UML-
Use case drevet analyse og design

Bente Anda

23.09.2004

I dag

- Domenemodeller
- Sekvensdiagrammer
- Use case realisering med GRASP patterns
- Klassediagram - designmodeller

Domenemodell – visualisering av konsepter

- Domenemodellen viser konsepter i applikasjonsdomenet og forholdet mellom dem: ideer, ting, objekter.
- Domenemodellen beskrives med UML klassediagrammer uten metoder, og utarbeides gjennom flere iterasjoner.
- Det er bedre å spesifisere for mange konseptuelle klasser enn for få.

Domenemodell forts.

- Use case modell og domenemodell utformes parallelt.
- Domenemodellen fanger opp informasjonen om entiteter som er beskrevet i use casene.
- Use casene presiseres ved utforming av domenemodellen.

Hvordan finne domeneklasser?

2 forskjellige tilnærminger:

- Lag en liste over kandidater til domeneklasser (for eksempel basert på liste s.134)
- Finn substantiver og substantivutrykk i use case beskrivelsene

Finn substantiver i use casene

Eksempel: Use case 'Generer spørreskjema'

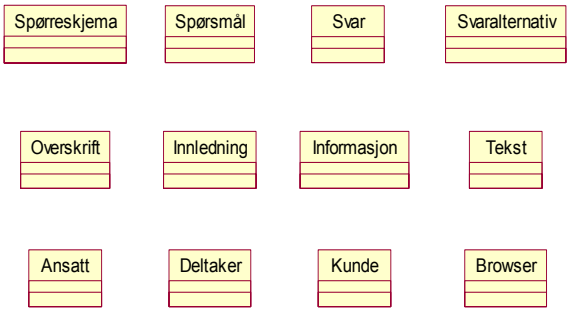
Aktør: Ansatt

1. Systemet ber om **overskrift, innledning** og **antall spørsmål**
2. Ansatt skriver inn nødvendig **informasjon**
3. Systemet sjekker at alle **felt** er utfylt
4. Systemet viser et **spørreskjema** der **tekst** til **spørsmål** skal fylles inn
5. Ansatt skriver inn **tekst** og **svaralternativ**
6. Systemet sjekker at riktig **antall spørsmål** har fått **tekst**
7. Ansatt ber om at **spørreskjemaet** blir lagret
8. Systemet lagrer **spørreskjemaet**

Domenemodellering

- Lag en liste over kandidater til klasser:
  Spørreskjema, Spørsmål, Svar, Svaralternativ, Overskrift, Innledning, Tekst, Informasjon, Ansatt, Deltaker, Browser, Kunde
- Ta bort unødvendige klasser
- Tegn klassene inn i domenemodellen
- Legg på assosiasjoner for å vise relasjoner
- Legg til attributter som er nødvendige for å oppfylle kravene

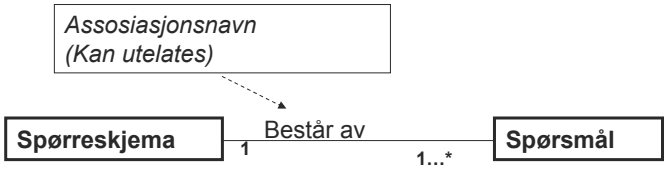
Overordnet domenemodell



Assosiasjoner

- En assosiasjon er en relasjon mellom to klasser som viser at det er en sammenheng mellom dem.
- Brukes i domenemodellen hvis informasjon om relasjonen skal lagres.

Eksempel



Assosiasjonen går i begge retninger.

Retningslinjer for assosiasjoner

Klasse A og Klasse B er assosiert hvis

- Et objekt av klasse A sender en beskjed til et objekt av klasse B
- Et objekt av klasse A oppretter et objekt av klasse B
- Et objekt av klasse A har attributter hvis verdier er objekter av klasse B eller mengder av objekter av klasse B
- Et objekt av klasse A mottar en beskjed med et objekt av klasse B som parameter

Se assosiasjonsliste i boka s.156

Forfin domenemodellen

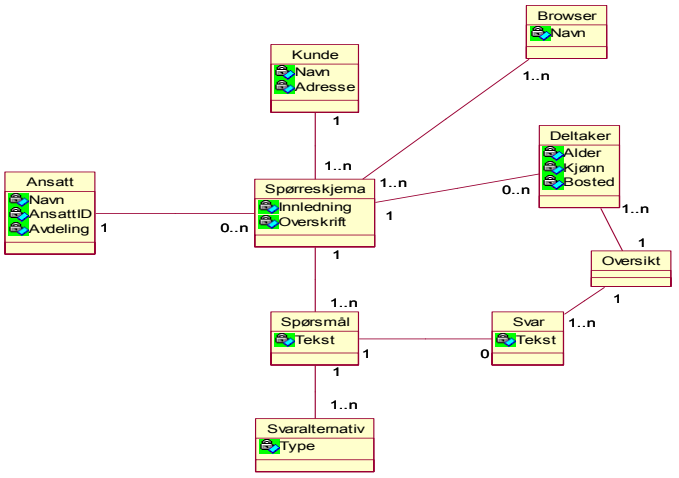
- Ta bort unødvendige klasser (Tekst, Informasjon)
- Noen klasser viser seg å være attributter (Overskrift, Innledning)
- Legg på multiplisitet

Attributter

- Inkluder de attributter som det må lagres informasjon om i følge use casene
- Attributter er vanligvis datatyper, f.eks int, Boolean, string, dato
- Modeller konsepter som klasser, ikke attributter. Hvis det er tvil, lag en klasse

  Eksempel: Lag en klasse 'Oversikt' som viser enkel statistikk som antall svar, deltakerens alder, geografiske tilknytning etc.

Domenemodell med assosiasjoner



Use case realisering

- Sekvensdiagrammer visualiserer operasjonene som genereres av aktørene
- For hvert use case lages et sekvensdiagram for normal hendelsesflyt og komplekse og hyppig forekommende variasjoner
- Stegene i use casene vises som meldinger som sendes mellom objektene

Sekvensdiagrammer

- Viser hvordan objekter kommuniserer ved hjelp av meldinger (metodekall)
- Viser hendelsesflyten i use casene
- Er et redskap for å lage et 'godt' objektdesign
- Lages parallelt med klassediagrammer

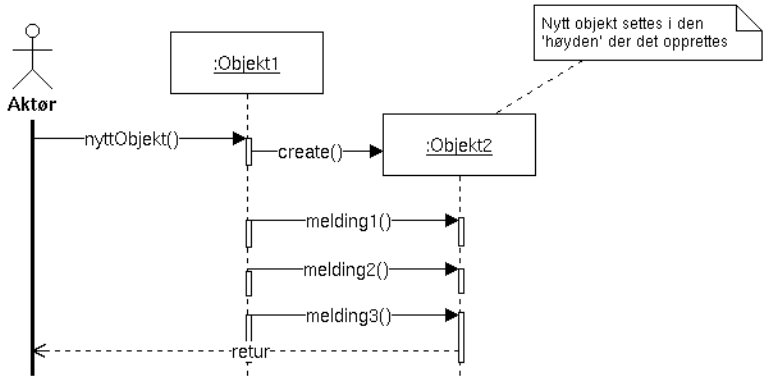
Objektdesign: Ansvarstilordning

- UML definerer ansvar som en 'kontrakt' i en klasse
- Ansvar er knyttet til objektet i form av dets oppførsel
  - *Handling:* Opprette objekt, beregning
  - *Kunnskap:* Vite om private data, vite om relaterte objekter
- Ansvar er ikke det samme som metoder, men metoder implementeres for å oppfylle ansvaret
- Ansvarstilordning: En utfordring under utforming av sekvensdiagrammer

Utforming av sekvensdiagrammer

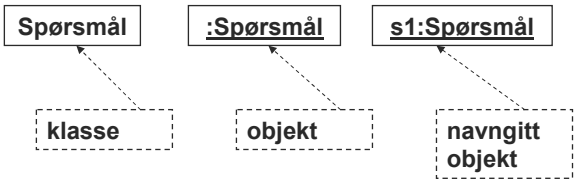
- Grensesnittklassen eller aktøren viser grensesnittet mellom system og aktør
- For **kodegenerering** må objektene skrives på formen **:klassenavn**
- Meldinger (metodekall) vises med piler. For **kodegenerering** må metodene ligge i klassen som pilen peker på
- Nytt objekt opprettes med create()
  NB! create() peker rett på det nye objektet

Grunnleggende notasjon for sekvensdiagram

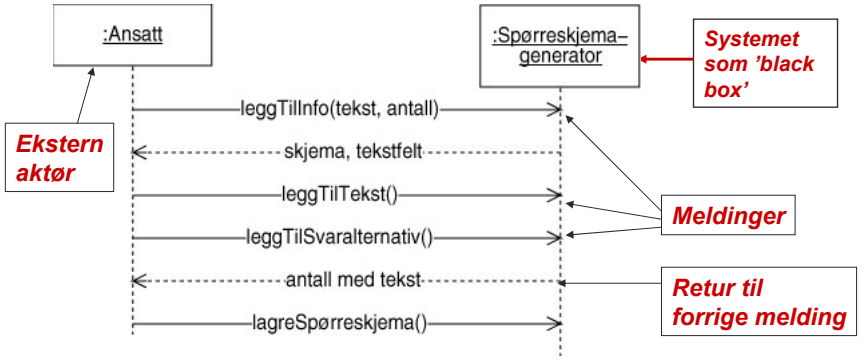


Klasser og objekter

I UML er klasser og objekter illustrert på samme grafiske måte, men navnet er understreket i objektene.



System Sekvensdiagram for 'Generer spørreskjema'

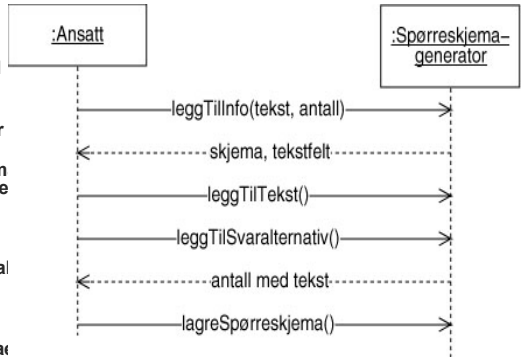


Use case vs. system sekvensdiagram

Use case 'Generer spørreskjema'

Aktør Ansatt

1. **Systemet ber om overskrift, innledning og antall spørsmål**
2. **Ansatt skriver inn nødvendig informasjon**
3. **Systemet sjekker at alle felt er utfylt**
4. **Systemet viser et spørreskjem der tekst til spørsmål skal fylle inn**
5. **Ansatt skriver inn tekst og svaralternativ**
6. **Systemet sjekker at riktig antal spørsmål har fått tekst**
7. **Ansatt ber om at spørre-skjemaet blir lagret**
8. **Systemet lagrer spørreskjema**



Kjennetegn på ’god’ design

- En god utforming gjør den jobben den er ment å gjøre
- En god utforming er enkel og elegant
  - Eleganse innebærer å finne akkurat riktig abstraksjonsnivå
- En god utforming er gjenbrukbar, utvidbar og enkel å forstå
- Et godt objekt har et lite og veldefinert ansvarsområde
- Et godt objekt skjuler implementasjons-detaljer fra andre objekter - *Grady Booch*

Modularisering

- Høy kohesjon
  - Et objekt skal bare ha ansvar for relaterte ting
- Lav kobling
  - Et objekt skal ha samarbeid med et begrenset antall andre objekter

Høy kohesjon

- Kohesjon er et mål på hva slags ansvar et objekt har og hvor fokusert ansvaret er
- Et objekt som har moderat ansvar og utfører et begrenset antall oppgaver innenfor ett funksjonelt område har høy kohesjon
- Objekter med lav kohesjon har ansvar for mange ting innen ulike funksjonelle områder

Lav kobling

- Kobling er et mål på hvor sterkt et objekt er knyttet til andre objekter
- Et objekt med sterk kobling er avhengig av mange andre objekter noe som kan gjøre endring vanskelig

Retningslinjer for god objektdesign

- 'Design patterns' er navngitte retningslinjer for hvordan ansvar skal fordeles i ulike situasjoner
- Brukes i prosessen med å forfine sekvensdiagrammer
- GRASP – 'Patterns of General Principles in Assigning Responsibilites' = Mønster for problemløsning

Noen GRASP patterns

- Ekspertprinsippet: La det objektet som har kunnskapen (dataene) også behandle den *(Eksempel 'Spørreskjema')*
- Kontrollobjektprinsippet: Velg objekt som håndterer systemhendelser
  (Eksempel: 'SpørreskjemaHåndterer' – use case kontrollobjekt)
- Skaperprinsippet: Legg ansvar for å opprette et nytt objekt i klassen som må vite om objektet *(Eksempel: 'SpørreskjemaGenerator')*

Ekspertprinsippet (Information Expert)

- Problem: Hva er det generelle prinsipp for å tilordne ansvar til objekter?
- Løsning: La det objektet som har kunnskapen (dataene) også behandle den
- Hvordan:
  - Begynn med å formulere ansvarsområdet:
    Hvilket objekt har ansvar for å vite om det totale antall svar på undersøkelsen?
  - Se etter mulige systemklasser i domenemodellen

Eksempel

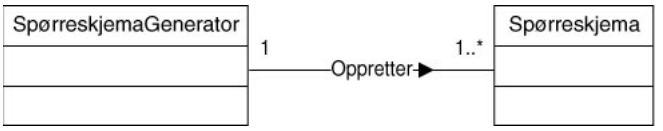
- Se i domenemodellen etter en klasse som kan brukes eller utvides til å lage en designklasse, for eksempel klassen 'Oversikt'
- Lag en systemklasse 'Oversikt' og gi den ansvaret for kunnskap om antall svar med metoden 'visAntallSvar()'

Skaperprinsippet (Creator)

- Problem: Hvem er ansvarlig for å opprette nye objekter?
- Løsning: La det objektet som må vite om de nye objektene lage dem
- Hvordan: Gi klasse B ansvaret for å opprette et objekt av klasse A dersom ett av følgende er sant:
  - B inneholder A-objekter
  - B bruker A-objekter
  - B har data som sendes til A-objektet når det opprettes

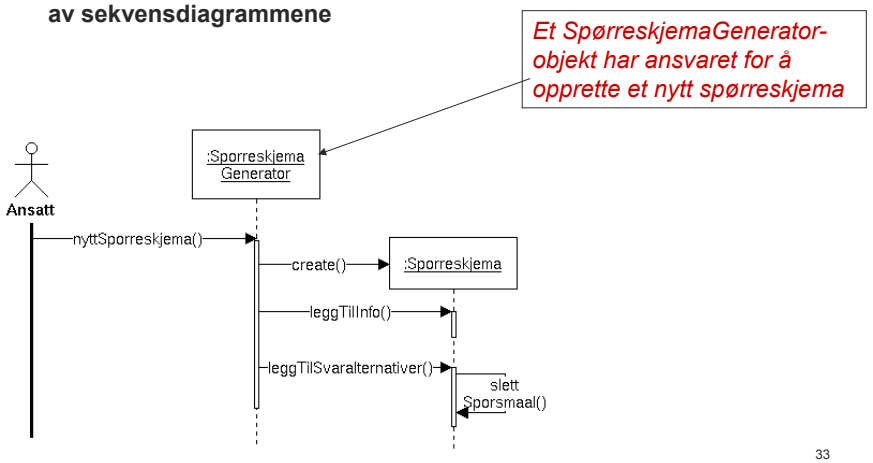
Finn systemklasse

- Se i domenemodellen etter mulige klasser
- Vi trenger en 'SpørreskjemaGenerator' klasse for å opprette nye skjemaer



Ansvar og sekvensdiagrammer

Ansvaret utdypes under utforming av sekvensdiagrammene



Kontrollobjektprinsippet (Controller)

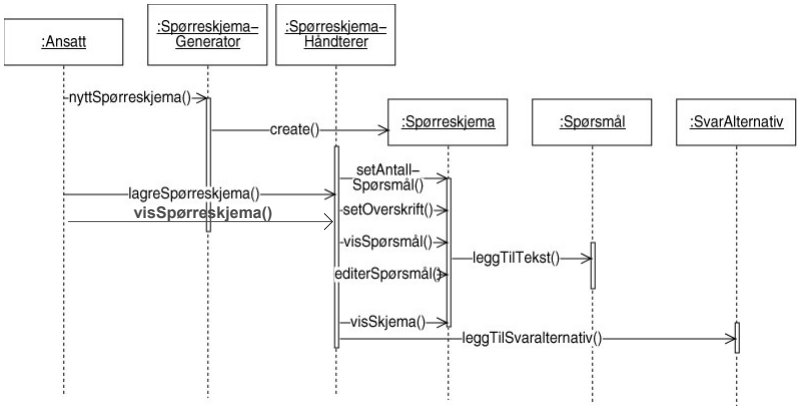
- Hvem er ansvarlig for å håndtere systemhendelser (en hendelse som genereres av en ekstern aktør)?
- Løsning: Tilordne ansvar for å håndtere en systemhendelse til en av følgende klasser:
  - En klasse som representerer systemet eller subsystemet (fasadekontroller)
  - En klasse som representerer et use case scenario der systemhendelsen forkommer ofte Navn: <UseCaseNavn> Håndterer

    Eksempel: 'SpørreskjemaHåndterer'

Kontrollobjekter

- Kontrollobjektet er et grensesnitt-objekt
- Kontrollobjekter delegerer oppgaver til andre objekter
- Hvert use case kan ha et kontrollobjekt

Generer spørreskjema



Designmodellen

- Lag design-klassediagram parallelt med sekvensdiagrammer
- Lag noen sekvensdiagrammer, oppdater klassediagrammet, utvid sekvensdiagrammet etc.
- Designklassene er systemklasser, ikke konseptuelle klasser som i domenemodellen

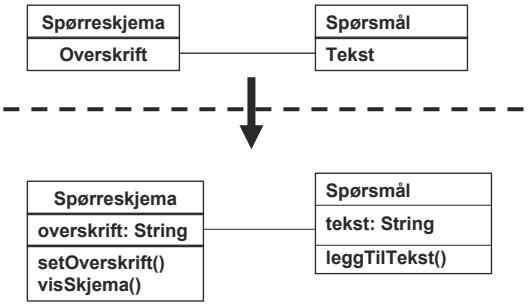
Framgangsmåte for designmodellering

- Identifiser klasser ved å gå gjennom alle sekvensdiagrammene
- Klassenavnene er inspirert av navn i domenemodellen
- Legg til metodenavn ved å analysere sekvensdiagrammene
  Eks: Meldingen leggTilTekst() sendes til Spørsmåls-objektet. Objektet må derfor inneholde en leggTilTekst() metode

Fra domenemodell til designmodell

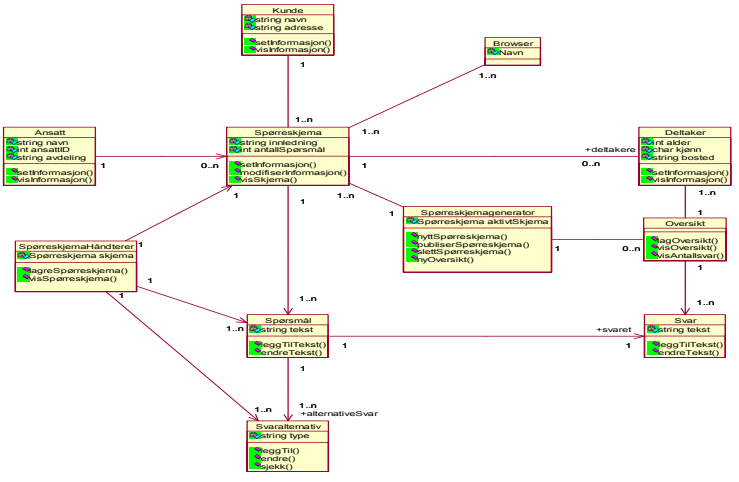
Noen av objektene som kommuniserer via meldinger hentes fra domenemodellen

Domenemodell –konseptuelle klasser danner grunnlag for objekter og objektnavn i designmodellen



Designmodell – systemklasser

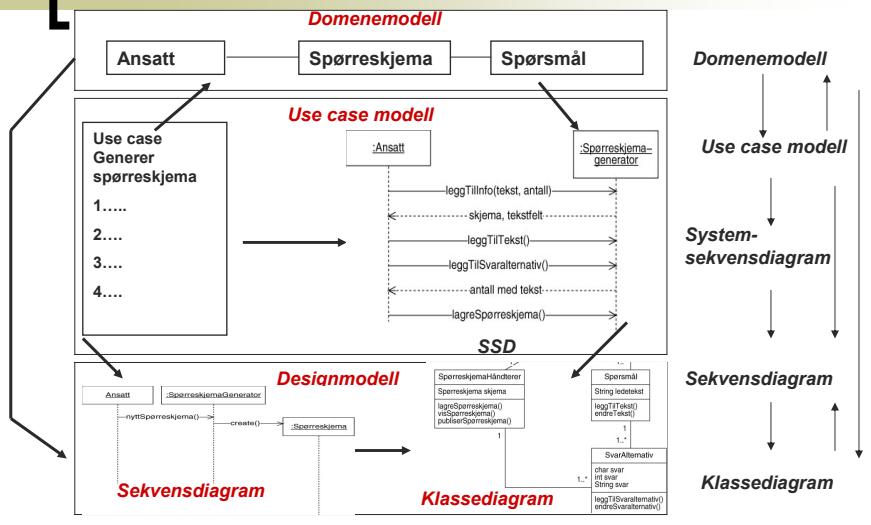
Designmodell



Ansvarsfordeling

- **Spørreskjemagenerator:** Lager nye spørreskjemaer, publiserer dem og viser oversikter.
- **Spørreskjema:** Kjenner sin egen overskrift, innledning og sine spørsmål
- **Spørsmål:** Kjenner sine egne tekster og kan modifisere disse
- **Svar:**Kjenner sitt eget svar og kan modifisere dette
- **SpørreskjemaHåndterer:** Kontrollobjekt som koordinerer objektene
- **Svaralternativer:** Kjenner lovlige svar på et gitt spørsmål.
- **Oversikt:** Vet om innholdet i besvarelsene og lager statistikk

Utviklingsprosess - oversikt



Oppsummering

- I objektorientert analyse utarbeides use case modell og domenemodell i parallell.
- Scenariene i et use case realiseres gjennom objekter som samarbeider. Dette illustreres i sekvensdiagrammer.
- Klassediagrammer og sekvensdiagrammer utarbeides i parallell, designklasser og metoder finnes ofte fra sekvens-diagrammene
- Bruk av patterns kan lette og forbedre designarbeidet