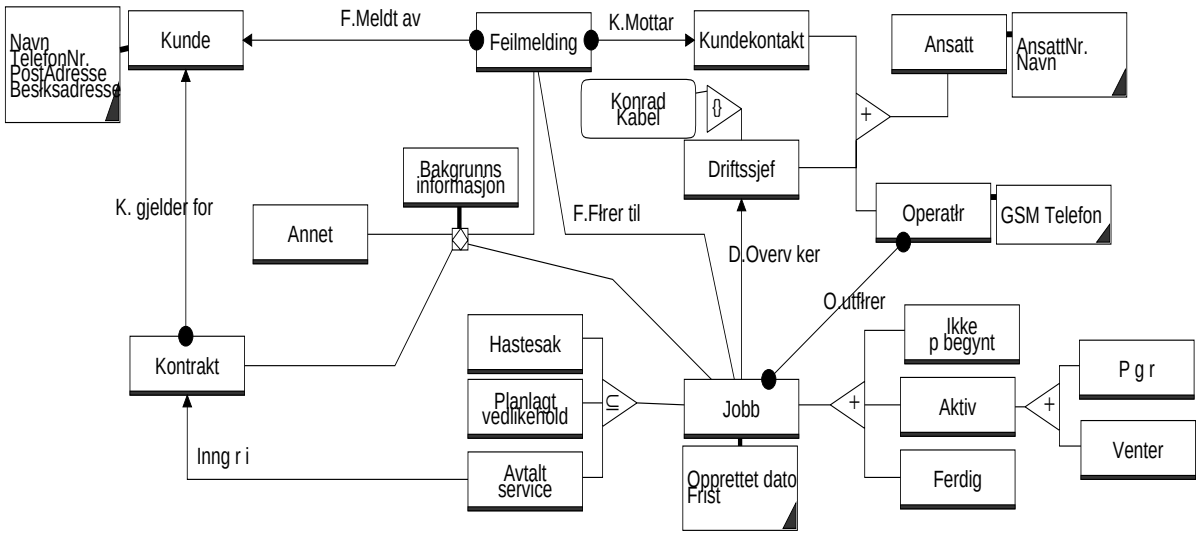
NORGES TEKNISK-
NATURVITENSKAPELIGE UNIVERSITET
INSTITUTT FOR DATATEKNIKK OG INFORMASJONSVITENSKAP

EKSAMEN I FAG 78050 (45160) SYSTEMERING 1
Onsdag 12. mai 1999
Tid: kl. $09^{\underline{00}}$-$13^{\underline{00}}$

Løsningsantydning

Oppgave 1 – Referent Modell (20%)

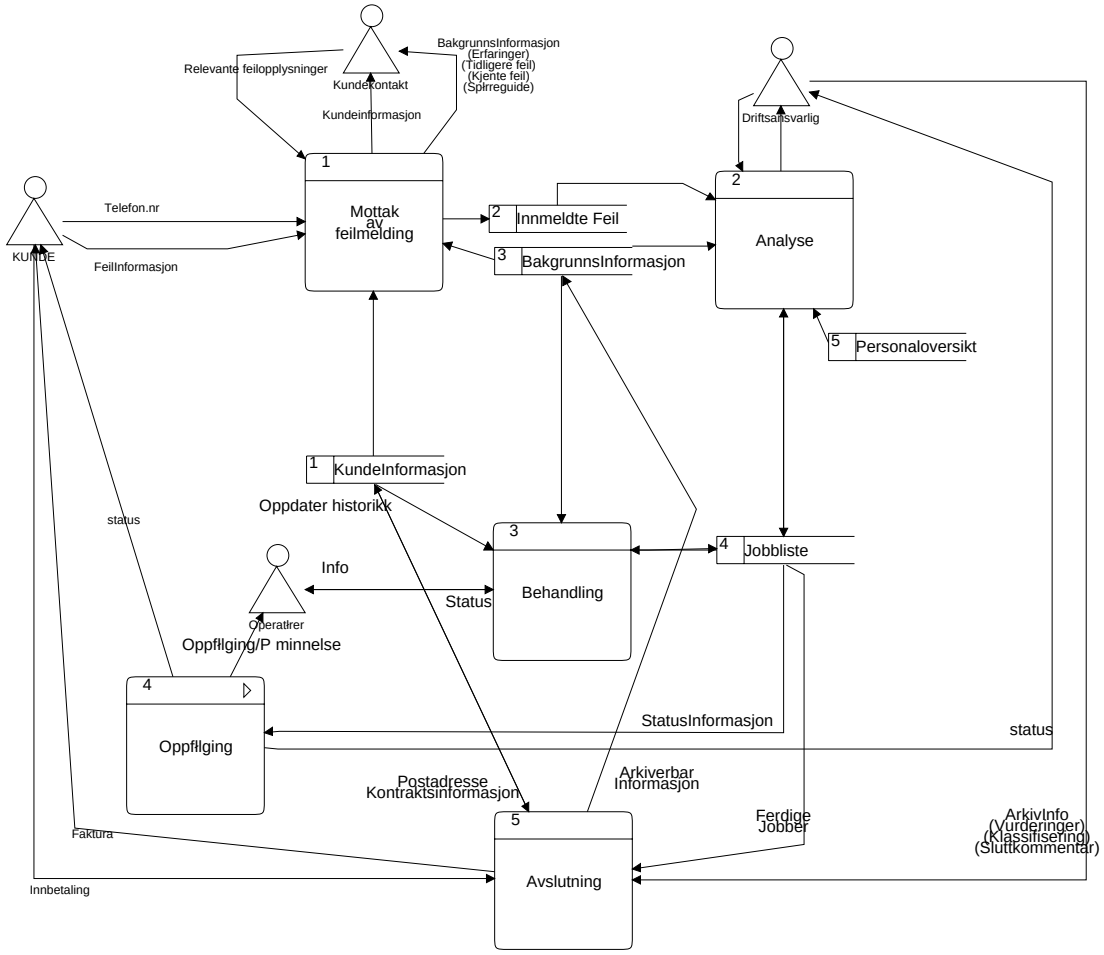


Kommentarer:

- Antar at det meste er sentrert rundt begrepet Jobb. Alt arbeid som skulle utføres – inklusive avtalt service, og planlagt vedlikehold - ble registrert som jobber. Alle jobber har en status. Dette har vi modellert ved å dele jobbene inn i ikke overlappende delmengder avhengig av hvilken status de har.
- Operatøren analyserer feilmeldinger og oppretter evt. en jobb. Alle jobber overvåkes av en ansvarlig driftssjef. Alle jobber må ha en operatør (evt. flere for stor-jobber) Ferdige jobber er også knyttet til en operatør, nemlig den operatøren som utførte jobben. Alle operatørere må ha noe å gjøre - lediggang er roten til alt ondt …
- En feilmelding kan skape flere jobber (kanskje noe søkt ??) og en jobb kan være et resultat av flere feilmeldinger.
- Som et lite ”triks”, er bakgrunnsinformasjonen representert som en egen referent, som ”består av” jobber, feilmedlinger, kontrakter og ”annet” etc. Det er i oppgaveteksten litt vanskelig å få tak på hva som egentlig er bakgrunnsinformasjon og hvordan denne håndteres. Samtidig er det et begrep som så mange skal ha tak i, derfor har vi modellert dette slik for å vise at dette er en samling utvalgt informasjon – ”hentet fra” de andre begrepene som er representert.
- Ingen datalagre er modellert. Vi kunne (burde kanskje?) ha markert med ”datalager”-symboler både på kunde (”kunderegister”) og på bakgrunnsinformasjon (”web-sider”).

Oppgave 2 – Prosessmodellering (DFD) (40%)

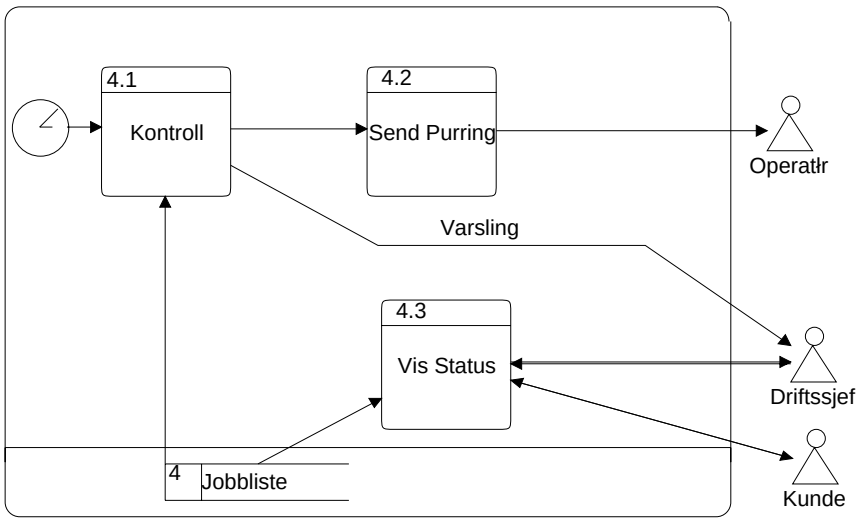
a) Overordnet DFD:



Kommentar:

- **P1: Mottak og registrering:** Prosessen registrerer kundens telefonnummer og slår opp kundeinformasjon. Prosessen leser også fra relevant bakgrunnsmateriale slik at dette er tilgjengelig for kundekontakten etter behov. Kundekontakten registrerer feilmeldingen i systemet
- **P2: Analyse:** Driftsansvarlige bruker bakgrunnsinformasjon til å tolke feilmeldinger og forstå hva som er galt. Driftsjefer har også oversikt over andre jobber som skal gjøres og tilgjengelig personale. Personell fordeles på jobber som lagres i systemet.
- **P3: Behandling:** Operatører får tilsendt jobber og jobbinformasjon. Prosessen leser ut og gjør nødvendig bakgrunnsinfo tilgjengelig. Vi antar at det er denne prosessen operatørene henvender seg til for å få mer informasjon om jobben.
- **P4: Oppfølging:** Prosessen sjekker ved gjevne mellomrom status og tidsfrister på alle jobber. Purring eller varsel skal sendes ting er i ferd med å gå over tiden. Prosessen skal "svare på" forespørsler om status fra kunder og driftssjefer.
- **P5: Avslutning:** Driftssjef går gjennom jobber og arkiverer på skikkelig vis det som føles nødvendig. Dersom kunden skal faktureres, sendes faktura automatisk, prosessen sjekker etter innkomne betalinger og sender evt. purring.

b) Dekomponer aktiviteten for ”overvåking” eller oppfølging av jobber ett nivå ned.



Slik diagrammet i oppgave a) er konstruert, endrer/oppdaterer operatørene status på jobbene mens de holder på, dvs. at statusinformasjon fra operatør til systemet går til prosess P3 behandling.

c) For at en prosessbeskrivelse skal kunne automatiseres på denne måten, stilles det visse krav til modellen. Hva slags informasjon ”mangler” i vanlige DFD-diagrammer for at disse skal kunne kjøres i et arbeidsflytsystem ? Studer modellene fra oppgavene a) og b) og vis med eksempler hva som må legges til for at et arbeidsflytsystem skal kunne kjøre dem.

En besvarelse kan ta utgangspunkt i forskjellige aspekter (alternativt og likeverdig):

- I pensumartikkel P6 i pensum er det gitt en beskrivelse av arbeidsflytens 3 R'er (Ruting, Regler og Ressurser) og 3 P' er (Prosess, Politikk -"Policies" og Praksis). Man kan forklare de av disse punktene som er interessante ifm. "enactment"/kjøring og forklare hva som mangler i en DFD modell. Dette var oppgave på øving 5 (frivillig).
- Man kan ta utgangspunkt i et av de arbeidsflytspråk som er gitt som eksempel i pensum: Behaviour Network Model (BNM) eller Action Port Model (APM, artikkel P7). Man kan beskrive egenskapene til disse språkene og forklare hvordan de forbedrer DFD for det formål å kunne automatisere/kjøre modellen.
- Man kan helt konkret ta utgangspukt i de ekempeloppgavene fra CASE'et som er listet i oppgaveteksten, og forklare hva som mangler i en DFD modell for at en automatisering skal oppfylle disse oppgavene.

Eksempelbesvarelse basert på de 3 R'er og P'er:

- **Ruter:** Det kanskje viktigste aspektet ved en arbeidsflyt som kjøres er evnen til å automatisk å fordele oppgaver og tilhørende informasjon/data til aktører. En DFD modell inneholder i utgangspunktet informasjon om oppgaver og dataflyt. Akkurat hvilke dokumenter/data som skal rutes/sendes må imidlerti spesifiseres i detalj.
  For at ruting skal kunne skje automatisk, må vi også spesifisere tids-perspektivet i detalj. Når skal en flyt sendes, hva er tidsfristen for en oppgave, skal vi sende purringer, evt. sende informasjon på nytt dersom vi ikke hører noe etc. Tidsperspektivet kan markeres i vha. timere og markering av trigger/terminator. Også konkrete tidsfrister må legges inn.

- **Roller:** En DFD modell sier ikke noe om hvem som faktisk utfører de forskjellige oppgavene. Denne “Rolle”-informasjon finnes i oppgavebeskrivelsen (f.eks.: Kundekontakt, Operatør, Driftssjef)

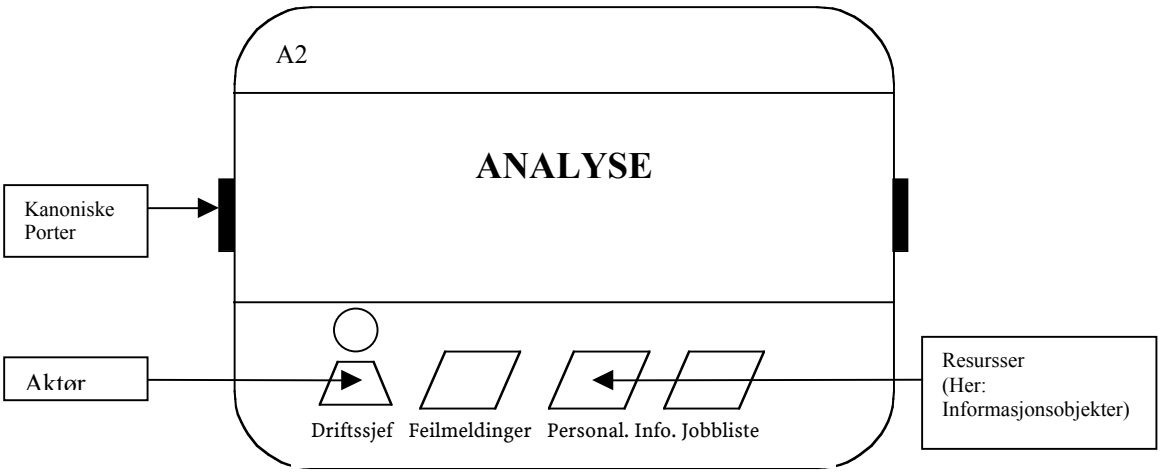
- **Regler:** I Marshak artikkelen (P6) er regler definert som knyttet til selve rutingen: "rules that determine what information is to be routed and to whom".
  En utvidelse av DFD i retning av de logiske portene i PrM som angir sammenhengen mellom de forskjellige flytene. Portene forteller om hvilke flyter som skal sendes, hvilke som må sendes sammen (AND), eller hvilke som er alternative (XOR) En PrM modell alene sier imidlertid ikke noe om de konkrete reglene/betingelsene som avgjør *om* en flyt skal sendes eller evt. *hvor* den skal sendes hen. *Hvor* den skal sendes, ligger i angivelsen av roller som nevnt tidligere. *Om* den skal sendes, må angis, f.eks. med pre-post betingelser for at en oppgave kan starte.
  I strukturert analyse fra tidligere (boka kap. 2), har vi modellert regelperspektivet ved hjelp av egne formalismer (beslutningstrær, beslutningstabeller osv.) En kopling av DFD til en regelmodell, slik det f.eks. er tenkt i PPP med kopling til PlD, vil dekke dette perspektivet.
- **Politikk:** Bestemmer hvordan en oppgave skal utføres eller definerer hva (og hvordan) prosessen skal utføres i detalj. Mange av oppgavene som utføres i en organisasjon vil være underlagt både interne og eksterne lover og regler, eller kriteria for hva som er korrekt utførelse av prosessen.
  Hvor mye "politikk" som er nødvendig for automatisereingen av en oppgave er vanskelig å si generelt, men de reglene som avgjør for at en oppgave er riktig utført, må kodes på hensiktsmessig måte, slik for at oppgaven i det hele tatt skal kunne automatiseres. Som nevnt gjøre mangjerne dette i egne språk, pseudokode, rgelmodell, …

De to siste P'ene er ikke direkte knyttet til evnen til å automatisere/kjøre prosessen. De er her tatt med bare for å komplettere beskrivelsen av de 3 R'er og P'er, men er ikke nødvendige i en besvarelse. Det vil imidlertid ikke nødvendigvis være galt å kommentere også dette, f.eks. ifm. praksis at en "kjørt" arbeidsflyt kan lagres i systemet i ettertid.

- **Prosess:** En DFD-modell representerer en beskrivelse av prosessen som skal kjøres. En arbeidsflytomgivelse er mye mer enn bare selve prosessmodellen!

- **Praksis:** Hvordan oppgaver faktisk utføres, erfaringer, beslutninger og rutiner som bare "er sånn". Prosessmodeller representerer naturlig nok ofte et idealisert eller forenklet bilde av prosessen. I virkeligheten vil mye prosessinformasjon og rutiner eller "regler" finnes i det vi kan kalle "etablert praksis". Det er ofte veldig vanskelig å fange inn slik praksis og formalisere denne i en prosessmodell- Ofte kan det være mer hensikstsmessig med intervjuer, erfarignsrapporter fra aktører, FAQ's eller transcript av diskusjoner eller samtaler for å forstå hvordan arbeidet utføres.
  I arbeidsflytsystemer er det ofte ønskelig å ta vare på prosessinformasjon fra utførte/"kjørte" prosesser, for å kunne se hva som faktisk skjedde.

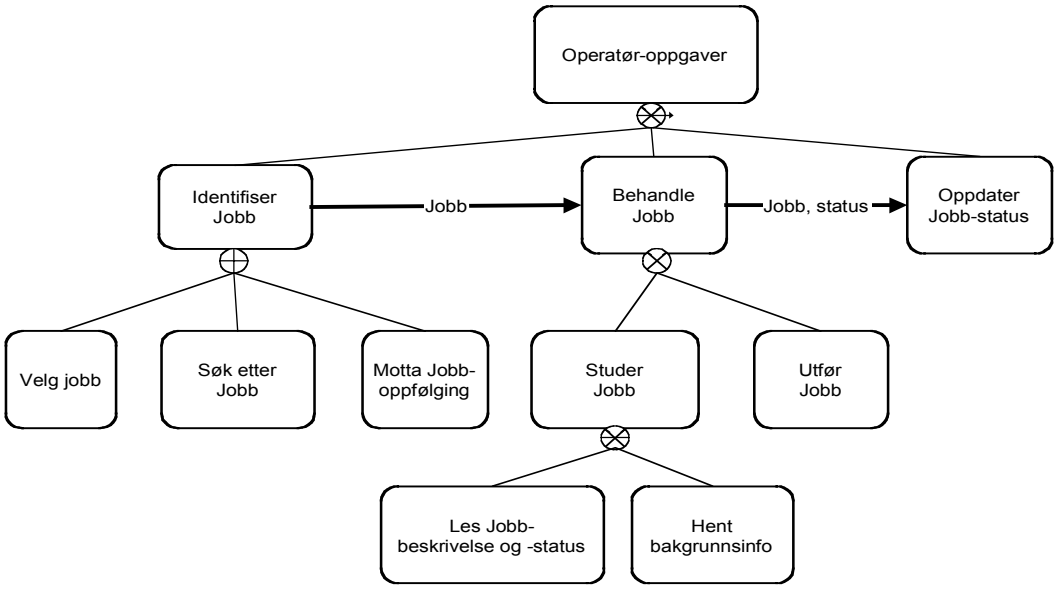
Figuren viser en eksempel APM modell som utvider DFD modeller med porter og ressurser. Man bygger altså ut med informasjon ihht. de nevnte områder; Roller, Regler/Ruting (portene), Ressurser. Man må fremdeles konkretisere denne informasjonen dersom prosessen skal kunne kjøres, f.eks. angi hvilken driftssjef som skal ha jobben (evt. sendes til den første og beste som er på jobb), man må angi faktiske tidspunkt, forsinkelser, etc.



Oppgave 3 – Brukergrensesnittmodellering (20%)

a) ***Oppgavemodeller*** brukes for å beskrive hvilke mål en bruker ønsker å nå, og hvordan hun strukturerer arbeidet for å nå målene. En skiller mellom to typer bruk av oppgavemodeller: 1) En beskriver hvordan dagens verktøy (manuelle eller IT-baserte) brukes for å nå mål og utføre oppgaver og 2) en beskriver hvordan en fremtidig/tenkt IT-løsning skal brukes for å nå/utføre de samme eller nye mål/oppgaver. Hensikten er uansett å fokusere på hva en bruker gjør/ønsker å gjøre og hvorfor, ikke hvordan IT-systemet virker.
Dialogmodeller brukes for å beskrive strukturen til et brukergrensesnitt, hvordan elementer i brukergrensesnittet sekvensieres og informasjonsflyt i et brukergrensesnitt. Dialogmodeller er designorientert, siden de beskriver hvordan faktiske konstruksjonselementer støtter oppgavene brukeren skal utføre.
Presentasjonsmodellen beskriver forholdet mellom elementer i dialogmodellen og synlige elementer i brukergrensesnittet og deres look-&-feel, f.eks. vinduer, layout og knapper.

b) Oppgavemodellen er vist i figuren under. Som vi ser er det flere koblinger mellom oppgavemodellen og de andre modellene:
Referentmodellen: Oppgavemodellen refererer til konseptene, både i oppgavenavnene og dataflytene. Dette omfatter informasjon som brukeren skal se, vurdere, reagere på og forandre. I vår modell er dette i første rekke Jobb, men implisitt ligger også forholdet mellom jobben og den som utfører den, altså utfører-relasjonen. Status-begrepet tilsvarer de disjunkte jobb-subsettene. Det nyttige med oppgavemodellen er at den viser hvordan konseptene og i hvilken kontekst de brukes, f.eks. hvilken informasjon som brukes sammen, hvordan en navigerer i informasjonen osv.
Prosessmodellen: En oppgavemodell tar typisk utgangspunkt i en aktør i prosessmodellen, dvs. aktøren er brukeren som oppgavemodellen gjelder. Prosessene som hun deltar i vil en finne igjen i oppgavemodellen, f.eks. Behandling. Dataflyt mellom aktøren og prosessene vil en ofte finne igjen i form av spesifikke kommunikasjonsoppgaver, f.eks. Motta Jobboppfølging. Både oppgavemodellen og dataflytdiagrammet er funksjonsorienterte. Det viktigste skillet mellom disse modelltypene er at oppgavemodellen er brukerorientert (en samler oppgavene til en brukertype) og presenterer oppgavene slik de (typisk) utføres. Prosessmodellen er systemorientert og angir f.eks. ikke om en konkret person kan tilsvare flere aktører/kan inneha flere av rollene i prosessmodellen.



Notasjon:

dataflyt

⊕ Disjunkt: Bare en av deloppgavene utføres

⊗ Aggregering: En vilkårlig utførelse av deloppgavene

⊗ Alternering: En deloppgave utføres om gangen

⊗→ Sekvens: Alle deloppgavene utføres i bundet sekvens

NB! Om syntaks av oppgavemodeller:
Den notasjon som er brukt her i LF'et er ikke forelest. Forlesningen har lagt vekt på at studentene skal kunne sette opp de riktige oppgaver og bygge opp et fornuftig hierarki. Videre skal de på en eller annen måte (gjerne med tekst) angi om samtlige underoppgaver må utføres, om bare noen av dem utføres (frivillig), bare den ene utføres (disjunkt), samt også i hvilken rekkefølge de utføres. Syntaksen over er en videreutvikling av språket som studentene bare har sett i LF'et til øving 4 (obligatorisk) og kan ikke forventes.

Oppgave 4 – Iverksettelse (20%)

a) Valget mellom å kjøpe (iverksettelse vha. Rammeverk/halvfabrikata) eller lage (tradisjonell skreddersøm) gjøres utfra følgende punkter:

- *Frekvens/viktighet:*Hvor ofte utføres prosessen(e) som systemet skal støtte ? Hvor kritiske er disse? Hvor mye koster det å utføre prosessen(e)?
- *Strategi:* Hvor strategisk viktig er prosessen(e)? Er det viktig for bedriften å skaffe seg et spesialisert system som kan gi dem konkurransemessige fortrinn på dette området?
- *Organisasjon:* Hvor dyrt er det å endre organisasjonsformer og rutiner til å passe med det nye systemet? Hvor viktig er at systemproteføljen til en organisasjon er sydd over samme lest (har common look-and feel) og støtter samarbeid på tvers av organisasjonsenhet, gruppering og plassering?
- *IT strategi generelt:* Passer systemet inn med de andre systemene man har? Aspekter vil være helhet, system-samarbeid, informasjonsutveksling mellom systemer… Går man for standardløsninger og har satset på en enhetlig systemprofil?
- *Drift, vedlikehold:* Det er lettere å vedlikeholde en portefølje av standardsystemer enn masse skreddersydde, enkeltstående løsninger. Hvor enkelt er det å modifisere systemet over tid, ettersom organisasjonen og "verden ellers" forandrer seg og skaper nye behov?
- *Avtaler:* Innkjøpsavtaler, vedlikeholdsavtaler.
- *Kvalitet:* Gjenbruk gir økt kvalitet, hyllevare/innkjøpte systemer er gjerne testet av flere.

b) Den viktigste forskjellen er at man ved bruk av rammeverk må forholde seg til et system, en arkitektur og en "tankegang"/filosofi som allerede eksisterer, og finne ut hvordan dette vil passe inn med organisasjonens ønsker og behov. Man må drive såkalt "Fit Analysis". Man får et større fokus på konfigurering/tilpasning som en sentral og egen aktivitet i implementasjonsfasen. Både system, mennesker, organisasjon og prosesser må tilpasse seg for å få til en vellykket iverksettelse av det nye systemet. Spørsmålet er hva som må tilpasses mest; Systemet eller organisasjonen? Det er ikke alltid gitt hva som er det billigste.

Fra SAP's side forutsettes det at man legger opp til en omorganiseringsprosess sammen med innføringen. Ved å ta utgangspunkt i gitte referansemodeller - "Business Blueprints" og gjennomføre omorganisering i forhold til disse, vil man sikre seg at det er mulig å implementere systemet innenfor rammene av den funksjonalitet SAP tilbyr. Dette krever naturlig en omlegging av prosesser og organisasjon.

De kompliserende faktorer vedrørende SAP kan kort sies å være følgende:

- Systemet er (veldig) stort og komplekst i seg selv.
- Det dekker mange sentrale og gjennomgripende funksjoner. Siden det finnes så mange "ferdige muligheter", har SAP utviklingsprosjekter også en tendes til å vokse underveis, til å omfatte stadig mer funksjonalitet og flere områder.
- Man ønsker gjerne å innføre SAP i en "global" setting, dvs. i flere organisasjonsenheter, på tvers av landegrenser og kulturer, for å støtte arbeids/forretnings-prosessene gjennomgripende.

Disse aspektene gjør tilpasningen svært vanskelig, kompleks og kostbar. Systemet er stort og komplekst å forandre på i seg selv. Ennå vanskeligere er det å forandre en hel organisasjon på tvers av org.enheter og kulturer. Dette stiller store krav til kommunikasjon, samarbeid og vilje/mulighet til tilpasning og integrasjon. Prosjekter av denne type krever solid forankring i alle ledd av organisasjonen, og må følges opp med informasjon, opplæring og trening.