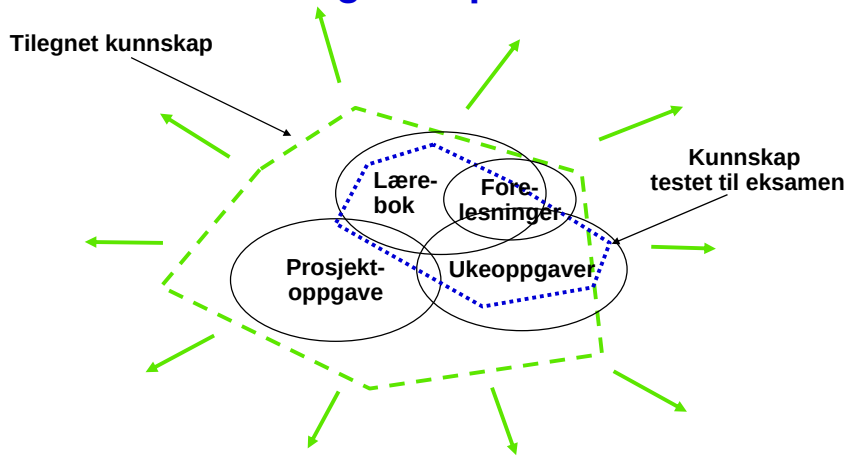
Oppsummering – INF1050

- Dagsorden
  - Produksjon av informasjonssystemer
  - Systemutviklingsprosessen
  - Eventuelt

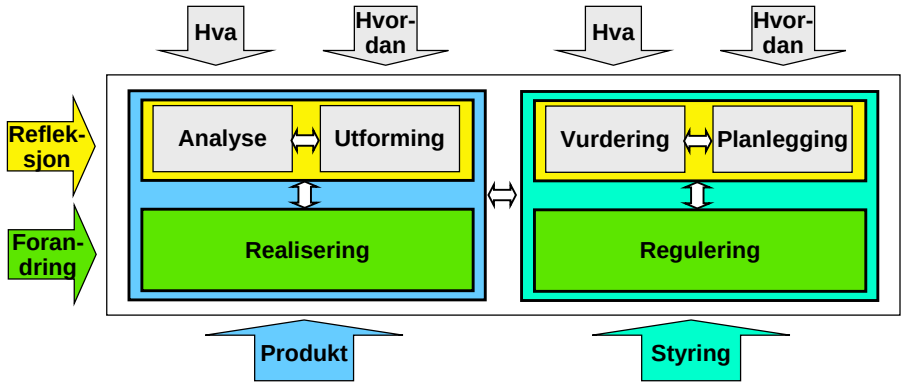
Læringskomponenter



Hovedtemaene i INF1050

- Produksjon av informasjonssystemer
- Styring av produksjon av informasjonssystemer
- Rammer for utvikling av informasjonssystemer
- Verktøy og plattformer

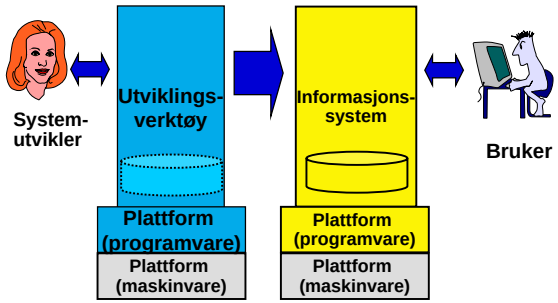
Figur 2-2. Et overordnet rammeverk for systemutvikling



Etter N. E. Andersen et.al. (1986):
Professionel Systemudvikling

Utviklingsverktøy og plattformer

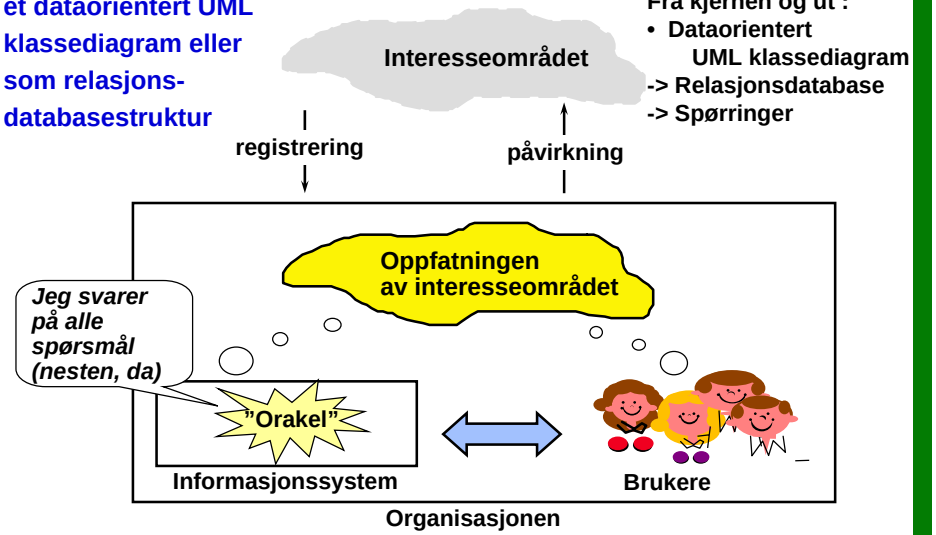
- Plattformer
  - Operativsystemer
  - Webtjenere
  - Databasehåndterings-systemer
  - Mellomvarer (Middleware)
  -
- Utviklingsverktøy
  - CASE-verktøy
  - Editorer, kompilatorer
  - Utviklingsmiljøer
  - Programmeringsspråk
  -



Fra kjernen og ut

Hva oraklet skal kunne vite uttrykkes et dataorientert UML klassediagram eller som relasjons-databasestruktur

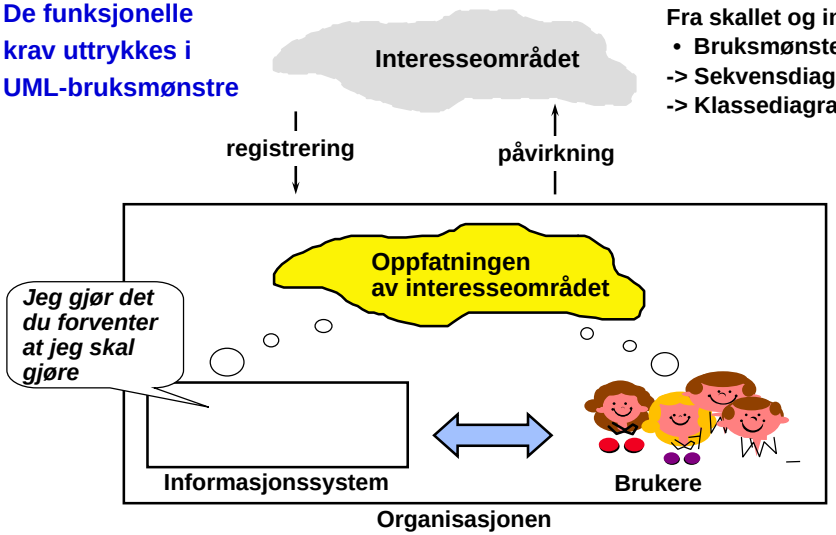
Fra kjernen og ut :
- Dataorientert UML klassediagram
-> Relasjonsdatabase
-> Spørringer



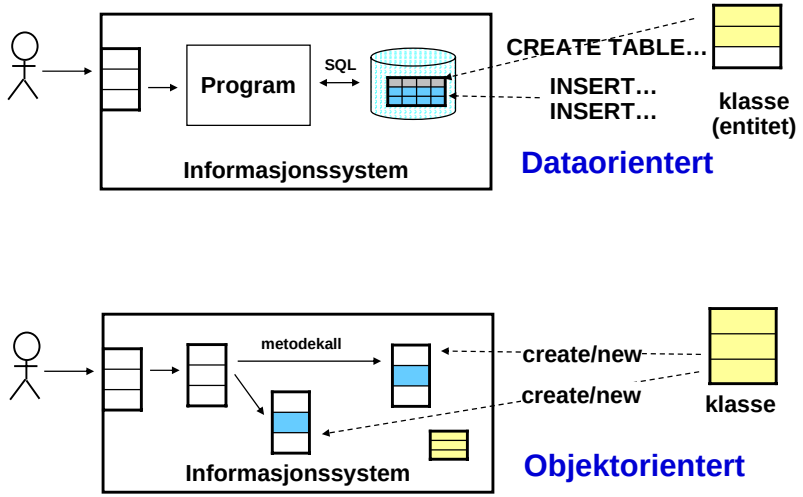
Fra skallet og inn

De funksjonelle krav uttrykkes i UML-bruksmønstre

Fra skallet og inn:
- Bruksmønster
-> Sekvensdiagram
-> Klassediagram



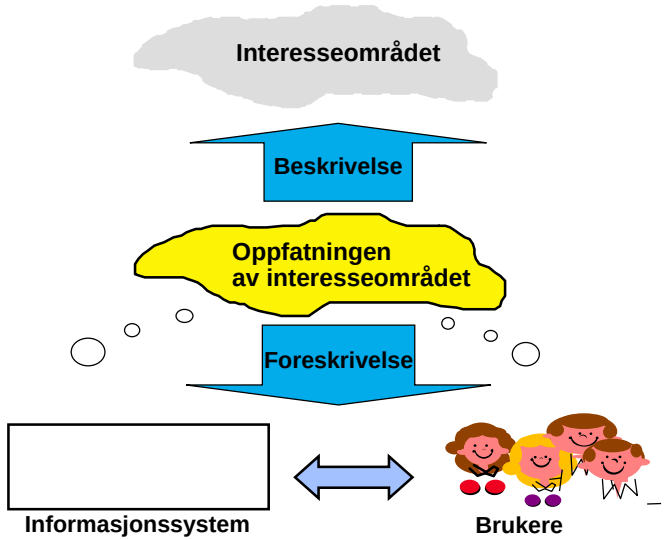
Dataorientert vs. objektorientert utforming



Utviklingsretninger og utforminger

	Fra kjernen og ut	Fra skallet og inn
Dataorientert arkitektur	☺	?
Objektorientert arkitektur	?	☺

Modellenes to formål



Fra kjernen og ut

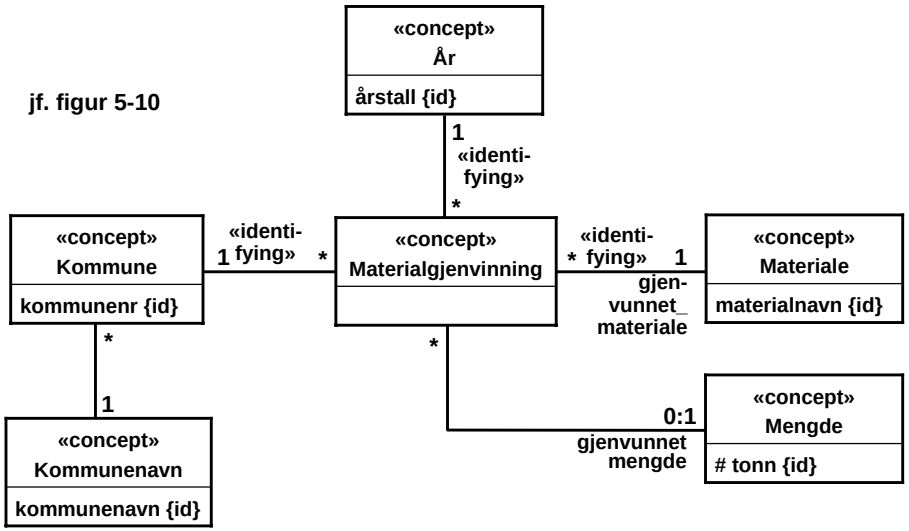
- UML i dataorientert systemutvikling
- Den dataorienterte Inf1050-metoden
- Hva jeg håper dere har lært (☺)

Hva brukes UML til i dataorientert SU?

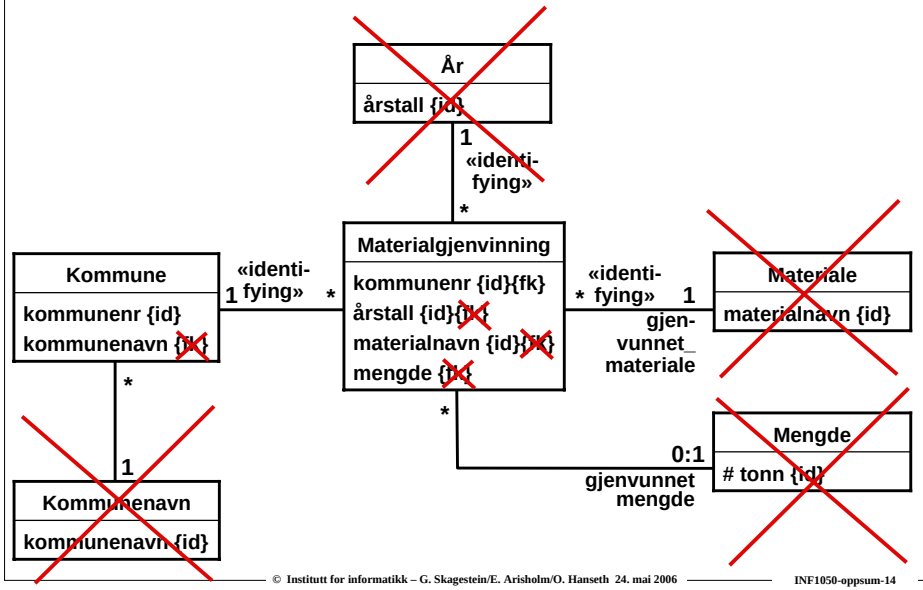
- Vi bruker bare klassediagrammer (erstatter ER- og ORM-diagrammer)
- Klassediagrammene brukes til å dokumentere ”hva systemet skal vite”
- Skranker er viktige i dataorienterte klassediagrammer
- Klassediagrammet kaster lys over terminologi og forretningsregler
- Ut fra det ferdige klassediagrammet er det mulig (automatisk?) å generere
  - SQL CREATE-kommandoer for å sette opp en relasjonsdatabase
  - Enkle skjermbilder for kommunikasjon med databasen
  - Enkle programmer som knytter skjermbildene til databasen

Dataorientert klassediagram (ugruppert)

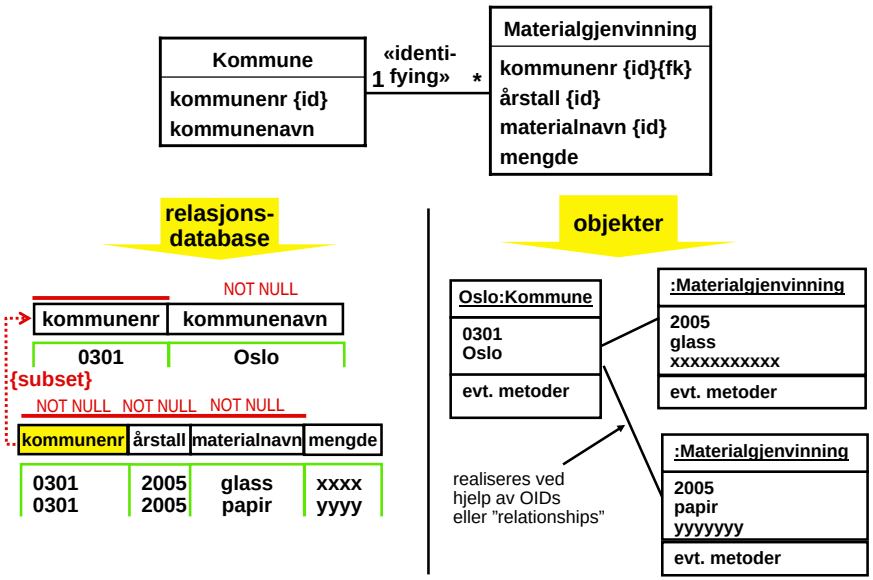
jf. figur 5-10



Dataorientert klassediagram (gruppert)



Realisering av et klassediagram



Sjekk datamodellen

- Er representasjonene i samsvar med begrepene?
- Er assosiasjonene velvalgte og fornuftige?
  - Gir de interessante opplysninger?
  - Er det sikkert at de ikke kan avledes?
  - Hvis tvil: Lag forekomsttabeller!
- Er multiplisitetene korrekte?
  - Hvis tvil: Lag forekomsttabeller!
- Er roller angitt der hvor latmannsregelen ikke holder?
- Er homogenitetsregelen anvendt?
- Er viktige skranker dokumentert?

Sjekk relasjonsdatabasestrukturen

- Er fremmednøkkelen generert på riktig side?
- Er {null} angitt for fremmednøkler der minimumsmultiplisiteten er 0?
- Har du fått med deg eventuelle sammensatte fremmednøkler?
- Er delmengdeskranker satt på mellom fremmednøkler og primærnøkler?
- Er undertrykking av tabeller begrunnet?
- Er NOT NULL angitt for alle ikke {null} attributter?
- Er valg av partisjonering, absorbsjon eller separasjon ved gruppering rundt underbegreper begrunnet?
- Er eventuelle avvik fra tredje normalform/BCNF begrunnet?
- Finnes det noe i relasjonsdatabasestrukturen som ikke finnes i datamodellen, eller omvendt?

Hva jeg håper du har lært

- Du forstår hva dataorienterte klassediagrammer (i motsetning til objektorienterte) uttrykker
- Du har fått en overordnet forståelse av hvordan dataorienterte klassediagrammer kan brukes
  - til å spesifisere krav om "hva systemet skal vite"
  - som grunnlag for å utforme en relasjonsdatabase (tabelldatabase)
  - som grunnlag for å utforme systemer rundt den sentrale databasen
- Du innser at det er et poeng å gjøre modellen så minimalistisk som mulig (ingen dobbeltlagring av kunnskap!), og du vet hvordan du skal få det til
- Du innser at skranker er viktige, og er i stand til å sette opp de mest sentrale i en modell
- Du kan nok SQL til å sette opp og foreta enkle spørringer i en relasjonsdatabase

Fra skallet og inn

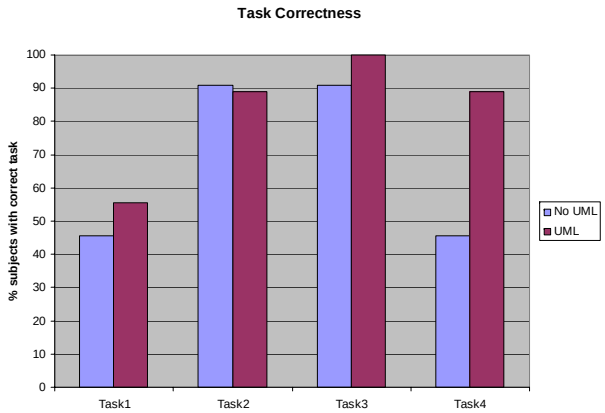
- UML i OO systemutvikling
- Den objektorienterte Inf1050-metoden
- Hva jeg håper dere har lært (☺)

Hva brukes UML til i OO systemutvikling?

- Notasjon som støtter opp under (et virvar av) metoder for
  - objektorientert analyse ("hva systemet skal gjøre"), og
  - objektorientert design ("hvordan systemet skal gjøre det")
- Dokumentasjon av systemets krav, design og implementasjon
  - for andre utviklere (og for deg selv)
  - Evt. for kommunikasjon med kunde/sluttbruker under utviklingsprosjektet
- Har også andre anvendelser, som for eksempel:
  - Generere prototyper, generere kode, simulere, lage tester, gjøre designinspeksjoner, måle designkvalitet før systemet kodes, automatisk restrukturering, gjenkjenning av designmønstre.

Nytteverdien av UML som dokumentasjon for nye utviklere – resultater fra et kontrollert eksperiment*

(m/Java endringsoppgaver og Tau UML)



* E. Arisholm, L. C. L. Briand, S. E. Hove and Y. Labiche. The Impact of UML Documentation on Software Maintenance: An Experimental Evaluation, Accepted for publication in IEEE Transactions on Software Engineering, 2006.

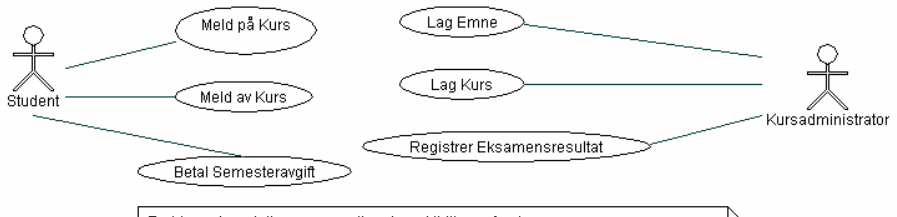
Dagens tilstand og veien videre

- **UML – Et tegneverktøy eller fremtidens systemutviklingsspråk?**
- **De fleste utviklere i dag bruker UML som et rent tegneverktøy for å lage relativt uformelle modeller av IT-systemer, som de deretter "koder" i et programmeringsspråk**
  - **Behov for å oppdatere ikke bare kode, men også UML-modellene, dersom de skal ha verdi som fremtidig dokumentasjon av et system.**
  - **Ofte gidder ikke systemutviklere benytte seg av eksisterende modeller for å forstå et system fordi**
    - **de av erfaring regner med at modellene ikke er oppdatert?**
    - **modellene er for upresise eller ufullstendige, slik at man likevel må se på koden?**
    - **koden uansett er den beste dokumentasjonen?**
- **Modelldrevet utvikling (MDA/MDD):**
  - **Toneangivende grupperinger (OMG, IBM m.fl.) begynner nå å lage teknologier hvor UML-modeller (tilsvarende de vi har laget i Inf1050) i praksis blir en sentral del av "koden".**
  - **Integrert verktøystøtte med automatisk kodegenerering er sentralt**

Inf1050 metoden (Iterativ)

- **Analyse av krav**
  - **(1) Identifiser aktører og deres mål**
  - **(2) Lag et høynivå bruksmønsterdiagram**
  - **(3) Spesifiser hvert bruksmønster tekstlig med normal hendelsesflyt og variasjoner**
- **Objektdesign**
  - **For hvert bruksmønster:**
    - **(4) Identifiser objekter og fordel ansvar mellom dem (CRC)**
    - **(5) Lag sekvensdiagram for normal hendelsesflyt og viktige variasjoner**
    - **(6) Lag klassediagram som tilsvarer sekvensdiagrammene**
  - **(7) Lag til slutt klassediagram på systemnivå**

Kursregistrering bruksmønstermodell



Endringer i modellen sammenlignet med tidligere forslag:
1) Vi skiller mellom "Emne" og "Kurs"
2) "Lag Emne" og "Lag Kurs" inneholder "Nytt Emne" og "Nytt Kurs" som variasjoner
3) Bruksmønster "Behandle dispensasjonssøknad" utgår
4) Variasjon på "Meld på kurs":
Dersom et kurs er fullt kan studenten velge å bli satt på venteliste.
5) Variasjon på "Meld av kurs":
Det finnes studenter på venteliste: Første på ventelista får plassen

Spesifikasjon av ”Meld på kurs” (i hht mal)

Navn: Meld på kurs

Aktør: Student

Trigger: Student ønsker å melde seg på et kurs

Pre-betingelse: Student har betalt semesteravgift og er logget inn på systemet

Post-betingelse: Student er meldt på kurset eller er satt på venteliste

Normal Hendelsesflyt:

1. Studenten velger emne
2. Systemet sjekker at studenten kvalifiserer til å ta emnet
3. Systemet finner kurs for emnet
4. Systemet sjekker om det er ledig plass på kurset
5. Systemet registrerer studenten på kurset

”Meld på kurs” (forts.)

Variasjoner:

1a. Emnet finnes ikke:

1. Studenten velger et annet emne eller avslutter

2a. Emnet forutsetter andre emner:

1. Systemet sjekker at studenten har bestått kurs for emner som forutsettes

1a. Studenten har ikke bestått kurs for emner som forutsettes:

1. Studenten velger et annet emne eller avslutter

3a. Det holdes ikke kurs i emnet dette semesteret:

1. Studenten velger et annet emne eller avslutter

4a. Kurset er fullt:

1. Systemet spør om studenten ønsker å bli satt på venteliste

1a. Studenten ønsker å bli satt på venteliste:

1. Systemet setter studenten på venteliste

Relatert informasjon:

I denne versjonen holdes administrasjon av gruppeundervisning utenfor systemet

Tre typer objekter

- **Forretningsobjekter (“entity objects”)**
- **Kontrollobjekter (”control objects”)**
- **Kantobjekter (“boundary objects”)**
- **Litt forenklet kan man si at denne tredelingen skiller mellom 1) objekter som skal *lagres* i en database, 2) objekter som *koordinerer* handlingene i et bruksmønster og 3) objekter som *kommuniserer* med aktørene.**

Men hvor mye ansvar bør kontrollobjektene ha, og i hvilken grad bør vi ”bevisstgjøre” forretningsobjektene våre?

Delegering av ansvar: CRC-kort for bruksmønsteret ”Meld på kurs”

Kant <<boundary>>	MeldPaaKurs
Kommuniserer med aktøren og kontrollobjektet	

Universitet <<entity>>	Emne Student
Oppslagsobjekt som vet hvilke emner og studenter som finnes i systemet	

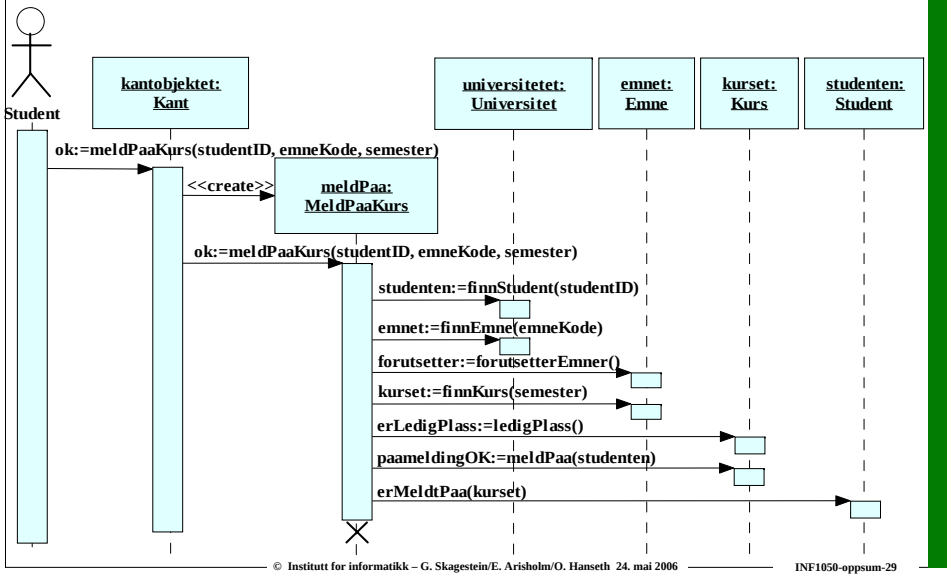
Emne <<entity>>	Kurs
Vet min emnekode Vet hvilke emner som forutsettes Vet hvordan man finner kurs i emnet	

MeldPaaKurs <<control>>	Universitet Emne Kurs Student Kant
Kontrollerer hendelses-forløpet i bruksmønsteret ”Meld på kurs”	

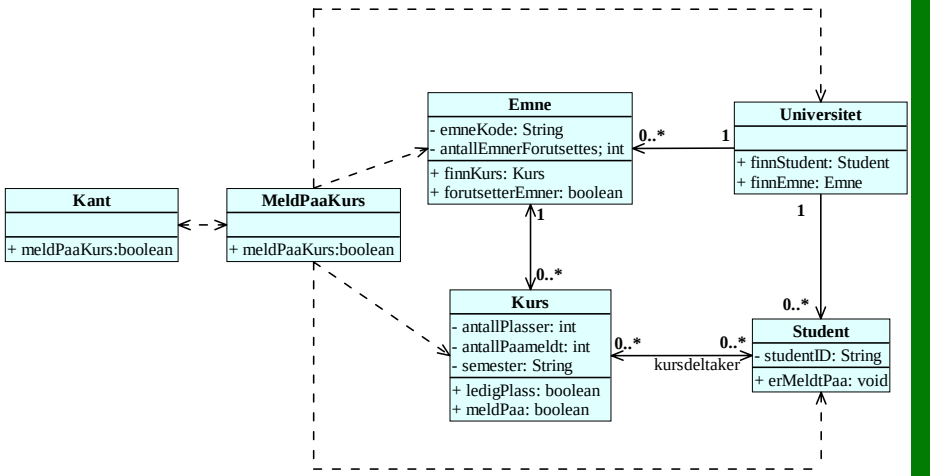
Student <<entity>>	Kurs
Vet min studentID Vet hvilke kurs jeg har tatt Vet hvilke kurs jeg er meldt opp til Vet hvilke kurs jeg står på venteliste på	

Kurs <<entity>>	Emne Student
Vet semesteret som (et kurs i) et bestemt emne holdes Vet antall plasser Vet hvilke studenter som er påmeldt Vet hvilke studenter som står på venteliste	

Normal hendelsesflyt for ”Meld på kurs”

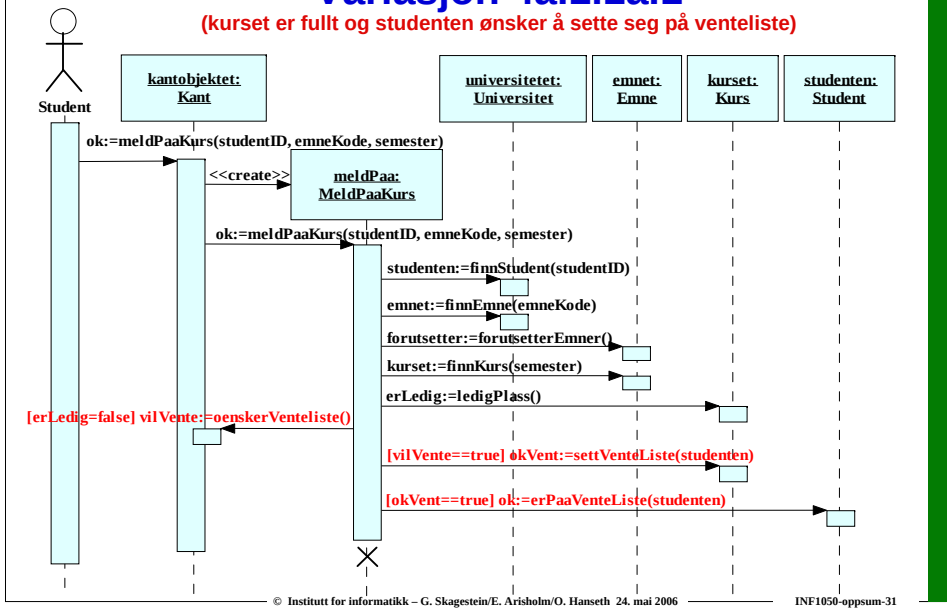


Klassediagram for normal hendelsesflyt

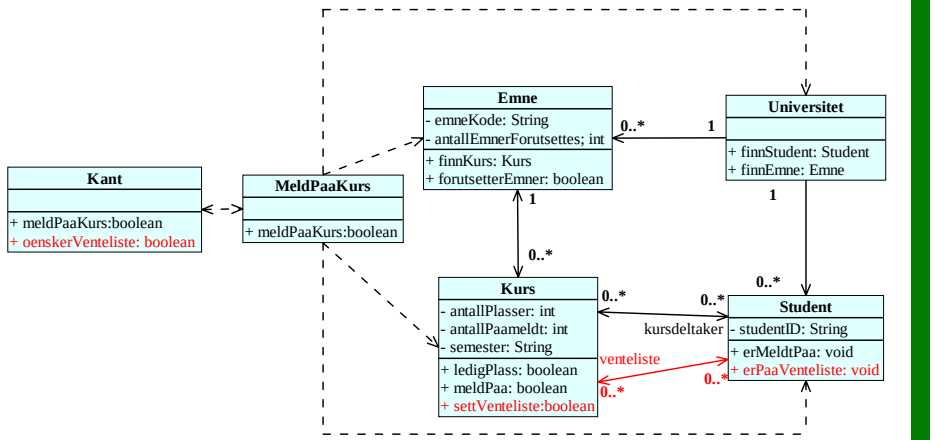


Variasjon 4a.1.1a.1

(kurset er fullt og studenten ønsker å sette seg på venteliste)

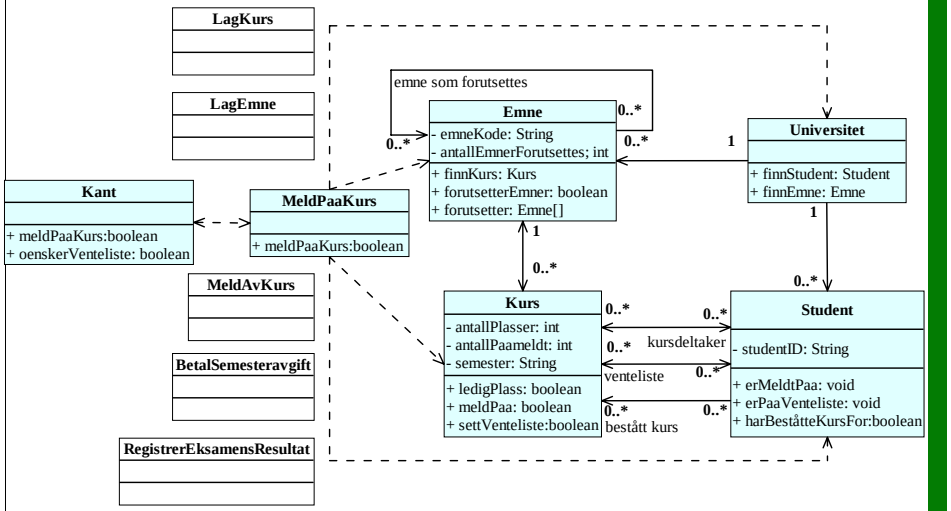


Klassediagram for normal hendelsesflyt og variasjon 4a.1.1a.1



Legger til nye metoder og assosiasjoner for variasjonen 4a.1.1a.1

Klassediagram på systemnivå



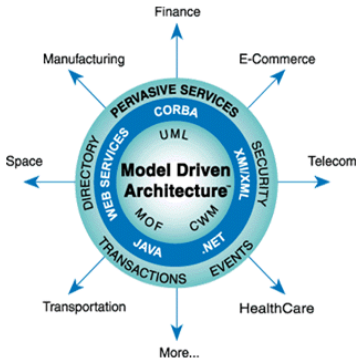
De andre bruksmønstrene vil introdusere nye kontrollobjekter og evt. nye forretningsobjekter, samt nye assosiasjoner, metoder og attributter på eksisterende forretningsobjekter. Kantobjektet vil få flere metoder (antar ett kantobjekt i systemet)

Hva håper jeg du har lært

- Du forstår hva bruksmønster-, sekvens- og klassediagrammer uttrykker og forstår sammenhengen mellom disse diagrammene
- Du har fått en overordnet forståelse av hvordan UML kan brukes til å
  - spesifisere krav, og
  - designe objektorienterte systemer som oppfyller kravene

Veien videre? ☺
http://www.omg.org/mda/

OMG Model Driven Architecture



How Systems Will Be Built

MDA® provides an open, vendor-neutral approach to the challenge of business and technology change. Based firmly upon OMG's established standards*, MDA aims to separate business or application logic from underlying platform technology. Platform-independent applications built using MDA and associated standards can be realized on a range of open and proprietary platforms, including CORBA®, J2EE, .NET, and Web Services or other Web-based platforms. Fully-specified platform-independent models (including behavior) can enable intellectual property to move away from technology-specific code, helping to insulate business applications from technology evolution, and further enable interoperability. In addition, business applications, freed from technology specifics, will be more able to evolve at the different pace of business evolution.

* Key standards that make up the MDA suite of standards include Unified Modeling Language (UML); Meta-Object Facility (MOF); XML Meta-Data Interchange (XMI); and Common Warehouse Meta-model (CWM).

Rammer

- Lover og regler
  - Personopplysningsloven
    - Relevante opplysninger
    - Samtykke i innsamling av persondata
  - Arbeidsmiljøloven
    - Medvirkningsrett
  - Regler for saksbehandling
- Teknologi
  - Hva finnes (databaser, maskiner, nettverk, web services, …)?
  - Standarder (globale, bedriftsinterne)
  - Systemer som skal integreres (bruksmessig og teknisk)
- Etikk

Styring av produksjon av informasjonssystemer

- Styringsmodell (fossefall, iterativ, inkrementell, …) velges avhengig av oppdraget (hva slags system vi skal utvikle)
  - hvilke usikkerheter er forbundet med oppdraget?
  - hvilke rammer settes for oppdraget?
- Estimering: hvor mye og hvilke ressurser trenger vi?
- Milepælsdefinisjoner
  - veldefinert, målbar prosjekttilstand
  - koblet til et bestemt tidspunkt
- Kontrakter og avtaler

Evolusjonære strategier for å håndtere usikkerheter

- **Iterativ utviklingsmodell (skrittvis forbedring)**
  - Gjentatte evalueringer, forbedringer og tilpasninger av kravspesifikasjon, arkitektur, programmer og informasjonssystemet
  - Evaluering av risiko før hver iterasjon

Barry W. Boehm (1988) A spiral model of software development and enhancement. IEEE Computer, May, 61-72

- **Inkrementell (del-leveranser)**
  - Gjentatte leveranser av mindre deler av systemet
  - Evaluering av nytte før hvert nytt delsystem utvikles

Tom Gilb (1988) Principles of software engineering management. Addison-Wesley, Wokingham, UK

Oppsummering

- *Hvem bestemmer hvilke systemer vi skal ha?*

→ Vi lager det systemet vi får til:
  - → det vi kan forstå av bruken (og brukerne)
  - → det vi kan få til teknisk (håndverk & teori)

- *Systemutviklingsprosessen*

→ Vi designer både systemet og prosessen med å lage systemet
  - → vi må velge strategi etter hvor de største problemene er

- *Brukergrensesnittdesign*

→ Brukergrensesnittet presenterer systemet for brukeren
  - → og informasjonen i systemet (presentasjon på ulike nivåer)
  - → NB husk at brukerne tenker med sin logikk

- *Evaluering av informasjonssystemer*

→ Systemet må virke i praksis
  - → riktig system (validering) og at systemet er riktig (verifisering)
    - → heuristisk testing (etter retningslinjer)
    - → realistisk testing med brukere

Informasjonssystem

Læringsmål

- Du har lært hva det innebærer å utvikle et informasjonssystem
  - hvordan man fastlegger systemets egenskaper
  - hvilke rammer som gjelder for utviklingen
  - hvordan man lager selve systemet
  - hvordan man mest effektivt får tatt systemet i bruk
  - hvordan utviklingsprosessen styres

Lykke til!