Complexity

Complexity of Software development:

1. **Changing requirement Engineering and Changing requirement issue:**

   Customers specify requirements many times in between development process. It involves three things or areas:

   - Maintenance: Correcting errors
   - Evolution: Changing requirements
   - Preservation: Way to keep ancient and decaying piece of software or managing and updating the existing software.

   “In agile, they welcome all the changes by the client at any time even if the whole process has to be redesigned. They support team development.”

   Refactoring and re-engineering are two of the most important methodologies in Software engineering, which cope with changing requirements.

   “Changing the external interface (visible to the user) and internal remains same.”

   If the total productivity of a person is 2 and there are 5 peoples, then total productivity of the group cannot be 10, it will always be less than 10.

2. **Flexibility :**

   - Flexibility in software engineering refers to the capacity of a software system or process to adapt to changing requirements, technologies, and circumstances without requiring significant redesign or rework. This is a critical quality in today's rapidly evolving technological landscape, where new requirements, user needs, and business goals can emerge over time.
   - It includes the following applied in Software engineering:
     - Scalability
     - Reusability
     - Modularity
     - Abstraction

    - Configuration Management
    - Adaptive Architecture
    - Continuous Integration and Continuous Deployment (CI/CD)
    - Version Control
    - Agile and Iterative Development
    - Documentation

Flexibility is incredibly seductive property, which makes the development process labor-intensive.

💡 Code consists of three things:
a. State -> data-items
b. Behavior -> functions
c. Identity -> variable-name

3. **Problems of charactering the behavior of discrete systems :**
    - No linear relation between size and complexity of project. If the size increase 2 times, then complexity increase is uncertain
    - There is Combinatorial explosion of possibilities (states) in large projects. It means that number of variables are very large, and they may take many values (states) in such large projects which is very difficult to handle.
    - As project size increases, logical complexity of program increases.
    - Example:
        - Three while loops, if program size doubles, then there will not be 6 whiles loops, there will be less than that, let’s say 5 of 6, but nested.

System of systems :

- External events may corrupt your state of the system. What we call as systems of system (SOS).

- As a programmer, we must care about the systems in which the system under development or will be developed will be placed by the client.
- **We are not looking into one system; we are looking at all the related systems. So, we must firstly study SOS and then study the architect of the design of our system. SOS is very important.**

Five Attributes of Complex System

1. **Hierarchy:**
    - Core modules: Most basic and important activities are carried out by these modules which are essential for functioning of system.
    - Non-core modules: They are increasing the functionality of core modules.
    - Hierarchy is good in large systems.
2. **Primitives :**

    Primitives are up to observer/designer/developer. Depends on user to user.
3. **Decomposition/separation of concerns :**
    - Decomposable parts are not necessarily complete independent. There will be loose coupling between them.
    - We try to keep connectivity between them very weak. This helps in separation of concerns. **(Concerns means problems and in object-oriented design, problems are objects, hence concern refers to objects).**
4. **Reusability :**
5. **Stable intermediate forms :**
    - Design should be such that various stable intermediate software created at each step.
    - Means we are moving from one software to another in building the whole complete software.
    - **These software must be usable, deliverable and identifiable.**

- Release core system, only having the functionalities and then study feedback and then add functionalities.
- Complex evolved from simple.
    - A large, complex software from complete basics === NO
    - A large, complex software from other software == YES

Complexity handling :

- Two approaches:
    - Decomposition
    - Abstraction

Canonical form of system means using object-oriented design.
Objects and process :

Objects and processes

- We have to identify the objects acting or operating on these functions.
- Functional oriented design: All functions are doing same related job in one module.

  Object oriented design: All functions in one object, and related abstractions in modules. (Objects are abstractions).

There are three analysis and design approaches :

- Top-down structured design, also called algorithmic design
- Data-driven design
- Object-oriented design

💡 Economy of Expressions also known as brevity of conciseness

💡 Algorithms and procedures are implementations of functions

💡 Data driven design is used earlier, but to make Object-oriented design, we need
to analyze our system as all the three view-points (multiple models).
