

SPACE 1 og 2

**Integrasjon (nesten) uten
programmering**

Eirik Maus

Norsk Regnesentral

eLandet Norge

19 oktober 2004

Temaer

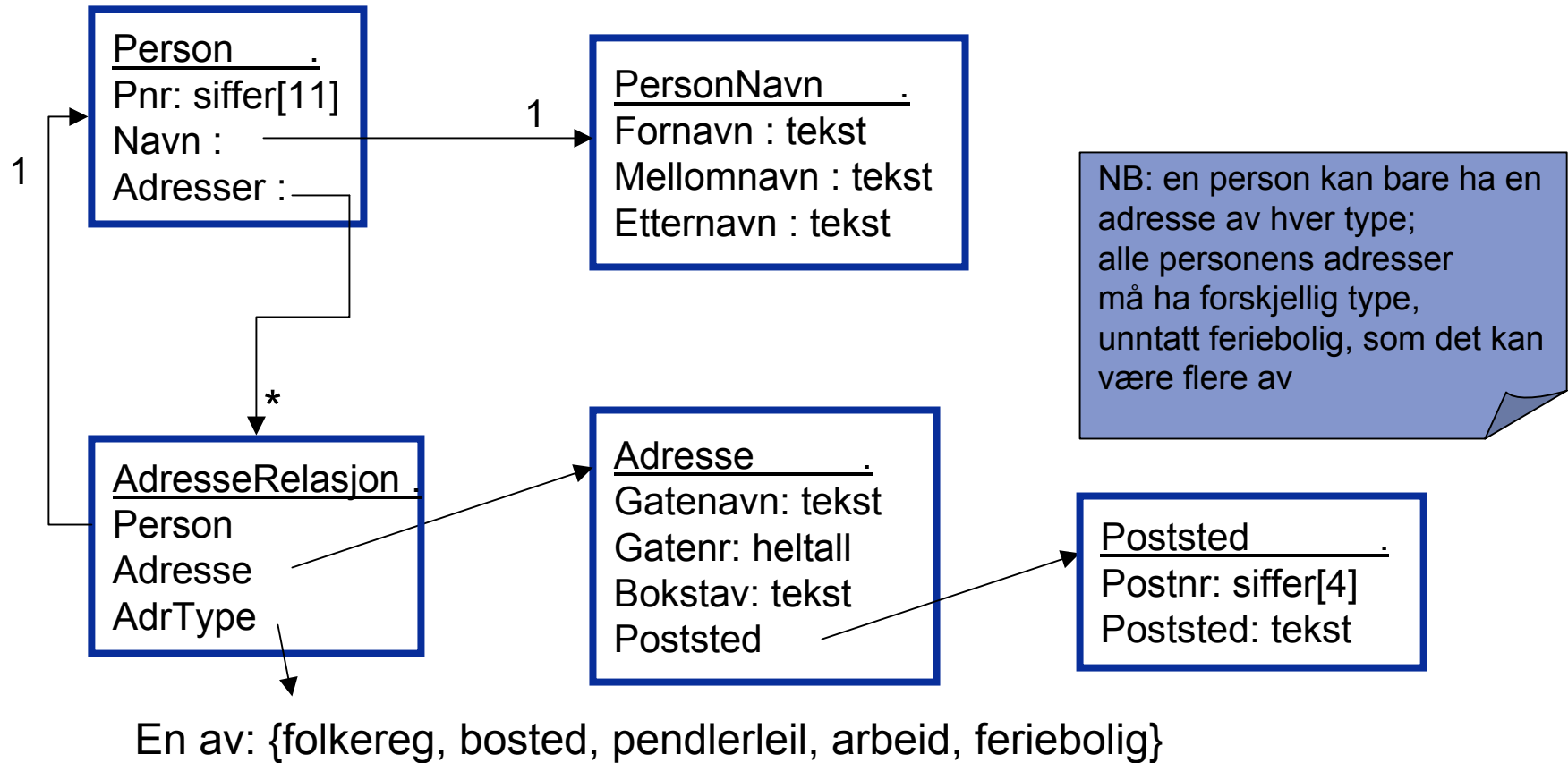
- ▶ Offentlig integrasjon og portal
- ▶ WebServices og andre tjenestegrensesnitt
- ▶ Space-prosjektet 1995-1998
- ▶ Erfaringer til foredling og videreføring i space 2

Offentlig integrasjoner

Grunnleggende prinsipper

- ▶ La datamaskiner gjøre jobben med å hente, fylle inn, søke og utveksle informasjon, ved å
- ▶ la tjenestene være tilgjengelige for dataprogrammer over internett, ved å
 - definere sammensatte datatyper (tilsv. "papirskjemaer"),
 - definere tilgjengelige grensesnitt,
 - med navngitte funksjoner,
 - som tar imot og/eller utleverer slike "skjemaer" i utfylt tilstand til programmet som brukte funksjonen.
 - (og lage logikken som får dette til å skje)

Informasjonsrelasjoner: eksempel



Tjenestegrensesnitt: eksempel

```
Datatype Person { siffer[11] Pnr, Navn navn ,... }
```

```
Interface PersonRegister
```

```
{
```

```
    MessagePort: getFolkeregAddress
```

- Inputs: Person p
- Outputs: Adresse a

```
    MessagePort: lookupPersons
```

- Inputs: Text fornavn, Text etternavn
- Outputs: Person[] treff

```
    MessagePort: addNewAddress
```

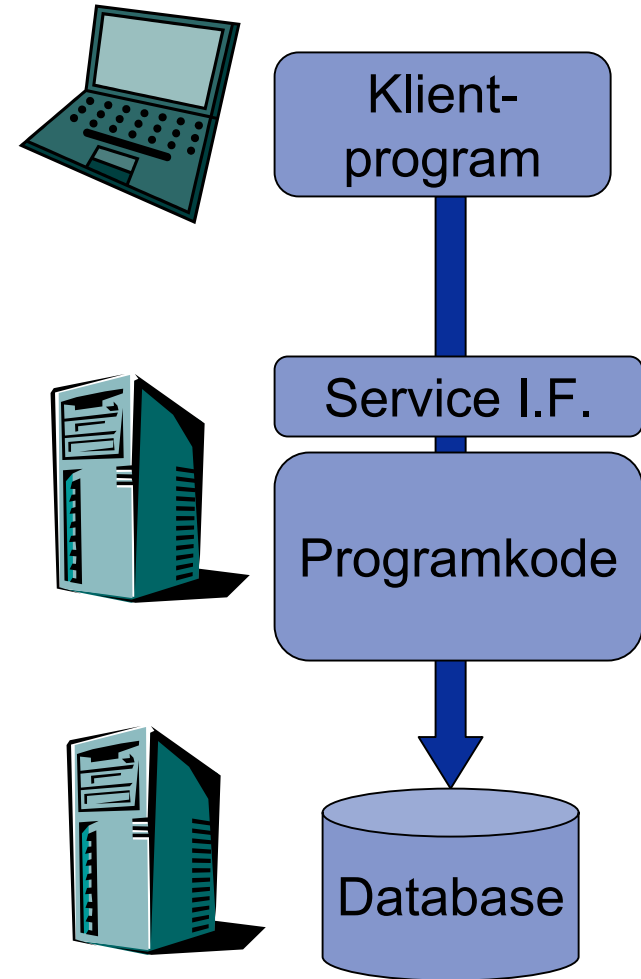
- Inputs: Person p, Adresse a, AdrType t
- Outputs: none

```
}
```

Tjenestefunksjonalitet

Det som står igjen er

- ▶ Lage programmet på baksiden som gjør noe når de ulike meldingene kommer inn
- ▶ Koble det til eksisterende datasystem
- ▶ Publisere tjenesten på en server
- ▶ Lage program som benytter tjenestene



SPACE 1994-1998

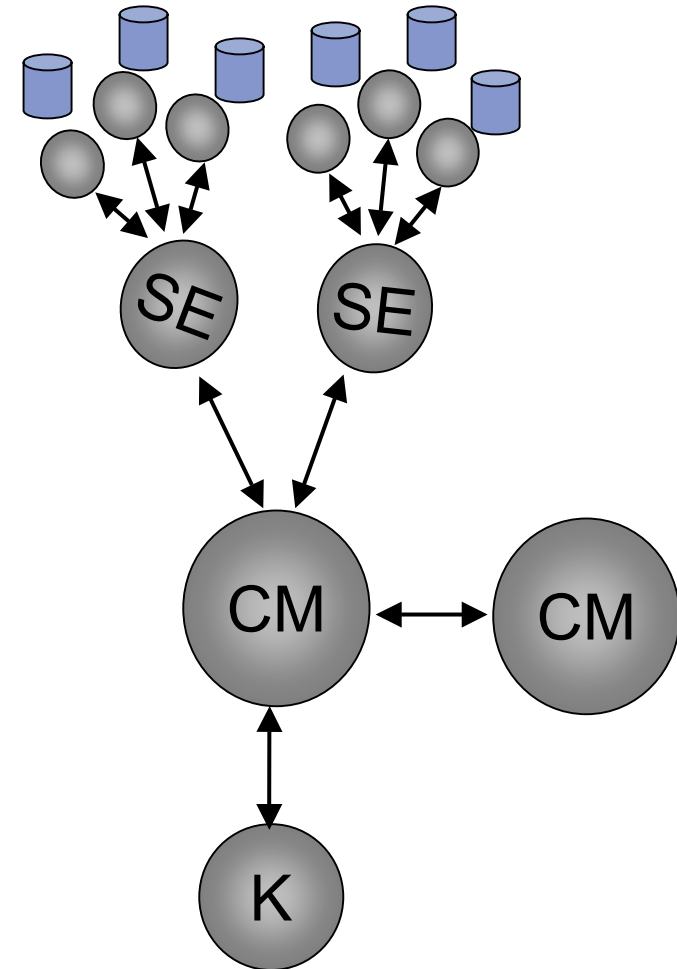
- ▶ Single Point of Access for Citizens in Europe
- ▶ Case: flytte i EØS-området
- ▶ Alle har rett til å bosette seg i EØS-området
- ▶ Men må på ørten offentlige kontorer
 - Registrere ny adresse i folkeregisteret, Flytte helse- og pensjonsforsikring, barnehagesøknad, importere bil, innregistrere i skattelister,...
 - Vise oversatt helseattest, passnummer, vielsesattest, ...
- ▶ Hva hvis alt dette kunne hentes elektronisk?
 - Med ETT besøk
 - På ETT kontor

SPACE demonstrator-system

- ▶ Demonstrator for flytting med automatisk overhenting av nødvendig registreringsinfo
- ▶ 3 use-cases
 - Give-Tailored-Advice
 - Prepare-Portfolio :
 - Retrieve-and-Open-Portfolio

Oppbygning og ansvar

- ▶ Citizen Data System wrapper
 - "database-interface"
- ▶ Sector Expert object
 - Sektor-spesifikke regler
- ▶ Country Master
 - Nasjonale regler
 - Andre land
 - Egne sektorer
- ▶ Klient
 - Egen CM
 - Brukerdialog-motor



Likheter og ulikheter

- ▶ Alle har 5 sektorer
 - Men med ulikt ansvar
- ▶ Alle har regler for info de trenger til registrering
 - Men ulike regler i hver sektor og land
 - Og ulike krav
- ▶ Alle må stille brukeren spørsmål
 - Ulike spørsål og betingelser for når
- ▶ Alle sektorer har datasystemer som kan levere info
 - Men ulike datasett hvert sted
 - Ulike nøkler behøves
- ▶ Dessuten: brukere må få dialogen på ulike språk

Konsekvenser

- ▶ Veldig mange land (12), sektorer (60) og datasystemer (hundrevis)
 - En implementasjon per tilfelle alt for dyrt
- ▶ Standardiserte grensesnitt/tjenester utenkelig
 - Alle sektorer ulike fra land til land, ulike regler
- ▶ Standardisert datastruktur/metadata umulig
 - Samme sektor har ulike databaser i landene
 - Ulike skjemaer og krav til registreringsinfo

Løsninger 1 : alle datatyper ut av programmet

- ▶ IKKE lag "hardkodede" datarelasjoner i program- eller grensesnitt-koden
- ▶ Trekk data og datastrukturer ut av meldingstypene
- ▶ Lag generell DataFelt-type
 - Med unike ID'er for de ulike felt-typene som er med i Space-systemet
- ▶ La alle metoder ta imot / returnere / behandle mengder av DataFelt

DataFelt

Verdi : text

Verditype: {tall, tekst, sant/usant}

Kilde : datakilde-ID

Feltnavn : NormalisertFeltNavn

- ▶ Nfn_person_fornavn
- ▶ Nfn_person_etternavn
- ▶ Nfn_person_fodselsaar
- ▶ Nfn_antall_barn
- ▶ Nfn_bil_merke
- ▶ Nfn_bil_modell
- ▶
- ▶ (ca 100 ulike felt-typer)

Løsninger 2 :

Beskriv Persondatasystemene

- ▶ For hvert datasystem (x/folkeregisteret)
 - Beskriv hva som kan slås opp og hvordan
- ▶ Hvilke NFN'er kan slås opp her?
- ▶ Hvilke NFN'er trengs som nøkkel til hvert oppslag?

- ▶ Når noen spør etter et datafelt, er det bare å finne databasen som inneholder feltet!

Løsninger 3 :

Fjern faste brukerdialoger

- ▶ Skal uansett bare spørre om det som er nødvendig
 - Ingen faste skjerm-skjemaer
- ▶ Dialoger lagres i sentral database
 - Med ID og språk;
 - Slås opp og vises ved behov

Løsninger 4 :

Regelstyrte dialoger

- ▶ Brukerspørsmål gir resultat:
 - DataFelt med NFN og verdi
- ▶ Regler avgjør hvilke dialoger som må vises
- ▶ Evaluerer reglene på bakgrunn av tidligere spørsmål og svar
 - DataFelt med NFN og verdi
- ▶ Hvert land / sektor har egne dialog-regler i egen database

Løsninger 5 :

Eksplisitte Regler for dok-krav

- ▶ Hovedspørsmålet:

Hvilken dokumentasjon trenger hver sektor i det nye landet for innregistrering?

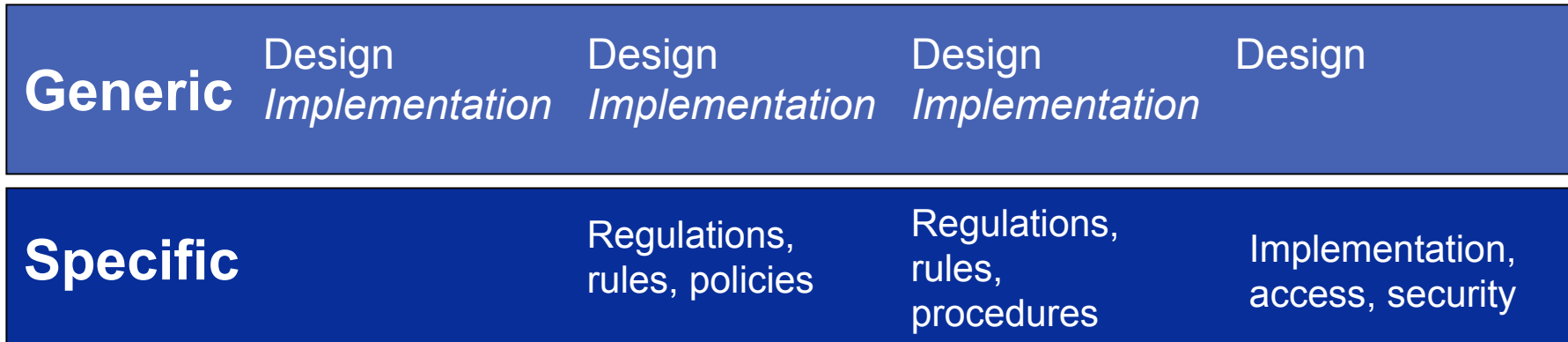
- ▶ Enkelt regelsett per land og sektor:

*"if nfn_tar_med_bil = true
then require nfn_bil_merke, nfn_bil_modell ..."*

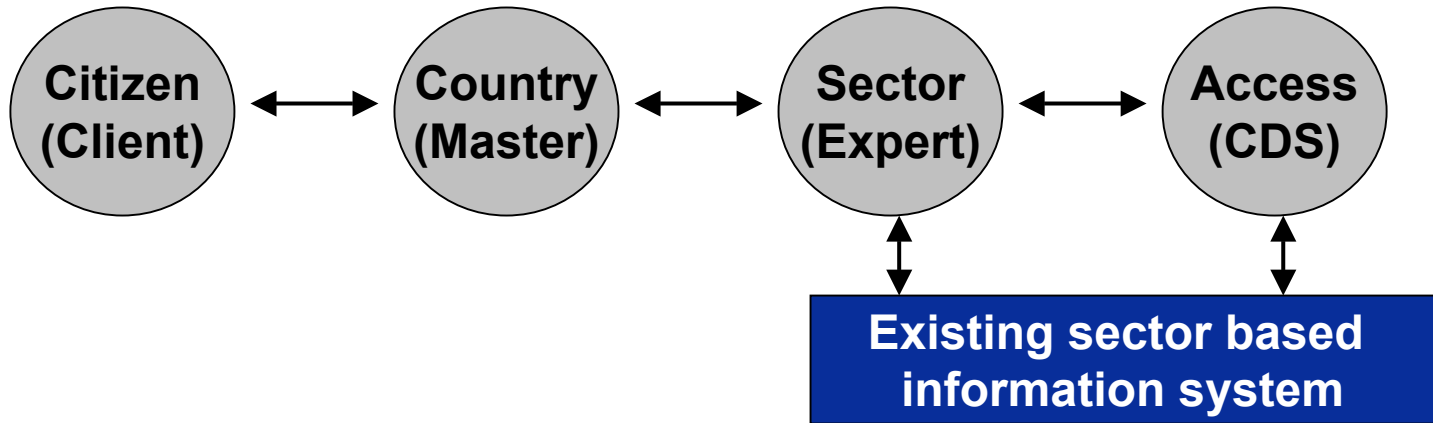
Gang i programmet

1. Bruker velger destinasjonsland
2. Henter generell dialog fra landet
3. Henter dialog-spørsmål på rett språk og viser
4. Henter spesifikk dialog fra sektorene i dest-land
 1. Sektorer sjekker generelle svar mot regler
 2. Returnerer mer spesifikk dialog iht regler
5. Henter krav til info / dokumentasjon fra dest-land og sektorer
 1. Sjekk svar mot regler om krav
 2. Returner de hvor regel traff
6. Skaff fra databaser
 1. hvilke databaser har disse info
 2. Hva trengs av nøkler? Spør bruker hvis mangler
 3. Slå opp data
7. Pakk og send til mottaker(e)

GENERIC TECHNICAL CONCEPT On Top of Existing System



**SPACE
system**



Scalability

New countries can be added by filling in data for a country master.

New sectors can be added by filling in data for a sector expert and access to information systems.

Resultater 1: Skille oppgave og policy

- ▶ Oppgave
 - Finn ut hva som trengs for å registrere flyttet person
 - Still nødvendige spørsmål
 - Finn ut hva som kreves av input til de ulike
 - Skaff riktige data og send over
- ▶ Policy (“forvaltningslogikk”?)
 - Hvilke regler som gjelder
 - Hvilke data som trengs under hvilke betingelser
 - Hvor dataelementer finnes
 - Hvordan man slår dem opp
 - Hvordan spørre bruker om det som ikke finnes lagret

Resultater 2: Gjenbrukbar kode

- ▶ En sektor-regelmotor-implementasjon kan brukes for samtlige 60 sektorer
 - Må kun fylle regel-databasen med andre regler
- ▶ Kan tilpasses nytt lovverk på 15 minutter!
 - Bare skriv om regler som blir påvirket og restart
 - **INGEN REPROGRAMMERING**
 - Ingen program-messige bindinger til lovverket

Resultater 3 : Men virker det da?

- ▶ Det er jo bare en demonstrator
- ▶ Identisk kopi av finsk folkeregister ble tilknyttet systemet
 - Plunder pga ikke-støttet hardware-plattform
 - Totalt tidsforbruk: 50 timerverk

SPACE2 : forbedringer 1

- ▶ Enklere dialogmekanisme.
 - Knytt brukerdialog direkte til NFN
 - Skal jo uansett spørre om det som ikke kan finnes i databaser og gjøre om svaret til NFN
- ▶ Støtt data-relasjoner og flere objekter
 - Space 1 har bare “bilens alder” “barnets navn”, “personens navn” osv.

SPACE2 : forbedringer 2

- ▶ Støtt flere og vilkårlige “mål” for bruker
 - Ikke bare “flytt til riket”
- ▶ Støtt nesting av mål:
 - “fravæersmelding + grunn:syk
=> utfør mål:sykemelding”
- ▶ Støtt vilkårlig lange runder med å evaluere betingelser for dialog / data som behøves
 - Ikke bare 2 runder (land-regler + sektor-regler)
- ▶ Enklere men kraftigere regler,
 - et sted mellom SQL og Excel-formler

Konklusjon

- ▶ Integrasjon mellom sektorer mulig nesten uten programmering
 - Endring av reglene er hvertfall mulig uten programmering
- ▶ Noe må lages for hånd
 - Hente data ut av databasen
 - og "lime det inn" i fagsystem
- ▶ Slipper vi de dyre konsulentene?
 - Ja, hvis dere klarer å lage reglene selv ...

Konklusjon 2

- ▶ Forutsetning for suksessen:
 - Modellere data og relasjoner først
 - Databeskrivelser utenfor systemet, ikke del av grensesnitt
 - Modellere forvaltningsregler utenfor implementasjonskoden
 - Lage generiske, fag-uavhengige grensesnitt for utveksling av regler og datafelt

Offentlig integrasjon: aktuelle problemstillinger

- ▶ Sentralisert "portal"
 - eller adhoc-linker ved behov?
- ▶ Enhetlig strukturering av hvordan info-elementer henger sammen med hverandre først
 - eller funksjoner/grensesnitt først og se hva de trenger av info inn og ut for å fungere
- ▶ Ad-hoc tjenestegrensesnitt
 - eller noen (få?) standardiserte grensesnitt-typer

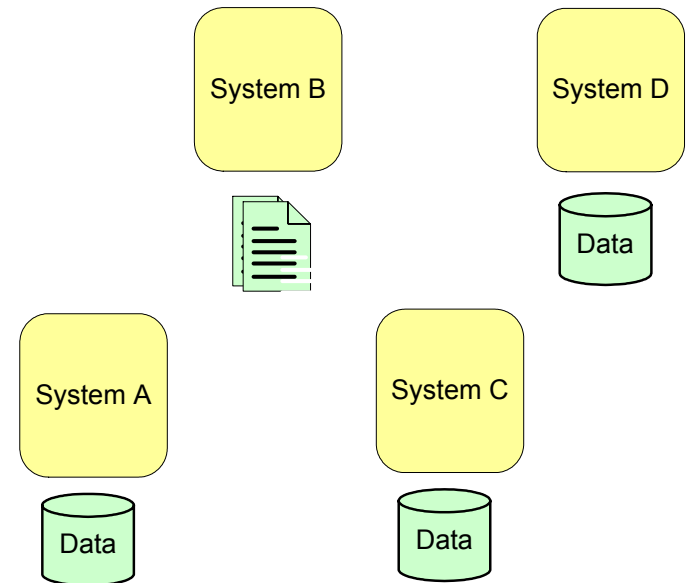
slutt

▶ Spørsmål?

Reserverfoiler: "integrasjon"

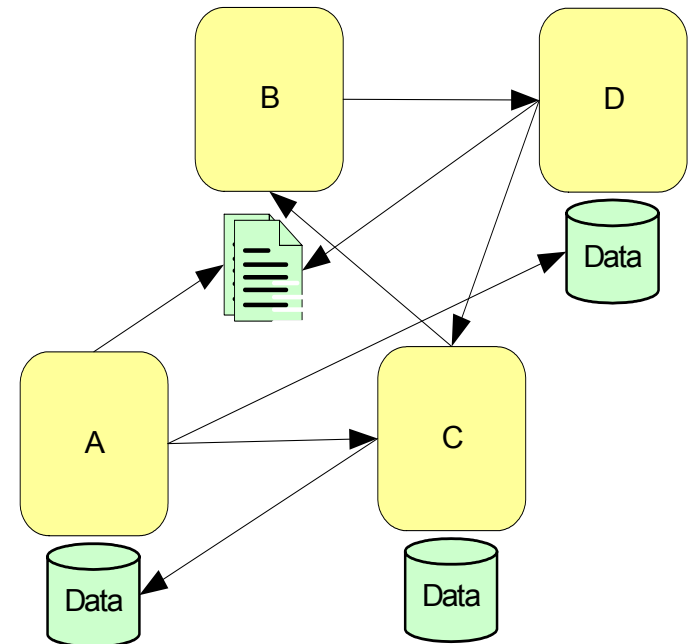
Fort og gæli: Lappverkintegrasjon

- ▶ Tenkt avdeling med 4 siloer
 - ▶ (En fylkesmiljøavdeling i danmark hadde 47 siloer i daglig bruk)
- ▶ ”Det er jo dumt å skrive ting inn i A og B som alt finnes i C”
- ▶ ”så da bare henter vi ut data fra C til A”
- ▶ ”vi kan gjøre det samme igjen”



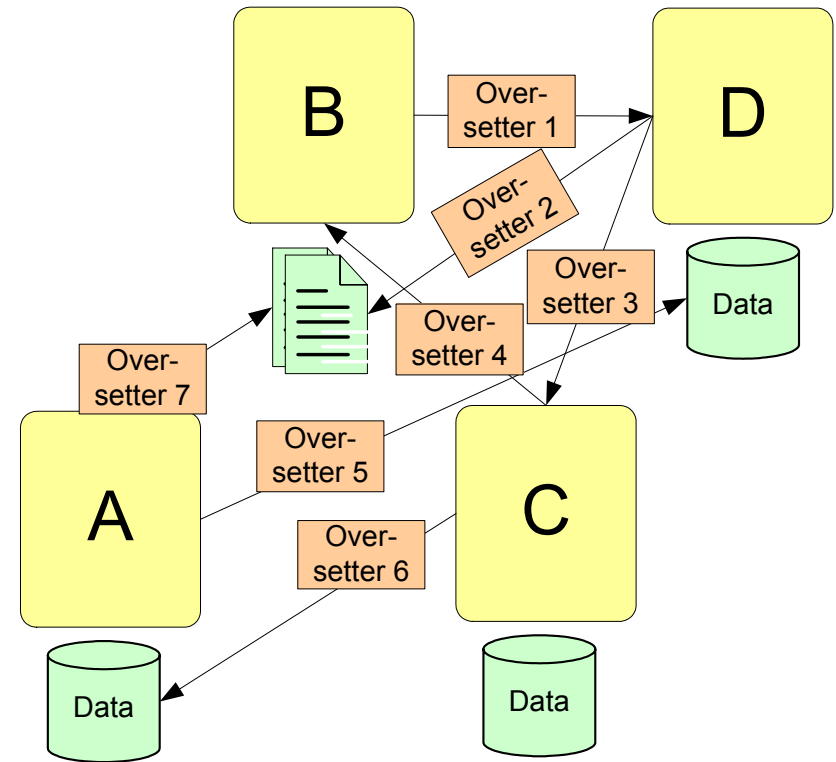
Hmmm

- ▶ ”Og så lager vi noen nye tjenester på tvers av de gamle systemene”
- ▶ ”Ja, god ide. Vi kan gjøre det flere ganger”



Dobbelt-hmmm

- ▶ ”Hei, vent, dataene er jo ikke like”
- ▶ ”Nei, men vi bare setter opp en boks som oversetter mellom dem, det er enkelt”
- ▶ ”Vi bruker XML! Det kan vi gjøre alle stedene. Det går fort.”

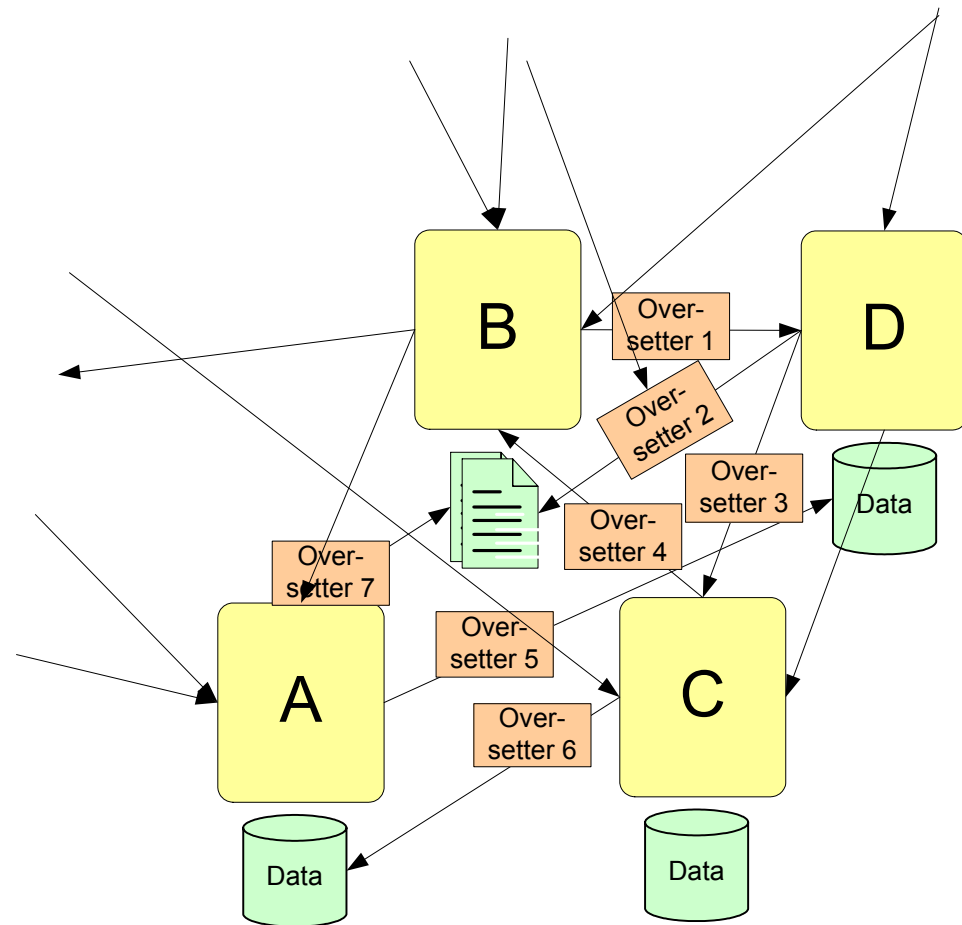


”Hei det er fra Drift. Kan vi ta ned system C for vedlikehold?”

Trippel-hmmm

Ledelsen:

- ▶ ”Vi lager noen nye integrerte tjenester på tvers av de gamle fagsystemene”
- ▶ ”Ja, integrasjon er synergi”



”Hei det er fra Drift. Den gamle serveren stoppet. Vet dere hvilken rekkefølge det skal startes opp i?”

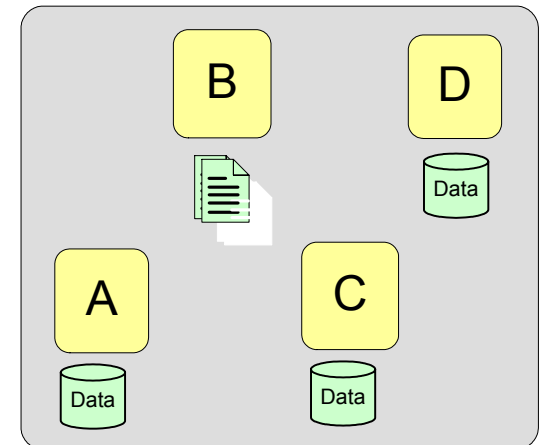
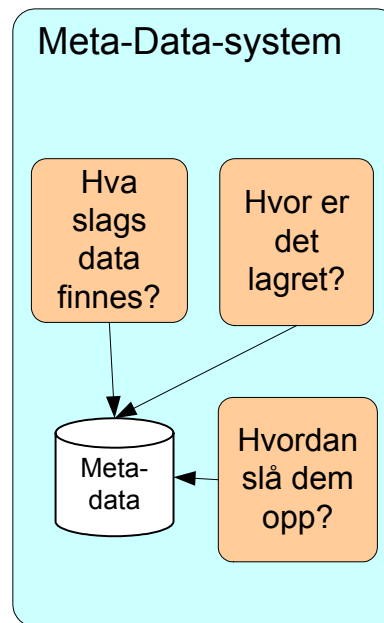
Hva skjedde?

- ▶ Hva gikk galt?
 - ▶ Ingen oversikt over hva som finnes av data i systemet og hvor det kommer fra
 - ▶ Kan aldri fjerne eller stoppe noe: Ingen aner sammenhengen, rekkefølgen eller konsekvensene av de mange ulike systemene
 - ▶ Enhver feil kan forplante seg hvor som helst
- ▶ Hva er bra?
 - ▶ Fikk resultater tidlig. For tidlig?

Reserverfoiler : space 2

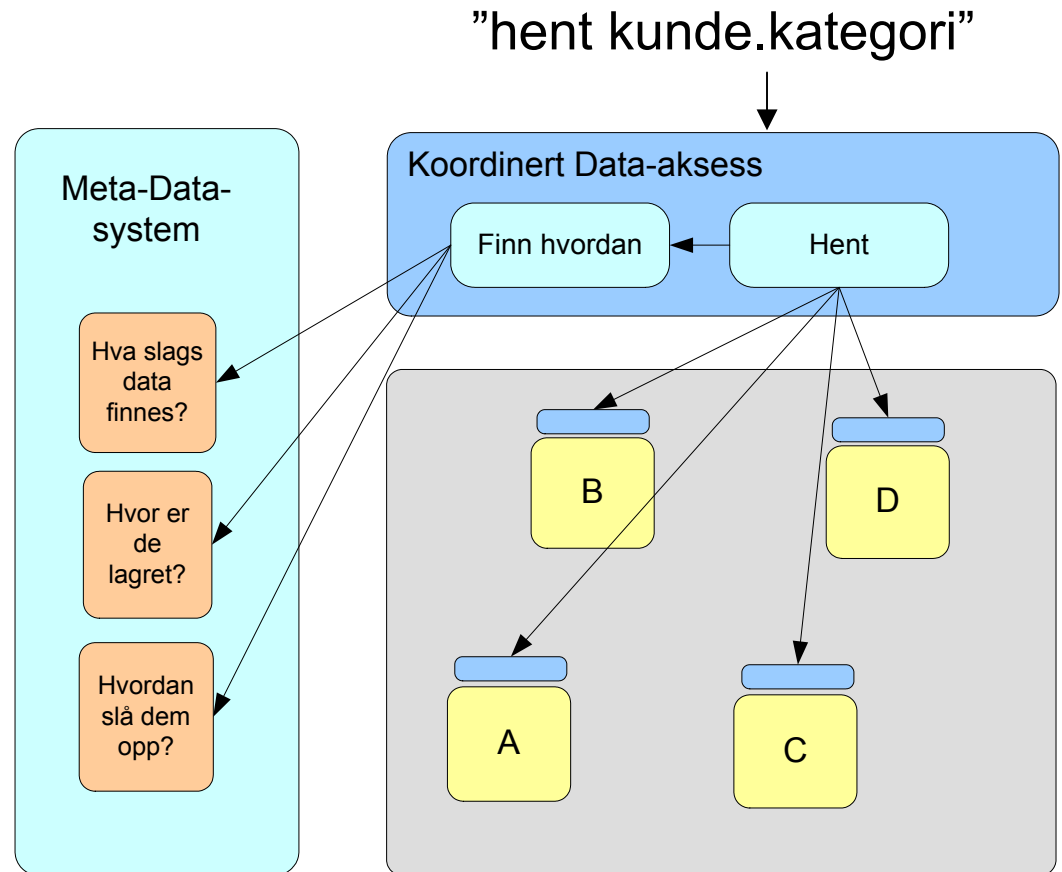
Space 2.1: Lag oversikt over data

- ▶ Hvilke data finnes?
- ▶ Hvor?
- ▶ Og hvordan hente verdien for det feltet?



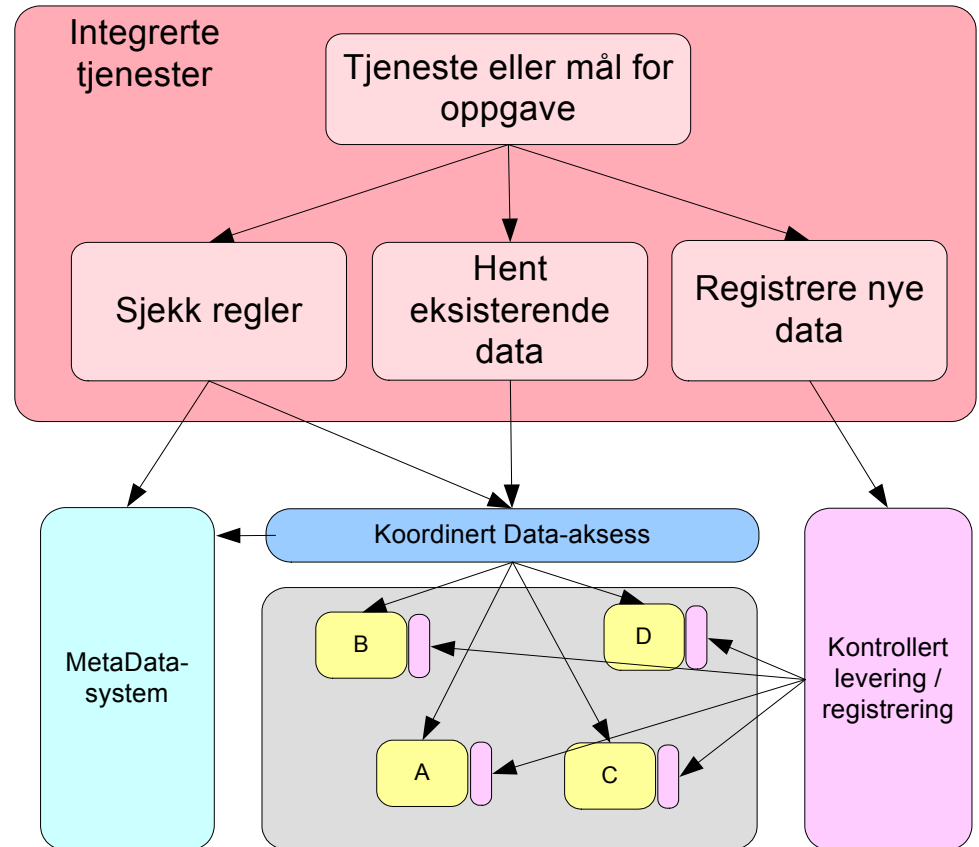
Space 2.2: Koordinert datainnhenting

- ▶ Generell felles tjeneste for å hente data ut fra hvilket som helst system
- ▶ En lokal oversetter for hvert system
 - ▶ Ikke en per klient
 - ▶ Ganske like:
 - ▶ mye gjenbruk



Space 2.3: Integrerte tjenester

- ▶ NÅ kan integrerte tjenester lages uten å skape kaos
- ▶ Kan alltid sjekke i regler og metadata om et system er i bruk
- ▶ Trenger ikke tenke på hvilke systemer data ligger i

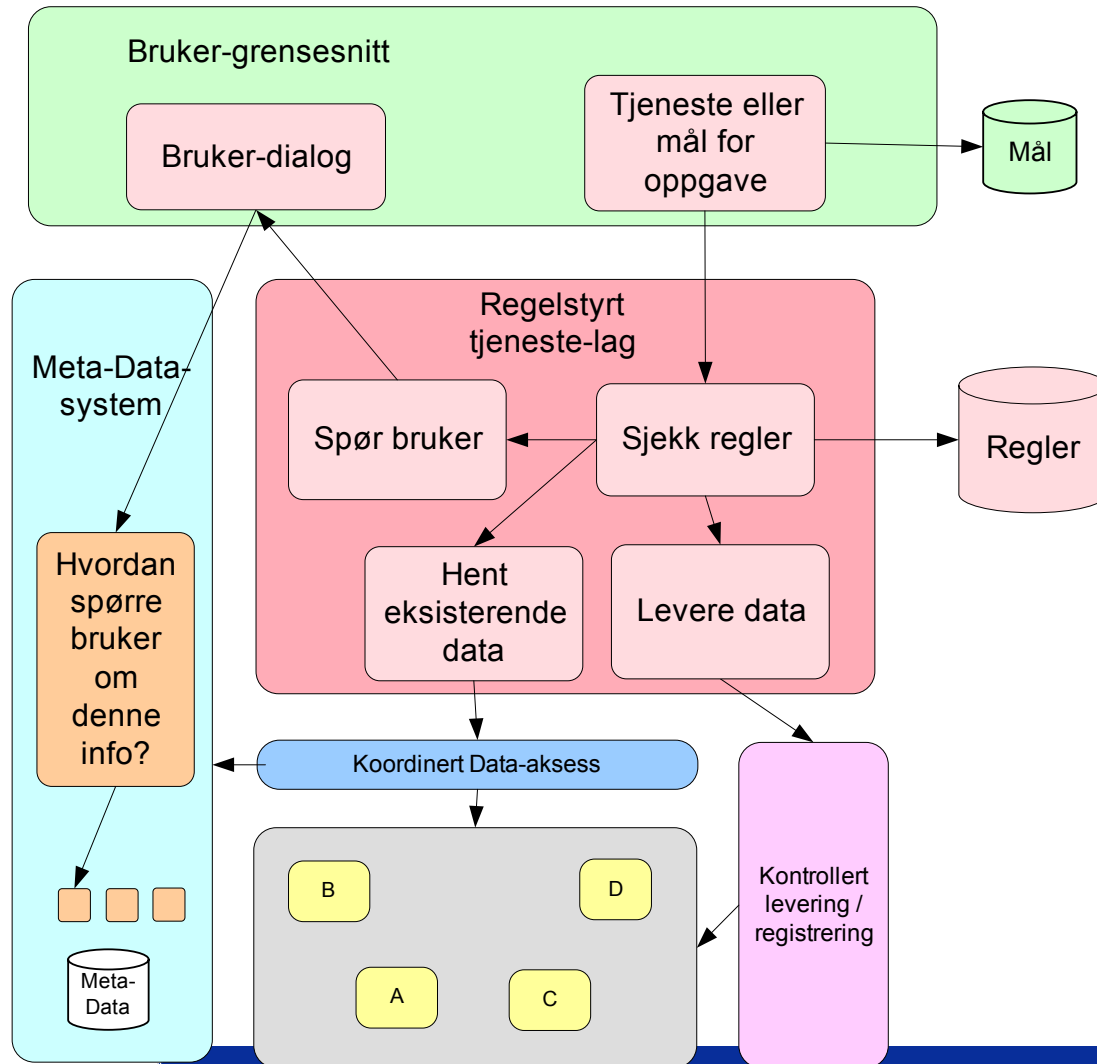


Space 2.4:

Regel- og måldefinerte tjenester

- ▶ Mål: Nye tjenester (av liknende type) uten å programmere!
(typisk oppgave: “registrere-bestilling-stamkunde”)
- ▶ Spesielt for hver tjeneste
 - ▶ Målet / oppgaven som utføres
 - ▶ Hvilke data som skal fylles inn: regler
- ▶ Felles for hver tjeneste / oppgave
 - ▶ Måte å navngi oppgaver (og del- / underoppgaver)
 - ▶ Måte å hente data
 - ▶ Måte å angi hvilke data som må innhentes til oppgaven
- ▶ Men hva med?
 - ▶ Måte å finne ut hvilke data som trengs til oppgaven
 - ▶ Brukergrensesnitt?

Space 2: Regel-basert tjeneste



SPACE 2: Alle sektorer "like"

- ▶ Felles
- ▶ Sektor-oversikt
- ▶ Data-innhenter
- ▶ Regel-innhenter
- ▶ Data-levering
- ▶ Per sektor:
- ▶ Levere data
- ▶ Egne regler
- ▶ Datamottak

