1 General

1.1 What is Software?

Software can describe a lot of things, some examples include:

- Executable programs
- Configuration files
- System documentation
- User documentation
- Support environment
- etc.

In general Software can be divided into three categories:

- Application Software
  - Interacts directly with the end user
  - General purpose software (To be used in other applications: Word processing, image processing, etc.)
  - Customized Software (Software specifically for a specific purpose: CAD, IDE, BIM, etc.)
- System Level Software
  - Does not interact directly with the end user
  - Responsible for keeping systems running (Operating System, firmware, drivers, etc.)
- Software as a Service (SaaS)
  - Runs on a server
  - Indirectly accessed via client (browser, remote shell, etc.)

Furthermore, some characteristics of Software include:

- Software does not wear out, its environment does
  - Software is subject to continous change in hardware, needs to be able to adapt
  - Software should be able to support new requirements, use cases etc.
- Software often lives longer than anticipanted
  - Almost impossible to know use cases in advance as it can be in use for years or even decades (Excel used in biology geneology → lead to unexpected behaviour)
- Software properties are hard to measure
  - How does the code relate to software quality?
  - How do we measure progress?
  - How do we measure resilience?

1.2 Software Engineering

Typically a software is designed to solve different needs of different groups involved in the development of the software.

- Customer / Client
  - Often the person / organisation that'll pay for the development
  - Sets a budget, timeframe, requirements etc.
  - → Requirements analysis
- User
  - Usually the person / organisation that'll use the software
  - Defines what the software is used for and subsequently what requirements this sets
  - → Use Case Analysis
- Manufacturer
  - Usually the person / organisation that'll design and develop the software
  - Is concerned with how to build the software in a way that satisfies the customers and users
  - → Domain Modelling, Architecture, Quality Assurance, Design Practice, Verification
- Maintainer
  - Usually the person / organisation that'll maintain the software during its lifetime
  - Responsible for maintenance of the software and updates to make it usuable for new demands and requirements
  - → Maintenance and Evolution

After all these aspects are consideres the software system is the built with the specific requirements in budget and time.

There are quite a few problems that can happen with Software:

- Unexpected Errors:
  - Few errors are obvious
  - Most of them are near impossible to test for and detect (Algorithmic error, arithmetic overflow)
  - Often go undetected for a long time as they're usually the result of very specifc inputs for complex computations
- Altough errors can occur, as long as they do no violate the requirements they are not considered errors:
  - INABIAF: It's not a bug, it's a feature
- Most errors are caused by missing verification, validation or documentation.
  - This usually indicates an insufficient match between requirements and implementation

Errors can also occur as a result of social aspects:

- Insufficient validation
- Inadequete Specification
- Constantly changing requirements
- Insuffciently trained software engineers
- Managment with lacking grasp on software development
- Unsuitable methods, languages, tools etc.