2. LECTURE 22.04.21

OUTLINE:
1) Software Lifecycle
2) Problem Statement
3) UML
4) analysis, design, implementation & delivery

1) SOFTWARE LIFECYCLE

↳ = Set of activities & their relationship to each other to support the development of a software system

Software lifecycle model
= abstraction representing the development of software for the purpose of understanding, maintaining or controlling the development of a software

ACTIVITIES

Requirement analysis	What is the problem?
System design	What is the solution?
Object design	What are the best mechanisms to implement the solution?
Implementation	How is the solution constructed?
Testing	Is the problem solved?
Delivery	Can the customer use the solution?
Maintenance	Are enhancements needed?

Tailoring: = adjusting a software model to fit a project
There is no „one size fits it all" software lifecycle model

1) NAMING: adjusting the naming of activities
2) CUTTING: removing activities that are not needed
3) ORDERING: defining the order of the activities

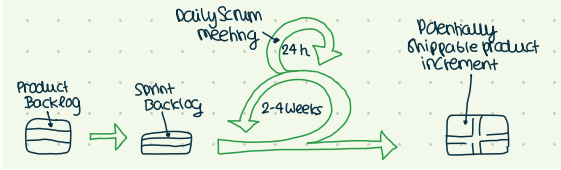
Controlling software development with a process

DEFINED process control model
- Planned
- Follows strict rules
- Avoids deviations

EMPIRICAL process control model
- Not entirely planned
- Inspect and adapt

EXAMPLE

– SCRUM –



2) PROBLEM STATEMENT = description of the problem addressed by the system

↳ describes:
- current situation
- functionality of the new system
- environment in which the system will be developed
- Deliverables expected by client
- Delivery dates (milestones)
- A set of acceptance criteria

3) UML (Unified Modeling Language)

Before implementation: create system models — consists of
- Functional model → represents functionality of the system
- Object model → represents structure of the system
- Dynamic model → represents behavior of the system

=) Notated in UML

Why do we use UML?
- reduces complexity → abstraction
- high level „programming language" => enables generation of source code
- communication between people

3 ways to use UML-Models

Communication:
UML provides a common vocab for informal communication

Analysis and design:
UML models enable developers to specify the future system

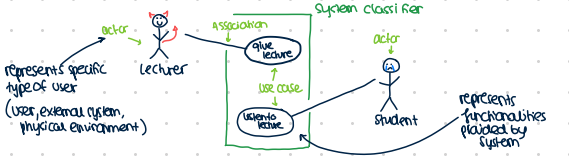
Archival:
UML models provide a way for storing the design and rationale of an existing system

APPLICATION DOMAIN
= environment in which the system is operating

SOLUTION DOMAIN
= Technologies used to build the system

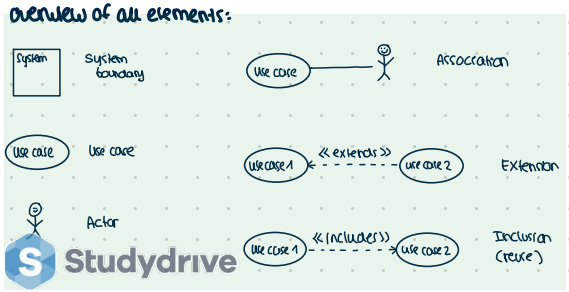
mini tutorial: use case – class diagrams
Describes the functional behavior of the system as seen by user
↳ describes static structure of system: → objects, attributes, associations

UML USE CASE:



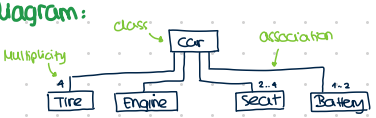
USE CASE ASSOCIATION:
«includes»: functional that is common (reuse)
«extends»: rarely invoked → exceptional functionality

Overview of all elements:



UML Class diagram:

Example:



=> represent structure & relationships of classes

USES IN DIFFERENT PHASES:

- requirement analysis → to model application domain objects
- system design → to model solution domain objects
- object design → to specify detailed behavior and types or attributes of classes

4) ANALYSIS

= finding application domain objects
- flow of events in use cases
- problem statement from customer

OR:
- Application knowledge
- General knowledge
- Solution knowledge

Abbott Technique:

Example	Grammatical construct	UML model object
"Murphy"	Proper noun	Object
"Toy"	Improper noun	Class
"Buy"	Doing verb	Operation
"Is a"	Being verb	Inheritance
"Has an"	Having verb	Aggregation
...	...	...