Institutt for datateknologi og informatikk, NTNU

SU1

UML 1

Nils Tesdal, 2017-2019

NTNU
Kunnskap for en bedre verden

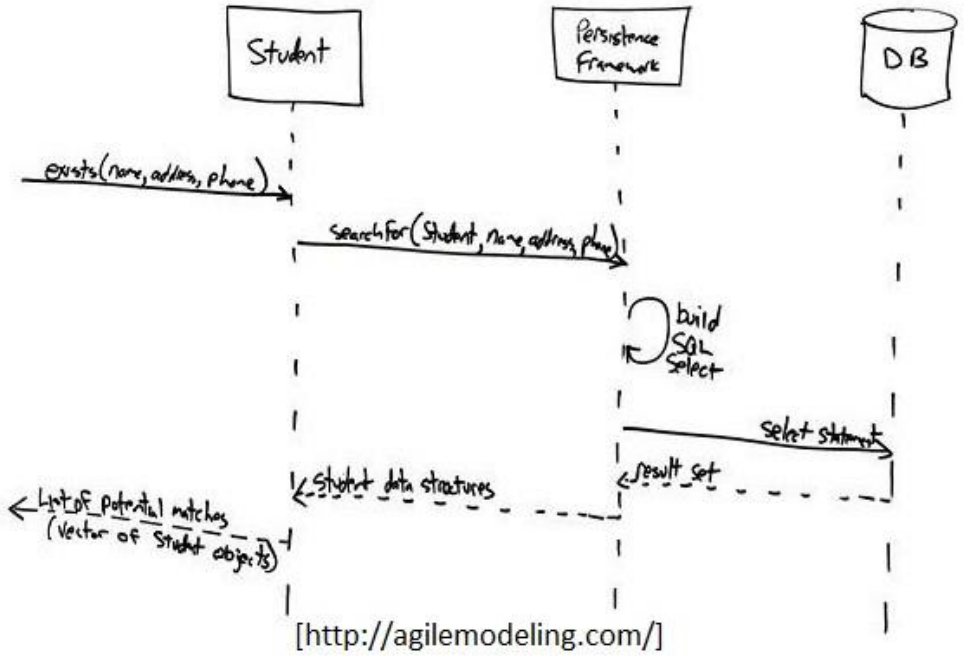
Hvorfor modellerer vi?

- Kommunikasjon
  - Der og da
  - Noen ganger trenger vi å illustrere konsepter visuelt i tillegg til verbalt
  - Tekniske konsepter lar seg godt beskrive med modeller av ulike slag.
  - Vi modellerer gjerne hurtig og udetaljert sammen på et whiteboard (**model storming**).

- Dokumentasjon
  - For ettertiden
  - En modell inneholder mye komprimert informasjon
  - Uttrykker ofte mer med en modell enn med mange sider i et dokument
  - En modell er mer kompakt og derfor lettere å vedlikeholde enn et langt dokument

Model Storming

- Vi kan modellere sammen i møter eller små arbeidsøkter for å løse et problem der og da.
- Det perfekte verktøy er da et whiteboard.
- Vi lager enkle, overordnete modeller hurtig.
- Digitale verktøy kan brukes, men det kan ta litt mye tid og derfor ta fokus bort fra den gode diskusjonen.
- Modellen blir ikke perfekt og må ikke nødvendigvis tas vare på.
- Ofte tar man likevel bilde av tavla og arkiverer det.
- Model storming er vanlig praksis i agile prosjekter (Scrum, Kanban).



[http://agilemodeling.com/]

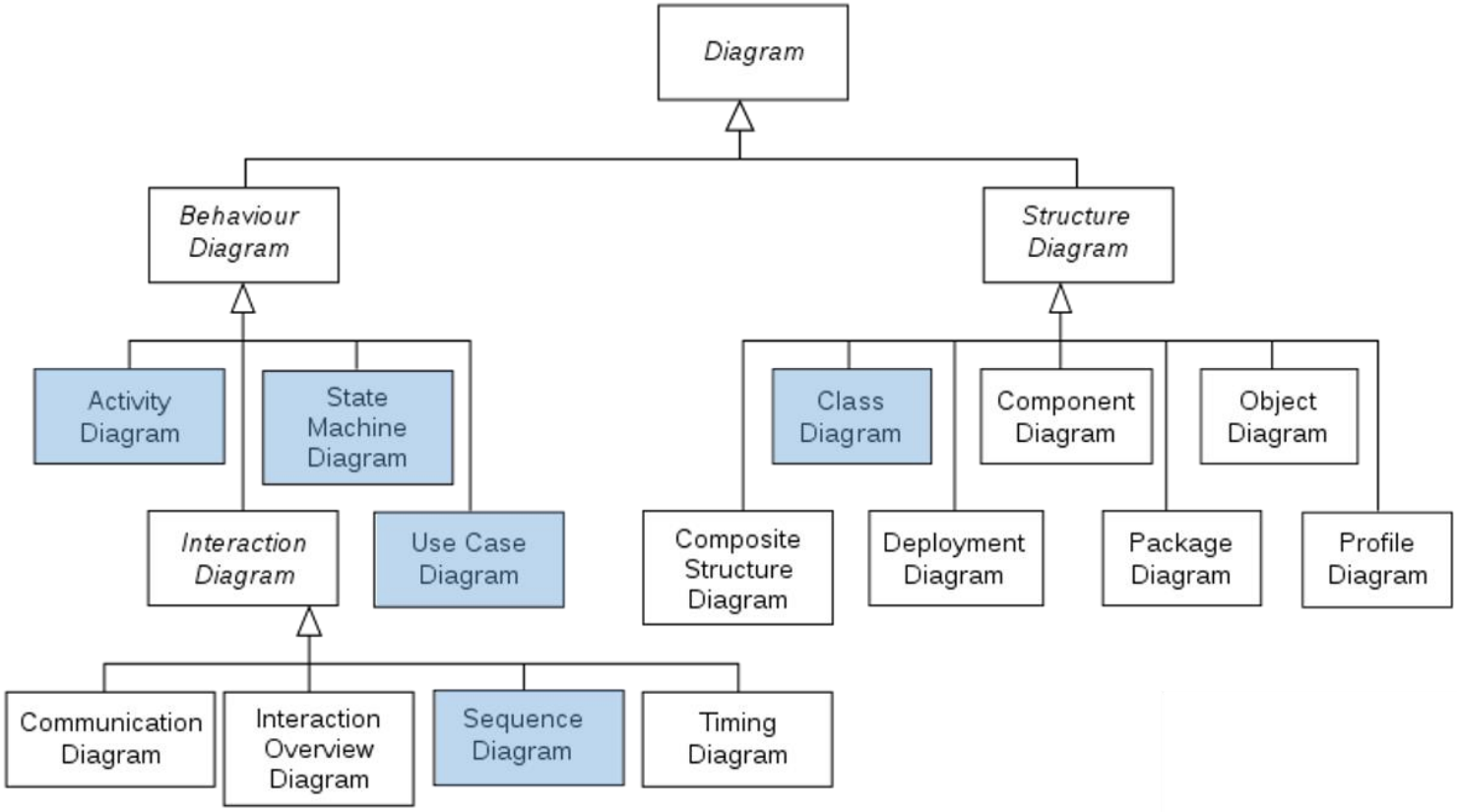
NTNU
Kunnskap for en bedre verden

Dokumentasjon

- Vi kan også lage mer detaljerte modeller som dokumenterer hvordan ting er gjort i et prosjekt.
- Dette gjøres gjerne for å dokumentere for ettertiden. Dette gjerne for at andre skal kunne ta over et prosjekt eller gjøre vedlikehold.
- Her vil gjerne en utvikler eller arkitekt modellere ved hjelp av et avansert verktøy, f.eks Visual Paradigm.
- Dette er vanlig praksis ved bruk av «**Unified Process**».

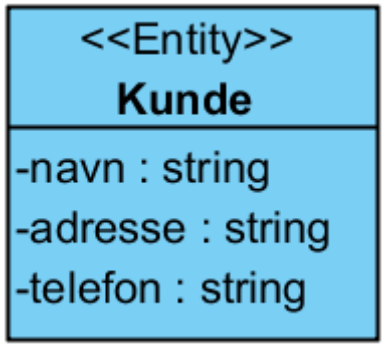
UML

Vi skal gå igjennom 5 av totalt 14 forskjellige diagrammer i UML:



Stereotyper

- Stereotyper er gjennomgående for alle typer objekter i alle diagrammer i UML. Det sier noe om type objekt, altså en **klassifisering**.
- Det er mer eller mindre det samme som arv, men brukes gjerne når det blir for omstendelig å bruke arv.
- En entitet kan for eksempel representeres med ***<<Entity>>***-sterotypen. Man kan også legge til egendefinerte stereotyper for å klassifisere objektene ytterligere.



Verktøy

Det finnes mange verktøy på nett. Disse er som regel ikke gratis, men har ganske billige abonnementer:

- LucidChart
- Gliffy

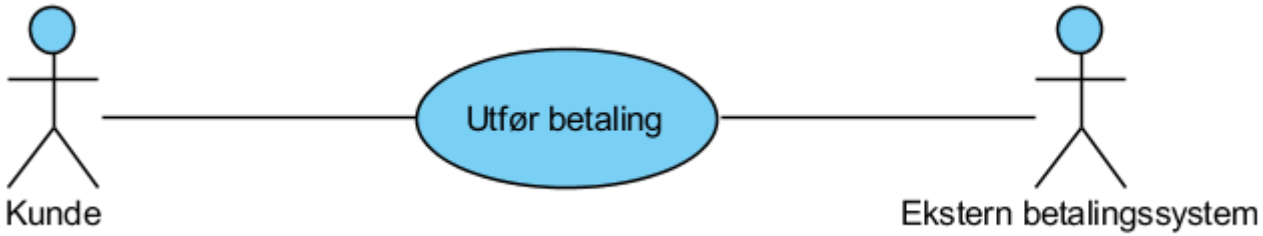
Det finnes også noen mere avanserte desktop-verktøy:

- Visual Paradigm (Community Edition som jeg bruker er gratis) https://www.visual-paradigm.com/download/community.jsp
- Visio med UML template (Dere har lisens?)

Use Case (Brukstilfelle)

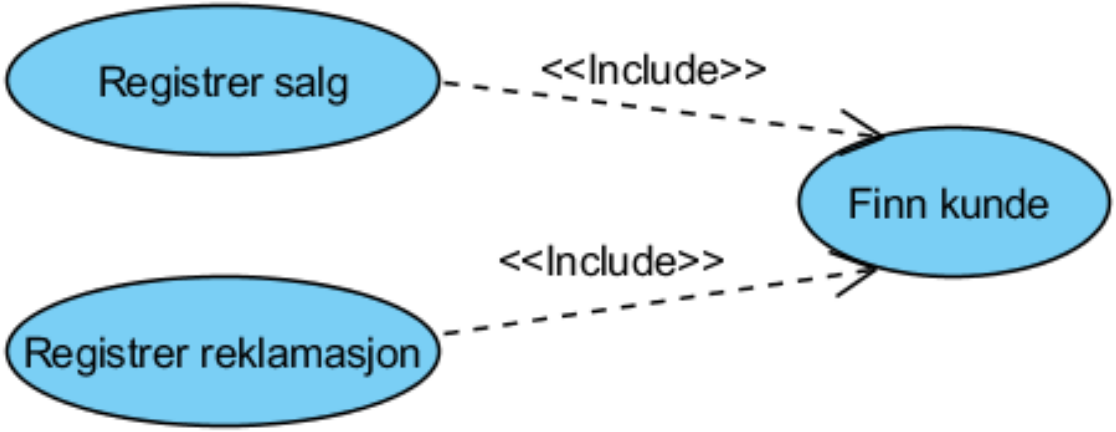
Et use case beskriver en typisk og spesifikk brukssituasjon av systemet. I modellen representeres den med en oval sirkel med en tittel som **alltid** inneholder et verb.

I tillegg ønsker vi å si hvem som er assosiert med et use case. En **aktør** er typisk en bruker av et system, gjerne gruppert etter roller, men kan i noen tilfeller også være et eksternt system som er involvert. En aktør assosieres med et use-case ved hjelp av en enkel linje uten piler (association):



<<Include>>

Vi kan øke detaljeringsgraden ved å bruke inkluderte use case. Vi sier da at et use case er en obligatorisk del av en annen use case.



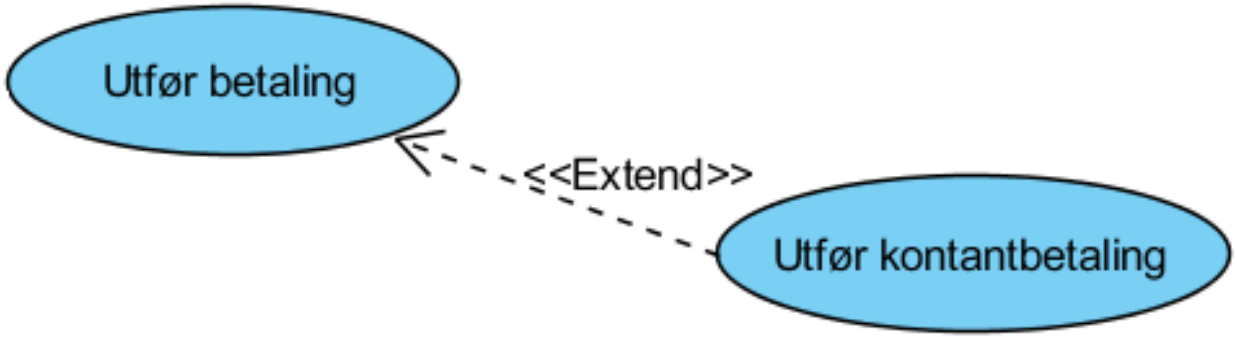
På denne måten kan vi også **gjenbruke** use case. Typisk kan søkefunksjonalitet brukes i flere forskjellige andre use case.

Merk at dette ikke sier noe om rekkefølge, kun at for å kunne registrere et salg må vi også finne en kunde.

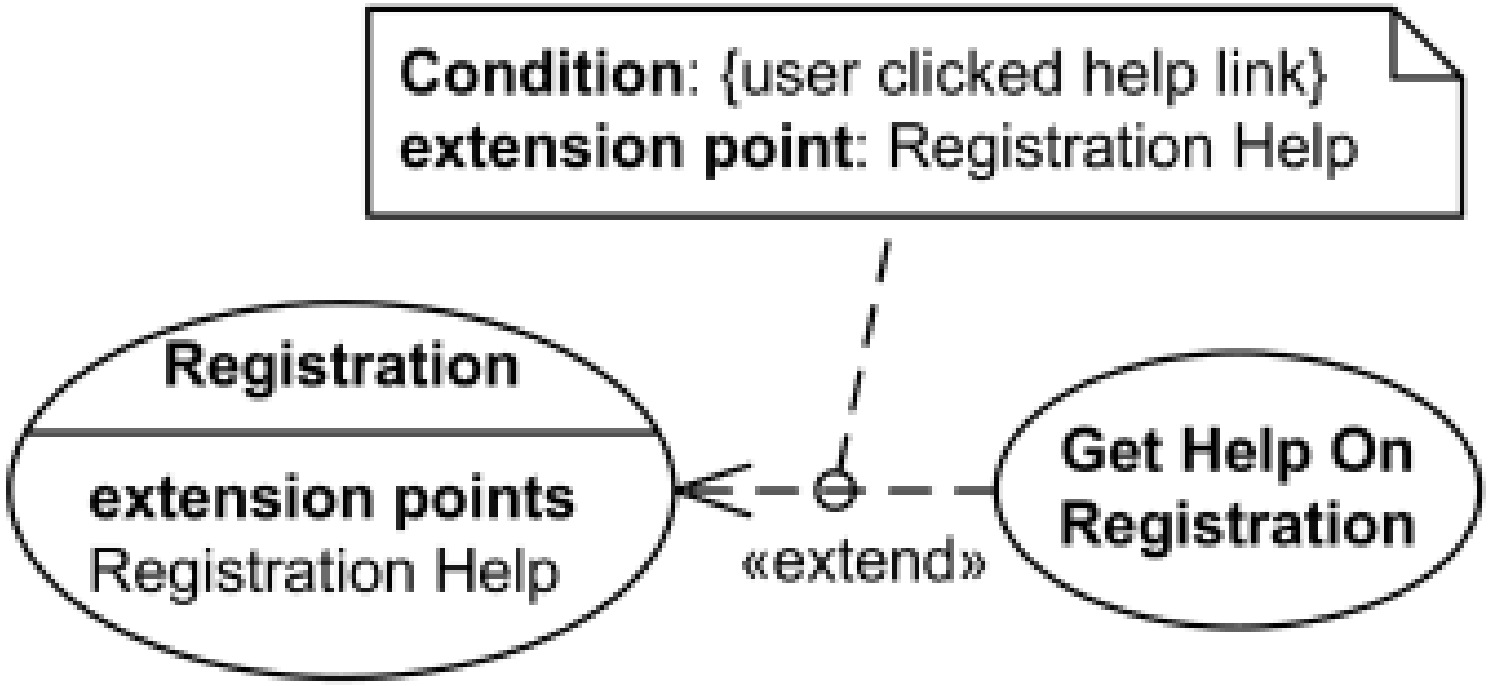
<<Extend>>

Vi kan også bruke utvidelser, use case relatert ved bruk av <<extend>>. Nå er ikke det relaterte use caset en obligatorisk del av et annet, men noe som blir utført i spesielle tilfeller. Utvidelsen har derfor gjerne en betingelse assosiert med seg. Hvis denne oppfylles skal også tillegget utføres. Merk at pila går motsatt vei. Tilleggs-caset *utvider* base-caset.

Vi kan for eksempel se for oss en situasjon der en betaling ikke går igjennom og en kunde må gjennomføre en kontantbetaling.

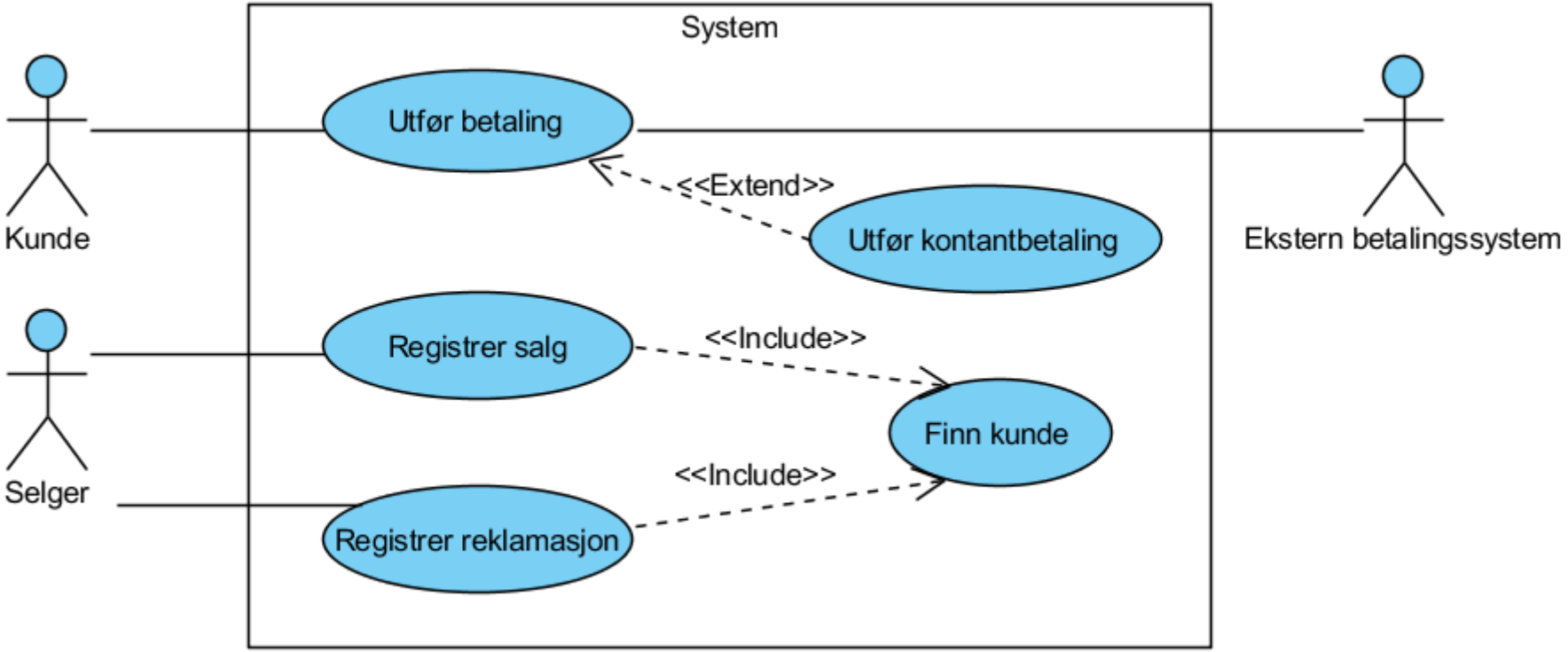


<<Extend>> med betingelse

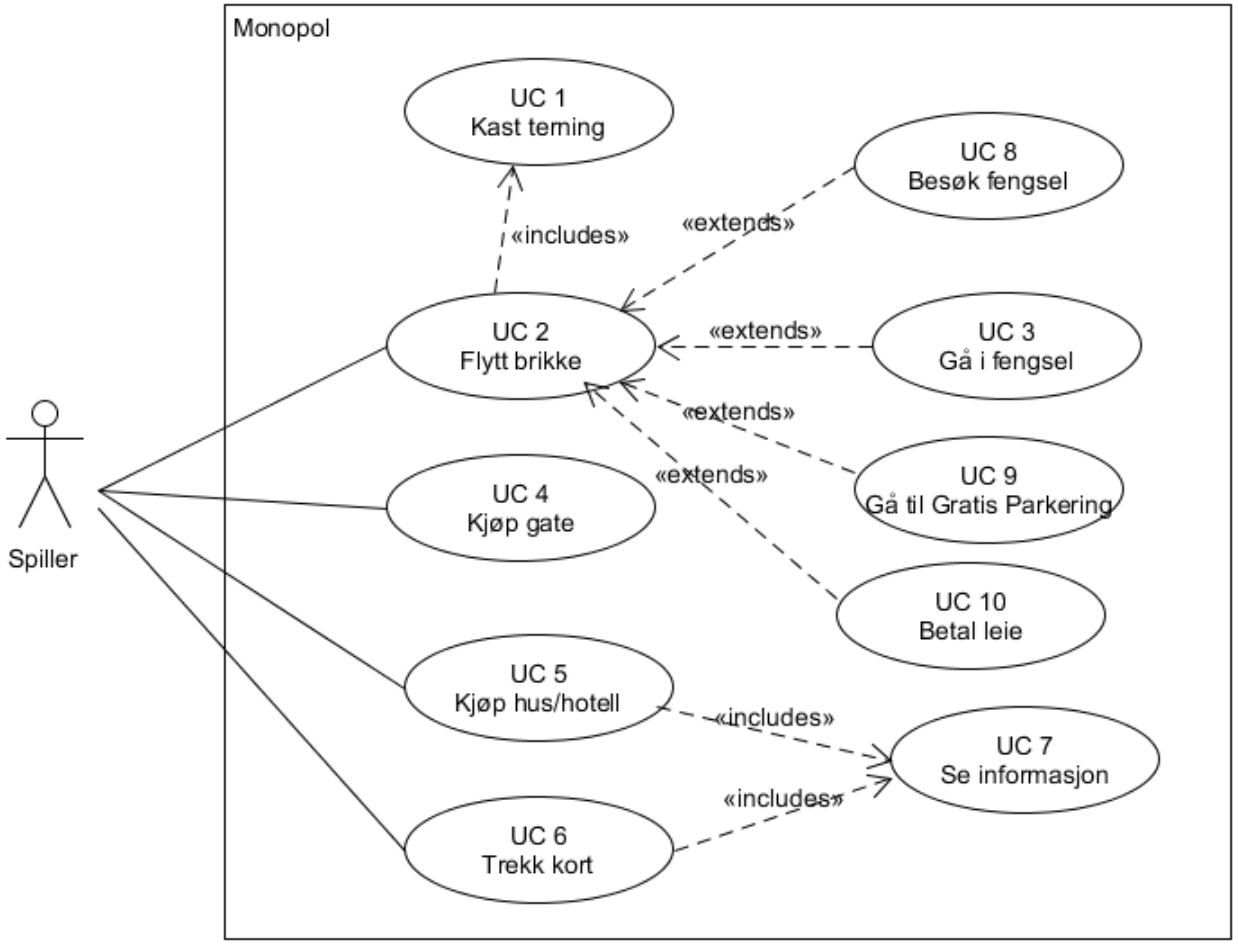


Systemgrense

Vi kan lage en systemgrense som forteller hvilke use case som tilhører et system. Dette vil typisk være kun ett system og nettopp det systemet vi skal lage, men det kan i mer komplekse tilfeller være snakk om flere systemer.



Use Case - Eksempel



[Grethe og Larman]

Use Case - Tekstlig beskrivelse

Vi beskriver hvert use case i detalj for å beskrive kravene til systemet.

Vi kan bruke følgende mal:

Navn	Sett navn («Flytt brikke») og evt nummer («UC 2») på UC
Mål	Hensikten med UC (å flytte brikke)
Aktør	Hvem initierer en UC («spiller», etc.)
Trigger	En situasjon eller hendelse som utløser en UC
Pre-betingelse	Beskriver hva som må være oppfylt for at en UC skal kunne utføres
Post-betingelse	Beskriver tilstanden til system og eventuelt til aktør etter at en UC er utført, både ved hovedløp og sideløp
Hovedløp	Normal hendelsesflyt; hva skjer når alt går bra – når målet til aktøren oppnås på enklest mulig måte
Sideløp	Beskriver hendelsesflyt inkludert fra hovedløp («include» i UC-diagram)
Unntak	Beskriver unntaksvis hendelsesflyt fra hovedløp («extend» i UC-diagram)
Tilleggsinformasjon	Ikke-funksjonelle krav; for eksempel «svartid må være bedre enn 10 sekunder»

Hovedløp

- Beskriver hendelser punktvis og nummerert, en hendelse per linje

1. Spiller velger kjøp hus/hotell
2. Spiller velger hvilke gate-gruppe som skal bygges på
3. Spiller blir presentert med informasjon via UC Vis informasjon
4. Spiller velger antall hus/hotell

- Vi beskriver som regel ikke GUI-spesifikke detaljer som knapper og felter, men opererer på et litt høyere nivå.
- Ved sideløp og unntak (<<include>> og <<extend>>), «forker» vi flyten. Her vi bruker vi nummeret fra hovedflyten og legger til undernummer

4.1 Spiller velger ugyldig antall hus
4.1.1 Spillet gir tilbakemelding til spiller om at det ikke er nok tilgjengelige hus
4.1.2 Gå tilbake til trinn 4 og velg nytt antall

- I sideløp og unntak kan vi eventuelt referere til andre use case. I eksempelet er unntakene ikke egne use case, men beskrives i sin helhet under «Unntak».

Use Case - Tekstlig beskrivelse - Eksempel

UC-navn	**Kjøp hus/hotell**
Aktør	Spiller
Pre-betingelse	1. Det er spillerens tur 2. Spilleren har ikke kastet terningene 3. Spilleren eier alle gatene i samme farge
Post-betingelse	Spiller har fått kjøpt hus/hotell
Hovedflyt	1. Spiller velger kjøp hus/hotell 2. Spiller velger hvilke gate-gruppe som skal bygges på 3. Spiller blir presentert med informasjon via UC Vis informasjon 4. Spiller velger antall hus/hotell 5. Spiller betaler 6. Spiller plasser kjøpte hus/hotell på brettet
Unntak	4.1 Spiller velger ugyldig antall hus 4.1.1 Spillet gir tilbakemelding til spiller om at det ikke er nok tilgjengelige hus 4.1.2 Gå tilbake til trinn 4 og velg nytt antall 5.1 Spiller har ikke nok kontanter 5.1.1 Spillet gir tilbakemelding om ikke nok kontanter 5.1.2 Transaksjon avbrytes

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.

Use Case - Oppgave

- Finn aktører for systemet.
- Finn use cases for systemet.
- Lag et use case diagram for systemet.
- (Lag en tekstlig beskrivelse av ett av use casene. Ta med eventuelle pre- og postbetingelser og få med minst én alternativ flyt)

User Stories

User stories er ikke UML og bruk av slike er ikke modellering, men de har tilsvarende funksjon som bruk av use case-modellering. Brukes gjerne som alternativ til use-case modellering i agile prosjekter.

```
Som <rolle>
Ønsker jeg <mål>
Slik at <fordel>
```

Use case't «**Bestill bord**» kan uttrykkes som en user story:

```
Som resepsjonist
Ønsker jeg å bestille bord på vegne av en kunde
Slik at kunden kan få spise på restauranten til ønsket tid
```

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.

User Stories - Akseptansekriterier

For at user story'ene skal fungere godt som kravdokumentasjon bør man også beskrive testbare akseptansekriterier for hver story. Disse tilsvarer på sett og vis den tekstlige beskrivelsen av et use case.

Sl. No.	Description	Acceptance Criteria
1	As a customer, I want to search for a book by title, so that I can find the book I want to buy	a. I can search based on an exact title b. I can search based on words within a title
2	As a customer, I want to add books to my shopping cart, so that I can build a list of books I want to buy	a. I can add books to my cart from search results b. I can edit the quantity of a specified book c. I can remove a book from my shopping cart d. I can proceed to check out from my shopping cart

User Stories – Scenarioer

Et alternativ eller tillegg til generelle akseptansekriterier er scenarioer.

Scenarioer kan skrives på denne formen (Gherkin):

```
Scenario: Tittel
Gitt [kontekst]
  Og [mer kontekst]...
Når  [hendelse]
Så   [resultat]
  Og [mer resultat]...
```

Vi bruker nøkkelordene «**Gitt**», «**Og**», «**Når**» og «**Så**» eller «**Given**», «**And**», «**When**» og «**Then**» hvis vi dokumenterer/modellerer på engelsk.

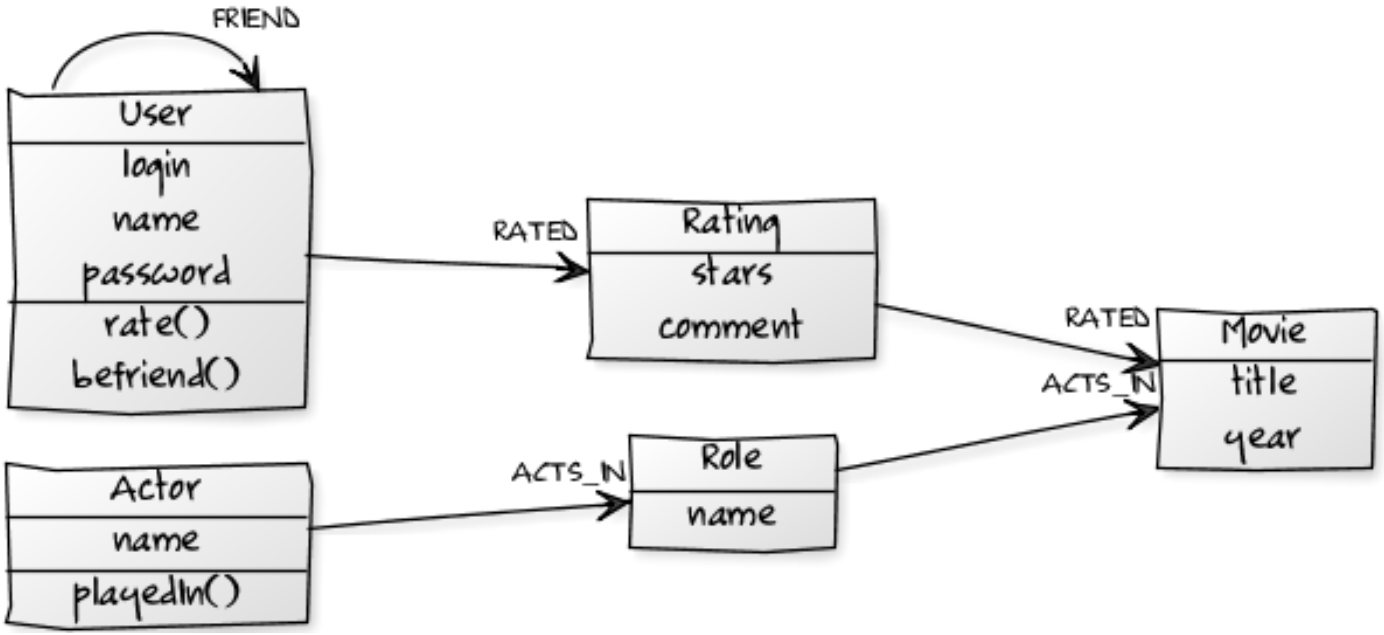
User Story med scenarioer - Eksempel

```
Som spiller
Ønsker jeg å kjøpe hus/hotell
Slik at jeg kan vinne spillet

Scenario: Happy days
Gitt at jeg velger å kjøpe hus/hotell
  Og velger hvilke gate-gruppe som skal bygges på
Når jeg velger antall hus/hotell
  Og betaler
Så skal kjøpte hus/hotell passeres på brettet

Scenario: Ugyldig antall hus/hotell
Gitt at jeg velger ugyldig antall hus
Så skal spillet informere om at det ikke er nok tilgjengelige hus
  Og jeg må gå tilbake og velge nytt antall

Scenario: Ikke nok kontanter
Gitt at jeg ikke har nok kontanter
Så skal spillet informere om at jeg ikke har nok kontanter
  Og transaksjonen skal avbrytes
```

Klassediagram

Klassediagrammet kjenner dere! Vi skal nå se på en spesifikk anvendelse av klassediagrammet, nemlig **domenemodellen**.

Vi finner **entiteter** i et **problemdomenet** og viser forholdet mellom dem:



[http://docs.spring.io]

Domenemodell

Domenemodellens funksjon er å få utviklere og domeneeksperter til å enes om en **felles forståelse av problemdomenet**. Utviklerne må forstå domenet de skal jobbe med og alle interessenter må snakke om det samme når man bruker begreper fra domenet. Vi trenger et **felles språk** for å unngå misforståelser i utviklingsprosessen. Navngiving av elementer i modellen er derfor veldig viktig.

En domenemodell skal ikke være detaljert. Den skal vise konseptene i et domene på en oversiktlig måte. Blir modellen for stor mister den sin funksjon, som er kommunikasjon. Husk at den skal kunne benyttes i samtale med ikke-tekniske interessenter.

Domenemodell

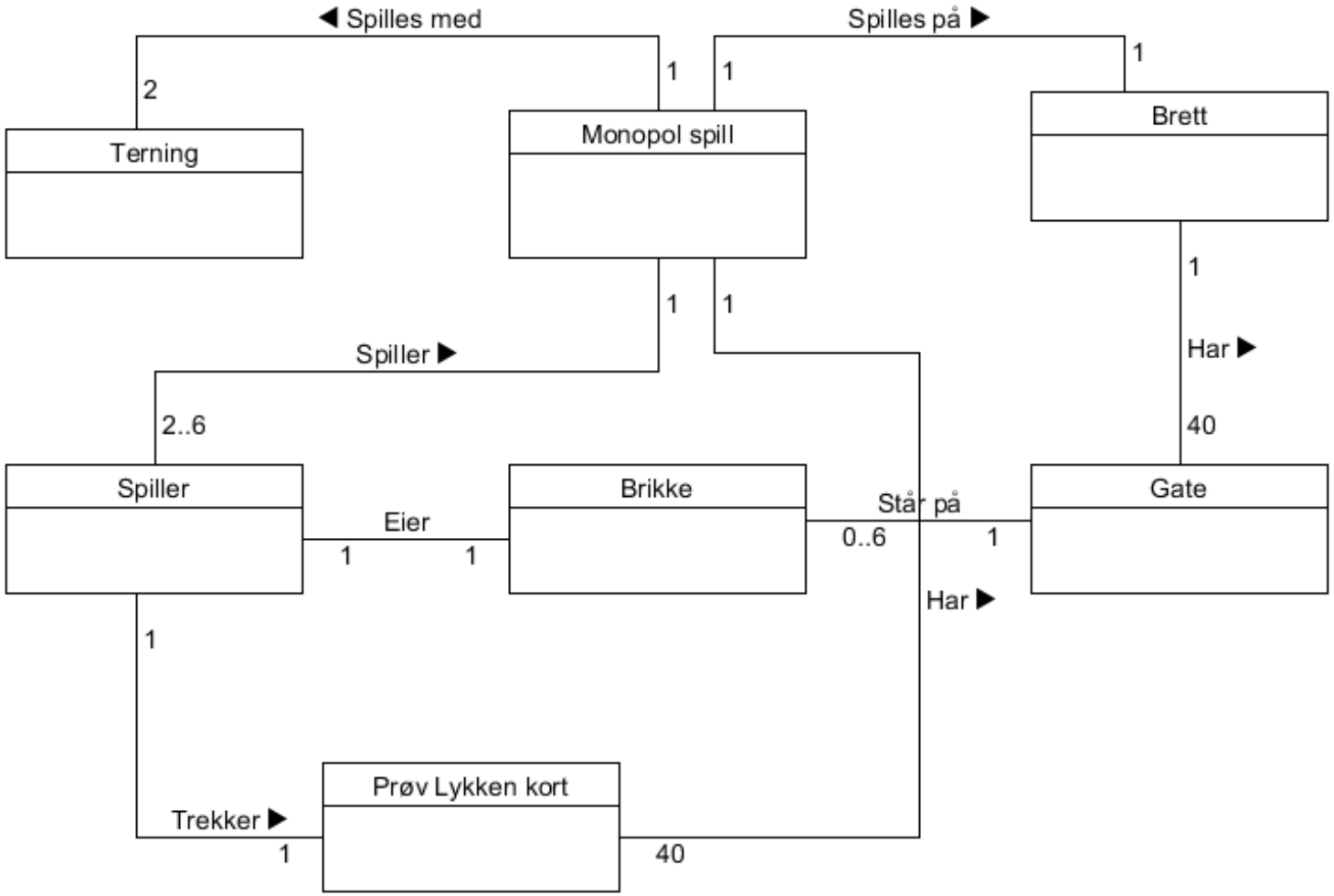
En domenemodell vil ofte ligne på et ER-diagram, men husk at hensikten er en helt annen.

- 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 situasjonsbetinget. 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

Domenemodell - Eksempel



Domenemodell - Oppgave

Lag en domenemodell for restaurantsystemet.