and sequence diagrams)**
- **structural** perspective>>model the organization of a system **(class diagram)**
- **behavioral** perspective>>model the dynamic behavior of the system and how it responds to events **(state diagram**)

UML diagram types

- **Activity** diagrams>>flow of activities / belongs to process perspective
- **Use case** diagrams, which show the interactions between a system and its environment.
- **Sequence** diagrams, which show interactions between actors and the system and between system components.
- **Class** diagrams, which show the object classes in the system and the associations between them.
- **State** diagrams, which show how the system reacts to internal and external events.

proposed system >>Incomplete and incorrect models are OK

existing system>> accurate representation(correct), May be incomplete.

detailed system description for system implementation>>have to be correct and complete.

Context models >> the operational context/what lies outside the system boundaries./show the other systems in the environment / belongs to external perspective

Interaction models(Use case diagrams and sequence) >>

Use cases:support requirements elicitation from stakeholders who interact directly with the system / discrete task or main functionality/ focus on automated processes/don't show order/ name should start with a verb.ex:Transfer Data

Use Case Diagram: give a simple overview of an interaction so have to provide more detail to understand what is involved details like: (a simple textual description ,a structured description in a table , a sequence diagram)

Use case actors: Users , Other systems ,Time (Time becomes an actor when the passing of a certain amount of time triggers some event in the system (out of control))

Relationships:

- **Association Relationship** >>between a use case and an actor only /use case must be initiated by an actor, with the exception of abstract use cases (in include and extend relationships)
- **Include Relationship** >>between one use case to use the functionality provided by another use case / 2 cases:If two or more use cases have a large piece of functionality that is identical ,A single use case has an unusually large amount of functionality/ one use case always uses the functionality provided by another use case
- **Extend Relationship** >> between use case and another use case the option to extend functionality provided by another use case / "extend" relationship is not used to represent options.
- **Generalization**>>between actors / The attributes and operations associated with higher-level classes are also associated with the lower-level classes/ Lower-level classes are sub classes, which inherit the attributes and operations from their super classes

Check List >> end-user using and providing system? / consider maintenance / at least 1 use-case / external-systems

Activity diagrams>> flow of functionality in a use case./may also be used to model the processing of data, where each activity represents one process step / define:

where the workflow starts >> filled circle
where it ends >> filled circle inside another circle.
what activities occur during the workflow >> Rectangles with round corners
what order the activities occur>> choice/decision(A diamond) / activity co-ordination(A solid bar)
Flow of work >> Arrows

Workflow Analysis>>This technique allows you to understand how a work process is completed when several people (and roles) are involved.

Abstract use cases>>not started directly by an actor / are the use cases that participate in an include or extend relationship

LECTURE 7 DESIGN PHASE

Specification Documents contains >>Overall system description/Interface requirements/System features/Nonfunctional requirements/Supporting diagrams and models
Requirements Management (RM) Tools :make it easier to keep documents up to date, add additional requirements and link related requirements.
structure chart : A high-level diagram created to illustrate the organization and interaction of the different pieces of code within the program/Helps analysts design programs/A hierarchical diagram that shows how an information system is organized/

It shows all components of code in a hierarchical format that implies:***Sequence*** (order of invoking components),***Selection*** (under what condition module is invoked)***,*** ***Iteration*** (how often component is repeated

•Composed of **Modules:** a self-contained component of a system that is defined by its function.

• **control module** : higher-level component that contains the logic for performing other modules, and the components that it calls and controls are considered **subordinate** modules.
Linking DFDs with structure chart >> Each process => one/ module Level => different level / Process on context model => top module
Structure Chart Symbols:
• **Data couple**: Diagrammatic representation of the **data exchanges** between two modules
•**Flag:** Diagrammatic representation of a **message** passed between two modules.

Program specifications Written documents that include explicit detailed instructions on how to program pieces of code/ describe what needs to be included in each program module
Four components are essential for program specification: Program information / Events/ Inputs and outputs / Pseudocode.
Measures of good design include

- **Cohesion:** how well the lines of code within each module relate to each other./ ***High Cohesion***
  1. ***Functional cohesion*** – all elements of the modules contribute to performing a single task.
  2. ***Temporal cohesion*** – functions are invoked at the same time.
  3. ***Coincidental cohesion*** – there is no apparent relationship among a module's functions.

Factoring is the process of separating out a function from one module into a module of its own/ make modules cohesive and create a better structure

- **Coupling :** measure of the independence of components./ between modules/ ***loosely coupled***.
  1. **Content Coupling**: When one component actually modifies another, then the modified component is completely dependent on modifying one. It is the worst coupling type.
  2. **Common Coupling:** When amount of coupling is reduced somewhat by organizing system design so that data are accessible from a common data store.
  3. **Control Coupling**: When one component passes parameters to control the activity of another component.
  4. **Stamp Coupling:** When data structures is used to pass information from one component to another.
  5. **Data Coupling**: When only data is passed then components are connected by this coupling.
- **appropriate levels of fan-in(** number of control modules that communicate with a subordinate ***high fan-in)*** **and fan-out(**number of subordinates associated with a single control/ low / limit a control module's subordinates to approximately **seven**)

UI DESIGN

- **adaptive design** >> ***several versions*** of *one* design and make each have *fixed dimensions*
- **responsive design** >> Work on **a *single, flexible* design** that would stretch or shrink to fit the screen / **GUI design approach used to create content that adjusts smoothly to various screen sizes**.

Responsive Design Core Principles
Fluid Grid System>> Elements occupy the same **percentage** of space however large or small the screen becomes. / use a CSS

•**Fluid Image Use >>** Unlike text, images aren't *naturally* fluid. apply a CSS command img {max-width: 100%;}

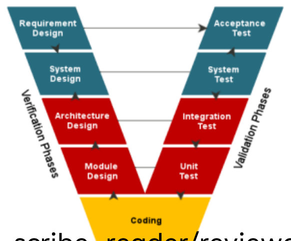
•**Media Queries >>** filters you use to detect the browsing device's dimensions and make your design appear appropriately.

LECTURE 8:

SOFTWARE TESTING(IMPLEMENTATION PHASE)

Testing goals:
1-the software **meets the customer requirements**
2-discover situations in which the behavior of the software is incorrect ex(system crashes ,incorrect computation and data corruption)

- **Verification: static process** of verifying documents, design and code to determine they satisfy the conditions imposed at the start of that phase.



Static Testing Types

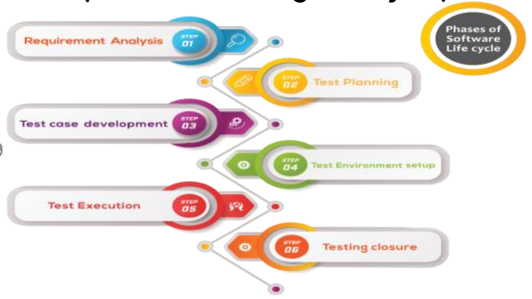
- **Peer Review.** peer to read, comment, and critique his work
- **Walkthrough**. others.
- **Inspection**. An author requests the services of a moderator, scribe, reader/reviewers in **a formal meeting.** The **moderator** books the room, sends out the material. The reviewers read the material before the meeting. During the meeting, the reader/reviewers take turns reading the work artifact out loud. **The scribe** takes notes of issues the reader/reviewers discovered.

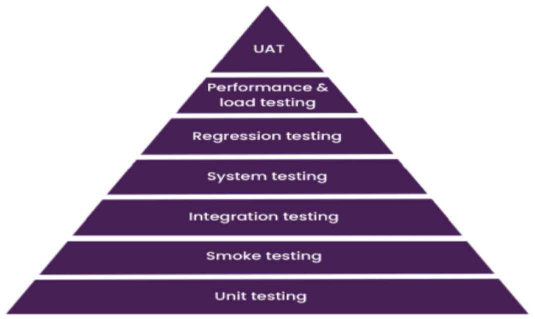
- **Validation: dynamic process** to ensure that the software satisfies the customer real requirements and expectations. Test data is entered, processed and the output is verified against what the user wants.

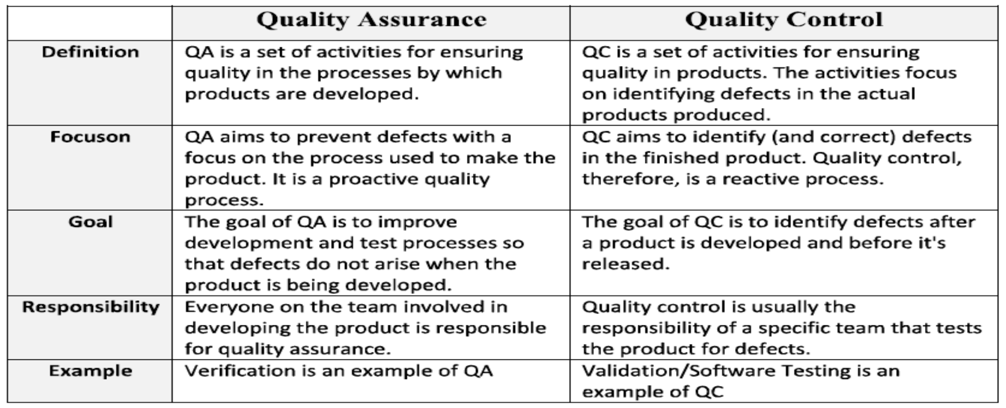
Dynamic Testing

what it is intended to do and to discover program defects / Can reveal the presence of errors NOT their absence>> Because testing all combinations of inputs and pre-conditions is not feasible except for trivial cases/ test software, you execute a program using artificial data (test data)/ program's non-functional attributes.

STLC (Software Testing Life Cycle)







	Quality Assurance	Quality Control
Definition	QA is a set of activities for ensuring quality in the processes by which products are developed.	QC is a set of activities for ensuring quality in products. The activities focus on identifying defects in the actual products produced.
Focuson	QA aims to prevent defects with a focus on the process used to make the product. It is a proactive quality process.	QC aims to identify (and correct) defects in the finished product. Quality control, therefore, is a reactive process.
Goal	The goal of QA is to improve development and test processes so that defects do not arise when the product is being developed.	The goal of QC is to identify defects after a product is developed and before it's released.
Responsibility	Everyone on the team involved in developing the product is responsible for quality assurance.	Quality control is usually the responsibility of a specific team that tests the product for defects.
Example	Verification is an example of QA	Validation/Software Testing is an example of QC

Functional Testing

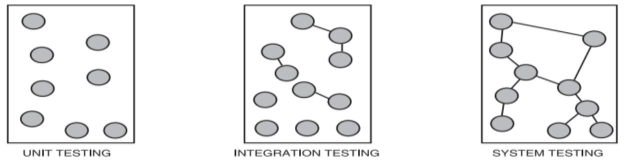


FIGURE 7.1 Levels of Testing.

Component/Unit Testing: testing individual components in isolation/ by the team developing/ defect testing process./components: Individual functions or methods within an object , Object classes with several attributes and methods/ Advantages of Unit Testing: The earlier a problem is identified, the fewer compound errors occur ,Fixing problems early is usually cheaper Easier debugging processes. .Developers can reuse code and migrate it to new projects.

techniques: white n black box

1- Black-Box Testing Technique :without having any knowledge of the interior workings (know the requirements only not the code) / Advantages **Test** case selection is **done before the implementation** of a program. Help in getting the design and coding correct with respect to the specification.

2-White-Box /glass testing / open-box Testing Technique: detailed investigation of internal logic and structure of the code/ May be performed at **all test levels**. / Advantages: Tests the internal details of the code,Checks all paths that a program can execute. Limitation: Wait until after designing and coding the program under test in order to select test cases.

Smoke Testing: non-exhaustive set of tests / most critical/important functions work./ executed "before" any detailed functional or regression tests / **The result of this testing** is used to decide if a build is stable enough **to proceed** with further testing./ Conducted by developers or testers.

Integration testing : individual units are combined and tested as a group./ purpose : expose faults in the interaction between integrated units. interfacing errors not detected in unit testing may appear.Timing problems (in real-time systems) are not detectable by unit testing.

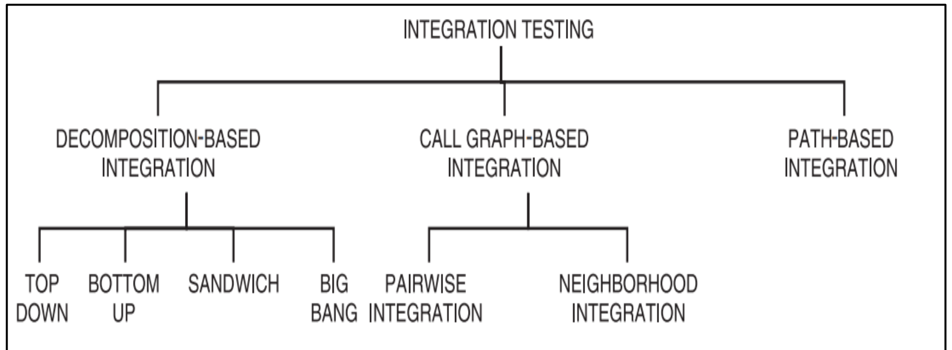


FIGURE 7.4

Decomposition-based Integration

Top-down Integration Approach : begins with main program, i.e., the root of the tree. Any lower-level unit that is called by the main program appears as a "stub." A stub is a piece of throw-away code that emulates a called unit

No. of Required **Stubs** = (No. of Nodes – 1)

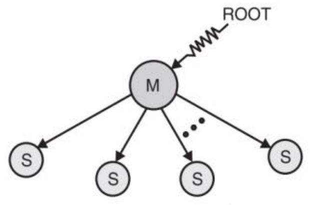


FIGURE 7.5 Stubs.

Bottom-up Integration Approach :we start with the leaves of the decomposition tree and test them with specially coded drivers.

No. of **drivers** required = (No. of nodes – No. of leaf nodes)

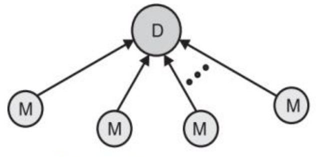


FIGURE 7.6 Drivers.

Sandwich Integration Approach: combination of top-down and bottom-up integration.

Big-bang IntegrationL: this approach waits until all the components arrive, and **one round** of integration testing is done / reduces testing effort and removes duplication in testing /is ideal for a product where the interfaces are stable with fewer number of defects.

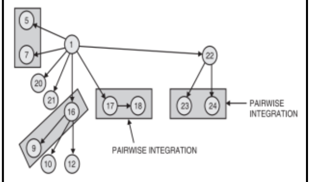


FIGURE 7.7 Pairwise Integration.

Call Graph-based Integration

Pairwise Integration : use the actual code / this sounds like big-bang integration, but we restrict a session to only **a pair of units in** the call graph.

Neighbourhood Integration: set of nodes that are one edg away from the given node.

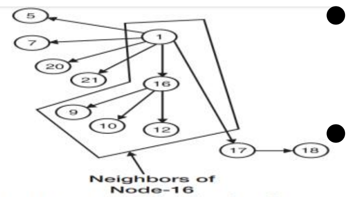
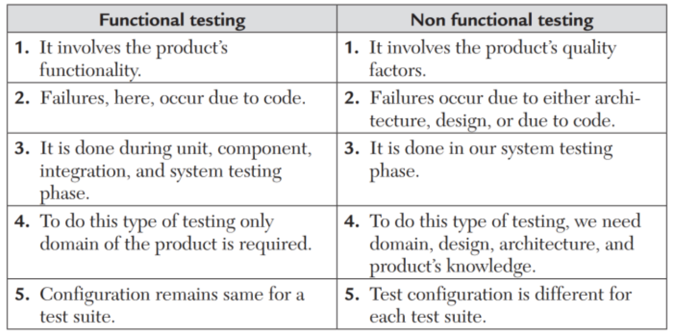


FIGURE 7.8 Neighborhood Integration.

The rest of the functional mn el slide el fatet :

- **System testing:** complete, integrated system to evaluate compliance with specified requirements. Tests are made on characteristics that are only present when the entire system is run.
- The purpose : evaluate the end-to-end system specifications. System testing is the only testing phase that **tests both functional and non-functional aspects Customer scenarios and usage patterns serve as the basis** for system testing.



Functional testing	Non functional testing
1. It involves the product's functionality.	**1.** It involves the product's quality factors.
2. Failures, here, occur due to code.	**2.** Failures occur due to either architecture, design, or due to code.
3. It is done during unit, component, integration, and system testing phase.	**3.** It is done in our system testing phase.
4. To do this type of testing only domain of the product is required.	**4.** To do this type of testing, we need domain, design, architecture, and product's knowledge.
5. Configuration remains same for a test suite.	**5.** Test configuration is different for each test suite.

- **Regression testing :** re-run every **change /** check that changes have not 'broken' previously working code. / manual testing process, regression testing is expensive but, with automated testing, it is simple
- **User acceptance testing (UAT):** users provide input and advice on system testing./ user's working environment have a major effect on the reliability, / quality team has a meeting with the client/ client will then give feedback./ very crucial activity done for all projects in all IT companies and the quality team is responsible for managing it. / **no separate user acceptance testing process in agile methods**. Comment. Because in Agile we make every sprint (every 2 weeks) sprint review meeting and customer checks the product and validate a progress.

Entry Criteria: **minimum set** of **conditions** that should be **met** (**prerequisite items)** before **starting** the software testing ex: Verify if the **testing environment** is **available/ testing tools/ testable code** is available./**test data** is available and validated for correctness of Data

Software Testing Estimation: **how long** a task would take to complete

testing estimation techniques:

PERT software testing estimation technique: Three-point Estimation
Test Estimate = (O + (4 × M) + P)/6 , O>> BEST CASE , M >> MOST LIKELY , P >> WORST CASE

WBS work breakdown structure >> Work Breakdown Structure >> stepsssss

Wideband Delphi technique >> distributed to a team comprising of 3-7 members for re-estimating the tasks. / **group iteration** to reach an agreement

- **Test Cycle Closure >>** Testing team **meet** , **discuss** and **analyze** testing **artifacts. Deliverables**:
  Test Closure report :**report** that is created once the **testing phase** is **successfully completed** by meeting **exit criteria** defined for the project./It is a **document** that gives a **summary** of all the **tests conducted.**/It also gives a **detailed analysis** of the **bugs removed** and **errors found**. /It is **created** by **test Lead**, **reviewed** by various **stake holders** like **test architect**, **test manager**, **business analyst**, **project manager** and finally approved by **clients**
- Test metrics: **decision points** that lead you to take action/ **judge how efficient** your testing efforts are ex(**Schedule slippage =** (actual end date – estimated end date) / (planned end date – planned start date) * 100 / **Number of tests run per time period =** number of tests run/total time**Fixed Defects Percentage =** (defects fixed/defects reported) * 100

Exit Criteria: minimum set of **conditions** or **activities** that should be **completed** in order to **stop** the software testing/ **given test activity** has been **completed** or **NOT**./defined in **Test Plan.**/

ex:**Verify if** all tests planned have been run. /the level of requirement coverage has been met/NO Critical or high severity defects that are left unresolved./high-risk areas are completely tested./ sw development activities are completed within the projected cost and the projected timelines.

•**bug report** is a critical tool in the software development process/ bridge between testers and developers, facilitating the identification and resolution of issues.
Its purpose is to **state the problem** as clearly as possible so that developers can replicate the defect easily and fix it.

Commonly used documented artifacts related to software testing:

1.Test Plan

A master test plan is developed during the analysis phase.

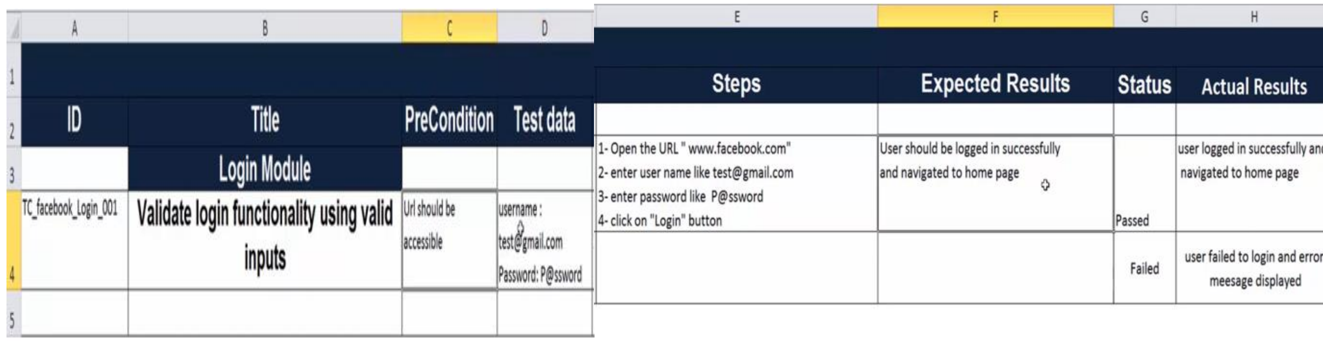
During the design phase, the unit, integration and system test plans are developed.

The dynamic testing is done during implementation.

2. Test Case: set conditions and inputs that can be used while performing testing tasks. are written to keep track of the testing coverage of a software. Ex(functional, negative, logical test cases, physical test cases, UI test cases,)

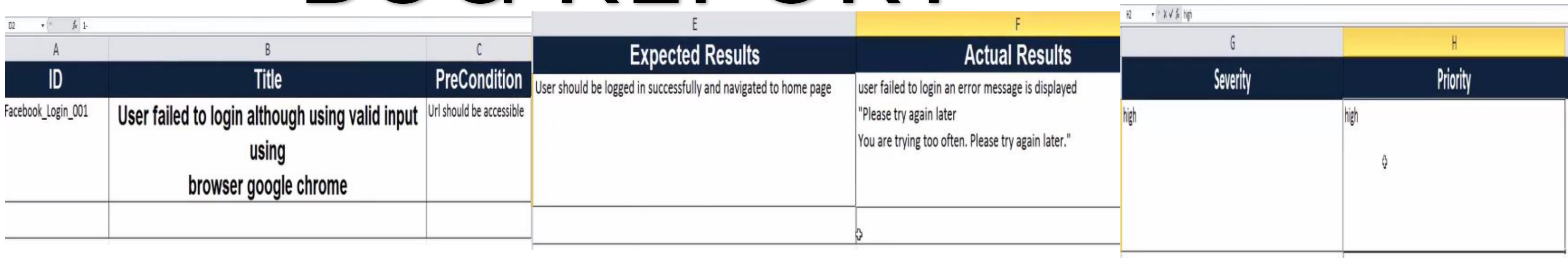
3.Requirements Traceability Matrix: a document that connects requirements throughout the validation process. Used by the validation team to ensure that requirements are not lost during the validation project.It shows the relationship between requirements and test cases
It is a matrix is used to trace requirements. It provides forward and backward traceability. It helps faster impact analysis and reliable assessment to ensure all the business requirements are covered.

TEST CASE



ID	Title	PreCondition	Test data	Steps	Expected Results	Status	Actual Results
	Login Module						
TC_facebook_Login_001	Validate login functionality using valid inputs	Url should be accessible	username : test@gmail.com Password: P@ssword	1- Open the URL " www.facebook.com" 2- enter user name like test@gmail.com 3- enter password like P@ssword 4- click on "Login" button	User should be logged in successfully and navigated to home page	Passed	user logged in successfully an navigated to home page
						Failed	user failed to login and error meesage displayed

BUG REPORT



ID	Title	PreCondition	Expected Results	Actual Results	Severity	Priority
Facebook_Login_001	User failed to login although using valid input using browser google chrome	Url should be accessible	User should be logged in successfully and navigated to home page	user failed to login an error message is displayed "Please try again later You are trying too often. Please try again later."	high	high

Test case design techniques(save testing time get the good test coverage):

❑ **Black-box (specification based)**

- **Equivalence Partitioning testing :**identify groups of inputs that have common characteristics and should be processed in the same way/ criteria for selecting equivalence classes: *coverage* : every input is in one class /*disjointedness* : no input in more than one class/ *representation* : if error with 1 member of class, will occur with all

- **Boundary Value Analysis BVA**: the density of defect is more towards the boundaries, reasons: decide whether they have to use <= operator or < operator Different terminating conditions of for-loops, equirements themselves may not be clearly understood, Can be applied on numeric values and dates./ called robustness testing

- **Decision Table Testing**: compact way to model complicated logic. They are ideal for describing situations in which a number of combinations of actions are taken under varying sets of conditions. Useful for apps : if-then-else logic. b. Logical relationships among input variables.c. Calculations involving subsets of the input variables. d. Cause-and-effect relationships between inputs and outputs
- **Use Case Testing**: very useful for designing **acceptance test cases** with customer/user participation.

Advantages Of Black-box Testing

Tests are performed from the user's point of view. So, there is higher chance of meeting customer's expectations./There is unbiased testing as both the tester and the developer work independently./It is suitable for the testing of very large systems./There is no need for any technical knowledge or language specification./Test cases can be designed as soon as the requirements are finalized.

❑ **White-box Testing (structure based)**

- **Statement Testing & Coverage**:the percentage of executable statements that have been exercised by a test case suite.
- **Branch Testing & Coverage**:used to test every possible branch in the control flow graph of a program.
- **Mutation Testing**:software testing in which certain statements of the source code are changed/mutated to check if the test cases are able to find errors in source code/goal of Mutation testing is ensuring the quality of test cases in terms of robustness that it should fail the mutated source code./Changes made in the mutant program should be kept extremely small / called **Fault-based testing (comment) >>** involves creating a fault in the program.

Advantages of White-box Testing: cover the following issues: Memory leaks Uninitialized memory Garbage collection issues (in JAVA)Performance analysis

❑ **Experience-based Testing:** tests are derived from the tester's **skill** and **insight** and their **experience** with **similar applications** and technologies.

- **Error Guessing** : enumerate a list of possible defects and to design tests that attack these defects. This systematic approach is called **fault attack**
- **Exploratory Testing**: **concurrent** test design, test execution, test logging and learning, based on a **test charter** containing test objectives, and carried out within **time-boxes**/ **useful where** there are few or inadequate specifications and severe time pressure,

LECTURE 9:

SOFTWARE TESTING(IMPLEMENTATION PHASE)

Regression Testing:

1- Corrective : no change & exciting test cases
2- Progressive : change on specification & new test cases
3- Selective : change on program entity & subset of the existing test cases
4- Retest-all strategy: change to a system is minor & reuses all test cases

Reducing Number of Test Cases

- Priority category scheme>> assign a priority code directly to each test description.
- Risk analysis>> highlights the potential problem areas/ The tester uses the results of risk analysis to select the **most crucial tests**.
- Interviewing to find out problematic areas

Choosing test techniques factors (type of system , regulatory standards,level of risk,type of risk,test objective,documentation available,knowledge of the testers,time and budget, …)

Bug life cycle:

•**New:**posted for the first time.

•**Assigned:** After the tester has posted the bug, he assigns the bug to the developing team.

•**Open:** The developer has started analyzing and working on the defect fix. the states namely **Duplicate, Deferred or Not a Bug**-based upon the specific reason.

Fixed: When developer makes necessary code changes ,the bug is passed to testing team.

•**Pending retest:** Here the testing is pending on the testers end.

•**Retest:** The tester do the retesting of the changed code

•**Verified:** The tester tests the bug again after it got fixed by the developer. If the bug is not present in the software, he approves that the bug is fixed and changes the status to “verified”.

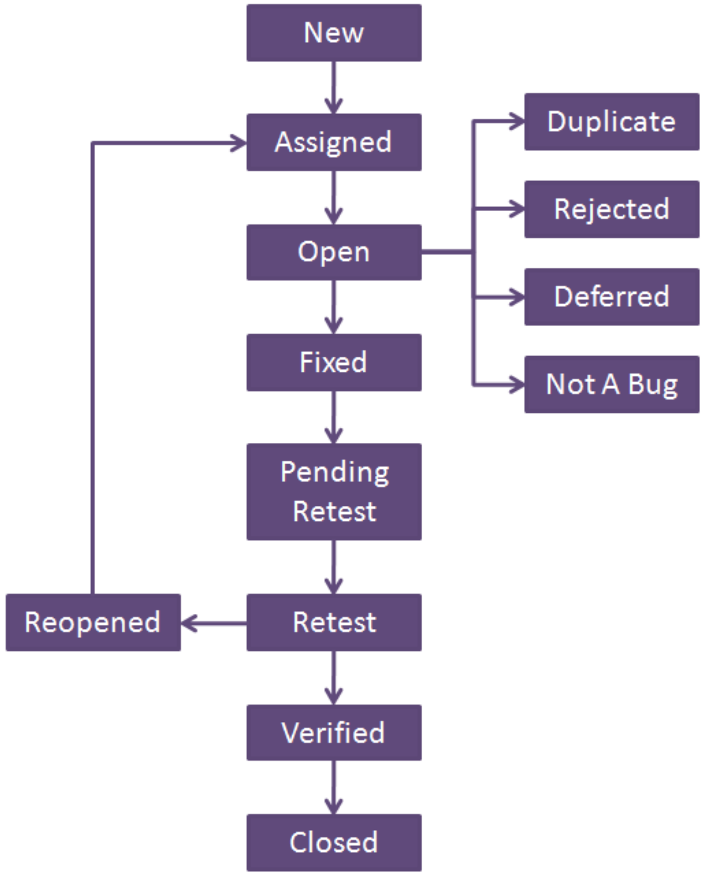
•**Reopen:** If the bug still exists even after the bug is fixed by the developerThe bug goes through the life cycle once again.

•**Closed:** If the tester feels that the bug no longer exists in the software, This state means that the bug is fixed, tested and approved.
Bug Statuses
•**Duplicate:** If the bug is repeated twice**.**

•**Rejected:** If the developer feels that the bug is not genuine

•**Deferred:** The bug is expected to be fixed in next releases.

•**Not a bug:** If there is no change in the functionality of the application. For an example: If customer asks for some change in the look and feel of the application like change of color of some text then it is not a bug**.**



Bug / Defect Lifecycle
http://ISTQBExamCertification.com

To obtain Further Details needed but not in use case

1. Simple Textual Description
2. Structural description in table
3. Sequence diagram

Who is a Business Analyst?

activities that business analysts perform include:

- **understanding** enterprise **problems** and **goals**
- **analyzing needs** and **solutions**
- devising **strategies**
- **driving change** and
- **facilitating** stakeholder collaboration

•COMMENT >>Systems analysts may serve as ***change agents*** who:

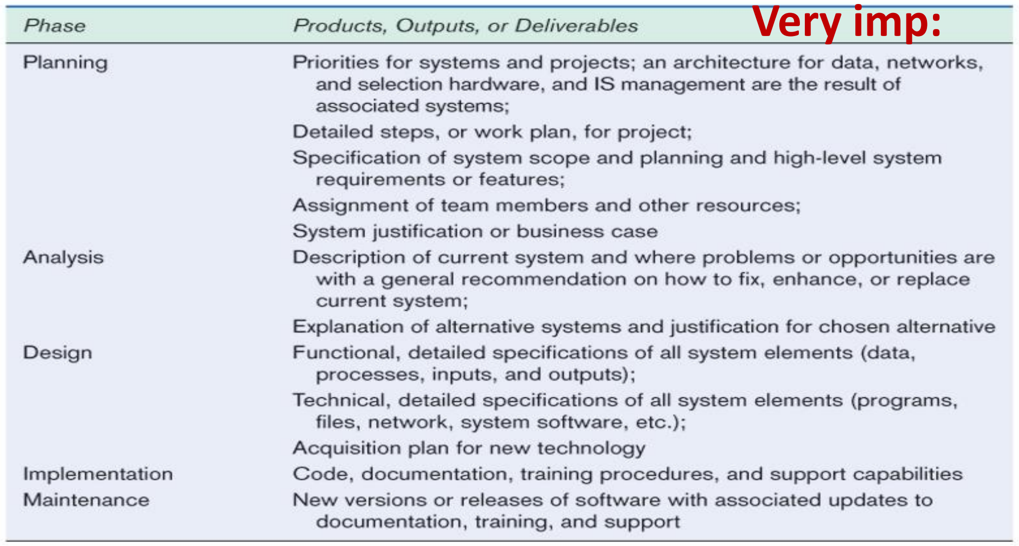
•**identify** the **organizational improvements** needed

•**design systems** to implement those changes, and

•**train** and **motivate** others to use the systems.

Table 1-2 Products of SDLC Phases

Very imp:



Phase	*Products, Outputs, or Deliverables*
Planning	Priorities for systems and projects; an architecture for data, networks, and selection hardware, and IS management are the result of associated systems;
	Detailed steps, or work plan, for project;
	Specification of system scope and planning and high-level system requirements or features;
	Assignment of team members and other resources;
	System justification or business case
Analysis	Description of current system and where problems or opportunities are with a general recommendation on how to fix, enhance, or replace current system;
	Explanation of alternative systems and justification for chosen alternative
Design	Functional, detailed specifications of all system elements (data, processes, inputs, and outputs);
	Technical, detailed specifications of all system elements (programs, files, network, system software, etc.);
	Acquisition plan for new technology
Implementation	Code, documentation, training procedures, and support capabilities
Maintenance	New versions or releases of software with associated updates to documentation, training, and support

el ba2ool 3aleeh msh mohem w bygy

Guidelines for Effective Interviewing

•***Plan*** the interview.

- Prepare interviewee: appointment, priming questions.
- Prepare agenda, checklist, questions.

•***Listen carefully*** and take notes (tape record if permitted).

•***Review notes*** within 48 hours.

•***Be neutral.***

JAD Participants:

•Session Leader: facilitates group process (group management, set agenda, neutral, resolving conflicts and disagreements)

•Users: active, speaking participants

•Managers: active, speaking participants

•Sponsor: high-level champion, limited participation

•Systems Analysts: should mostly listen

•Scribe: record session activities (word processor, CASE tool)

•IS Staff: should mostly listen

Survey/Questionnaire (Cont.):

Strengths

•***Quick*** and relatively ***inexpensive*** to administer.

•***Easier*** to ***collect information*** from a ***larger audience***.

•***Does not*** typically require ***significant time***.

•Suitable for stakeholders are ***geographically dispersed***.

•Effective for obtaining ***quantitative data*** for ***statistical analysis***.

Limitations

•To achieve unbiased results***, specialized skills*** in statistical ***sampling*** methods are needed when surveying a subset of potential respondents.

•The ***response rates*** may be ***too low*** for statistical significance.

•Use of ***open-ended questions*** requires ***more analysis***.

•***Ambiguous questions*** may be ***unanswered*** or answered incorrectly.
Require ***follow-up questions*** or more survey iterations depending on the answers provided.

•Deliverables of Requirements Determination:

•From interviews and observations

- Interview transcripts, observation notes, meeting minutes

•From existing written documents

- Mission and strategy statements, business forms, procedure manuals, job descriptions, training manuals, system documentation, flowcharts

•From computerized sources

- JAD session results, CASE repositories, system prototype displays and reports

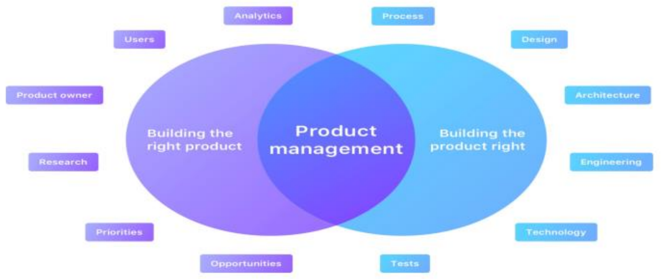
Comparisons:

Functional or black-box testing	Structural or white-box testing or glass-box testing
1. This method focus on *functional requirements* of the software, i.e., it enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program.	**1.** This method focuses on *procedural details*, i.e., internal logic of a program.
2. It is NOT an alternative approach to white-box technique rather is a complementary approach that is likely to uncover a different class of errors.	**2.** It concentrates on internal logic, mainly.
3. Black-box testing is applied during *later stages* of testing.	**3.** Whereas, white-box testing is performed *early* in the testing process.
4. It attempts to find errors in the following categories: **a.** Incorrect or missing functions **b.** Interface errors **c.** Errors in data structures or external database access **d.** Performance errors **e.** Initialization and termination errors	**4.** Whereas, white-box testing attempts errors in following cases: **a.** Internal logic of your program **b.** Status of program
5. It disregards *control structure* of procedural design (i.e., what is the control structure of our program, we do not consider here).	**5.** It *uses control structure* of the procedural design to derive test cases.

	Positive Testing	Negative Testing
1.	Positive testing tries to prove that a given product does what it is supposed to do.	Negative testing is done to show that the product does not fail when an unexpected input is given.
2.	A positive test case is one that verifies the requirements of the product with a set of expected output.	A negative test case will have the input values that may not have been represented in the SRS. These are unknown conditions for the product.
3.	The purpose of positive testing is to prove that the product works as per the user specifications.	The purpose of negative testing is to try and break the system.
4.	Positive testing checks the product's behavior.	Negative testing covers scenarios for which the product is not designed and coded.
5.	Positive testing is done to verify the known test conditions.	Negative testing is done to break the product with unknown test conditions, i.e., test conditions that lie outside SRS.

THE DIFFERENCE BETWEEN ALL THE PRODUCT ROLES

	Project Manager	Product Owner	Product Manager
What they own	A project with beginning and end	A set of feature requirements and delivery of epics	The product, value to users, and P&L for the company
How they plan	Using a Gantt Chart project plan	Using a backlog of issues, stories and tasks	Using a Product Roadmap
Who they deal with	Every stakeholders of the project	Engineering, Design and Product Managers	Stakeholders, customers, and everyone building product
What means success	The shortest critical path and delivering on time	Correctly defined requirements and delivered to production	Product-Market-Fit and retained, happy customers



Cohesion	Coupling
Represent relationships within module	Represent relationships between modules
Highly cohesion gives the best software	Loosely coupling gives the best software

Adaptive	Responsive
Several versions of one design to fit each screen	A single design stretch or shrink to fit screen
Not flexible design	Flexible design

Cohesion	Vs	Coupling
Cohesion is the concept of intra module.		Coupling is the concept of inter module.
Cohesion represents the relationship within module.		Coupling represents the relationships between modules.
Increasing in cohesion is good for software.		Increasing in coupling is avoided for software.
Cohesion represents the functional strength of modules.		Coupling represents the independence among modules.
Highly cohesive gives the best software.		Where as loosely coupling gives the best software.
In cohesion, module focuses on the single thing.		In coupling, modules are connected to the other modules.

Black, White and Experier

Black (Specification Based)	• Based on **requirements** • From the requirements, tests are created • Specification Models can be used for systematic test case design **Techniques** • Equivalence Partitioning • Boundary Value Analysis • Decision Tables • State Transition Testing • Pairwise Testing
White (Structure Based)	• Based on **code and the design** of the system • The tests provide the ability to derive the extent of coverage of the whole application **Techniques** • Statement coverage • Branch Coverage • Decision Coverage
Experience (Black box)	• Based on the **knowledge of the tester** • Using past experienced use & intuition to "guess" where errors may occur **Techniques** • Error Guessing • Exploratory Testing

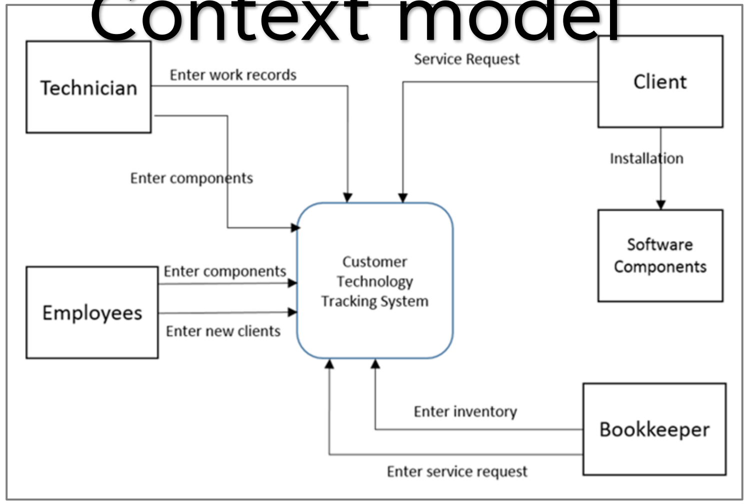
Context Model	Activity Model
Show environment outside system	Show flow of functionality in use case
External perspective	Process perspective

Fan-in	Fan-out
Number of control modules communicate with a subordinate	Number of subordinate modules associated to a control module
Highly fan in is the good solution	A large number of subordinates should be avoided max 7

Steps in Building the Structure Chart

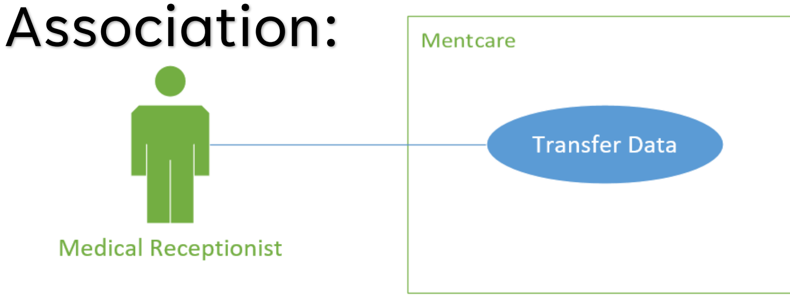
1. Identify top level modules and decompose them into lower levels
2. Add control connections (loops, conditional lines, etc.)
3. Add couples (identify the information that has to pass among the modules i.e. data couple and control couple)
4. Review and revise again and again until complete

Context model

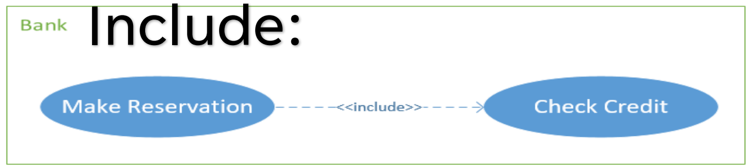


relationships

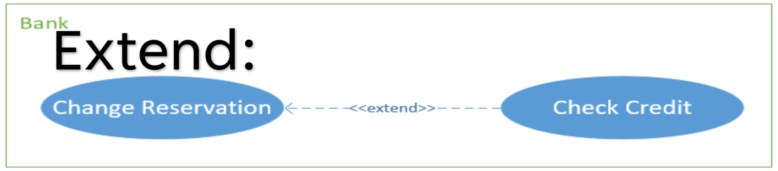
Association:



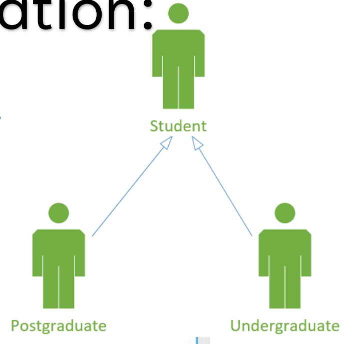
Include:



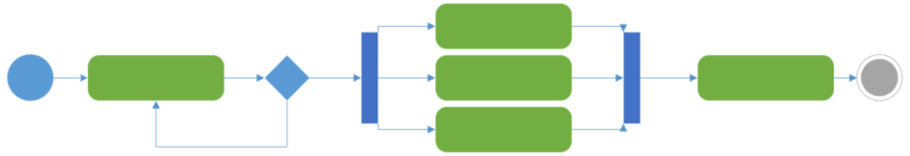
Extend:



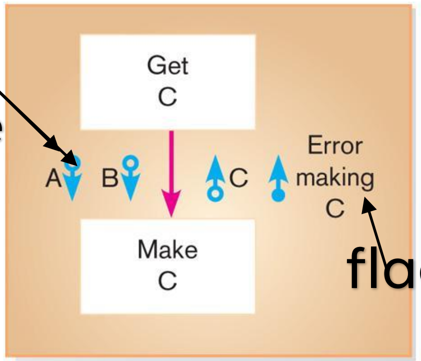
Generalization:



Activity diagram

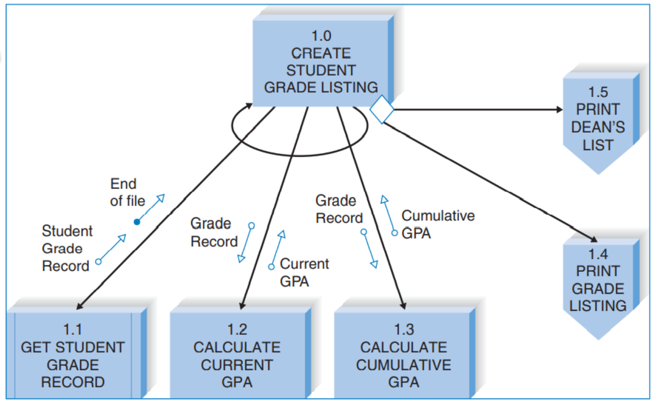


Data couple



flag

Structure chart:



Very impppppp:

- ✔ Library modules have been created whenever possible.
- ✔ The diagram has a high fan-in structure.
- ✔ Control modules have no more than seven subordinates.
- ✔ Each module performs only one function (high cohesion).
- ✔ Modules sparingly share information (loose coupling).
- ✔ Data couples that are passed are actually used by the accepting module.
- ✔ Control couples are passed from “low to high.”
- ✔ Each module has a reasonable amount of code associated with it.