


1. Introduction to Software Engg.

- 
- 1. Scope & Necessity of Software Engineering***
 - 2. Software Crisis***
 - 3. Program vs. Software Product***
 - 4. Structured Programming***
 - 5. Object Oriented Programming***
 - 6. Exploratory Programming Style***
 - 7. Modern Programming Style***

1. *Scope & Necessity of SE:*

- ❖ **Engineering approach** for software development
- ❖ Systematic collection of **past experience**
- ❖ No **scientific basis** or **theoretical proof**
- ❖ Absence lead to difficulty in developing large & complex programs

Problem Complexity:

- ❖ Increases exponentially with size
- ❖ Techniques to reduce problem complexity:
 - **Abstraction**: simplified by omitting **irrelevant details**
 - **Decomposition**: divided into **smaller problems** with **minimum interactions** among them

2. Software Crisis:

- ❖ **Huge expenses** on software purchases than hardware
- ❖ Not on account of **increased features**
- ❖ **Causes:** larger problem sizes, lack of adequate training, increasing skill shortage, and low productivity improvements
 - Software products are difficult to alter, debug, and enhance; use resources non-optimally; often fail to meet the user requirements; far from being reliable; frequently crash; and often delivered late

Solutions to Software Crisis:

- ❖ Spread of **software engineering practices** among the engineers
- ❖ Further advancements to the **software engineering discipline** itself

3. Program vs. Software Product:

Program	Software Product
Small in size and has limited functionality	Extremely large and complex
Developed by individuals for personal use	Most users are not involved with the development
A single developer is involved	A large number of developers are involved

Program vs. Software Product: (Con..)

Program	Software Product
User interface may not be very important	User interface must be carefully designed and implemented
Very little documentation is expected	Must be well documented
Individual development style based on intuition is used	Must be developed using the accepted software engineering principles

4. Structured Programming:

- ❖ Uses three types of program constructs i.e. **selection, sequence and iteration**
- ❖ Avoids **unstructured control flows** by restricting the use of GOTO (JUMP) statements
- ❖ Consists of a **well partitioned** set of modules
- ❖ Uses **single-entry, single-exit** program constructs

Structured Programming: (Con..)

- ❖ Easier to read, understand, debug and maintain
- ❖ Requires less effort and time for development
- ❖ Usually fewer errors are made

5. OO Programming:

- ❖ **Natural objects** are identified first
- ❖ **Relationships** among objects are determined
- ❖ Objects act as a **data hiding entity**
- ❖ **Advantages:** simpler (due to abstraction), better understandability, better problem decomposition, code and design reuse, and easier maintenance

6. *Exploratory Programming Style:*

- ❖ Based on **error correction**
- ❖ Not **cost-effective**, since errors are detected only during the final product testing
- ❖ Not suitable for large and complex programs
- ❖ Coding is synonymous to software development
- ❖ No **consistent documents**

7. Modern Programming Style:

- ❖ Follows **software engineering principles**
- ❖ Based on **error prevention**
- ❖ Develops software through several **well-defined phases**

Modern Programming Style: (Con..)

- ❖ **Phase containment of errors:** detect and fix errors in the same phase in which they occur
- ❖ Coding is a small part of the overall **software development activities**
- ❖ Attention is given to **requirements specification**

Modern Programming Style: (Con..2)

- ❖ **Systematic and standard testing techniques**
- ❖ **Consistent and standard documents** give better visibility of design and code
- ❖ **Project management** ensures well planning, monitoring and controlling project activities