Introduction to Software Engineering

Notes by Mannan Ul Haq

What is Software?

Software refers to a set of instructions, programs, or data that enable a computer system to perform specific tasks or functions.

Software consists of three main components:

Instruction: (Computer program) when executed provide desire features, functions and performance.

Data Structures: That enables the program to efficiently manipulate information.

Documentation: That describes the operations and use of programs.

Software Applications:

1. **System Software**:
   - **Definition**: System software manages computer hardware and provides a platform for running application software.
   - **Examples**: Operating systems like Windows, macOS, Linux, and device drivers.
2. **Application Software**:
   - **Definition**: Application software is designed to perform specific tasks or functions for end-users.
   - **Examples**: Word processors (e.g., Microsoft Word), spreadsheet programs (e.g., Microsoft Excel), web browsers (e.g., Google Chrome), and media players (e.g., VLC Media Player).
3. **Engineering/Scientific Software**:
   - **Definition**: Software used in engineering and scientific fields for analysis, simulation, design, and modeling.
   - **Examples**: CAD (Computer-Aided Design) software like AutoCAD, and simulation software like MATLAB.
4. **Embedded Software**:
   - **Definition**: Embedded software is programmed to perform specific functions within embedded systems, typically found in electronic devices.
   - **Examples**: Digital cameras, automotive control systems, and industrial machinery, printers, sensors.
5. **Product-line Software**:
   - **Definition**: Product-line software refers to a family of related software products developed from a common set of components.

- **Examples**: Software like Microsoft Office (includes Word, Excel, PowerPoint), and Adobe Creative Cloud (includes Photoshop, Illustrator, Premiere Pro).

6. **Web Applications (WebApps)**:
   - **Definition**: Web applications are software applications accessed and operated through web browsers over a network, typically the internet.
   - **Examples**: Online banking systems, e-commerce platforms (e.g., Amazon, eBay), and social media platforms (e.g., Facebook, Twitter).
7. **AI Software**:
   - **Definition**: AI (Artificial Intelligence) software use human-like intelligence to perform tasks such as reasoning, learning, problem-solving, and decision-making.
   - **Examples**: CHAT-GPT, Machine learning algorithms, natural language processing (NLP) systems, computer vision applications, and virtual assistants (e.g., Siri, Alexa).

Software—New Categories

1. **Open World Computing**: This involves distributed computing, where tasks are spread across multiple computers.
2. **Ubiquitous Computing**: This uses wireless networks to integrate computing into everyday life.
3. **Netsourcing**: This means using the Web as a powerful computing resource.
4. **Open Source**: This refers to software whose source code is freely available to the public. It's great for collaboration but can also pose some risks.

Software Engineering:

Software engineering is a disciplined approach to the design, development, testing, and maintenance of software systems. It applies engineering principles and methodologies to ensure the quality, reliability, and efficiency of software products.

Process Frame-Work:

Framework Activities:

1. **Communication**:

   Establishing effective communication channels among clients, developers, testers, and managers, to ensure clear understanding of requirements.

2. **Planning**:

   Creating a comprehensive project plan outlining objectives, scope, and resources to guide the development process.

3. **Modeling**:
   - **Analysis of Requirements**: Gathering, analyzing, and documenting user needs and system requirements to define the scope, features, and constraints of the software system.

- **Design**: Developing a detailed blueprint or architecture for the software system based on the requirements.

4. **Construction**:
   - **Code Generation:** Implementing the design by writing code in programming languages.
   - **Testing**: Verifying and validating the software to ensure it meets the specified requirements and functions correctly under various conditions.
5. **Deployment**:

   Releasing the software into production environments, installing and configuring it on target system.

Umbrella Activities:

1. **Software Project Management**:

   Planning, organizing, and controlling software projects to ensure they're completed on time, within budget, and with the desired quality.
2. **Formal Technical Reviews**:

   Systematic evaluations of software documents and code to find and fix problems early in the development process.
3. **Software Quality Assurance**:

   Implementing processes to ensure that software meets specified requirements and standards throughout the development lifecycle.
4. **Software Configuration Management**:

   Managing changes to software configurations, ensuring that versions are controlled and consistent across the development team.
5. **Work Product Preparation and Production**:

   Creating and maintaining documentation, reports, and deliverables associated with software development activities.
6. **Reusability Management**:

   Identifying and promoting the reuse of software components, libraries, and frameworks to improve efficiency and consistency in development.
7. **Measurement**:

   Defining and collecting metrics to assess the progress, quality, and performance of software projects and processes.
8. **Risk Management**:

   Identifying, analyzing, and mitigating risks that could impact the success of software projects.

The Essence of Practice:

Polya Suggests:

1. Understand the Problem:

- Who has the stack in the solution of problem? Who are stack holders?
- What data, functions, features are required to solve problem.
- Can analysis model be created.
- Can the problem be solved by putting them or solve into smaller parts.

2. Plan the Solution:

- Have you seen the problem before?
- Have you solve that problem before?
- Can subproblems be defined?
- Can you represent solution graphically.

3. Carry out the Plan:

- Is the solution according to the plan?
- Is each part of the solution is correct?

4. Examine Result:

- Possible to test each part of solution?
- Does the product is according to the requirements, provided by the user?

Hooker's General Principles:

1: The Reason It All Exists

2: Keep It Simple, Stupid!

3: Maintain the Vision

4: What You Produce, Others Will Consume

5: Be Open to the Future

6: Plan Ahead for Reuse

7: Think!

Legacy Software

Why Must It Change?

1. **Adapting to New Environments**: Software needs to be updated to work with new computing environments or technologies.
2. **Enhancing for New Requirements**: Software must be improved to meet new business needs.

3. **Extending for Interoperability**: Software must be modified to work with newer systems or databases.
4. **Re-architecting for Networks**: Software needs to be redesigned to function effectively in a network environment.