COMP1531

🐶 Software Engineering
7.1 - Requirements - Overview

In this lecture

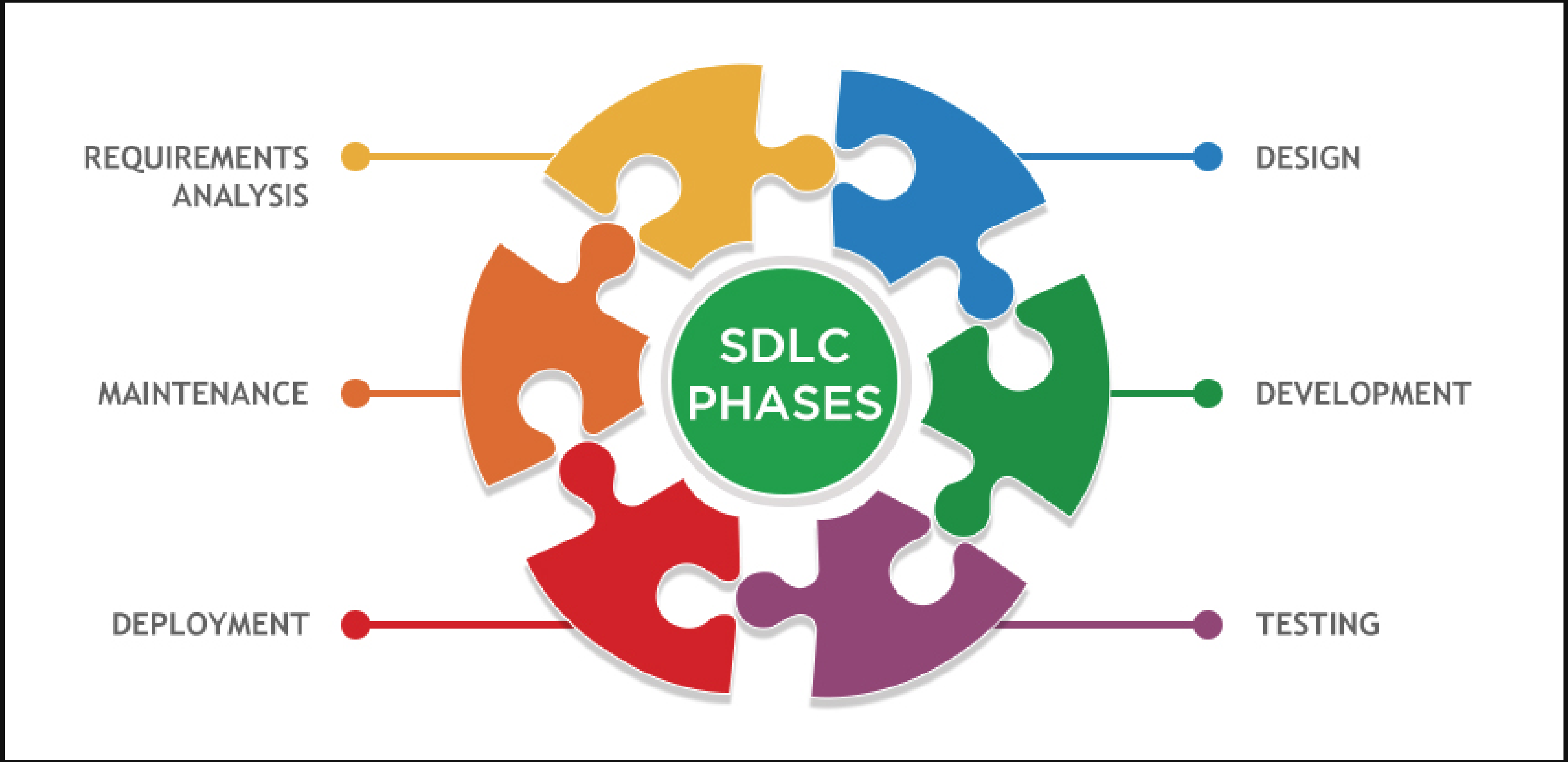
Why?

- The most important part of building a system is figuring out what you need to do

What?

- Requirements
- Functional V Non-functional requirements
- Requirements Engineering

SDLC


REQUIREMENTS ANALYSIS
DESIGN
MAINTENANCE
SDLC PHASES
DEVELOPMENT
DEPLOYMENT
TESTING

Requirements

IEEE defines a requirement as:

A condition or capability needed by a user to solve a problem or achieve an objective

We would also describe requirements as:

- Agreement of work to be completed by all stakeholders
- Descriptions and constraints of a proposed system

“The hardest single part of building a software system is deciding what to build. No part of the work so cripples the resulting systems if done wrong”
(Brooks, 1987)

Requirements

What are some examples of requirements?

Can we come up with some requirements that are set out by the COMP1531 course?

Functional v Non-Functional

Functional requirements specify a specific capability/service that the system should provide. It's *what* the system does.

Non-functional requirements place a constraint on *how* the system can achieve that. Typically this is a performance characteristic.

Great reading on the topic

Functional v Non-Functional

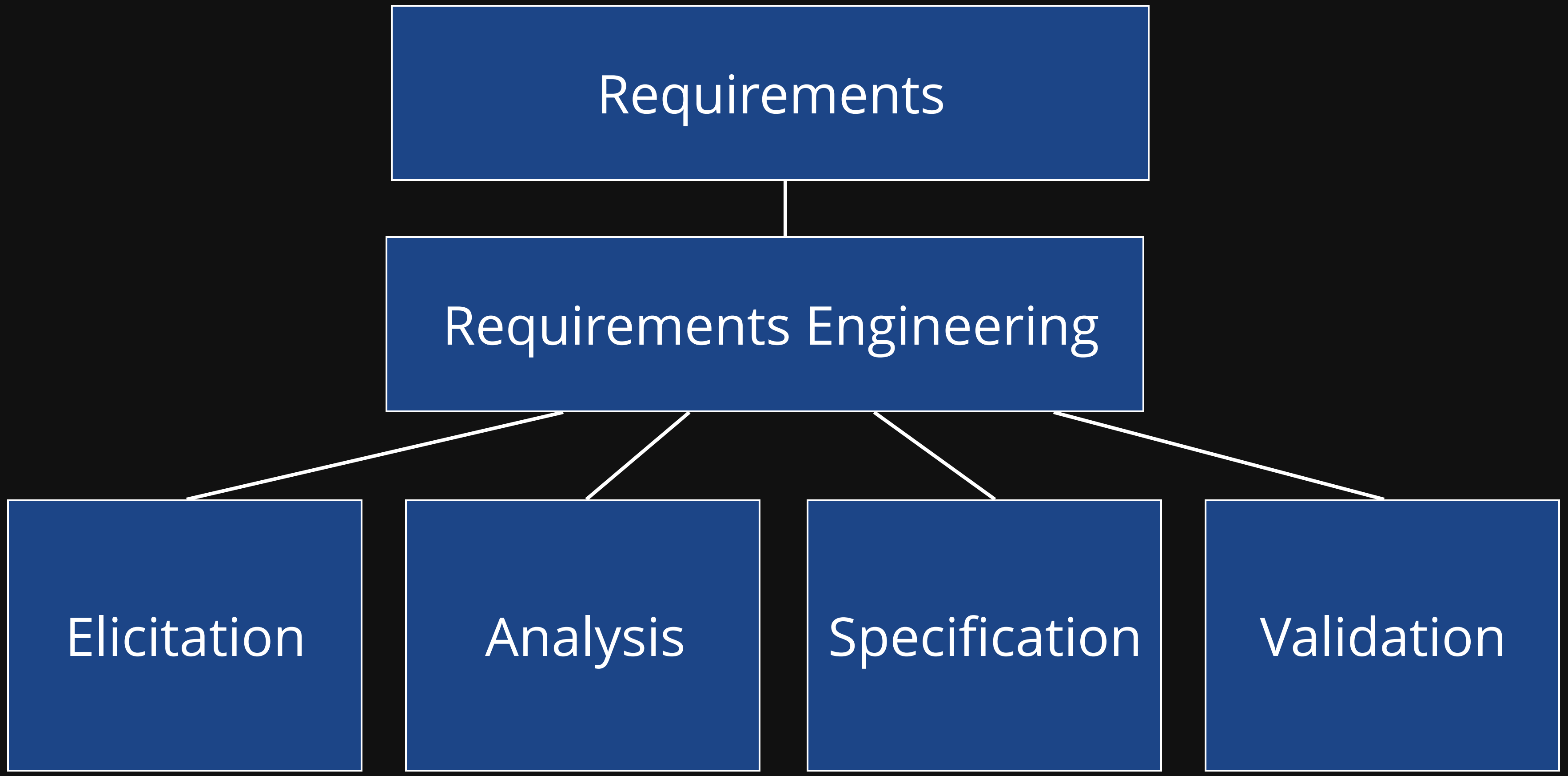
For example:

Functional: The system must send a notification to all users whenever there is a new post, or someone comments on an existing post

Non-functional: The system must send emails no later than 30 minutes after from such an activity

Deriving Requirements

Requirements don't just appear in thin air. We have to derive them, and to do that we apply the process of requirements engineering.



Requirements Engineering

Requirements Engineering is:

- A **set of activities** focused on identifying the purpose and goal of a software system
- A **negotiation process** where stakeholders agree on what they want. Stakeholders include:
  - End user(s)
  - Client(s) (often businesses)
  - Design team(s)

Requirements Engineering

Requirements engineering often follows a logical process across 4 steps:

1. Elicitation of raw requirements from stakeholders
2. Analysis of requirements
3. Formal specification of requirements
4. Validation of requirements

RE | Step 1 | Elicitation

Questions and discovery

- Market Research
- Interviews with Stakeholders
- Focus groups
- Asking questions "What if? What is?"

RE | Step 2 | Analysis

Building the picture

- Identify dependencies, conflicts, risks
- Establish relative priorities
- Usually done through:
  - User stories (discussed today)
  - Use cases (discussed next week)

RE | Step 3 | Specification

Refining the picture

- Establishing the right sense of granularity
  - There is no perfect way to granulate
- Often the stage of breaking up into functional and non-functional
- E.G. Try and granulate "The system shall keep the door locked at all times, unless instructed otherwise by an authorised user. When the lock is disarmed, a countdown shall be initiated at the end of which the lock shall be automatically armed (if still disarmed)"

RE | Step 4 | Validation

Checking you haven't gotten lost

Going back to stakeholders and ensuring requirements are correct

Challenges during RE?

What are some challenges we may face while engaging in Requirements engineering?

- Requirements sometimes only understood after design/build has begun
- Clients/customers sometimes don't know what they want
- Clients/customers sometimes change their mind
- Developers might not understand the subject domain
- Limited access to stake holders
- Jumping into details or solutions too early (XY problem)

Optional (Monorail requirements)

https://tharunka.arc.unsw.edu.au/src-approves-plans-for-monorail/

Feedback