Løsningsforslag: Ukesoppgaver uke 11: Use case og kravhåndtering

Modellering av krav

Formålet med disse oppgavene er å få trening i å beskrive funksjonelle krav ved å modellere use cases, og å få en bedre forståelse av begrepene aktør, use case, use case diagram, tekstlig beskrivelse av et use case og domenemodell.

Oppgave 1

Hva er et use case, og hvorfor er det nyttig å lage use cases?

Kan ses på som en slags historie, som beskriver hvordan systemet oppnår et mål av verdi for en aktør. Viktig at et use case beskriver en komplett funksjonell enhet som kan testes. Vi skiller mellom use case som tekstlig beskrivelse og use case diagram. Vi har med både hovedflyt, alternativ flyt og primær aktør og eventuelt sekundær aktør i et tekstlig beskrivelses use case.

Det er nyttig å lage use case fordi det hjelper oss å se hvilke steg/sekundær aktører som må til for at primær aktør skal nå målet sitt, og hvor det kan oppstå alternativ flyt.

Use case kommuniserer tydelig:

- Hva de forskjellige målene med systemet er
- Hvem som bruker systemet, og til hva systemet skal brukes
- Hvilke andre aktører systemet interagerer med

Oppgave 2

Hva skiller aktører fra interessenter?

Skillet mellom aktør og interessent omhandler rolle/tilknytning til bruken og utviklingen av systemet.

En aktør er en samlebetegnelse for å angi alle grupper av personer som anvender systemet, eller øvrige systemer som blir anvendt av systemet.

En interessent er en betegnelse for alle personer, grupper eller organer, som påvirkes av eller påvirker systemets utvikling/bruk. Både direkte og indirekte.

Aktører må være enten:

- brukere av systemet
- andre systemer som brukes av/bruker systemet

Interessenter kan være:

- brukere av systemet
- inkluderer alle som påvirker/påvirkes av systemets kravspesifikasjon og utvikling

Oppgave 3

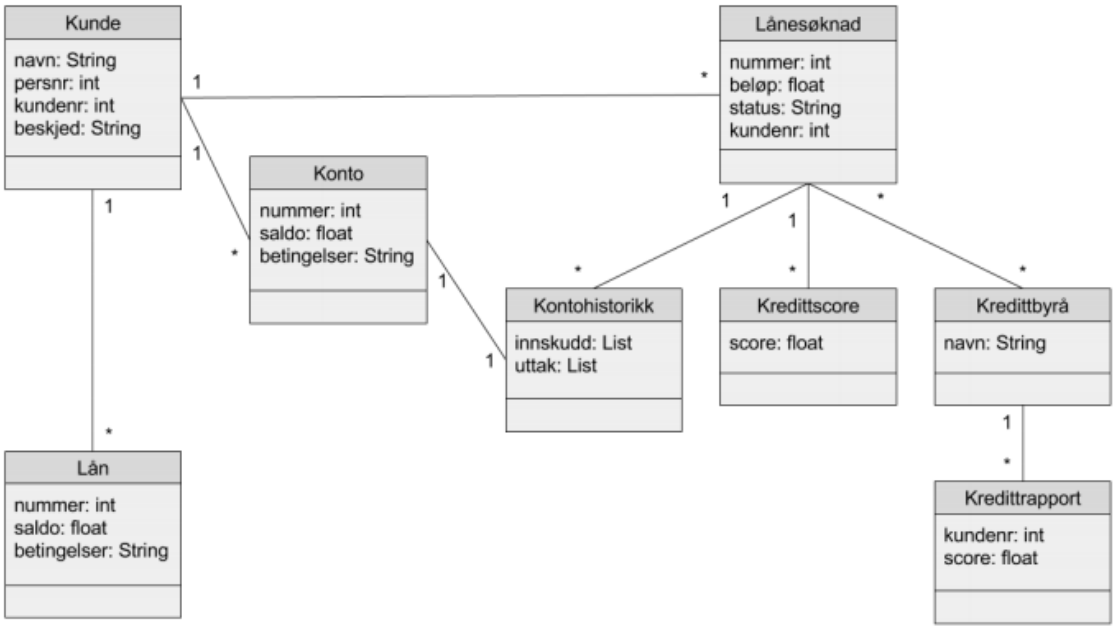
Hva er en aktør i et use case diagram, og hva er forskjellen på en primær og sekundær aktør?

I et use case diagram er aktøren strekfigurene. Disse må ha et rollenavn. Primæraktører har et mål av verdi i systemet. Sekundæraktører realiserer målene til primæraktørene.

Eks:
UML: Kunde (primær aktør) setter inn penger
UML: Bank (sekundær aktør) sørger for at pengene blir satt inn

Oppgave 4
Hva er en domenemodell, og hvorfor er det nyttig å lage en domenemodell for et gitt system?

En domenemodell er en representasjon av de ulike objektene i (domenet) et system består av. Modelleres på samme måte som et klassediagram, men uten metoder. Det er nyttig for å: kartlegge hvilke objekt som man bør ta hensyn til; kommunisere at man har forstått domenet; se på relasjoner mellom objektene.



Oppgave 5
Anta følgende beskrivelse av et system som skal håndtere bord og bordbestillinger på en restaurant:

Systemet skal støtte 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.

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.

a) Finn aktører for systemet.

Marker de du ser har et mål / hjelper til med å nå mål i teksten:
Resepsjonist: motta samtaler og håndtere bestillinger
Hovmester: plassere gjester
Videre kan man generalisere disse to til "Ansatt": se oversikt over bestillinger/plasseringer

b) Finn use cases for systemet.

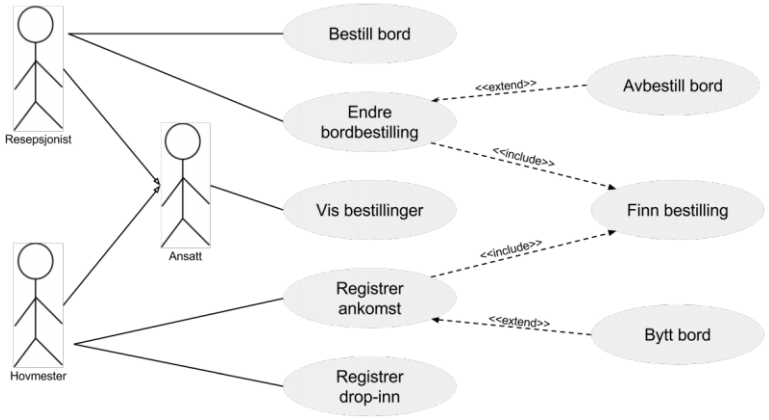
Hva skal aktørene kunne gjøre? Hvilke resultater vil aktøren oppnå?
Identifiser de viktigste oppgavene (se etter verb):
- Skape / Lagre / Endre / Lese / Slette



c) Lag et use case diagram for systemet.

"include-relasjonen": Indikerer at et (sub) use case inneholder nødvendig funksjonalitet for gjennomførelsen av et annet basiscase.

"extend-relasjonen": Utvider oppførselen / funksjonalitet til et basiscase, som utføres under spesielle omstendigheter.



d) Anta at en kunde kan bestille bord på nett i tillegg til over telefon. Gjør eventuelle modifikasjoner i løsningene i a), b) og c).

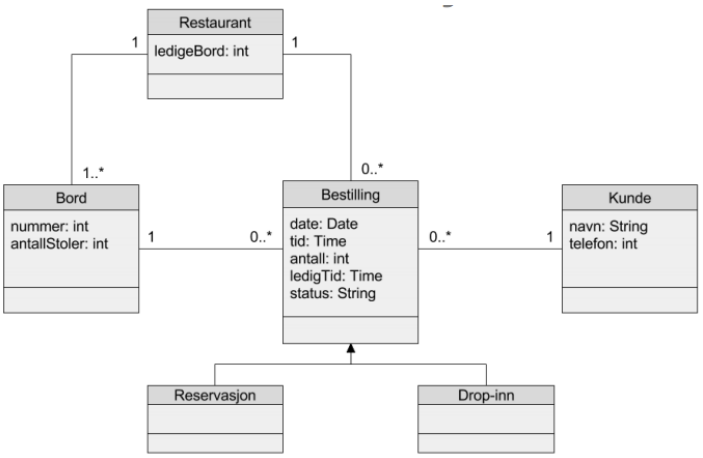
Analyse:

A. Ny aktør **Kunde,** med mål: **Bestille bord på nett**
B. Nytt use case?
    a. Skal man opprette et nytt use case ‘‘Bestill bord på nett?’’
    b. Skal man beholde ‘‘Bestill bord’’?
C. Oppdater use case-diagrammet

e) Lag en enkel domenemodell for systemet.

Domenemodell: klassediagram uten metoder
Hvilke klasser er aktuelle å modellere? Abstraher objekter fra den naturlige verden
For hver klasse:

- Hvilke attributter (variabler) er naturlig å inkludere?
- Hvilket forhold er det mellom klassene? Én-til-én, Én-til-mange, Mange-til-én, Mange-til-mange



Oppgave 6
a) Hva er en tekstlig beskrivelse av et use case?

En tekstlig beskrivelse av en use case tar for seg **interaksjonen mellom systemet og bruker.** Det er en nummerert liste som beskriver interaksjonen steg for steg. En tekstlig beskrivelse skal inneholder følgende:

Navn på use case → Hvilken funksjonalitet dreier det seg om?
Aktør → hvem/hva er det som interagerer med systemet?
Prebetingelse(r) → Hva skal til for å starte use caset?
Postbetingelse(r) → Hva skal til for å avslutte use caset?
Hovedflyt → Hvilke steg inngår i gjennomførelsen av use caset?
Alternativ flyt → Hvilke steg inntreffer ved avvik i hovedflyten?

Obs: viktig at alternativ flyt også termineres, enten ved å gå tilbake til hovedflyten eller ved å avslutte.

b) Lag én tekstlig beskrivelse av ett av use casene du fant i 5 b. Ha med eventuelle pre- og postbetingelser og få med minst én alternativ flyt.

Navn: Bestill bord
Aktører: Resepsjonist
Prebetingelse: ingen
Postbetingelser (mål): Bord er reservert på kunde
Hovedflyt:

1. Kundebehandler ber systemet om å finne et ledig bord på en gitt tid og dato
2. Systemet finner ledige bord
3. Kundebehandler registrerer kundens navn, adresse og telefonnummer
4. Systemet lagrer kundens navn, adresse, og telefonnummer
5. Systemet ber om bekreftelse på reservasjon om bord
6. Kundebehandler bekrefter reservasjon av bord
7. Systemet registrerer reservering av bord på kunde

Alternativ flyt:
2.1. Systemet finner ingen ledige bord
2.2. Returnerer til hovedflyt, steg 1.

Modellering av alternativ flyt – Identifiser avvik i hovedflyten; der hovedflyten har alternativ. Hva skjer i slike tilfeller? Hvordan responderer systemet?

Alternativ til alternativ flyt i oppgaven:
Bruker ber systemet om å finne et ledig bord til en gitt tid og dato:
Systemet returnerer ledig bord → INGEN LEDIGE BORD
Bruker registrerer kundens navn og telefonnummer:
Systemet lagrer kundens navn og telefonnummer → UGYLDIG NAVN/TELEFONNUMMER
Systemet ber om bekreftelse på reservasjon av bord
Bruker bekrefter reservasjon av bord → BRUKER AVKREFTER RESERVASJON

Oppgave 7
Hva er et sekvensdiagram?

Et sekvensdiagram er et interaksjonsdiagram
Modellen viser en interaksjonssekvens mellom objektene som finner sted under et bestemt use case.

Sekvensdiagrammer viser:
- Hvilke objekter som inngår i et bruksmønster
- Interaksjonen mellom objektene/deres rekkefølge
- Data/informasjon som sendes mellom objektene

Oppgave 8
Hvorfor er det nyttig å benytte sekvensdiagram?

Det er viktig å kunne vise hva som skjer/burde skje ved kjøretid av systemet. Altså: Vise den dynamiske oppførselen til et program. Sekvensdiagram gir en oversikt over nødvendige klasser og objekter for å gjennomføre et bruksmønster. Det kan gjøre det enklere å implementere et system. Det er et svært kodenært diagram, som gjør det mulig å generere kode automatisk fra et sekvensdiagram. Det viser også hvordan objektene kommuniserer, og gir oversikt over data som sendes mellom objektene, og rekkefølgen på dataoverføringen.

Oppgave 9
Et bilutleiefirma ønsker et informasjonssystem som kundebehandlerne kan benytte for utleie av biler. Under er det et forslag til en tekstlig beskrivelse av bruksmønsteret Reserver bil. Lag et sekvensdiagram for hovedflyten og én av de alternative flytene i den tekstlige beskrivelse.

Tekstlig beskrivelse av use case 'Reserver bil'
Navn: Reserver bil
Aktør: Kundebehandler
Prebetingelse: Ingen
Postbetingelse: Leiekontrakt for spesifisert bil og kunde med gitte utleiedatoer er opprettet

Hovedflyt:
1. Kundebehandler velger tidsintervall (hentedato og returdato)
2. Systemet returnerer en liste over tilgjengelige biler innenfor de spesifiserte datoene
3. Kundebehandler velger én av bilene.
4. Systemet ber om kundenr og finner kunden i systemet
5. Systemet bekrefter at bilen er reservert for den gitte perioden

Alternativ flyt 2.1: Det finnes ingen tilgjengelige biler i valgt tidsintervall.
2.2. Systemet opplyser om at det ikke er tilgjengelige biler innenfor oppgitt tidsintervall.
2.3. Kundebehandler oppgir et nytt tidsintervall (steg 1) eller avslutter bruksmønsteret.

Alternativ flyt 4.1: Kunden finnes ikke.
4.2. Systemet oppretter ny kunde og returnerer til steg 3.

Analyse av den tekstlige beskrivelsen:
1. Identifiser de aktuelle objektene/aktørene:
   - Hvilket system er det snakk om? -- Reservasjonssystem for utleie av biler
   - Har vi eventuelle undersystemer? -- Bilregister/Kunderegister (avhengig av hvordan systemet er implementert)
   - Har vi eventuelle objekter? -- Kunde (objekter for de ulike kundene)/Kontrakt (objekt for utleiekontrakt)
   - Hvem skal interagere med systemet? → Kundebehandler
2. Oppsett (rekkefølgen gis av flyten)
   - Aktøren (Kundebehandler) plasseres til venstre i sekvensdiagrammet
   - Aktøren interagerer med Reservasjonssystemet
     1. Kundebehandler velger tidsintervall (hentedato og returdato)
        Reservasjonssystemet interagerer…
        Først med Bilregisteret
     2. Returnerer liste over tilgjengelige biler
        Deretter med Kunderegisteret
     4. Finner kunden i systemet
        Kundeobjekt må være på plass før avtalen tegnes
        Kontrakten opprettes til slutt
     5. Systemet bekrefter at bilen er reservert for den gitte perioden

Foreløpig oppsett → Har nå et skjelett for hvordan sekvensdiagrammet skal se ut
Kan starte å modellere den tekstlige beskrivelsen

3. Modeller hendelsesforløpet ved å følge den tekstlige beskrivelsen:
   Rekkefølgen fra hoved- og alternativ flyt bestemmer rekkefølgen i diagrammet
   Hvert steg i den tekstlige beskrivelsen er tilnærmet lik en pil i diagrammet
   Data som sendes mellom objektene reflekteres i diagrammet
   Metodekall (med parametere) → Heltrukket pil
   Returverdier → Stiplet pil
   Create (for å opprette objekter) → Stiplet pil

