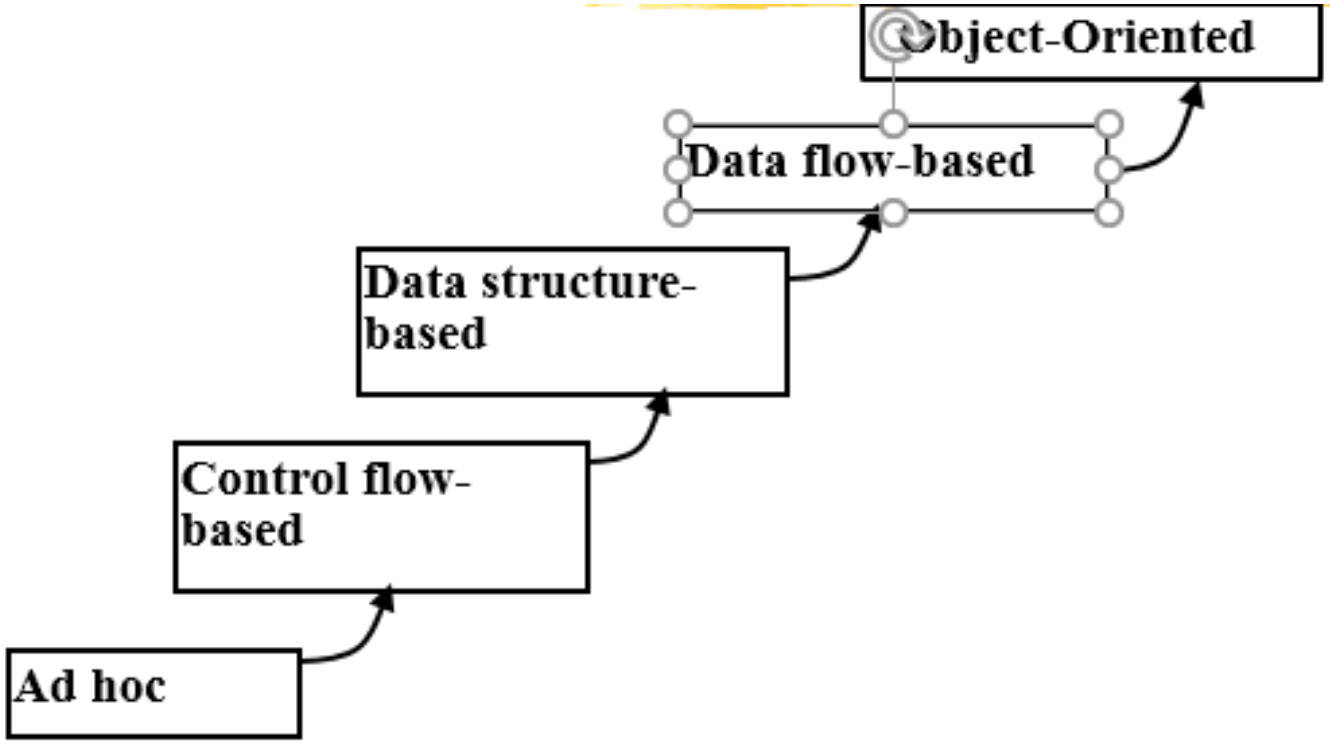
Software Engineering

Prepared by: Neha Tripathi

Emergence of Software Engineering

Software engineering discipline is the result of advancement in the field of technology. In this section, we will discuss various **innovations and technologies** that led to the emergence of software engineering discipline.

Emergence of Software Engineering



Early Computer Programming (1950s): (Exploratory programming style)

- Programs were being written in **assembly language**.
- Programs were limited to about **a few hundreds of lines** of assembly code.
- Every programmer developed his **own style** of writing programs:

  -according to his **intuition** (exploratory programming).

High-Level Language Programming (Early 60s)

- High-level languages such as **FORTRAN, ALGOL, and COBOL** were introduced:
  - This reduced software development efforts greatly.
- Software development style was still **exploratory**.
  - Typical program sizes were limited to a **few thousands of lines** of source code.

Control Flow-Based Design (late 60s)

- **Size and complexity** of programs increased further:
  - exploratory programming style proved to be insufficient.
- Programmers found:

  -**very difficult to write cost-effective and correct programs.**
- Programmers found:

  -**programs written by others very difficult to understand and maintain.**
- To cope up with this problem, experienced programmers advised: ``Pay particular attention to the design of the program's control structure.''
- **A program's control structure indicates:**
  - **the sequence in which the program's instructions are executed.**
- To help design programs having good control structure:
  - **flow charting technique** was developed.

- Using **flow charting technique**:
  - one can represent and design a program's control structure.
  - Usually one understands a program:
    - by mentally simulating the program's execution sequence.
- A program having a messy flow chart representation:
  - difficult to understand and debug.
- It was found:
  - GO TO statements makes control structure of a program messy
  - GO TO statements alter the flow of control arbitrarily.
  - The need to **restrict use of GO TO** statements was recognized.

- But, soon it was conclusively proved:
  - only **three programming constructs** are sufficient to express any programming logic:
    - **sequence** (e.g. a=0;b=5;)
    - **selection** (e.g. if(c=true) k=5 else m=5;)
    - **iteration** (e.g. while(k>0) k=j-k;)
- Everyone accepted:
  - it is possible to solve any programming problem without using GO TO statements.
  - This formed the basis of **Structured Programming methodology.**

Structured Programming

- A program is called **structured**
  - **when it uses only the following types of constructs:**
    - **sequence,**
    - **selection,**
    - **iteration**
- Unstructured control flows are avoided.
- Consist of a neat set of modules.
- Use single-entry, single-exit program constructs.
- However, violations to this feature are permitted:
  - due to practical considerations such as:

    -**premature loop exit to support exception handling.**

Structured programs are:

- **Easier to read and understand,**
- **easier to maintain,**
- **require less effort and time for development.**

Data Structure-Oriented Design (Early 70s)

- Soon it was discovered:
  - it is important to pay more attention to the **design of data structures** of a program
    - than to the design of its control structure.
- Techniques which emphasize designing the data structure:
  - derive program structure from it:

    -are called **data structure-oriented design techniques.**

- Example of data structure-oriented design technique:
  - **Jackson's Structured Programming(JSP)** methodology

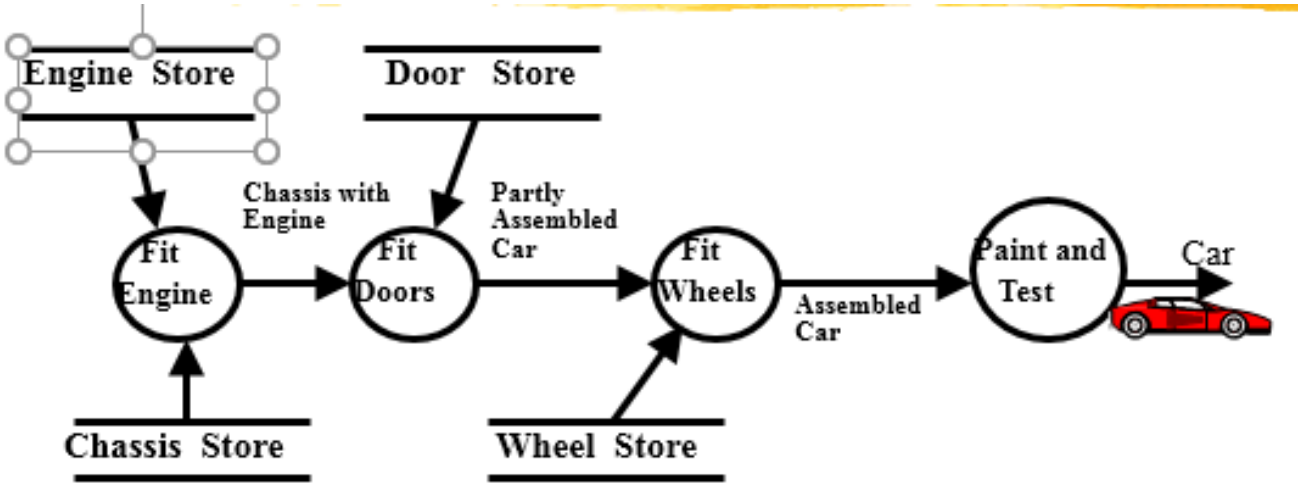
    -developed by Michael Jackson in 1970s.

- JSP technique:
  - program code structure should correspond to the data structure.
- In JSP methodology:
  - a program's data structures are first designed using notations for
    - sequence, selection, and iteration.
  - Then data structure design is used :
    - to derive the program structure.
- Several other data structure-oriented Methodologies also exist:
  - e.g., Warnier-Orr Methodology.

Data Flow-Oriented Design (Late 70s)

- Data flow-oriented techniques advocate:
  - the **data items** input to a system must first be identified,
  - **processing required** on the data items to produce the required outputs should be determined.
- Data flow technique identifies:
  - different processing stations (functions) in a system
  - the items (data) that flow between processing stations.
- Data flow technique is a generic technique:
  - **can be used to model the working of any system**
    - not just software systems.
- A major advantage of the data flow technique is its **simplicity.**

Data Flow Model of a Car Assembly Unit



Object-Oriented Design (80s)

- Object-oriented technique:
  - an intuitively appealing design approach:
  - natural **objects** (such as employees, pay-roll-register, etc.) occurring in a problem are first identified.
- **Relationships** among objects:
  - such as composition, reference, and inheritance are determined.
- Each object essentially acts as
  - a data hiding (or data abstraction) entity.

- Object-Oriented Techniques have gained wide acceptance:
  - **Simplicity**
  - **Reuse possibilities**
  - **Lower development time and cost**
  - **More robust code**
  - **Easy maintenance**

Differences between the exploratory style and modern software development practices

- **Use of Life Cycle Models**
- Software is developed through several well-defined stages:
  - requirements analysis and specification,
  - design,
  - coding,
  - testing, etc.
- Emphasis has shifted
  - from **error correction to error prevention**.
- Modern practices emphasize:
  - detection of errors as close to their point of introduction as possible.

- In exploratory style,
  - errors are detected only during testing,
- Now,
  - focus is on **detecting as many errors as possible in each phase** of development.
- In exploratory style,
  - coding is synonymous with program development.
- Now,
  - **coding is considered only a small part of program** development effort.
- A lot of effort and attention is now being paid to:
  - **requirements specification**.
- Also, now there is a distinct design phase:
  - **standard design techniques** are being used.

- During all stages of development process:
  - **Periodic reviews** are being carried out
- Software testing has become systematic:
  - **standard testing techniques** are available.
- There is **better visibility of design and code**:
  - visibility means production of good quality, consistent and standard documents.
  - In the past, very little attention was being given to producing good quality and consistent documents.
  - We will see later that increased visibility makes software project management easier.

- Because of good **documentation**:
  - **fault diagnosis and maintenance are smoother now.**
- Several **metrics** are being used:
  - help in **software project management, quality assurance**, etc.
- Projects are being thoroughly **planned**:
  - estimation,
  - scheduling,
  - monitoring mechanisms.
- Use of **CASE tools**.

Thank you!