Tempus SERVA

Kravspecifikation træning

UML modeller

Program

- UML introduktion
- Usecase model →
- Klasse diagram →
- State diagram →
- Andre modeller

→ Øvelse / diskussion

UML introduktion

Unified Modelling Language

- Beskrivelsesramme der sikrer entydighed
- Kan skrives af ikke-teknikere
- Kan læses igen af programmører

Værktøjsunderstøttelse

- Tegning: **creately.com**, Dia, Visio, gliffy.com (klasser)
- Modeller:
  - **Open source**: Modellio, Posseidon, Eclipse, Netbeans,
  - Kommercielle: Visual Paradigm, Rational

UML introduktion

Vi kigger på et begrænset sæt

Anvendelse

- Usecase diagram
- Usecase beskrivelser

Modellering

- Klasser: Statiske egenskaber
- States: Dynamiske egenskaber

Usecase diagram

Hvad:

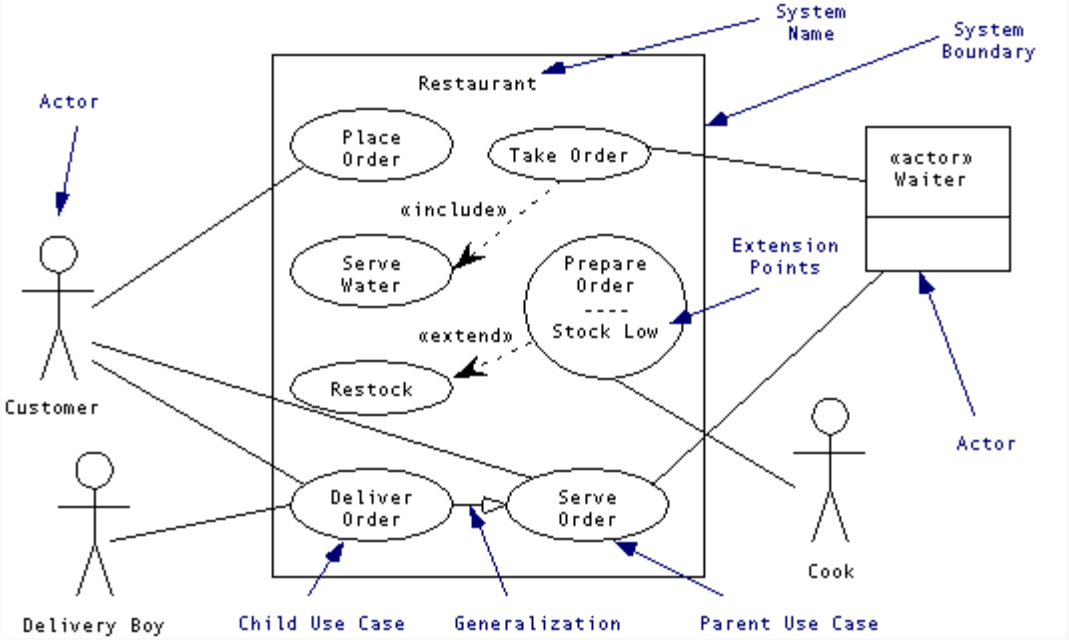
- Brugere og handlinger

Hvorfor:

- Overblik over funktionalitet
- Nedbrydning af abstrakte opgaver
- Afdække afhængigheder og bindinger

Hvornår:

- Altid

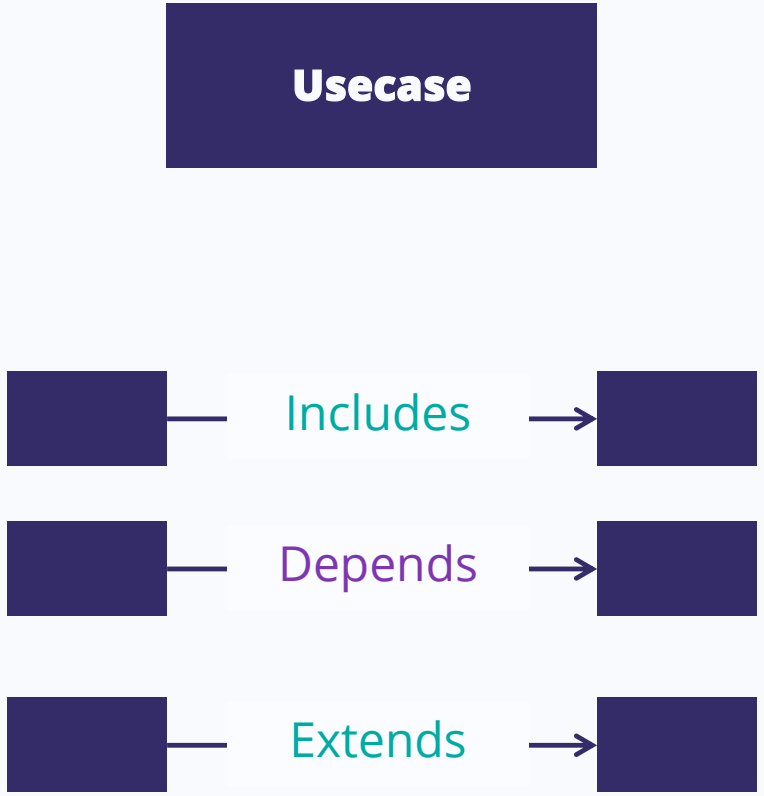


Usecase diagram

Komponenter

- Usecase
- Aktør
- Relation
  - Afhænger af
  - Indeholder
  - Variant af

Aktør



Usecase: Definition

→ **En selvstændig arbejdsopgave**

- **Postiv**: ”Send ansøgning”, ”Søg efter rapport”
- **Negativ**: ”Udfyld felt”, ”Sagsbehandling

→ **En selvstændig arbejdsopgave**

- ”Jeg tager en kaffe efter jeg har …. ” <usecase>
- Veldefineret start og slut
- Et afgrænset tidsrum

Usecase: Beskrivelse

Minimum:

- Titel
- Beskrivelse
- Aktør anvendelse

Optioner

- ID nummer (f.eks. ”U26”)
- Steps i processen
- Frekvens (hvor ofte udføres den?)
- Varianter (for at spare plads/tid)
- Relationer

Usecase: Aktør

Varianter:

- En type af personer (f.eks. ”Chefer”)
- Et navngivent system (f.eks. ”DeMars”)

- Ofte vil man inkludere dem i hinanden, for at lette beskrivelser og diagrammer

Usecase: Relation

Depends (”Afhænger af”)

- [Spise banan] depends [Købe frugt]

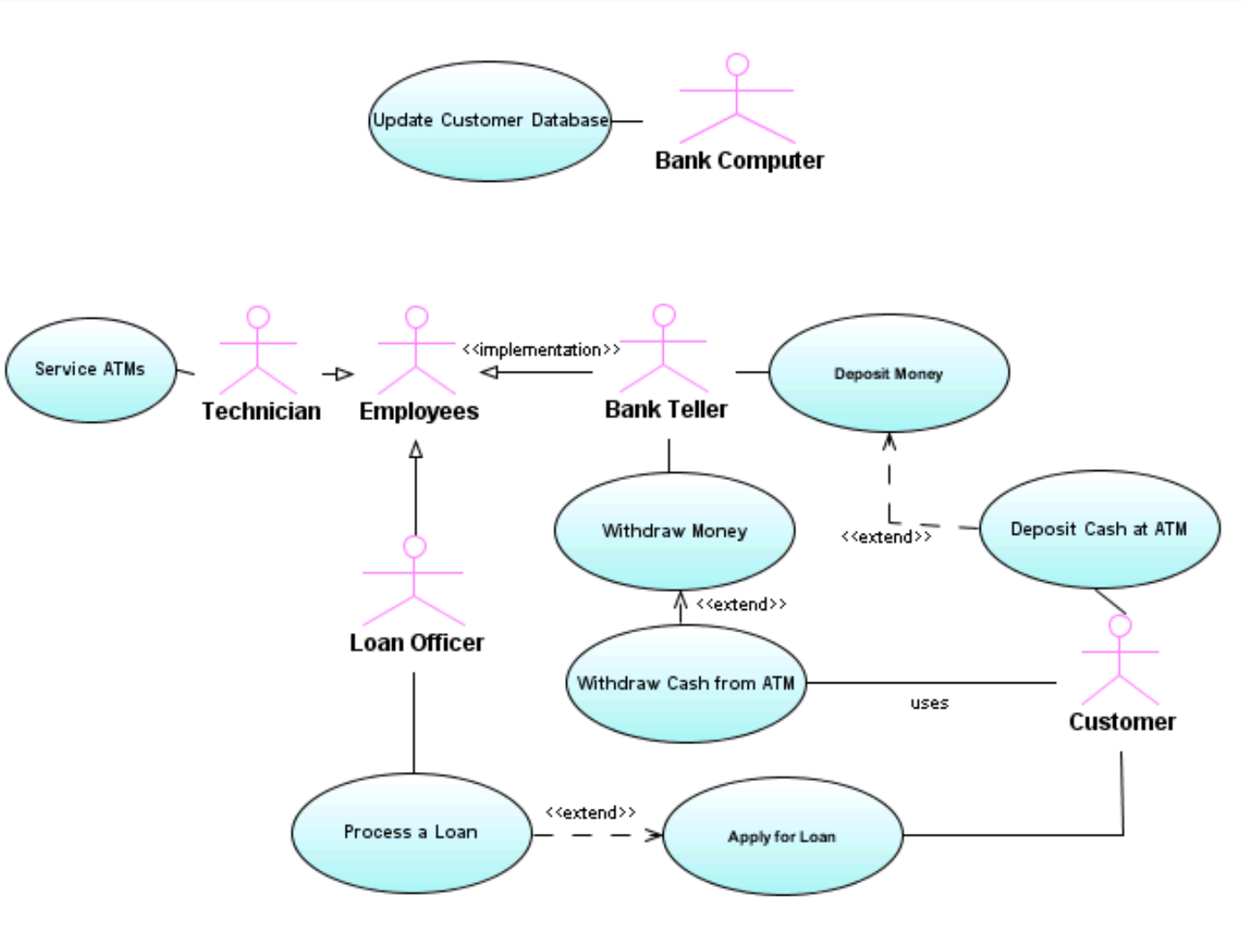
Includes (”Indeholder”)

- [Spise banan] indeholder [Skrælle banan]

Extends (”Udvider”)

- [Avanceret søgning] extends [Søgning]

Usecase: Diagram



Usecase: Øvelse

→ **Øvelse: Internet butik**

Egne papir skitser

- Definer aktører
- Definer usecases

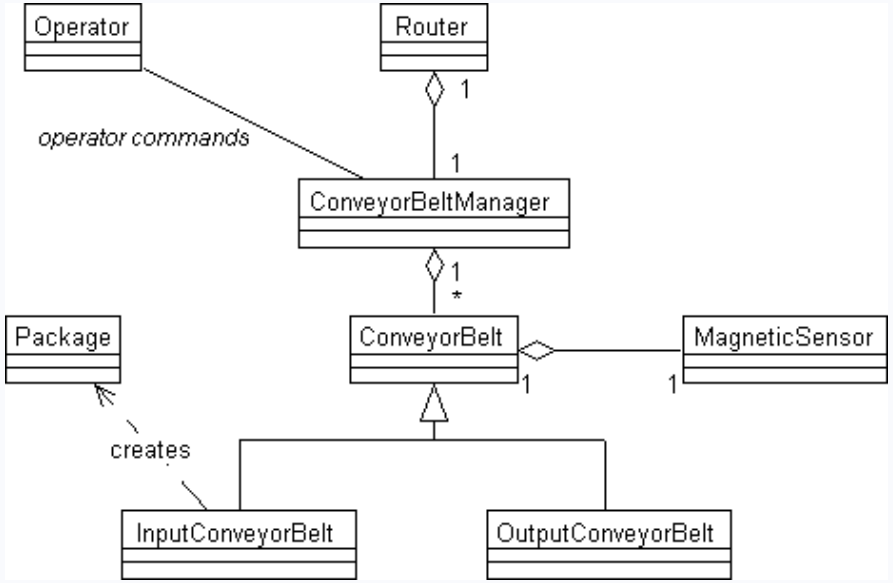
Fælles opsamling på tavlen

Klassediagram

Hvad
Entiteter i systemet

Hvorfor
Afklaring af model bindinger
Input til endelig datamodel

Hvornår
Når systemets indhold er komplekst



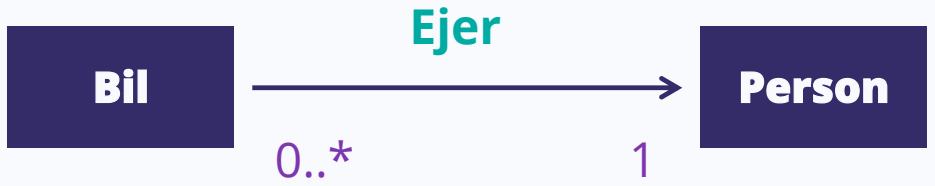
Klassediagram

Klasse

- Attributter
- Funktioner

Relationer

- Betydning
- Kardinalitet
- Type: ”Association” eller ”Aggregation”



Klassediagram

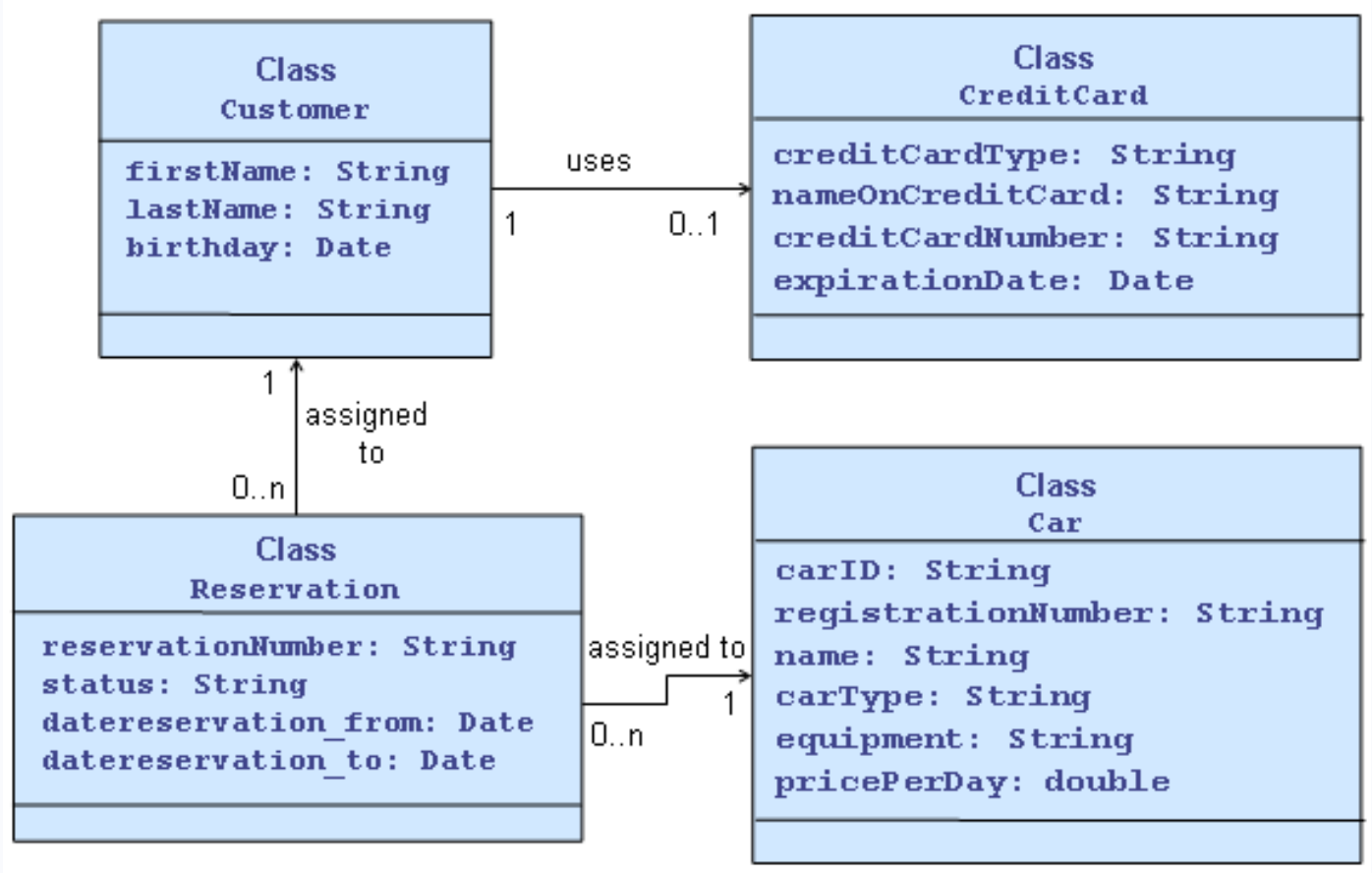
Klasser er abstrakte og generiske

- ”FMC”, ”FKIT” → ”Afdeling”
- ”Lones bil”, ”Johns bil” → ”Bil”

Input til klasser

- Forklaringsskitser fra brugerne
- Relationer mellem klasser (f.eks. ”booking”)
- Rapportering (attributter, relationer)

Klassediagram



Klassediagram

→ **Øvelse: Kursus booking**

Fælles på tavlen

- Navngiv klasser
- Tilføj relationer

State diagram

Hvad

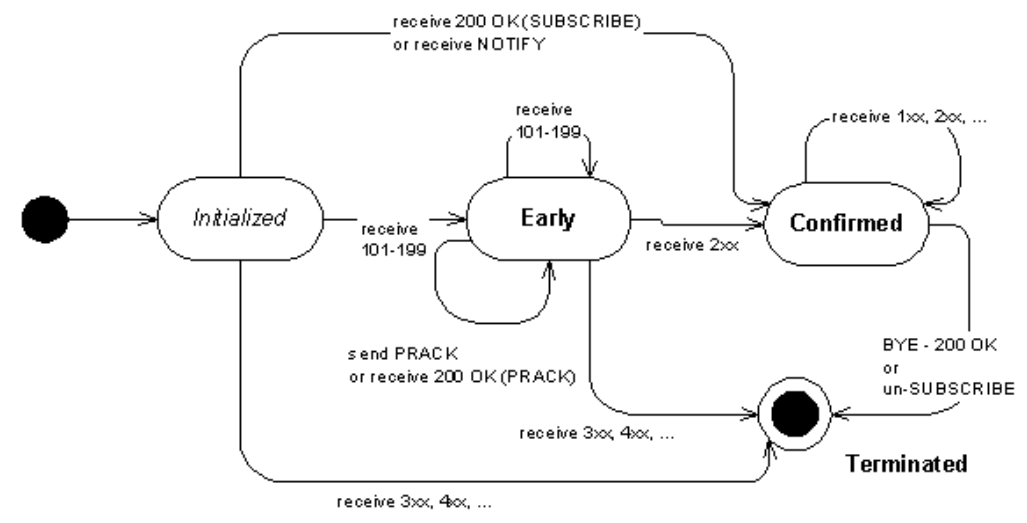
- Tilstande for **én** entitet

Hvorfor

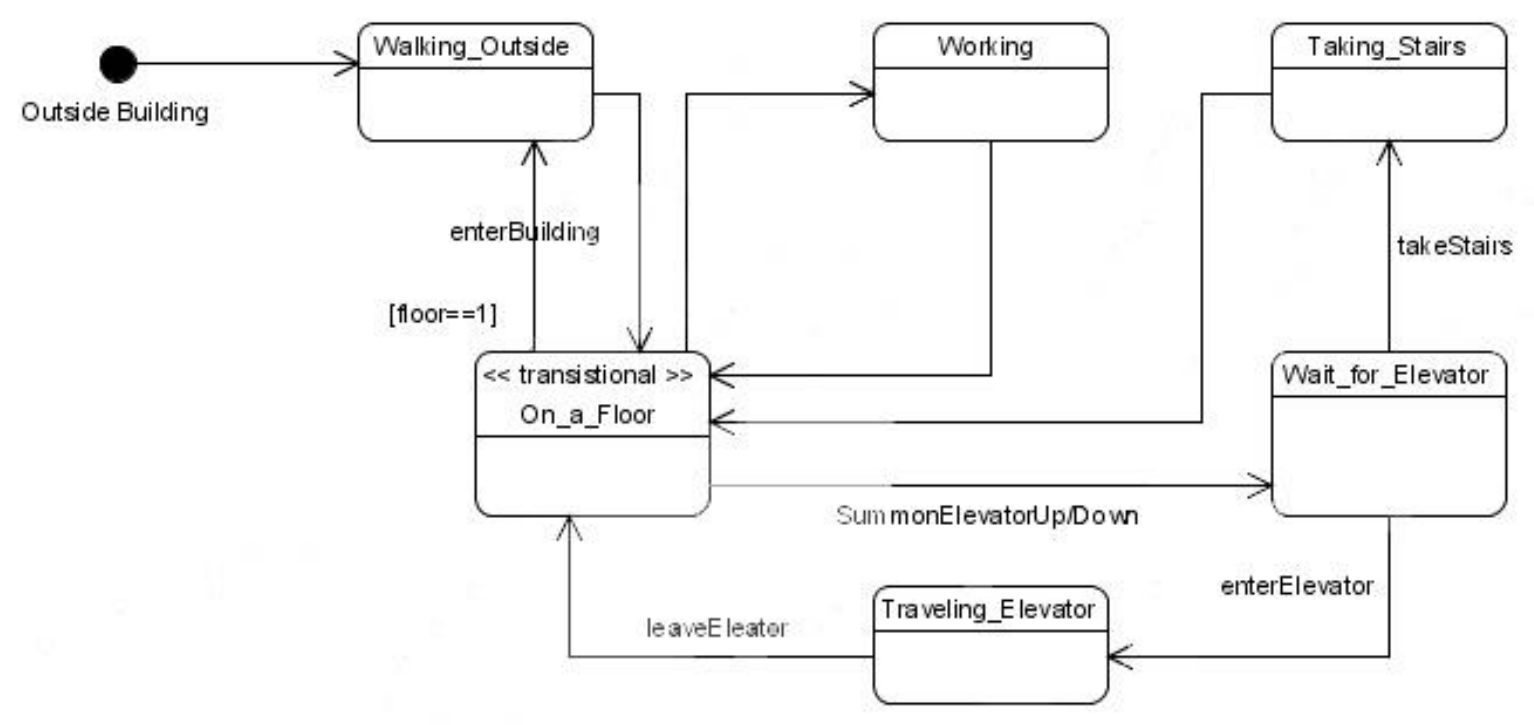
- Livscyklus analyse af data
- Checkliste for handlinger og usecases

Hvorfor

- En for hver **vigtig** entitet i systemet



State diagram



State diagram

→ **Øvelse: Leasing bil**

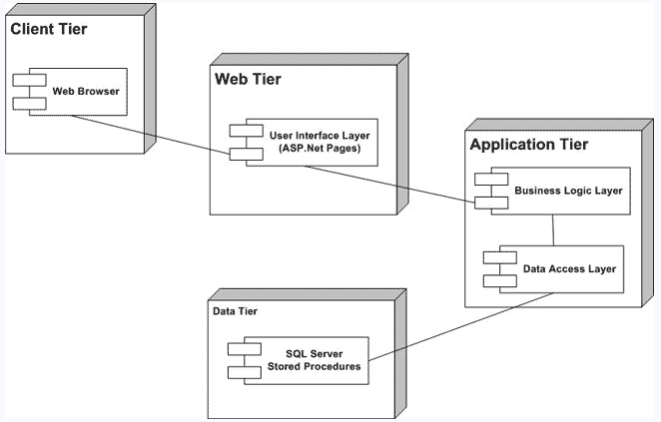
Fælles på tavlen

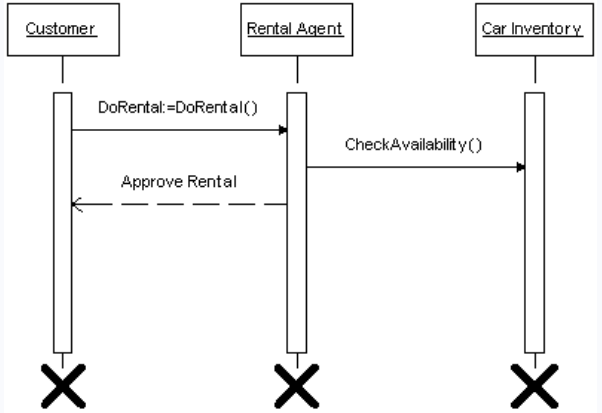
- Definer start og slut punkt
- Definer mellemliggende stadier

Andre modeller

Tekniske diagrammer

- Squence / interaction
- Deployment / component
- Activity (”omvendt state”)





Andre modeller

Ikke UML diagrammer

- Domæne billede
- E/R diagram
- Flowcharts