

Intro til UML og OOAD

Intro UML.

Unified Modeling Language (UML) er en standard for udseende af diagrammer til beskrivelse af strukturer og forløb i objekt-orienterede softwaresystemer.

UML er ikke kun en standard for udseende af diagrammer men også for dataformatet som f.eks. XMI og XML formater. UML bliver oftest brugt i OOAD og i løsningsforslag når virksomheder udveksler diagrammer mellem forskellige leverandørers og kunder.

Der er udviklet en ny version af UML som har fået betegnelsen UML2 som blev offentliggjort af Object Management Group (OMG) i marts 2005.

Se video UML for beginner:

- <https://www.youtube.com/watch?v=tyvoISFGZwl>
- <https://youtu.be/OkC7HKtiZC0>
- <https://www.youtube.com/watch?v=vAHHdnIV8rU>

Læs mere om UML:

- <http://htx-elev.ucholstebro.dk/wiki/index.php?title=UML>
- <https://www.superusers.dk/artikel/uml/>
- <https://da.wikipedia.org/wiki/UML>
- <https://www.uml.org/what-is-uml.htm>
- <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/>
- <https://www.microsoft.com/da-dk/microsoft-365/growth-center/resources/guide-to-uml-diagramming-and-database-modeling>

UML Værktøjer

Adfærdsdiagrammer / Konceptkort

Diagramtyper som vises systemets adfærd, dvs. en beskrivelse af, hvordan systemets omverden og objekterne i systemet arbejder sammen. Diagrammerne vil til en vis grad have et tidsmæssigt perspektiv.

Video forklaring: <https://www.youtube.com/watch?v=8XGQGhli0I0>

Use Case

I systemudvikling er en use case eller brugsmønster en teknik til at afdække krav. Det kan enten være krav til et nyt system eller krav til ændring af et eksisterende system.

Hver use case indeholder en eller flere scenarier der viser hvordan systemet skal interagere med en bruger eller et andet system for at løse en specifik opgave som systemet skal kunne.

En use case skal typisk ikke indeholde teknisk forklaringer og skal skrives i et sprog som kan forstås af systemets slutbrugere.

Video eksempel: <https://youtu.be/zid-MVo7M-E>

Eksempel på Use Case:

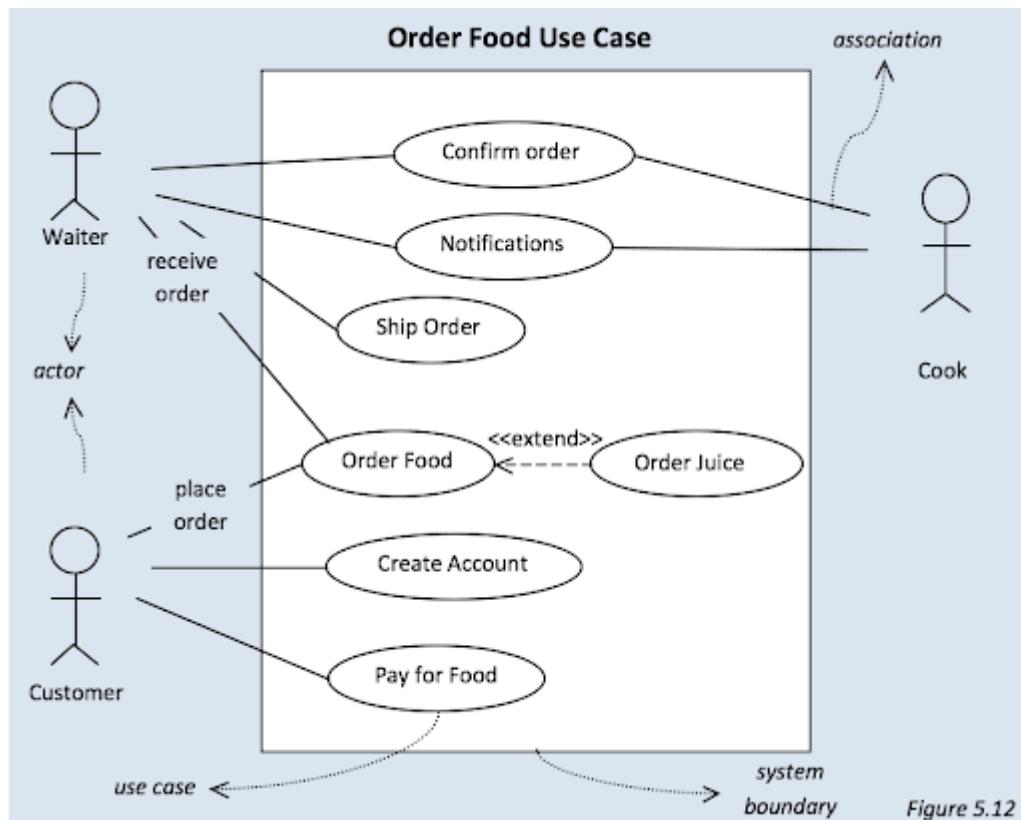


Figure 5.12

FlowChart / Rutediagramm

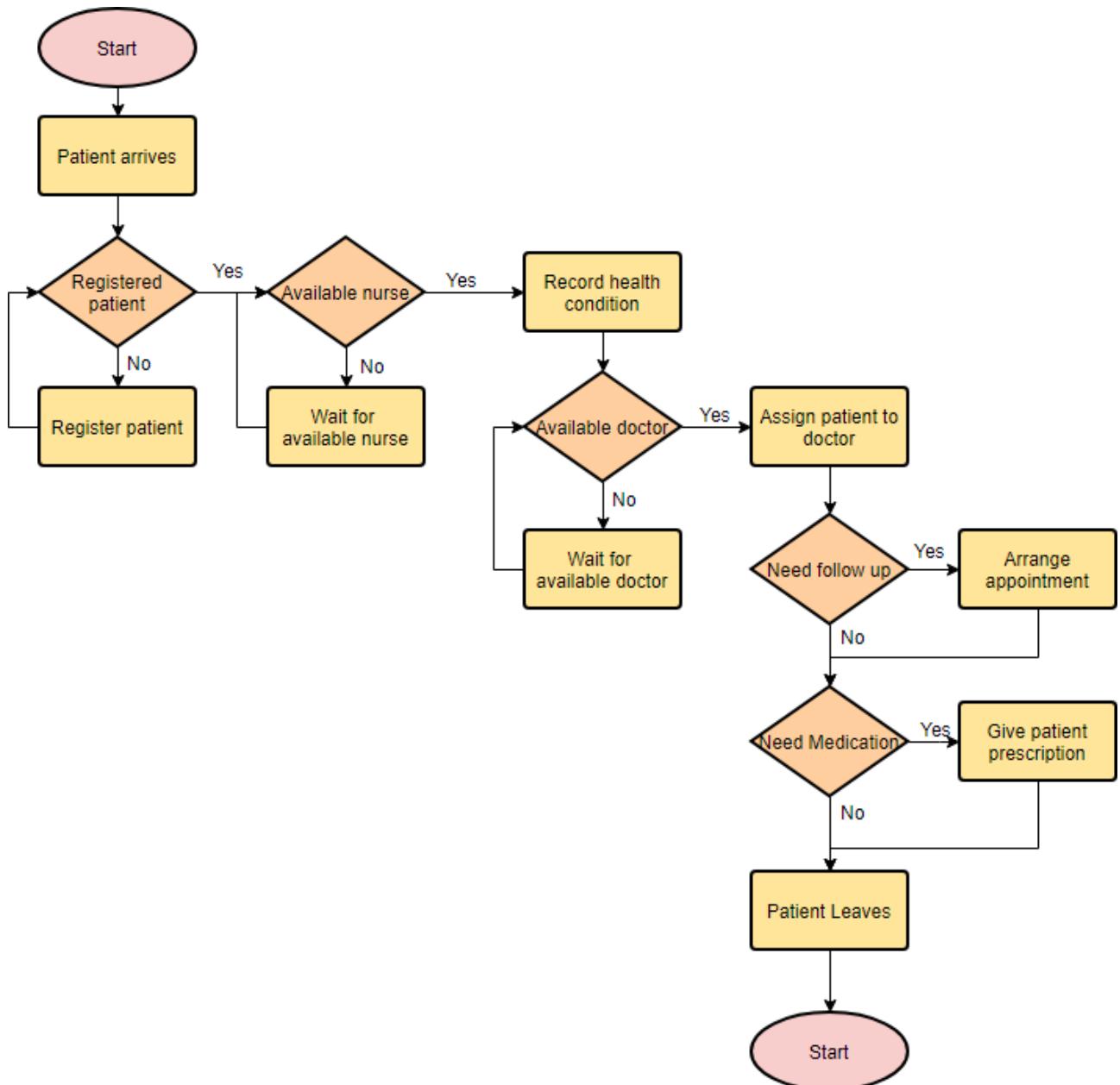
FlowChart også kaldt rutediagrammer, og er en måde at afbilde algoritmer/arbejdsgange/processer i et produkt, så man tydeligt kan se, i hvilken rækkefølge ting skal ske, hvilke beslutninger der skal træffes, og hvilke processer der skal udføres afhængigt af valget.

FlowChart er oftest bearbejdet ud fra standard symboler som benytter ISO standarden og den mest brugte standard er ISO5807 i UML.

Se følgende videoer som forklar FlowChar:

- <https://www.youtube.com/watch?v=iJmcgQRk048>
- <https://www.youtube.com/watch?v=7qclxWFkIAk>

Eksempel på FlowChart:

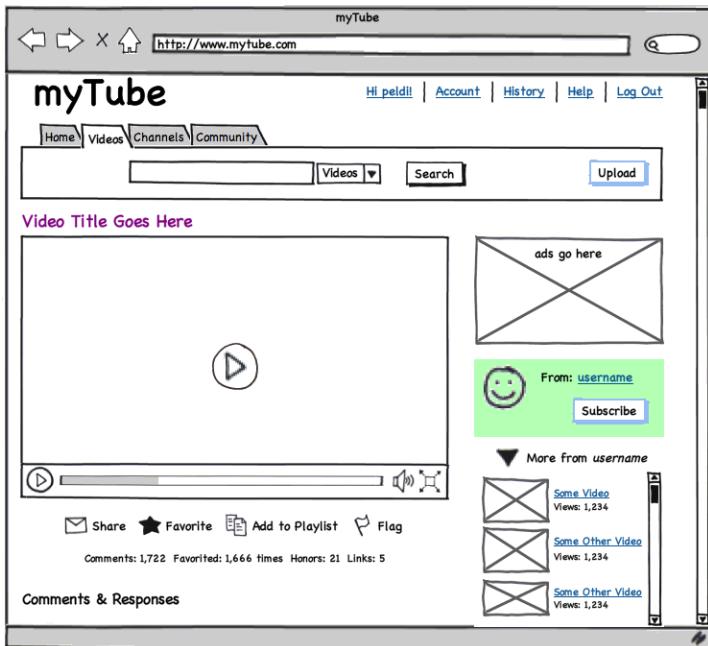


Mockup / Wireframe

Mockup også kendt som Wireframe er en model / design af produktet som også kaldt produkt design. Produktudviklerne / software designe anvender mockup til præsentation og designevaluering over for kunden.

En mockup er ikke nødvendigvis funktionsdygtig, men har den tilstrækkelig funktionalitet til at man kan nå frem til et slutproduktet.

Eksempel på mockup:



Video eksempel på mockup: <https://www.youtube.com/watch?v=Hoj7v0j8w6o>

Læse mere om mockup:

<https://design.tutsplus.com/tutorials/what-is-a-mockup--cms-32231>

<https://www.vikingcodeschool.com/web-design-basics/what-are-mockups>

<https://www.uxpin.com/studio/blog/what-is-a-mockup-the-final-layer-of-ui-design/>

ER / ERD (Entity Relationship)

Et ER diagram er et todimensionelt geometrisk symbolsk repræsentation af information i henhold til en eller anden visualiseringsteknik, det er også kendte som enhedsrelationsdiagram (entity relationship) eller som forkortet ER / ERD.

Video om Entity Relationship (ER):

<https://www.youtube.com/watch?v=QpdhBUYk7Kk>

<https://www.youtube.com/watch?v=-CuY5ADwn24>

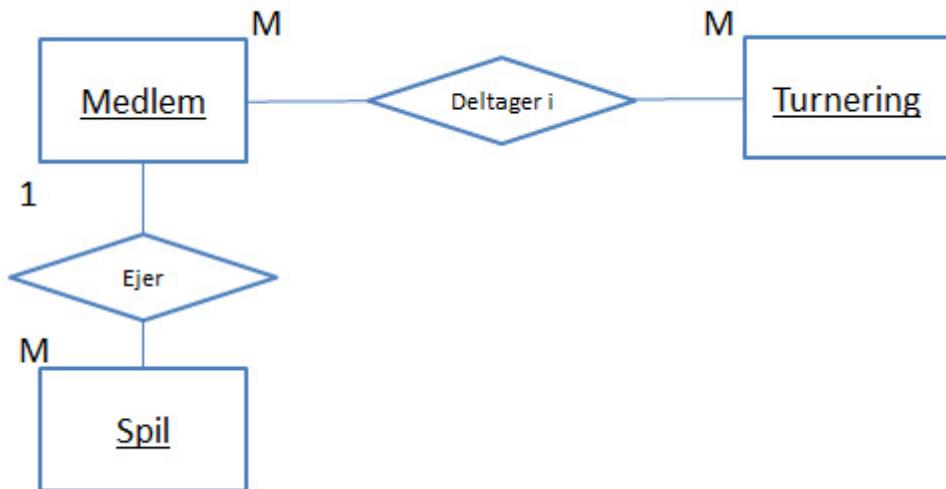
Læse mere om ER:

<https://informatik.systime.dk/index.php?id=1071>

<https://slideplayer.dk/slide/2835148/>

<https://www.smartdraw.com/entity-relationship-diagram/>

Eksempel på ER:



ER diagram viser forholdet mellem entitetssæt (relationships), der er gemt i en database. En enhed i denne sammenhæng er et objekt, en komponent af data. Et entitetssæt er en samling af lignende enheder.

Se også eksemplet: https://mars.merhot.dk/w/index.php/6238_Databaser_Agenda/ER_diagrammer

Klassediagrammet

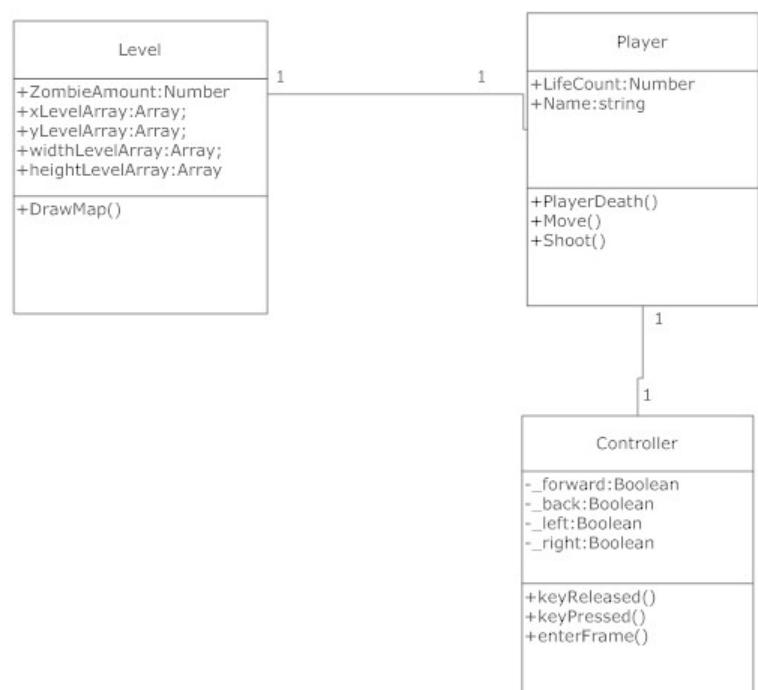
Den mest kendte UML-diagramtype er klassediagrammet, som definerer en måde at beskrive udvalgte dele af et objektorienteret systems klasser. Diagrammet viser strukturen mellem de udvalgte klasser, f.eks.

- Arv: Hvis en klasse er subklasse af en anden klasse eller implementerer et **interface** (grænseflade-klasse).
- Associering: Hvis der er en sammenhæng mellem to klasser, så en klasse har en anden klasse som medlemsvariabel (kan også være gensidigt).
- Aggregering: Er egentlig et specialtilfælde af en associering, der viser et "består af"-forhold. Viser, at et objekt eksisterer på grund af et andet objekt. Hvis det overordnede objekt slettes, slettes de underordnede objekter også. Ved "almindelige" associeringer kan sammenhængen til et andet objekt derimod skiftes ud.
- Afhængighed: Andre tilfælde, hvor en klasse er afhængig af kendskabet til en anden klasse, i praksis ved at den anden klasse optræder som parameter i en metode eller en lokal variabel.

I forbindelse med objektorienteret analyse og design bruges UML både i analyse- og designfasen. I analysefasen har et klassediagram stort set samme rolle, som et ER-diagram (eng: *Entity-Relationship Diagram*).

Eksempel på klassediagrammet:

I disse diagramtyper vises systemets adfærd, dvs. en beskrivelse af, hvordan systemets omverden og objekterne i systemet arbejder sammen.
Diagrammerne vil til en vis grad have et tidsmæssigt perspektiv



Se også eksemplet på hvordan man dokumentere et klassediagrammet:
<http://legmedkode.blogspot.com/2010/04/klassediagram.html>

Video hjælp med klassediagrammet
<https://youtu.be/UI6lqHOVHic>
<https://youtu.be/3cmzqZzwNDM>

Øvelse: 1.

Virksomheden B&B ønsker at få lavet en web-shop med mulighed for en administrativ grænseflade, hvor de nemt kan tilføje nye sider og produkter. De har derfor anmodet praktikcenteret om at lave et koncept, hvor de viser et løsnings forslag ud fra UML modeller.

B&B ønsker derfor følgen:

- En kort forklaring på løsningen
- Use Case
- FlowChart
- Mockup
- ER / ERD
- Klassediagram

Øvelse: 2.

Svar på følgen spørgsmål og skrive dette med i dokumentation.

1. Forklar hvad er UML?
2. Hvilke fordel er der ved at bruge UML?
3. Hvor bruger man oftest UML til?
4. Forklar hvad er forskellen på ER diagram og Klassediagram?
5. Hvilke værktøj bruger man til at vise sin ide designe over programmet?

Hjælp til øvelsen:

- <http://kom.aau.dk/~dimon/old-control/teaching/ssuslides8.pdf>

Indhold i dokumentation

1. Forside
2. Indholdsfortegnelse
3. Svar på spørgsmål i øvelse 2
4. Forklaring og løsningens forslag til øvelse 1
5. Konklusion

Rapporten skal sendes til din instruktør. Øvelsen forsætter med OOAD som er udvidelse til øvelse med UML.

Se også intro videoen om Objekt Design før du starter på OOAD:

<https://www.youtube.com/watch?v=oL82p9QqFKs> / <https://youtu.be/hb7Q33ysCwl>

Intro til OOAD

Indledning

Arbejdet med Objektorienteret Analyse og design er en iterativ process idet analyse processen kan afdækker krav og behov undervejs som kan føre til ændringer og tilføjelser til de forskellige afsnit.

Den overordnede fremgangsmåde er følgende:

- Forstå/definer systemets omgivelser og dets anvendelse (rigt billede, use cases)
- Design systemets overordnede arkitektur
- Identificer de vigtigste objekter i systemet (klassediagrammer)
- Opstil design-modeller
- Definer interfaces mellem objekter

Unified Modelling Language UML er en formalisme til at beskrive OO design med.

UML definerer model-notation ikke en design-metodik.

UML bruges til at dokumentere design og hjælpe udviklerne til at kommunikere indbyrdes

UML diagramtyper (adfærds diagrammer):

- **Use case diagrammer** – en specifikation af en handlingssekvens mellem en ekstern aktør og systemet
- **Klassediagrammer** – beskriver det statisk klasse hierarki med indbyrdes relationer
- **Interaktionsdiagrammer** – En demonstration af hvordan objekter kan samarbejde herunder i hvilken rækkefølge de påvirker hinanden samt oprettelse / nedlæggelse af objekter
- **Tilstandsdiagrammer** – Beskriver et objekts tilstand i et forløb – minder om flowcharts
- **Aktivitetsdiagrammer** – Beskrivelse af "workflow" for hele eller dele af systemet på et overordnet plan

UML diagramtyper (struktur diagrammer):

- **Pakke-diagrammer** – Bruges til at vise indretningen og organiseringen af modelementer i mellem- til storskala-projekt.
- **Deployment-diagrammer** bruges til at vise: (i client-server arkitektur)
 - "Intelligente" enhed i systemets
 - Kommunikation mellem subistemet
 - Software-komponenter på de enkelte enheder

Table of Contents

Indledning	1
Objektorienteret analyse og design	3
Hvad er objektorientering?.....	3
Kravspecifikation	3
Objektorienteret analyse.....	4
Use cases og specifikationer.....	4
Objektorienteret design.....	4
Statisk modellering	4
Klassediagram	4
Dynamisk modellering	5
Tilstandsdiagram	5
Interaktionsdiagram.....	5
Aktivitetsdiagram	5
Mock up.....	5

Kilde: https://da.wikipedia.org/wiki/Objektorienteret_analyse

Objektorienteret analyse og design

Hvad er objektorientering?

- **Objekter**
En helhed med identitet, tilstand og adfærd
- **Et objekt tilhører en klasse**
- **Klasse:**
En beskrivelse af en samling af objekter med samme struktur, adfærdsmønster og attributter
- **Til enhver klasse hører en mængde af objekter**

Kravspecifikation

Virksomhedsbeskrivelse:

Hvad beskæftiger virksomheden sig med? Beskriv hvordan de arbejder.

Formålet med programmet:

Beskrive hvilke opgaver programmet skal løse for virksomheden.

Målgruppe:

Hvem skal bruge programmet. Lav en liste med aktørerne og deres rolle (f.eks.: administrator, kontor personale, IT-supportere osv).

Programmets struktur:

Lav en beskrivelse af programmet gerne med en skitse til et layout (mockup) udfra samtaler med kunden.

Tekniske specifikationer:

Her skal der være en liste over hvilke funktioner programmet kan udføre fra brugergrænsefladen og hvilke grænseflade der er til interne og eksterne systemer (databaser, webservice mm.)

Ønsker til design:

Evt. layout, farver osv.

Tidsplan:

Her skal der være et estimat på tidsforbrug fordelt på de enkelt faser.

Sikkerhed:

Her skal der stå noget om hvilke foranstaltning virksomheden skal gøre sig med hensyn til **GDPR**, sikring af data, backup strategier osv.

Objektorienteret analyse

Formålet med enhver analyseaktivitet i softwarens livscyklus er at skabe en model af systemets funktionelle krav, der er uafhængig af implementeringsbegrensninger.

Den største forskel mellem objektorienteret analyse og andre former for analyse er, at ved hjælp af den objektorienterede tilgang organiserer krav omkring objekter, som integrerer både adfærd (processer) og tilstande (data) modelleret efter virkelige objekter, som systemet interagerer med.

I andre eller traditionelle analysemетодer betragtes de to aspekter: processer og data separat. For eksempel kan data modelleres ved hjælp af ER-diagrammer og adfærd ved flowdiagrammer eller strukturdiagrammer.

Use cases og specifikationer

Ofte bruges **Use cases** (adfærdsmønstre) på et problemområdet (opgaven). Idet Use cases kan beskrive en række ensartede, afgrænsede og afsluttede anvendelser af systemet, og ofte indledes analysearbejdet med at finde frem til mængden af use cases i systemet.

Use cases beskrives med en eller flere **use case diagram(er)** samt en eller flere **use case specifikation(er)**.

I specifikationerne beskrives interaktionen mellem bruger og systemet i detaljer, men der skal ikke vælges en bestemt teknisk løsning.

Objektorienteret design

Statisk modellering

Klassediagram

En klasse er beskrivelsen af en række objekter med samme adfærdsmønster, attributter og struktur.

Et eksempel fra en salgsvirksomhed kan illustrere sammenhængen:

Antag at virksomheden har kunder som "Jens Hansen", "Inger Melgaard" og "Andersen og co.".

Disse tre kan betragtes som objekter fra klassen "Kunde".

I objektorienteret analyse findes de relevante klasser og en beskrivelse af sammenhængene mellem dem i et klassediagram. Der er flere måder at finde klasser på, f.eks.

- interview af personer fra problemområdet
- tekstanalyse
- gennemgang af checklister

Hvor der i de to først tilfælde fokuserer på at finde navneord og derudfra udsøger sig relevante begreber, som kan indgå i beskrivelsen af systemet. De relevante begreber kan suppleres med check af lister med forslag til klasser, især hvis der ikke er så store muligheder i de to første tilgangsmåder.

I analyseklasse diagrammet fokuseres på begreber, der er anvendes i problemområdet. Der gåes ikke alt for detaljeret til værks i beskrivelserne, og der skal undgåes at træffe designmæssige beslutninger for tidligt.

Eksempel: Ofte vil man f.eks. have brug for "Adresse" som attribut i en klasse, f.eks. i en "Kunde"-klasse, den attribut vil sandsynligvis i designet blive opdele i flere attributter: "vej", "postnummer" etc., så vil der oftest være tale om et valg af, hvordan attributterne konkret skal repræsenteres, og dette valg udskydes til design fasen.

Dynamisk modellering

Tilstandsdiagram

Er næsten det samme som flowcharts. Diagrammet kan vis den tilstand (state) et objekt er i, og hvilke tilstande det potentiel kan komme i. Ligesom i flowcharts er der følgende:

- Begyndelsestilstand
- Udfør en handlinger
- Når en betingelse er opfyldt skiftes til en ny tilstand

Interaktionsdiagram

Interaktionsdiagrammer (sequence diagram) bruges til at beskrive, hvordan objekter samarbejder om at løse en opgave i problemområdet.

Aktivitetsdiagram

Et aktivitetsdiagram beskriver et "Workflow" for hele systemet eller del af det. Det ligner tilstandsdiagrammer, men er ikke begrænset til enkelte objekter.

Mockup

En grafisk fremstilling af hvordan programmets brugergrænseflade kan se ud som udvikleren forestiller sig den lavet.