# **Metode**

Her præsenteres en objektorienteret analyse og design metode, der bygger på fire hovedgrundlag: Modellér IT-systemets omgivelser, fremhæv arkitekturen af systemet, genbrug mønstre der udtrykker velafprøvede designideer, og skræddersy metoden til opgaven og situationen.

**Objekt:** En enhed der har en identitet, adfærd og attributter.

**Klasse:** En samlet beskrivelse af objekter med samme struktur, adfærdsmønster og attributter.

**System:** Realisering af krav til model, funktioner, og grænseflader. (Læs: MVC designmønster)

De generelle fordele ved objektorienteret programmering er, at man kan modellere virkeligheden så det giver mening strukturelt og funktionelt.

**Analyseobjekter**: I analysen beskrives et objekts adfærd gennem de hændelser det udfører eller påføres. Fx. et kundeobjekt kan bestille og sende varer, to funktionaliteter der forekommer på bestemte tidspunkter og involverer en eller flere objekter og klasser.

Beskriver fænomener uden for systemet såsom personer og ting der typisk er selvstændige i forhold til systemet. Ikke altid noget vi kan styre, men vi skal regisrtere de hændelser de udfører eller påføres.

**Designobjekter:** I designet beskrives et objekt derimod ved de operationer det kan udføre og dermed stiller til rådighed for andre objekter i systemet. Beksriver fænomener i systemet som er under vores kommando. Vi beskriver deres adfærd ved de operationer, som datamaskinen kan udføre.

## Modellér omgivelserne

**Problemområde:** Den del af omgivelserne der administreres, overvåges eller styres af systemet

**Anvendelsesområde:** En organisation, der administrerer, overvåger, eller styrer et problemområde.

I et lønsystem omfatter anvendelsesområdet personalekontoret, mens problemområdet er de ansatte, deres overenskomster og arbejdstid.

I et lufttrafikstyringssystem er anvendelsesområdet luftlederens arbejde, og problemområdet er flyenes position, landingsbaner, terminaler, osv. Luftlederen, der skal fortolke systemet korrekt, har allerede en forståelse og et fagtermssprog for hvordan tingene skal fungere, og det er derfor ekstremt vigtigt at problemområdet er modelleret korrekt i forhold til brugernes forventninger. Det kræver derfor, at udviklerne studerer hvordan luftledere og piloter og andet personale skal interagere med systemet.

## Fremhæv arkitekturen

**System:** En samling komponenter, der realiserer krav til model, funktioner og grænseflade.

Analyse og design skal føre til en overordnet forståelse af systemet.

Systemarkitekturen skal være let at forstå, fleksibel og brugbar.

Systemarkitekturen i denne metode beskrives simpelt som med tre komponenter: modelkomponentet, funktionskomponentet, og grænsefladekomponentet.

*Hvis du er i tvivl om hvordan det her skal forstås er det præcis ligesom vi gjorde i vores OOP eksamensopgave - Stregsystem var modelkomponentet, StregsystemController var funktionskomponentet, og StregsystemUI var grænsefladekomponentet. Hvis du vil læse mere om denne model kaldes den MVC (Model View Controller model) på engelsk.*

## Genbrug mønstre

Det er altid en god idé, at anvende velkendte og allerede etablerede designmønstre i sit system. Dette kan være designmønstre som singleton, factory, osv. Dem lærer vi mere om senere. Designmønstre skal selvfølgelig omskrives så de passer til systemet, men at genbruge principperne i dem er god praksis.

## Skræddersy metoden

OOA&D arbejder med fire hovedaktiviteter, som du kan læse om på side 12 og 13. Pointen ved dette underafsnit er, at der er ikke én rigtig måde at gøre tingene på, og der er ikke én rigtig rækkefølge at behandle de fire hovedaktiviteter i. Det er op til dig selv at vurdere hvilket perspektiv der er det vigtiste at arbejde med først.

## Resultater

Det håndgribelige resultat af analyse og design er en dokumentation der beskriver hvordan systemet skal bygges af programmørerne.

## Unified proces og notation

Som notationsmiddel bruges UML som vi er vant til fra P2. Unified Software Development Process er en proces ligesom top-down processen, hvor man i stedet analyserer, designer, programmerer og kvalitetssikrer dele af programmet i inkrementer.

# Opgaver

1. Hvad er forskellen mellem objekter og klasser?

Et objekt er en enhed med identitet, adfærd og attributter. En klasse er en samlet beskrivelse af flere objekter med samme struktur, adfærdsmønster og attributter.

4. Hvorfor skal systemudviklere forstå omgivelserne for de systemer, de designer?

Når man forstår omgivelserne for de system man skal lave kan man derefter modellere det præcist og i et forståeligt forhold til virkeligheden. Dette gør implementation af vigtig funktionalitet lettere for systemudvikleren, når hans udgangspunkt for funktionalitet ligger i det fagtermssprog / funktionsflow der allerede er til stede.

5. Hvordan kan du skelne mellem et systems problemområde og anvdendelsesområde? Kan de delvist overlappe?

Problemområdet er de omgivelser, der skal styres, administreres eller overvåges af ens system. Anvendelsesområdet er den organisation der skal administrere, styre eller overvåge problemområdet.

# Forelæsning

Representations always depend on your knowledge at the point of creating the representation.

The process of development circles between the user organisation and the development team. Once the problem is defined by the user organization, you can decide system specification, develop it and give it to the user organization.

OOA&A has a different approach. It focuses, rather, on what changes we want to do, based on the wishes and needs of the users. Identify their needs, the problem, and their visions.

There are two ways of thinking about system development. One is the world view where the system development is regarded as construction; rational, algorithmic. The other world view is that systems development is evolution, where competences and knowledge is richer than information and data. We have emotions, feelings, and experiences, and these things matter during development. Usually the development lead by the second world view is more dynamic in nature when compared to the construction.

## Construction

Based on rational thinking. The steps are defind by 1. Analysis, 2. Decision , and 3. Action.

Construction is based on the idea of stepwise refinement of a solution to a given well defined and **stable** problem. We specifically assume that the problem does not change in nature.

The traditional approach to systems development in the contructionary sense is the waterfall approach.

## Evolution

The evolutionary world view presents an iterative approach. You complete all the phases from analysis to testing, and then evaluate if the resulting system is what you want. The world view assumes that you learn things during development that you wouldn’t know before.

## OOA&D

A method for analyzing and designing systems that:

1. Creates clarity about systems requirements
2. Establish a basis for implementation

The outcome is supposed to be

An analysis document: a description of system requirements.

And a design document: overview of the systems parts and their interactions.

We use UML as the notation tool.

## Analysis

During analysis we are looking for phenomena in the context of the IT-system - in the real world. We are looking for identify objects, their identity and their behaviour.

## Design

Here we are dealing with phenomena in the IT-system - not in the real world.