Systemutvikling - Modellering av krav

Høgskolen i Oslo og Akershus

01. sept. 2014

Yngve Lindsjørn- yngve.lindsjorn@hioa.no

Temaer i dagens forelesning

- Modellering av krav
- UML diagrammer
  - **Bruksmønster (Use Case)**
  - Sekvensdiagram
  - Domenemodell
  - Aktivitetsdiagram

Gode beskrivelser av krav

er viktig for

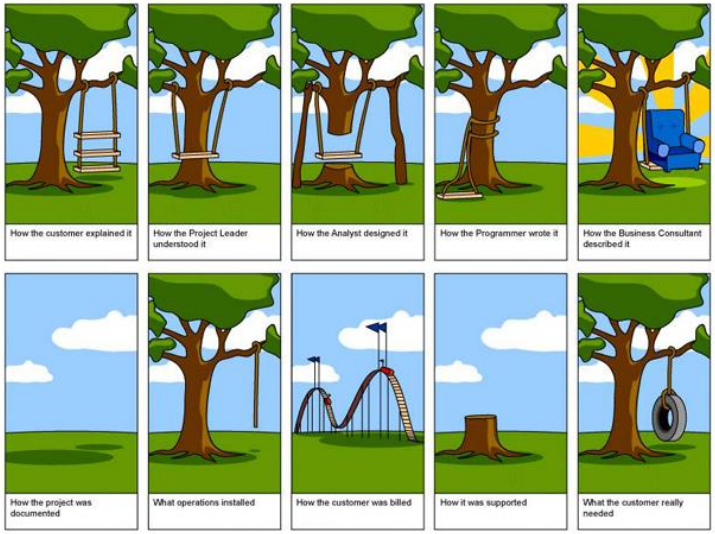
- kontrakt oppdragsgiver – leverandør
- planlegging og oppfølging
- arkitektur, design og test
- å støtte videreutvikling og vedlikehold

Kravene bør være

- forståelige (alle interessenter/stakeholders må kunne forstå kravspesifikasjonen),
- testbare (vi må kunne avgjøre om det ferdige systemet gjør det det skal), og
- sporbare (vi må vite hvilken del av koden som skal endres når det kommer nye krav).

Utfordringer i kravhåndtering

- Kommunikasjon
- Felles forståelse
  - bl.a. av når et krav er realisert?
- Sammenhengen fag-IT
  - Hvis fagsiden dominerer, kan funksjonalitet bli besluttet uten tilstrekkelig innsikt i hva som kan realiseres i systemet
  - Hvis IT dominerer, kan den tekniske sjargongen bli dominerende og dermed blir det vanskelig å få kravene riktige
- Sporbarhet fra krav til arkitektur, design og kode



Krav

- Hva (ikke hvordan)
- Forretningskrav
  - Hva organisasjonen ønsker
- Brukerkrav
  - Hva brukerne ønsker å kunne gjøre
- Systemkrav
  - Hvordan systemet skal oppføre seg sett fra brukers perspektiv

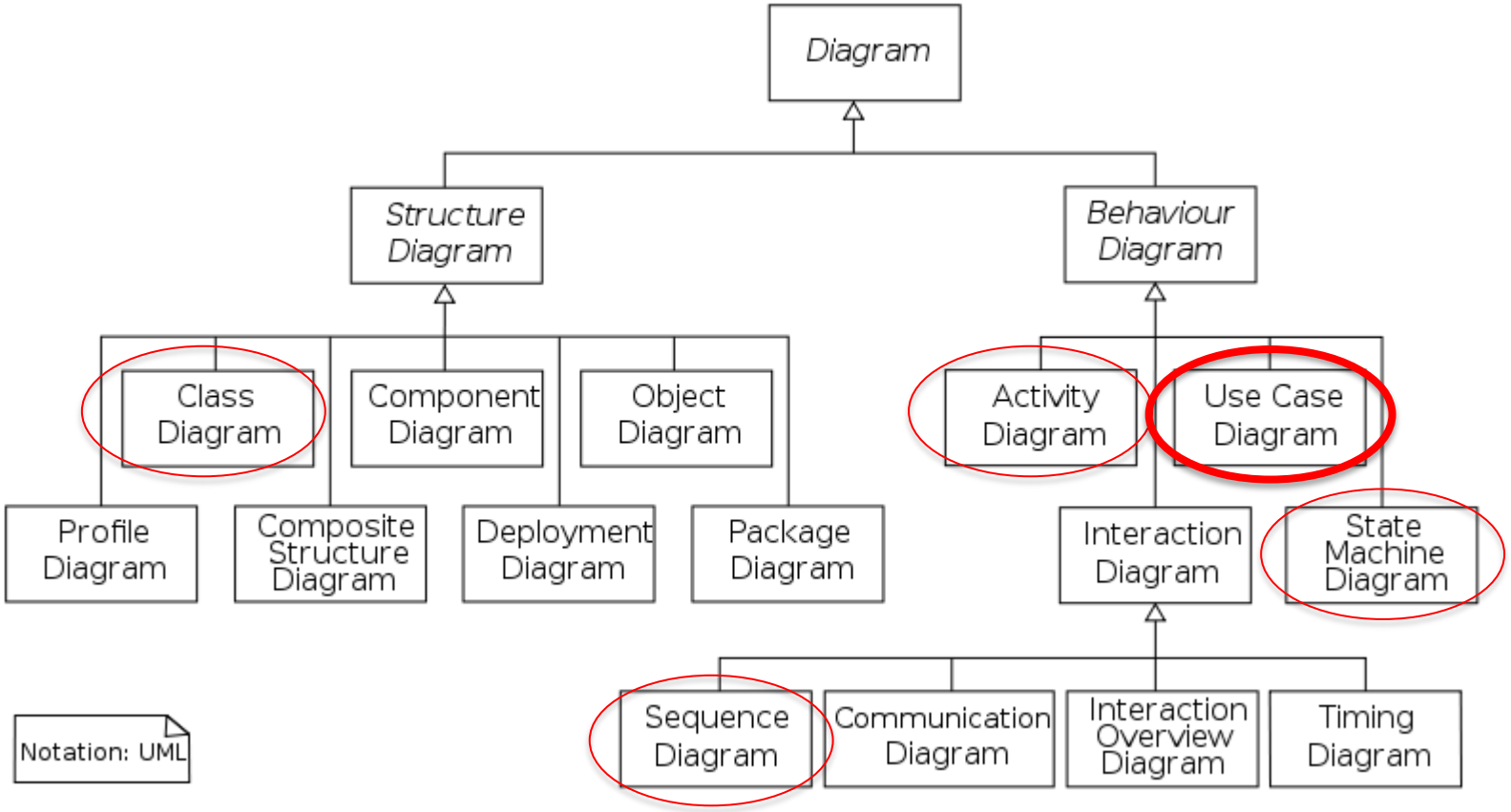
Beskrive krav

- Tekst
- Strukturert tekst
  - User story (brukerhistorie)
  - Use case (brukstilfelle)
- Modeller
  - UML
  - BPMN

Diagrammer i UML

- *Aktivitetsdiagrammer* viser forretningsprosesser og -arbeidsprosesser
- *Use case diagrammer* viser systemets funksjonaliteten og samspillet mellom systemet og omgivelsene (brukere, andre systemer, komponenter)
- *Sekvensdiagrammer* viser samspill mellom system og omgivelser og mellom de forskjellige delene av systemet (mer detaljert enn use case diagrammene)
- *Klassediagrammer* viser objektklassene i systemet og assosiasjonene mellom disse klassene
- *Tilstandsdiagrammer* viser hvordan systemet reagerer på interne og eksterne hendelser

UML - diagrammer



Kilde: http://en.wikipedia.org/wiki/Unified_Modeling_Language#Structure_diagrams

Use case modellering
Identifiser brukere → aktører

- En aktør representerer en rolle som et menneske, et annet system eller en hardwarekomponent har når det kommuniserer med dette systemet
- En aktør kommuniserer med systemet via ett eller flere use case
- Primære aktører har ett eller flere mål med bruk av systemet

Use case modellering
Identifiser brukere → aktører

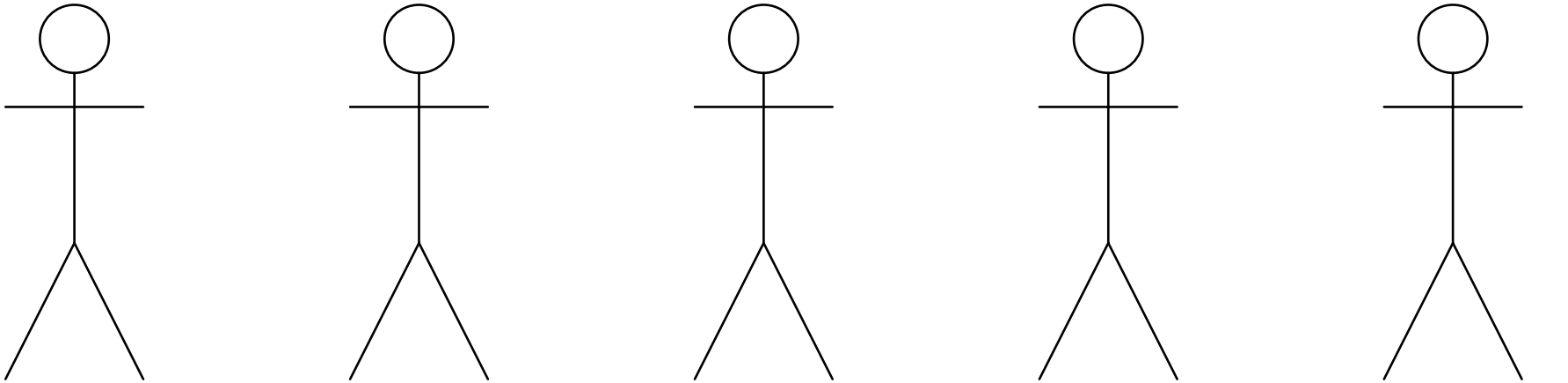
Oppgave:

Finn aktører for et system for behandling av lånesøknader

Use case modellering
Identifiser brukere → aktører

Eksempel:

Aktører for et system for behandling av lånesøknader



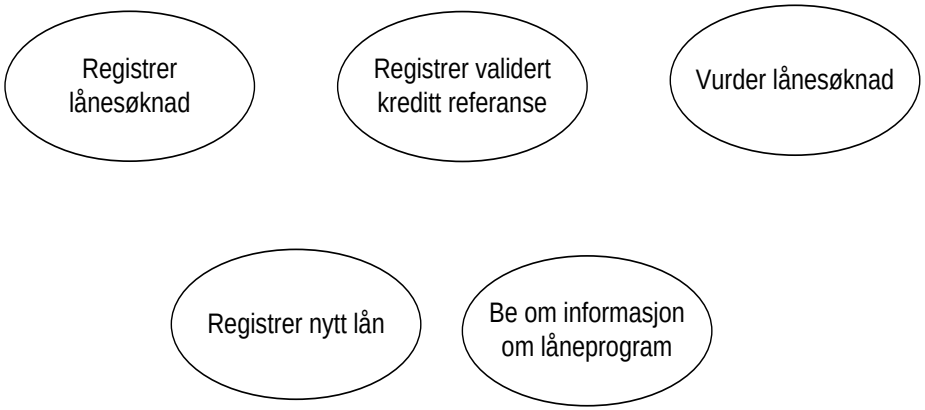
Finne aktører

- I workshop
  - Brainstorming
- Fra tekstlige krav
- Blant prosjektets interessenter
- Spørsmål
  - Hvem skal bruke systemet?
  - Hvem skal administrere systemet?
  - Hvem
    - tilbyr informasjon til
    - bruker informasjon fra, eller
    - fjerner informasjon fra systemet?
  - Hvilke eksterne ressurser skal systemet bruke?
  - Hvilke andre systemer skal kommunisere med dette systemet?

Use case modellering: Identifiser aktørenes mål → Use case

- Et use case beskriver hvordan systemet oppnår et mål av verdi for en aktør
  - En historie
  - Et komplett use case består av flere ulike hendelsesforløp (flyt)
- Et use case beskriver en komplett funksjonell enhet
  - One person – one place – one time
- og er testbart

Eksempel: Use case for lånesystemet



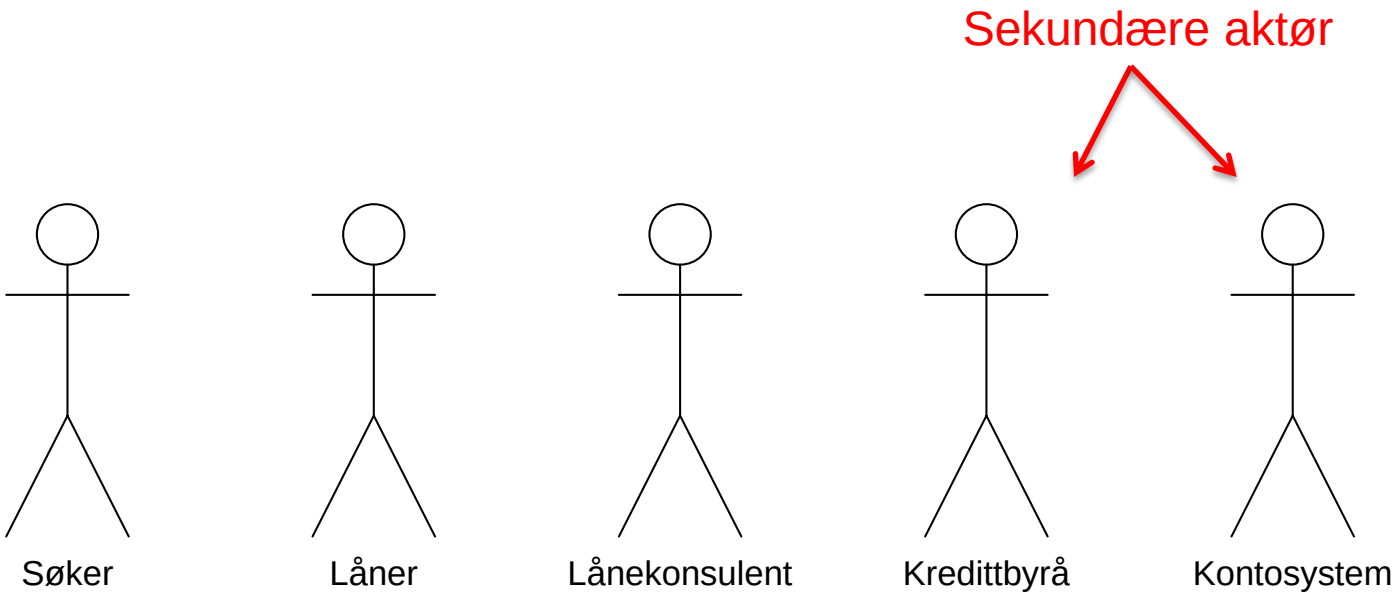
Finne use case

- Kan bruke de samme metodene som brukes for å finne aktører
  - F.eks. gjennomføres workshops, gjerne i flere runder
- Fra prosessmodeller over forretnings- og arbeidsprosesser
- Spørsmål

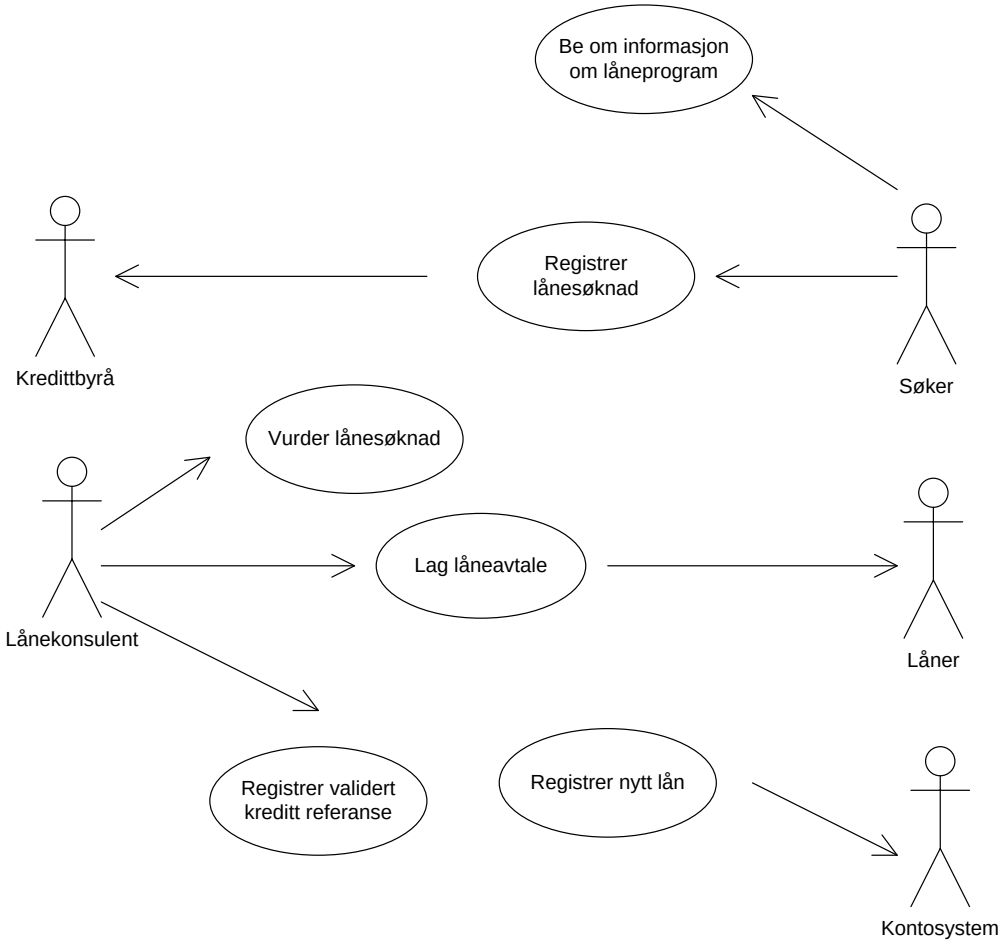
  For hver aktør:
  - Hvilke mål ønsker aktøren å oppnå med bruk av systemet?
  - Hvilke resultater vil aktøren oppnå med bruk av systemet?
  - Hva er de viktigste oppgavene som aktøren ønsker at systemet skal kunne utføre?
  - Vil aktøren skape, lagre, endre, lese eller slette data i systemet?
  - Vil aktøren ha behov for å informere systemet om eksterne endringer?
  - Har aktøren behov for å bli informert om hendelser i systemet?

Use case modellering
Identifiser sekundære aktører

- Aktører som ikke har egne mål, men som er nødvendige for å realisere målene til de primære aktørene



Use case modellering: Tegn use case diagram



Aktører og use case fra tekstlige krav

#	*Krav*	*Aktør*	*Use case*
1	*Systemet skal på oppfordring vise informasjon om alle tilgjengelige låneprogrammer.*	*Søker*	*Be om informasjon om låneprogram*
2	*En lånesøker skal kunne søke om lån online ved å registrere all nødvendig informasjon i et skjema. Informasjonen sjekkes automatisk og gis en kreditt score basert på informasjon fra kredittbyrå samt kundens forhold til banken.*	*Søker, Kredittbyrå, Kontosystem*	*Registrer lånesøknad*
3	*Søkeren skal til enhver tid kunne se status for lånesøknaden.*	*Søker*	*Se status*
4	*Registrerte lånesøknader skal vurderes for godkjenning. Hvis søknaden godkjennes skal dette markeres i systemet og et nytt lån skal opprettes,*	*Lånekonsulent*	*Vurder lånesøknad*
5	*Lånekonsulenten skal kunne registrere kredittinformasjon for lånekunder som valideres manuelt som for eksempel eksterne bankkonti og inntektskilder.*	*Lånekonsulent*	*Registrer validert kredittreferanse*
6	*Låneavtaler skal kunne opprettes og sendes til kunden*	*Lånekonsulent, Låner*	*Lag låneavtale*
7	*Nye lån skal registreres i bankens system for kontooversikter*	*Kontosystem*	*Registrer nytt lån*

Use case modellering: Detaljert beskrivelse av hovedløp

1. Søkeren fyller ut en lånesøknad
2. Systemet validerer informasjonen i lånesøknaden
3. Systemet henter søkerens kontohistorie med banken fra kontosystemet
4. Systemet innhenter kredittrapport for søkeren fra et kredittbyrå
5. Systemet beregner søkerens kreditt score basert på kredittrapport og kontohistorie
6. Systemet informerer søkeren om at søknaden er mottatt og blir vurdert
7. Systemet setter status på lånesøknaden til ”Initiell kredittsjekk ferdig”
8. Systemet legger lånesøknaden i oppgavelisten til en lånekonsulent

Tilleggsinformasjon

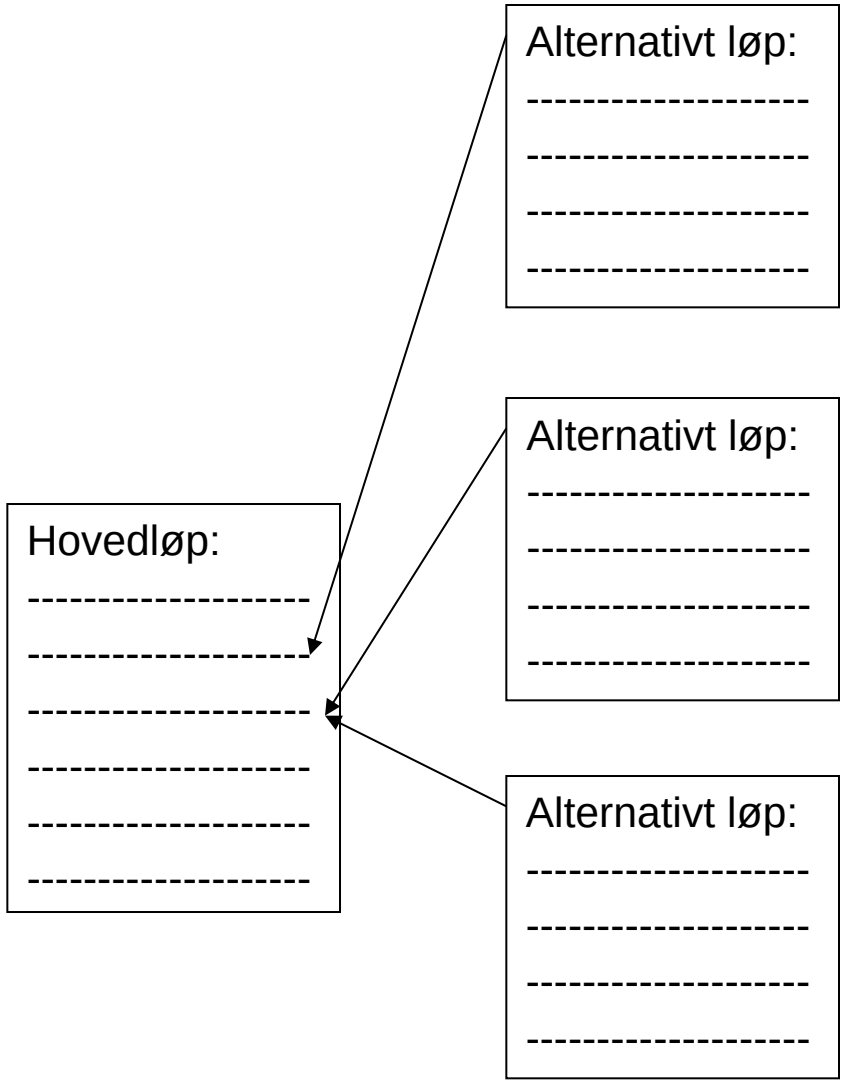
- Input
  - Lånesøker må oppgi: Navn, adresse, telefon, fødselsdato, arbeidsgiver, årslønn og samlet gjeld

- Forretningsregler
  - Lån gis bare til personer med fast jobb

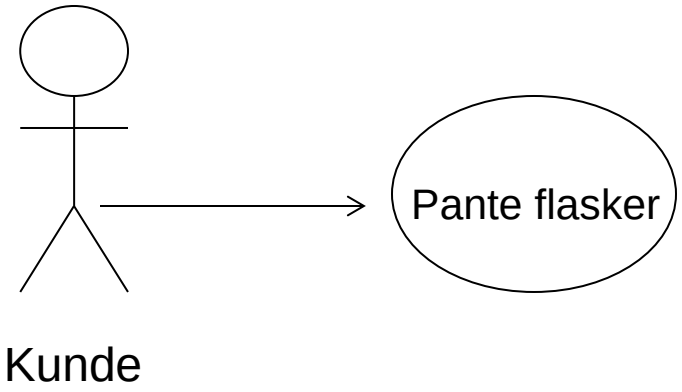
- Ikke-funksjonelle krav
  - Brukervennlighet
    1. Ny bruker skal kunne registrere søknad på mindre enn 10 min.
    2. Mindre enn 1 av 10 brukeres skal avslutte registreringen uten å fullføre.
  - Ytelse
    1. Systemet skal håndtere inntil 100 samtidige brukere

Alternative løp

- Use casene kan oppnå sitt mål på flere måter, og kan feile på flere måter, så detaljerte use case har også
  - Alternative løp, og for hver av disse
  - startkriterier, og
  - postbetingelser
- Alternativer (inkl. feilsituasjoner) er viktige da det ofte er mer ”uenighet” blant prosjektets interessenter om hva som skjer i de tilfellene enn for hovedløpet.
- Ethvert steg i hovedløpet kan være utgangspunkt for et alternativ.



Eksempel-pante flasker



Use case ”Registrer lånesøknad”

Hovedløp:

1. Søkeren fyller ut en online lånesøknad
2. Systemet validerer informasjonen i lånesøknaden
3. Systemet henter søkerens kontohistorie med banken fra kontosystemet
4. Systemet innhenter kredittrapport for søkeren fra et eksternt kredittbyrå
5. Systemet beregner søkerens kreditt score basert på kredittrapport og kontohistorie
6. Systemet informerer søkeren om at søknaden er mottatt og blir vurdert
7. Systemet setter status på lånesøknaden til ”Initiell kredittsjekk ferdig”
8. Systemet legger lånesøknaden i oppgavelisten til en lånekonsulent

Alternativt løp 1, steg 2: Informasjonen i lånesøknaden er ikke komplett og korrekt

A1.1. Systemet returnerer lånesøknaden til søker for ytterligere utfylling.

A1.2. Systemet setter lånesøknadens status til ”Avventer”.

Alternativt løp 2, steg 3: Det er ikke tilstrekkelig kredittinformasjon om søkeren (kredittrapport er ikke tilgjengelig eller er for dårlig)

A2.1. Systemet sender beskjed til søker om at søknaden er avslått

A2.2. Systemet setter lånesøknadens status til ”Avslått”.

Pre- og postbetingelser

Use case ”Vurder lånesøknad”:

Prebetingelse:

Lånesøknaden har status ”Initiell kredittsjekk ferdig”

Postbetingelser:

Lånesøknaden har status ”Godkjent”,

Lånesøknaden har status ”Informasjon mangler”, eller

Lånesøknaden har status ”Avslått” og søker har fått beskjed om at søknaden er avslått

Mer presist v.h.a. domeneobjekter:

Prebetingelse:

Lånesøknad.status = ”Initiell kredittsjekk ferdig”

Postbetingelser:

Lånesøknad.status = ”Godkjent” or

Lånesøknad.status = ”Informasjon mangler” or

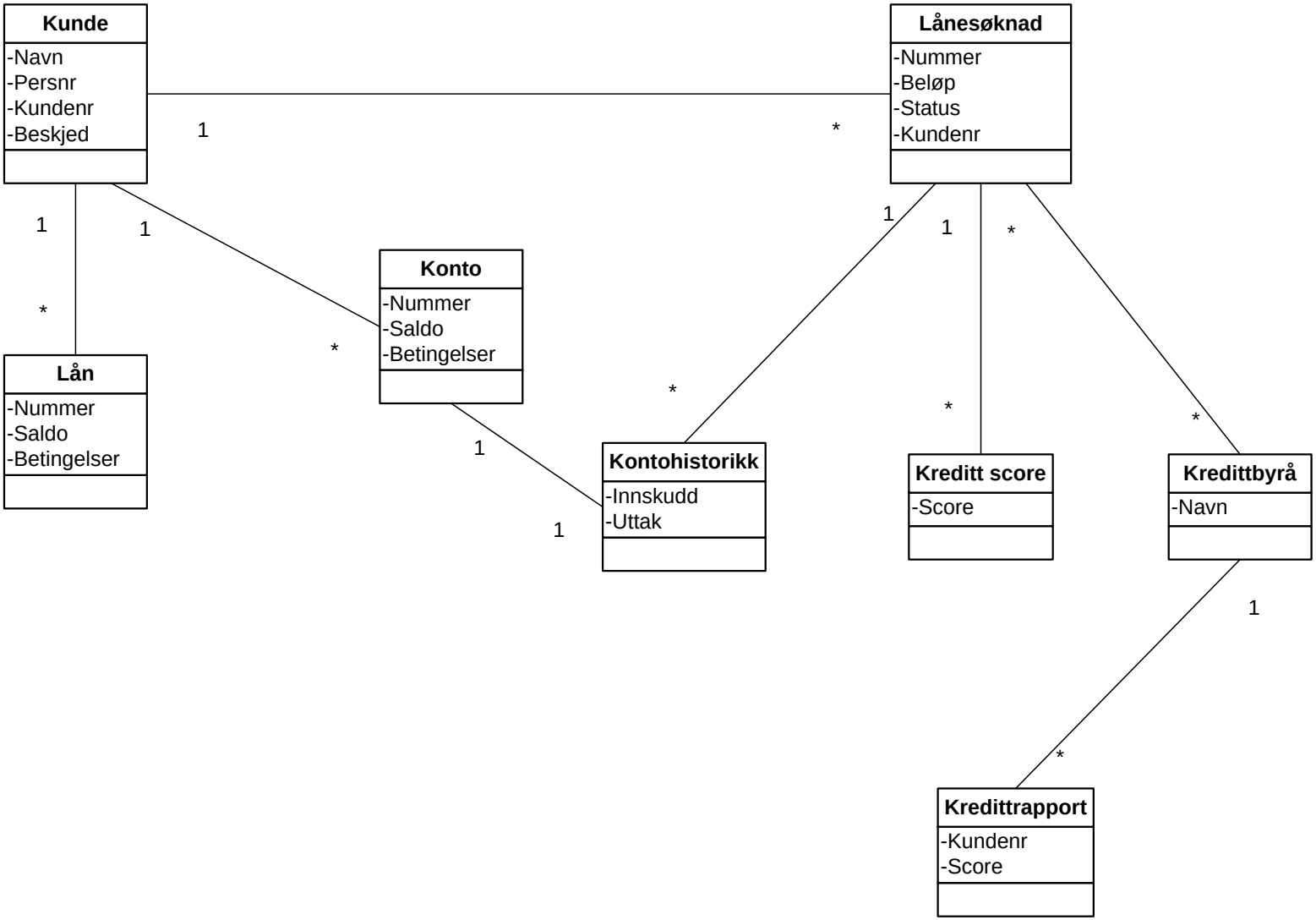
(Lånesøknad.status = ”Avslått” and Kunde.beskjed = ”Sendt”)

Pre- og postbetingelser er bl.a. nyttige i funksjonell test

Domenemodell

- Beskrives v.h.a UML klassediagrammer uten metoder
- Domenemodellen viser objekter i problemdomenet, dvs. objekter som skal håndteres av systemet,
- Hensikten med domenemodellen er å forstå objektene og få en oversikt over terminologi.
- Domenemodellen er nyttig i forbindelse med use case modellering fordi
  - Domenemodellen viser informasjonen om objekter i use casene.
  - Den er et viktig verktøy for å sjekke at use casene er beskrevet med riktig detaljeringsnivå.
  - Klassene i domenemodellen kan brukes i utforming av pre- og postbetingelser for use caset.

Domenemodell forts.



Oppgave

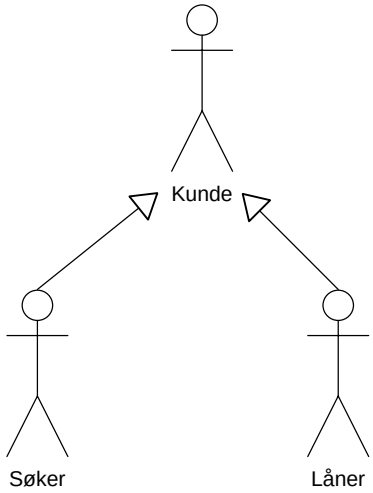
Hvilke av disse setningene er et use case?

1. ”Reserver bok”
2. ”Behandle retur av leiebil”
3. ”Reduser kostnadene”
4. ”Varesalg”

- Hint:
  - Hvem er primær aktør?
  - Hvilke(t) mål oppfyller use caset?

Use case modellering: Relasjoner mellom aktører og use case

- Relasjoner mellom aktører:
  - Definere en generell aktør hvis to eller flere mer spesialiserte aktører har en del felles oppførsel

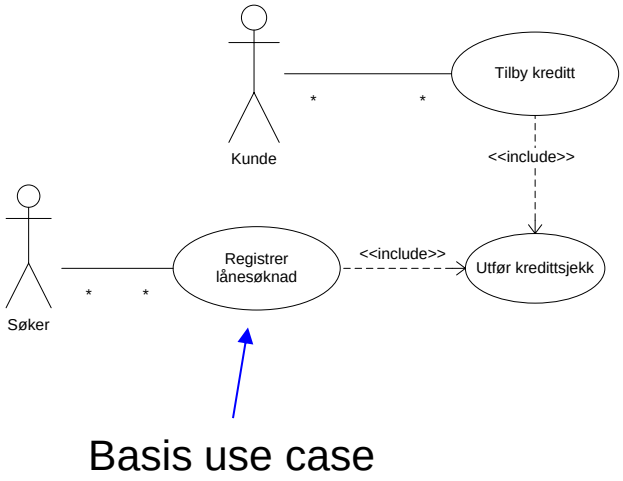


Søker og låner kan gjøre alt det som Kunde kan + mer

- Relasjoner mellom use case:
  - **Include-relasjonen:** Et use case kan være en del av ett flere andre use case.
  - **Extend-relasjonen:** Et use case som beskriver tilleggsoppførsel som utføres under gitte omstendigheter

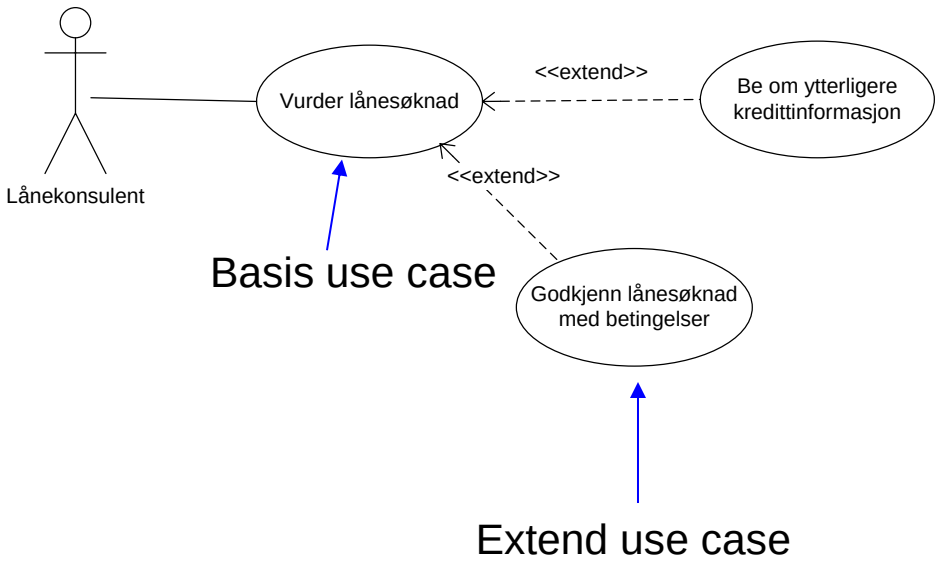
Include-relasjonen

- To eller flere use cases kan ha en felles del (noen like steg). Denne delen kan da legges ut i et eget use case som disse use casene kan inkludere.
  - Include kan også brukes for å forenkle store use case med mange steg
  - Include kan også brukes for å håndtere steg som kan forekomme når som helst i utførelsen av use caset
- Basis use caset vet hvilke use case det inkluderer

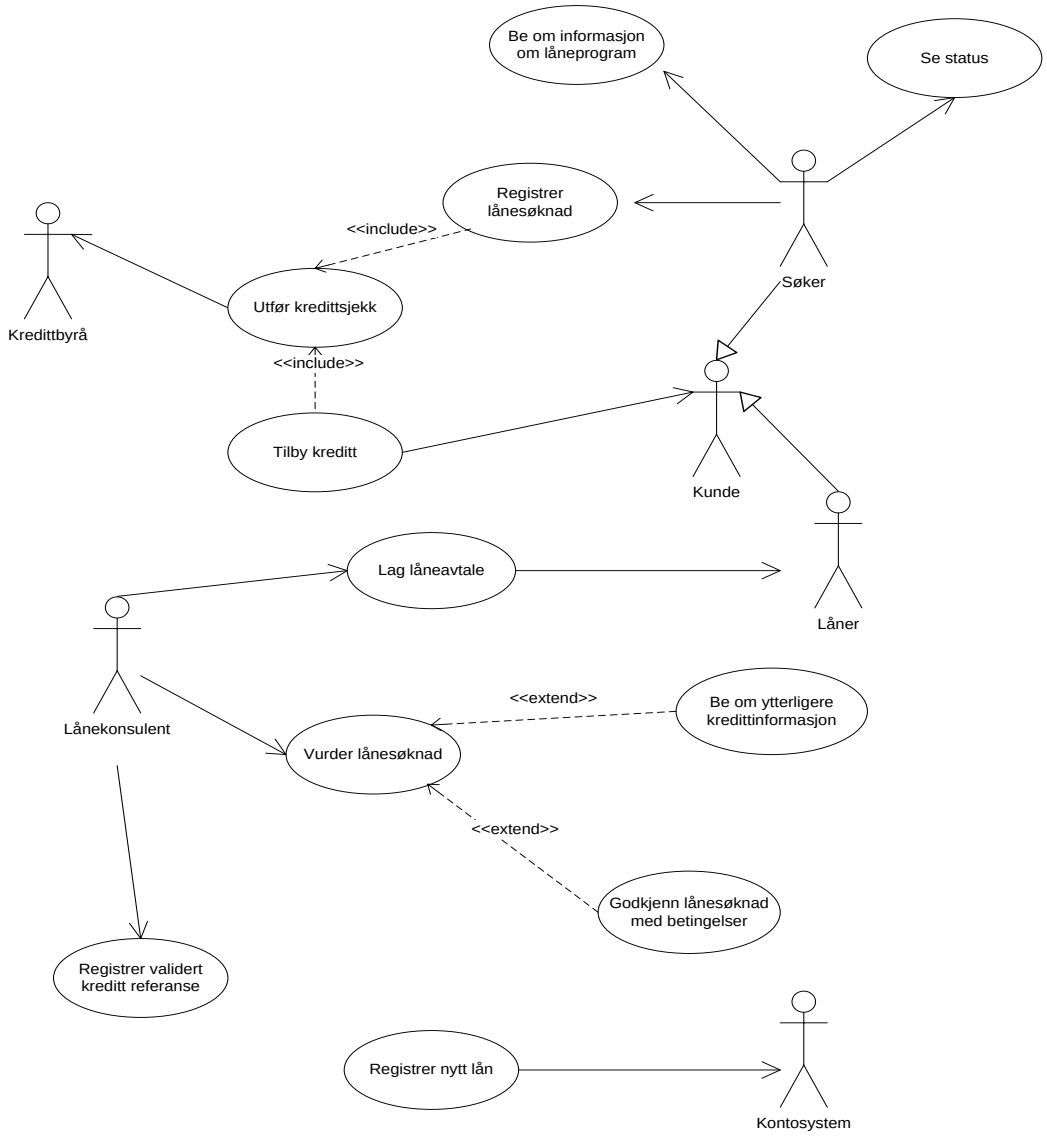


Extend-relasjonen

- Alternativ oppførsel som utføres i noen tilfeller kan skrives som eget use case som utvider (extender) et annet
- Extend use case beskriver hvordan oppnå et tilleggsresultat
- Basis use caset er fullstendig definert uten extensions, disse utvider funksjonaliteten
- Basis use caset kjenner sine extended use case
- Bruk av alternativ flyt vs. bruk av extenduse case:
  - Alternativ flyt beskriver hva som skjer ved avvik i normal flyt, mens
  - Extend use case beskriver hvordan oppnå tilleggsresultat.



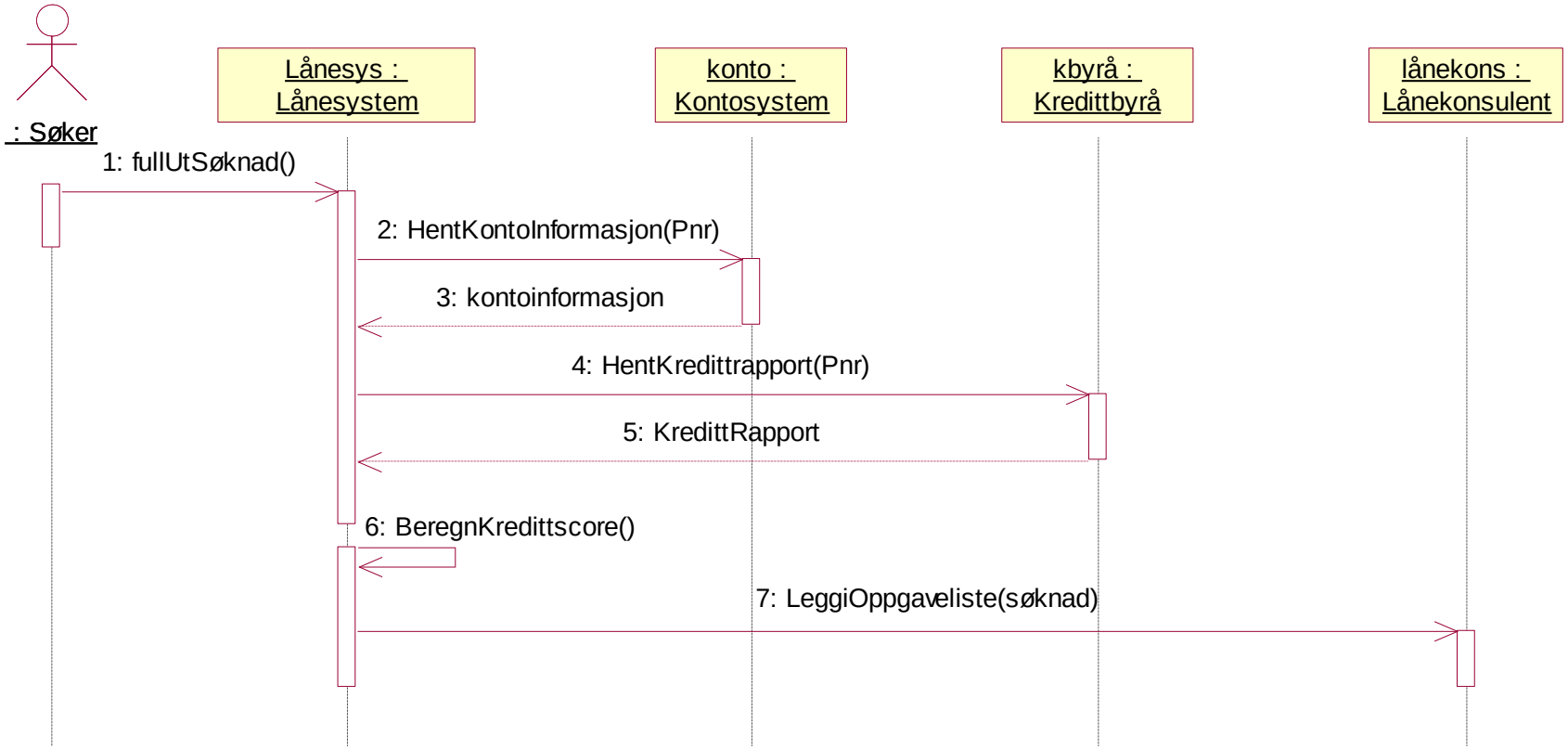
Detaljert use case diagram



Sekvensdiagrammer

- Løpene i et use case kan modelleres med sekvensdiagrammer
- For hvert use case lages typisk sekvensdiagram for hovedløp og for komplekse og hyppig forekommende alternative løp
- Stegene i use casene vises som meldinger som sendes mellom objektene

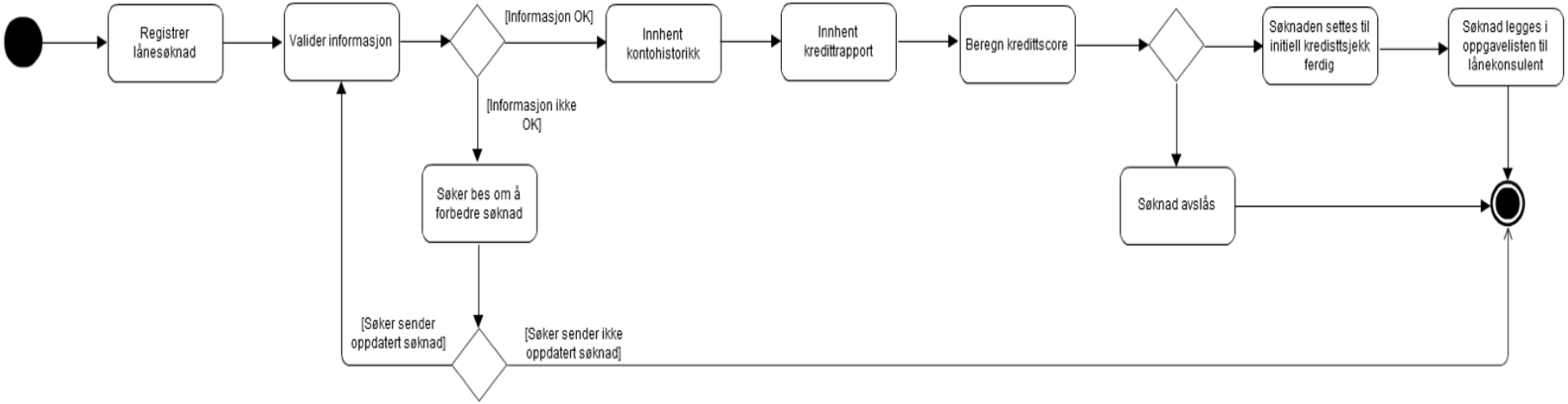
Sekvensdiagram “Registrer lånesøknad”



Aktivitetsdiagrammer

- Et aktivitetsdiagram kan grafisk representere hendelsesflyten i et use case.
- Stegene i use casene vises som aktiviteter
- Beslutninger underveis vises som
- Aktivitetsdiagrammer og sekvensdiagrammer brukes noe overlappende, men sekvensdiagrammer er typisk mer kodenært mens aktivitetsdiagrammer er mer forretningsnært.

Aktivitetsdiagram “Registrer lånesøknad”

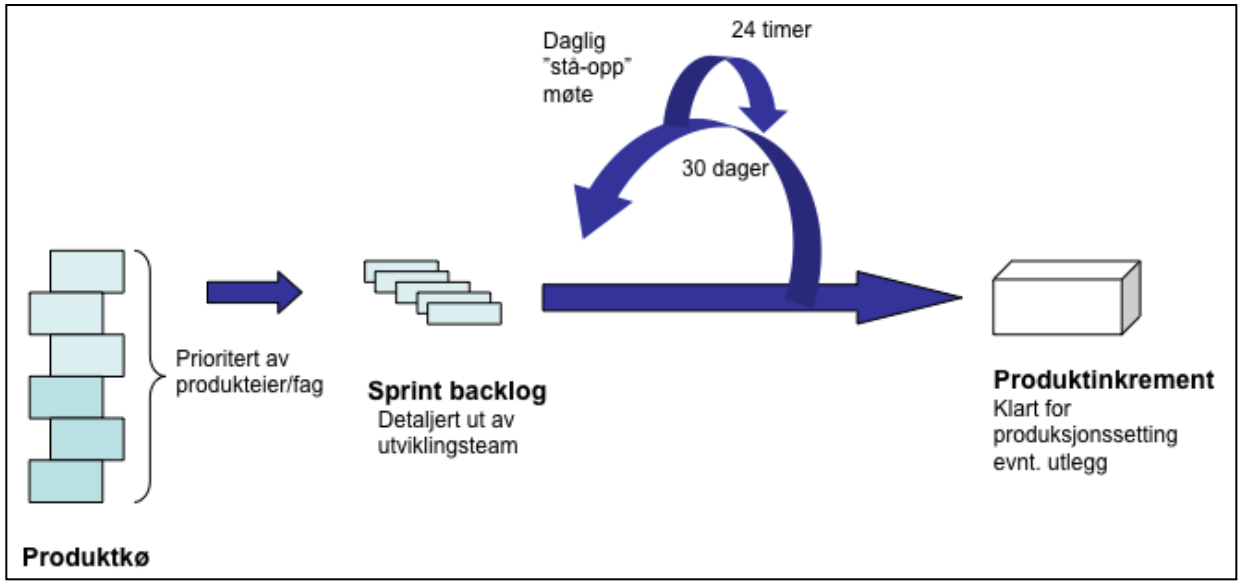


Use case vs. smidig utvikling

- I smidig utvikling jobber produkteier sammen med utviklere i samme team
- Det er mindre behov for detaljerte beskrivelser av krav og ofte brukes user stories (en ”lett” versjon av use case)

  Eks: **Som** lånekonsulent
  ønsker jeg å kunne vurdere lånesøknader
  slik at jeg kan gi en riktig og rask vurdering

- Krav utvikles underveis og beskrives ”on demand”
  - Først tilstrekkelig for prioritering i produktkøen
  - Så tilstrekkelig for prioritering i sprint backloggen



Use case vs. user stories forts.

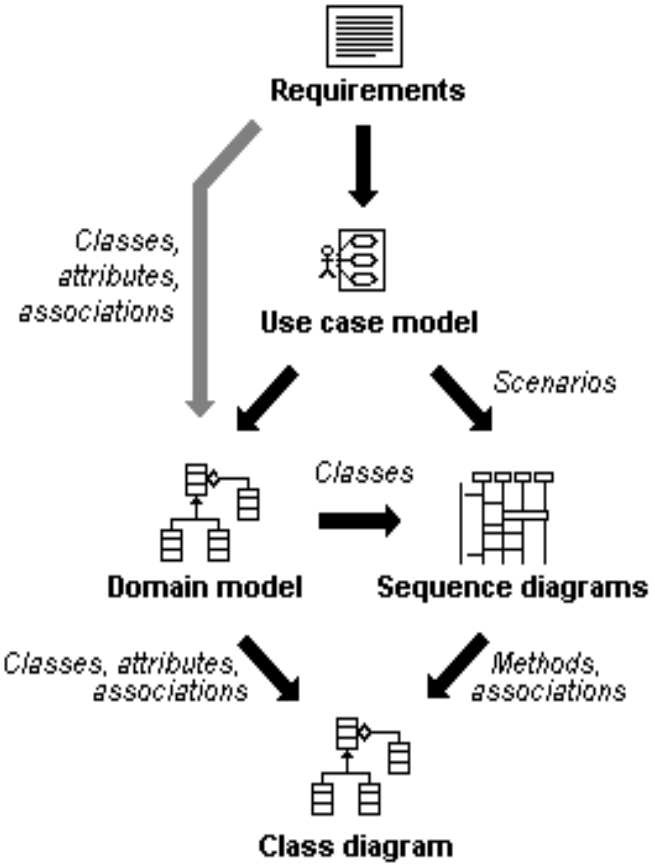
- Likheter. Begge viser
  - Hvem som skal bruke systemet
  - Hva de skal gjøre med det
  - Hvorfor de skal gjøre det
- Forskjeller
  - Omfang, kompletthet, livslengde, hensikt
  - User stories er godt egnet for å finne krav og bruke disse i smidig utvikling i samarbeid med produkteier/kunde
  - Use case er mer detaljert, har flere bruksområder videre i prosjektet og er mer egnet som dokumentasjon
- Men, det er en flytende overgang mellom dem.

Use case i prosjektplanlegging

- Planlegg hvilke use case som skal implementeres i hvilke iterasjoner av prosjektet:
  - Implementer use casene i henhold til hvor viktige de er og/eller hvor vanskelige de antas å være å implementere.
  - Hovedflyt implementeres først, deretter alternativene.
  - Estimer hvor mange use case (eller hendelsesflyt og alternative flyt) som kan implementeres i en iterasjon.

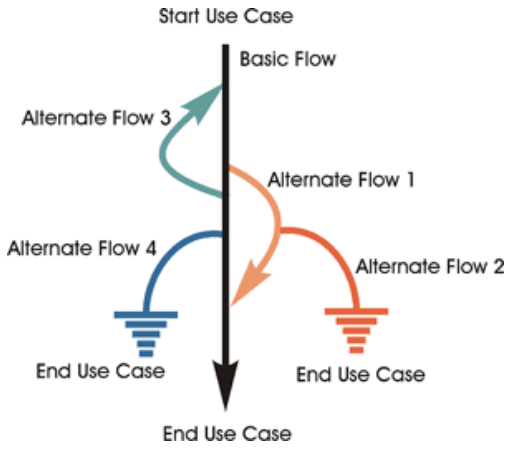
Use case i design

- Hendelsesflyt i use casene detaljeres ut i sekvensdiagram
- Domenemodellen utvides til klassediagram med systemklasser



Use case i funksjonelle tester

- Use case kan være utgangspunkt for testprosedyrer (manuelle eller automatiserte)
- Dette gir
  - Effektiv utforming av tester
  - Fokus på test av egenskaper som er viktig for bruker
  - Mål på testdekning i forhold til brukstilfeller

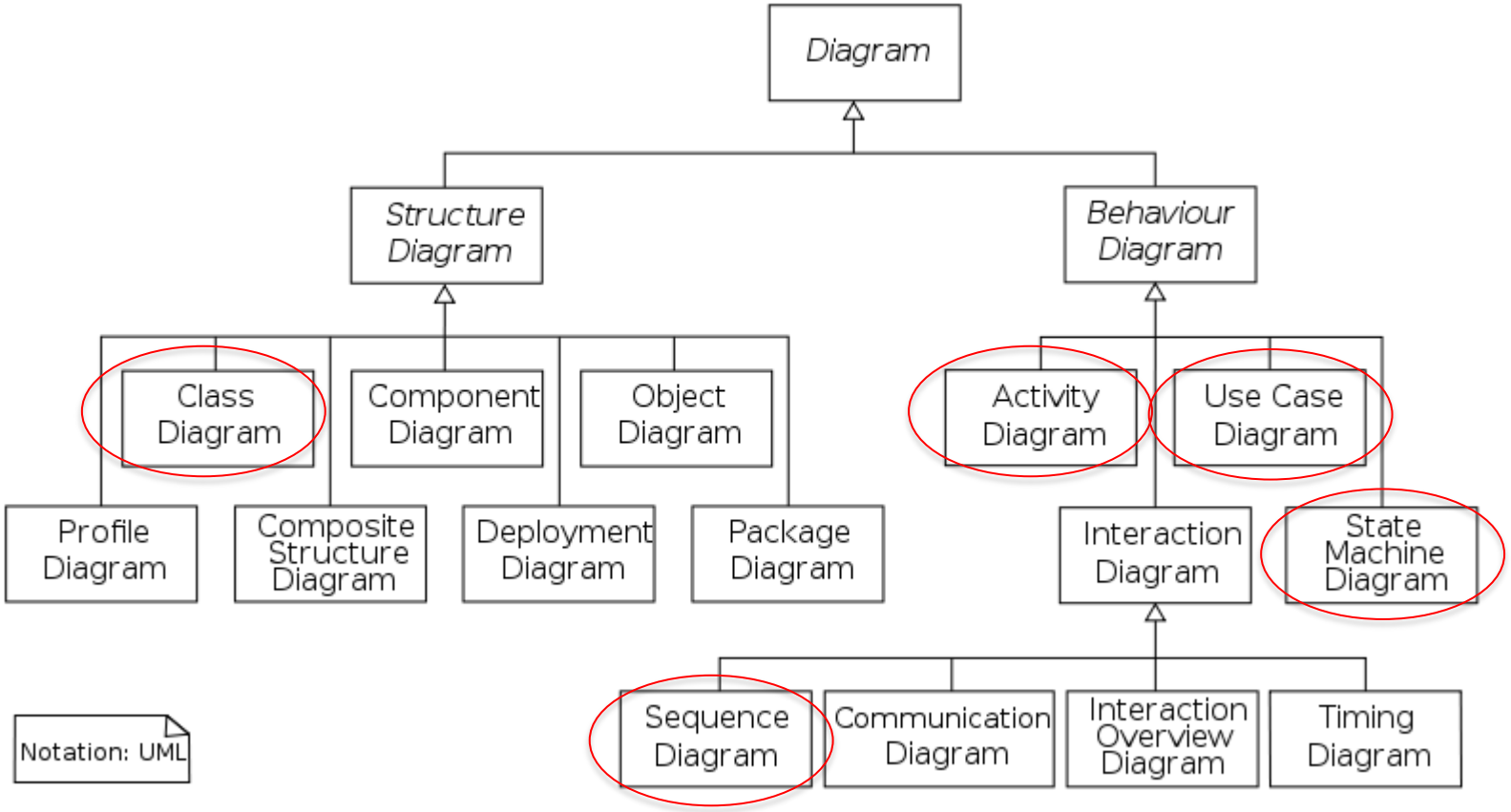


En testprosedyre for hver vei gjennom use caset

Testprosedyre:

1. Prebetingelse
2. Input data
3. Steg
4. …
5. ..
6. Forventet resultat

UML - diagrammer



Kilde: http://en.wikipedia.org/wiki/Unified_Modeling_Language#Structure_diagrams