[simula . research laboratory]

Use case drevet design med UML

Bente Anda

14.09.2006

[simula . research laboratory]

I dag

- Oppgaven fra forrige forelesning
- System sekvensdiagrammer
- Operasjonskontrakter
- GRASP patterns
- Designmodeller med sekvens- og klassediagram

[simula . research laboratory]

Oppgave 2 fra 11/9

- To forskjellige systemer skulle modelleres v.h.a. use case: Bordbestillinger på restaurant og Overvåking av produksjonsprosess.
- Undersøkte
  - typiske problemer
    - Aktører: Restaurantsystemet - system, call og table. Prosessystemet – system, conveyor belt, workstation, sensor og actuator. Manglet mål pr. primær aktør.
    - Use case: Restaurantsystemet – manglet display, Prosessystemet – manglet Create or update workflow plan/operation.
    - Lite riktig bruk av include og extends, men kanskje mye en konsekvens av litt manglende use case.
  - effekt av forkunnskap
    - Spørsmål om 1) INF1050, 2) andre kurs, 3) hvorvidt pensum var lest på forhånd samt 4) egen oppfatning av mestring av use case modellering var mål på forkunnskap.
    - 1-3 hadde liten betydning for resultatet, men 4 hadde stor betydning (max. score = 6):

Mestret UCM	N	Median	Ave Rank
Dårlig	13	2,000	10,5
Akseptabelt	14	3,500	19,4
Godt	4	4,000	22,3
Overall	31		16,0

P = 0,011 (adjusted for ties)

  - effekt av domenekunnskap
    - Antok at restaurantsystemet ville virke mer kjent. I tillegg var spørsmålene om hvorvidt 1) spesifikasjonen var forståelig og 2) applikasjonsdomenet var kjent mål på domenekunnskap.
    - 1 og 2 hadde ingen betydning, men det var en liten forskjell mellom systemene:

Oppg	N	Median	Ave Rank
Restaurant	17	3,000	17,3
Prosess	14	2,000	14,5
Overall	31		16,0

P = 0,382 (adjusted for ties)

[simula . research laboratory]

System sekvensdiagrammer

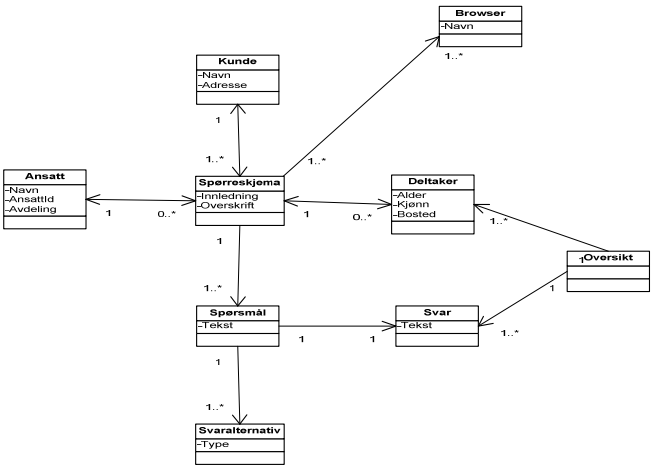
- System sekvensdiagrammer visualiserer operasjonene som genereres av aktørene.
- System sekvensdiagrammene identifiserer nødvendig input til use casene (brukergrensesnitt, kommunikasjon med andre systemer)
- Domenemodellen er input til konstruksjon av system sekvensdiagrammene (utgangspunkt for klasser) og use casene.

Eksempel: Spørreskjemagenerator

Kravspesifikasjon:

Et meningsmålingsinstitutt ønsker å få laget et system der spørreskjema er på Internett/Web. Systemet skal gjøre det enkelt å legge et spørreskjema ut på Web, enkelt for andre å fylle ut skjemaene på Web. Svarene skal lagres på et format som kan eksporteres til andre verktøy (f eks "strukturert tekst" som kan importeres til et regneark). Deltakerne skal kunne lagre svarene underveis og fortsette utfyllingen av skjemaet senere. Til noen av spørsmålene er det nødvendig å lese en del tekst. Meningsmålingsinstituttet ønsker å ha en enkel oversikt over svarene som er kommet inn, f eks hvor mange som har svart på de ulike spørsmålene. Det som skal lages er altså ikke et enkelt spørreskjema på Web, men en "spørreskjema-generator" for Web.

Domenemodell med assosiasjoner

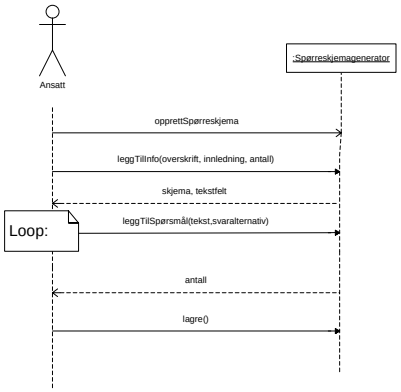


[simula . research laboratory]

Use case og system sekvensdiagram

Use case 'Generer spørreskjema'

1. Ansatt ber om å få opprette nytt spørreskjema
2. Systemet ber om overskrift, innledning og antall spørsmål som spørreskjemaet skal bestå av.
3. Ansatt skriver inn nødvendig informasjon
4. Systemet sjekker at alle felt er utfylt
5. Systemet viser et spørreskjema der tekst til spørsmål skal fylles inn.
6. Ansatt skriver inn tekst og evnt. svaralternativ til hvert av spørsmålene
7. Systemet sjekker at riktig antall spørsmål har fått tekst
8. Ansatt ber om at spørreskjema blir lagret
9. Systemet lagrer spørreskjemaet



[simula . research laboratory]

Operasjonskontrakter

- Kontrakter med pre- og post betingelser på operasjoner i use casene for å gjøre use casene mer presise.
- Prebetingelser beskriver forutsetningene for å gjennomføre operasjonen.
- Postbetingelser beskriver endringer i tilstandene til objektene i domene modellen.

[simula . research laboratory]

Eksempel på kontrakt

- Operasjon: leggTilInfo(overskrift, innledning, antall)
- Kryssreferanse: Use case ’Generer spørreskjema’
- Prebetingelse: Et nytt spørreskjema er opprettet
- Postbetingelser:

  - spørreskjema.overskrift er blitt overskrift

  - spørreskjema.innledning er blitt innledning

  - antall spørsmål er opprettet og assosiert med spørreskjema

[simula . research laboratory]

Sekvensdiagrammer

- Interaksjonsdiagrammer er det fremste verktøyet for tilordning av ansvar til objektene (vi fokuserer her på sekvensdiagrammer)
- 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
- Sekvensdiagrammene viser hvordan objekter kommuniserer ved hjelp av meldinger (metodekall)
- Sekvensdiagrammene lages parallelt med klassediagrammer

Et UML sekvensdiagram

- viser hendelsesflyten i et use case
- viser **interaksjoner** (samarbeid) mellom objekter i systemet
- viser rekkefølgen på meldingene (abstrakt) som sendes mellom objektene
- kan brukes til å identifisere **metodene** (konkret) til objektene i systemet

[simula . research laboratory]

Objekt design

- Hvilke er de riktige objektene?
- Hva er ansvarsområdene til objektene, hva skal de vite og hva skal de gjøre?
- Hvem samarbeider objektene med?
- Objektene i er av typene kant-, kontrol- og entitetsobjekter
  - Entitetsobjekter holder data
    - Modellerer varig informasjon og assosiasjoner
    - Ofte abstraksjoner av entiteter i den virkelige verden (domene-klasser)
    - Kan finnes ved å lete etter substantiver i kravspesifikasjoner
  - Kantobjekter kommuniserer med omverdenen
    - Modellerer systemets grensesnitt
  - Kontrollobjekter kontrollerer interaksjonene i hele systemet eller i et use case
    - Modellerer oppførsel til et eller flere use case
    - Håndterer kontrollflyten
- Merk også at en klasse betyr forskjellige ting avhengig av perspektiv

[simula . research laboratory]

Allokere ansvar til klasser

- Et *ansvar* er noe systemet må gjøre (basert på kravene til systemet)
  - Hvert funksjonelle krav må tilordnes en eller flere klasser
    - Alle ansvarsområdene til en gitt klasse skal være *klart relatert*
    - Hvis en klasse har for mange ansvarsområder, vurder *å splitte* den i distinkte klasser.
    - Hvis en klasse ikke har noe ansvar så er den antageligvis *overflødig*
    - Når et ansvar ikke kan tilordnes til en eksisterende klasse, opprett en ny *klasse*.
  - For å finne ansvarsområder
    - Analyser use casene
    - Se etter verb og substantier som beskriver *handlinger*

[simula . research laboratory]

Objektdesign: Ansvarstilordning

- Ansvar er knyttet til objektet i form av dets oppførsel
  - *Handling:* Opprette objekt, beregning, initiere handlinger i andre objekter, kontrollere og koordinere handlinger i andre objekter.
  - *Kunnskap:* Vite om private data, vite om relaterte objekter, vite om ting som det kan utlede eller beregne
- Ansvar er ikke det samme som metoder, men metoder implementeres for å oppfylle ansvaret
- Kategorier av ansvar:
  - Sette (set) og hente (get) verdier av attributter
  - Opprette og initialisere nye instanser
  - Hente fra og lagre til persistent minne
  - Slette instanser
  - Legge til og slette linker for assosiasjoner
  - Kopiere, konvertere, endre og outputte
  - Beregne numeriske resultater
  - Navigere og søke
  - …

[simula . research laboratory]

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 implementasjonsdetaljer 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

[simula . research laboratory]

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

[simula . research laboratory]

Patterns

- Patterns er navngitte retningslinjer for hvordan ansvar skal fordeles i ulike situasjoner.
- Patterns representerer sammenstilling av erfaring
- Patterns brukes i prosessen med å forfine sekvensdiagrammer
- GRASP – 'Patterns of General Principles in Assigning Responsibilites' =
  Mønster for problemløsning
- Sentrale patterns er Ekspertprinsippet, Kontrollobjektprinsippet og Skaperprinsippet

[simula . research laboratory]

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 (spørreskjemageneratoren) ?

  Hvilket objekt har ansvar for å ha oversikt over alle bookingene (restaurantsystemet)?
- Se i domenemodellen etter en klasse som kan brukes eller utvides til å lage en designklasse.

  Det finnes en klasse 'Oversikt' i domenemodellen, lag en designklasse 'Oversikt' og gi den ansvaret for kunnskap om antall svar med metoden 'visAntallSvar()'.

  Det finnes ingen klasse som har oversikt over alle bookingene, lag en designklasse 'Restaurant' og gi den ansvaret for dette.

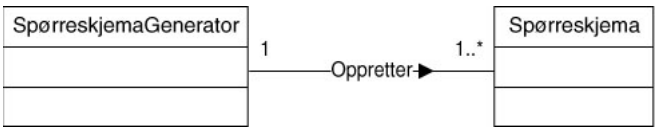
[simula . research laboratory]

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 registrerer A-objekter
  - B bruker A-objekter
  - B har data som sendes til A-objektet når det opprettes

Forts. skaperprinsippet

- Se i domenemodellen etter mulige klasser
- *Designklassen 'Restaurant' oppretter bookinger*
- *Vi trenger en 'SpørreskjemaGenerator' klasse for å opprette nye skjemaer*



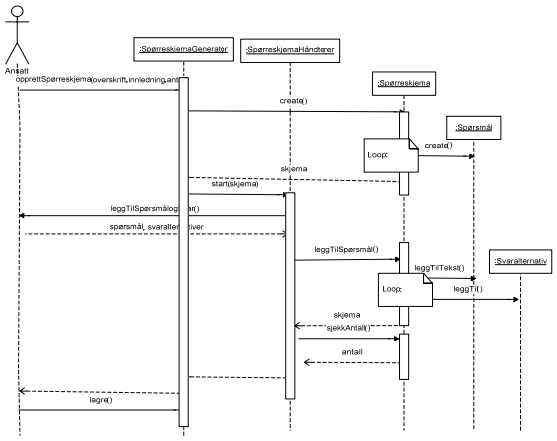
Kontrollobjektprinsippet (Controller)

- Problem 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:

    'SpørreskjemaHåndterer' håndterer use caset 'Generer spørreskjema'

    'Bookingsystem' håndterer alle use casene i restaurantsystemet
- Kontrollobjekter delegerer oppgaver til andre objekter

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

[simula . research laboratory]

Analyse vs. designmodel

- *Analysemodellen* utelater mange klasser som er nødvendige i et komplett system
  - Er typisk en domenemodell
  - Kan inneholde mindre enn halvparten av klassene i systemet.
  - Uavhengig av spesielle
    - brukergrensesnittsklasser
    - arkitekturklasser (f.eks. design patterns klasser)
- Den komplette *designmodellen* inneholder
  - Domenemodellen
  - Brukergrensesnittsklasser
  - Arkitekturklasser (f.eks. slik at klasser kan kommunisere)
  - Utility klasser (f.eks. håndtering av mengder og strenger)

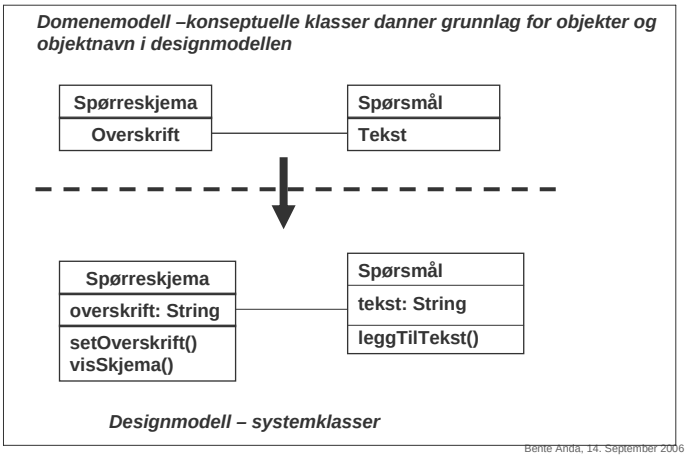
[simula . research laboratory]

Framgangsmåte for designmodellering

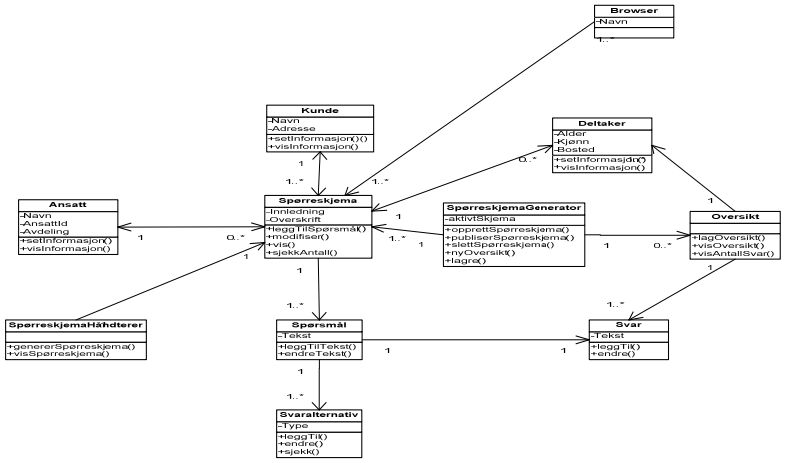
- Identifiser klasser ved å gå gjennom alle sekvensdiagrammene
- Klassenavnene er inspirert av navn i domenemodellen
- Legg til metodenavn ved å analysere sekvensdiagrammene
  Eks: Meldingen leggTilTekst() sendes til Spørsmåls-objektet. Objektet må derfor inneholde en leggTilTekst() metode

Fra domenemodell til designmodell

Noen av objektene som kommuniserer via meldinger hentes fra domenemodellen



Designmodell for spørreskjemagenerator

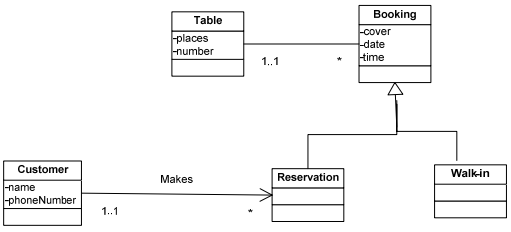


Ansvarsfordeling

- **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

Oppgave

- Utvid domenemodellen for restaurantsystemet til en designmodell



The system shall support the processes of making reservations and allocating tables to customers in a restaurant.

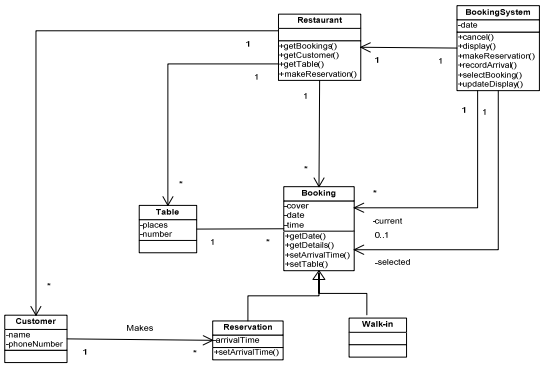
Customers contact the restaurant to make or cancel advance bookings, and a receptionist receives these calls. Bookings are entered for a particular table together with the number of covers. The restaurant runs three sittings in an evening: ”Pre-theatre”, ”Dinner” and ”Supper”, but bookings can be made for time periods that span more than one sitting. A contact name and phone number is recorded for each booking.

When a party arrives, it is seated at its table by the head waiter. The corresponding booking is crossed out. If the party is seated at a table other than the one booked, an arrow is drawn from the original booking to the new table. Customers may, cancel bookings in advance. The time by which a table must be vacated can also be recorded.

It is, of course, possible to eat without making an advance booking if a free table is available. This is recorded as a table occupancy, but no record of name or telephone number is made.

When new bookings are recorded, or changes are made to existing bookings, the display should be immediately updated, so that the staff is always working with the latest information available.

Designmodell for restaurantsystemet



Oppsummering

- I objektorientert analyse utarbeides use case modell og domenemodell i parallell.
- System sekvensdiagrammer visualiserer interaksjonen mellom aktør og system og nødvendig input til systemet.
- Operasjonskontrakter kan gjøre use casene mer presise.
- 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