

Software Engineering



Introduction

■ Software

- Software is more than just a program code
- A program is an executable code, which serves some computational purpose
- Software is considered to be collection of executable programming code, associated libraries and documentations
- Software, when made for a specific requirement is called software product

■ Engineering

- All about developing products using well defined principles, methods and procedures



What is Software Engineering?

- Software engineering is an engineering branch associated with development of software product using well-defined scientific principles, methods and procedures
- The application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software
- Establishment and use of sound engineering principles in order to obtain software that is reliable and work efficiently on real machines
- The process of developing a software product using software engineering principles and methods



Software Types

- System Software
- Application Software
- Engineering/Scientific Software
- Embedded Software
- AI Software
- Legacy Software
- Web/Mobile Software



Why SE is important

- Helps to build complex systems in timely manner
- Ensures high quality of software
- Imposes discipline to work
- Minimizes software cost
- Decreases time



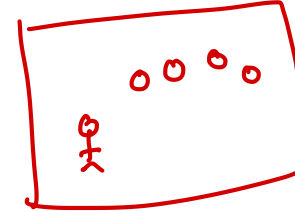
Software Evolution

▪ S-type (static-type)

- Works strictly according to the pre-defined specifications and solutions
- Solution and method to achieve it can be understood immediately before coding starts
- Least subjected to the changes
- E.g. Calculator program for mathematical computation

▪ P-type (practical-type)

- Software with a collection of different procedures
- Is defined by exactly what procedures can do
- The specification can be described and solutions are not obvious instantly
- E.g. Gaming software



▪ E-type (embedded-type)

- Works closely as the requirement of real-world environment
- Has a high degree of evolution as there are various changes in laws, taxes etc. in the real world
- E.g. online trading software

tax ST. \Rightarrow GST



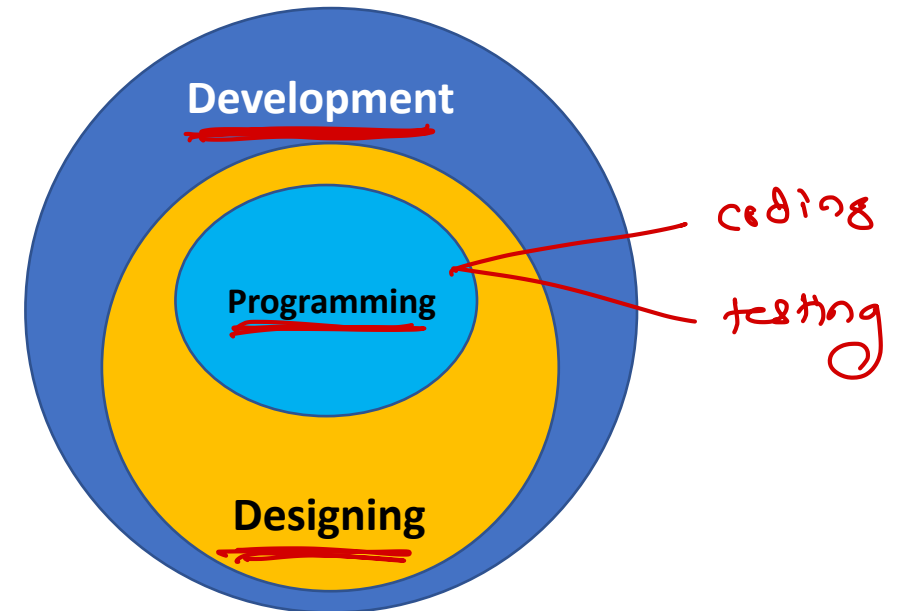
E-Type software evolution laws

- **Continuing change**
 - An E-type software system must continue to adapt to the real world changes, else it becomes progressively less useful
- **Increasing complexity**
 - As software evolves, its complexity tends to increase unless work is done to maintain or reduce it
- **Conservation of familiarity**
 - The familiarity with the software or the knowledge about how it was developed, why was it developed in that particular manner etc. must be retained at any cost, to implement the changes in the system
- **Continuing growth**
 - In order for an E-type system intended to resolve some business problem, its size of implementing the changes grows according to the lifestyle changes of the business
- **Reducing quality**
 - An E-type software system declines in quality unless rigorously maintained and adapted to a changing operational environment
- **Feedback systems**
 - The E-type software systems constitute multi-loop, multi-level feedback systems and must be treated as such to be successfully modified or improved.
- **Self-regulation**
 - E-type system evolution processes are self-regulating with the distribution of product and process measures close to normal.
- **Organizational stability**
 - The average effective global activity rate in an evolving E-type system is invariant over the lifetime of the product.



Software Paradigms

- Refer to the methods and steps, which are taken while designing the software
- There are many methods proposed and are in work today, but we need to see where in the software engineering these paradigms stand
- These can be combined into various categories, though each of them is contained in one another
- Consists of
 - Requirement gathering
 - Software design
 - Programming



Characteristics of good software

- A software product can be judged by what it offers and how well it can be used
- Well-engineered and crafted software is expected to have the following characteristics
 - Operational
 - Transactional
 - Maintenance



Operational

- This tells us how well software works in operations
- Can be measured on:
 - ✓ ■ Budget
 - ✓ ■ Usability
 - ✓ ■ Efficiently
 - ✓ ■ Correctness
 - ✓ ■ Functionality
 - ✓ ■ Dependability
 - ✓ ■ Security
 - ✓ ■ Safety



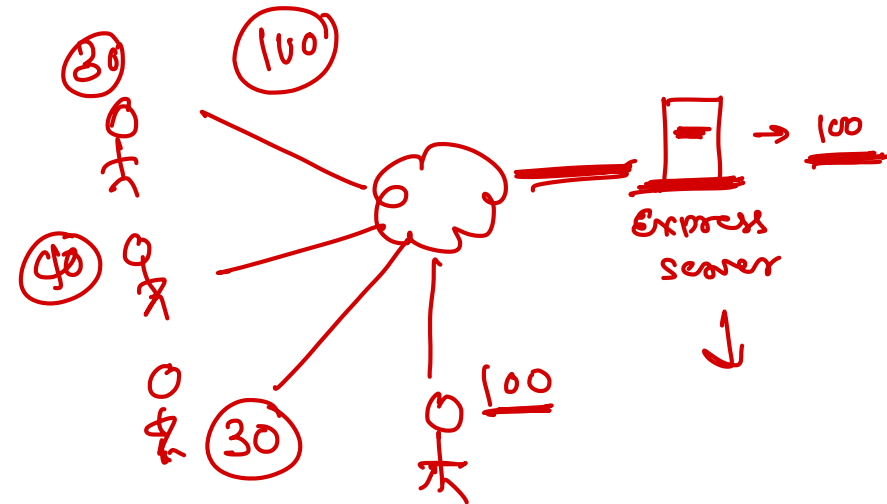
Transitional

- This aspect is important when the software is moved from one platform to another
- Can be measured on
 - ✓ ■ Portability
 - ✓ ■ Interoperability
 - ✓ ■ Reusability
 - ✓ ■ Adaptability



Maintenance

- Briefs about how well a software has the capabilities to maintain itself in the ever-changing environment
- Can be measured on
 - ✓ Modularity
 - ✓ Maintainability
 - ✓ Flexibility
 - ✓ Scalability
 - ↳ High Availability

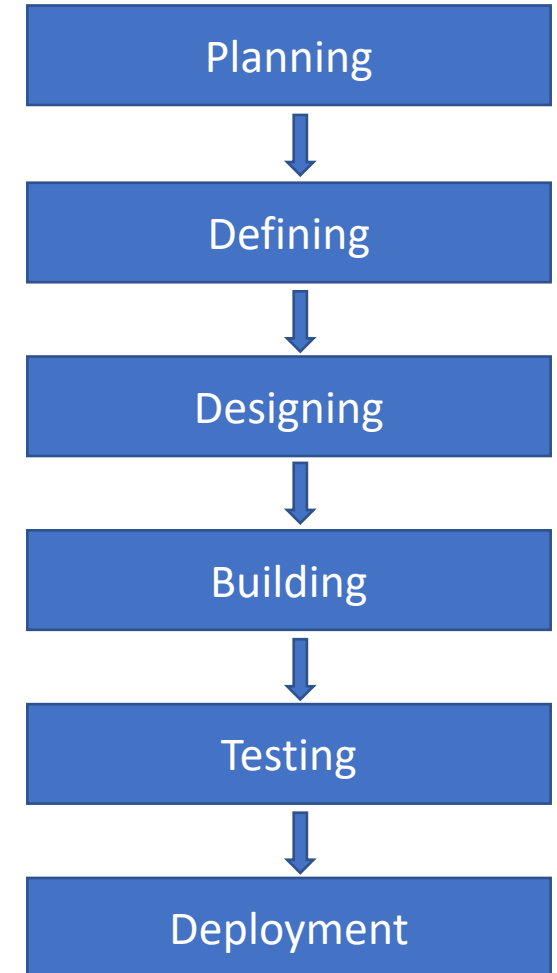


SDLC



Overview

- Also called as Software Development Process
- Is a well-defined, structured sequence of stages in software engineering to develop the intended software product
- Is a framework defining tasks performed at each step in the software development process
- Aims to produce a high-quality software that
 - meets or exceeds customer expectations
 - reaches completion within times and cost estimates
- Consists of detailed plan (stages) of describing how to develop, test, deploy and maintain the software product



Planning and Requirement Analysis

- The most important and fundamental stage in SDLC
- It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry
- This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas
- Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage
- The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks



Defining Requirements

- Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts
- This is done through an **SRS (Software Requirement Specification)** document which consists of all the product requirements to be designed and developed during the project life cycle



Designing the Product Architecture

- SRS is the reference for product architects to come out with the best architecture for the product to be developed
- Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification
- This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product
- A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party modules (if any)
- The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS



Building or Developing the Product

- In this stage of SDLC the actual development starts and the product is built
- The programming code is generated as per DDS during this stage
- If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle
- Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers, etc. are used to generate the code
- Different high level programming languages such as C, C++, Java, PHP etc. are used for coding
- The programming language is chosen with respect to the type of software being developed



Testing the Product

- This stage is usually a subset of all the stages as in the modern SDLC models
- The testing activities are mostly involved in all the stages of SDLC
- However, this stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS



Deployment in the Market and Maintenance

- Once the product is tested and ready to be deployed it is released formally in the appropriate market
- Sometimes product deployment happens in stages as per the business strategy of that organization
- The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing)
- Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment
- After the product is released in the market, its maintenance is done for the existing customer base



SDLC Models

- There are various software development life cycle models defined and designed which are followed during the software development process
- Also referred as Software Development Process Models
- Models
 - ✓ ■ Waterfall Model
 - ✓ ■ Iterative Model
 - ✓ ■ Spiral Model
 - ✓ ■ V-Model
 - ✓ ■ Big Bang Model
 - ✓ ■ Agile Model



Waterfall Model

■ Requirement Specification

■ Design

■ Implementation

■ Verification and Testing

■ Deployment

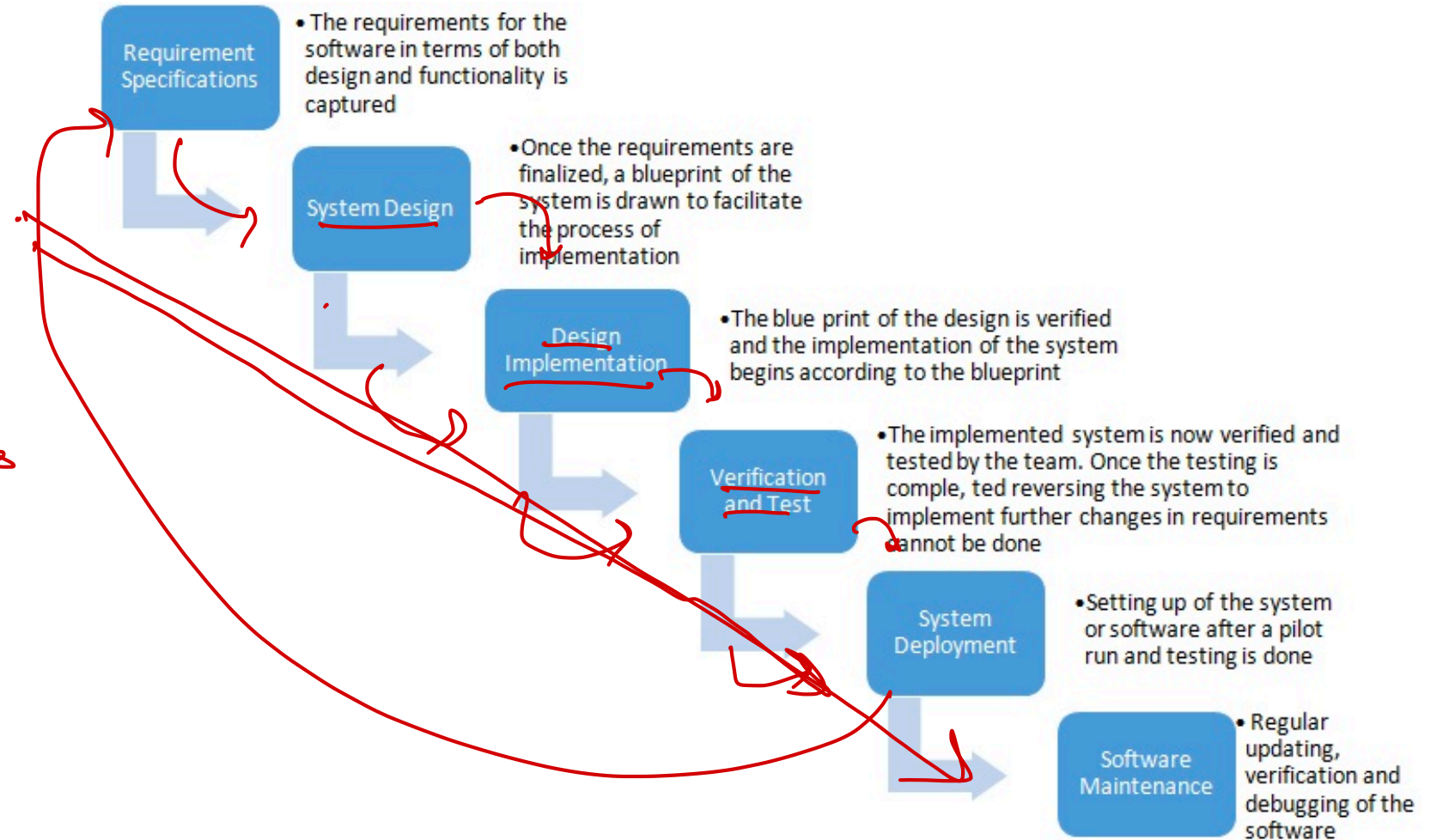
■ Maintenance

cloud (SE)

docker

kubernetes

ecommerce
\$\$, 10 months



e-commerce

- order
- user
- order
- profile

Watermodel

e-commerce

Iterative

- order →
- user →
- product → ① add / ② edit, ③ search...
- profile → ④ delete

Iterative Model

- implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented

