INTRODUCTION TO SOFTWARE ENGINEERING

Lesson 01 – What is software engineering?

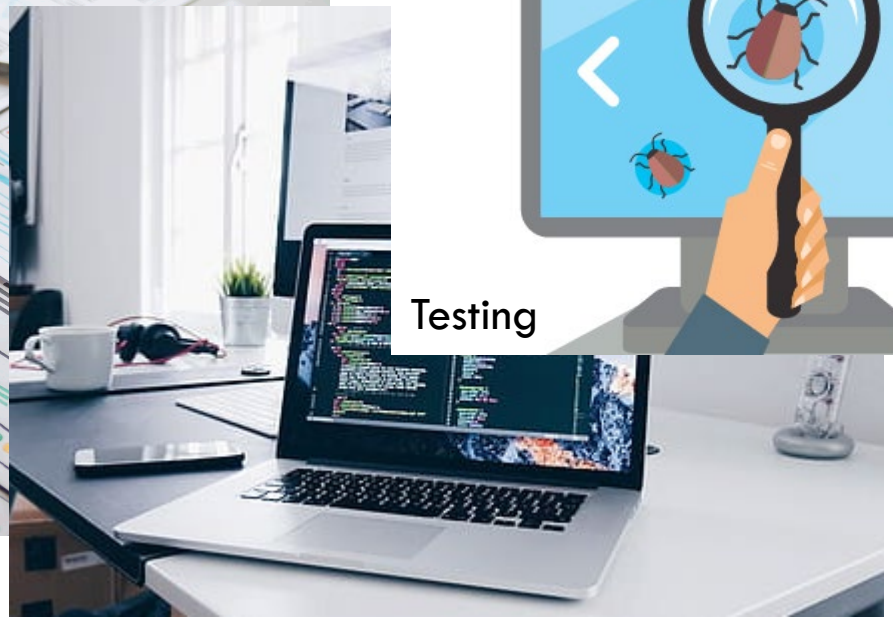
OUTLINE

1. Overview of software engineering
2. Software development life cycle (SDLC)
3. Key terminology and concepts

1. OVERVIEW OF SOFTWARE ENGINEERING

Software engineering involves the design, development, testing, and maintenance of software applications. Software engineers use engineering principles and programming languages to create software solutions for users.

Design



Develop

Testing

2. SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

A framework that describes the activities performed at each stage of a software development project.

Common phases of SDLC:

1. Initiation
2. Requirements definition
3. Design (functional and technical)
4. Development (or construction)
5. Testing (or verification)
6. Deployment (or implementation)
7. Maintenance & Review

2.1. INITIATION

Activities should be taken into accounts are:

1. Project objectives & Scope (the bigger project size the longer time to realize)
2. Preliminary survey & feasibility
    1. Technical (do we have skill like this kind of project?)
    2. Economic (profitable from it in term of budget and company image?)
    3. Operational (available time and resources to make it?)
3. Project proposal and schedule (suggest solutions and timeline)
4. Identify assumptions & constraints (total cost and criteria to make it work)

2.2. REQUIREMENTS DEFINITION

Focus on what the system will do for the users. It should include:

1. Problem/Opportunity definition (problems are reasons to build system to solve it)
2. Analyze current system (are there places can process can be digitalized?)
3. Focus on decisions and related information needs
   (choose a best digitalize process and required data)
4. Define business functionality (features of the system)
5. Plan for training, user acceptance
   (target users, features, and make sure end-user accepts and use the system)

2.3. DESIGN (FUNCTIONAL AND TECHNICAL)

Functional design:

1. Focus on business needs (usability, reliability)
2. Logical design
    1. Inputs (data dictionary and sample data)
    2. Outputs (forms and reports)
    3. Presentation (user interface designs)
    4. Processes (navigations, interactions, validations)
    5. Databases (structured data storage)
    6. Personnel (users of the system)

Technical design:

1. Finalize architecture and acquire hardware (Protocols, RAM size, ...)
2. Complete technical definition of data access and other system components (data security, module communication protocol)

2.4. DEVELOPMENT (OR CONSTRUCTION)

Related to writing codes to construction a system software based on required features and designs. It includes:

1. Analysis, codes, unit testing (test each main functions)
2. Develop test plans (sample data, use cases)
3. Revise schedule, plan and costs (make sure we know the upfront project status)

2.5. TESTING (OR VERIFICATION)

1. Program Testing
    1. Structured walkthrough
    2. Code inspection
    3. Unit test
    4. Pairs testing
2. Verification, stress, user and security testing

2.6. DEPLOYMENT (OR IMPLEMENTATION)

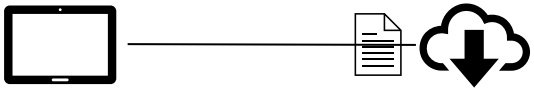
1. Cut-over
    1. Parallel conversion (cross OS test)
    2. Direct cut-over (replace/upgrade new one)
    3. Pilot conversion (small group of clients' test)
    4. Phased conversion (split system to smaller modules and test one by one)
2. User training

2.7. MAINTENANCE & REVIEW

1. Post-implementation audit
   1. Ends - information requirements (information, performance)
   2. Means – process (result)
2. Maintenance (correcting bugs & scheduled maintenance)
3. Enhancement (adding functionality)

3. KEY TERMINOLOGY AND CONCEPTS

Latency: the amount of time required for a single data to be delivered successfully. (3 seconds to download)



Throughput: the measure of amount of data transmitted successfully in a system, in a certain amount of time. (max 20 requests per second)

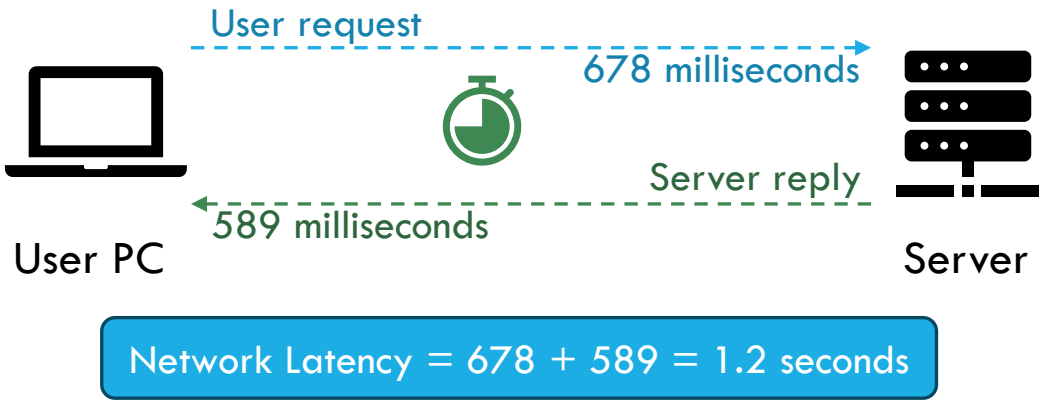
Availability: the percentage of time the system is up and working for the needs. (99.98% accessible)

Redundancy: duplicated entities aim to scale up the system and reduce over all down-time. (3 servers to share workloads)

Time: a measure of sequences of events happening which is measured here in seconds in its SI unit. (SI – Système d'internationale d'unités)

3.1. LATENCY

The time taken to response from server to client machine is called Latency. It is measured in milliseconds.



3.1. LATENCY

Components Affecting Latency:

1. **Packet Size** (Smaller the packet chunk size faster the transmission and lower the latency.)
2. **Packet Loss** (Transmit huge packets of various sizes in medium losses to very few packets.)
3. **Medium of transmission** (Optical fiber is the fastest way of transmission.)
4. **Distance between Nodes** (Poor signal increase latency.)
5. **Signal strength** (Good signal strength reduces latency.)
6. **Storage delays** (Stored info in DB, fetching it requires very little time which increase latency.)

3.2. THROUGHPUT

The amount of data transmitted successfully in a system, in a certain amount of time.

Throughput is considered as how much data is transmitted successfully over a period of time (bits per second or bps).

Ex. measure total number of bits received within one second

3.3. AVAILABILITY

The percentage of time the system is up and serve the user is called Availability.

$$\text{Availability} = \frac{\text{Uptime}}{(\text{Uptime} + \text{downtime})}$$

Example: A bank system served the users for a year (365 days * 24 hours). Every 2 months the system is downed for maintenance for 3 hours. What is Availability rate of this bank system?

(Uptime + downtime) = $365 * 24$ hours = 8760 hours

$Downtime = 3\frac{12}{2}\ hours = 18\ hours$

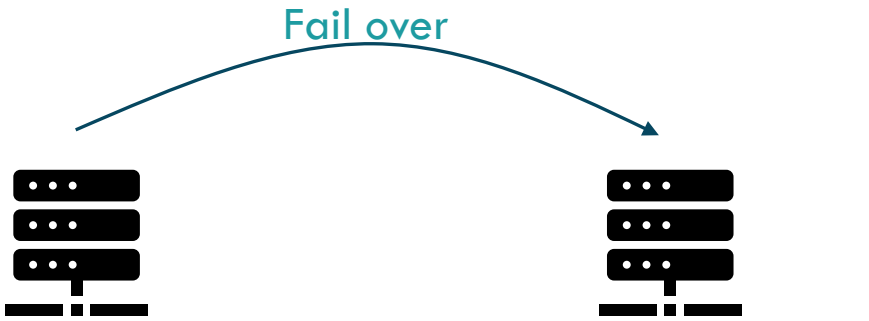
(1 year = 12 months, system down 3 hours for every 2 months)

$Uptime = 8760 - 18\ hours = 8742\ hours$

$$\boldsymbol{Availability = \frac{8742}{8760} = 99.79\%}$$

3.4. REDUNDANCY

Concepts of duplicating entities to scale up the system and to cut down overall downtime.



Total Availability: $\mathbf{1 - (1 - 0.9979)(1 - 0.9979) = 99.99956\%}$

3.5. TIME

Time is a measure of sequences of events happening which is measured here in seconds in its SI unit.

It is measured using a clock which is of two types:

- **Physical Clock**: responsible for the time between systems.
- **Logical Clock**: responsible for the time within a system.

Different SI unit challenges:

- Drifting (delayed caused by different SI unit)
- Accounting for propagation delay (Ex. assign IP address to a new domain name)
- Accounting form processing delay (Ex. A process wait for result of another process)