UML 1

* **Hvorfor modellerer vi?**
  + Kommunikasjon
    - Enten for å kommunisere direkte, der og da eller for å dokumentere til ettertid.
    - Vi får komprimert ting med diagram i motsetning til beskrivelser med setninger.
  + Dokumentasjon
    - Om vi skal gi prosjektet videre til noen for å videre utvikle eller vedlikeholde.
    - Når vi lager ting må det vedlikeholdes, og når ting blir komprimert gjøres dette lettere
* **Use case**
  + Beskriver en typisk og spesifikk brukssituasjon av systemet.
  + Ha med verb i tittelen
  + Vi vil vite hvem som er assosiert med et use case
    - En aktør er typisk en bruker av et system
    - En aktør assosieres med et use-case ved hjelp av en enkel linje uten piler (association)
  + Inkluderte use case gir oss en use case med økt detasjeringsgrad.
    - Da sier vi at et use case er en obligatorisk del av en annen use case.
    - Dette gjør det mulig å gjenbruke use cases, som f.eks kan en søkefunksjonalitet brukes i flere forskjellige andre use case.
  + Noe annet vi kan bruke utvidelser (extend). Da er ikke det relaterte use caset en obligatorisk del av et annet, noe som blir utført i spesielle tilfeller. Pilen vil gå motsatt vei fra include.
  + Systemgrense forteller oss hvilke use case som tilhører et system, dette vil typisk være kun ett system og nettopp det systemet vi skal lage.
  + For å illustrere forskjellen mellom include og extends, kan vi bruke monopol. Når det er vår runde og vi skal flytte brikken må vi kaste terningen, men vi trenger ikke nødvendigvis å gå i fengsel, det er et spesialtilfelle, dermed er det extend, mens å kaste terningen er include.
  + For å få beskrevet en use case i mer detalj brukes tekstlig beskrivelse, der vi har med:
    - Aktører
    - Pre betingelser
    - Postbetingelser
    - Hovedflyt
    - alternativ flyt/unntak.

Use Case - Oppgave

Anta et system som skal håndtere bordreservasjoner og bordplassering i en restaurant. Kunder kontakter restauranten for å bestille eller avbestille bord. En resepsjonist mottar samtalene. Bestillinger legges inn for et bestemt bord sammen med antall personer. For hver bestilling registreres en kontaktperson med navn og telefonnummer. Det skal også være mulig å bestille via internett. Når gjester ankommer, blir de plassert ved sitt bord av hovmesteren, og deres bestilling markeres med “ankommet”. Hvis gjestene plasseres ved et annet bord enn det som var registrert med bestillingen, så registreres bordbyttet i bestillingen. Tidspunktet da et gitt bord må være ledig igjen kan også registreres. Kunder kan endre bestilling eller avbestille bord på forhånd. Det er selvfølgelig mulig å spise uten å ha bestilt på forhånd hvis det er ledige bord. Når gjester får bord uten å ha bestilt dette, markeres det i systemet med tidspunkt, bord og antall, men uten navn og telefonnummer. Når nye bestillinger registreres i systemet, eller eksisterende bestillinger endres, skal skjermbildet umiddelbart oppdateres, slik at de ansatte på restauranten alltid har oppdatert informasjon tilgjengelig.

**Aktør**

-Hovmester & resepsjonist (ansatte)

-Kunde

**Use Case**

-Reserver bord

-Avbestille bord

-Endre bestilling

-Registrer bordbytte

-registrer ankomst

-registrer drop-in

-vis bestillinger

-reserver bord på nett

* **User Stories**
  + Dette er ikke UML og ved bruk av disse modellerer du ikke, men de har tilsvarende funksjon som bruk av use case-modellering.
  + Brukes gjerne som alternativ til use-case modellering i agile prosjekter.
  + User stories formateres slik at du har:
    - Som <rolle>
    - Ønsker jeg <mål>
    - Slik at <fordel>
    - Slik at kan ofte virke overfladisk og kan droppes, det viktige er «Som <rolle>» og «Ønsker jeg <mål>»
  + Merk at user stories som regel representerer et mindre stykke funksjonalitet enn et Use Case og eksempelet over ville kanskje blitt delt opp i mindre biter.
  + Om disse skal fungere godt som kravdokumentasjon bør man også beskrive testbare akseptansekriterier for hver story.
    - Disse tilsvarer den tekstlige beskrivelsen av et use case.
    - Et alternativ eller tillegg til generelle akseptansekriterier er scenarioer.
      * Vi bruker nøkkelordene «Gitt», «Og», «Når» og «Så» eller «Given», «And», «When» og «Then» hvis vi dokumenterer/modellerer på engelsk.
* **Domenemodellen**
  + En spesifikk anvendelse av klassediagrammet
  + Beskriver entiteter (subjekter) og ikke klasser, og viser forholdet mellom dem.
  + Funksjonen til domenemodellen er å få utviklere og domeneeksperter til å enes om en felles forståelse av problemdomenet.
  + For å unngå misforståelser i utviklingsprosessen trenger et felles språk, dermed er navngiving av elementer i modellen viktig.
  + En domenemodell skal ikke være detaljert.
    - Den skal vise konseptene i et domene på en oversiktlig måte.
    - Skal kunne benyttes i samtale med ikke tekniske interessenter.
  + Ligner på et ER-diagram, men hensikten er en helt annen:
    - En ER-modell er en implementasjonsmodell som detaljert beskriver hvordan en database er implementert.
    - En domenemodell er mer konseptuell og skal mer overordnet beskrive entitetene og sammenhengen mellom disse i et problemdomene.
  + Hva vi tar med i modellen er situasjnsbetinget. Vi kan ta med:
    - **navn på assosiasjoner** som forklarer forholdet mellom objekter (klasser).
    - **attributter** hvis disse er viktige for forståelsen av domenet.
    - **metoder** hvis disse er viktige for forståelsen av domenet
    - **multiplisitet** hvis det er viktig for forståelsen av domenet
    - **kommentarer** ved hjelp av gule lapper hvis noe føles uklart