- Need
  - - A problem or opportunity to be addressed.
  - - Needs can cause changes by motivating stakeholders to act.
  - - Changes can also cause needs by enhancing the value delivered by existing solutions.
- Change
  - - The act of transformation in response to a need.
  - - Change works to improve the performance of an enterprise.
- Solution
  - A specific way of satisfying one or more needs in a context.
  - A solution satisfies a need by resolving a problem faced by stakeholders or enabling stakeholders to take advantage of an opportunity.
- Stakeholder
  - - A group or individual with a relationship to the change, the need, or the solution.
  - Stakeholders are often defined in terms of interest in, impact on, and influence over the change.
  - Stakeholders are grouped based on their relationship to the needs, changes, and solutions.
  - A stakeholder is an individual or group that a business analyst is likely to interact with directly or indirectly.
- Value :
  - The worth, importance, or usefulness of something to a stakeholder within a context.
  - can be seen as potential or realized returns, gains, and improvements. It is also possible to have a decrease in value in the form of losses, risks, and costs
  - Value can be tangible or intangible
    - Tangible value is **directly measurable**.
    - Intangible value is **measured indirectly**.
- Context :
  - The circumstances that influence, are influenced by, and provide understanding of the change.
  - Changes occur within a context.
  - The context is everything relevant to the change that is within the environment.
  - Context may include attitudes, behaviors, beliefs, competitors, culture, demographics, goals, governments, infrastructure, languages, losses, processes, products, projects, sales, seasons, terminology, technology, weather, and any other element meeting the definition.
- **Outsourcing** : Turning over responsibility of some or all of an organization's information systems applications and operations to an outside firm.
- • The purpose of Conduct Elicitation is to draw out, explore, and identify information relevant to the change.
- Elicitation Techniques:

1- Traditional Methods

a- Interviewing and Listening
b- Interviewing Groups
c- Survey/Questionnaire
d- Directly observing Users
e- Analyzing procedures and other documents

2-Modern Methods

a- Joint Application Design
b- Prototyping

- Interviewing : Dialogue with stakeholder to obtain their requirements.
- Interview Guide: is a document for developing, planning and conducting an interview.
- Agenda for the interview with approximate time limits
- Guidelines for Effective Interviewing

• Plan the interview.
• Listen carefully and take notes (tape record if permitted).
• Review notes within 48 hours.

• Be neutral.

- Nominal Group Technique (NGT) :
    - A facilitated process that supports idea generation by groups
    - NGT applies to problems with the existing system or ideas for new features in the system being developed.
    - The 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.
- Observation yields only a small segment of data from a possibly vast variety of data sources
- Directly observing Users : you receive only a snapshot image of person or task
- Joint Application Design (JAD):
    - Brings together key users, managers, and systems analysts
    - Purpose: collect system requirements simultaneously from key people
    - Conducted off-site
- CASE tools
    - Used to analyze existing systems
    - Help discover requirements to meet changing business conditions
- Prototyping
    - Is used to elicit and validate stakeholder needs through an iterative process that creates a model or design of requirements.
    - used to optimize user experience, to evaluate design options, and as a basis for development of the final business solution
- Process Modeling : Graphically represent the processes that capture, manipulate, store, and distribute data between a system and its environment and among system components.
- Use case modeling :
    - helps analysts analyze the functional requirements of a system.
    - helps developers understand the functional requirements of the system without worrying about how those requirements will be implemented
    - is a picture that **shows system behavior**, along with the key actors that interact with the system.
- Boundary : A boundary is the dividing line between the system and its environment. • **Use case**s are **within** the boundary. • **Actors** are **outside** of the boundary.
- A connection is an **association** between an actor and a use case
- < extends > Relationship -> Specialized use case extends the general use case
- < include > Relationship -> Links to general purpose functions, used by many other use cases
- Data Flow Diagram (DFD) : A picture of **the movement of data** between external entities and the processes and data stores within a system.
- A data flow can be best understood as data in motion, moving from one place in a system to another.
- Context diagram :
    - shows the system boundaries, external entities that interact with the system, and major information flows between entities and the system.
    - only one process symbol, and no data stores shown.
- Data store, Data flow ,Source and sink labels should be noun phrases.
- Process labels should be verb phrases.
- Bidirectional flow between process and data store is represented by two separate arrows.
- Forked data flow must refer to exact same data item
- Joined data flow must refer to exact same data item
- Data flow from a process to a data store means update (insert, delete or change).
- Data flow from a data store to a process means retrieve or use.
- Functional Decomposition : An iterative process of breaking a system description down into finer and finer detail

- Level-0 DFD : Representation of system's major processes at high level of abstraction
- If this is the lowest level of the hierarchy, it is called a **primitive DFD.**
- DFD Balancing : The conservation of inputs and outputs to a data flow process when that process is decomposed to a lower level
- Time is **not represented** well on DFDs.
- Gap Analysis : The process of discovering discrepancies between two or more sets of data flow diagrams or discrepancies within a single DFD
- Logic modeling involves representing internal structure and functionality of processes depicted on a DFD.
- Logic modeling can also be used to show when processes on a DFD occur
- **Decision Tables**
    - **A matrix representation of the logic of a decision**
    - Specifies the possible conditions and the resulting actions
    - Best used for **complicated decision logic**
- 3 Parts of a Decision Table :

    1. Condition stubs: Lists condition relevant to decision
    2. Action stubs: Actions that result from a given set of conditions
    3. Rules: Specify which actions are to be followed for a given set of conditions

- **Indifferent Condition**: Condition whose value does not affect which action is taken for two or more rules
- **Decision Trees** : A graphical representation of a **decision situation (**Read from left to right**)**
- Main components :
    - Decision points represented by nodes
    - Actions represented by ovals
    - Particular choices from a decision point represented by arcs
- A sequence diagram depicts the interactions **among objects during a certain period of time.(struct chart اخد بالي الفرق بينه وبين)**
- the lifeline : Each object is shown as a vertical dashed line
- A message is shown as a solid arrow from the sending object to the receiving object.
- Messages Types :
    1- A synchronous message
    2- return message: may provide the caller with some return value(s) or simply acknowledge to the caller that the operation called has been successfully completed
    3- An asynchronous message
- You can avoid cluttering(تشويش) up your diagrams by minimizing the use of return messages
- A state diagram is used to model the dynamic behavior of a class in response to time and changing external stimuli
- Uses of state chart diagram :
    - To state the events responsible for change in state (we do not show what processes cause those events).
    - To model the dynamic behavior of the system .
    - To understand the reaction of objects/classes to internal or external stimuli.
- The purpose of requirement traceabillity is to ensure that requirements and designs at different levels are aligned to one another.
- Conceptual Data Modeling : A detailed model that captures the overall structure of data in an organization
- **Primary deliverable** is an **entity-relationship (E-R) diagram or class diagram**
- **Second deliverable** is a set of entries about data objects to be **stored in repository or project dictionary.**
- Class: a logical grouping of objects with similar attributes and behaviors
- Encapsulation: the technique of hiding internal implementation details of an object from external view
- Class Diagram : A diagram showing the static structure of an object-oriented model
- Multiplicity: indicates how many objects participate in a give relationship

- UML associations are analogous to E-R relationships. UML multiplicities are analogous to E-R cardinalities. UML association classes are analogous to E-R associative entities
- Associative Class : An association with its own attributes, operations, or relationships
- Derived items are represented with a slash (/).
- Derived attributes are calculated based on other attributes.
- Types of superclasses :
    - Abstract: cannot have any direct instances
    - Concrete: can have direct instances
- Polymorphic Operations : The same operation may apply to two or more classes in different ways
- Methods : the implementation of an operation
- class-wide attribute: value common to an entire class
- Aggregation : A part-of relationship between a component and an aggregate object
- Composition : An aggregation in which the part object belongs to only one aggregate object and lives and dies with the aggregate object

Finalizing Design

- Specification Documents Contains:
    - ♦ Overall system description.
    - ♦ Interface requirements.
    - ♦ System features.
    - ♦ Nonfunctional requirements.
    - ♦ Supporting diagrams and models.
- Requirements management tools (RM) : make it easier to keep documents up to date, add additional requirements and link related requirements
- structure chart :
    - A high-level diagram , is created to illustrate the organization and interaction of **the different pieces of code** within the program.
    - 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)
- Modules: a self-contained component of a system that is defined by its function
- A control module : is a higher-level component that contains the logic for performing other modules, and the components that it calls and controls are considered subordinate modules.
- Each process of a DFD tends to represent one module on the structure chart.
- If leveled DFDs are used, then each DFD level tends to correspond to a different level of the structure chart hierarchy.
- The process on the context-level DFD would correspond to the top module on the structure chart.
- Special symbols used in structure charts :
    - Data couple: Diagrammatic representation of the data exchanges between two modules.
    - Flag: Diagrammatic representation of a message passed between two modules.
- Measures of good design include

    ♦cohesion, ♦coupling, ♦appropriate levels of fan-in and fan-out
- Cohesion refers to how well the lines of code within each module relate to each other.
- Cohesion Types :
    - Functional cohesion : all elements of the modules contribute to performing a single task.
    - Temporal cohesion : functions are invoked at the same time.
    - 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.
- Coupling involves how modules are interrelated. (Modules should be loosely coupled.)

- Data coupling occurs when modules pass parameters or specific data to each other. It is preferred.
- Content coupling occurs when one module actually refers to the inside of another module. It is the worst coupling type.

Cohesion	coupling
code performs one function effectively	code performs among functions
The highly cohesion is the best software	Loosely coupling gives the best software

- Fan-in describes the number of control modules that communicate with a subordinate.
- **high fan-in** indicates that a module is reused in many places on the structure chart
- Fan-out :
  - is the number of subordinates associated with a single control.
  - A large number of subordinates associated with a single control should be avoided
  - The general rule of thumb is to limit a control module's subordinates to approximately **seven**
- Assess the Structure Chart for Quality
  - Data couples that are passed are actually used by the accepting module
  - Control modules have no more than 7 subordinate
  - Loose coupling
  - High cohesion
- Four components for program specification:

  - Program information; - Events; - Inputs and outputs; - Pseudocode.
- adaptive design : Craft 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
  - is a GUI design approach used to create content that adjusts smoothly to various screen sizes.
- relative units -> (%)
- Responsive Design Core Principles :

■ Fluid Grid System ■ Fluid Image Use ■ Media Queries

- Fluid Grid System: use relative unit -> a single design that expands or shrinks according to users' screen size.
- Fluid Image Use : Unlike text, images aren't naturally fluid. That means they default to the same size and configuration from one device's screen to the next.
- Media Queries : These are filters you use to detect the browsing device's dimensions and make your design appear appropriately.

Seven Testing Principles

1. Testing shows presence of defects
2. Exhaustive testing is impossible
3. Early testing : To find defects early, testing activities shall be started as early as possible in **SDLC.**
4. Defect clustering : Testing effort shall be focused proportionally to the expected and observed defect density of modules.
5. Pesticide paradox : If the same tests are repeated over and over again, eventually the same set of test cases will no longer find any new defects.
6. Testing is context dependent
7. Absence-of-errors fallacy : Finding and fixing defects does not help if the system built is unusable and does not fulfill the users' needs and expectations.

- quality assurance :
  - Set of activities for ensuring quality in the **processes** by which products are developed
  - Aim -> To prevent defects and focus in process
  - Goal -> to improve development and test processes
  - verification
- quality control :
  - Set of activities for ensuring quality in the **products**

    - The activities focus on identifying defects int actual products produced
    - Aim -> to identify defect in finished product
    - Goal -> to identify defect after a product is developed and before it's released
    - Validation / system testing
- artificial data -> (test data).
- black-box testing :
    - The technique of testing without having any knowledge of the interior workings of the application
    - The tester is unaware of the system architecture and does not have access to source code.
- White-box testing :
    - glass testing or open-box testing.
    - White-box testing is the detailed investigation of internal logic and structure of the code.
    - in order to perform white-box testing on an application, a tester needs to know the internal workings of the code.
- Function 2 depends on Function 1. How to test them? • If F1 is not ready, a **driver** is needed to be able to test F2. • If F2 is not ready, a **stub** is needed to be able to test F1.
- User acceptance testing (UAT) -> users have different perspective than the developers. Moreover, the influences from the user's working environment have a major effect on the reliability, performance, usability and robustness of a system.
- Regression testing :
    - All tests are re-run every time a change is made to the program.
    - is testing the system to check that changes have not 'broken' previously working code.
- Localization Testing -> quality team , Sanity Testing -> testers , Smoke Testing -> developers or testers ,UAT -> users
- Smoke test -> verify the main functionality but not in deep , بعملهinitial build في
- Sanity test -> verify the bug those fixed previous build & new features , بعملهat the end of phase SDLC في
- **Test Estimation : a management activity which approximates how long a task would take to complete.**
- PERT -> Program Evaluation Review Technique
- Work Breakdown Structure Technique (WBS) -> structure chart ع اقدر اطبقه
- Wideband Delphi Technique -> method speaks more on experience rather than any statistical formula
- Exit Criteria : is a minimum set of conditions or activities that should be completed in order to **stop** the software testing. **defined in Test Plan**
- Entry Criteria : is a minimum set of conditions that should be met (prerequisite items) **before starting the software testing**.

Software Testing Reports

- 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**
- The **physical** test case describes in practical terms **what has to be done**
- The **logical** test case describes what is tested and not **how it is done**.
- Positive test cases ensure that users can perform appropriate actions when using valid data.
- Negative test cases are performed to try to "break" the software by performing invalid (or unacceptable) actions, or by using invalid data.
- Requirements Traceability Matrix (RTM) is a document that connects requirements throughout the validation process.
- **Test Closure Report** is created by **test Lead,** reviewed by various **stakeholders** like **test architect, test manager, business analyst, project manager** and finally approved by **clients**.
- Decision coverage is stronger than statement coverage; 100% decision coverage guarantees 100% statement coverage, but not vice versa(مش العكس)

- Statement coverage is the assessment of the percentage of executable statements that have been exercised by a test case suite.
- Statement test cases for a segment of code will always =< of Branch/Decision test cases
- Branch/Decision testing is a **more complete form** of testing than statement testing
- **fault attack :** A structured approach to the error guessing technique is to enumerate a list of possible defects and to design tests that attack these defects.
- **Exploratory testing** is concurrent test design, test execution, test logging and learning, based on a test charter containing test objectives, and carried out within time-boxes.
- Test-driven development (TDD) : was introduced as part of **agile methods** such as **Scrum and Extreme** Programming.
- Benefits of test-driven development:
- Code coverage
- Regression testing
- Simplified debugging
- **system documentation**

Documentation

- Internal documentation system documentation that is part of the program source code (written in a program as comments) or is generated at compile time
- External documentation system documentation that includes the outcome of structured diagramming techniques such as data flow and E-R diagrams
- **Support**: providing ongoing educational and problem-solving assistance to system users. It can be expensive and time consuming. Vendors usually charge for monthly support.
- **Help desk**: a single point of contact for all user inquiries and problems about a particular information system or in a particular department.
- Installation strategies :
  - Direct Installation : Old system turned off; new system is turned on. (greatest risk , Least expensive.)
  - Parallel Installation : Old and new systems are run simultaneously until the end users and project coordinators are fully satisfied that the new system is functioning correctly and the old system is no longer necessary (Low risk , Highest cost)
  - Pilot Installation : This approach allows for the conversion to new system, using either direct or parallel method, at a single location.(less risky in terms of any loss of time or delays in processing.)
  - Phased Installation : Allows for the new system to be brought online as ' A series of functional components that are logically ordered to minimize disruption to the end users and the flow of business. (Low risk. , Long time.)

- Maintenance :
  - is the longest phase in the SDLC
  - a subset of the activities of the entire development process.
  - new version of the software and new versions of all design documents created or modified during the maintenance effort.
  - changes made to a system to fix or enhance its functionality.
- Types of System Maintenance
  - Corrective maintenance: changes made to a system to repair flaws in its design, coding, or implementation
  - Adaptive maintenance: changes made to a system to evolve its functionality to changing business needs or technologies
  - Perfective maintenance: changes made to a system to add new features or to improve performance
  - Preventive maintenance: changes made to a system to avoid possible future problems

- Factors of The Cost of Maintenance :
  ♦Hidden defects ♦ Number of customers for a given system ♦ Quality of system documentation ♦ Maintenance personnel ♦ Tools ♦ Well-structured programs
- Measuring Maintenance Effectiveness :
  ♦ Number of failures ♦ Time between failures ♦ Type of failures
- Mean Time Between Failures (MTBF) : Measurement of error occurrences that can be tracked over time to indicate the quality of a system
- Configuration management (CM):
  1. Software systems are constantly changing during development and use.
  2. is concerned with the policies, processes and tools for managing changing software systems.
  3. You need CM because it is easy to lose track of what changes and component versions have been incorporated into each system version.
- Version management (VM):
  1. is the process of keeping track of different versions of software components or configuration items.
  2. It is the process of managing codelines and baselines
- A **codeline** is a sequence of versions of components of source code with later versions in the sequence derived from earlier versions.
- A **baseline** is a definition of a specific system. It specifies the component versions that are included in the system plus a specification of the libraries used, configuration files, etc.
- The **mainline** is a sequence of system versions developed from an original ***baseline***
- Version control (VC) :
  1. systems identify, store and control access to the different versions of components.
  2. use the concept of :
     - a project repository : maintains the 'master' version of all components. It is used to create baselines for system building.
     - a private workspace.
- **Centralized** systems, where there **is a single master repository that maintains all versions of the software components** that are being developed. **Subversion** is a widely used example of a centralized VC system.
- **Distributed** systems, where **multiple versions of the component repository exist at the same time**. **Git** is a widely used example of a distributed VC system.
- Centralized and distributed VC systems support independent development of shared components in **different ways**
- Distributed VC System : A **'master' repository** is created on a server that maintains the code produced by the development team.
- Benefits of distributed version control:
  1. It provides a backup mechanism for the repository.
  2. It allows for off-line working so that developers can commit changes if they do not have a network connection.
  3. Developers can compile and test the entire system on their local machines and test the changes that they have made
- **Distributed version control** is essential for **open-source development**
- The deltas(a list of differences): define how to re-create earlier system versions
- **System building** is the process of creating a complete, executable system by compiling and linking the system components, external libraries, configuration files, etc.
  - The development system, which includes development tools such as compilers and source code editors.
  - The build server, which is used to build definitive, executable versions of the system
  - The target environment, which is the platform on which the system executes.
- **Continuous integration (CI)** involves rebuilding the mainline frequently, after small source code changes

- Agile methods recommend that very frequent system builds should be carried out, with automated testing used to discover software problems.
- Change management :
  - is intended to ensure that system evolution is a managed process and that priority is given to the most urgent and cost-effective changes.
  - process is initiated when a system stakeholder completes and submits a change request CR describing the change required.
  - This could be a bug report, or a request for additional functionality to be added to the system
- Configuration management (CM) :
  - is concerned with the policies, processes and tools for managing changing software systems.
  - You need CM because it is easy to lose track of what changes and component versions have been incorporated into each system version
- Configuration Management Activities :
  - Release system
  - Release management
  - Change management
  - Version management
  - Component management
  - System building
  - Proposals change
  - System version
- Release Management:
  - A system release is a version of a software that is distributed to customers.
  - two types of release:
    - major releases, deliver significant new functionality (**customers usually have to pay for software vendor)**
    - minor releases, repair bugs and fix customer problems that have been reported (usually distributed **free of charge**)
    - Operating system OS 10.**9**.**2** This means **minor** release 2 of **major** release 9 of OS 10.
- Release tracking : In the event of a problem, it may be necessary to reproduce exactly the software that has been delivered to a particular customer