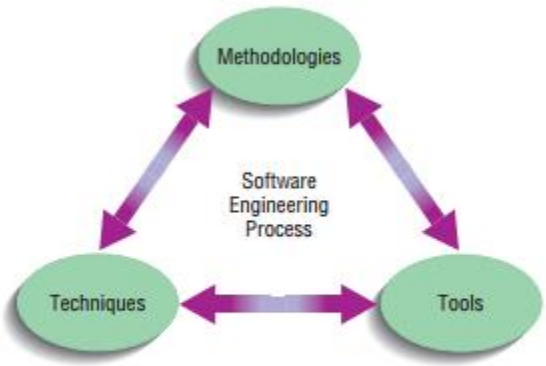
<u>Software engineering process-</u> methodologies, techniques, and tools

information systems are created through software engineering process which involves proven methodologies, techniques, and tools.



- ***Methodologies*** are a sequence of step-by-step approaches that help develop your final product: the information system. Most methodologies incorporate several development techniques, such as direct observations and interviews with users of the current system.
- ***Techniques*** are processes that you, as an analyst, will follow to help ensure that your work is well thought-out, complete, and comprehensible to others on your project team. Techniques provide support for a wide range of tasks, including conducting thorough interviews with current and future users of the information system to determine what your system should do, planning and managing the activities in a systems development project, diagramming how the system will function, and designing the reports, such as invoices, your system will generate for its users to perform their jobs.
- ***Tools*** are computer programs, such as computer-aided software engineering (CASE) tools, that make it easy to use specific techniques.

These three elements— methodologies, techniques, and tools—work together to form an organizational approach to systems development.

System

Understanding systems and how they work is critical to understanding systems analysis and design (software development).

Definition of a System and Its Parts

system is an interrelated set of business procedures (or components) used within one business unit, working together for some purpose. For example, a system in the payroll department keeps track of checks, whereas an inventory system keeps track of supplies. The two systems are separate. A system has nine characteristics.

1. **Components:** A system is made up of components**.** Component irreducible part or aggregation of parts that makes up a system; also called a subsystem.
2. **Interrelated components:** Dependence of one part of the system on one or more other system parts. The function of one component is tied to the functions of the other e.g. the sales order component might depend on others such as shopping cart component
3. **Boundary:** A system has boundary within which all its components are contained and that establishes the limits of the system separating it from other system. Components within the boundary can be changed, whereas systems outside the boundary cannot be changed

   -The line that marks the inside and outside of a system and that sets off the system from its environment.

4. **Purpose:** All of the components work together to achieve some overall purpose for the larger system. The overall goal or function of a system. It's the systems reason for existence

5. **Environment:** Everything external to a system that interacts with the system.an information system interacts with its environment by receiving data and information.
6. **Interfaces:** Point of contact where a system meets its environment or where subsystems meet each other.
7. **Constraints:** A limit to what a system can accomplish. A system must face constraints in its functioning because of limitations (in terms of its capacity, speed or capabilities) to what it can do and how it can achieve its purpose within its environment. Some of these constraints are imposed inside the system and others are imposed by the environment. The system is constrained if electrical power is cut. A system takes input from its environment in order to function
8. **Input:** Inputs are the information that enters into the system for processing. The system receives inputs from its environment
9. **Output:** The main objective of a system is to get an output which is helpful for its user. Output is the final outcome of processing.

Important Information System Concepts

Systems analysts need to know several other important systems concepts:

1. Decomposition
2. Modularity
3. Coupling
4. Cohesion

Decomposition

Decomposition is the process of breaking down a system into its smaller components? These components may themselves be systems (subsystems) and can be broken down into their components as well. Decomposing a system also allows us to focus on one particular part of a system, making it easier to think of how to modify that one part independently of the entire system. Decomposition is a technique that allows the systems analyst to:

1. Break a system into small, manageable, and understandable subsystems
2. Focus attention on one area (subsystem) at a time, without interference from other areas.
3. Concentrate on the part of the system pertinent to a particular group of users, without confusing users with unnecessary details
4. Build different parts of the system at independent times and have the help of different analysts

Modularity

Modularity is a direct result of decomposition. It refers to dividing a system into chunks or modules of a relatively uniform size. Modules can represent a system simply, making it easier to understand and easier to redesign and rebuild. For example, each of the separate subsystem modules for the MP3 player shows how decomposition makes it easier to understand the overall system.

Broadly speaking, **modularity** is the degree to which a system's components may be separated and recombined.

Modular programming is a software design technique that emphasizes separating the functionality of a program into independent, interchangeable **modules**, such that each contains everything necessary to execute only one aspect of the desired functionality.

Example1: computers cell phones use modularity to overcome changing customer demands and to make manufacturing process more adaptive to change

Example2: Enterprise resource planning (**ERP**) is business process management software that allows an organization to use a system of integrated applications to manage the business and automate many back office functions related to technology, services and human resources.

Coupling

Means that subsystems are dependent on each other. Subsystems should be as independent as possible. If one subsystem fails and other subsystems are highly dependent on it, the others will either fail themselves or have problems functioning.

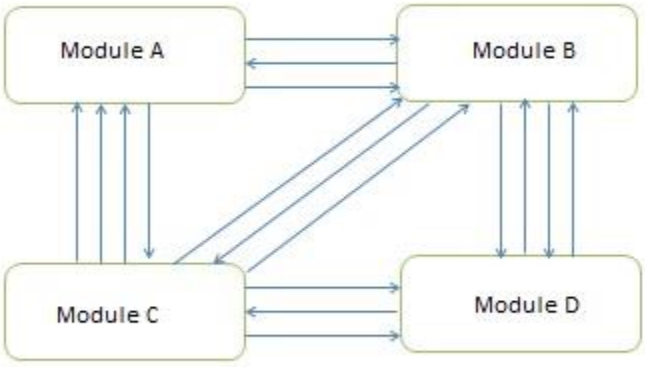
Coupling it is a relationship that exist between system components. It is a tool that help in understanding the system. Even if one identifies all components, unless one understands the relationship between them, one does not understand how they function.

Example: To understand a DBMS we need to know the type of relationship between tables.

Coupling could result because of **several factors**: a module may refer to variables defined in another module or a module may call methods of another module and use the return values. The amount of coupling between modules can vary. In general, if modules do not depend on each other's implementation, i.e., modules depend only on the published interfaces of other modules and not on their internals, we say that the coupling is **low**. In such cases, changes in one module will not necessitate changes in other modules as long as the interfaces themselves do not change. Low coupling allows us to modify a module without worrying about the ramifications of the changes on the rest of the system. By contrast, **high** coupling means that changes in one module would necessitate changes in other modules, which may have a **domino effect** and also make it harder to understand the code.

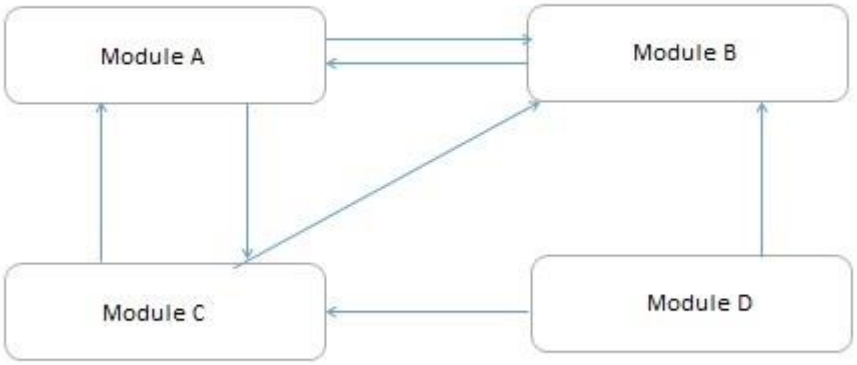
High Coupling

These type of systems have interconnections with program units dependent on each other. Changes to one subsystem leads to high impact on the other subsystem.



Low Coupling

These type of systems are made up of components which are independent or almost independent. A change in one subsystem does not affect any other subsystem.



Loose and Tight coupling

In computing and systems design a **loosely coupled** system is one in which each of its components has, or makes use of, little or no knowledge of the definitions of other separate components. The reverse holds for **tightly coupled** system

Disadvantages of tightly coupled system

Tightly coupled systems tend to exhibit the following developmental characteristics, which are often seen as disadvantages:

1. A change in one module usually forces a **ripple effect** of changes in other modules.
2. Assembly of modules might require **more effort and/or time** due to the increased inter-module dependency.
3. A particular module might be **harder to reuse and/or test** because dependent modules must be included.

Cohesion

It is the extent to which a subsystem performs a single function. Each module provides certain functionality; cohesion of a module tells us how well the entities within a module work together to provide this functionality. Cohesion is a measure of how focused the responsibilities of a module are. If the responsibilities of a module are unrelated or varied and use different sets of data, cohesion is reduced. Highly cohesive modules tend to be more reliable, reusable, and understandable than less cohesive ones.

To increase cohesion, we would like that all the constituents contribute to some well-defined responsibility of the module. This may be quite a challenging task. In contrast, the worst approach would be to arbitrarily assign entities to modules, resulting in a module whose constituents have no obvious relationship.

The best modules are those that are functionally cohesive

Logical cohesion

Logical cohesion is when parts of a module are grouped because they are logically categorized to do the same thing even though they are different by nature (e.g., grouping all mouse and keyboard input handling routines).

Temporal cohesion

Temporal cohesion is when parts of a module are grouped by when they are processed - the parts are processed at a particular time in program execution

(e.g., a function which is called after catching an exception which closes open files, creates an error log, and notifies the user).

Procedural cohesion

Procedural cohesion is when parts of a module are grouped because they always follow a certain sequence of execution (e.g., a function which checks file permissions and then opens the file).

Communicational/informational cohesion

Communicational cohesion is when parts of a module are grouped because they operate on the same data (e.g., a module which operates on the same record of information).

Sequential cohesion

Sequential cohesion is when parts of a module are grouped because the output from one part is the input to another part like an assembly line (e.g., a function which reads data from a file and processes the data).

Functional cohesion (best)

Functional cohesion is when parts of a module are grouped because they all contribute to a single well-defined task of the module

Types(classes) of information systems

Given the broad range of people and interests represented in an organization, it could take several different types of information system to satisfy all of an organization's information systems needs.

Several different classes of information systems exist. This classes are distinguished by what the system does or by the technology used to construct the system.

Part of the systems analyst job is to determine which kind of system will best address the organizational problem or opportunity in a given situation. In addition, different classes of system might require different methodologies,

techniques and tools for development. Some major types of information systems includes the following;

Transaction Processing System (TPS)

-Automate the handling of data for business activities or transactions. Major focus is capturing transaction data and keeping it in a computerized database. For example, a bank's TPS captures information about withdrawals from and deposits to customers account. Data about each transaction are captured, transactions are verified and the validated transactions are stored.

-The analysis and design of a TPS requires the analysts to focus on the firm's current procedures for processing transactions. How does the organization track, capture process and output data?

-Also known as **OLTP** -Online Transactional Processing system

-Goal: improve transaction processing by increasing speed, enhancing productivity, integrating it with other systems, simplifying processes

-TPS are a major source of information (data) to other information systems

> **Other examples: sales order entry system, hotel reservation system, employee record keeping system**

Management Information System (MIS)

-Use raw data from TPS systems, and converts them into meaningful aggregate form. **MIS are designed to process transactional data into standard reports.** Converts Data to Information

For example, a TPS keeps track of sales, and **a Management Information System** can pin point the products items that are selling quickly and those selling slowly. As a result, the MIS can direct the manufacturing department regarding what to produce when.

-Developing an MIS call for good understanding of what kind of information managers require and how managers use information in their jobs

-Goal: provide information to management to help them manage the business

Decision Support System (DSS)

-DSS are designed to help decision make with decisions. Whereas an MIS produce a report, a DSS provides an interactive environment in which decision makes can manipulate data and models of business operations quickly.

-DSS help managers make decisions by analyzing data in different ways. Managers can make changes to their data, for example changing the interest rates and see how those changes affect the parts of the business they manage.

-Managers tries to determine what it takes to turn a downward trend into an upward trend.

-DSS has three parts: A **Database** which can be extracted from a TPS or DSS, a **mathematical or graphical model** of business processes and a **user interface** (dialogue module) that provides a way for decision maker to communicate with the DSS.

-DSS Interactively assist with decision making by applying mathematical or logical models and a dialogue of interactions to solve unstructured problems (allow for what ifs)
Goal: provide comparisons of alternatives and recommendation of preferred option

The analysis and design of DSS often concentrates on the three main DSS components – database, model base and user dialogue

Figure 1.7 Depictions of Classes of Information Systems: TPS, MIS, DSS