INTRODUCTION TO SOFTWARE DESIGN

EGCO343 SOFTWARE DESIGN

KANAT POOLSAWASD
DEPARTMENT OF COMPUTER ENGINEERING
MAHIDOL UNIVERSITY

WHAT IS SOFTWARE ?

- Software is a set of instructions, data or programs used to operate computers and execute specific tasks.

- Software is a generic term used to refer to applications, scripts and programs that run on a device. It can be thought of as the variable part of a computer, while hardware is the invariable part.

TYPES OF SOFTWARE (1)

- The two main categories of software are **application software** and **system software**.
- An **application is software** that fulfills a specific need or performs tasks.
- **System software** is designed to run a computer's hardware and provides a platform for applications to run on top of.

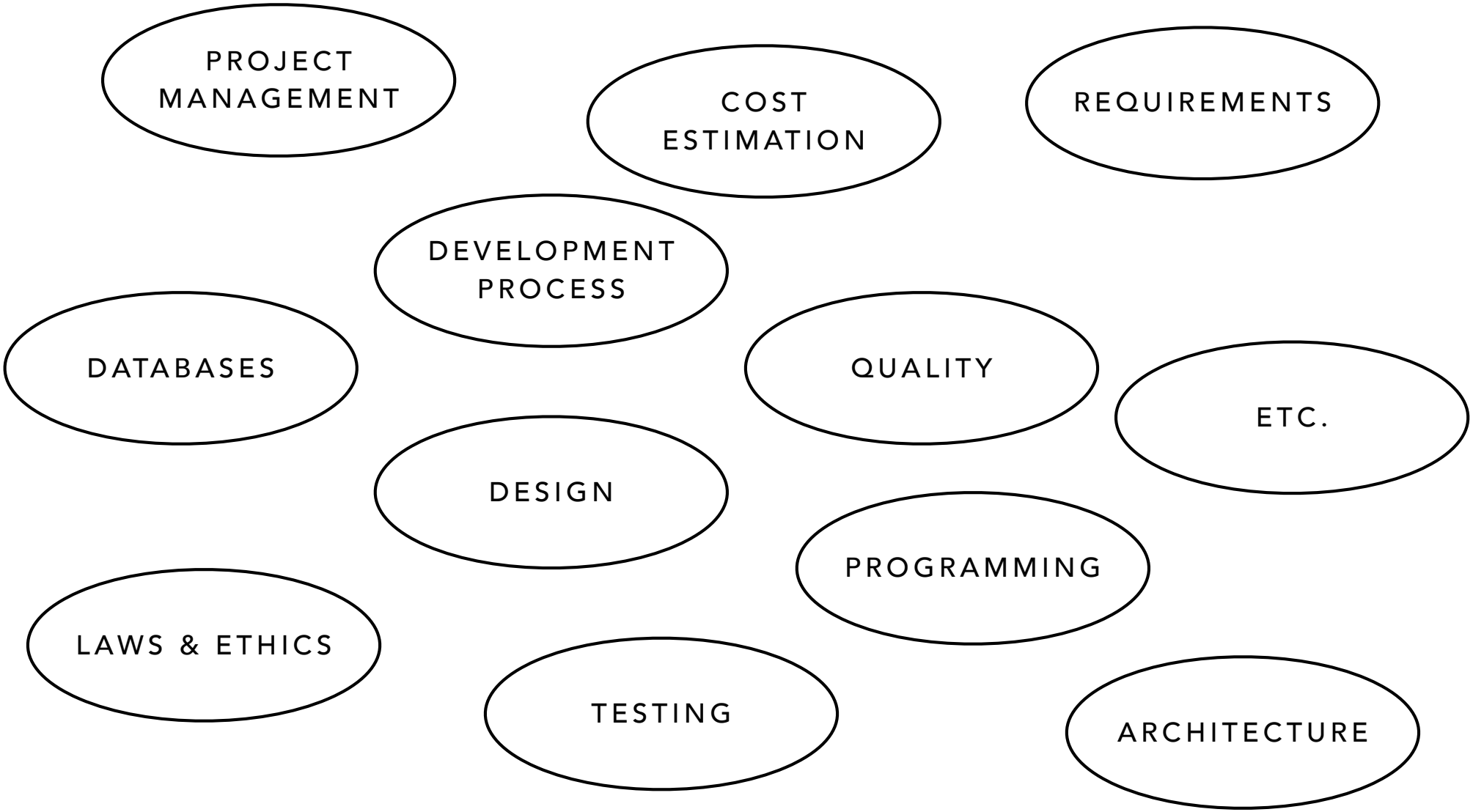
TYPES OF SOFTWARE (2)

- Actually, there are many types of software. It depends on what kind of classification.
  - Desktop Application vs. Web Application
  - TPS (Transaction Processing System) vs. Realtime System
  - General-Purpose vs. Customized (Bespoke)
  - General Computing System vs. Embedded System
  - Driver vs. Firmware
  - Safety-Critical System
  - Etc.

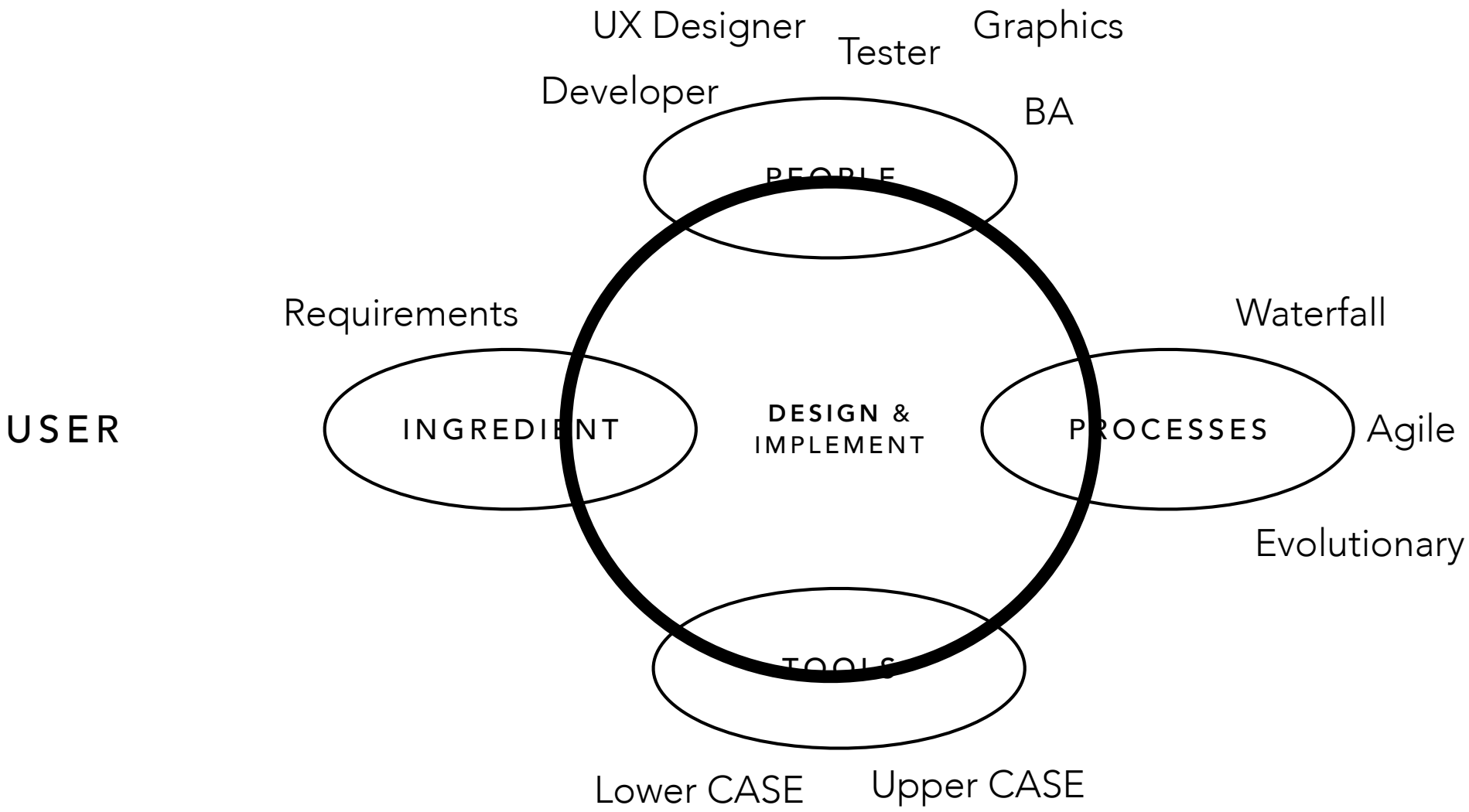
SOFTWARE STACK

https://www.techtarget.com/searchapparchitecture/definition/software

TOPICS IN SOFTWARE DEVELOPMENT



TOPICS IN SOFTWARE DEVELOPMENT



SOFTWARE DEVELOPMENT ROLES

- Project Manager
- System Architect
- Business Analyst (BA)
- UX Designer
- Programmer (Developer)
  - Front-End / Back-End
- Database Designer / Database Administrator
- Tester
- Customer (User)
- Etc.

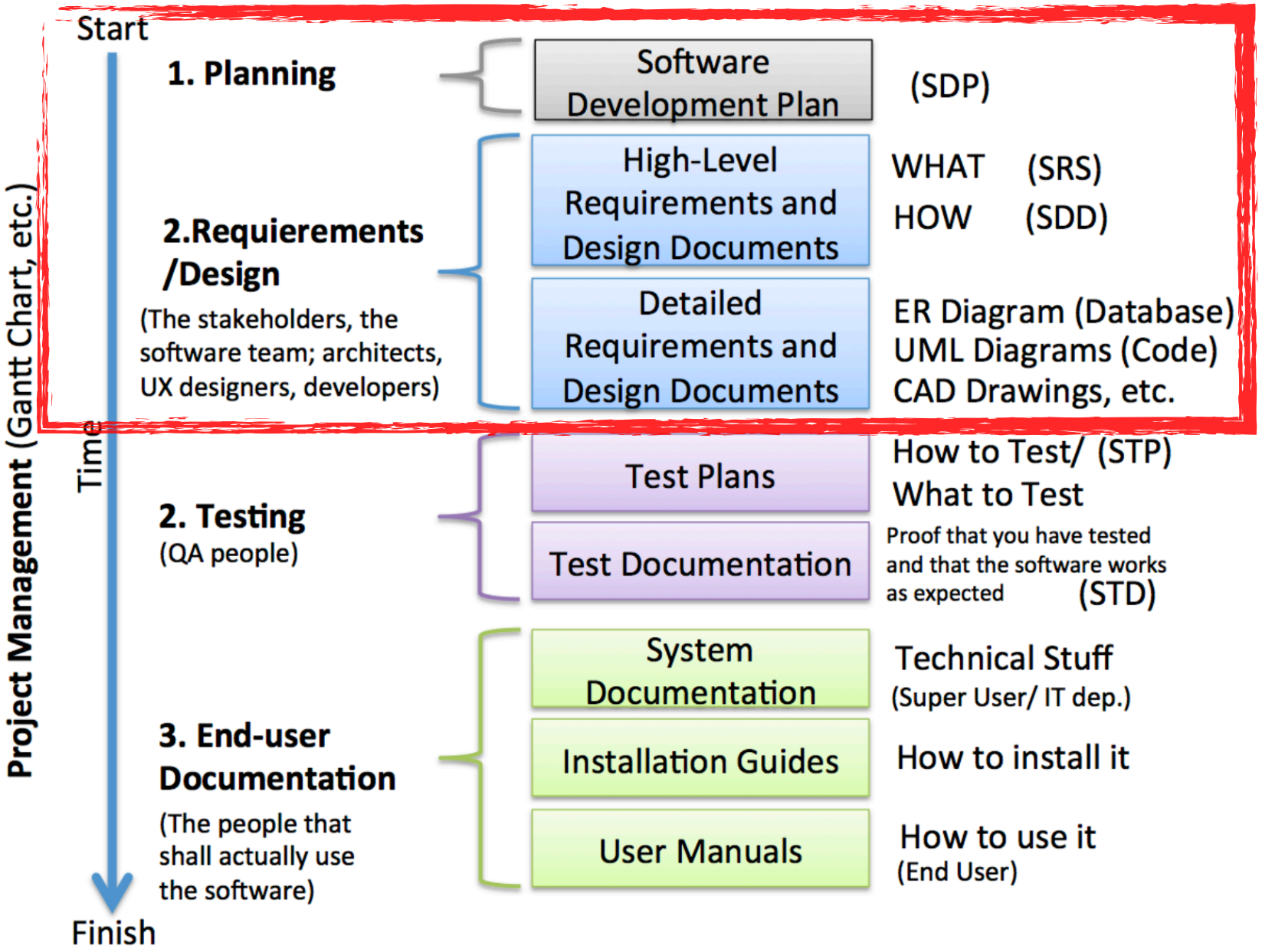
SOFTWARE DEVELOPMENT TOOLBOX

- Computer (PC)
- Programming Language
- IDE (Integrated Development Environment)
- Framework
- SCC (Source Code Control) Tool
- ALM (Application Lifecycle Management) Tool
- CI/CD (Continuous Integration/Continuous Deployment)
- and, knowledge about Software Engineering (SE)

CASE (COMPUTER-AIDED SOFTWARE ENGINEERING) TOOL

- Software systems that are intended to provide automated support for software process activities.
- CASE systems are often used for method support.
- **Upper-CASE:** Tools to support the early process activities of requirements and design.
- **Lower-CASE:** Tools to support later activities such as programming, debugging and testing.

SOFTWARE DEVELOPMENT PROCESS



SOFTWARE DEVELOPMENT PROCESS

Analysis

(Logical)

Software Requirements

Analysis Tools

- DFD
- UML

Design

(Physical)

Software Architecture

Front-End

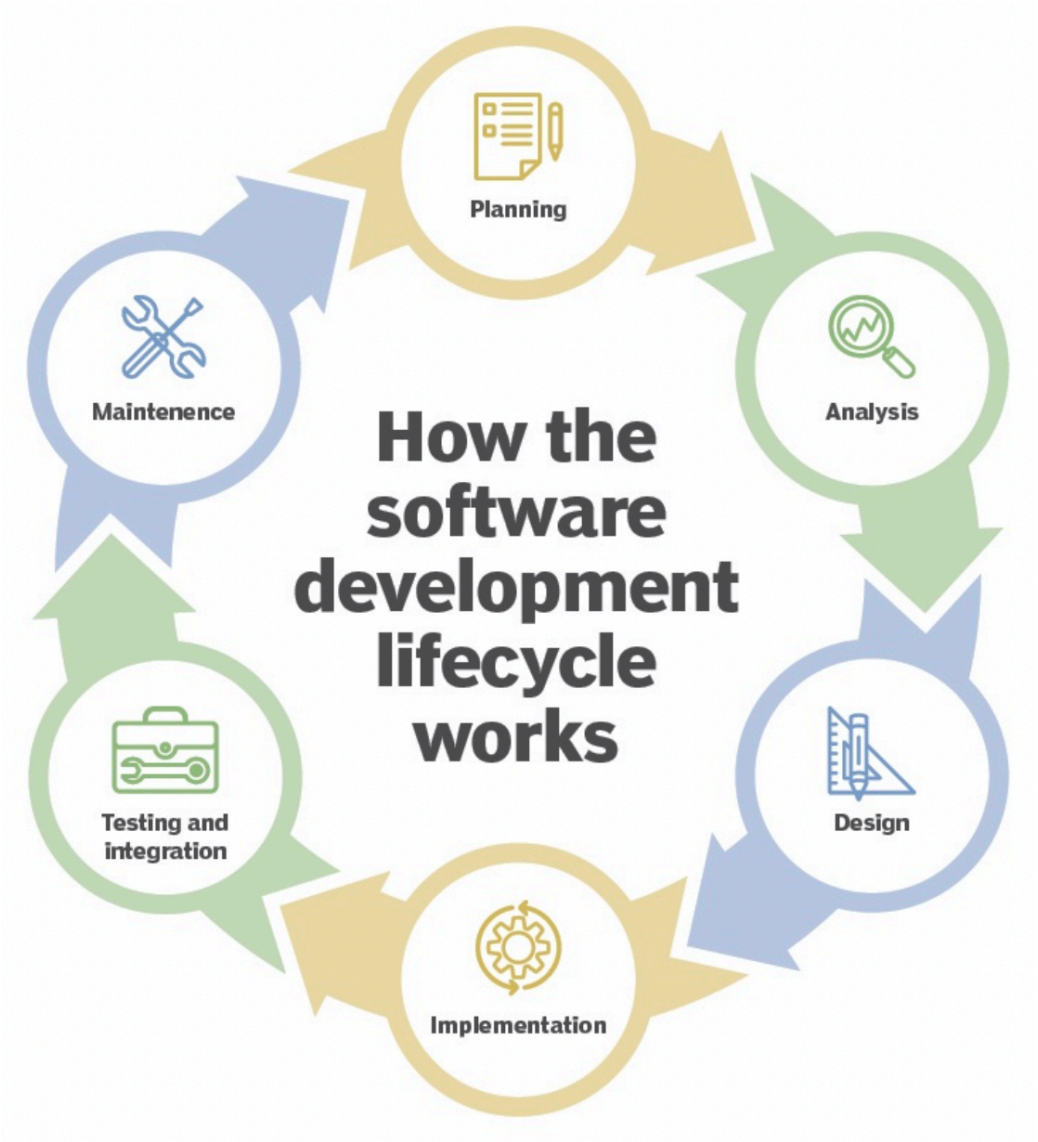
- UX/UI

Back-End

- Services (API)

Model (Database)

SOFTWARE DEVELOPMENT LIFE CYCLE



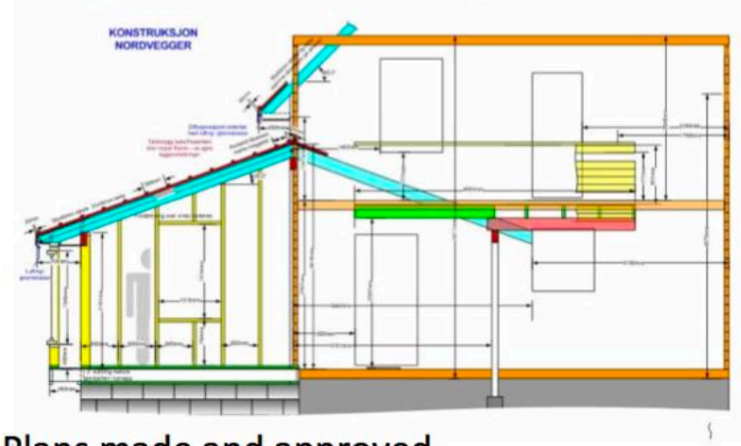
https://www.techtarget.com/searchapparchitecture/definition/software

SYSTEM DEVELOPMENT METHODOLOGIES

- A methodology is a formalized approach to implementing the SDLC (i.e., it is a list of steps and deliverables).
- The category of systems development methodologies:
  - Waterfall
  - Incremental Development
  - Evolutionary Development
  - Agile Development
  - Etc.

SOFTWARE ITERATIONS AND RELEASES

Requirements/Design



Plans made and approved

Alpha

Foundation finished, building structure started

Beta

Building structure finished,
Inside work on track

RC

Furniture, Flowers and
small adjustments missing

RTM

Ready for Sale or Move in

WHAT IS SOFTWARE DESIGN ?

- Software design is a process to transform user **requirements** into some suitable form, which helps the programmer in software coding and **implementation**.

- Software design is the first step in SDLC (Software Design Life Cycle), which moves the concentration from problem domain to solution domain. It tries to specify how to fulfill the requirements mentioned in SRS.

OBJECTIVES OF SOFTWARE DESIGN

- **Correctness**: Software design should be correct as per requirement.
- **Completeness**: The design should have all components like data structures, modules, and external interfaces, etc.
- **Efficiency**: Resources should be used efficiently by the program.
- **Flexibility**: Able to modify on changing needs.
- **Consistency**: There should not be any inconsistency in the design.
- **Maintainability**: The design should be so simple so that it can be easily maintainable by other designers.

SOFTWARE DESIGN LEVELS

- Software design yields three levels of results:
  - **Architectural Design** - It identifies the software as a system with many components interacting with each other.
  - **High-level Design** - The high-level design breaks the single entity-multiple component concept of architectural design into less-abstracted view of sub-systems and modules and depicts their interaction with each other.
  - **Detailed Design** - Detailed design deals with the implementation part of what is seen as a system and its sub-systems in the previous two designs.

MODULARIZATION

- Modularization is a technique to divide a software system into multiple discrete and independent modules, which are expected to be capable of carrying out task(s) independently.
- These modules may work as basic constructs for the entire software. Designers tend to design modules such that they can be executed and/or compiled separately and independently.

COUPLING AND COHESION

- When a software program is modularised, its tasks are divided into several modules based on some characteristics.
- As we know, modules are set of instructions put together in order to achieve some tasks.
- There are measures by which the **quality of a design** of modules and their interaction among them can be measured.
- These measures are called coupling and cohesion.

CONCURRENCY

- Concurrency is implemented by splitting the software into multiple independent units of execution, like modules and executing them in parallel.
- In other words, concurrency provides capability to the software to execute more than one part of code in parallel to each other.

WHAT ARE THE COSTS OF SYSTEM?

- Roughly 60% of costs are development costs, 40% are testing costs. For custom software, evolution costs often exceed development costs.
- Costs vary depending on the type of system being developed and the requirements of system attributes such as performance and system reliability.
- Distribution of costs depends on the development model that is used.

WHAT ARE THE KEY CHALLENGES ?

- **Heterogeneity:** Developing techniques for building software that can cope with heterogeneous platforms and execution environments.
- **Delivery:** Developing techniques that lead to faster delivery of software.
- **Trust:** Developing techniques that demonstrate that software can be trusted by its users.

PROFESSIONAL AND ETHICAL RESPONSIBILITY

- System analyst involves wider responsibilities than simply the application of technical skills.
- System analyst must behave in an honest and ethically responsible way if they are to be respected as professionals.
- Ethical behaviour is more than simply upholding the law.

ISSUES OF PROFESSIONAL RESPONSIBILITY (1)

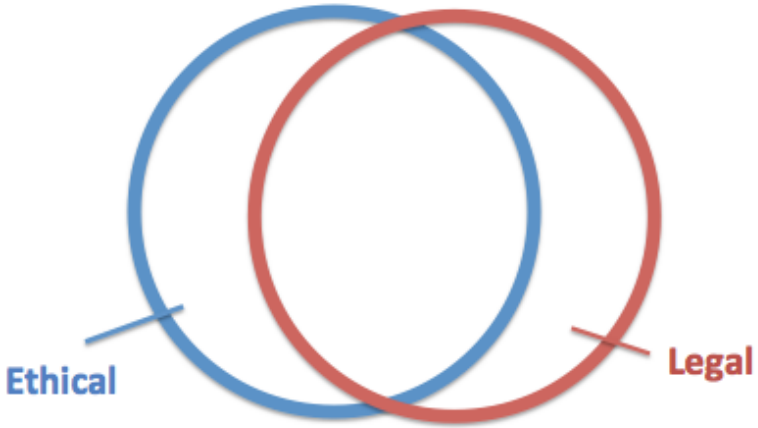
- **Confidentiality**
  Engineers should normally respect the confidentiality of their employers or clients irrespective of whether or not a formal confidentiality agreement has been signed.
- **Competence**
  Engineers should not misrepresent their level of competence. They should not knowingly accept work which is out-with their competence.

ISSUES OF PROFESSIONAL RESPONSIBILITY (2)

- **Intellectual property rights**
  Engineers should be aware of local laws governing the use of intellectual property such as patents, copyright, etc. They should be careful to ensure that the intellectual property of employers and clients is protected.
- **Computer misuse**
  Software engineers should not use their technical skills to misuse other people's computers. Computer misuse ranges from relatively trivial (game playing on an employer's machine, say) to extremely serious (dissemination of viruses).

ETHICAL ≠ LEGAL

- In very basic terms, ethics are about right and wrong, while laws involve civil or criminal penalties, such as lawsuits, fines, or imprisonment. There is a lot of overlap between what's wrong (unethical) and what can subject you to civil or criminal penalties (illegal), but that doesn't meant that the two terms are equivalent.



CONCLUSION

- The first half covers **Software Analysis**, and the second half focuses on **Software Design**.
- Software Analysis begins with gathering **software requirements** and **analyzing** them using tools such as DFD and UML.
- The second half focuses on Software Design, taking into account the **software architecture** as well as **front-end and back-end** operations.
- The Front-End focuses on **UX/UI design**, while the Back- End handles **services** delivered through APIs.