Object Oriented Software Engineering

Dagmawi Lemma (PhD)
Assistant Professor, AAU
courses.dag@gmail.com
https://sites.google.com/site/dagmawi2012/

1. Introduction

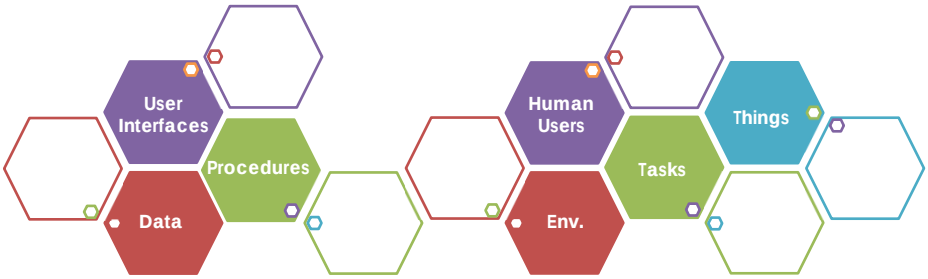
Software Engineering

From Programming to Engineering

- Originally computers were considered just as a machine that is meant to be **configured** to solve a particular problem.
- Later the required configuration revealed itself as programming.
- As computers become a general purpose machine, the importance of software becomes indispensable
  - Also the development process began to involve several stakeholders who have diversified, and even conflicting, requirements

Software Elements

- Software in general refers to the program that runs in the computer.
- But nowadays, it is more than that, and each of the element will have implication on how we think about software and how we develop it, as well as while considering the platform/environment on/in which it runs.



From Programming to Eng'…

- Furthermore, as software become core part of computer and many business, organizations, institutes, etc. are relying on computers, thus faults in software are no longer tolerable
  - Often the fault was rooted to developing software with erroneous requirements:
    - Missing user requirements
    - Wrong understanding (assumption) of user requirements
    - Implementing undesired requirements (feature)
- The situation leads to what is known as **software crisis** and amplified the need for engineering approach.

Software Crisis

- The time of intense and the looming danger and challenges that resulted from lack of effective software development leads to crisis and the situation was considered as **software crisis**
- Software crisis was a term coined in used to express the difficulty of developing **useful, effective,** and **efficient** software within specific time and budget; and manifested in several ways:
  - Projects running over-budget
  - Projects running over-time
  - Software was very inefficient
  - Software was of low quality
  - Software often did not meet requirements
  - Projects were unmanageable and code difficult to maintain
  - Software was never delivered
- As a result of the challenges faced in developing software, which is efficient and effectively solve the user's problem, it was noted that software development need to be done as a process that follows engineering approach – hence software engineering evolve as a discipline.

What is Software Engineering?

- To begin with, what is engineering?
  - It is about problem solving
    - Note that you solve problem in various ways
      - E.g., discussion, supply the need, etc.
    - So how is engineering solve a problem?
      - Examples
        - » A problem of accessing city leads to constructing road
        - » A problem of shelter leads to building houses
        - » A problem lifting heavy objects to a height leads to construct crane,
        - » A problem of organizing data leads to database system, etc.
  - **Engineering solves a problem by delivering a product.**

What is SE? *(continued)*

- Engineering is a **process** of developing a **product** by *applying mathematical and scientific methods* with the intention of **solving a well defined problem**.
- Software Engineering is a **process** of developing a **software** by *applying mathematical and scientific methods* with the intention of solving **problem** by improving efficiency in the use of computer while performing some **predefined user task**.

The Problem

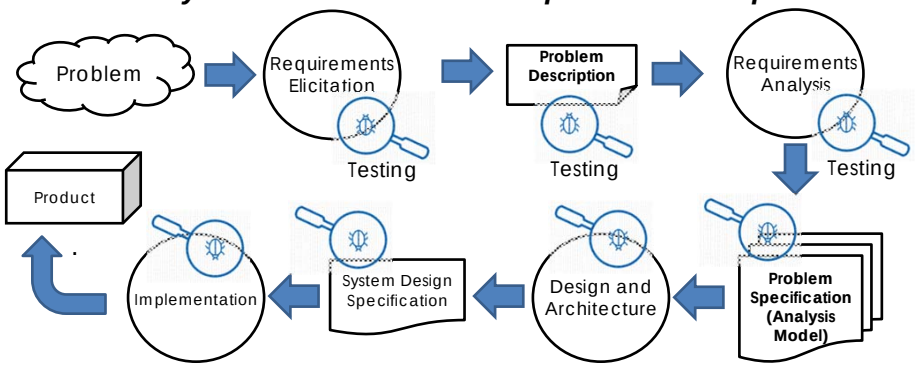
- Problem to be solved through engineering is often related to:
  - Inability to do some task due to lack of capacity, and limitation of human beings in performing their task or some activity.
    - E.g., calculating complex mathematical equation, remembering booking detail, etc.
  - Aspiration to efficient and better way of accomplishing task
    - E.g., performing routine job diligently, supporting large number of customers cost effectively, etc.
- Hence the problem to be solved through **engineering aims to improve human capability and efficiency while users (individuals or organizations) perform various activity.**

The Problem (*continued*)

- Most software development projects are initiated:
  - with the intention of solving users' problem to support users perform their task/s
    - which are not humanly possible
      - E.g., it is humanly not possible to remember all the expiry dates of drugs shelved in a drugstore
    - efficiently and effectively by increasing productivity, safety, satisfaction, etc.

The Process

- In engineering, a process is a series of interrelated tasks, which are needed to develop/construct a certain ***product.***
- These series of tasks, together, transform inputs into a given output leading to the final output (i.e., the product) or the tasks ***may meant to maintain and fix problems in the product***.

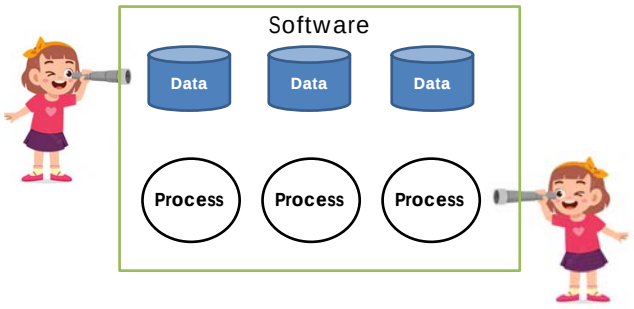


Software Development Approaches

- Two possible views we can have about software;
  - **The data**: that is expected to be used and/or maintained by the software.
  - **The function or procedure**: that process the data as per some business logic and changes the state of the system.
- Accordingly, the software development approach might follow two possible approaches that considers the data and the function in different perspective.
- A system is a collection of various elements those abide to some rule and works/collaborate together to achieve some goal.
  - Software, being a system, what are the elements in it?

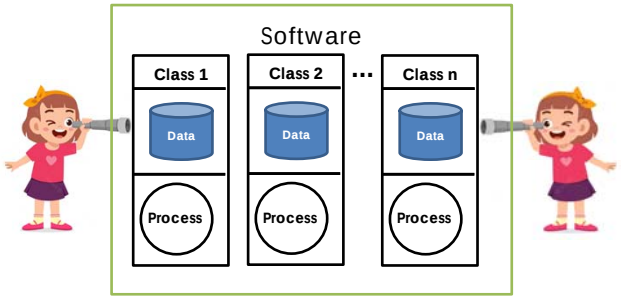
Software Development Approaches

- Perspective 1: software is a collection of functions (procedures) those work together to achieve some common goal.
  - Thus the data and the function shall be studied and designed separately.
  - Having such consideration, we can follow procedural (structured) software development approach.



Software Development Approaches

- Perspective `2: software is a collection of objects where the object contains a function that manipulates data of the respective object; these objects interact to one another achieve some goal.
  - Thus the data and the function that are contained in each object shall be analyzed and designed by identing the objects which contain those data and function.
  - Having such consideration, we can follow object oriented software development approach.
  - Technically, an object is instance of a class
  -



Software Development Approaches

- Therefore, broadly we classify the software development approach into two:
  - Structured/Procedural
    - The process, the role and the data are separately studied (analyzed, designed, implemented, and tested)
  - Objects
    - Assumes that software is collection of objects interacting one another to solve the problem.
    - The objects collaborate
    - Objects are instance of class that comprises of the process and data, while roles are considered as external players known as actors.

The Water Fall

"Way back when"

- Royce, W. W. (1970, August). Managing the development of large software systems. In *proceedings of IEEE WESCON* (Vol. 26, No. 8).
  - essential steps common to all computer program development
    - Analysis and Coding
    - System requirement, software requirement, analysis, design, coding, testing, operation
    - Iterative relationship between successive phases

Types of Process Model

- Three generic process models (Sommerville **2007**)
  - The waterfall model
  - Evolutionary development
  - Component-based SE
- Or a process model can be linear, incremental, agile

Types of Process Model

- In principle, the result of each phase in any of the process model is one ore more documents that are approved.
  - In practice there is no clear cut rather the phases overlap
  - The cost of producing and approving documents as well as the number of iterations are expensive
    - Freezing parts of the development ☹
    - Problems are left for later resolutions
    - Ignore the problem
    - Solve the problem by programming around (without documentation)

Types of Process Model

- Waterfall model
  - Logically ordered as in SDLC
  - It is however practical if the requirements are well understood and unlikely to change radically during the process
- But what about the documentation?
  - Documentation is often rigorous and bulky
- Other approaches also seems to practice the waterfall but by considering small portion of the entire system
  - Hah! The water fall seems to appear as sub model
  - No! the SDLC is reflected rather

Types of Process Model

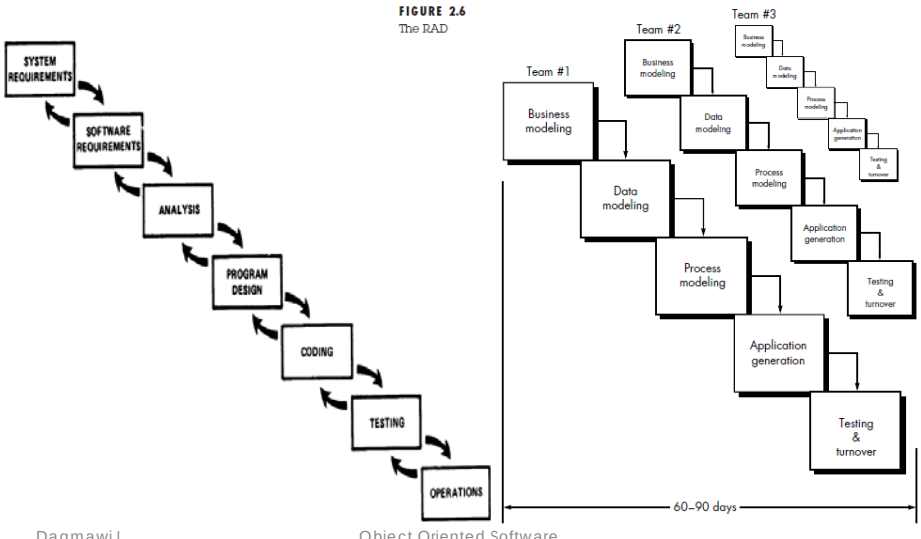
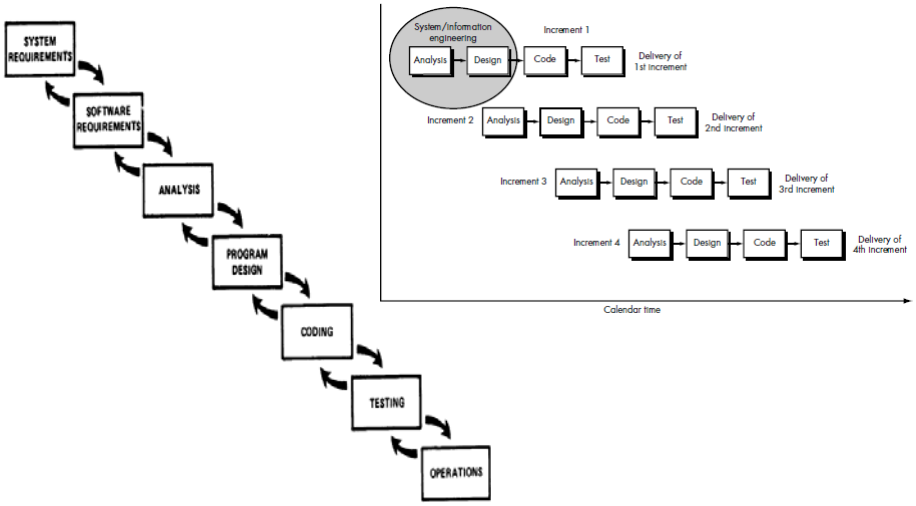


FIGURE 2.6
The RAD

Types of Process Model



Types of Process Model

- If we still have the sequence of the activities in RAD, Incremental, Spiral, etc. as in the waterfall, why do we consider them as different process model?
  - REMARK:
    - Process descriptions is more than the sequence of activities: it may also include principles to be followed and description about:
      - Products
      - Roles
      - Pre- and post-conditions,
      - tools and techniques etc.

The Paradigm shift

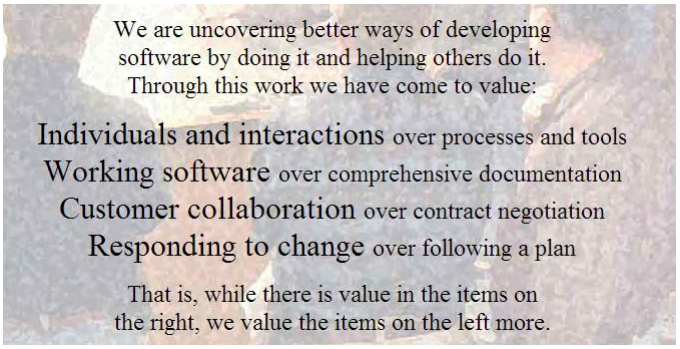
- Some of the reason why software project fails
  - Poorly documented client requirements
    - When do we say it is poorly documented?
  - Impractical schedule
  - Change management problems
  - Testing issues
  - Overly rigid processes

Agile methods

- The overhead cost of software development could be extensive
  - The cost can be in planning, documenting, managing, etc.
  - For small and medium-sized business systems, the overhead involved could be large enough and sometimes dominates the process.
    - More time could be spent working on how the system should be developed rather than on the program development and testing
  - The overhead can be easily justified for heavyweight systems but not for small and medium-sized systems

Agile methods

- The agile manifesto http://agilemanifesto.org/iso/en/



Agile methods

- Agile manifesto
  - Collaboration
  - Communication
  - Focus on people instead of on the process
  - Delivering a working software is a measure of success
    - We need the outcome not how we did the outcome

Agile methods

- Long milestones → short ones (sprint)
- Extensive documentation → less documentation
- Process overhead → product delivery
- Evolved as SD Process and management tool

Agile methods

- **Principles behind the Agile Manifesto**
  1. satisfy the customer through early and continuous delivery of valuable software
  2. At anytime in the development process welcome changing requirements
  3. Deliver working software frequently, from a couple of weeks to a couple of months
  4. Work together daily throughout the project.
  5. Motivate team members
  6. Face-to-face conversation
  7. Measure of progress by the working software
  8. Maintain a constant pace between stakeholders
  9. Continuous attention to technical excellence and good design enhances agility.
  10. Simplicity--the art of maximizing the amount of work not done--is essential.
  11. The best architectures, requirements, and designs emerge from self-organizing teams.
  12. Reflects on how to become more effective

Agile methods

Principle	Description
Customer involvement	Customers should be closely involved throughout the development process. Their role is provide and prioritise new system requirements and to evaluate the iterations of the system.
Incremental delivery	The software is developed in increments with the customer specifying the requirements to be included in each increment.
People not process	The skills of the development team should be recognised and exploited. Team members should be left to develop their own ways of working without prescriptive processes.
Embrace change	Expect the system requirements to change, so design the system to accommodate these changes.
Maintain simplicity	Focus on simplicity in both the software being developed and in the development process. Wherever possible, actively work to eliminate complexity from the system.

Agile methods

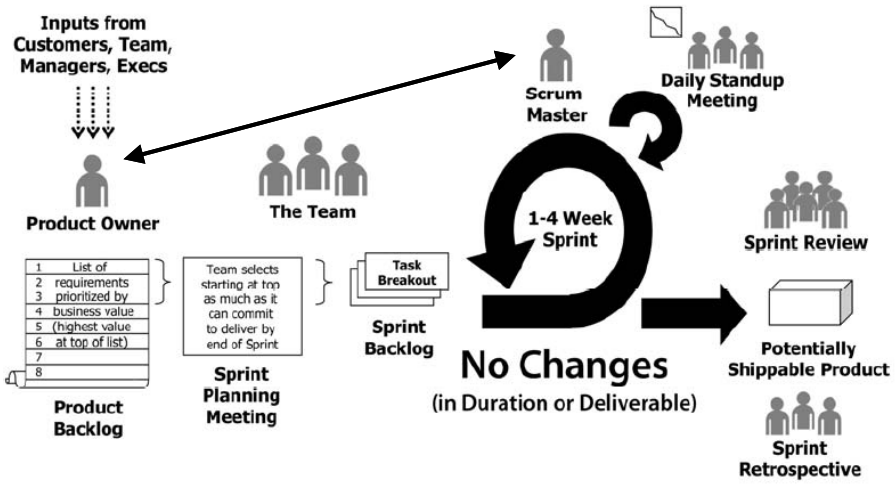
- Best practice
  - Design a clear, concise user story
  - Employ business value prioritization
  - Reflect on the project
- To Success
  - Define Agile principles best suited for the project (organization)
  - Establish and define a glossary of terms
  - Over communicate

Agile methods

- The case of **SCRUM**
  - a set of practices used in agile project management that emphasize daily communication and the flexible reassessment of plans that are carried out in short, iterative phases of work



Scrum



Scrum

- Scrum structures product development in cycles of work called **Sprints,**
- Sprints are typically 1-4 weeks in length and the duration is fixed (if completed or not)
- Scrum emphasizes producing working product at the end of the Sprint is really "done"; in the case of software, this means code that is fully tested and potentially shippable.

Scrum

- Requirements are defined as **user stories** and prioritized and documented as **product backlog.**
  - User story:
    - concise, clear descriptions of the functionality in terms of its value to the end user of the product

      "As a (role), I want (feature), so that (benefit)."
  - Product backlog (wish list)
    - Collection of user stories
    - Exists (and evolves) over the lifetime of the project
    - Continuously updated by the Product Owner to reflect changes

Scrum

- Primary Roles
  - Product owner
  - Scrum Master
  - Team

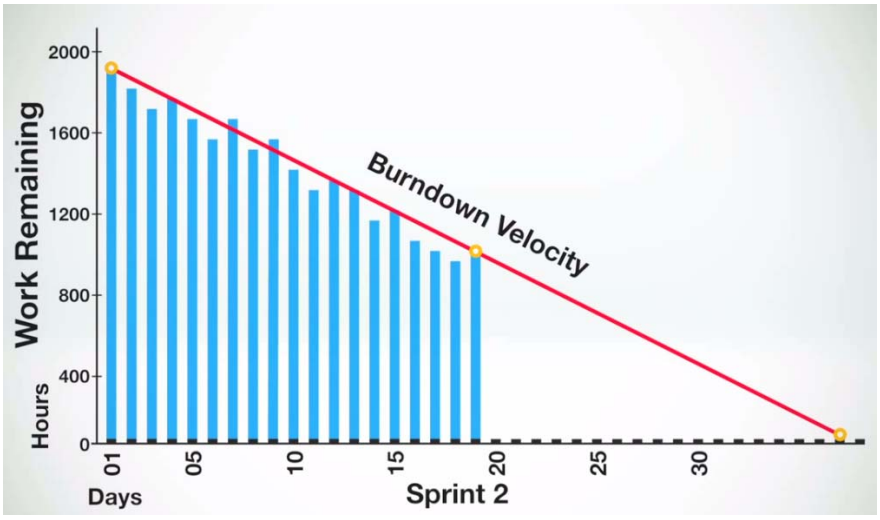
Scrum

- The process
  - Have product backlog
  - Prioritized list of what's required.
    - It is required to make prioritization and decisions (by Product Owner)
  - Sprint planning
    - Facilitated by the Scrum Master.
    - Product Backlog can vary significantly in size; however, the larger products will often be broken into smaller ones.
    - Team select items form the Product Backlog and commit to complete by the end of the sprint.

Scrum

- The process (continued)
  - Daily scrum meeting is:
    - a short (15 minutes or less) stand-up meeting that happens every work day,
    - an opportunity for team to report to itself on progress and obstacles,
    - One by one, each member of the team reports three (and only three) things to the other members of the team
    - No discussion but only reporting. The Scrum Master take notes and makes sure solutions are delivered
  - Sprint Burndown Chart:
    - is used to track the progress of a sprint,
    - shows the amount of work that has been completed in an epic or sprint, and the total work remaining,
    - are used to predict your team's likelihood of completing their work in the time available,

Scrum



Scrum

- The process (continued)
  - Sprint review
    - the team demos what they've built during the Sprint (not presentation)
  - Sprint Retrospective is
    - meeting that occurs at the end of each sprint,
    - review of the process (the sprint) is made by discussing on discuss what went well during the previous sprint cycle and what can be improved for the next sprint,
  - Starting the Next Sprint
    - Following the Sprint Review Meeting, the Product Owner takes all the input, as well as all new priorities that have appeared during the Sprint, and incorporates them into the Product Backlog; new items are added, and existing ones are modified, reordered, or deleted. Once this updating of the Product Backlog is complete, the cycle is ready to begin all over again, with the next Sprint Planning Meeting.

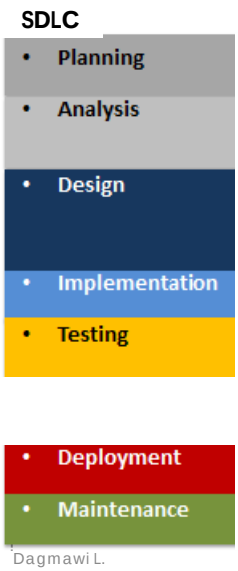
Scrum

- The process (continued)
  - Release Planning
    - Sprints continue until the Product Owner decides the product is almost ready for release

From Agile to DevOps



From Agile to DevOps



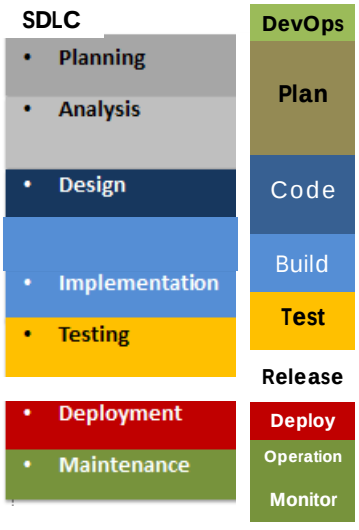


- In earlier process models, requirements were expected to be captured initially in its complete form from the client/users
  - Later it was noted that requirements will change along the process and the notion of stakeholder include many participants in addition to those who owns the system (the client) or the end-users
- Agile, gives more value to meeting requirements than contract negotiation – hence changes are welcome
  - As the results it is common to constantly adjust requirements
  - changes come not only from the business and stakeholders, but also based on the feedback collected from the development team
  - The development team basically starts reflecting beyond technical aspect as they learn more about the system they are developing
- Agile Phases are **plan, code, test** and **review**

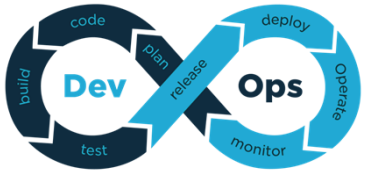
What is DevOps?

- DevOps is a Philosophy and Practice
  - focused on agility
    - So it focuses on collaboration between the client and development team
  - It also focuses on collaboration between the development team and the operation team
  - Practice automation within IT and development team processes
    - Each phase involves tools

Phases in DevOps



- In DevOps, requirements are not only shaped by the client and stakeholders and the development team, but also by feedbacks coming from operations and monitoring stage.
  - This means the initial requirements affect the development, and the development will affect the operations (how things used to be done), and the operations will give feedback affecting the requirements.



Phases in DevOps

- Plan
  - At this stage the client and the development team defines the requirements
  - Features of the system to-be is listed
  - User stories (wishlist) is created and recorded
  - Product backlog is created
- Code
  - Design will be conducted and codes for the designed units are created
- Build
  - Units are integrated and are built as one package
- Test
  - Testing will be conducted (end-to-end) test will be done and units are tested with black box strategy

Phases in DevOps

- Release
  - The version of the build and new features will be recorded and confirmation is given to deploy the build
- Deploy
  - The software is packaged after release and deployed to the production server
    - i.e., the software is transferred from the developers' machine to the client's platform
- Operate
  - Operation Team does the required configuration at the respective platform (server) and facilitates the platform (e.g., network)
- Monitor
  - Issues will be recorded and assessed for the possible impact on the end-user

Modelling Tools for OOSE

- Modelling is the process of creating **model** that represents things in the real world or concepts in a certain domain with the intention of allowing ideas to be investigated.
  - Model is used to show the abstraction of what is being modelled.
  - Abstraction provides us with a specific (partial but relevant **perspective**) of the thing or concept being modelled.
- UML – Unified Modelling Language is used to document and communicate models being created during software development process.
- Note that the software can be sought from various perspective (e.g., data, process, modules, platform, etc.)
  - As it is difficult to represent all these perspectives in one model, UML provides various diagrams for each perspectives.
  - As a language, it has its own rules (syntax)

Modelling Tools for OOSE

- Modelling perspectives in OOSE and diagrams provided in UML

