FFI RAPPORT

UTVIKLING AV MENNESKE-MASKIN-SYSTEMER

BRÅTHEN Karsten, NORDØ Erik, JENSSEN Arne Cato

FFI/RAPPORT-2001/04234

FFIE/730/134

Godkjent
Kjeller 3 januar 2002

Vidar S Andersen
Forskningssjef

UTVIKLING AV MENNESKE-MASKIN-SYSTEMER

BRÅTHEN Karsten, NORDØ Erik, JENSSEN Arne Cato

FFI/RAPPORT-2001/04234

FORSVARETS FORSKNINGSINSTITUTT
Norwegian Defence Research Establishment
Postboks 25, 2027 Kjeller, Norge

FORSVARETS FORSKNINGSINSTITUTT (FFI)
Norwegian Defence Research Establishment

P O BOX 25
NO-2027 KJELLER, NORWAY

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE
(when data entered)

REPORT DOCUMENTATION PAGE

1) PUBL/REPORT NUMBER	2) SECURITY CLASSIFICATION	3) NUMBER OF PAGES
FFI/RAPPORT-2001/04234	UNCLASSIFIED	177
1a) PROJECT REFERENCE	**2a) DECLASSIFICATION/DOWNGRADING SCHEDULE**	
FFIE/730/134	-	

4) TITLE

UTVIKLING AV MENNESKE-MASKIN-SYSTEMER

(HUMAN-MACHINE SYSTEMS ENGINEERING)

5) NAMES OF AUTHOR(S) IN FULL (surname first)

BRÅTHEN Karsten, NORDØ Erik, JENSSEN Arne Cato

6) DISTRIBUTION STATEMENT

Approved for public release. Distribution unlimited. (Offentlig tilgjengelig)

7) INDEXING TERMS

IN ENGLISH:	IN NORWEGIAN:
a) Man-Machine Systems	a) Menneske-maskin-systemer
b) Systems Engineering	b) Systemteknikk
c) User Modelling	c) Operatørmodellering
d) User Interfaces	d) Brukergrensesnitt
e)	e)

THESAURUS REFERENCE: **INSPEC**

8) ABSTRACT

The report gives an introduction to the design of complex human-machine systems (HMS). First, definition and characterization of human-machine systems are given. Then, a process model for the development of HMSs is outlined which consists of an analytical and experimental part. Phases and activities of these parts are elaborated throughout the report. Additionally, performance models of human skill-, rule- and knowledge-based behaviour in socio-technical systems are discussed. The report is written as a compendium for an M Sc course in Human-Machine Systems Engineering at the Center for Technology at Kjeller.

9) DATE	AUTHORIZED BY	POSITION
	This page only	
3 January 2002	Vidar S Andersen	Director of Research

ISBN-82-464-0614-0

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE
(when data entered)

FORORD

Vår interesse for fagområdet menneske-maskin-systemer (MMS) ble vakt gjennom arbeider med regulerings- og estimeringstekniske problemstillinger i forbindelse med våpensystemer ombord på marinefartøyer. Det ble raskt klart for oss at utvikling av beregningsalgoritmer i datamaskinen bare var en liten del av jobben og at samspillet mellom disse og operatørene av systemet var sentralt for å oppnå en tilfredsstillende ytelse av det samlede systemet. Typiske spørsmål som dukket opp var hvordan vi skulle kunne kombinere den kunnskap som operatørene hadde med beregningsalgoritmene og hvordan vi skulle presentere resultater av f eks estimeringsalgoritmer på en form som var lett forståelig for en operatør uten annen bakgrunn i statistikk enn den intuitive forståelsen som folk flest har. Dette var på midten av 80-tallet og den raske utviklingen av datateknologi gav oss nye muligheter til å angripe disse problemstillingene.

Mens disse problemene i første rekke er knyttet til operatørgrensesnittet som sådan, ble det også klart for oss at viktigere enn utformingen av dette er de systemtekniske sidene, spesielt for større systemer hvor det inngår flere operatører og et bredt spekter av oppgaver skal dekkes. Slike systemtekniske problemstillinger er bl a å sørge for at operatørgrensesnittet er tilpasset de oppgaver som operatøren er tildelt, avgjøre hvordan fordelingen av oppgaver mellom forskjellige operatører bør være og i hvilken grad oppgaver skal automatiseres og om det er behov for å gi operatørene støtte for at tildelte oppgaver skal kunne utføres på en tilfredsstillende måte. I tillegg må disse avklaringene gjøres som en integrert del av de øvrige systemtekniske aktivitetene ved utviklingen av et større system.

I de siste årene har det kommet en strøm av bøker som omhandler brukergrensesnitt for datasystemer. Det som i stor grad kjennetegner disse bøkene er at de omtaler den type grensesnitt som er mest vanlig når boka ble skrevet (i dag vil det si grafiske bruker-grensesnitt) og at brukergrensesnittet ikke blir omtalt som en del av et system. Ved at disse bøkene er så sterkt knyttet opp til den teknologien som er tilgjengelig når boka blir skrevet mister de raskt sin aktualitet i likhet med størsteparten av bøker som skrives om datateknologi. Ut fra den vinklingen som er skissert ovenfor dekker denne type bøker ikke våre behov. Heller ikke bøker skrevet om systemergonomi dekker det vi ønsker fordi de ikke i tilstrekkelig grad trekker inn system- og programvaretekniske problemstillinger som er nødvendig for å komme frem til et fremtidig mål om integrerte utviklingsmetoder for menneske-maskin-systemer. Pga disse forholdene har vi funnet det nødvendig å skrive dette kompendiet.

Kompendiet gir en innføring i menneske-maskin-systemer og forsøker å dekke de viktigste områdene, men med hovedvekt på systemtekniske sider. Selv om teknologi i en viss grad berøres, har vi valgt å legge vekt på forhold som er av allmenngyldig karakter og som ikke er uaktuelt når ny teknologi blir introdusert. Kompendiet er derfor behovsorientert mer enn teknologiorientert. Kompendiet berører også i mindre grad psykologiske problemstillinger, delvis pga vår egen faglige bakgrunn, men også pga at vi mener de i alt vesentlige empiriske arbeidsmetodene innen psykologi er mindre egnet for utvikling av store og komplekse systemer hvor en meget stort antall konstruksjonsavgjørelser kunne tenkes vært underlagt empirisk verifikasjon. Kompendiet

omhandler modellering av menneske-maskin-systemer da vi anser modellering som helt essensielt ved utvikling av alle typer av systemer av en viss kompleksitet. En utfordring innenfor mennneske-maskin-systemer er å fremskaffe egnede modeller av den menneskelige systemkomponenten og kompendiet beskriver grunnlaget for slike modeller, deres muligheter og begrensninger. En annen utfordring er å få til en systemutvikling som integrerer de ulike disiplinene til et integrert hele. Vi forfekter en operatørorientert utviklingsmetode for menneske-maskin-systemer som består av en analytisk og en eksperimentell del. Kompendiet omtaler de forskjellige stegene i en slik metode grundig, så som funksjons- og oppgaveanalyse og konstruksjon av operatørgrensesnittet. Kompendiet vil forhåpentligvis gi et tilstrekkelig grunnlag for videre fordypning og være en rettleder når man står ovenfor en konkret utvikling.

Kompendiet blir benyttet i hovedfagskurset UNIKI 356 "Utvikling av menneske-maskin-systemer" ved Universitetsstudiene på Kjeller, Universitetet i Oslo. Kurset har vært holdt siden 1991 og stoffet har blitt noe tilpasset underveis, men hovedtrekkene fra starten ligger fast. Vi vil med dette takke studentene som har tatt kurset for de tilbake-meldingene vi har fått. Likeledes vil vi takke UNIK for at de ga oss muligheten for å skrive grunnlaget for dette kompendiet høsten 1994 og Forsvarets forskningsinstitutt for at de gir oss mulighet for samarbeidet med UNIK og for at Instituttet har gitt oss utfordrende oppgaver som har krevd av vi har måttet fordype oss innenfor dette fagfeltet. Og endelig gir vi en unnskyldning til våre familier for kvelder og weekender som har "gått til spille" på den familiære arena pga arbeidet med kurset og kompendiet.

INNHOLDET AV KOMPENDIET

Kompendiet består av en introduksjon, en oppsummering og seks kapitler hvor ett omhandler modellering, mens de fem andre beskriver ulike faser i utvikling av menneske-maskin-systemer.

Introduksjonen definerer hva vi mener med et menneske-maskin-system og hva som skiller det fra tradisjonelle interaktive systemer. Det gis eksempler på MMS-er og hvilke likheter og forskjeller det er på MMS-er innenfor ulike anvendelsesområder. Videre argumenterer vi for vårt ståsted og vår tilnærming til MMS. Til slutt gir vi en kort oversikt over forskjellige fagdisipliners bidrag til fagområdet.

I kapittel 2 introduseres en systemarbeidsmodell som er egnet for utvikling av MMS og som store deler av kompendiet er bygget opp rundt. Begrepet systemarbeidsmodell blir definert og vi gjennomgår tradisjonelle systemarbeidsmodeller og problemer man møter hvis disse blir benyttet ukritisk ved utvikling av MMS. Deretter foreslås en operatør-sentrert systemarbeidsmodell for MMS som består av en analytisk og en eksperimentell del. Til slutt blir de viktigste aktivitetene i de ulike utviklingsfasene gjennomgått. Kapittel fire til og med syv omtaler systemarbeidsmodellens faser og deres aktiviteter i mer detalj.

Før vi går inn på de enkelte fasene, er det et eget kapittel om modellering, kapittel 3. Modellering og simulering er helt sentralt ved utvikling av et hvert system av en viss kompleksitet, så vi legger forholdsvis stor vekt på dette området. Etter å ha definert hva vi mener med en modell og hvorfor modeller utvikles, gir vi en svært enkel innføring i systemteori som en basis for det som kommer senere i kompendiet. Hovedvekten av kapitlet omhandler modeller av operatører i tekniske systemer. For dette benytter vi et rammeverk som inndeler operatøradferd i en arbeidssituasjon inn i tre typer; ferdighets-, prosedyre- og kunnskapsbasert adferd med tilhørende klassifisering av informasjonen som utveksles mellom operatør og de tekniske delene av systemet som signaler, tegn og symboler. Med dette rammeverket som bakgrunn, introduserer vi maskinanalogier til operatøradferd innenfor de ulike typene av adferd. For ferdighetsbasert adferd beskrives operatørmodeller for manuell regulering og monitorering. For prosedyrebaserte oppgaver som blir utført av en operatør, er det en rekke forskjellige modellrepresentasjoner som kan benyttes og som beskriver forskjellige sider av adferden. De viktigste av disse blir kort beskrevet. Om det er mulig å modellere kunnskapsbasert operatøradferd er et aktuelt diskusjonstema innen forskning. Denne typen modellering blir i liten grad berørt. En konkret operatørjobb vil være en blanding av oppgaver som krever alle tre typer adferd og vi avslutter modelleringskapitlet med å skissere egenskaper til modeller av sammensatte operatøroppgaver og modeller som dekker hele MMS-et. Realistiske modeller lar seg sjelden løse analytisk, så kapitlet avsluttes med en kort omtale av simulering av MMS, med eksempler på programmeringsspråk som er utviklet spesielt for å simulere bemannede systemer.

Kapittel 4 omhandler de tre første fasene i systemarbeidsmodellen, nemlig etablering av krav, funksjonsanalyse og analyse av operatøroppgaver. Den første fasen, etablering av krav, består av fem hovedaktiviteter. Disse aktivitetene omfatter definisjon av målsetning og overordnede krav, analyser av scenarier (beskrivelse av typiske hendelsessekvenser som MMS-et må kunne håndtere), operatørpopulasjon og eventuelt eksisterende systemer (det finnes nesten alltid et eksisterende system!), samt å etablere en overordnet

systemstruktur. Kapitlet beskriver kort metoder og teknikker for de ulike aktivitetene og hvilke spørsmål som vi ønsker å svare på ved å gjennomføre dem.

Den neste fasen, funksjonsanalyse, inklusiv allokering, beskriver hvordan vi modellerer det totale MMS-ets funksjoner/oppførsel som skal til for å nå målsetningen og de overordnede krav til systemet. Det gis et innblikk i ulike modelleringsteknikker som kan benyttes. Disse er i hovedsak hentet fra system- og programvareteknikk. Funksjons-allokering som tildeler funksjoner til operatør og maskin med utgangspunkt i funksjons-modellen beskrives, samt at vi diskuterer hovedprinsipper for utforming av operatør-støttesystemer og generelle problemer knyttet til automatisering. Funksjonsallokering blir ansett som et svært viktig steg under utvikling av MMS, men det er dessverre ingen akseptert metode for hvordan allokeringen bør gjennomføres.

Kapittel fire avsluttes med å beskrive operatøroppgaveanalyse, dvs analyse av de (delene av) systemfunksjoner som er tildelt operatøren eller hvor operatøren spiller en vesentlig rolle sammen med maskin for å utføre funksjonen. Forskjellige beskrivelsesrepresenta-sjoner gjennomgås og analyse av kognitive oppgaver blir også berørt.

Mens kapittel 4 omhandler fasene som ligger forut for utvikling av selve operatørgrense-snittet, beskriver kapittel 5 spesifikasjon og konstruksjon av operatørgrensesnitt og kapittel 6 egenskaper ved de ulike delene som benyttes for å realisere et operatørgrense-snitt. Kapittel 5 omtaler først noen viktige prinsipper som må følges for å oppnå gode operatørgrensesnitt for deretter å gjennomgå en veletablert konstruksjonsmetode som baserer seg på et kommunikasjonsorientert syn på grensesnittet.

Kapittel 6 omtaler svært kort de forskjellige teknologiske generasjoner av operatør-grensesnitt og egenskaper ved vanlig benyttede dialogtyper som direkte manipulerende og objektorienterte grensesnitt, samt kunstig virkelighetsgrensesnitt. Vi diskuterer også kort egenskaper til betjeningsorganer og informasjonsgivere som er vanlige i dag. Kapitlet avsluttes ved å berøre programvare og verktøy for å realisere moderne operatørgrensesnitt.

Den eksperimentelle delen av systemarbeidsmodellen omtales i kapittel 7 med vekt på målemetoder benyttet ved gjennomføring av forsøk med prototyper. Statistisk analyse av måledata fra forsøk gjennomgås også kort med noen kommentarer av "statistisk signifikans" som er sentralt når alternativer sammenlignes opp mot hverandre. Kompendiet avsluttes med en oppsummering av de foregående kapitlene og noen tanker om fremtiden til dette fagområdet.

INNHOLD

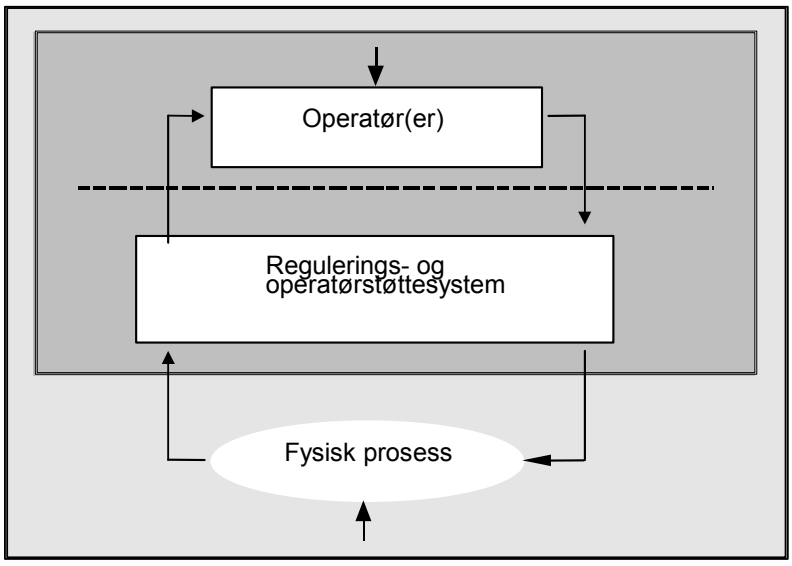
UTVIKLING AV MENNESKE-MASKIN-SYSTEMER

1 INTRODUKSJON

Nye systemer skaper nye problemer.
Komplekse systemer skaper nye problemer straks de er skapt.
Komplekse systemer gir kompliserte svar - ikke løsninger.
Mennesker i et system gjør aldri det systemet sier de skal.
Gudmund Hernes*: Hvorfor mer går galt* (1981)

Den type *menneske-maskin-systemer* (MMS) som dette kompendiet omhandler er illustrert i en enkel skisse i figur 1.1. En lang rekke reguleringssystemer vil ha en slik oppbygging og operatørenes rolle varierer avhengig av hvilken automatiseringsgrad som er benyttet. Kompendiet vil fokusere på operatørenes rolle i slike halv-automatiske systemer, hvordan operatørene samspiller med det automatiserte reguleringssystemet og hvordan støtte kan utformes for å avhjelpe operatørene i deres planleggings-, analyse- og beslutningsoppgaver.

I figur 1.1 er operatørene og regulerings- og operatørstøttesystemet omsluttet for å understreke at operatørene betraktes som en *systemkomponent* på linje med det tekniske utstyret. Som de tekniske delene, må operatørene «konstrueres». Eller mer presist, operatørenes *jobb* må syntetiseres. *Operatørgrensesnittet* er representert med den stiplede linja.



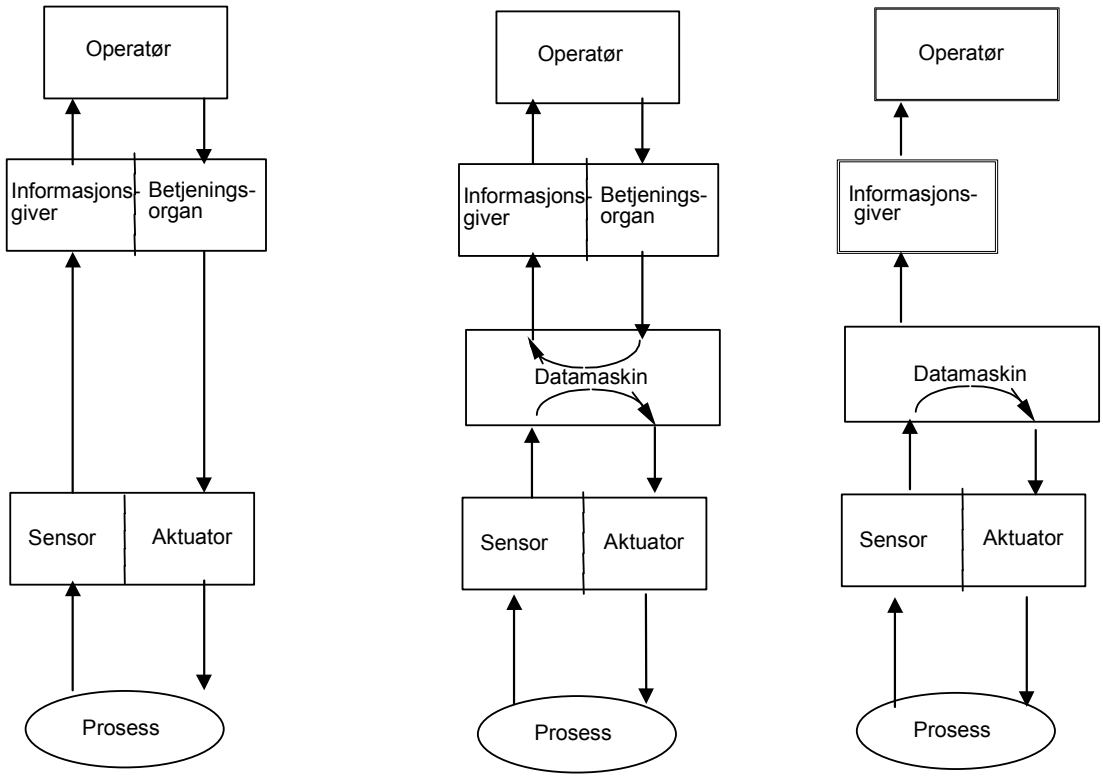
Figur 1-1 Menneske-maskin-system (MMS)

Brukergrensesnitt (eng: user interface, human-computer interface) er blitt et svært aktuelt tema

innen informatikk. Det er flere trekk som skiller den type systemer vi tar for oss her fra den fokus som tradisjonell informatikk har på brukergrensesnitt:

- Operatørene er i en lukket sløyfe med en *fysisk prosess*. Operatørene er den ytterste reguleringssløyfa. Selv om det benyttes høy grad av automatisering, må man unngå å dekoble operatøren for mye fra prosessen.
- Datamaskiner blir benyttet som mellomliggende lag mellom operatørene og prosess. Dette laget bør til en viss grad være *transparent* og hjelpe operatøren å få oversikt over tilstanden til den fysiske prosessen.
- *Dynamisk system*. Dynamikken er bestemt av den fysiske prosessen. Det er et sanntidssystem. Operatørene må f eks ikke bare ta en riktig beslutning, men beslutningen må gjøres til rett tid.
- Det er gjerne ikke bare én operatør, men flere som arbeider sammen i en *gruppe*.
- For å kunne betjene et slikt system må operatørene både forstå og ha kunnskap om reguleringssystemet og den fysiske prosessen.
- Systemet opereres over et *lengre tidsintervall* som kan strekke seg fra timer til uker.

For å skille de som betjener denne type systemer fra «brukere» av datamaskiner (f eks typiske kontorautomatiseringssystemer), har vi valgt å benytte betegnelsen *operatør* og ikke *bruker*. For operatørene er dette en jobb, et yrke, en profesjon. Brukere av datamaskiner må selvsagt også ha kunnskap om oppgavene de ønsker å løse, men kompleksitet og spesialisering er typisk en helt annen enn det vi vanligvis finner for typiske kontoroppgaver.



Figur 1-2 Manuell regulering, overvåkende og automatisk regulering

I figur 1.2 er bemannede reguleringssystemers plass mellom ren *manuell regulering* og *automatisk regulering* vist i litt større detalj. Vår fokus er på de delvis automatiserte systemene, men vi vil også behandle manuell regulering. I delvis automatiserte systemer har operatørene oppsyn med og overvåker reguleringssystemet og prosessen, og griper bare inn hvis nødvendig (eng. supervisory control, Sheridan (1987)).

Vi får flere og flere MMS av den typen vi har skissert ovenfor og samfunnet gjør seg mer og mer avhengig av dem. Det også en rekke eksempler på hvor store konsekvenser det kan ha hvis de på noen måter feiler. To eksempler er ulykken ved Three Mile Island atomreaktor i 1979 og nedskytingen av et Iransk passasjerfly fra et amerikansk marinefartøy i 1987.

Noen eksempler på MMS er:

1. Industrielle prosessreguleringssystemer
2. Automatiseringssystemer i stykkproduserende industri
3. Sivile kontroll- og overvåkningssystemer
4. Militære kampsystemer og kommando- og kontrollsystemer
5. Fartøysstyring

Disse MMS-ene har mange fellestrekk, men er også forskjellige ut fra egenskapene til den fysiske prosessen som styres og overvåkes, og i hvilken grad styringen av den er automatisert (og dermed operatørenes rolle i systemet). Noen faktorer som er med på å karakterisere MMS er:

- Størrelsen på *tilstandsrommet*. Antall tilstandsvariable som beskriver f eks industrielle prosessreguleringssystemer er stort. Dette fører blant annet til at det blir en vanskelig avveining mellom å ha oversikt og å ha tilstrekkelig detaljkunnskap. Dette løses gjerne ved at man har et hierarki av prosessbilder hvor hvert nivå gir et visst detaljeringsnivå. Siden det kan være svært mange bilder (gjerne flere hundre) oppstår det et navigasjonsproblem for operatøren.
- *Tidskonstanter* for en prosess er parametere som karakteriserer typiske reaksjonstider i systemet. Mens fartøysstyring kan ha tidskonstanter i sekunders området har f eks prosesskontroll og kommando- og kontrollsystemer tidskonstanter i størrelsesområdet minutter og timer.
- *Antall målinger*. Industrielle prosessreguleringssystemer kan ha flere tusen målepunkter.
- Antall reguleringssløyfer. Industrielle prosessreguleringssystemer kan ha flere hundre reguleringssløyfer på ulike nivåer.
- *Antall objekter*. I sivile og militære overvåkningssystemer ønsker man å overvåke mange (opptil flere tusen) objekter med ensartet oppførsel. Tilstandsrommet til hvert objekt kan være lite (antall tilstandsvariable under ti), men antallet objekter gjør at det er vanskelig å få oversikt.

- *Observerbarhet*. For militære systemer har man typisk et observerbarhetsproblem. Datamengden fra målingene kan være stor, men informasjonsmengden kan være liten. Man har lav observerbarhet av tilstandene. Årsaken kan være at man benytter målemetoder hvor man forsøker å unngå å røpe sin egen tilstedeværelse eller at motparten vanskeliggjør eller forstyrrer måleprosessen. Mer generelt har vi et observerbarhetsproblem i alle systemer hvor uobserverbare menneskelige kognitive prosesser spiller en stor rolle for operasjonen og ytelsen til systemet.
- *Måleusikkerhet*. Som ved alle andre typer systemer vil målinger i MMSer være beheftet med usikkerhet. Dette har betydning både for operative forhold (hvor godt operatører kjenner systemtilstanden) og for evalueringen av et system.
- *Menneskelig påvirkning*. Mens industrielle automatiseringssystemer styrer og overvåker prosesser som følger fysiske lover, kan oppførselen til de prosessene som overvåkes i tillegg til fysiske lover være styrt av menneskelig inngripen. Ikke minst i militære systemer gjør dette det vanskeligere å forutsi forløpet til prosessen.

Tradisjonelt fokuseres det i for stor grad på operatørgrensesnittet som sådan. Dvs hvordan operatøren skal betjene systemet vha knapper, menyer, osv og hvordan vi skal presentere data til operatøren, f eks hvilke farge og hvilken form som skal benyttes. Dette er vel og bra, men det hjelper lite om man har et godt operatørgrensesnitt hvis operatøren ikke kan løse sine *oppgaver* på en tilfredsstillende måte vha det. Etter hvert som systemene vokser i kompleksitet vil det være en større utfordring å utvikle systemer med rett funksjonalitet tilpasset operatøren enn det vil være å utforme selve operatørgrensesnittet. Utfordringene vil ligge i å avklare hvilke funksjoner som systemet skal inneholde, hvilken automatiseringsgrad som bør benyttes, hvordan oppgaver bør fordeles mellom operatører og hvilken operatørstøtte som bør gis. Vi sier ikke at utforming av operatørgrensesnittet ikke er viktig, men det er nødvendig å gjennomføre en rekke aktiviteter *før* man går i gang med konstruksjon av grensesnittet.

1.	**Funksjonsanalyse** *Hvilke funksjoner og oppgaver skal systemet løse?* *Hvordan skal systemet struktureres og hvor mange operatører er det behov for?*
2.	**Funksjonsallokering. Automatiseringsnivå/filosofi** *Hvordan skal funksjoner fordeles mellom operatør(er) og maskin?* *Hvilken rolle skal operatøren(e) ha?*
3.	**Oppgaveanalyse. Operatørstøtte** *Hvilken og hva slags operatørstøtte må systemet inneha for at operatøren(e) skal kunne utføre tildelte oppgaver på en tilfredsstillende måte?*
4.	**Konstruksjon av menneske-maskin-kommunikasjon (MMK)/Operatørgrensesnitt** *Hvilken informasjon og når skal informasjon presenteres for operatøren(e)?* *Hvilke kommandoer skal være tilgjengelig for operatøren(e) til hvilken tid?* *Hvordan skal informasjon presenteres?* *Hvordan skal operatøren(e) gi kommandoer?*
5.	**Fysisk konstruksjon av arbeidsplass** *Hvilke informasjonsgivere og betjeningsorganer skal benyttes?* *Hvordan skal informasjonsgivere og betjeningsorganer utformes?* *Hvordan skal arbeidsplass og miljø utformes?*

Tabell 1.1 Problemstillinger ved utvikling av MMS

Det er behov for en dreining fra *hvordan* mot *hva*. De spørsmål som må avklares og den logiske rekkefølgen, og dermed under hvilken fase av systemutviklingen de må besvares, er vist i tabell 1.1.

Litt ulikt bøker innen feltet vil vi konsentrere oss om de tre første punktene i tabellen ovenfor. Punkt 4, konstruksjon av menneske-maskin-kommunikasjon (MMK), vil også i en viss grad bli dekket, men da igjen med hovedvekt på «hva»-delen. Fysisk konstruksjon av arbeidsplass vil ikke bli berørt.

Tradisjonelt har man i liten grad inkludert aktiviteter som omhandler bruksaspekter i de tidlige fasene av en systemutvikling. Dette gjelder også senere under utviklingen. Det som gjerne har blitt gjort er å gjøre brukerforsøk *etter* at utstyret/systemet er mer eller mindre ferdig utviklet. Årsakene til dette kan være flere, men ett moment er den sterke påvirkningen fra psykologi og deres tradisjonelt empiriske arbeidsmetoder. Mulighetene for å påvirke utforming av utstyr blir på denne måten marginal og en slik arbeidsmetodikk er inadekvat for utvikling av komplekse MMS hvor et meget stort antall problemstillinger kunne vært underlagt empirisk verifikasjon. Det er behov for en mer proaktiv holdning og operatøraspekter må stå i sentrum av utviklingen fra første dag. For å kunne få til en slik dreining mener vi at det ikke bare er behov for å bringe inn tradisjonell kompetanse om menneskelige faktorer tidlig inn i et prosjekt. Vel så viktig er det å bevisstgjøre system- og programvare-ingeniører slik at de er bedre i stand til å gjennomføre nødvendige analyser. Bare på denne måten kan de nødvendige analysene bli en naturlig del og integrert sammen med andre systemanalyser. For å besvare hovedspørsmålene (f eks antall operatører, arbeidsfordeling, informasjonsutveksling, osv.) vil systemtekniske metoder være mer egnet enn empiriske metoder. Det vil være behov for å tilpasse slike metoder til å kunne benyttes for MMS, slik at også operatørenes jobb/oppgaver blir underlagt tilsvarende analyser, konstruksjon og evaluering som de andre systemkomponentene. Industri og forsvar har etter hvert innsett dette, delvis pga den rivende utviklingen spesielt innenfor informasjonsteknologi, som gir marginale konkurransefortrinn på teknisk utstyr alene. Vi ser altså en vridning fra teknologisorientert til behovs- og bruker-orientert utvikling.

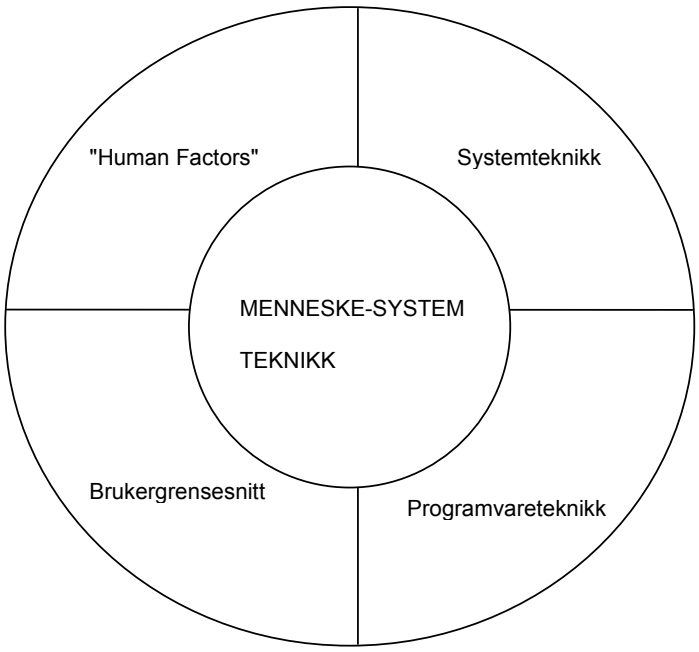
Menneske-maskin-systemer som disiplin har sin bakgrunn fra automatiseringsmiljøet. Her har man vært opptatt av egenskaper til den totale sløyfa av beslutninger, kommunikasjon, regulering og tilbakekobling gjennom den fysiske prosessen og tilbake til operatøren. Dette i motsetning til fokus på operatørgrensesnittet isolert. Brukergrensesnitt (underforstått datamaskin brukergrense-snitt), som disiplin, er typisk opptatt av generelle prinsipper for interaksjon mellom bruker og (data)maskin uten spesielt hensyn til i hvilken sammenheng den foregår. Altså i hovedsak en «nedenfra-opp»-holdning. Dette er viktig for å optimalisere og utvikle nye betjeningsteknikker til bruk i enkeltstående verktøy som tekstbehandling, regneark, osv, altså typiske kontorapplikasjoner. Ved innføring av moderne informasjonsteknologi for å støtte operatørene i store industrielle og militære systemer vil en slik angrepsmåte ikke føre frem. Operatørenes rolle og kvalitet av MMK må vurderes i lys av mål og begrensninger i den gitte applikasjonen. Man må ta et helhetssyn og dette fører til en mer systemorientert og problemdrevet innsats. Med andre ord, man er opptatt av egenskapene til MMS som en helhet. En nærmere diskusjon av

disse problemstillingene finnes i Rasmussen (1987), hvor kognitiv systemteknikk blir beskrevet og sammenlignet med forskning omkring brukergrensesnitt sprunget ut fra informatikk og psykologi, se også Rasmussen *et al*. (1994).

Vårt ståsted og fokus kan oppsummeres som følger:

- Systemorientert, dvs
  - Operatøren som en del av et system
  - Opptatt av MMS som helhet, ikke operatørgrensesnittet isolert sett
  - Større vektlegging av *hva* systemet skal oppnå i forhold til *hvordan*
- Vektlegging av de tidlige fasene av en systemutvikling
- Bruk av systemtekniske metoder
- Kombinasjon av «ovenfra-ned»- og «nedenfra-opp»-tilnærming

For å oppnå en utviklingsmetode som skissert ovenfor, er det behov for å ta kunnskap fra flere fagområder å integrere disse til en enhetlig utviklingsmetode. De viktigste områdene som *menneske-system-teknikk* består av og henter metoder og kunnskap fra er vist i figur 1.3. Som det vil fremgå av resten av kompendiet er vår ambisjon å integrere disse elementene til en utviklingsmetode som er egnet for utvikling av komplekse *menneske-maskin-system*er. Ordet menneske-system-teknikk er inspirert av kombinasjon av de to engelske ordene Human Engineering og System Engineering.



Figur 1-3 Menneske-system-teknikk

Etter å ha skissert vårt fokus og ståsted ovenfor er det nedenfor kort gjennomgått hvilket fokus en rekke fagfelt har og hvilke bidrag de gir ved utvikling av menneske-maskin-systemer. Det er lett å forstå at mange faggrupper kan gi nyttige innspill. Det er derfor også mange måter å betrakte et MMS på, og de forskjellige aspektene kan vektlegges i forskjellig grad.

Ett klart skille er om fokus er rettet på operatørene eller på de tekniske delene av MMS. Likeledes går det klare skiller avhengig av hvilket detaljnivå betraktningene skjer på.

Psykologi
Psykologi spiller en viktig rolle innenfor MMS med forståelse av menneskets egenskaper og begrensninger i interaksjon med omverdenen. Både lavnivå sensor-motoriske egenskaper og høyere nivå kognitive prosesser er viktige. Persepsjonspsykologi omfatter læren om hvordan mennesket oppfatter verden omkring seg med sine sanser. Forståelse av disse egenskapene gir oss retningslinjer og er til hjelp når avgjørelser om hvordan informasjon skal kodes og presenteres for en operatør skal tas. Eksempler kan være hvilke størrelser som er akseptable på tegn og bilde-elementer, bruk av farger, gruppering av informasjon, bruk av tegn, symboler og lyd. Kognitiv psykologi (se f eks Eysenck og Keane (1990) eller Ellis og Reed Hunt (1972)) omfatter studium av læringsprosesser og egenskaper til høyere nivås prosesser i forbindelse med problemløsning og informasjonsbehandling. Kognitiv psykologi gir oss retningslinjer og er til hjelp ved utforming av støttefunksjoner og hjelpemidler for operatørens problemløsning. Kognitiv psykologi er også viktig i forbindelse med forsøk på å automatisere ustrukturerte og ufullstendige problemområder som vanskelig lar seg formalisere og modellere eksakt. Denne grenen av psykologien beskriver også hvordan mennesker tar beslutninger og hvilke faktorer som påvirker deres valg (Hastie og Dawes (2001), Plous (1993)).

Fysiologi
Kunnskap om menneskets motoriske egenskaper er viktige ved at den gir retningslinjer ved valg av betjeningsorganer. Viten om motorisk dynamikk og nøyaktighet kan hjelpe oss til å kunne velge de betjeningssteknikkene som er best egnet i en bestemt situasjon.

Ergonomi/Antropoteknikk
Tradisjonell ergonomi omfatter studier og utforming av det fysiske grensesnittet mellom menneske og maskin (eller fysiske gjenstander). I motsetning til psykologi er man opptatt av menneskets fysiske dimensjoner og hvordan betjeningsorganer og informasjonsgivere skal utformes for å være best mulig tilpasset disse. I tillegg til utforming av spesifikt utstyr, er ergonomi også opptatt av utforming av operatørarbeidsplassen, dens umiddelbare nærhet og fysiske miljø (temperatur, lys, støy). Plassering av de enkelte enhetene sett i forhold til rekkevidder og belastninger er viktige problemstillinger i denne forbindelse.

Estetikk og industridesign
Utforming av det fysiske grensesnittet skal ikke bare være funksjonelt og tilpasset menneskets fysiske dimensjoner. Den må også falle i smak rent estetisk. Selv om dette tradisjonelt er lite forstått og vektlagt, skal man ikke undervurdere dette aspektet da mangt et produkt har mislykkes nettopp på grunn av disse forhold. Industridesign handler om å kombinere ergonomi med estetikk.

Sosiologi og organisasjonsteori
MMS opererer i en større sammenheng eller organisasjon og betjenes gjerne av flere operatører.

Det er derfor viktig å forstå og undersøke de mekanismene som oppstår når flere personer samarbeider i en gruppe for å nå felles mål og hvordan en slik gruppe forholder seg til andre personer og grupper i en større organisasjon. Forståelse av disse aspektene er viktig ved fordeling av oppgaver, myndighet og ansvar til de enkelte operatør for at gruppen som helhet skal fungere best mulig. Den resulterende sosiale struktur som dannes innenfor gruppen vil i stor grad kunne ha betydning på effektiviteten av det totale MMS.

De sosiologiske effektene ved å innføre ny teknologi og mer automatisering er et meget viktig felt, men lite forstått. Innføring av nye systemer er også i stor grad teknologidrevet, slik at slike aspekter ofte kommer i bakgrunnen.

Retter vi fokus på de teknologiske delene av et MMS kan man dele opp i følgende områder:

Betjeningsorganer og informasjonsgivere
Teknologi og fysiske prinsipper for omforming av maskininterne signaler til signaler for menneskelig persepsjon samt operatørsignaler til interne maskinsignaler er viktig for den kvalitet, pålitelighet og nøyaktighet som det er mulig å oppnå ved utveksling av informasjon mellom menneske og maskin. Bruk av katodestrålerør- og LCD-skjermer, og kapasitive og induktive trykkfølsomme paneler er eksempler på teknologi som blir benyttet i dag. I løpet av de siste årene har det kommet en jevn strøm av nye betjeningsorganer og informasjonsgivere. Nytt og avansert utstyr som vil få betydning i fremtiden er f eks talegjenkjenning og -syntese, «head-up»-skjermer, datahansker, osv.

Programvareteknikk
En altoverveiende del av operatørgrensesnittet realiseres vha programvare. Utvikling av programvare, programmeringsspråk, grafisk databehandling, ekspertsystemer og kunnskaps-baserte systemer vil alle ha betydning for hvilke grensesnitt som er mulige og hvor lett de er å implementere. Man ser også at operatørgrensesnittbehov er en av de viktigste pådrivere for teknologisk utvikling innenfor programvareteknologi og det er utviklet programmerings-teknikker og verktøy som er spesielt tilpasset MMK. Objektorientert programmering, konstruk-sjon og spesifikasjon har vist seg som spesielt egnet for å implementere typiske grensesnitt som er basert på direkte manipulasjon, grafikk og vindusteknikker. Ekspertsystemteknologi som f eks skall og verktøy for å lagre kunnskap og inferensregler er viktige for å implementere såkalte intelligente operatørgrensesnitt.

Operatørgrensesnittet vil også ha betydning for selve konstruksjonen av det totale programvare-system og hvordan dette bør spesifiseres. Bruk av prototyper som et viktig element under utvikling av programvare er ett eksempel på at MMK-aspekter har hatt direkte betydning for hvordan programvare utvikles.

Reguleringsteknikk og kybernetikk
Reguleringsteknikk eller kybernetikk er viktig for MMS da operatøren direkte opptrer som et element i reguleringssløyfene om enn på et overordnet nivå. Tilbakekobling som prinsipp er

viktig for informasjonsutvekslingen generelt. I forbindelse med modellering av MMS har reguleringsteknikk og estimeringsteknikk spilt en betydelig rolle spesielt i forbindelse med ferdighetsbasert adferd (se kapittel 3.3), og vil fortsatt spille en viktig rolle f eks gjennom de arbeidene som forgår i forbindelse med modellering, analyse og regulering av diskrete hendelsessystemer. Nye metoder og teknikker som blir tatt i bruk innenfor regulering, f eks regulering basert på fuzzy logikk og kunstige nevrale nett, er også felter som knytter menneskelig informasjonsbehandling og reguleringsteknikk sammen.

Samspillet mellom operatører og automatiserte estimerings- og reguleringsalgoritmer er et annet område som gir kybernetikk en viktig rolle ved utvikling av MMS. En kybernetisk betraktningsmåte og innfallsvinkel til MMS er viktig for mange aspekter i forbindelse med utvikling av MMS, kanskje spesielt i forbindelse med vurderinger og avgjørelser av automatiseringsnivåer og informasjonsutveksling.

Selv om Wiener (1948) ikke gav noe eksplitt definisjon av kybernetikk i sin bok, har den undertittel «kommunikasjon og regulering i levende vesener og maskiner». Samspill, fellestrekk og analogier mellom mennesker og maskiner har derfor alltid stått i fokus innenfor kybernetikken.

Generell systemteori, systemteknikk og systemering
I systemanalyse og systemteori er man opptatt av generelle egenskaper og strukturer til systemer. Hvordan disse egenskapene innvirker på de målene som systemet skal tilfredsstille og avveininger mellom parametere for å optimalisere systemer mhp effektivitet, ytelse, ressursforbruk, kostnad osv er noen problemstillinger. Operatøren blir betraktet som en systemkomponent og behandles i denne sammenhengen i sammenlignbare termer som de andre komponentene i totalsystemet. Bruk av en systemanalytisk tilnærming ved f eks beslutninger omkring hvilken rolle operatøren skal spille i totalsystemet er helt avgjørende. Mens mange av de andre områdene befatter seg med delaspekter av MMS, er man innenfor systemteknikk opptatt av MMS som et totalsystem, se f eks Meister (1991), uavhengig av hva slags teknologi som benyttes for å realisere dets komponenter. I mange tilfeller er dette et oversett og lite påaktet felt innenfor MMS, men som har stor betydning etterhvert som kompleksiteten av MMS har blitt større.

2 OPERATØRSENTRERT UTVIKLING

Dette kapitlet beskriver en *operatørsentrert systemarbeidsmodell* som vi ofte vil referere til senere i kompendiet. Med operatørsentrert (eller brukersentrert) menes ganske enkelt at vi fokuserer på operatørene og deres behov under systemutviklingen helt fra starten av. Alternative betegnelser på systemarbeidsmodell er prosjektmodell og prosessmodell.

En *systemarbeidsmodell* defineres slik i (Bræk *et al*., 1985): «En oversikt som i store trekk viser hvilke faser, aktiviteter, og beslutningspunkter som normalt inngår i et prosjekts livsløp, og hvilke dokumenter som produseres». Behovet for å nytte en slik modell melder seg når det er

snakk om utvikling av store og kompliserte systemer der mange personer skal arbeide sammen over en lang tidsperiode, gjerne flere år. En *systemarbeidsmetode* beskriver mer eller mindre konkret/detaljert *hvordan* aktiviteter i systemarbeidsmodellen skal utføres, eventuelt med støtte av dataverktøy.

Et system gjennomløper mange stadier eller *faser* i løpet av dets «liv»:

1. Identifikasjon av behov
2. Definisjon av krav og spesifikasjoner
3. Mulighetsanalyse
4. Planlegging
5. Logisk og fysisk konstruksjon (design)
6. Realisering (implementasjon) og testing
7. Drift (operativ fase) og vedlikehold
8. Evaluering og utfasing

Følgende deler er sentrale i en systemutvikling:

- Systemarbeidsmodell
- Metoder og retningslinjer
- Utviklingsverktøy
- Beskrivelser og dokumentasjon

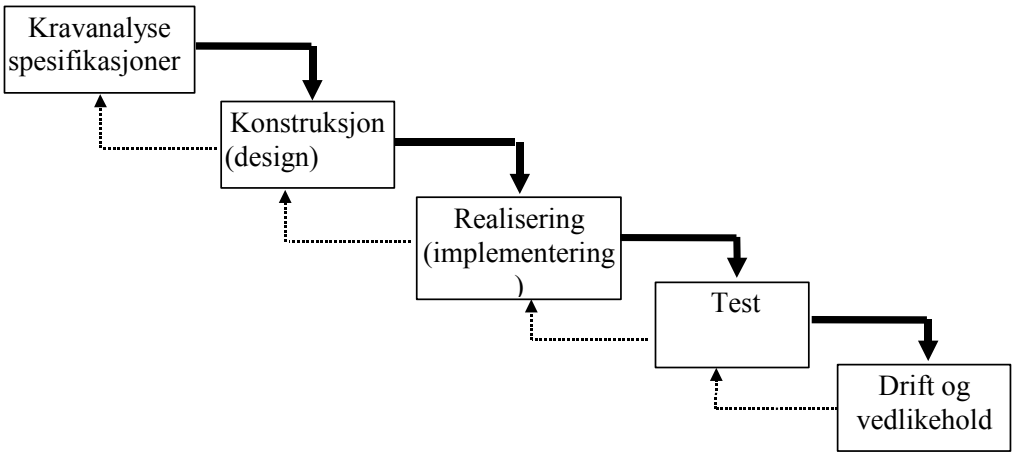
I det etterfølgende skal vi i grove trekk se på hva som skiller en del av de mest kjente systemarbeidsmodellene.

2.1 Systemarbeidsmodeller

En systemarbeidsmodell beskriver altså hvilke aktiviteter som må utføres i de ulike fasene av et systems livsløp. Modellen angir rekkefølgen av aktivitetene og knytter disse til hvilke beslutninger som normalt skal tas. Den kan angi kriterier for oppstart og avslutning av faser/aktiviteter. Når det gjelder en systemarbeidsmodell for utvikling av MMS bør denne være operatørsentrert og ha en sterk kopling til systemarbeidsmodeller for programvareutvikling. Utvikling av programvare er den klart største kostnadsfaktoren i forbindelse med utvikling av MMS. Ulike typer av systemarbeidsmodeller for programutvikling er:

- Kode og fiks (prøve og feile)
- Vannfallsmodellen
- Prototyping
- Spiralmodellen
- Kombinasjoner, f eks prototyping for kravsetting og vannfallsmodellen for de andre fasene

En stegvis utviklingsmodell er illustrert i figur 2.1. Ideelt skal fasene utføres sekvensielt uten tilbakekopling. Denne modellen kalles ofte *vannfallsmodellen* (eller fossefallsmodellen), jfr figur 2.1. Modellen legger relativt liten vekt på iterasjon mellom fasene og interaksjon mellom utviklere og brukere. Den forutsetter egentlig at det er liten usikkerhet mhp brukerbehov og at brukerkrav effektivt lar seg identifisere i startfasen (spesifikasjonsfasen). Fordi tilbakekopling i praksis er nødvendig er dette i noen grad innført i varianter av modellen, men hovedfilosofien er fortsatt at hver fase tilnærmet kan ferdigstilles før den neste begynner.

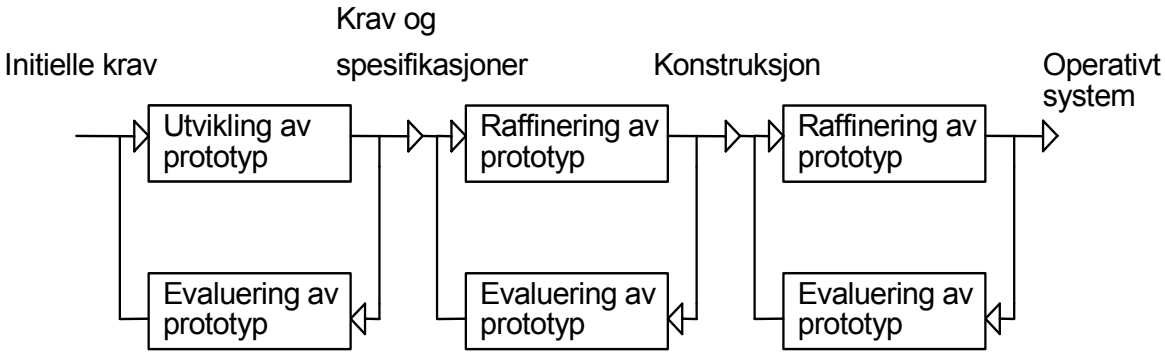


Figur 2-1 Vannfallsmodellen

Når et komplisert MMS utvikles vil man i dag nesten alltid lage en eller flere *prototyper*. Prototyping er en nyttig metode under hele utviklingen. I den initielle fasen kan prototyper brukes for å illustrere og vise konsepter som kan danne basis for systemløsninger. Senere kan de brukes til å til å bestemme automatiseringsnivå og detaljere funksjonalitet. Under konstruksjonen av selve MMK er prototyping idag nærmest betraktet som en nødvendighet.

Prototyping er konstruksjon av en kjørbar systemmodell. Den kan betraktes som en (sterkt) forenklet realisering av et MMS og representerer en fysisk modell av brukergrensesnittet. Ofte brukes betegnelsen «man-in-the-loop»-prototyp (simulator) for å understreke at operatøren kan bruke prototypen tilnærmet likt det operative systemet.

Prototyping kan bidra vesentlig til å etablere og validere brukerkrav. Prototyping er videre et viktig virkemiddel for å sikre brukermedvirkning. Dersom prototypen raffineres til et operativt system slik som det er skissert i figur 2.2 kalles det *evolusjonær prototyping.* Det er imidlertid vanskelig å sikre kvaliteten til programvaren fordi prototyping er en iterativ prosess der man raskt og effektivt ønsker å teste ut alternative løsninger. Når et MMS/programvaresystem skal realiseres kommer det også inn diverse tekniske/operative hensyn som ofte ikke er ivaretatt under raffineringen av prototypen. Dette gjør at man sjelden benytter denne system-arbeidsmodellen fullt ut. Prototyping er imidlertid med som et viktig element i så godt som alle typer systemarbeids-modeller som er i bruk i dag. Ved evolusjonær prototyping implementeres funksjonalitet med minst usikkerhet først.



Figur 2-2 Evolusjonær prototyping

Dersom man utvikler mer frittstående prototyper som ikke direkte er hengt opp i utviklingen av det endelige MMS, kalles det *eksperimentell prototyping*. På engelsk nyttes betegnelsen «throwaway approach». Slike prototyper konstrueres (ideelt sett) raskt og billig og «kastes» etter bruk. Rask tilbakemelding fra brukerne og uttesting av alternative løsninger vektlegges. Man vil som regel begrense seg til å studere deler av systemet og valget er ofte motivert utfra det som vurderes som mest kritisk/usikkert. I motsetning til evolusjonær prototyping implementeres altså den funksjonaliteten som er mest usikker først. I kapittel 7 skal vi diskutere hvordan man evaluerer prototyper og hvilke problemer man da støter på.

Spiralmodellen (Boehm, 1988) kombinerer ulike deler av andre modeller, bl a eksperimentell prototyping. Denne modellen er sterkt iterativ og gir en gradvis og risikodrevet utbygning av systemet (dvs fokus på reduksjon av risiko).

En *operatørsentrert modell* har som målsetning en behovsdrevet og ikke teknologidrevet utvikling. En løsning som er teknisk eller «ingeniørmessig» optimal er ikke nødvendigvis den beste for operatøren og den organisasjonen han tilhører. Operatørenes systemperspektiv skal dominere og styre utviklingen og det forutsetter således operatørmedvirkning under hele utviklingen, f eks ved at disse er en del av utviklingsgruppen eller utgjør en såkalt *referanse-gruppe*. Kjennskap til operatørpopulasjonen er ansett som svært viktig, f eks ved etablering av representative referansegrupper. Tidlig prototyping av løsninger, og evaluering av disse i samarbeid med operatørene, er en viktig del av denne modellen. Modellen representerer på mange måter en kombinasjon av en «ovenfra-ned»- og «nedenfra-opp»-utvikling og kan også sees på som en vektlegging av et "utenfra-inn"-perspektiv i motsetning til et "innenfra-ut"-perspektiv.

I moderne system- og programutviklingsteknikk understrekes betydningen av å legge tilstrekkelig innsats i de første fasene. Kostnadene tidlig under en utvikling er som regel relativt små (f eks 10 %) samtidig som man da allerede har bestemt størsteparten av utviklingskostnadene (f eks 90 %). Ved å øke den tidlige innsatsen, og derved forhåpentligvis kvaliteten av arbeidet i startfasen, vil man kunne redusere de totale utviklings- eller levetidkostnadene.

Kostnadene ved å rette opp systemer i operativ drift er vesentlig større enn dersom man endrer kravene eller spesifikasjonene tidlig i utviklingen. Et realistisk forholdstall kan være så stort som 100:1. Når det gjelder MMS er en vesentlig del av ressursene forbundet med programvare-utviklingen av brukergrensesnittet. Likevel er mange av endringene til systemer som har vært i drift relatert til problemer med nettopp brukergrensesnittet, eller mer generelt; samspillet mellom menneske og maskin. Formålet med en operatørsentrert utviklingsmetode er å redusere disse problemene, og derved lage et bedre og mer kosteffektivt system.

2.2 Systemarbeidsmodeller for utvikling av MMS

Før vi går inn på de enkelte elementene i modellen skal vi utdype hva vi forstår med operatørsentrert utvikling. Hovedfilosofien, slik f eks Rouse (1991) uttrykker det, er at mennesker er ansvarlige for å oppnå systemmålsetninger og at de, *uansett automatiserings-nivå*, e*r ansvarlige for driften av kompliserte systemer.* Det er nærmest utenkelig å tillegge et automatisk system et juridisk, etisk eller sosialt ansvar! Kort sagt, operatøren må ha kontroll («be in charge»). Målsetningen for systemutviklingen bør derfor være «å støtte mennesker slik at de kan oppnå målsetninger som de er ansvarlige for». Denne tankemåten kan eksemplifiseres slik: hensikten med en pilot (dvs operatøren) er ikke å kunne fly et fly som frakter mennesker fra A til B, men *hensikten med flyet er å understøtte piloten* som er ansvarlig for å transportere mennesker fra A til B! Poenget er at man ikke tar utgangspunkt i teknologi for å oppnå systemmålsetningen, men bruker og integrerer teknologi for å understøtte operatørene på en måte som er i samsvar med deres rolle og tilhørende ansvar.

Hva innebærer dette mer konkret når et system skal realiseres? Rouse (1991) nevner følgende aspekter som eksempler:

- Utnytte og videreforedle de menneskelige egenskaper, f eks til mønstergjenkjenning
- Unngå eller redusere de menneskelige begrensninger, f eks tendensen til å gjøre feil. Systemet bør derfor f eks ha angremuligheter og bør eventuelt også kunne detektere og varsle operatøren om mulige feil.
- Etterstrebe brukerens akseptanse og tilfredstillelse med systemet. Vil operatøren eller organisasjonen bruke systemet når det settes i drift, og på en måte som er iht systemmålsetningen?

Man kan snakke om brukersentrert utvikling på ulike nivåer:

1. Grensesnittnivå (operatørens detaljerte aktiviteter)
2. Operatørens jobb og de oppgavene denne består av
3. Organisasjonsnivå (definerer konteksten til nivåene over og involverer også mange andre enn operatørene av systemet)

Kompendiet omhandler aspekter knyttet til alle nivåene, *men fokus er på det mellomste nivået.*

Tradisjonell systemutvikling tar i liten grad hensyn til operatørene i startfasen, og tildels også i den videre prosessen. Operatøren er ofte det mest fleksible elementet ved konstruksjon av et

system. Overforbruk av slik fleksibilitet kan imidlertid gå utover ytelse og sikkerhet og resultere i et unødvendig dårlig system. Industri og forsvar har tildels innsett behovet for en vridning i fokus fra teknologi til menneskene i systemet. Man har f eks anslått at personell utgjør opptil 50 % av levetidskostnadene for enkelte typer fartøyer i USAs marine (Bost & Oberman, 1994).

Som eksempel på denne trenden skal vi kort nevne to initiativer relatert til militære systemer. Forsvaret i Norge har fra 1994 innført prosjektstyringssystemet «Prinsix» som skal brukes i alle nye materiellanskaffelser (Prinsix, 1991). Prinsix omfatter:

- Systemarbeidsmodell
- Prosjekthåndbøker og opplæringsprogram
- Ulike typer dataverktøy

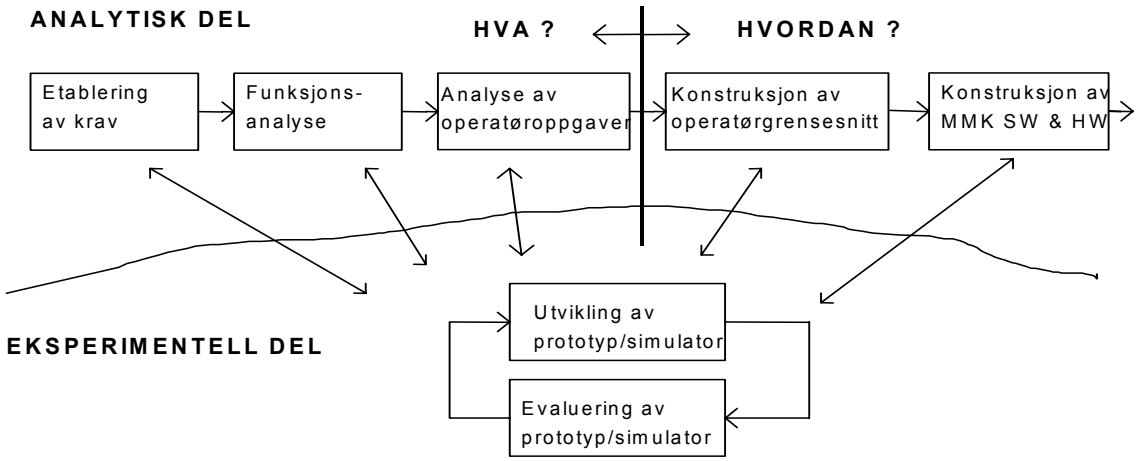
Hovedformålet med å innføre Prinsix er å sikre at tilstrekkelig analysearbeid blir gjort, og at viktige beslutninger tas *før* man inngår kontrakter. Man ønsker også i større grad å sikre seg mulighet til å endre kurs under utviklingen, f eks som resultat av teknologisk utvikling som ofte kan påvirke prosjekter som går over lang tid. Under definisjonsfasen (dvs kravanalyse/spesifikasjon) etableres basis for valg av systemløsning og prosjektgjennomføring. Med «system» menes her *alt* som berører innføring av materialet under dets levetid, som f eks vedlikehold, personellbehov og deres tilhørende opplæringsbehov. Begrepet «operatører» bør således også inkludere personell som driver vedlikehold.

Prinsix har en del likhetstrekk med MANPRINT (Manpower and Personnel Integration) som er en omfattende systemarbeidsmodell og metode utviklet av militære anskaffelsmyndigheter i USA og store europeiske NATO-land (Booher, 1990). Hovedformålet med MANPRINT er *integrasjon* av seks ulike områder som er av betydning for brukere av systemer: bemanning (antall), personell (kvalifikasjoner), opplæring, MMS-teknikk, sikkerhet og helserisiko (forbundet med operering av utstyr).

2.3 Operatørsentrert systemarbeidsmodell for utvikling av MMS

Vi skal heretter beskrive en operatørsentrert systemarbeidsmodell som på mange måter danner et rammeverk for innholdet i kompendiet. Vi vil imidlertid begrense oss til selve *utviklingen* av MMS. Dvs vi vil i liten grad komme inn på utprøving av ferdig system, operativ drift og vedlikehold. Modellen består av en *analytisk/formell* del og en mer *eksperimentell* del. De to delene er komplementære og det vil være mer eller mindre sterke koblinger mellom dem, som antydet i figur 2.3. En formell modell av MMS (fra analytisk del) kan representere ett sterkt bindeledd fordi man da delvis kan automatisere utviklingen av prototyper og simulatorer og fordi man derved også sikrer konsistens mellom de to delene.

Vi skal nå gi en oversikt over aktivitetene innenfor hver av fasene i denne systemarbeids-modellen. Aktivitetene og begrepene som nyttes beskrives mer detaljert i senere kapitler.



Figur 2-3 Systemarbeidsmodell for utvikling av MMS

2.3.1 Aktiviteter innenfor den analytiske delen av systemutviklingen

Den analytiske delen starter med *etablering av krav* som omfatter:

- Bestemme overordnede krav til MMS (systemmålsetning)
- Scenarioanalyse (Mission Analysis)
- Operatøranalyse
- Analyse av eksisterende systemer
- Identifikasjon av overordnet systemstruktur (dvs de viktigste systemkomponenter) og overordnede systemfunksjoner

En funksjon beskriver en del av systemets oppførsel, dvs hva systemet utfører.
Dernest gjennomføres *funksjonsanalyse* som omfatter:

- En detaljering av systemfunksjoner og systemstruktur
- Bestemmelse av automatiseringsfilosofi/nivå og hvilken rolle operatøren skal ha i systemet
- Fordeling av funksjoner (funksjonsallokering) til henholdsvis operatør og maskin (maskin- og programvare)

I modellen vår konsentrerer vi oss deretter om utviklingen av den delen av totalsystemet som (direkte) har med operatører å gjøre. *Analyse av operatøroppgaver* inneholder:

- Detaljering av operatøroppgaver (støtte- og kontrollfunksjoner)
- Identifikasjon av informasjonsbehov og informasjonsflyt mellom operatør/maskin og operatørene seg i mellom
- Vurdering av krav (f eks kunnskap) som stilles til operatøren og vurdering av arbeidsbelastning

2.3.2 Aktiviteter innenfor den konstruksjonsmessige delen av systemutviklingen

I de neste fasene er fokus på systemløsningen. Først utføres *konstruksjon («design») av operatørgrensesnittet* og stikkord for denne fasen er:

- Etablere retningslinjer som skal gjelde for operatørgrensesnittet
- Valg av klasse og typer av interaksjon (f eks direkte manipulering)
- Valg av betjeningsteknikker (f eks identifikasjon av objekt ved direkte kontakt med dets representasjon)
- Valg av logiske betjeningsorganer (f eks pekeenhet)

I den siste fasen realiseres MMK:

- Valg av fysiske betjeningsorganer, program- og maskinvarekomponenter, f eks mus, tastatur, datamaskiner, grafikkpakker og vindussystemer
- Konstruksjon og koding av MMK-programvare

I forbindelse med analysene over nyttes ofte (grafiske) formelle beskrivelser. Disse kan også brukes til å spesifisere og konstruere operatørgrensesnitt og de kan brukes som basis for simuleringer.

2.3.3 Aktiviteter innenfor den eksperimentelle delen av systemutviklingen

I *parallell* med den analytiske delen gjennomføres den eksperimentelle delen som illustrert i figur 2.3. Den eksperimentelle delen omfatter både simulering og prototyping. Simulering er som regel sterkt koblet til de modellene/beskrivelsene som man utvikler av de første fasene av den analytiske delen og typen simulering vil avhenge av hvor langt man er kommet i utviklingen. Ulike typer av simulering er:

- Symbolsk/logisk simulering
- Tidssimuleringer (deterministisk og stokastisk)
- Ressurssimuleringer
- Algoritme- og ytelsessimuleringer

Tilsvarende vil nivået av MMK prototyping variere i løpet av utviklingen:

- Statiske bilder
- Avspilling av bildesekvenser
- Avspilling av dynamiske bilder
- «Man-in-the-loop»-simuleringer der operatøren benytter en prototyp tilnærmet likt det operative systemet og der man simulerer omgivelsene med en viss realisme.

Bruk av prototyping inkluderer mer eller mindre formelle evalueringer. Viktige aspekter er:

- Målsetning med aktivitetene

- Gjennomføring av forsøk:
  - Dataregistrering (Logging under kjøring, observasjons- og intervjuteknikker, bruk av spørreskjemaer og verbal protokollanalyse)
  - Valg av forsøkspersoner og type forsøk
  - Vurdering av intern validitet og «trussel» mot denne
  - Vurdering av ekstern validitet (generaliserbarhet)
- Kvantitative og kvalitative teknikker for måling av arbeidsbelastning:
  - Oppførsel-/ytelsesbaserte målinger
  - Subjektive målinger
  - Psykofysiologiske målinger
- Bruk av statistikk (hypotesetesting, statistisk og «praktisk» signifikans)

Under utvikling av MMS er det behov for mange typer dataverktøy:

- Systemanalyseverktøy for kravhåndtering og systemmodellering
- Simulerings- og prototypverktøy
- DAK for konstruksjon av arbeidsplassutforming
- Verktøy for spesifikasjon av programvare
- Verktøy/programvaresystemer for realisering av MMK-grensesnitt
- Database for retningslinjer og «tommelfinger-regler»

Som omtalt over vil modellene som utvikles i den analytiske delen av utviklingen representere ett bindeledd med det eksperimentelle systemarbeidet. Vi skal derfor i neste kapittel ta for oss modellering av MMS.

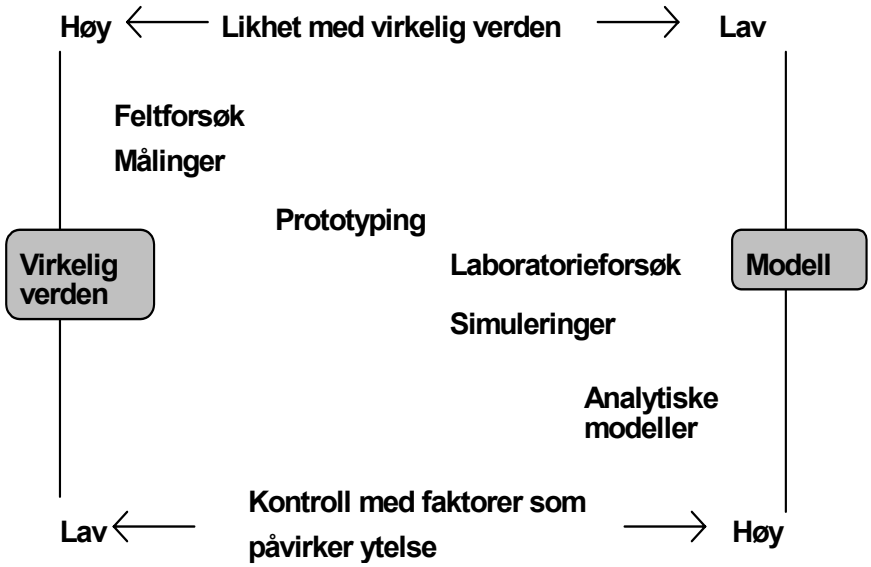
3 MODELLERING

Modellering - hva er det og hvorfor gjør vi det? I dette kapitlet skal vi gi en oversikt over modellering av MMS. Et system kan studeres indirekte vha en modell av systemet. Dvs at man kan generere informasjon ved hjelp av modellen og deretter analysere denne. En modell er en *representasjon* av det man mener er viktige egenskaper til systemet. Representasjonen kan f eks være matematiske ligninger, diagrammer, skjemaer, tekst eller fysiske skalamodeller. På samme måte som en arkitekt åpenbart har nytte av å studere tegninger eller skalamodeller av bygninger, vil en som utvikler MMS ha nytte av å analysere modeller av det systemet som skal konstrueres og realiseres. Denne analogien er kanskje litt misvisende fordi f eks programvaren i MMS vil ha flere "dimensjoner" og være vesentlig mer komplisert. Brooks (1987) uttrykker dette slik: «The complexity of software is an essential property, not an accidental one. Hence, descriptions of a software entity that abstract away its complexity often abstract away its essence». En forholdsvis komplett modell av MMS vil ha mange dimensjoner og man må i praksis vanligvis benytte flere modeller og ulike typer av *projeksjoner* av disse.

En modell uttrykkes i et modellspråk og kravet til dette språket vil avhenge av hva som skal modelleres og formålet med modellen. Hva som er «viktige egenskaper» er egentlig en subjektiv vurdering gjort av modellutvikleren med utgangspunkt i hans spesifikke problem og behov for analyse. Vurderingen av hva som skal (bør) modelleres vil avhenge av *hva* man ønsker å analysere. Det å snakke om en riktig eller «god» modell er meningsløst dersom bruksområde ikke er definert. En modell vil derfor variere mhp grad av formalisme, abstraksjonsnivå og perspektiv. I et MMS/programvaresystem vil f eks fokus kunne være på informasjonsbehandling og involvere kompliserte datastrukturer eller man vil kunne ønske å fokusere på dynamiske aspekter, dvs når og under hvilke betingelser aksjoner utføres.

Modellering anses i dag som helt essensielt ved utvikling av kompliserte systemer. Selve modelleringsprosessen er ofte i seg selv svært nyttig og gir økt innsikt og forståelse. Modellene vil også ofte være et viktig grunnlag for systemdokumentasjonen som produseres under utviklingen. En viktig begrunnelse for dataverktøy som understøtter systemarbeid er delvis automatisert produksjon av dokumentasjon iht systemarbeidsmodellen, feks iht standarder som Mil Std 490B. I dette kapitlet skal vi imidlertid distansere oss fra dette aspektet og se mer på *grunnlaget* for modelleringen av operatører i et MMS.

Vi ønsker å fokusere på modeller som kan være utgangspunkt for analytiske beregninger og simuleringer av forskjellige aspekter knyttet til samspillet mellom operatør og maskin. Disse kan brukes til å *prediktere* systemoppførsel/ytelse og derved benyttes til å optimalisere konstruksjonen av systemet. En modell av operatørens oppførsel kan også inngå i systemet selv slik at dette blir mer adaptivt. En kompakt forklaring av data i et sett med modellparametre er en representasjon av kunnskap. Modeller av operatøren er derfor en viktig basis ved utvikling av intelligente grensesnitt og ekspertsystemer.



Figur 3-1 Sammenheng mellom forhold knyttet til måling av ytelse i MMS og kontroll med faktorer som påvirker denne. Etter (Beevis et al., 1992)

Vi vil i hovedsak diskutere ytelsesevaluering/prediksjon knyttet til kvantitative mål som tid, nøyaktighet og feil. Ulike metoder for å måle systemytelse er vist i figur 3.1. Figuren illustrerer at man ved valg av metode står ovenfor en avveining. Likheten med virkelige forhold er høy ved feltforsøk. Under komplekse feltforsøk er imidlertid kontrollen med kjente (og ukjente) faktorer som påvirker oppførselen forholdsvis liten. Feltforsøk har også den åpenbare svakhet at det krever et virkelig system eller en prototyp. Simuleringer tar sikte på å prediktere ytelsen under mer kontrollerte forhold, men gyldigheten av å generalisere resultatene til virkelige forhold er mer usikker. Analytiske modeller kan fange opp viktige karakteristikker av systemet på en svært effektiv måte, men evnen til å prediktere ytelsen i den virkelige verden er begrenset.

Ved å manipulere en modell av et system på ulike måter kan man oppnå å få «ny» kunnskap om systemet raskere, til mindre kostnad og uten de ulempene og farene som forsøk med det virkelige systemet ville kunne medføre. Evaluering av operatørens ytelse er like viktig som andre system-komponenter, men generelt sett mye vanskeligere. Empiriske data fra forsøk kan uttrykkes i empiriske modeller og disse kan inngå som delmodeller i simuleringsstudier eller som parameterverdier i ulike typer av analytiske modeller.

Modeller bør om mulig *valideres* vha målinger. Eventuelt bør man også kontrollsjekke enkle varianter av modellen vha analytiske beregninger. Er antakelsene holdbare? Er modellen i tilstrekkelig grad lik virkeligheten til at resultatene har praktisk relevans? Man må imidlertid huske på at avvik mellom målinger og modeller også kan skyldes målefeil!

3.1 Systemteori

Som det tidligere har blitt fremhevet er det viktig for utvikling av menneske-maskin-systemer at det legges til grunn et systemperspektiv. Som en basis for et slikt perspektiv vil dette kapitlet gi en kort og svært enkel innføring i generell systemteori, ved å definere noen begreper i tilknytning til hva som menes med et system og hva som må til for å beskrive egenskaper til systemer. Som vi skal se senere vil denne basisen være nyttig i forbindelse med mange aspekter, som f eks modellering, spesifikasjon, osv.

3.1.1 Systembegrepet

Det finnes mange forskjellige definisjoner av begrepet system som varierer sterkt i formalitet og anvendelighet. Her skal vi benytte en forholdsvis uformell definisjon:

> Et *system* defineres som en samling eller kombinasjon av elementer eller deler som samvirker for å oppnå et felles mål. Elementene danner tilsammen en enhet hvor dens egenskaper er forskjellig fra egenskapene til mengden av elementene. Se f eks Blanchard og Fabrycky (1981) eller Thomé (1993).

For en mer matematisk orientert definisjon av system, se f eks Kalman (1969).

Ikke alle mengder av elementer danner et system. En tilfeldig gruppe elementer vil danne en mengde med relasjoner mellom elementene, men de vil ikke bli kvalifisert som et system, pga mangel på enhet, funksjonelle relasjoner og et «nyttig» felles mål. Mengden av komponenter har følgende egenskaper:

- Egenskapene og oppførsel til hver av komponentene i mengden har innvirkning på egenskapene og oppførsel til mengden som helhet.
- Egenskapene og oppførsel til hver av komponentene er avhengig av egenskapene og oppførsel til minst en av de andre komponentene i mengden.
- Alle mulige delmengder av komponenter har de samme to egenskapene som beskrevet ovenfor.

Målet eller hensikten til et system må være eksplisitt definert. Ut fra det mål eller den hensikt som er definert for et system er det mulig å etablere effektivitetskriterier for hvor godt systemet funksjonerer. Det å etablere mål og hensikt med tilhørende effektivitetskriterier for et system er en viktig oppgave.

Legg også merke til at definisjonen fremhever samvirket mellom delene (dvs hvordan delene er knyttet sammen og hva som utveksles mellom dem) som systemet består av. Interaksjonen mellom elementene i systemet er vel så viktig som egenskapene til elementene selv og det er denne interaksjonen som gir opphav til «nye egenskaper» til systemet («emergent properties») og som er bakgrunn for å si at «et system er mer enn summen av sine komponenter, eller at to pluss to blir fem». Likeledes vil enkeltelementer ikke kunne forstås i isolasjon fra systemet som helhet.

Systembegrepet er også relativt. Det er opp til betrakteren å inkludere hva som danner et system. Det som i en sammenheng blir betraktet som et system, kan i neste omgang inngå som en komponent i et annet, større system. Hva som betraktes som et system må altså ses i forhold til den hensikt avgrensningen av systemet skal oppfylle. Et system inngår nesten alltid i et større system. Vi kan derfor ikke betrakte et system isolert sett, men må også ta i betraktning hvordan det blir påvirket og påvirker det systemet det inngår i. Vi har altså et hierarki av systemer hvor de lavere nivåene er *komponenter* som inngår i et system. Hver komponent kan igjen bestå av lavere nivås komponenter. Hvis vi har flere nivåer av komponenter benyttes gjerne betegnelsen *subsystem* på et eller flere nivåer av hierarkiet.

Et systems deler kan være av hvilken som helst type; det være seg mennesker, programvare, eller elektroniske, mekaniske, kjemiske komponenter osv. Det som omtales her er altså uavhengig av hvordan systemet materaliserer seg og er egenskaper til systemer og beskrivelse av systemer generelt som gjelder for alle typer systemer.

3.1.2 Systemgrense og systemomgivelse

Da systembegrepet er relativt, er det nødvendig å definere dets avgrensning for å gjøre begrepet

absolutt i en gitt sammenheng. Denne avgrensningen kalles *systemgrensen*. Alt som befinner seg utenfor et systems grense, er definert som *systemomgivelsen* (også kalt *omegn*). Det er svært viktig at denne avgrensningen mellom systemet og dets omgivelse blir klargjort og eksplisitt formulert. For kompliserte systemer er denne avgrensningen mellom systemet og dets omgivelse ikke opplagt og kan skape forvirring hvis den ikke blir eksplisitt beskrevet og forstått av alle parter som er involvert.

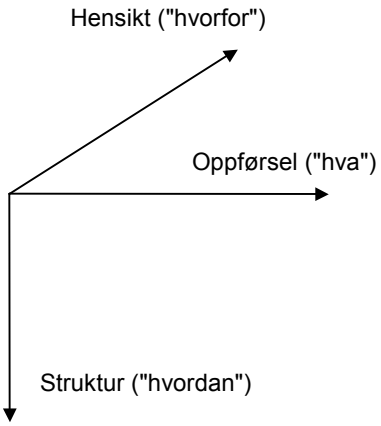
Avhengig av om det er noen relasjon med omgivelsen, er systemet *lukket* eller *åpent*. Et lukket system har ingen relasjoner av betydning til omgivelsene, mens et åpent system har det. Nesten alle systemer er åpne. Systemer i omgivelsen blir betegnet *eksterne systemer*. Når vi betrakter et gitt system er det altså nødvendig å ta hensyn til og innlemme deler av dets omgivelse (dvs de eksterne systemer som systemet har en relasjon til). Det er altså ikke nok å betrakte systemet isolert hvis det er åpent, men de eksterne systemene som har en relasjon til det gitte systemet må også tas i betraktning. For å kunne studere og analysere et system, må vi altså ta med så mye av omgivelsen at vi kan betrakte det resulterende systemet som lukket. Dvs at de relasjonene som det resulterende systemet har med sin omgivelse kan sees bort fra. Det er også vanlig at de eksterne systemene ikke blir underlagt en like detaljert analyse og beskrivelse som det gitte systemet. Men, det er altså egenskapene til det lukkede systemet av eksterne systemer og det gitte system som f eks setter kravene til de beskrivelsene og modellene som er nødvendig å utarbeide under en systemutvikling.

3.1.3 Beskrivelse/modellering av systemer

Når vi ønsker å beskrive (eller modellere) systemer kan vi benytte forskjellige representasjoner. De forskjellige representasjonene varierer i hvor formelle og presise de er. Her skal vi forsøke å gi en enkel fremstilling av hva som inngår i en systembeskrivelse og noen begreper som benyttes. Med andre ord ønsker vi å få frem hva en systembeskrivelse/modell består av, begreper som benyttes i denne beskrivelsen og sammenhengene mellom det som inngår i en systembeskrivelse. På en måte beskrives kravene til en representasjon som kan benyttes for å modellere/beskrive systemer. Legg merke til at en systembeskrivelse også tilfredstiller vår definisjon av et system, men vi må alltid ha klart for oss forskjelligen mellom systemet selv og en representasjon av det. Når vi arbeider med mer abstrakte systemer, f eks programvare-systemer kan dette skillet bli uklart. Vi har heller ikke hele tiden vært konsekvent i vår språkbruk. Der det ikke skulle være rom for misforståelser har vi noen steder benyttet system i stedenfor modellen av systemet for å gjøre teksten mer lesbar og mindre tung.

Et system beskrives av dets *struktur* (også kalt arkitektur) og av dets *oppførsel*. I tillegg vil også systemets *hensikt* være nyttig ved arbeid med utvikling av systemer. Et systems struktur, oppførsel og hensikt kan ses på som tre ortogonale dimensjoner av en systembeskrivelse som vist i figur 3.2. Systemets struktur beskriver hva systemet *er*, mens systemets oppførsel beskriver hva systemet *gjør*. Systemets hensikt beskriver hvorfor systemet er som det er og hvorfor det oppfører seg som det gjør. Nedenfor vil vi konsentrere oss om systemstruktur og systemoppførsel. Det betyr ikke nødvendigvis at hensikt ikke er viktig, men beskrivelse av hensikt gjøres gjerne mye mer uformelt og den representeres med vanlig tekst. Spesielt for

MMS skal ikke hensiktsbeskrivelse nedvurderes, da det er viktig for systembrukere å forstå hensikt for at de skal opparbeide tiltro til systemet.



Figur 3-2 Dimensjoner i en systembeskrivelse

Systemstruktur
Et systems struktur definerer dets bestanddeler eller hva systemet består av. Et system består av komponenter, med sine attributter og relasjoner med andre komponenter.

- *Komponenter* - er de delene som et system bygges opp av. Komponenter kan anta forskjellig form, fysisk eller abstrakt.
- *Attributter* - Komponentens attributter betegner dens egenskaper.
- *Relasjoner* - er sammenhenger mellom komponentene.

Et system består altså av et sett med sammenkoblede komponenter som virker sammen for å nå et felles mål.

I de fleste tilfeller er et systems struktur tidsinvariant. Nedenfor kommer vi nærmere inn på hvordan systemstruktur gjerne beskrives.

Systemoppførsel
De «målrettede» aksjonene som et system utfører kalles dets *oppførsel* (også kalt et systems funksjon). Et systems oppførsel kan være å prosessere fysisk materiale, energi eller informasjon. Oppførsel innebærer *inngang (eller pådrag), prosessering og utgang* som funksjon av tid. Inngangsvariablene eller pådraget betegnes gjerne **u**(t) og utgangsvariablene betegnes **y**(t). Til oppførsel er det tilknyttet ytelse. Ytelse kan f eks være hvor hurtig prosesseringen foretas, med hvilken nøyaktighet den utføres, osv.

Mens systemets struktur typisk er tidsinvarient eller statisk, er oppførsel tidsavhengig. Tidsvariasjonen kommer til uttrykk enten ved at inngangsvariable forandrer seg over tid (for statiske systemer), eller ved at utgangsverdien ikke bare er avhengig av den inngangsverdien som påtrykkes i øyeblikket, men også av forgående inngangsverdier (dynamisk system). Den

klasse av MMS som vi er opptatt av er dynamiske. Et dynamisk system har altså hukommelse. Hukommelsen til systemet beskrives av dets *tilstand*. Dvs at tilstanden til et system beskriver dets oppførsel ved et gitt tidspunkt.

Tilstanden til et system defineres som:

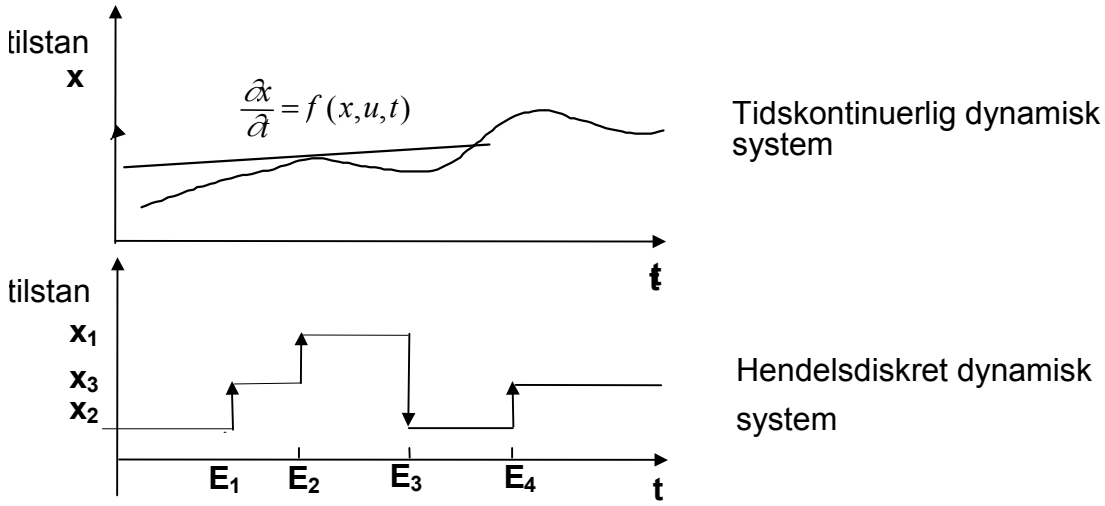
> Tilstanden til et system ved tidspunkt t_o , $\mathbf{x}(t_o)$, er den informasjonen som er nødvendig ved t_0 slik at utgangen $\mathbf{y}(t)$, for alle $t > t_0$ er entydig bestemt fra denne informasjonen og fra inngangen $\mathbf{u}(t)$, $t > t_0$. (Cassandras,1994).

Beskrivelser av systemoppførsel skiller seg ut fra det utfallsrommet som velges for systemets tilstandsvektor (tilstandsrom). Tilstandsrommet kan enten velges å være *kontinuerlig* eller *diskret*, eller begge deler, kalt *hybrid*. Kontinuerlige tilstandsrom beskrives av n-dimensjonale vektorer av reelle (eventuelt komplekse) tall, mens diskrete tilstandsrom består av en endelig diskret mengde. Hybride tilstandsrom består både av kontinuerlige og diskrete tilstander. Legg merke til at det er tilstandsrommet til systembeskrivelsen som er kontinuerlig, diskret eller hybrid. Visse systemer faller naturlig inn i de forskjellige kategoriene, men et gitt system kan ofte modelleres på alle tre måter avhengig av hva vi ønsker å benytte modellen til. Menneskeskapte systemer betraktes gjerne som diskrete, f eks et produksjonssystem, et kommunikasjonssystem, en havn, osv, mens naturlige systemer gjerne betraktes som kontinuerlige. Et diskret tilstandsrom gir en mer abstrakt, overordnet modell av et system, også kalt en kvalitativ/symbolsk modell. Det er derfor ikke uvanlig å modellere et komplekst system diskret selv om det naturlig kan betraktes som kontinuerlig. En modell som fremkommer ved å diskretisere tilstandsrommet kan også gi svar på spørsmål som den kontinuerlige modellen ikke kan gi svar på. Det er også vanlig å benytte en diskret modell som et overliggende rammeverk for kontinuerlige modeller for å oppnå hybride modeller.

Forskjellen mellom et kontinuerlig og et diskret tilstandsrom, er illustrert i figur 3.3 hvor tilstandstrajektorene for et en-dimensjonalt kontinuerlig system og et diskret system er vist.

Når vi benytter et diskret tilstandsrom forandrer tilstanden verdi (fra en diskret verdi til en annen) bare ved gitte diskrete tidspunkt ved at en *hendelse* inntreffer. En hendelse inntreffer instantant og resulterer i en instantan overgang fra en diskret tilstandsverdi til en annen. En hendelse kan f eks være at en betingelse blir oppfylt (avstanden til et objekt er mindre enn 10 km), en hendelse kan være knyttet til en aksjon (f eks at en operatør trykker på en knapp), eller rett og slett at en hendelse inntreffer, f eks en feil oppstår i en datamaskin. De mulige hendelsene som kan inntreffe og som resulterer i forandring av tilstandsverdi for et gitt diskret system erstatter pådragsvektoren, $\mathbf{u}(t)$ for en kontinuerlig beskrivelse. Hendelser er ikke synkronisert med en klokke (slik som tidsdrevne systemer er), men er resultat av asynkrone og mulige samtidige prosesser. F eks er hendelsen at en operatør trykker på en knapp helt uavhengig av

klokka i datamaskinen som knappen er koblet opp til. Diskrete systemer er altså *hendelsesdrevet* Vi kaller derfor modellen for *hendelses-diskret* i motsetning til tids-diskret. Den tids-diskrete modellen har svært lite til felles med en hendelses-diskret modell..



Figur 3-3 Tilstandstrajektor fortids kontinuerlig system og diskret hendelsessystem

Algebraiske ligninger benyttes for å beskrive kontinuerlig statiske systemoppførsel, mens differensial/differens ligninger benyttes for å beskrive oppførsel (dynamikk) for tids-kontinuerlige/-diskrete dynamiske beskrivelser. Altså:

Statisk:	$\mathbf{y}(t) = \mathbf{g}(\mathbf{u}(t))$		
	$\mathbf{y}(k) = \mathbf{g}(\mathbf{u}(k))$		
Dynamisk:	$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t))$	$\mathbf{x}(0) = \mathbf{x}_0$	(tilstandsligning)
	$\mathbf{y}(t) = \mathbf{g}(\mathbf{x}(t))$		(utgangs-/måleligning)
	$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k),\mathbf{u}(k))$	$\mathbf{x}(0) = \mathbf{x}_0$	
	$\mathbf{y}(k) = \mathbf{g}(\mathbf{x}(k))$		

Hvis vi benytter lineære modeller har vi tilsvarende (tar bare med de tids-diskrete variantene):

Statisk:	$\mathbf{y}(k) = D\mathbf{x}(k)$	
Dynamisk:	$\mathbf{x}(k+1)=\Phi\mathbf{x}(k)+\Delta\mathbf{u}(k)$	$\mathbf{x}(0) = \mathbf{x}_0$
	$\mathbf{y}(k) = D\mathbf{x}(k)$	

Her er D, Φ og Δ matriser.

Mens vi har et velutviklet apparat for å modellere og analysere tids-kontinuerlige og tids-diskrete beskrivelser, mangler vi dette for hendelses-diskrete dynamiske systemer. Vi mangler altså en analog til differensialligningen for modellering av dynamikk av hendelses-diskrete systemer. Det å utvikle «tilsvarende» for denne typen systemer er i dag et aktivt og viktig

forskningsområde.

Køteori er et eksempel på sammenhengen mellom hendelses-drevne og tids-drevne systemer. Teoretisk kan man benytte differensiallikninger for å analysere køsystemer, men pga kompleksiteten i de virkelige systemene man ønsker å modellere må man vanligvis utvikle et dataprogram som simulerer systemet. Ifm utviklingen av programmet benyttes typisk simuleringsspråk som er spesielt tilpasset simulering av hendelses-diskrete modeller. Disse modellene blir en type algoritme eller regelbasert beskrivelse som kan simuleres. Det er viktig å huske på at hendelses-diskrete systemer også beskriver et dynamisk system på samme måte som et tidsdrevet system.

Av de modelleringsteknikkene som benyttes for hendelses-diskrete beskrivelser kan det skilles mellom de som inkluderer tid og de som ikke gjør det. De som ikke inkluderer tid kalles *logiske modeller*, da de kan benyttes for å studere sekvenser av hendelser og tilstandsoverganger, altså spørsmål knyttet til den logiske oppførselen av systemet. Tidsmodellene inkluderer også tidspunktene når hendelser og tilstandsoverganger inntreffer og inkluderer nok informasjon til å rekonstruere trajektoren i figur 3.3.

De viktigste modelleringsteknikkene er:

Logiske:
- Tilstandsmaskiner, se f eks Hopcroft og Ullman, (1979)
- Petri nett, se f eks Peterson (1981)
- Kommuniserende sekvensielle prosesser, se Hoare (1985)

Teknikker som inkluderer tid:
- Tilstandsmaskiner utvidet med tid
- Petrinett utvidet med tid
- Diode algebra, også kalt min-max algebra
- Markovkjeder

Her skal vi begrense oss til modeller som baserer seg på bruk av tilstandsmaskin og utvidelser av denne for at den skal kunne benyttes for å modellere komplekse systemer. Dette omfatter hierarkiske og kommuniserende tilstandsmaskiner, Petrinett og Markovkjeder. Hvis vi benytter (endelig) tilstandsmaskin-formalismen (tilstandsmaskin med utgang) som en hendelses-diskret modell, består modellen av syv komponenter $(E, X, Y, \Gamma, \delta, \lambda, x_0)$ hvor (se appendiks A):

> E er inngangsalfabetet, dvs en mengde inngangssymboler som representerer hendelsene som kan påtrykkes systemet; hendelsesmengden $\{e_1, e_2, e_3, ...e_m\}$. Inngangsalfabetet tilsvarer pådragsvektoren for kontinuerlige systemer.
>
> X er en endelig mengde med tilstander, $\{x_1, x_2, x_3, ...x_n\}$ som tilsvarer

utfallsrommet for tilstandsvektoren for kontinuerlige systemer.

Y er utgangsalfabetet, dvs en mengde utgangssymboler som representerer hendelsene som er resultat av de påtrykte hendelsene og tilstanden, hendelsesmengden $\{y_1, y_2, y_3, ...y_l\}$. Utgangsalfabetet tilsvarer utfallsrommet til utgangs/målevektoren for kontinuerlige systemer.

Et statisk hendelsesdiskret system beskrives vha

$$Y = g(U) \tag{3.1}$$

hvor

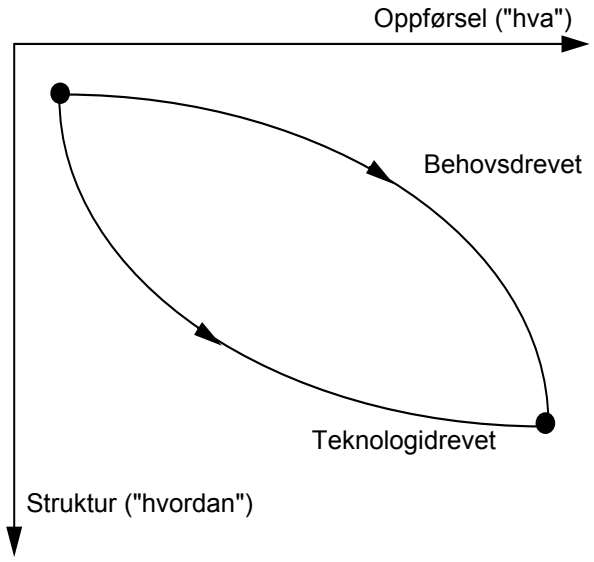
- U er mengden av inngangshendelser
- Y er mengden av utgangshendelser og
- g er en funksjon: $U \rightarrow Y$.

Beskrivelse av oppførsel er det som er vanskelig ved en systembeskrivelse. Som Bræk og Haugen (1993) påpeker, må vi kunne uttrykke dynamisk oppførsel på en endelig statisk form. For kontinuerlige beskrivelser har vi en velutviklet matematisk basis, mens denne er mye svakere for diskrete og hybride beskrivelser.

Allokering
Selv om et systems struktur og oppførsel kan betraktes som to uavhengige eller ortogonale dimensjoner av en systembeskrivelse, et det nødvendigvis en relasjon mellom dem. Denne relasjonene kalles *allokering.* Oppførsel allokeres til systemets struktur, eller systemets struktur blir allokert oppførsel. Dette gjøres ved at oppførsel blir allokert til komponenter. Komponentenes oppførsel og systemets konnektivitet (se nedenfor) blir definert ved allokeringen. Allokeringen av funksjoner til komponenter er selvfølgelig ikke entydig. Det er mange mulige valg av hvilke deler av et systems oppførsel som skal utføres av visse komponenter. Å oppnå en allokering som oppfyller visse effektivitetskriterier er en av de viktigste aktivitetene under systemutvikling, og utvikling av MMS er intet unntak.

3.1.4 Utvikling av beskrivelser/modeller i en systemutvikling

Ideelt sett kan vi betrakte en systemutvikling sett fra et modelleringssynspunkt som illustrert i figur 3.4. De to aksene representerer økende detaljgrad av hhv oppførsel og struktur. Når vi starter opp har vi en viss kunnskap om både hvilken struktur og oppførsel systemet må ha. Det videre arbeidet består i å detaljere og videreutvikle disse modellene til vi ender opp med de beskrivelse/modellene som danner utgangspunkt for produksjon. Det vil være mange veier mellom disse to punktene i struktur-oppførselsplanet som representerer forskjellige måter å angripe systemutviklingen på. Man kunne tenke seg å avklare alle forholdene rundt system-strukturen før man begynte å detaljere oppførselen eller motsatt at all oppførsel ble avklart før strukturen ble sett på. Begge disse fremgangsmåtene er urealistiske (selvom den siste har blitt fremhevet som en farbar vei av mange innenfor programvareteknikk). Både oppførsel og struktur må utvikles i parallell og mer realistiske trajektorer er da de som er vist i figuren.



Figur 3-4 Systemutviklingstrajektorer

Avhengig om det er systemstrukturen som «driver» systemutviklingen eller om det er systemoppførselen, kaller vi systemutviklingen for teknologi- eller behovsdrevet. Det bør tilstrebes en behovsdrevet utviklingstrajektor.

En angrepsmåte ved en systemutvikling vil først være å avgrense systemet som skal utvikles fra sin omgivelse. Altså å definere systemgrensen. Deretter vil det være naturlig å beskrive systemets oppførsel mot omgivelsene samt å avklare systemets grensesnitt til omgivelsene.

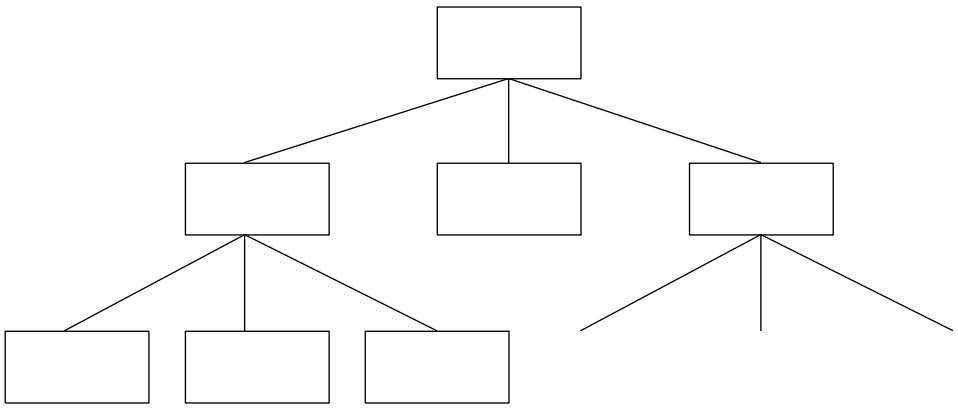
Deretter *dekomponeres* (kalles også partisjonering) oppførsel og struktur. For systemets struktur betyr dekomponering å etablere lavere nivås subsystemer og komponenter. For systemets oppførsel betyr dekomponenering å dele opp oppførselen i mindre blokker (også kalt moduler) som tilsammen gir den resulterende oppførselen på nivået over, men beskrevet i større detalj. Det er viktig at oppførselen til blokkene på de lavere nivåene også er dynamiske systemer slik at tilstander bibeholdes under dekomponeringen. For hvert nytt nivå under dekomponeringen knyttes strukturen og oppførsel sammen ved at oppførsel allokeres til de subsystemene og komponentene som er definert. Ut fra denne allokeringen og hvordan blokkene av oppførsel er avhengig av hverandre, blir sammenkoblingen mellom subsystemene/komponentene bestemt. Dvs at *grensesnittene*, *kanalene* og *flyt* (informasjon, masse, energi) over grensesnittet/på kanalene blir bestemt gjennom allokeringsprosessen.

Denne prosessen med dekomponering av oppførsel og struktur, og allokering gjentas for hvert nytt nivå ned til et detaljeringsnivå som man er fornøyd med. Det bør nødvendigvis ikke være samme antall nivåer for oppførsel og struktur.

Den systemtekniske angrepsmåten er altså å starte med systemet som en enhet for deretter å dekomponere og analysere det. Den motsatte av dekomponering/partisjonering kalles

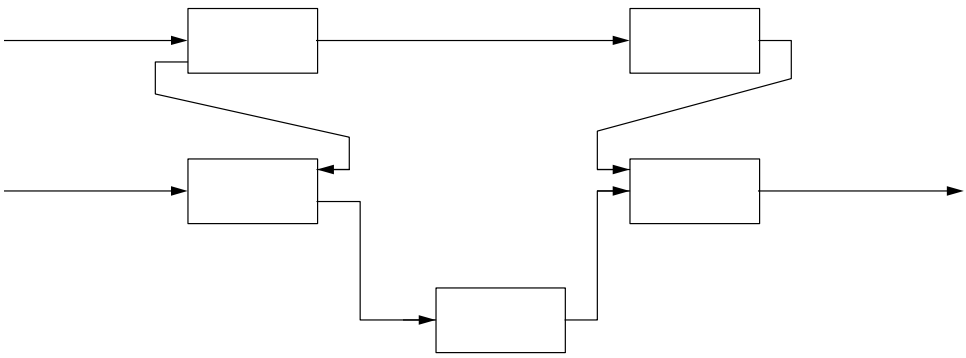
aggregering. Legg merke til at dekomponering/aggregering er forskjellig fra *generalisering/-spesialisering* som også benyttes når systemutviklingen gjøres objekt-orientert. Spesialisering/generalisering beskriver (u)likheter mellom typer ved at egenskaper som gjelder flere typer bare beskrives én gang.

Gjennom dekomponeringsprosessen oppstår det to hierarkier, ett for systemstruktur og ett for systemoppførselen. Hierarkiene vises gjerne grafisk som opp-ned trær, hvor noden i de to hierarkiene enten er subsysten/komponent eller blokker av oppførsel. Roten i treet er enten systemet selv eller systemets funksjon avhengig av om det er struktur eller oppførsel treet viser. Dette er vist i figur 3.5.



Figur 3-5 Systemhierarkier

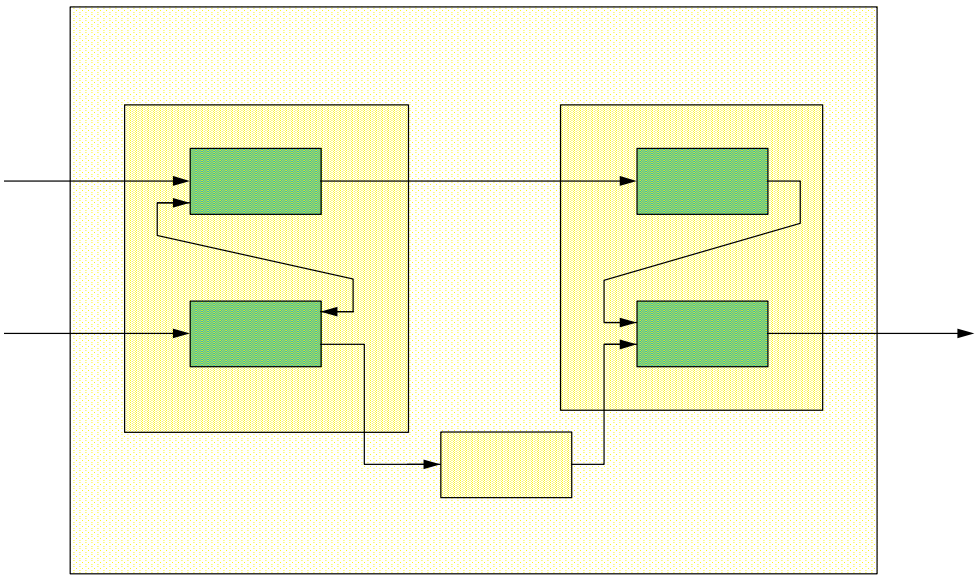
Gjennom dekomponering av oppførsel og allokering av oppførsel til systemstrukturen, bestemmes sammenknytning av subsystemer/komponenter og sammenknytningen av blokker av oppførsel (dvs innganger og utganger mellom blokkene). Denne sammenknytningen kalles for *konnektivitet* og beskriver for strukturen hvordan subsystemer/komponenter er knyttet sammen og om retningene på flytene på kanalene (på et gitt nivå i hierarkiet). Konnektiviteten for oppførselen beskriver hvordan innganger og utganger mellom blokkene av oppførsel er knyttet sammen (på et gitt nivå).



Figur 3-6 Systemkonnektivitet

Konnektivitet blir gjerne vist grafisk vha rettede grafer, f eks som et blokkdiagram (for struktur eller oppførsel), dataflytdiagram eller tilstandskart (for oppførsel) hvor nodene i grafen er blokker av oppførsel eller subsystemer/komponenter. Se figur 3.6.

Det er også vanlig å kombinere de to grafiske presentasjonene. Det kan gjøres på flere måter, f eks ved at flere nivåer av hierarkiet vises i et blokkdiagram som antydet på figur 3.7. En annen mulighet er å kombinere struktur og oppførsel i samme diagram, ved at f eks et diagram av konnektiviet av systemets oppførsel vises sammen med et blokkdiagram av systemstrukturen. Diagrammer som kombinerer både struktur og oppførsel uten å gjøre diagrammet for komplekst er spesielt nyttige.



Figur 3-7 Sammensatt diagram

Legg merke til at f eks et blokkdiagram ikke viser den tidsvarierende delen av oppførselen. Den viser bare hvordan inn- og utganger av blokker av oppførsel er knyttet sammen, dvs hvordan flyten er mellom blokkene. Det og bare ta med visse sider av en beskrivelse/modell kalles *projeksjon* og er en viktig måte å håndtere komplekse beskrivelser på. Hvis alle sidene av en beskrivelse skulle tas med i én enkelt beskrivelse ville den kunne bli vanskelig å forstå for utviklerne. Et bedre alternativ kan da være å benytte flere projeksjoner som tilsammen gir den totale beskrivelsen. Et eksempel på bruk av projeksjoner er bruk av de tre projeksjonene objektbeskrivelse, funksjonsbeskrivelse og dynamikkbeskrivelse innenfor objektorienterte systemutviklingsmetoder. Dette beskrives nærmere i kapittel 4.2.1.

3.2 Operatørmodellering

Operatørens forståelse/kunnskap om et system kalles ofte for hans/hennes *interne representasjon* eller *interne modell*. Operatørens mentale modell er også et begrep som ofte nyttes. Den interne modellen kan omfatte:

- Oppgaver og prosedyrer som skal utføres
- Egenskaper til systemet (struktur, oppførsel, osv.)

Operatøren utvikler en intern modell ved utdanning, trening og praksis. *En operatørmodell vil være en modell av slike interne modeller.* En viktig forutsetning ved operatørmodellering er at operatøren er godt trenet, dvs at han har en god intern modell av systemet. Operatørmodellering er minst vanskelig og har størst gyldighet dersom:

- Man forutsetter relativt strenge krav til tidsrespons fra operatør
- Oppgavene er klart definert

Når man skal modellere et MMS er det en fordel dersom vi kan beskrive operatør og maskin i et enhetlig modellspråk. Operatøren vil derfor ofte bli beskrevet i mekanistiske modeller.

Modellkategorier. Vi kan trekke et overordnet skille mellom ulike operatørmodeller avhengig av om de:

- modellerer oppførsel eller kun ytelse
- er kvalitative eller kvantitative
- er normative, deskriptive eller en blanding av disse

En *ytelsesmodell* er en modell av hvor godt en oppgave utføres. Dette kalles for en «black-box» modell, dvs vi modellerer ikke hva som egentlig skjer når oppgaven utføres. En *oppførsels-modell* derimot, er i tillegg en modell av handlingsmønsteret (riktignok på et begrenset detaljeringsnivå) som fører til et gitt resultat. Dette kalles en «white-box» modell, dvs det er synlig hva som skjer. Man bør merke seg at man altså får ytelse fra en oppførselsmodell, men ikke oppførsel fra en ytelsesmodell.

De modellene vi skal diskutere av operatøroppgaver er ytelsesmodeller og gir seg altså ikke ut for å beskrive hvordan operatøren egentlig utfører en oppgave, bare hvor godt den blir utført. Når det gjelder operatøroppførsel på et mer overordnet nivå vil vi derimot nytte oppførselsmodeller.

Kvantitative modeller gir resultater i form av tall og størrelser, mens kvalitative modeller gir ulike typer av beskrivelser, f eks sammenhenger ved bruk av skjemaer, tegninger, regler, osv.

Normative modeller er modeller av hvordan en oppgave ideelt sett bør utføres dersom man nytter løsningsmetoder fra f eks matematisk logikk (f eks setningskalkyle), sannsynlighetsteori (f eks Bayes regel) og fysikk (f eks Newtons lover). En normativ modell krever at både problemet og optimaliseringskriteriet for valg av løsning er veldefinert. Kontrollerte forsøk har vist at menneskers behandling av informasjon ofte avviker sterkt fra den normative modell (Anderson, 1990, Essens *et al*., 1994, Enge *et al*., 1984). F eks vil mennesker ikke veie usikker informasjon iht Bayes regel. Operatøren er som regel for konservativ, dvs han er ikke villig til å endre estimatet i tilstrekkelig grad ut fra nye data. Operatøren vil typisk også bruke apriori-

informasjon på en måte som avviker fra det normative, f eks ved *ikke* å ta hensyn til den. Man skulle tro at opplæring og utdannelse innenfor fagfeltene som er nevnt over vil ha stor betydning. En rekke forsøk har imidlertid vist at dette har liten betydning dersom selve formuleringen av problemet er ukjent (Anderson, 1990) eller dersom man kun benytter intuisjon (f eks under tidspress). Dette sier noe om at også representasjonen av informasjonen er viktig.

Man har hatt en viss suksess med å beskrive hvordan operatører faktisk utfører enkelte typer oppgaver. Dette kalles en *deskriptiv modell*. Ofte tar man utgangspunkt i en normativ modell og endrer denne til en såkalt *normativ-deskriptiv modell*. F eks kan man modellere ekstra støy ifm menneskelig persepsjon og begrenset hukommelse ifm informasjons-behandlingen. Så ser man på hva den normative modellen predikterer gitt disse nye begrensningene. En normativ-deskriptiv modell er en kombinasjon av hvordan vi mener at en operatør "burde" oppføre seg og hvordan han/hun faktisk oppfører seg. Den normative modellen kan benyttes både i planleggingen av forsøk og når disse evalueres. Avvik mellom målinger og det den normative modellen predikterer gir en indikasjon på hvilke kognitive begrensninger (mht behandling av tilgjengelig informasjon) som operatøren(e) har i det gitte forsøket (systemet).

Operatørens tildels lave ytelse i visse forsøkssituasjoner kan gi et inntrykk av at det er mye å tjene på å automatisere med utgangspunkt i den normative løsningen. Dette er ikke nødvendigvis riktig. I en virkelig situasjon har man ofte manglende og ufullstendig kunnskap om kvaliteten av informasjonen og hva som er kriteriet for valg av løsning. Videre, dersom operatøren selv er informasjonskilden må han formalisere informasjonen før en datamaskin kan behandle denne. Dermed er forutsetningene for den normative løsningen ofte ikke tilstede og operatørens ytelse kan da faktisk være bedre (Skare, 1990, Nordø, 1983). Lav operatørytelse relativt til normativ ytelse indikerer imidlertid en potensiell mulighet til forbedring av MMK (Wohl *et al*., 1989).

ANALOGI	REPRESENTASJON
Ideell observatør	Estimeringsteori
Servomekanisme	Reguleringsteori
Tidsdelt datamaskin	Køteori
Vag resonering	Fuzzy logikk
Kunnskapsbasert system	Regelbaserte modeller
Mønstergjenkjenner	Statistiske modeller

Tabell 3.1 Alternative representasjoner for modellering av operatør (etter Rouse,1991)

Operatøranalogier. Etter at operatøroppgavene er detaljert i tilstrekkelig grad, kan man prøve å finne den beste representasjonen for hver av disse deloppgavene. Bruk av analogier er et vanlig utgangspunkt ved utvikling av modeller, så også for operatørmodeller. En oversikt over viktige analogier og deres respektive representasjon er gitt i tabell 3.1. Vi vil komme tilbake til disse representasjonene.

3.3 Et rammeverk for modellering av operatøroppførsel

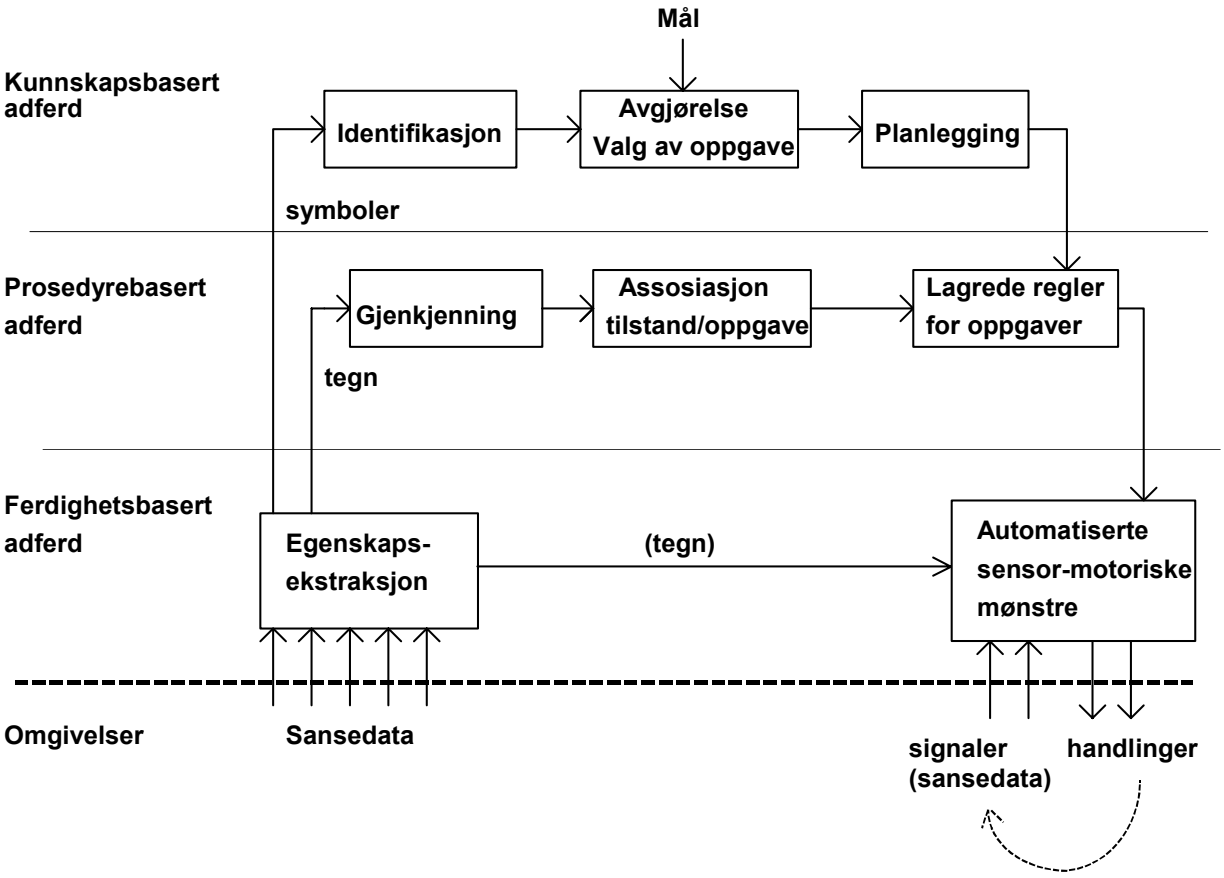
I dette kapitlet skal vi diskutere et rammeverk for modellering av operatøroppførsel (adferd) i teknologiske systemer utviklet av Rasmussen (1983, 1986). Arbeidene hans er mye referert til og godt kjent i systemteknikk/ingeniør-miljøet mens det kan virke som om de er mindre diskutert innenfor det psykologiske fagmiljøet. Rammeverket gir en kvalitativ beskrivelse av og deler inn operatøroppførsel i tre hovedkategorier.

Beskrivelsene av de ulike kategoriene og forskjellene mellom dem danner etter vår mening et godt rammeverk for modellering av operatøren i et MMS. Fra et ingeniørmesig synspunkt er rammeverket av særlig interesse fordi det beskriver koblinger til kjente oppførsels- og ytelsesmodeller som brukes av ingeniører for tekniske systemer. Videre kan modellen brukes som et overordnet hjelpemiddel ved konstruksjon og evaluering av MMK.

De tre hovednivåene av operatøroppførsel i et MMS er vist i figur 3.8. Figuren viser hvordan sansedata behandles på de ulike nivåene og hvordan dette til slutt resulterer i handlinger fra operatøren. Tabell 3.2 gir eksempler og oppsummerer viktige karakteristikker for de ulike nivåene.

Ferdighetsbasert (FB) oppførsel er direkte kobling fra sansedata til motoriske bevegelser. FB oppførsel er knyttet til «automatiserte sensor-motoriske mønstre» i figur 3.8. Dette er oppførsel som gjennom lengre tids trening er så godt innøvd at den skjer automatisk (utenfor bevisst kontroll). Oppførselen er optimalisert mhp ressursforbruk. Underbevisst styring genererer de resulterende bevegelsene og denne kan derfor ikke forklares eksplisitt. Bevisstheten vil gripe inn dersom modi av bevegelse endres (f eks gå forsiktig! Her er det glatt!). En modell av operatørens interne representasjon vil være en dynamisk tilstandsrommodell som «simuleres» og inndata, kalt *signaler*, er kontinuerlige tid-rom variable, f eks posisjonen til et objekt som funksjon av tida som benyttes for å oppdatere operatørens interne modell. Operatørene er som sagt ikke nødvendigvis bevisst hvilke signaler som prosesseres: "Man looks rather than sees".

Motoriske bevegelser er i noen grad kontrollert vha *tilbakekobling*. Vi skal senere (kapittel 3.4 og appendiks B) beskrive nytteverdien og virkningen av tilbakekobling i systemer. Et eksempel på tilbakekobling i FB oppførsel kan være det å gripe etter et objekt med hånda: observer avvik i posisjon av hånd relativt til et objekt, beregn stimuli til muskler i hånd, utfør bevegelse og gjenta inntil avvik er null. I kompliserte sekvenser av bevegelser som må utføres hurtig, som f eks håndskrivning, musikk eller sport, er nytten av tilbakekobling sterkt redusert pga forsinkelsene i operatørens sensoriske apparat, og utførelsen er basert på regenerering av tidligere (vellykkete) bevegelsesmønstre. Oppførselen kan i reguleringstekniske termer karakteriseres som *foroverkoblet* basert på den interne dynamiske modellen.



Figur 3-8 Rasmussens rammeverk av operatøroppførsel. Etter (Rasmussen, 1983)

Type oppførsel	**Informasjon - og informasjonsbehandling**	**Karakterisering av informasjonsbehandling**	**Eksempler**
Ferdighetsbasert (FB)	*Signaler* (tid/rom variable) som inndata i multi-variabelt reguleringssystem	Parallell med høy kapasitet, ubevisst (automatisert oppførsel). Rask og predikterbar respons	Manuell styring, signal deteksjon, estimering
Prosedyrebasert (PB)	*Tegn* (relatert til tilstand/situasjon i omgivelser) som utløser lærte prosedyrer (regler)	Sekvensiell og bevisst. Målrettet (ofte implisitt knyttet til situasjon). Tidsrespons avhengig av situasjon/oppgave, men ofte godt predikterbar	Prosedyre-oppgaver, overvåkning
Kunnskaps-basert (KB)	*Symboler* (variable) knyttet til intern modell som nyttes til problem-løsning (funksjonell resonering)	Målrettet, sekvensiell og bevisst. Sen respons, lite predikterbar	Planlegging, beslutninger

Tabell 3.2 Typer av oppførsel

Prosedyrebasert (PB) oppførsel er kobling fra gjenkjente mønstre til aksjoner. Man tenker seg da at operatøren er en mønstergjenkjenner, dvs at han først ekstraherer bestemte egenskaper i sansedataene og gjenkjenner visse mønstre i disse, såkalte *tegn*. Disse er knyttet til en tilstand eller situasjon i omgivelsene som fordrer at visse oppgaver utføres. Assosiasjonen mellom tilstand/oppgave er rask og ikke særlig krevende. PB er målrettet, men målet er ofte implisitt og knyttet til selve situasjonen. Tegn brukes til bevisst å velge eller modifisere regler som kontrollerer sekvensering av FB aksjoner. Merk at tegn ikke prosesseres av operatøren direkte, men at de aktiverer lagrede prosedyrer. PB oppførsel kalles ofte for *regelbasert oppførsel* og aktuelle modellrepresentasjoner er tilstandsdiagrammer, fuzzy mengder, nettverksmodeller og ekspertsystemer. PB oppførsel kan bli eksplisitt forklart. Denne typen oppførsel vil selvsagt også utvikle seg gjennom praktisk trening ved at man danner nye regler eller modifiserer regler på bakgrunn av erfaring.

Studier har vist at operatørens kjennskap til standard prosedyrer (dvs regler) bidrar mest til en god prediksjon av ytelsen i svært mange jobbsituasjoner. Kunnskap om prinsipper og teori har kun i mindre grad innflytelse på ytelsen (Rouse, 1991). Modellering av PB oppførsel er derfor svært viktig.

Kunnskapsbasert (KB) oppførsel er karakterisert ved bevisst analytisk resonnering/vurdering med funksjonell og strukturell prosesskunnskap representert i operatørens interne modell. KB oppførsel kalles derfor også for *modellbasert oppførsel.* KB oppførsel er nødvendig i ukjente/uforutsette situasjoner eller kompliserte situasjoner der man ikke har veletablerte regler å falle tilbake på. Som for PB oppførsel, ekstraheres også her visse egenskaper i sansedataene, *symboler*, som er relatert til operatørens interne modell. Systemets tilstand identifiseres på basis av symbolene. Valget av oppgave er basert på aktuelt mål, tilhørende analyse av avvik fra ønsket tilstand og hvordan dette avviket kan reduseres eller elimineres. Deretter planlegges gjennomføringen med utgangspunkt i relevant PB oppførsel. Vi skal senere illustrere denne sekvensen vha noen figurer.

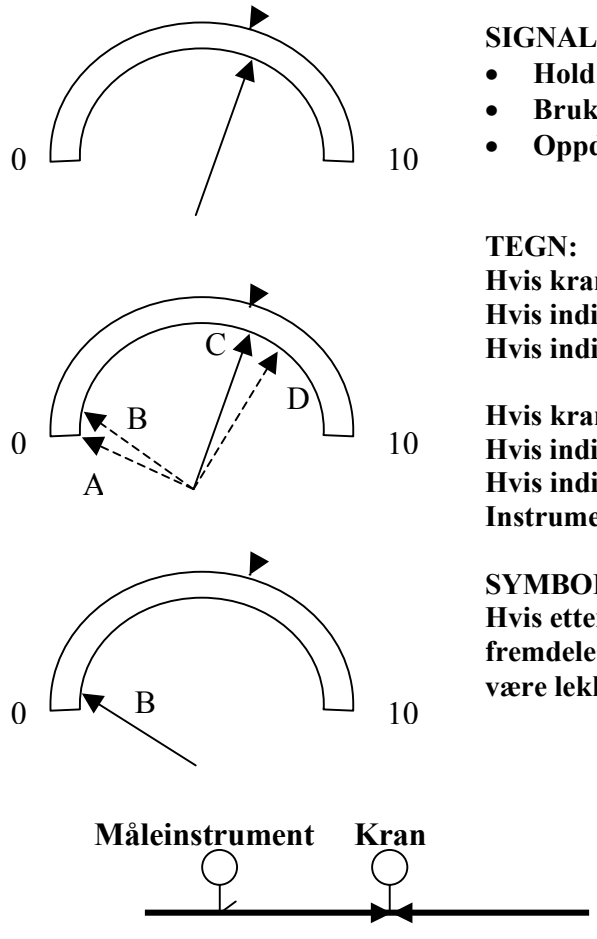
Forskjellen mellom signaler, tegn og symboler er i stor grad relatert til hvordan de tolkes og brukes av operatøren og har ikke noe med hvordan data blir presentert for operatøren. Dette er godt eksemplifisert i figur 3.9. Den fysiske indikatoren på kontrollpanelet kan tolkes både som et signal, tegn og symbol. Tolket som signal kreves det ingen systemkunnskap. Tolket som tegn kreves det strengt tatt heller ingen systemkunnskap, men tegn vil ofte være assosiert med tilstander i systemet (kran åpen) og oppgavene vil peke på funksjoner (juster strømning, kalibrer instrument). Tegn brukes imidlertid ikke til funksjonell resonnering. Tolket som symbol vil f eks indikasjon B (gitt at kranen er stengt og kalibrering utført) kunne gi følgende resonnement: Bevaring av masse (væske) innenfor systemet gir at $Y = B$. Hvis kranen er stengt, så må $Y = 0$ og instrumentets indikasjon skal også være 0. Hvis instrumentet likevel viser $B > 0$ har vi følgende muligheter:

1. Det er en feil med instrumentet. Indikasjonen skulle vært null.
2. Det er en feil med kranen. Kranen er likevel åpen, $Y = B > 0$.
3. Det er en lekkasje ut av systemet, $Y \neq B$.

Forskjellen mellom tegn og symboler kan enkelt uttrykkes slik: «You act on signs, but you understand symbols».

Det er ingen skarp grense verken mellom FB/PB eller PB/KB. Etter hvert som operatøren får praktisk erfaring vil oppførsel kunne endre kategori. En situasjon som opprinnelig er ukjent krever KB oppførsel. Dersom den utføres tilstrekkelig ofte vil dette kunne resultere i at regler etableres og brukes, dvs PB oppførsel. Tilsvarende vil en overlæring av visse typer PB oppførsel føre til at denne automatiseres som FB oppførsel. Bilkjøring er et typisk eksempel på at KB/PB oppførsel kan endres til FB oppførsel. Mennesker vil prøve å holde informasjonsbehandlingen på det lavest mulige nivå som sikrer en akseptabel gjennomføring av oppgaven.

Anderson (1983) benytter tilsvarende tredeling av oppførsel og kaller nivåene: Kognitiv tenkning (KB), assosiativ tenkning (PB) og autonom oppførsel (FB).



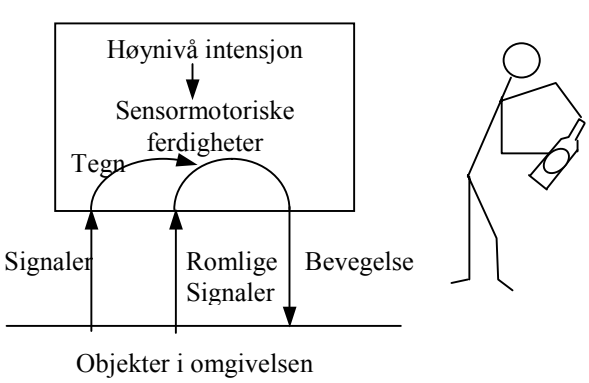
SIGNAL:
- **Hold på settpunkt**
- **Bruk avvik som feilsignal**
- **Oppdater kontinuerlig**

TEGN:
Hvis kran åpen, og
Hvis indikasjon C: OK
Hvis indikasjon D: juster strømning

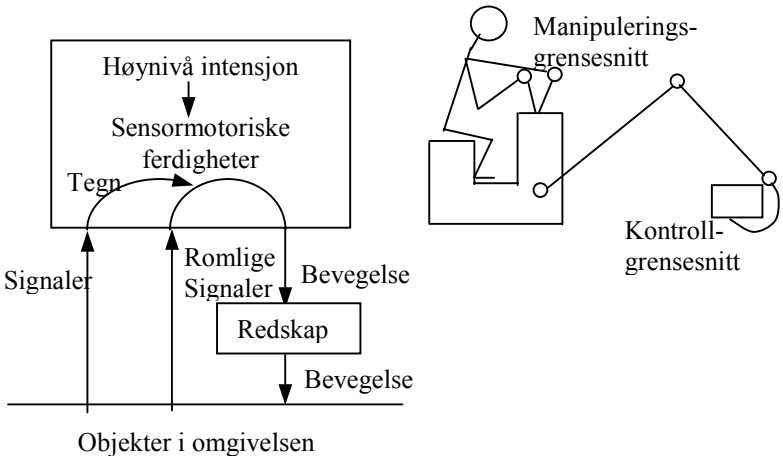
Hvis kran stengt, og
Hvis indikasjon A: OK
Hvis indikasjon B: kalibrering av Instrument

SYMBOL:
Hvis etter kalibrering, indikasjon er fremdeles B, tenk funksjonelt (kan være lekkasje)

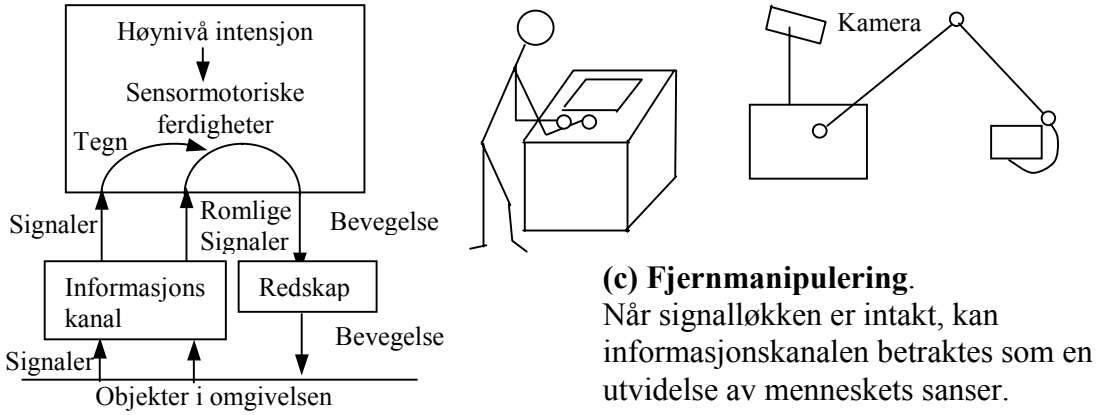
Figur 3-9 Tolkning av informasjon. Etter (Rasmussen, 1983)



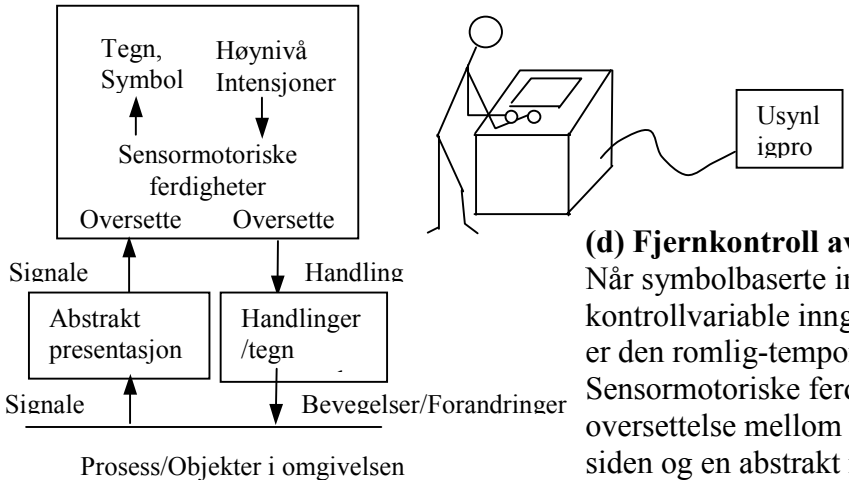
(a) Direktemanipulering.
Det eksiterer en ubrutt romlig-temporal signalsløyfe.



(b) Indirekte manipulering.
Manipuleringsgrensesnitett er ikke det samme som kontroll-grensesnittet. Redskapet kan betraktes som en utvidelse av menneskets motoriske svstem



(c) Fjernmanipulering.
Når signalløkken er intakt, kan informasjonskanalen betraktes som en utvidelse av menneskets sanser.



(d) Fjernkontroll av usynlig prosess.
Når symbolbaserte indikatorer og diskrete kontrollvariable inngår i kontroll av en usynlig prosess, er den romlig-temporale signal-sløyfen brutt. Sensormotoriske ferdigheter fungerer da bare som oversettelse mellom signaler og handlinger på den ene siden og en abstrakt representasjon av prosessen og intensjoner på den andre.

Figur 3-10 Ulike nivåer av avstand mellom aktør og manipulert objekt. Etter Rasmussen (1983)

Når det gjelder informasjon fra operatøren til omgivelsen kan vi bruke tilsvarene inndeling som over:

- Direkte aksjoner på fysiske elementer i omgivelsene
- Indirekte aksjoner på fysiske elementer i omgivelsene
- Kodede meldinger

Kommunikasjon på KB nivå krever et intelligent system som forstår meningsinnholdet.

Figur 3.10 beskriver muligheten av å opprettholde operatørens tolkning og aksjoner i rom/tid ifm manipuleringsoppgaver etter hvert som operatøren fjernes mer og mer fra den fysiske prosessen. Figuren illustrerer et typisk trekk ved teknologisk utvikling, nemlig at det i stadig større grad kommer nye typer presentasjons- og betjeningsutstyr *mellom* operatøren og prosessen. Et annet utviklingstrekk er økende automatisering som fører til at mer av operatør-oppgavene knyttes til overvåkning, monitorering og det å ta beslutninger (Beevis *et al.*, 1992). Dvs man har en overgang fra FB til PB/KB oppførsel. Dette stiller større krav til kognitive aspekter ved operatøroppførsel, dvs kunnskap, oppfattelsesevne, hukommelse, forestillelsesevne, vurderingsevne og resoneringsevne. Teknikker for funksjons/oppgaveanalyse er egnet til å beskrive FB/PB oppførsel, men ikke KB oppførsel.

Mange av rutineoppgavene i dagens MMS er PB og disse kan som sagt beskrives forholdsvis godt. En rekke teknikker har blitt foreslått også for KB oppførsel, men disse har hatt begrenset suksess og ingen er egentlig blitt særlig anerkjent. Dette diskuteres nærmere i kapittel 4.3.

Kunstig virkelighet («Virtual Reality») innebærer et nytt trekk i den teknologiske utvikling som på sett og vis er motsatt av det vi har beskrevet over. Her nyttiggjøres menneskets egenskaper til tolkning av tid/rom og aksjoner i tid/rom som et alternativ til kodede meldinger i mer tradisjonelle grensesnitt. Dette skal vi komme tilbake til i kapittel 6.3.8.

Beskrivelse av en beslutningsprosess. Rasmussen (1986) gir en generell beskrivelse av stegene i en beslutningsprosess:

- *Deteksjon* av en hendelse/situasjon, som er en forutsetning for aksjon. Operatøren er i en årvåken tilstand.
- *Observasjon* av viktige data
- *Identifikasjon* av systemets *nåværende tilstand*
- *Tolkning* av mulige konsekvenser og mål
- *Evaluering* av mulige mål og valg av et av disse
- *Tolkning* av mål som resulterer i *valg av ønsket tilstand*
- *Definisjon* av operatøroppgave som fører frem til ønsket tilstand
- *Formulering* av prosedyre basert på valgt oppgave

- *Utførelse* (eksekvering) av prosedyren

Beslutningsprosessen er framstilt i figur 3.11 der en også har relatert de ulike stegene til de tre hovednivåene av operatøroppførsel. Operatøren vil ofte gjenkjenne situasjoner og dette vil kunne forenkle prosessen ved at mange av stegene utelates. En forenklet versjon av beslutningsprosessen er vist i figur 3.12.

Som vi tidligere var inne på vil KB oppførsel kreve ulike typer representasjoner av systemet og omgivelsene. En prøve- og feile-strategi som utføres i praksis er sjelden akseptabel og operatøren må derfor rent tankemessig simulere hva som vil skje gitt ulike handlinger. Dette krever at operatøren har en god forståelse av systemet og dets tilstand på mange ulike plan, både funksjonelt og strukturelt. Kunnskap om systemets oppførsel er nødvendig for å kunne evaluere konsekvenser av aksjoner og deres forplantning i systemet.

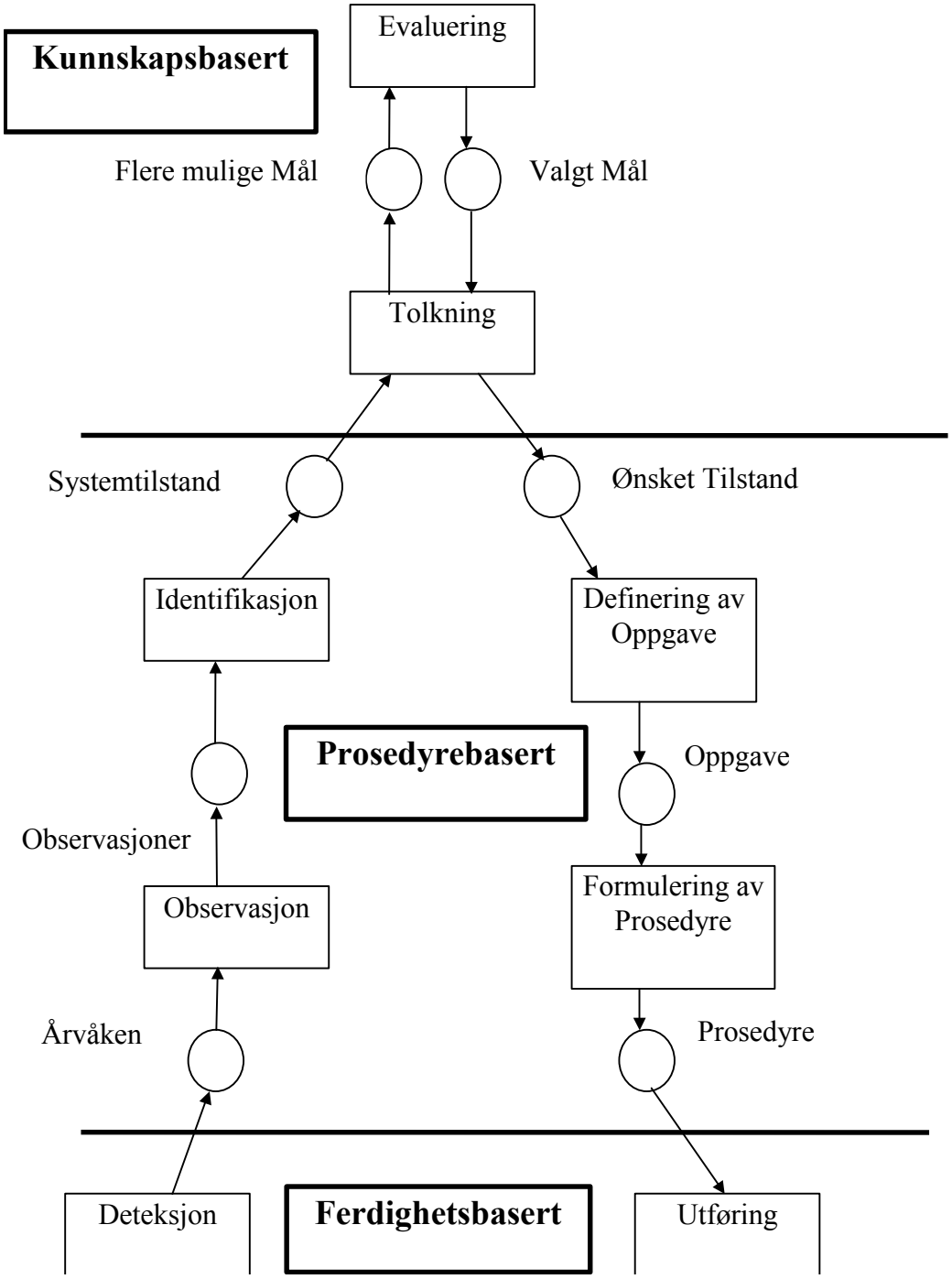
På tross av økende automatisering forventes det at operatøren skal være i stand til å gripe inn dersom det er nødvendig. Operatøren må derfor vite intensjonene til systemet og formålet med den automatiske styringen. Operatøren må ha en oppdatert intern modell. MMS og dets MMK må derfor sørge for at det er *mulig* for operatøren å ha en tilstrekkelig god og oppdatert forståelse av systemtilstanden. Hva som er tilstrekkelig vil primært avhenge av operatørens jobb og hans ansvar i forhold til systemmålsetningen. Som regel er det også påkrevet at han har tilgang til ulike beskrivelser og modeller av systemet og datastøtte for dette blir mer og mer alminnelig.

Rasmussen understreker betydningen av at operatørens interne modell er tilpasset den oppgaven han i øyeblikket skal utføre. Systemet bør bidra aktivt til en slik tilpasning ved representasjoner på forskjellige nivåer tilpasset operatørens oppgaver.

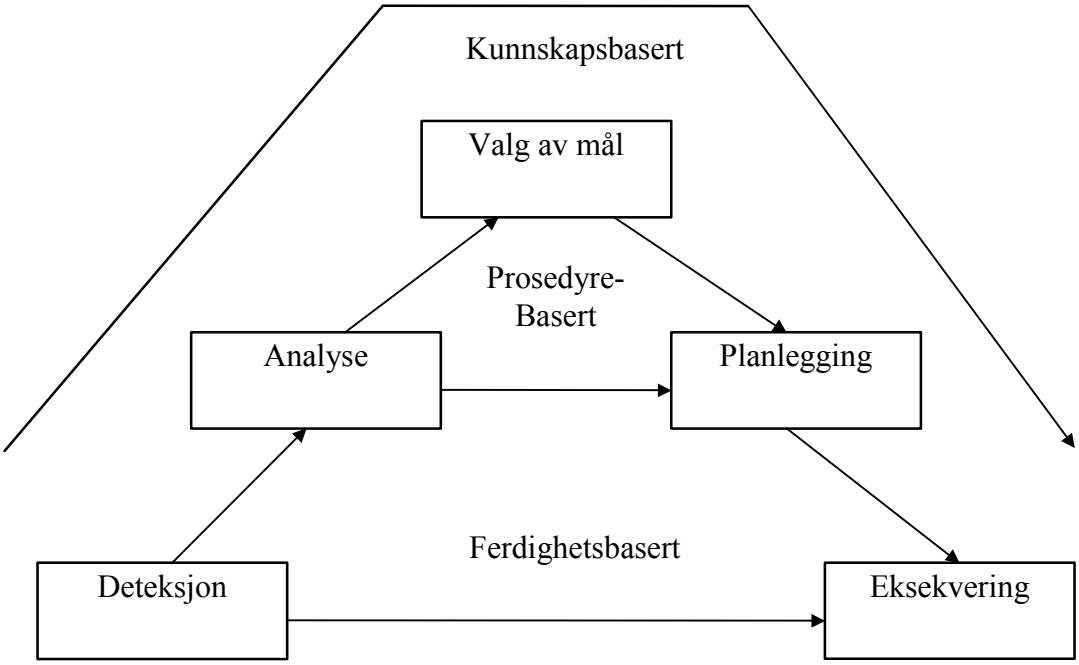
Grunnleggende typer av mekanismer for å tilpasse den interne modellen av systemet til en situasjon/oppgave er:

- Aggregering
- Abstraksjon
- Analogier (anvendelse av tidligere løsninger på lignende type av problem)

Rasmussen bruker betegnelsen «modelltransformasjoner».



Figur 3-11 Steg i en beslutningsprosess. Etter (Rasmussen, 1986)



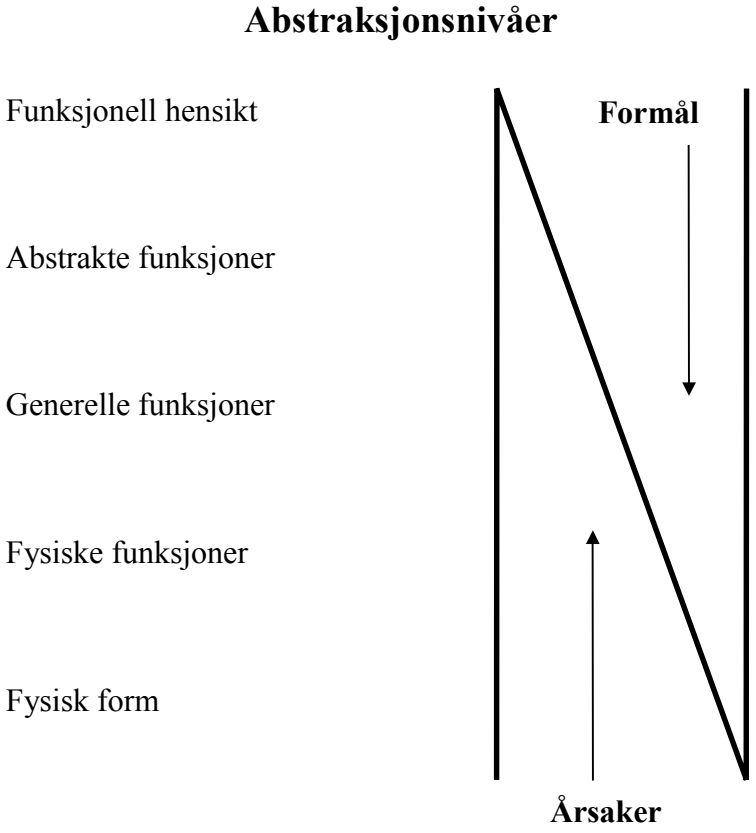
Figur 3-12 Forenklet beskrivelse av beslutningsprosessen. Etter (Rasmussen, 1986)

I (Rasmussen, 1986) identifiseres følgende typer av *abstraksjonshierarkier*:

- Klassehierarki: Spesialisering/generalisering som vi kjenner fra objekt-orientert modellering. Eksempel: Bil -> personbil -> Toyota Corolla.
- Oppførselshierarki: Kausale (årsak-effekt) relasjoner.
- Systemkomponenthierarki: Systemstruktur.
- Mål-middelhierarki: Funksjoner på et gitt nivå (HVA) styres på et høyere nivå av formålet med funksjonen og samvirke med andre funksjonen (HVORFOR). På nivået under beskrives realiseringen av funksjonen (HVORDAN). Eksempel: reguleringssløyfe -> reguleringsalgoritme -> program- og maskinvare.

Rasmussen hevder at man ifm utvikling av systemer i liten grad har skilt mellom abstraksjonsnivå (mål-middel hierarki) og aggregering (system-komponent hierarki). Den første dimensjonen er fra konkret til abstrakt der representasjonsform endres og den andre angir detaljeringsgraden i beskrivelsen gitt en representasjonsform. Han argumenterer videre for betydningen av å opprettholde dette skillet. Videre understrekes likheten mellom beskrivelser av systemer som benyttes under en systemutvikling og de interne modellene som operatøren benytter under drift.

En mer detaljert beskrivelse av mål-middelhierarkiet for representasjon av tekniske systemer er vist i figur 3.13.



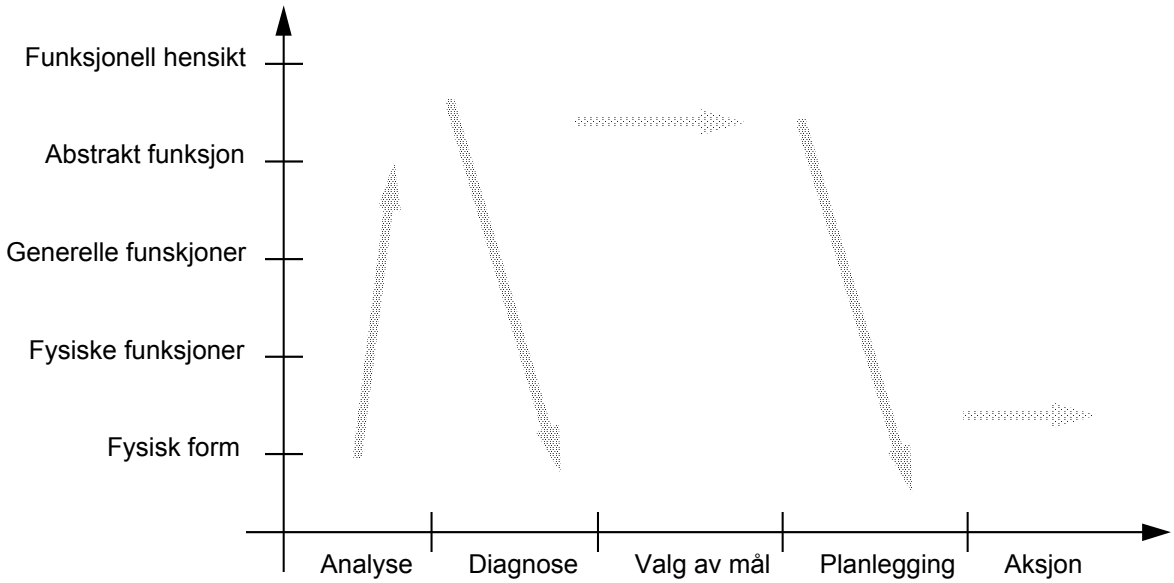
Figur 3-13 Mål-middel abstraksjonshierarki. Etter (Rasmussen, 1986)

En kort beskrivelse av de ulike nivåene, er:

- *Fysisk form* - fysisk form, plassering og struktur av systemets komponenter
- *Fysiske funksjoner* - de fysiske prosessene (mekaniske, elektriske, kjemiske) til systemet og dets komponenter
- *Generelle funksjoner* - inkluderer type av funksjonelle relasjoner *uavhengig* av fysisk realisering, f eks reguleringssløyfe
- *Abstrakte funksjoner* - overordnet oppførsel, generaliserte nettverk, f eks informasjonsflyt eller masse/energiflyt. Basert på hensikten med systemet
- *Funksjonell hensikt* - inkluderer beskrivelse av hensikt med systemet, dvs systemmålsetning og kriterier (effektivitetsmål). Beskriver koblinger (innganger/utganger) og oppførsel mot omgivelsene

Viktige operatøroppgaver i et MMS er koblet til korreksjon av feil. Feil kan oppstå i komponenter eller pga feilhandlinger utført av operatøren. Ifølge Rasmussen kan hendelser bare defineres som feil dersom disse refereres til ønsket tilstand, normal funksjonering eller systemhensikt/målsetning. *Årsaker til feilaktig funksjonering* skyldes typisk endringer i den fysiske/materielle verden, f eks kortslutning i en elektrisk komponent. Dvs at disse forklares nedenfra og opp i abstraksjonshierarkiet. *Grunn til riktig funksjonering* forklares imidlertid ovenifra og ned i hierarkiet. Denne forskjellen har Rasmussen observert ved verbal

protokollanalyse av diagnostisk søk i informasjonssystemer. Tilsvarende har Øgaard (1990) beskrevet feilkorreksjon i prosesskontroll vha abstraksjonshierarkiet og stegene i den generelle avgjørelsesprosessen som illustrert i figur 3.14.



Figur 3-14 Feilkorreksjon beskrevet vha mål-middelhierarkiet (Øgaard, 1990)

Figuren illustrerer bruken av hierarkiet. Operatøren vil bruke forskjellige nivåer avhengig av den aktuelle oppgaven. Dersom nivå skiftes, vil typen av beskrivelse endres. Informasjon om korrekt (forventet) oppførsel er gitt av nivået over og informasjon om ressurser og begrensninger av nivået under.

Hvilke konsekvenser får Rasmussens ideer og modeller for utforming av MMK? I hovedsak kan vi gi noen generelle retningslinjer:

- Baser presentasjonen på informasjon direkte tilpasset den oppgaven som skal utføres
- Velg representasjon av informasjon slik at operatøren ikke må transformere denne til en annen representasjon for å utføre den aktuelle oppgaven
- Informasjon og strukturer bør velges slik at en oppnår direkte aksjoner på tid-rom variable, f eks at operatøren bør kunne operere direkte på skjermbildet (jfr direkte manipulering).

I fagtidsskriftene finnes det en rekke eksempler på at man konkret har prøvd å bruke abstraksjonshierarkiet ifm utforming av grensesnitt, se f eks (Vicente & Rasmussen, 1992, Øgaard, 1990, Bisantz & Vicente, 1994 & Vicente, 1996, Treurniet *et al*, 1999 og Chalmers, 2001). Konstruksjon basert på disse prinsippene omtales som "ecological interface design", se Vincente & Rasmussen, 1992 og Vincente, 1999).

3.4 Modellering av ferdighetsbaserte oppgaver

Som tidligere nevnt er informasjonen som operatøren behandler ved ferdighetsbaserte oppgaver tid-rom-variable, kalt signaler. Basisen for modelleringen av denne type av oppgaver er derfor hentet fra reguleringsteknikken og den viktigste analogien er operatøren som en regulator. Nendenfor er det beskrevet noen operatøroppgaver som er ferdighetsbaserte. Disse er manuell regulering, monitorering av et kontinuerlig system og avlesning av et antall enkeltinstrumenter.

Det er utviklet matematiske modeller for en lang rekke andre operatøroppgaver enn det som blir gjennomgått her, og oversikter gis f eks i Rouse (1981), Sheridan og Ferrell (1974) og Skare (1990).

3.4.1 Manuell regulering

Med *manuell regulering* menes at en person ved sine sanser observerer ønsket verdi for tilstander i et gitt system samt målte verdier av de samme tilstander, enten separat eller i kombinasjon og hvor han vha betjeningsorganer minimaliserer feilen mellom disse.

Det er flere oppgaver som krever kontinuerlig manuell regulering, selv om flere av denne type oppgaver har blitt automatisert i de senere årene. Eksempler er:

- Fartøysstyring (båt, bil, fly, helikopter, sykkel)
- Følgeoppgaver

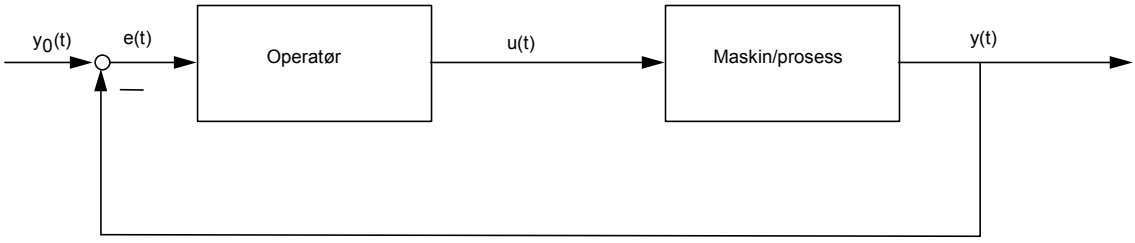
I tillegg er reguleringoppgaver benyttet mye i forbindelse med grunnforskning for forståelse av menneskelig motorikk og persepsjon.

Arbeidet med å modellere operatører som regulatorer hadde sitt utspring under den 2. verdenskrig sammen med den generelle fremveksten av reguleringsteknikk. Behovet stammet fra utvikling av våpenssystemer som styring av anti-luft kanon, bombekastere, osv. Dvs, typiske følgeoppgaver.

I de senere år har det ikke vært så mye fokus på dette feltet, mye pga økt automatisering og innføring av datateknologi. Operatørenes reguleringoppgaver er blitt mer overordnet slik at MMK-en i hovedsak er blitt symbolsk og baserer seg ikke lenger på kontinuerlige tid-rom variable. Imidlertid ser vi nå en dreining tilbake til en MMK basert på kontinuerlige tid-rom variable, litt i forbindelse med grafiske direkte manipulerende operatørgrensesnitt, men for fullt i forbindelse med kunstige virkeligheter. Konstruksjon av denne type operatørgrensesnitt vil dra nytte av en tilnærmelsesmåte som beskrevet her. Ytterligere eksempler hvor modeller av manuell regulering kan være nyttige er derfor:

- Kunstig virkeligheter
- Augmentert virkelighet
- Fjernstyring av farkoster og roboter, f eks undervannsfarkoster

Situasjonen som operatøren er i ved manuell regulering er gitt i figur 3.15. Operatørens oppgave er på basis av målinger fra maskinen/prosessen og en ønsket referanse å sørge for at avviket til den ønskede referansen blir så lite som mulig. Vi begrenser oss til monovariable systemer og antar at prosessen er lineær.



Figur 3-15 Manuell regulering

Vi kan først forsøke og etablere en modell av operatøren ut fra intuitive betraktninger. Denne utledningen er basert på Rouse (1981).

En første tilnærming vil naturlig være å anta at:

$$u(t) = K_1 e(t) \tag{3.1}$$

En slik modell tar imidlertid ikke hensyn til at operatøren har en viss reaksjonstid, som er i området 0,15-0,20 s. En mer realistisk modell vil derfor være:

$$u(t) = K_2 e(t-\tau) \tag{3.2}$$

Denne modellen tar imidlertid ikke hensyn til dynamisk respons (treghet) i menneskets motoriske system. En videre raffinering av modellen vil derfor være:

$$u(t) = k_1 u(t-dt) + k_2 e(t-\tau) \tag{3.3}$$

eller som transferfunksjonen:

$$h(s) = \frac{K_3 \, e^{-\tau s}}{1 + Ts} \tag{3.4}$$

Imidlertid er vi i stand til i en viss grad å kompensere for transportforsinkelse i det perseptuelle systemet og responstid i det motoriske systemet. Dette gjøres på to måter: Ved å øke forsterkningen og ved å estimere den tidsderiverte av avviket og benytte denne til å prediktere avviket til nåtidspunktet. Disse kompensasjonene kan inkluderes i modellen ved å utvide den til:

$$u(t) = c_1 u(t - dt) + c_2 e(t - \tau) + c_3 e(t - \tau - dt) \tag{3.5}$$

I tillegg til en modell av den prosessen som manuelt reguleres kan denne modellen benyttes. Problemet er at parametrene i modellen vil variere ut fra prosessmodellen. Med andre ord, operatøren er en *adaptiv* regulator, noe som ikke er overraskende. Operatøren er adaptiv i vanlig forstand ved at parametrene blir tilpasset en gitt prosess (læring), men også i den forstand at operatøren kan tilpasse seg forskjellige type prosesser. Ved å benytte denne modellen av operatøren, måtte vi derfor ha ett sett med modellparametre for hver type prosess. Disse modell-parametrene måtte også bestemmes eksperimentelt slik at dette ikke er en farbar vei. Det vi ønsker er en modell som ikke er så avhengige av den prosessen som styres. En slik modell ble utviklet på 60-tallet av McRuer og kollegaer, McRuer *et al*. (1965).

Ideen bak denne modellen er å anta at operatøren adapterer seg til prosessen slik at det oppnås god stabilitet og respons, dvs gode følgeegenskaper. Derfor vil en samlet modell av *både* operatør og prosess være mye mindre sensitive til hvilken prosess som operatøren regulerer/styrer. Modellen kalles *kryssmodellen* (på engelsk *crossover model*) og er gitt som:

$$y(t) = y(t - dt) + ce(t - \tau) \tag{3.6}$$

eller alternativt som sløyfetransferfunksjonen:

$$h_0(s) = h_{operatør}(s)h_{prosess}(s) = \frac{\omega_c e^{-\tau s}}{s} \tag{3.7}$$

Altså en forsterkning, tidsforsinkelse og integrasjon hvor

$$\omega_c \approx \omega_{c0}(h_{prosess}(s)) + 0{,}18\omega_r$$
$$\tau = \tau_0 - 0.08\omega_r$$
$$\tau_0 \approx \frac{\pi}{2\omega_{c0}(h_{prosess}(s))}$$

Parameteren ω_{c0} er verdien for den åpne sløyfens kryssfrekvens som operatøren benytter når båndbredden for referansen går mot null. Typiske verdier er gitt i tabell 3.3. ω_r angir båndbredden til referansen. Representative verdier for τ_0 er også gitt i tabell 3.3.

Modellen kalles Kryssmodellen fordi den har best gyldighet i området rundt kryssfrekvensen, ω_c. Den effektive transportforsinkelsen i Kryssmodellen inkluderer både reaksjonstid og treghet i det motoriske systemet.

Båndbredden til referansen betegnes altså ω_r. I forbindelse med utvikling av modellen ble det benyttet båndbredder på 1,5, 2,5 og 4,0 rad/s. I tabell 3.3 er det gitt tilnærmede verdier for kryssfrekvensen og beregnede transportforsinkelser når referansebåndbredden er null for noen forskjellige transferfunksjoner for prosessen, McRuer *et al.* (1965). En lang rekke av aktuelle prosesser for manuell regulering kan tilnærmes med disse transferfunksjonen i området rundt

kryssfrekvensen, f eks retningsstyring og banestyring av farkoster.

$h_{prosess}(s)$	τ_0 (s)	ω_{c0} (rad/s)
K	0,30	5,0
K/s	0,35	4,5
K/s^2	0,5	3,0

Tabell 3.3 Tilnærmede verdier for parametere i Kryssmodellen

Som vi ser har man oppnådd at bare en av parametrene i modellen er sensitiv for prosess-egenskapene. I tillegg til de transferfunksjonene som er gitt i tabell 3.3, ble også prosesser med transferfunksjoner

$$\frac{K}{s-2}, \quad \frac{K}{s(s-1/T)}$$

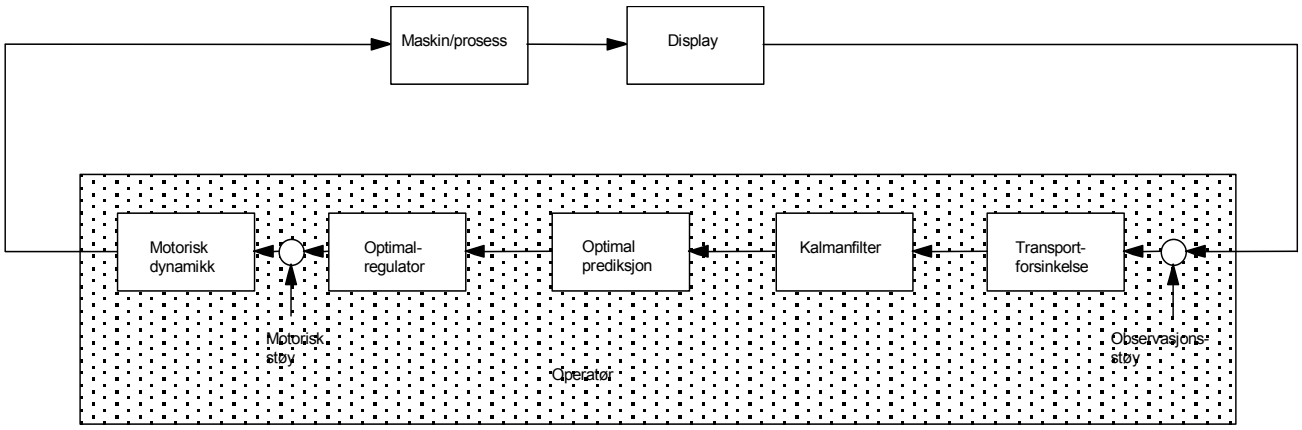
benyttet i eksperimentet, altså ustabile prosesser.

Kryssmodellen har også blitt videreutviklet til en *utvidet* Kryssmodell som tar hensyn til fallet i fase som observeres for lave frekvenser for «vanskelige» prosesser.

Kryssmodellen er altså en-dimensjonal, ved at både måling til og pådrag fra operatøren er skalare. I fler-dimensjonale situasjoner er det vanlig å anta uavhengige monovariable modeller i tillegg til at krysskoblinger mellom modellene blir tatt hensyn til. Ved manuell fartøysstyring (som er en multivariabel situasjon) er det vanlig å dele styringen inn i flere oppgaver og modellere operatøren som flere nøstede tilbakekoblingssløyfer.

Modeller basert på tilstandsromformuleringer

Med fremveksten av «moderne» reguleringsteknikk på 60-tallet var det naturlig å benytte denne også til å utvikle modeller for manuell regulering. Ved å benytte slike modeller er det lettere å modellere multivariable (eventuelt også tidsvarierende) situasjoner. Man antar her at en motivert trenet operatør vil styre en prosess optimalt med visse personlige begrensninger. Disse begrensningene er unøyaktigheter og begrensninger i avlesning (persepsjonsstøy), unøyaktig-heter i motorikk (motorisk støy) og unøyaktigheter i den interne modellen. I tillegg tas det hensyn til de dynamiske egenskapene som beskrevet ovenfor. Vi skal ikke her gå inn på de matematiske detaljene i denne modellen, se Baron *et al.* (1969), men begrense oss til en kvalitativ beskrivelse basert på strukturen av modellen gitt i figur 3.16. Modellen betegnes som *Baron-Kleinman-Levison modellen* etter de som først utviklet den, Baron, Kleinman og Levison (1971).



Figur 3-16 Baron-Kleinman-Levison optimal reguleringsmodell

Den første blokken er altså reaksjonstid (0,15 s) som vi har vært inne på tidligere i tillegg til at det er lagt til et støyledd for persepsjonsstøy. Basert på de observerte målingene estimerer operatøren tilstandene (se neste kapittel) og dette er modellert som et Kalman-filter. Disse tilstandene blir prediktert fram til nå-tidspunktet som modellerer operatørens evne til å kompensere for sine transportforsinkelse. Pådrag beregnes deretter optimalt ut fra en objektfunksjon. Dette pådraget påtrykkes prosessen gjennom det motoriske systemet som modelleres som ovenfor som et første-orden system (tidskonstant på ca 0,1 s) i tillegg til et additativt støyledd. Støybidragene er bestemt empirisk og kovariansen til avlesningsstøyen er ca $0{,}01\pi$ av den tilsvarende måleverdien (dvs ca 3% støy i forhold til signalet), mens den tilsvarende motoriske støyen er ca $0{,}003\pi$ eller 1%.

Modellen har blitt validert mot de samme prosessmodellene som Kryssmodellen gitt i tabell 3.3. Den har også blitt benyttet i forskjellige flyanvendelser. Den er også blitt utvidet til å inkludere menneskelige beslutninger, monitorering og deteksjon for å oppnå mer komplette modeller, f.eks. Baron *et al.* (1981). Dette er også gjort for flyanvendelser.

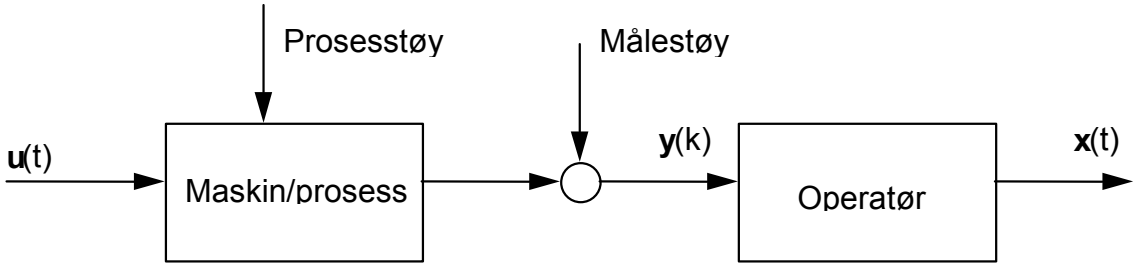
For ytterligere informasjon om Kryssmodellen, Baron-Kleinman-Levison modellen og om manuell regulering generelt, se f eks Rouse (1981), Sheridan og Ferrell (1974), Hess (1987) og Skare (1990).

3.4.2 Manuell monitorering/estimering

Manuell estimering inngår hvor operatøren skal overvåke en prosess, f eks som en del av det å kunne detektere feil eller at en spesiell situasjon har oppstått, f eks avstanden og relativ fart til et mål er slik at det må tas en beslutning om fyring av våpen. Estimering inngår også som en del av manuell regulering og ved bruk av optimalreguleringsmodeller vil estimeringsdelen benytte den samme modellen som vi skal beskrive her for overvåking/monitorering.

Situasjonen ved manuell estimering er gitt i figur 3.17. Ett spørsmål som er aktuelt er om vi

antar at resultatet av monitoreringen er estimerte verdier av de målte tilstandene eller tilstandene selv. Det er mest vanlig å anta at operatøren estimerer hele tilstandsvektoren, da det er bare på denne måten at han kan prediktere prosessen frem i tid.



Figur 3-17 Manuell monitorering/overvåking

Det antas at prosessen som overvåkes er lineær og at prosess- og målestøy kan antas som gaussisk hvit støy med null middelverdi og at de er ukorrelerte med hverandre og prosessens initialtilstand. For det tids-diskrete tilfellet har vi da:

$$\mathbf{x}(k+1)=\Phi\mathbf{x}(k)+\Delta\mathbf{u}(k)+\Omega\mathbf{v}(k) \tag{3.8}$$

$$\mathbf{y}(k)=D\mathbf{x}(k)+\mathbf{w}(k) \tag{3.9}$$

En normativ modell av operatøren som en estimator i et slikt tilfelle vil være et *Kalmanfilter*, se f eks Gelb (1978).

Estimatet er gitt ved:

$$\hat{\mathbf{x}}(k) = \bar{\mathbf{x}}(k) + K(k)(\mathbf{y}(k) - D(k)\bar{\mathbf{x}}(k)) \tag{3.10}$$

hvor filterforsterkningen er gitt ved

$$K(k) = \bar{X}(k)D^{T}(k)[D(k)\bar{X}(k)D^{T}(k) + W(k)]^{-1} \tag{3.11}$$

og dets kovariansmatrise er

$$\hat{X}(k) = (I - K(k)D(k))\bar{X}(k) \tag{3.12}$$

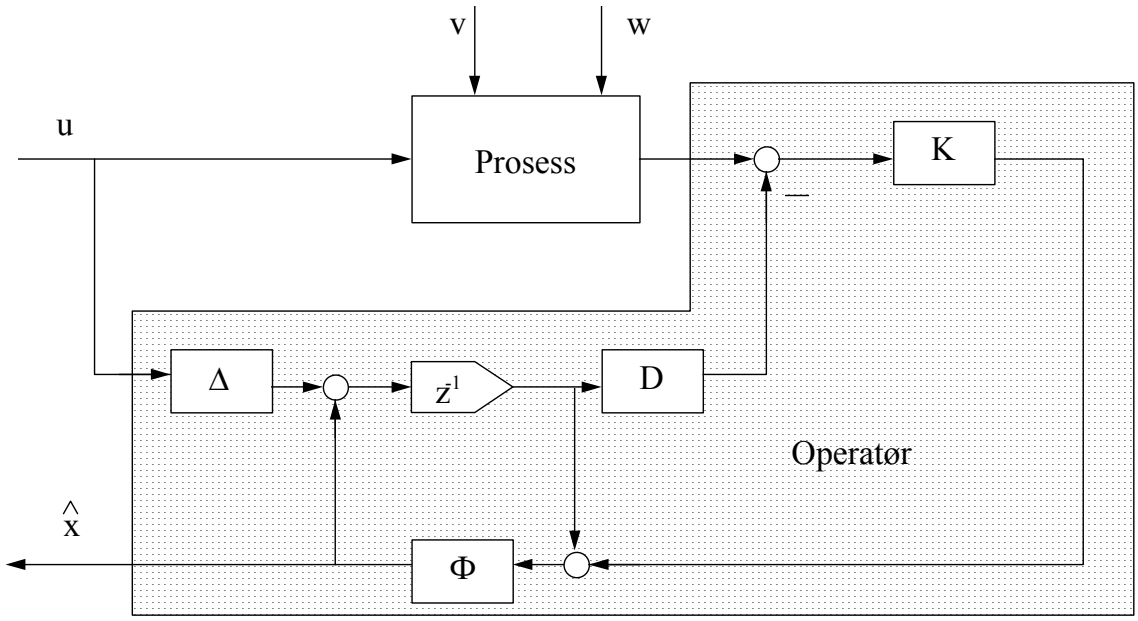
Prediksjon frem ett tidsskritt gjøres for tilstanden ved følgende ligning:

$$\bar{\mathbf{x}}(k+1) = \Phi(k)\hat{\mathbf{x}}(k) + \Delta(k)\mathbf{u}(k) \tag{3.13}$$

og for kovariansmatrisen ved:

$$\bar{X}(k+1) = \Phi(k)\hat{X}(k)\Phi^T(k) + \Omega(k)V(k)\Omega^T(k) \tag{3.14}$$

Figur 3.18 viser strukturen på Kalmanfilter-modellen. Estimatet er altså en vekting mellom den prediktert tilstandsvektor og avviket mellom virkelig og estimert målevektor, hvor vekting gjøres i henhold til forholdet mellom nøyaktigheten mellom den predikterte tilstanden og målingene. Er det mye målestøy i forhold til nøyaktigheten til prediktert tilstand tar filteret ikke så mye hensyn til de nye målingene og estimert tilstand blir svært lik prediktert tilstand. Legg merke til at filteret også gir ut nøyaktigheten av estimatene. Som det tidligere har blitt antydet er gjerne operatøren gjerne en konservativ estimator og opererer nesten «ballistisk». Dvs han tar ikke tilstrekkelig hensyn til nye målinger og legger for stor vekt på sin interne modell. Det er derfor en konstruksjonsutfordring å få han til å bruke tilgjengelig informasjon i stor nok grad. Det at operatøren opererer for mye i åpen sløyfe er et generelt problem og kan være en kilde til feilhandlinger. Det er derfor viktig å få operatøren til å «åpne» seg opp, dvs benytte oppdatert informasjon om prosessen.



Figur 3-18 Struktur på Kalmanfilteret

Rouse (1977) utviklet en normativ-deskriptiv modell ved å anta at operatøren benyttet to interne modeller med forskjellige tidskonstanter av prosessen, en i korttidshukommelsen og en i langtidshukommelsen. Basert på hver av disse ble det produsert to estimater som ble vektet som følger for å oppnå et resulterende estimat:

$$\hat{\mathbf{x}}(k) = \alpha\,\hat{\mathbf{x}}_{lang}(k) + (1-\alpha)\,\hat{\mathbf{x}}_{kort}(k) \tag{3.15}$$

Ut fra forsøk med forskjellige prosesser viste det seg at vekten mellom langtids- og korttidsestimatene korrelerte med hvor forutsigbar den enkelte prosess var. Langtids-

hukommelsesestimatet fikk større vekt ved prosesser som var lite forutsigbare.

3.4.3 Presentasjonsformater

Som vist i figur 3.15 har operatøren bare tilgjengelig avviket («compensatory display») og modellene som vi har beskrevet tidligere i dette kapitlet bygger på denne antagelsen. Dette er selvsagt en forenkling av mange situasjoner hvor operatøren direkte også har tilgang på annen informasjon, f eks tilgang til referansen et stykke frem i tid ved bilkjøring. I de situasjoner hvor dette ikke er direkte tilgjengelig, f eks styring av ubåt, er det mulig å øke ytelsen ved å presentere mer informasjon enn bare avviket ved nå-tidspunktet. Aktuell informasjon for presentasjon er:

- referansen og måleverdi separat ved nåtidspunktet («pursuit display»)
- referansen ved nåtidspunktet og et stykke frem i tid («preview display»), f eks «highway-in-the-sky» for fly og elektronisk vei på sjøen, Bråthen (1995), for båter.
- *modellbasert presentasjon*, dvs tilstandsvariable i nåtidspunktet og prediktert et stykke frem i tid basert på en antagelse om hvordan operatørens pådrag vil være i det samme tidsrommet.

De forenklede modellene som er beskrevet her, må utvides for å ta hensyn til slike presentasjonsformater. Dette blir ikke omtalt her, men se f eks Sheridan og Ferrell (1974).

Den modellbaserte presentasjonsmåten faller i en type presentasjonsformater som kalles *predikto bilder*. En annen presentasjonsform som faller i samme kategori kalles *«quickening display»* hvor ideen er å lage en presentasjon som er en veiet sum av de opp til n´te ordens deriverte av målingen (helst ved å ta ut tilstander direkte fra prosessen uten å måtte deriverte det målte signalet) og la dette være feilsignalet til operatøren. På den måten «fremskyndes» operatørens pådrag ved at de deriverte blir gjort tilgjengelige for operatøren og gjør det lettere for han å prediktere tilstandene frem i tid. Ut fra å vise et tidsvarierende signal greier vi å estimere den tidsderiverte, men akselleras jon og høyere ordens deriverte har vi problemer med. Siden dette er et konstruert avvik kombineres gjerne et slikt bilde med et konvensjonelt avviksbilde. «Quickening displays» har blitt benyttet i fly og er bl a beskrevet i Stokes, Wickens og Kite (1990). Mer informasjon om forskjellige presentasjonsformater finnes også i Sheridan og Ferrell (1974) og Rouse (1981).

Modellene av manuell styring kan prediktere hvilken ytelse som man kan forvente. I tillegg kan modellene benyttes ved konstruksjonen av egnede bilder. Ved f eks å benytte modellen basert på optimal regulering kan vi se på effekten ved å legge til og trekke fra tilstander som vises til operatøren ved å forandre målematrisa. Tilsynelatende ville det være ønskelig å presentere så mange av tilstandene som mulig, men ved å øke antall tilstander vil persepsjonsstøyen øke ved at det blir mindre tid for avlesning av hver enkelt av tilstandene. Dette er kort behandlet nedenfor. Det blir derfor en avveining mellom antall tilstander som vises og størrelsen på persepsjonsstøyen. Hvordan informasjonen blir presentert vil også selvsagt påvirke persepsjonsstøyen. Hvilken skalering som benyttes og egenskaper ved det medium det presenteres på, f eks oppløsning, vil ha betydning og igjen kan dette undersøkes vha modellen.

En måte å minske persepsjonsstøy uten å gå ned på antall tilstander som presenteres vil være å benytte *integrerte* bilder, ved å samle flere av tilstandsvariablene sammen til én presentasjonsenhet. Ut fra et forslag om et integrert bilde kan modellen igjen benyttes for å evaluere dette presentasjonsformatet basert f eks hvordan de individuelle tilstandene påvirker det integrerte bilde. Dvs tilsvarende som skalering av enkelttilstander. Eksempler på bruk av modellene for konstruksjon av operatørbilder finnes f eks i Baron og Levison (1977).

3.4.4 Fordeling av oppmerksomhet

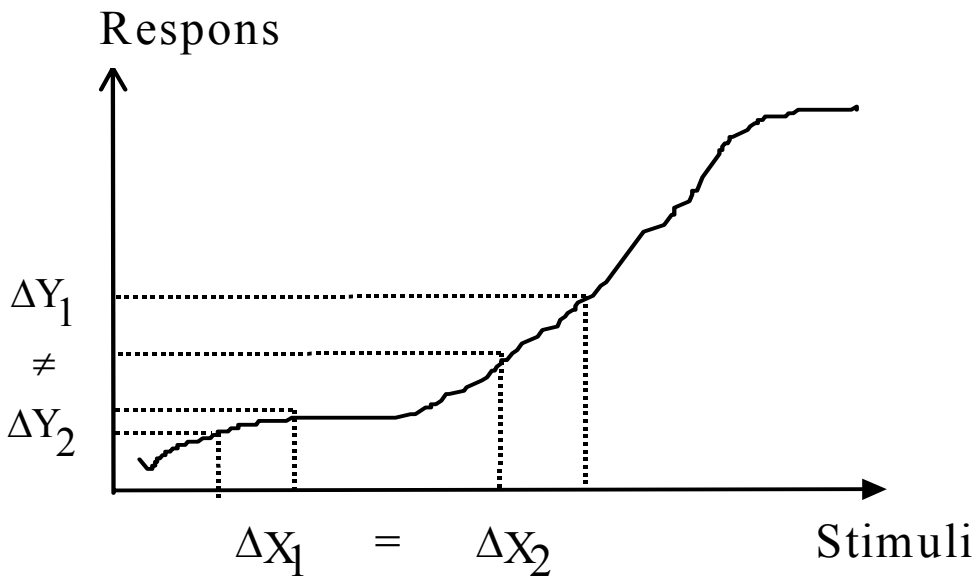
Som tidligere nevnt vil persepsjonsstøy øke ved å øke antallet tilstandsvariable som presenteres. Dette skjer fordi operatøren vil bruke mer tid på avlesning av instrumentene noe som fører til økt arbeidsbelastning. Ved å ta utgangspunkt i egenskapene til signalene som blir presentert vil det være mulig å komme frem til hvor ofte avlesning foretas av operatøren.

For å komme frem til dette antar vi at signalene er kontinuerlige, at de er uavhengige og er like viktige. Videre tar det like lang tid å avlese hvert signal og tiden det tar å skifte mellom avlesningene neglisjeres. Nyquist samplingsteorem sier at det er tilstrekkelig å taste (avlese) et signal med båndbredde på ω Hz med 2ω tastinger pr sekund for å kunne gjenskape signalet. Hvordan en operatør avleser i forhold til dette under antagelsene som beskrevet ovenfor ble undersøkt av Sender (1964). Det viste seg at operatørene undertastet høyfrekvente signaler, mens lavfrekvente signaler ble overtastet. Båndbredden for signalet var i området 0,05-0,6 Hz. Undertasting av høyfrekvente (> 0,3 Hz) signaler peker i retning av at operatøren har en intern modell som benyttes til å prediktere signalet frem i tid, eller med andre ord operatøren er i stand til å estimere den tidsderiverte av signalet. Ved også å presentere rateinformasjon ble signalet tastet med ω. Overtastingen av lavfrekvente signaler kan forklares med at operatøren glemmer. Senders data peker i retning av at operatøren taster med $1{,}34\omega$ et signal med båndbredde ω. Dette er altså lavere enn hva vi skulle forvente fra Nyquist samplingsteorem.

3.4.5 Operatør som ulineær systemkomponent

Like store endringer av stimuli vil generelt sett oppfattes ulikt og gi ulik reaksjon hos en operatør. Dette er vist i figur 3.19. Følsomheten overfor stimuli er variabel og vil avhenge av psykologiske og fysiologiske forhold og det vil kunne være store individuelle variasjoner.

Operatøren er i stor grad adaptiv og har evnen til raskt å modifisere ytelsesparametere, f eks å gjøre avveining mellom hurtighet og nøyaktighet. Tilbakekobling om ytelsen er nødvendig for effektiv avveining. En utilstrekkelig eller dårlig tilbakemelding gir ikke-optimal adapsjon. Adapsjon vil alltid kreve noe tid og ressurser og ekstrem adapsjon medfører stress. Konklusjonen er altså at operatører må betraktes som en (tildels sterkt) ulineær systemkomponent.



Figur 3-19 Operatøren er en ulineær systemkomponent

Omtrent alle fysiske systemer er ulineære, men de kan ofte modelleres tilstrekkelig godt med en lineær modell. Praktisk bruk av reguleringsteori forutsetter i stor grad en lineær systemmodell. Effekten av ulineariteter kan reduseres vha tilbakekobling som beskrevet i appendiks B.

3.5 Modellering av prosedyrebaserte oppgaver

I forrige underkapittel omhandlet vi FB oppførsel og basis her var kontinuerlige tilstandsvariable og differensiallikninger for å beskrive fysiske prosesser og operatører, dvs kontinuerlige systemer. Vi skal nå se på noen forskjellige representasjoner som kan nyttes for modellering av PB oppførsel. I et MMS vil hendelser aktivere operatøroppgaver og vi kan følgelig betrakte operatøren som en hendelsessensitiv informasjonsbehandler. Vi skal her beskrive noen representasjoner av hendelsesdrevne systemer på en uformell (og til tider noe upresis) måte der vi vektlegger aspekter som er viktig ved praktisk bruk. Disse måtene å beskrive denne typen oppførsel på er i stor grad hentet fra programvare- og systemteknikk og blir også benyttet for funksjonsanalysen, ref underkapittel 4.2.1.

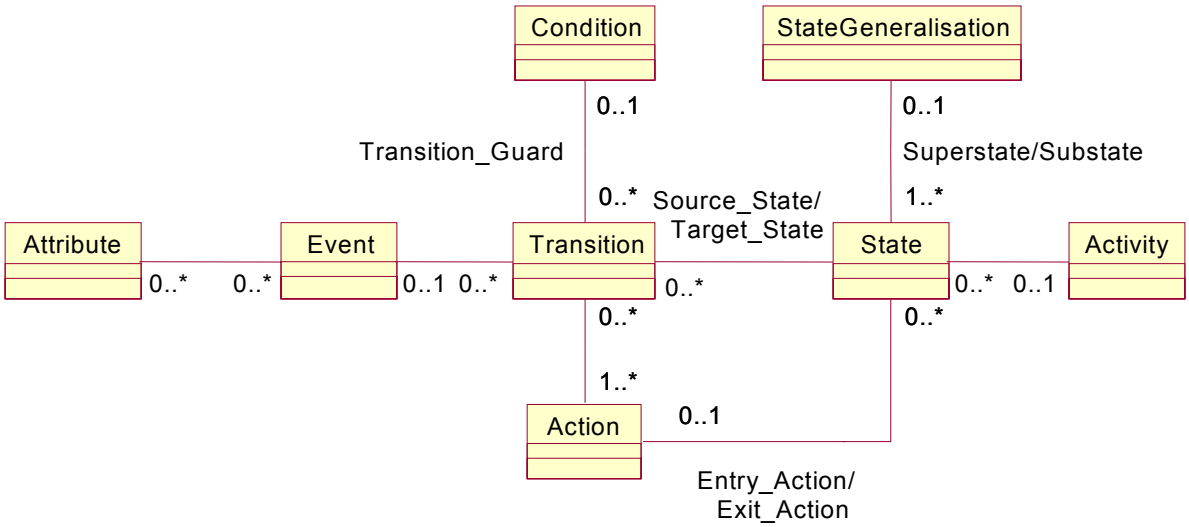
3.5.1 Tilstandsmaskinen

Sekvensdiagrammer. Sekvensdiagrammer beskriver rekkefølgen av påvirkninger mellom to eller flere systemer. Dersom vi lar meldinger representere «påvirkninger» kalles dette for *meldingssekvensdiagram.* Sekvensdiagrammer brukes en god del til å vise interaksjoner mellom en operatør og maskin (evt. andre operatører), se Figur 3-21. Innenfor MMS omtales teknikken som operasjonssekvensdiagrammer («Operational Sequence Diagrams - OSD»). Et OSD blir imidlertid fort komplisert og uoversiktlig dersom mange mulige alternative grener skal vises samtidig. Se diskusjon i underkapittel 4.3. Et *transisjonsdiagram* er en forenkling av sekvensdiagrammet som kun beskriver stimuli (innmeldinger) og respons (utmeldinger) for ett enkelt system. Slike beskrivelser kan samles på en mer kompakt måte i et tilstandsdiagram (se

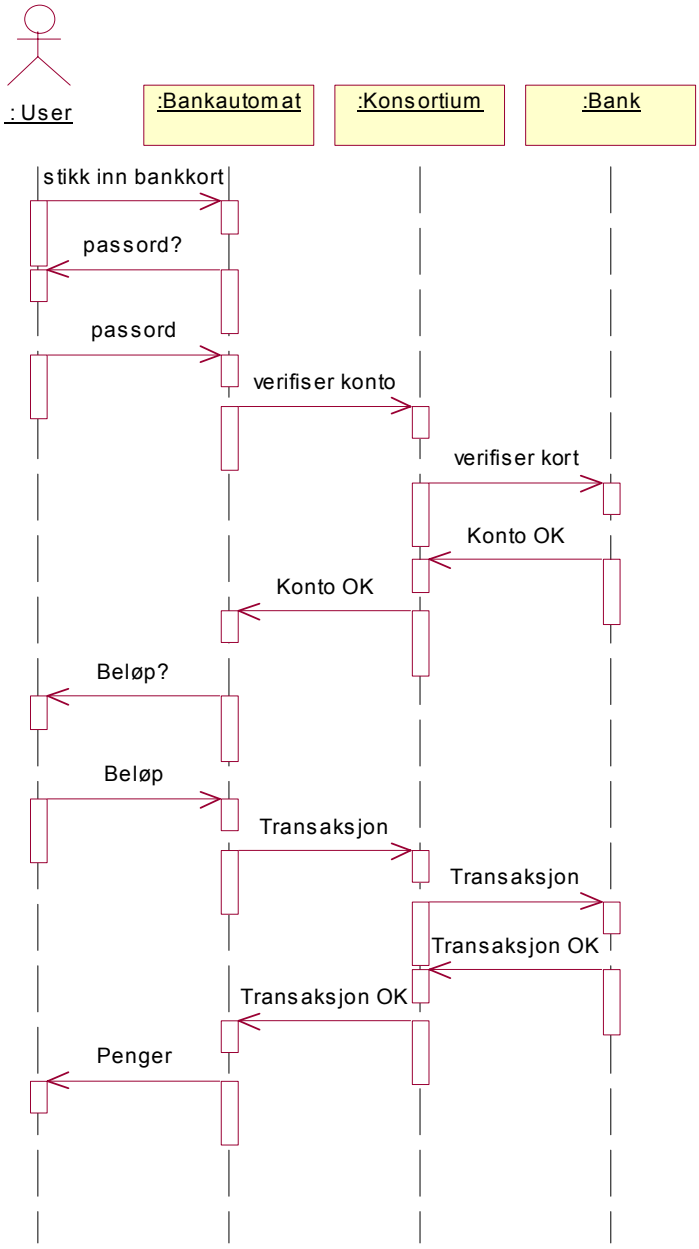
senere).

Tilstandsmaskin. Vi har tidligere vært inne på betydningen av PB/KB oppførsel i moderne MMS. Modellering av slik oppførsel krever modeller som behandler symboler (heretter brukt som fellesbetegnelse på tegn/symboler i underkapittel 3.3) og ikke bare tall (signaler). Den vanligste dynamiske (dvs med hukommelse) symbolprosesserende modellen er *endelig tilstandsmaskin (TM)*. Den beskriver stimuli-respons oppførsel, f eks hos en operatør. Dvs hvordan sekvenser av stimuli er relatert til sekvenser av respons.

En endelig TM består av et (endelige) sett *inngangssymboler* (hendelser) og *utgangssymboler.* Maskinen har hukommelse representert ved et endelig mengde med *tilstander* (disse kan representeres med en enkel tilstandsvariabel som kan innta et endelig antall diskrete verdier). Utgangssymbolet er bestemt av inngangssymbol og tilstand. Tilsvarende oppdateres tilstanden, også bestemt av inngangssymbol og tilstand. TM starter fra en bestemt tilstand (*initialtilstand*). Overgangen mellom tilstander kalles *transisjoner.* I appendiks A har vi definert noen sentrale begrep og gitt en mer formell beskrivelse av TM. I digitale systemer med binærvariable er inndata, tilstand og utdata gitt som vektorer. Vektornotasjon brukes ofte i andre sammenhenger også. En TM er framstilt skjematisk i figur 3.22. Figuren viser at man ved bruk av TM i praksis også må definere egenskapene til køen.

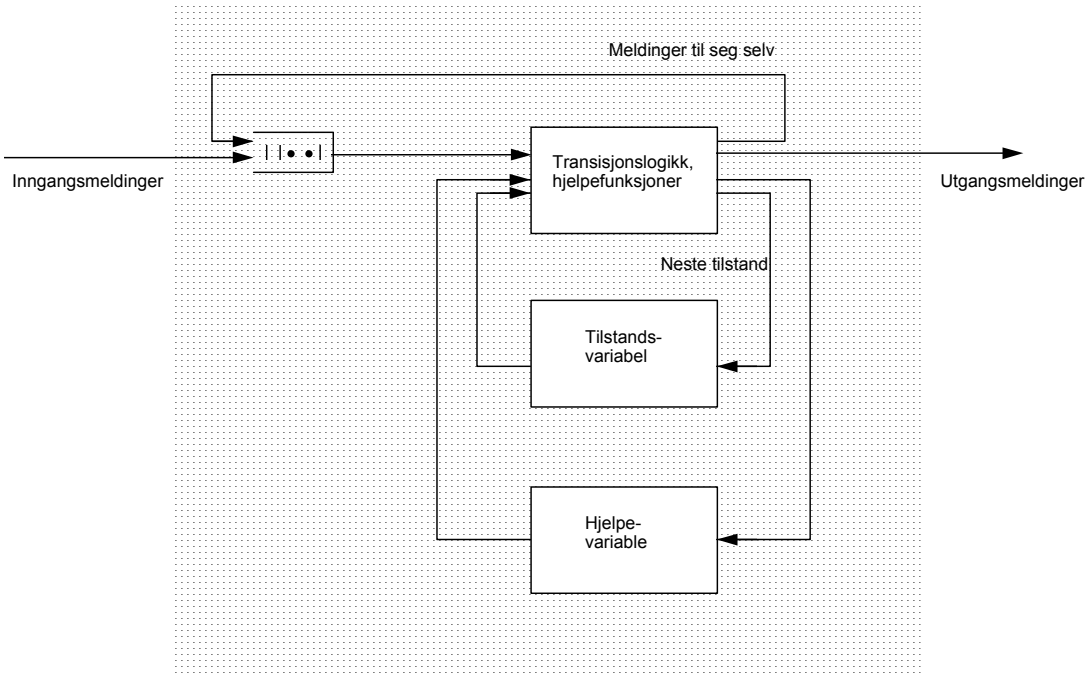


Figur 3-20 Klassediagram for tilsdandsdiagram (Rumbough et al, 1991)

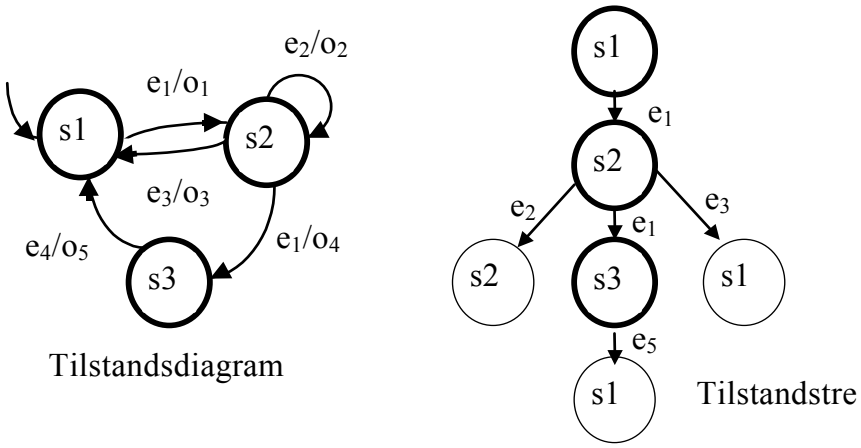


Figur 3-21 Sekvensdiagram (Rumbaugh et al.*, 1991)*

En TM kan visualiseres på flere måter som vist i figur 3.23. Den kan vises som et tilstands-diagram, tilstandstre eller som en tabell (tilstand-tilstand eller hendelse-tilstand). Et tilstands-diagram er en rettet graf der nodene representerer tilstander. I tilstandstreet representerer hver gren en delsekvens der rotnoden angir initialtilstand og siste node er lik startnode (tilstand) for påfølgende delsekvens.



Figur 3-22 Utvidet tilstandsmaskin



	s1	s2	s3
s1		e_1/o_1	
s2	e_3/o_3	e_2/o_2	e_1/o_4
s3	e_4/o_5		

Tilstand-tilstand tabell

	s1	s2	s3
e_1	$s2/o_1$	$s3/o_4$	
e_2		$s2/o_2$	
e_3		$s1/o_3$	
e_4			$s1/o_5$

Tilstand-hendelse tabell

Figur 3-23 Ulike framstillinger av en tilstandsmaskin

En rettet graf kan representeres ved en logisk matrise og det finnes enkle algoritmer som kan beregne hvorvidt det er mulig å komme fra en tilstand til en annen, og hvor mange transisjoner

dette krever (Dahl & Belsnes, 1978).

Identifikasjon er hvordan man med utgangspunkt i inn- og utdata kan spesifisere parametere og eventuelt også struktur i den typen modell man har valgt for å beskrive oppførselen. Det er to hovedklasser av metoder: menneskelig inferens/vurdering og databaserte algoritmer. Det har blitt forsket mye på algoritmisk identifikasjon av TM, men i følge Rouse et al. (1989) finnes det ennå ingen metode som gir entydige resultater. Hukommelsen i TM kompliserer identifikasjonsproblemet ved at man får en avveiningssituasjon mellom ekstra tilstander eller mer komplisert neste-tilstandsfunksjon.

3.5.2 Utvidelser av tilstandsmaskinen.

Dersom man prøver å beskrive dynamiske systemer med en viss kompleksitet ved hjelp av en TM, vil man fort støte på en rekke problemer:

1. *Den er rent sekvensiell.* I virkelige systemer vil det være mange mer eller mindre uavhengige komponenter. Dvs at en stor del av stimuli og respons i systemet er uordnet. En TM må eksplisitt liste alle mulige kombinasjoner og kan derfor ikke beskrive parallellitet på en tilstrekkelig effektiv og konsis nok måte. Et eksempel: En heis vil kunne ha 2 tilstander (oppe, nede). Dersom det er N uavhengige heiser i et bygg blir antallet tilstander for totalsystemet lik 2^N. Dette kalles kombinatorisk eksplosjon.
2. *Den kan i praksis ikke lagre eller operere på data.* Eksplisitte tilstandsverdier gir en svært ineffektiv representasjon av data. Alle aksjoner som er nødvendige for å generere utgangssymboler ved tilstandsoverganger kan relateres til en eller flere transisjoner, men de navngis eller beskrives ikke eksplisitt. Gjenbruk av aksjoner (typer/instanser) kan ikke beskrives. Aksjoner kan alternativt også relateres til tilstander istedenfor eller i tillegg til transisjoner. TM er altså ikke koblet til de andre perspektivene som er nødvendig når en integrert systemmodell skal lages.
3. *Den har ikke strukturingsmekanismer.* For å kunne håndtere og forstå beskrivelser av kompliserte systemer er det nødvendig å kunne aggregere/partisjonere TM både mhp tilstand og transisjoner. F eks kan heissystemet over partisjoneres i N uavhengige maskiner med $2*N$ tilstander. Et system med N tilstander vil maksimalt kunne ha N^2 transisjoner. Strukturering kan ofte redusere dette ned mot O(N).
4. *Den inkluderer ikke tid.* I sanntidssystemer er det ikke bare viktig at ting skjer i riktig rekkefølge, men også til rett tid!

Ad 1. Parallellitet kan representeres ved å innføre flere uavhengige TM som på en eller annen måte kan kommunisere med hverandre. I programvareteknikk beskrives uavhengig oppførsel som parallelle *prosesser* (eller kvasi-parallelle prosesser dersom disse kjøres på en tidsdelt datamaskin). Hver prosess har sin egen TM som illustrert i figur 3.24. Kommunikasjon skjer vha meldingsmekanismer i programmeringsspråket og/eller operativsystemet. Prosessbegrepet brukes også i spesifikasjonsspråk. I Objekt Orientering (OO) snakker man ofte om objekter på samme måte som prosesser, dvs kommuniserende objekter som har selvstendig oppførsel og som kan innkapsle hver sin TM. Alternativt kan parallellitet beskrives med Petrinett. Dette er tilstandsmaskiner hvor mer enn en tilstand kan være aktiv samtidig . Denne type modeller blir beskrevet mer inngående i underkapittel 3.5.3.

Ad 2. I den utvidete TM («Extended Finite State Machine - EFSM») innføres data. Dette gjøres ved å innføre hjelpevariable der attributter knyttes til meldingene (hendelsene) og verdier av utførte operasjoner kan lagres. I tillegg defineres operasjoner på hjelpevariable (fra primitiver som lesing/skriving til mer kompliserte operasjoner). Hjelpevariable er spesielt nyttig ved generering av utdata. Den utvidete TM er formelt beskrevet i appendiks A.

En vanlig brukt syntaks for transisjoner er: *hendelse* og/eller *betingelse*/*aksjon.* Denne er brukt i Object Modeling Technique (OMT) beskrevet i (Rumbaugh *et al.*, 1991). Betingelsen er gitt ved ulike variable og kan transformeres til en logisk variabel. I OMT skiller man mellom:

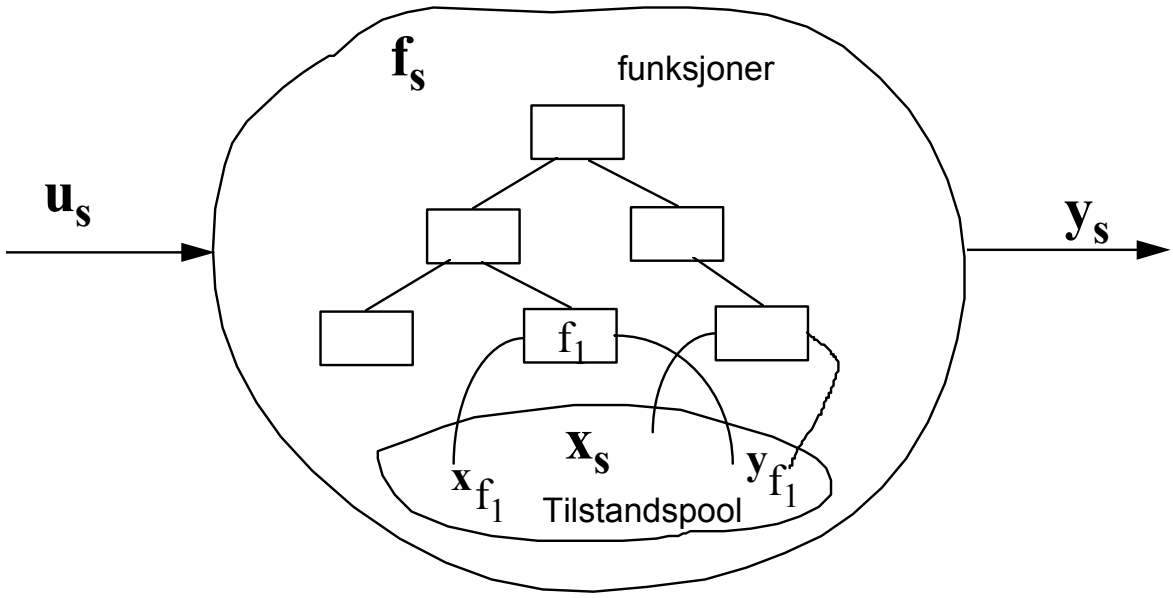
- Aksjon («action») som er en instantan operasjon koplet til en hendelse. Med instantan menes at tidsforbruket er ubetydelig i forhold til tidsoppløsningen i TM og
- Aktivitet («activity») som tar tid og som er assosiert med en tilstand.

Dvs at man kobler operasjoner både til tilstand og transisjon (hendelse). Se «telefon-eksempel» i figur 3.28.

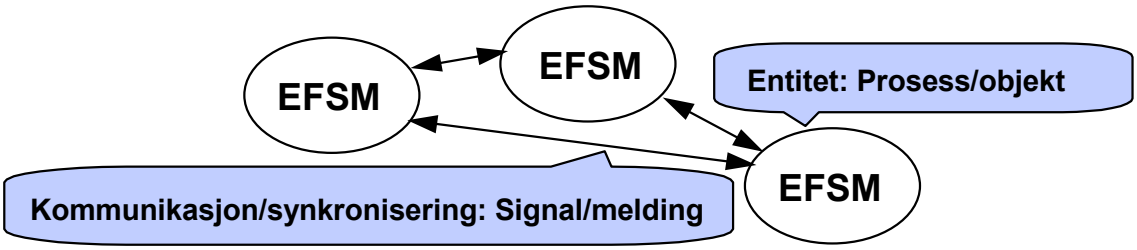
Ad 3. *Strukturering.* En primitiv funksjon i programvare har normalt ikke tilstand, mens et programsystem vil ha det. I tradisjonell funksjonsdekomponering beskrives et system ofte som et hierarki av funksjoner *uten* tilstand (se f eks kapittel 4.2.1). Tilstandvariable i programsystemet er globale og aksesseres av alle funksjoner som vist i figur 3.24. Den dynamiske oppførsel blir altså ikke dekomponert eller bibeholdt under detaljeringen av beskrivelsen. Jobling *et al.* (1994) anser dette som et hovedproblem med tradisjonell funksjonsdekomponering og en slik framgangsmåte bryter med hvordan man betrakter dynamiske systemer innenfor systemteori. Jobling *et al.* mener at OO beskrivelser løser problemet knyttet til dekomponering av tilstand (oppførsel).

Ad 4. Tid. Tid kan innføres som en global variabel tilgjengelig for alle TM og kan knyttes til hendelser eller testes direkte i betingelser.

SDL (Specification and Description Language) er et standardisert språk for beskrivelse og spesifikasjon av sanntidsystemer som er spesielt mye brukt under utvikling av tele-kommunikasjons-systemer. OSDL eller SDL-92 er en OO utvidelse av språket (Bræk & Haugen, 1993). Systemet og dets omgivelser beskrives i SDL som en struktur av *blokker* forbundet med kanaler. Blokker kan dekomponeres og oppførselen beskrives i *prosesser.* Hver prosess er modellert som en utvidet TM og kommunikasjon mellom prosessene er kun mulig med signaler som genereres og konsumeres av de utvidete TM-ene, jfr figur 3.25. *Blokktyper* kan gjenbrukes når nye blokker skal beskrives. Den nye blokken arver data, aksjoner og TM, jfr figur 3.26. Disse kan da eventuelt utvides eller redefineres dersom ønsket iht vanlig OO. Prosesser i SDL kan godt betraktes som objekter. (O)SDL-modeller kan eksekveres og implementering i program- og maskinvare er delvis automatisert.



Figur 3-24 Bruk av globale tilstander i «tradisjonell» funksjonsdekomponering (Jobling et al.*, 1994)*

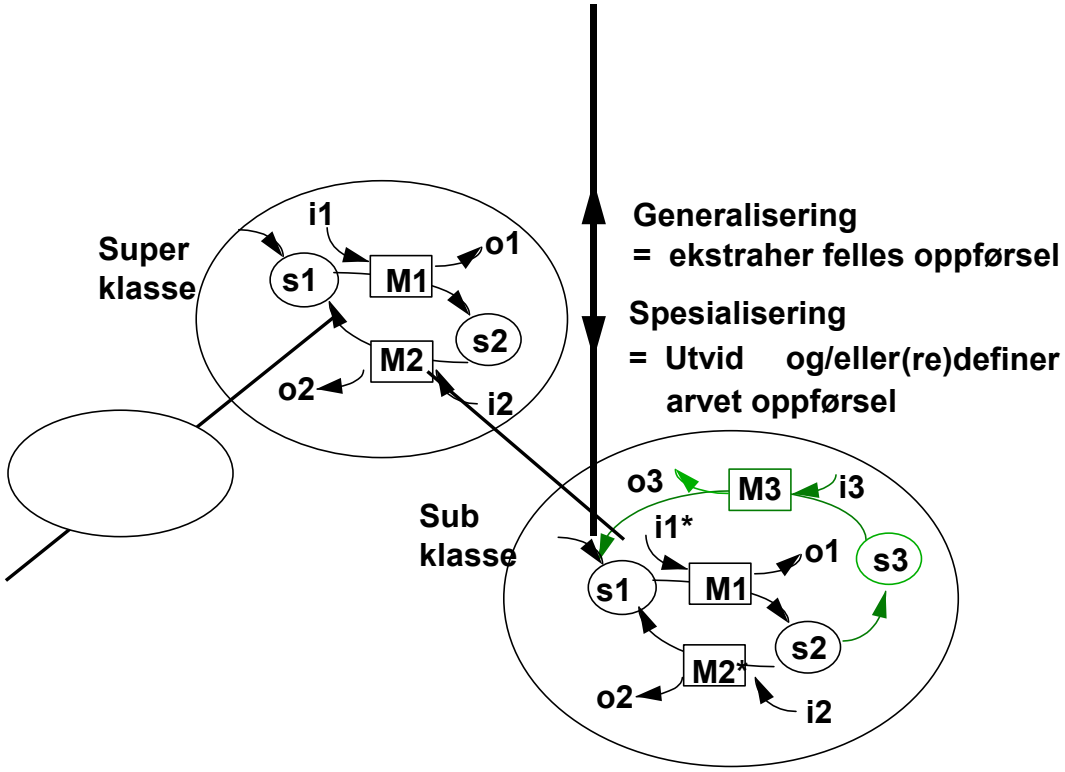


Figur 3-25 Beskrivelse av parallell oppførsel

Tilstandskart («Statecharts») er et beskrivelsesspråk som benytter aggregerte tilstander («superstates») som vi her vil kalle en supertilstand (Harel, 1987). Supertilstander er en logisk AND eller OR kombinasjon av tilstander, evt. supertilstander. Man kan følgelig definere et tilstandshierarki. Tilstandskart kan også beskrive aggregerte transisjoner mellom supertilstander. Dette gir en effektiv modellering av avbrudd. Et eksempel på et tilstandskart er vist i figur 3.28.

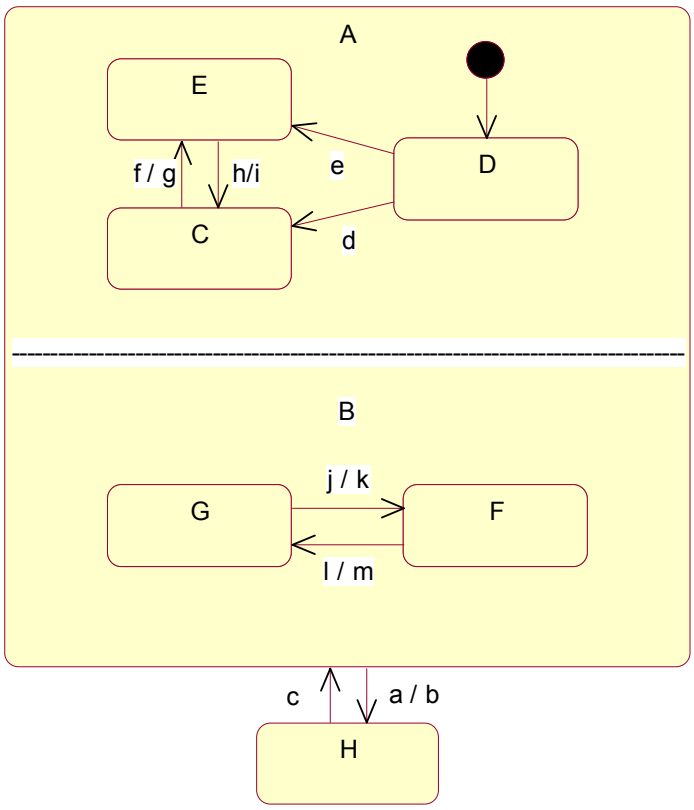
Oppførselsgrafer («Behaviour graphs») er et beskrivelsesspråk som ikke eksplisitt beskriver TM (Bråthen, Nordø &Veum, 1994). Basiselementene i dette språket er *diskret funksjon*, *diskret dataelement («item»)* og *tilstand*. En diskret funksjon har som inndata ett diskret dataelement (av flere mulige) og genererer et sett av diskrete dataelementer. Et diskret dataelement representerer en hendelse som trigger den diskrete funksjonen. Et nettverk av diskrete funksjoner (kalt «F-nett») kan transformeres til et tilstandsdiagram dersom man assosierer tilstander med kantene i nettverket. Dvs at de diskrete funksjonene beskriver aksjoner knyttet til tilstandstransisjoner. Et «F-nett» kan aggregeres til en såkalt *tidsfunksjon*. Man kan videre lage

hierarkier av tidsfunksjoner. Sekvens og parallellitet av funksjoner beskrives som kontrollnoder i oppførselsgrafene. Parallelle grener refereres til som «prosesser». Se eksempel i figur 4.6.

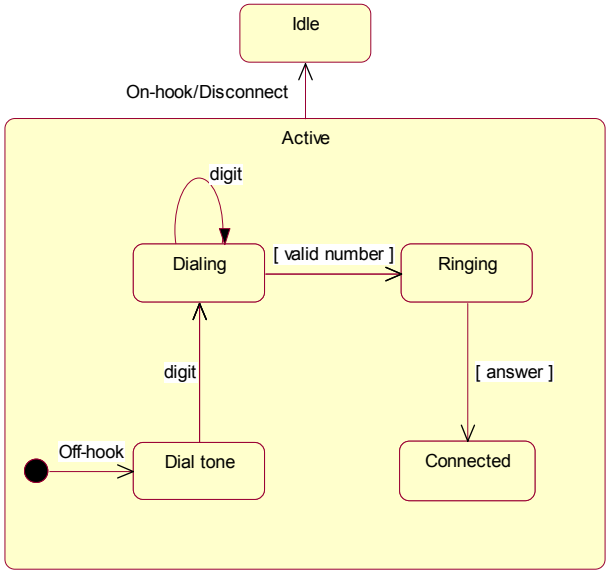


Figur 3-26 Arv av oppførsel. Oppførselen i subklassen er spesialisert ved at man har utvidet med en ny tilstand (s3) og metode (M3) og redefinert i1 og M2

Språk som OMT/UML, (O)SDL, Oppførselsgrafer, Tilstandskart og utvidelser av Petrinett adresserer hver for seg alle «problemene» knyttet til bruk av TM som diskutert over, men i varierende grad og med tildels store forskjeller i formalisme. Muligheten til formelt å analysere modellene varierer. Simuleringer er som regel den primære analysemetoden. Språkene har ulike typer av grafiske notasjoner og kan i prinsippet eksekveres. Den praktiske nytteverdien ved disse aspektene avhenger sterkt av tilgjengelige dataverktøy som støtter språket. Ingen av språkene er ideelle, dvs egnet til å beskrive alle typer perspektiver på en effektiv måte. Ved valg av språk bør man stille seg de samme spørsmål som vi kort diskuterte angående modeller i innledningen av kapittel 3.



Figur 3-27 Tilstandskart. Den store firkanten viser en supertilstand bestående av to parallelle tilstandsmaskiner (A og B). Initiell tilstand er D og F.



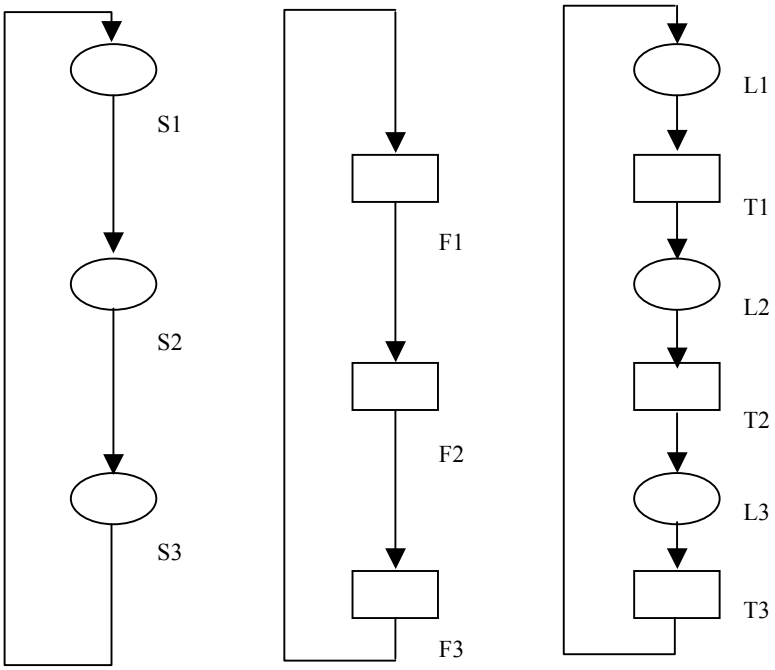
Figur 3-28 Telefoneksempel (1). Dynamisk modell (tilstandsdiagram), (Rumbaugh et al. 1991)

3.5.3 Petrinett

Petrinett brukes for å modellere hendelsesdrevne systemer som inneholder

- Samtidighet/parallellitet
- Konkurranse om begrensede ressurser
- Kødannelser
- Ruting
- Synkronisering

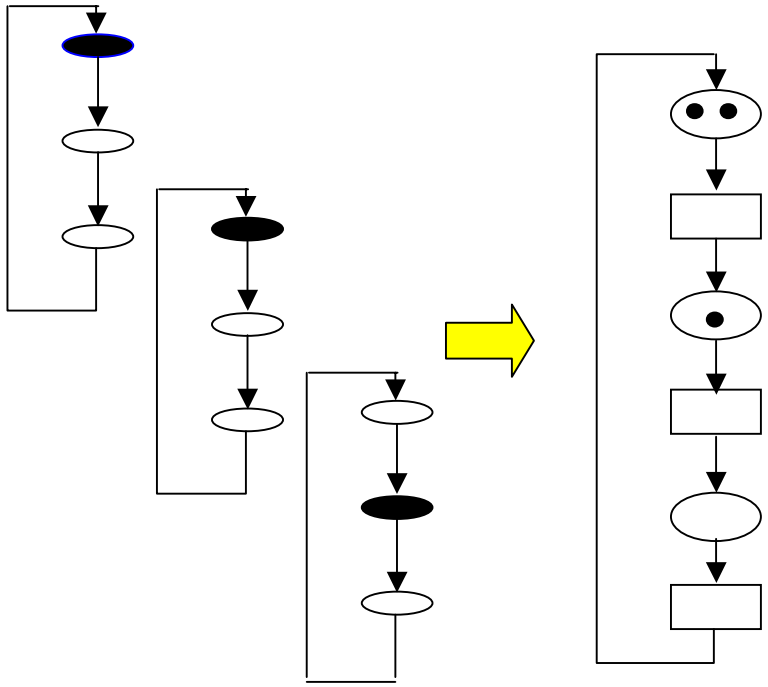
De består av lokasjoner (tilstander) og transisjoner (funksjoner/aksjoner) som i grafiske fremstillinger representeres av henholdsvis sirkler (eller ellipser) og firkanter (eller streker). Petrinett gir derfor en mer eksplisitt representasjon av disse to typer av elementer enn tilstandsmaskiner (hvor funksjoner/aksjoner er implisitte) eller oppførselsgrafer (hvor tilstander er implisitte). Se figur 3-29.



Figur 3-29 Tilstandsorientert modellering, funksjonsorientert modellering og Petrinett-modellering av et sekvensielt system

Parallellitet modelleres ved at flere lokasjoner kan være aktive samtidig. Dette markeres med å plassere et merke (token) i de aktive lokasjonene (figur 3-30). Man kan dermed modellere flere parallelle prosesser i ett diagram – noe som kalles for folding. På den måten unngår man å tegne flere like tilstandsdiagrammer som ville vært nødvendig i tilstandskart-formalismen. Denne teknikken kan føres et trinn videre ved å innføre forskjellige typer merker – såkalte fargede

merker. Her kan en merketype (farge) være en kompleks datatype. Forholdsvise enkle nett kan på denne måten modellere kompliserte systemer.



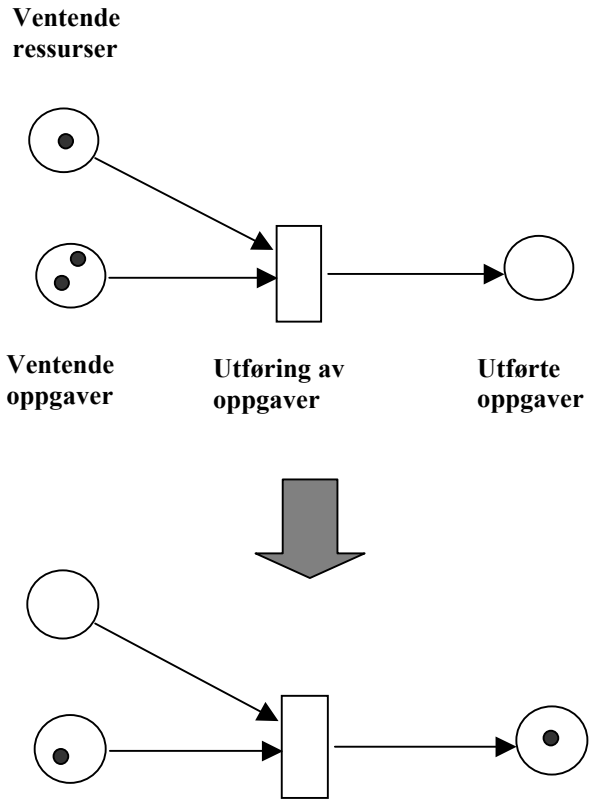
Figur 3-30 Modellering av parallelle prosesser med Petrinett

Synkronisering modelleres i Petrinett ved at en transisjon med flere innganger må ha minst ett merke i hver inngangslokasjon for å kunne aktiveres. Når dette inntreffer vil merkene som er med på å aktivere transisjonen fjernes fra inngangslokasjonene og nye merker plasseres i utgangslokasjonene. Vi sier at transisjonen *fyrer*. Dette er vist i et eksempel i figur 3-31. I høynivå Petrinett vil antall og type av utgangsmerkene kunne bestemmes av vilkårlig komplekse funksjoner. Når det fins flere merker i en inngangslokasjon, velges typisk merket som skal bindes ved transisjons-fyringen ut ved å trekke et tilfeldig tall.

I den enkleste typen Petri-nett er rekkefølgen av transisjoner udefinert. I en systemtilstand der flere transisjoner er aktivert, kan hvilken som helst være den neste som blir utført. I Petrinett med tidsutvidelse (tidspetrinett) fjernes denne usikkerheten ved at en transisjonsfyring tar en angitt tid. Tidsintervallet kan enten være fast eller varieres tilfeldig etter en bestemt sannsynlighetsfordeling. Denne siste type nett kalles stokastiske tidspetrinett.

I praktiske verktøy for simulering av Petrinett[1] fins det ofte flere utvidelser som gjør modelleringsjobben enklere. For eksempel er det fordelaktig å dele komplekse modeller opp i flere enklere modeller. Dette er ideen bak *hierarkiske* Petrinett hvor en transisjon kan representere et helt subnett.

[1] Et slik verktøy er Design/CPN som er gratis tilgjengelig fra http://www.daimi.aau.dk/designCPN/



Figur 3-31 Utføringen av en oppgave må vente til minst en oppgave og minst en ressurs er tilgjengelig i inngangslokasjonene

Greninskripsjoner er uttrykk som evalueres til bager av merker. Bager er mengder som kan inneholde flere eksemplarer av hvert merke. Uttrykkene kan inneholde konstanter, variable og være betinget av logiske uttrykk med if-then-else-struktur. *Guard-uttrykk* brukes for å spesifisere hvilke typer merker som kan aktivere en transisjon. Disse gir mulighet for å spesifisere aktiverende bindinger som ligger mellom konstanter (fullstendig begrenset) og variable (fullstendig ubegrenset). Noen verktøy tillater også å assosiere *kode* fra et vanlig programmeringsspråk med transisjoner. Dette kan for eksempel brukes for å skrive resultater fra en simulering til filer eller for å beregne ulike typer ytelsesparametere.

3.5.4 Markovkjeder

Vi skal her se på en annen type utvidelse av TM. I et system vil hendelser kunne inntreffe mer eller mindre tilfeldig og det kan derfor være naturlig å modellere disse som stokastiske. Tilsvarende kan lovmessighetene være så kompliserte å identifisere at man velger å se på disse som stokastiske. Tilstandsvariabelen i TM, som representerer tilstandene, vil følgelig kunne betraktes som en stokastisk variabel i et gitt tidspunkt. La $x_n = i$ betegne at tilstanden etter n-te transisjon er "i" og la p_i^n angi sannsynligheten for dette. Transisjonssannsynlighetene defineres da slik:

$$p_{i,j}^{n,n+1} = \mathrm{P}\left[x_{n+1} = j \middle| x_n = i \right] \text{ der } p_{i,j}^{n,n+1} \geq 0 \text{ og } \sum_{j \in M} p_{i,j}^{n,n+1} = 1 \tag{3.16}$$

hvor M er mengden av tilstander. Når de betingede tilstandssannsynlighetene oppfyller

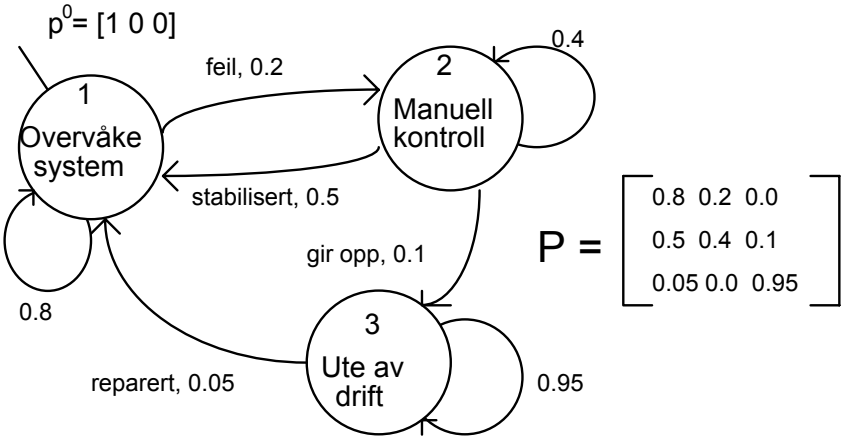
$$P[x_{n+1} = j \mid x_n = i, x_{n-1} = k, \cdots, x_0 = l] = P[x_{n+1} = j \mid x_n = i] \tag{3.17}$$

kalles prosesse en *Markovkjede* (Karlin & Taylor, 1975). Dvs at neste tilstand bare avhenger av nåværende tilstand og ikke av systemets tidligere historie.

På samme måte som vi lot en logisk matrise representere TM, benyttes en matrise med tilsvarende dimensjon, men hvor elementene er transisjonssannsynligheter istedenfor {0,1}. La **P** betegne denne matrisen. Sannsynlighetsvektoren for $\mathbf{x_n}$ kan da beregnes slik:

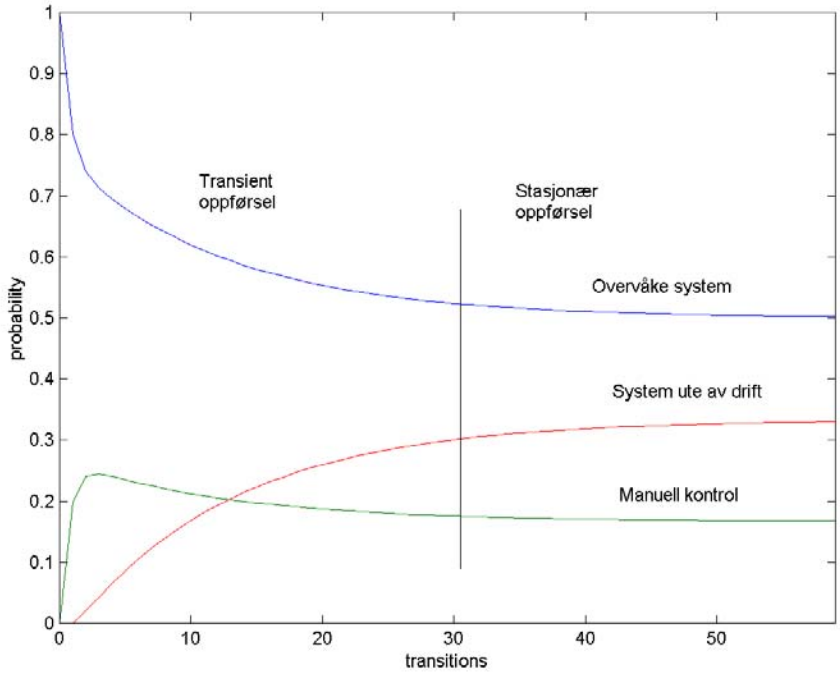
$$\mathbf{p}^n = \mathbf{p}^0 \cdot \mathbf{P}^n \tag{3.18 }$$

der $\mathbf{p}^0$ er initiell sannsynlighetsvektor. I figur 3.33 under har vi beskrevet et enkelt MMS. Operatøren antas å overvåke et automatisert system. Dersom feil oppstår må operatøren overta styringen. Vanligvis vil han etter en stund greie å stabilisere systemet. I noen situasjoner vil han ikke greie dette og systemet blir ute av drift. Etter noen tid vil spesialister få reparert og igangsatt systemet og operatøren begynner på nytt å overvåke systemet.



Figur 3-32 Eksempel på Markovkjede illustrert som tilstandsdiagram

I figur 3.35 har vi beregnet sannsynlighetene for de ulike tilstandene som funksjon av antall transisjoner. Dersom vi assosierer en viss arbeidsbelastning med de ulike tilstandene, f eks 1: moderat, 2: høy og 3: ingen, kan vi si noe om operatørens gjennomsnittlige arbeidsbelastning. Merk forskjellen mellom transient og stasjonær oppførsel i figuren. Dersom vi empirisk hadde observert systemet f eks de 20 første transisjonene og operatørens skift varte betydelig lengre, da ville resultatene vært noe misvisende mht operatørens gjennomsnittlige arbeidsbelastning



Figur 3-33 Sannsynlighet for tilstandene som funksjon av antall transisjoner

Markovmodeller har bl a blitt brukt til å prediktere sannsynligheten for at operatøren skifter oppmerksomhet fra et instrument til et annet. En enkel modell er beskrevet i (Boff & Lincoln, 1986) der man har N instrumenter som hver får en del av det totale antall fikseringer, dvs definert vha den stasjonære sannsynlighetsvektoren $\mathbf{p}^{\infty}$. Et sett med transisjonssannsynligheter som er konsistent med denne er:

$$p_{i,j}^{n,n+1} = p_j^{\infty} \qquad (3.19)$$

Man kan så beregne sannsynligheter for å observere overganger mellom ulike instrumenter. Vi bør merke oss at det finnes andre sett med verdier for transisjonssannsynlighetene som gir samme stasjonære sannsynlighetsvektor, jfr eksempelet over.

Antakelsen om at samplingsmønsteret kun avhenger av forrige tilstand (instrument) kan være tvilsom. Markovmodellen kan godt utvides slik at dette håndteres. En tilhørende TM krever da at antall tilstander økes og vi får fort en kombinatorisk eksplosjon. En større innvending mot modellen er at man i dag vanligvis viser mange informasjonstyper (for ulike funksjoner) på *en* skjerm og ikke på individuelle instrumenter.

3.5.5 Fuzzy logikk

En helt annen type modellering av hendelser er fuzzy mengder (FM) hvor man fokuserer på vage beskrivelser av hendelser - i motsetning til tradisjonell logisk beskrivelse (inntreffer/inn-

treffer ikke) i TM eller stokastiske som i Markovkjeder. Felles for alle er imidlertid at de på ulik måte kan beskrive PB oppførsel hos en operatør.

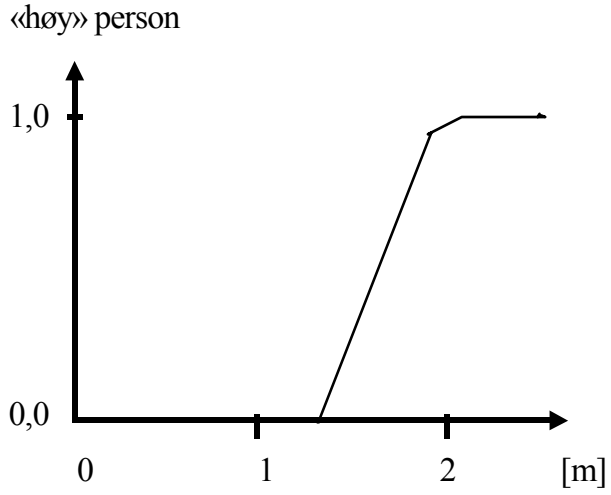
Det er mulig å skille mellom flere ulike kilder til usikkerhet:

- Unøyaktige målinger (deterministisk prosess som ikke kan måles eksakt)
- Tilfeldige hendelser (selve utfallet av prosessen er usikkert)
- Vage beskrivelser

De to første kan modelleres som stokastiske prosesser, som tidligere beskrevet. Et eksempel på den tredje usikkerheten er utsagnet «personen er høy». Usikkerheten omkring tolkningen av utsagnet «høy» krever at man innfører en konvensjon om betydningen av dette. Det er vanlig også her å velge en stokastisk modell, men et alternativ er FM som nettopp er utviklet for å kunne beskrive slike vage språklige begreper (Zadeh, 1973). For modelleringen av utsagnet A = «høy» velger vi først et verdiområde eller en verdimengde U, f eks U $\in$[1,0, 2,5] som burde dekke alle aktuelle høyder. En FM «A» i U defineres ved en medlemskapsfunksjon («membership function»):

$$\mu_A(y) : U \rightarrow [0,1] \qquad y \in U \tag{3.20}$$

Dvs. at et hvert element i U tilordnes en verdi mellom 0 og 1 som beskriver *graden av tilhørighet* til A. I motsetning til vanlig logikk og «hard» mengdelære kan et element være delvis med i en mengde. En mulig medlemskapsfunksjon er vist i figur 3.36.



Figur 3-34 Medlemskapsfunksjon for «høye» personer

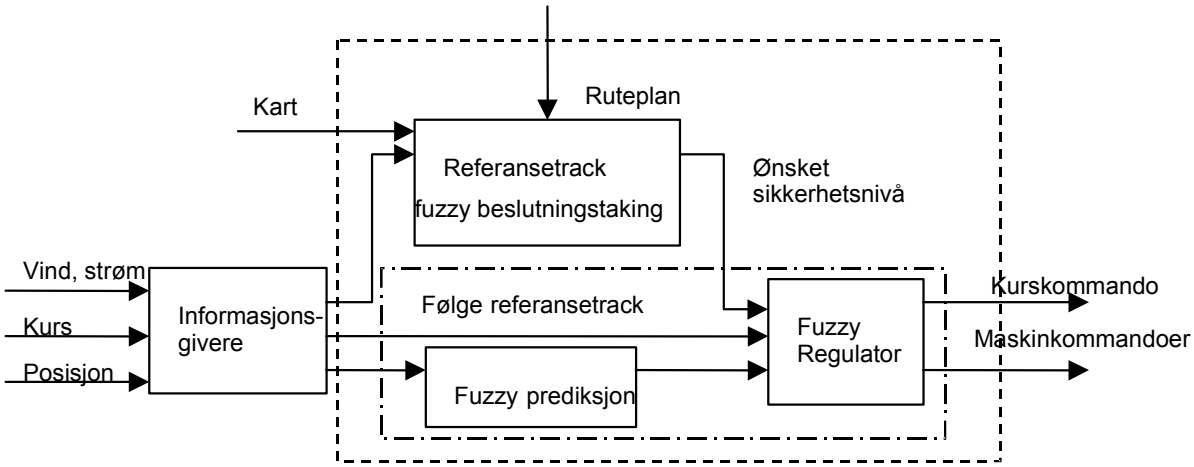
Hvilken forskjell er det mellom medlemskapsverdi og sannsynlighetsverdi 0,95 for y = 1,90? Dersom man observerer personen med eksakt nøyaktighet endres apriorisannsynligheten (0,95) til 0 eller 1 avhengig av om personen er 1,90 m eller ikke. Medlemskapsverdien for y = 1,90 er imidlertid uendret. Fuzzy mengder modellerer usikkerhet på en annen måte enn sannsynlighets-

teori. Fuzzy teori er «ortogonal» til sannsynlighetsteori mht hendelser - den fokuserer på vage beskrivelser av hendelser, istedet for usikkerheten om hendelsene vil inntreffe eller ikke.

Operatøren uttrykker ofte sin forståelse av en kompleks prosess i form av regler, på en måte som er i samsvar med FM teori. Medlemskapsfunksjonen er subjektiv og vil variere mellom operatører og ulike brukergrupper. Et eksempel på bruk av FM er en modell av navigatører på tankbåter hvor en mulig regel er:

Hvis avstand til grensen for skipsled er større enn *liten* **og** strøm er *mindre enn sterk* **så** er sikkerhet *veldig stor.*

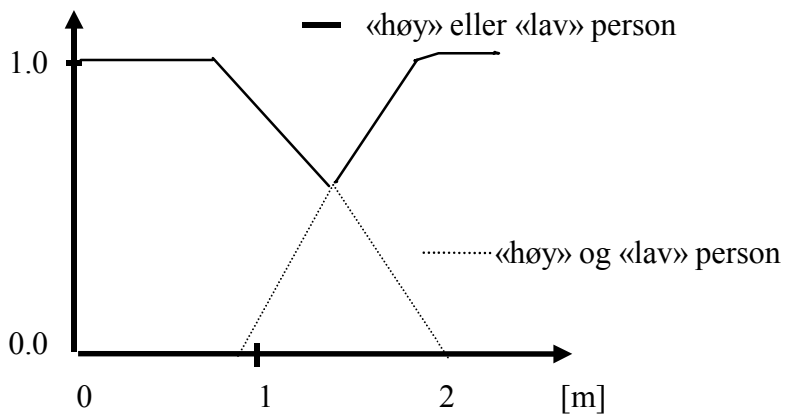
En overordnet modell for dette eksempelet er vist i figur 3.35.



Figur 3-35 Fuzzy modell av navigatør (Papenhuijzen (1985))

I FM teori har man definert tilsvarende operasjoner på mengder som i vanlig logikk. I figur 3.36 har vi innført FM «personen er lav» i tillegg til «personen er høy». Operasjonene union og snitt er i FM definert som hhv maksimum og minimum av tilhørighetsverdiene som illustrert i figur 3.36. I tillegg har man innført såkalte beregningsregler for gradsadverb, f eks *veldig* stor = stor^2 eller *litt* stor = stor^{0,5}.

Betingete utsagn er like viktig i FM som i sannsynlighetsteori (jfr Bayes regel) og tradisjonell logikk (regler). Anta f eks at følgende er gitt: **Hvis** x er stor **så** er y liten. Dette betingete utsagnet (eller regelen) definerer en fuzzy relasjon mellom mengdene til x og y. Gitt tilhørighetsverdiene til x, f eks fra utsagnet: *x er veldig lav,* kan man vha denne relasjonen utlede tilhørighetsverdiene til y. Fuzzy relasjoner er f eks definert i Zadeh (1973).



Figur 3-36 Union («eller») og snitt («og») av fuzzy mengdene lave og høye personer

En *fuzzy algoritme* er en sekvens av fuzzy utsagn som gir en tilnærmet løsning på et spesifikt problem. Fuzzy algoritmer er ofte egnet til å modellere operatørens operering av komplekse MMS. Algoritmene består av utsagn om variable, sammensatte utsagn og betingete utsagn.

En generell skisse av et fuzzy reguleringssystem er vist i figur 3.37. Vi skal kort forklare prinsippet for fuzzy regulering, eller fuzzy inferens vha. de gitte reglene (Mendel, 1995). En målt verdi av en inngangsvariabel, x', må først transformeres til en fuzzy inngangsmengde A^* (kalt "fuzzification"). Anta gitt en fuzzy regel: **Hvis** A **så** B. Da vil den fuzzy utgangsmengden B^* ha følgende tilhørighetsverdier:

$$\mu_{B^*}(y) = \max_{x \in A^*}\left[\mu_{A^*}(x) * \mu_{A\to B}(x,y)\right] = \max_{x \in A^*}\left[\mu_{A^*}(x) * \mu_A(x) * \mu_B(y)\right] \qquad (3.21)$$

der * er snittoperatoren. En vanlig måte å definere FM for inngangsvariabelen er vha "singleton fuzzifier". Da settes tilhørighet for målt verdi lik 1 og alle andre verdier lik 0, dvs:

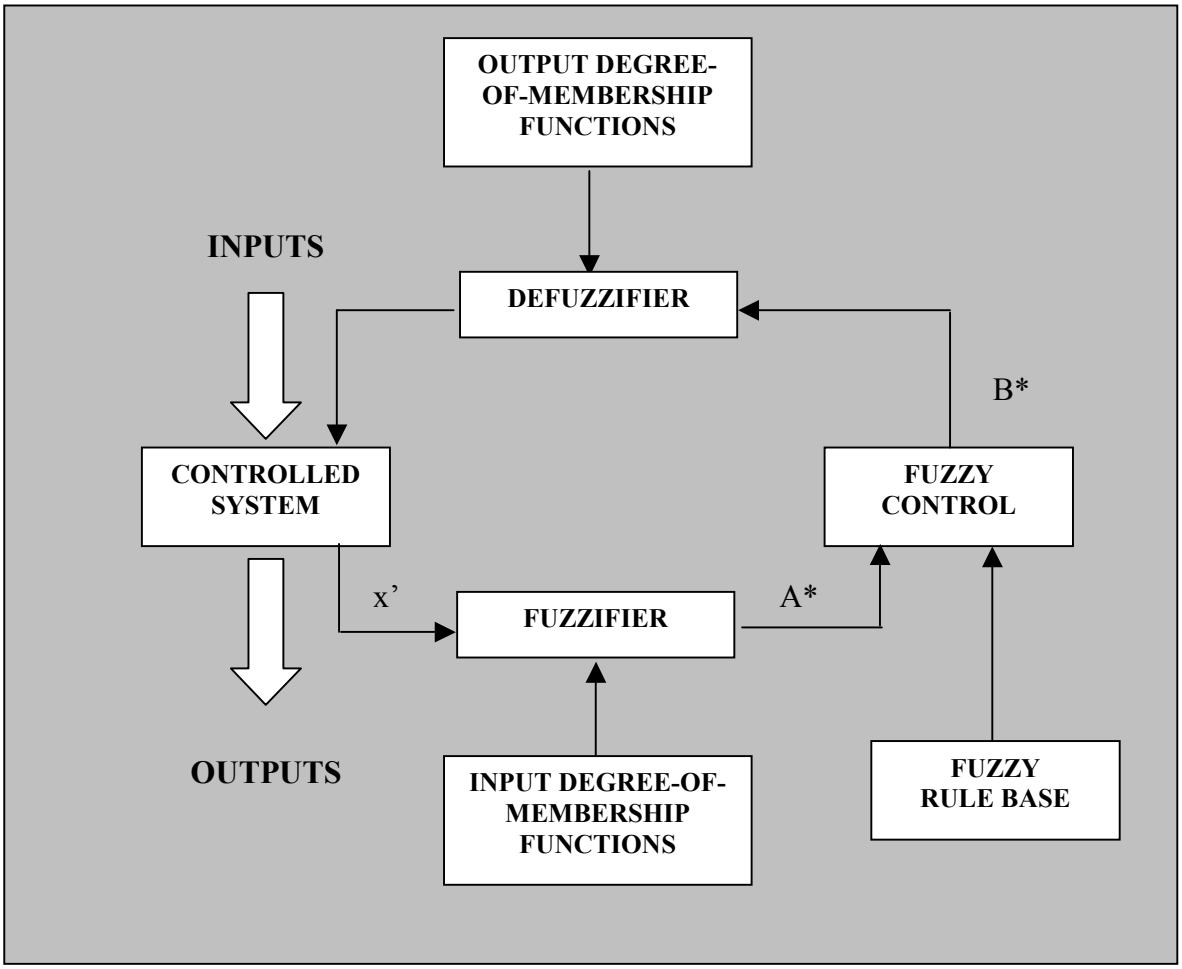
$$\mu_{A^*}(x) = \begin{cases} 1 & x = x' \\ 0 & x \neq x' \end{cases} \qquad (3.22)$$

Dersom man ønsker å modellere støy settes tilhørighet for målingen lik 1 og man lar tilhørigheten avta mot 0 etter hvert som avviket fra målingen øker.

Dersom vi benytter "singleton fuzzifier" kan beregningen av utgangsmengden forenkles:

$$\mu_{B^*}(y) = \mu_{A\to B}(x', y) = \mu_A(x') * \mu_B(y) \qquad (3.23)$$

Dvs at FM B kuttes iht tilhørigheten som målingen har med A, jfr figur 3.38.

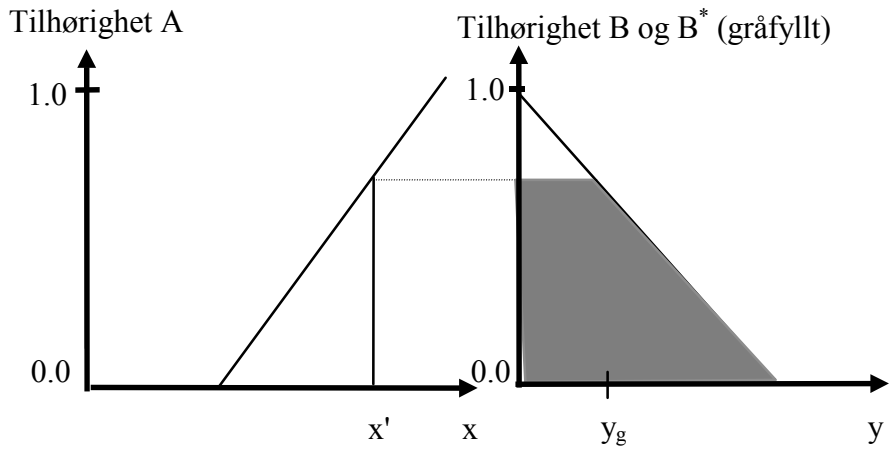


Figur 3-37 Fuzzy reguleringssystem (Brubaker, 1992)

Dersom utgangsvariabelen y skal brukes som pådrag til et reguleringssystem må vi transformere den fuzzy utgangsmengden B^* til en enkelt utgangsverdi (kalt "defuzzification"). Det er flere måter å velge denne på, f eks maksimumsverdien eller tyngdepunktet (vist som y_g i figur 3.38). Vi ser at maksimumsverdien ikke er entydig definert i dette tilfellet. Dersom vi har flere regler kan vi vekte utgangsverdiene iht tilhørighetsverdiene de har.

3.6 Modellering av sammensatte oppgaver

Ved gjennomgangen av Rasmussens modell av operatøroppførsel i kapittel 3.3 så vi at operatøren vil utvise både kontinuerlig og diskret oppførsel på ulike nivåer. Prosess og regulerings- og operatørstøttesystem vil også inneholde både kontinuerlige og diskrete elementer som illustrert i figur 3.39. MMS er altså *hybride* systemer, se f eks Grossman *et al*. (1993), dvs en kombinasjon av komponenter som kan ha både kontinuerlig og diskret oppførsel.

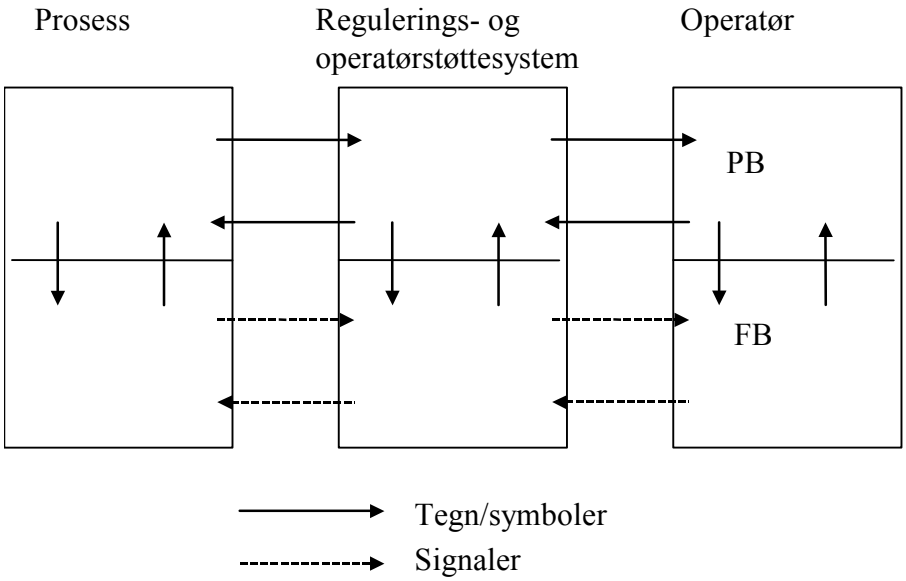


Figur 3-38 Fuzzy inferens: $\mu_{B^*}(y) = \mu_{A \to B}(x', y)$

Når vi skal lage en modell av det totale MMS er en mulighet å benytte diskrete hendelsesmodeller som basis og som et rammeverk for eventuell integrasjon av kontinuerlige modeller. Formålet med en diskret simuleringsmodell er å gjenskape (til et visst nivå) aktivitetene som enhetene vil utføre i det virkelige systemet og derved få kunnskap om potensiell oppførsel og ytelse. Dette gjøres ved å definere systemets tilstander og «konstruere» aktiviteter som bringer systemet fra tilstand til tilstand ved bestemte hendelser.

Diskrete hendelsesdrevne systemer beskriver som tidligere nevnt en stykkevis konstant tilstandstrajektor, men man kan godt erstatte hvert konstante segment med resultatet fra løsningen av differensiallikninger. Tilstandsoverganger kan f eks være relatert til en mer overordnet styring av kontinuerlige, fysiske prosesser. Dvs initiering av differensiallikningene og setting av parameterverdier. Man kan videre definere hendelser relatert til de kontinuerlige variablene i tilstanden som gir en overgang til nye tilstander, f eks at en kontinuerlig dynamisk variabel overskrider en gitt terskelverdi. På denne måten kan man lage en sammensatt modell av FB og PB oppførsel.

Et eksempel på et komplisert og hybrid dynamisk system er operasjon av en storflyplass. De viktigste objektene er fly, luftrom, rullebaner, utganger, transportkjøretøyer, parkeringsplass, passasjerer, bagasje, billett/innsjekkingsskranker, sikkerhetsporter og ulike typer operatører. I dette systemet er det mange typer av kontinuerlige og hendelsesdrevne interaksjoner mellom oppgavene og ressursene.

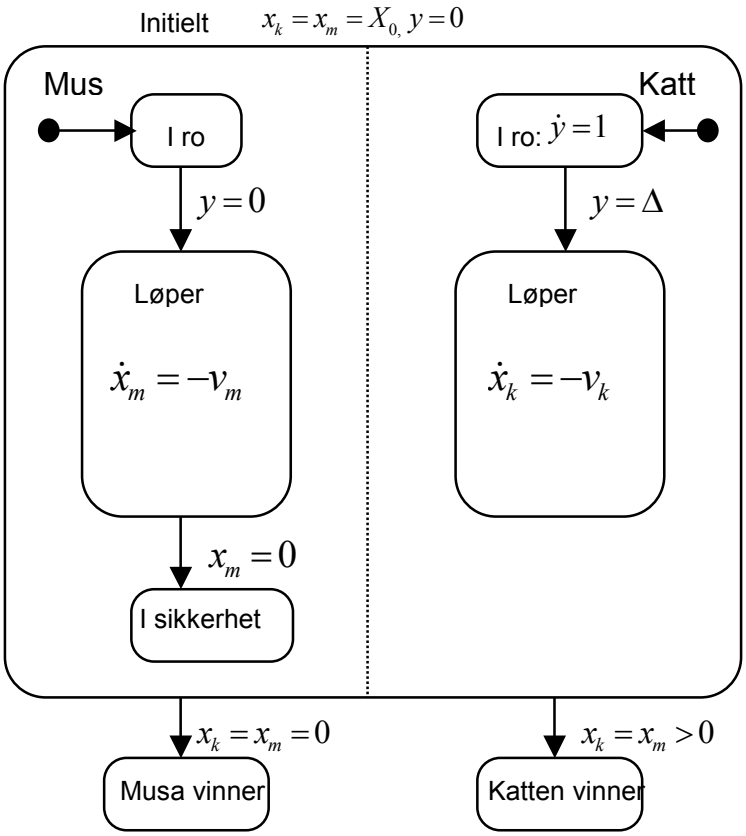


Figur 3-39 Generell struktur på modell av MMS

Tilstandsmaskiner har blitt brukt som et rammeverk for signalbehandlingsmodeller i en kombinert symbol-signal modell (Fishwick & Ziegler, 1992). På tilsvarende måte er det mulig å lage en integrert modell av operatøroppførsel der man antar at FB oppførsel styres av symbolbehandling, dvs PB oppførsel, se f eks Rouse, Hammer og Lewis (1989).

Tilstandsmaskiner har blitt brukt som et rammeverk for signalbehandlingsmodeller i en kombinert symbol-signal modell (Fishwick & Ziegler, 1992). På tilsvarende måte er det mulig å lage en integrert modell av operatøroppførsel der man antar at FB oppførsel styres av symbolbehandling, dvs PB oppførsel, se f eks Rouse, Hammer og Lewis (1989).

Tilstandskart kan også benyttes som et overliggende rammeverk. Et eksempel på bruk av hybride tilstandskart er gitt i figur 3.40. Her har man modellert en katt som jakter på en mus. Avstanden fra musa og katten til musehullet er hhv gitt ved x_m og x_c. Både musa og katten er begge initielt i sin tilstand «I ro» i en avstand X_0 fra musehullet. Musa går umiddelbart over i tilstanden «Løper». Tidsvariabelen «y» integreres opp i kattens «I ro»-tilstand og gir en transisjon over til «Løper» når $y = \Delta$. Avstandene integreres «ned» som spesifisert. Dersom musa rekker fram til musehullet ($x_m = 0$) går den over i tilstanden «i sikkerhet» og deretter til «Musa vinner» når katten kommer fram. Dersom katten i stedet tar igjen musa før den kommer fram ($x_m = x_c > 0$) får vi et avbrudd fra de parallelle «Løper»-tilstandene ut til tilstanden «Katten vinner».



Figur 3-40 Hybride tilstandskart. Spesifikasjon av «katt og mus». Fra (Maler et al., 1992)

3.7 Simulering

Som nevnt i innledningen til kapittel 3 er det i hovedsak tre teknikker for måling av ytelse: analytisk modellering, simulering og måling (i et virkelig system). Vi kan studere et system indirekte vha en modell av systemet. Dvs at man kan generere informasjon ved hjelp av modellen og deretter analysere denne. Dersom man har kompliserte og sammensatte modeller må man gjøre simuleringer på en datamaskin får å få fram resultatene. Hovedfordelen med simulering er at en tilstrekkelig nøyaktig «måling» av ytelsen ofte kan utføres innenfor tids- eller kostnadsbegrensninger. Simulering, på ulike nivåer, blir i stadig større grad brukt også innenfor MMS for avveiningsanalyser og for å validere og verifisere løsninger (lager vi det riktige systemet? og bygger vi systemet riktig iht spesifikasjonene?) (Beevis *et al.* , 1992).

Vanligvis er man interessert i ytelsesevaluering/prediksjon knyttet til kvantitative mål som tid, nøyaktighet og feil. Slik ytelsesanalyse av MMS kan ha mange ulike målsetninger. Vi kan ønske å sammenlikne systemer eller alternative delløsninger. Vi kan ønske å karakterisere arbeidsbelastningen til operatørene i systemet. Dette kan igjen brukes til å bestemme antallet operatører og kravene til dem. Ytelsesanalyse av operatører/MMS har egentlig mye til felles med tilsvarende analyse av «rene» maskinsystemer (Jain, 1991). Detaljeringsnivået i analysen kan variere fra et overordnet nivå der man f eks ser på dynamisk oppførsel relatert til aggregerte oppgaver/tilstander og til et detaljnivå der man har modellert oppførselen til operatøren og eventuelle andre komponenter i detalj. Simulering av MMS blir bl a brukt under analysen av

operatøroppgaver som beskrevet i kapittel 4.3.

Bruk av simulering er en særdeles nyttig arbeidsmetode når systemet utvikles og tilpasses sitt endelige miljø. Simulering er også svært nyttig ved opplæring og trening av operatørene og kan inngå i selve systemet eller i en separat treningssimulator. Formålet med en treningsimulator er å lære å utøve generell ferdighet og å få erfaring med sjeldne situasjoner (dvs øke hyppigheten av disse). Det siste er ofte helt nødvendig for å kunne håndtere krisesituasjoner. Tilsvarende kan man stimulere en modell av MMS under utviklingen og analysere systemets reaksjonsevne i normale og mer ekstreme situasjoner.

Et diskret hendelsesdrevet simuleringsprogram består av mange kvasi-parallelle prosesser (aktive objekter). Hver prosess har assosiert en tid som angir når den skal aktiveres og som derved definerer dens plass i en hendelsesliste. Simuleringstiden oppdateres i diskrete hopp hver gang en hendelse inntreffer. Administrasjon av denne hendelseslista og aktiveringen av prosessene er vanligvis en implisitt del av simuleringsspråket, som f eks den innbygde klassen «Simulation» i SIMULA (Birtwistle *et al.,* 1973). I dette språket er det tre basismekanismer for å kontrollere prosessaktiveringen; «activate», «passivate» og «hold».

Det finnes mange spesialiserte simuleringsspråk/verktøy, men generelle programmerings-språk som SIMULA eller C++ blir likevel ofte brukt. Spesialiserte simuleringsverktøy har vanligvis innebygget mekanismer for innsamling av resultater, generering av statistikk og grafisk presentasjon av resultatene. Det er utviklet egne språk som er egnet for simulering av hybride systemer. Det finnes også noen språk som er spesielt rettet mot MMS. Et av de mest kjente av disse er SAINT («Systems Analysis and Integrated Networks og Tasks») (Chubb 1981, Beevis *et al*. 1992, Wortman, 1978). Vi skal kort beskrive dette språket og gi noen eksempler på bruken av det.

SAINT er et hybrid simuleringsspråk som spesifikt ble utviklet for å simulere operatøroppgaver beskrevet av et nettverk av diskrete deloppgaver. Operatøroppgaver, eller PB aktiviteter, modelleres som et nettverk av noder og grener der nodene representerer deloppgaver som tar en viss tid. Grenene representerer ordningen av oppgavene i tid. Beskrivelsen er hierarkisk, dvs hver oppgave kan beskrives i større detalj av et eget nettverk. For hver oppgave må man i SAINT spesifisere tidsforbruket og hvilke oppgaver som skal utføres umiddelbart etterpå. Man kan også spesifisere oppstartbetingelser og avslutningseffekter, f eks at en bestemt ressurs må være tilgjengelig før oppgaven utføres og at denne frigis når oppgaven er utført. Ressursen kan være operatøren selv eller ressurser knyttet til hans kapasitet til informasjonsbehandling. F eks er det aktuelt å modellere at han kun kan utføre en kognitiv oppgave om gangen. Tidsforbruket angis ofte stokastisk med en fordelingstype og parametere for denne. Valg av påfølgende oppgave(r), dersom det er flere muligheter, kan være av typen:

- Stokastisk, dvs sannsynligheten for initiering av neste oppgave er spesifisert, jfr Markovkjeder.
- «Taktisk», dvs neste oppgave velges iht verdien av en variabel som resultat av informasjonsbehandlingen (i oppgaven)

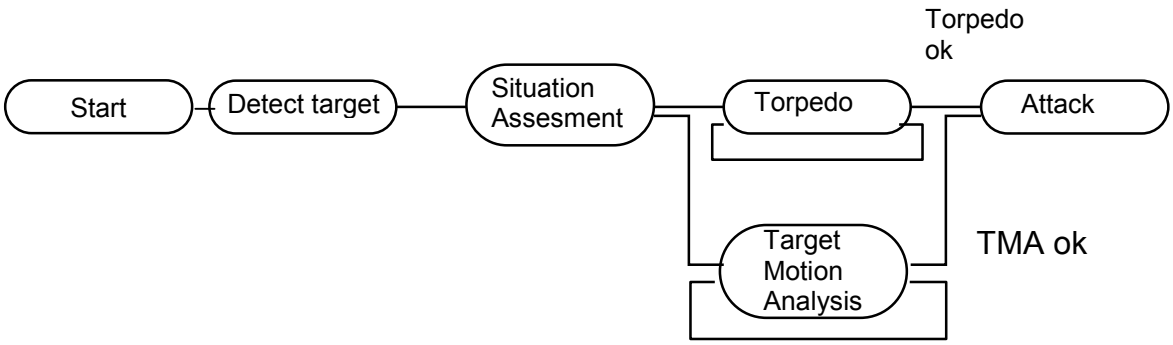
- Multippelt, dvs at en eller flere oppgaver velges iht verdier av variable som resultat av informasjonsbehandlingen

Flere oppgaver kan altså utføres i parallell. For å kunne modellere informasjonsbehandling og valg kan brukeren skrive et (Fortran) program for hver oppgave.

I SAINT kan man ikke modellere avbrudd (fullstendig eller midlertidig) av en påbegynt oppgave, se f eks diskusjon i (Nordø & Bråthen, 1994). Dette er en svært relevant problem-stilling når PB og KB oppførsel skal modelleres.

Et SAINT nettverk er vist i figur 3.41. Eksempelet modellerer på et overordnet nivå oppgavene i et kampledelsessytem for undervannsbåter (Skare & Nordø, 1990). Sjefsoperatøren styrer arbeidet, fordeler oppgaver og mål (som skal overvåkes, eventuelt angripes) til de andre tre operatørene og utfører selv en del oppgaver (assosiasjon, klassifikasjon og trusselanalyse). En viktig problemstilling i denne studien var arbeidsbelastningen til sjefsoperatøren. Det er nødvendig at han har ledig kapasitet slik at han også får tid til mer langsiktig taktisk planlegging (en oppgave som ikke er med i den mer "prosedyre-baserte" SAINT-modellen). Ved å delegere noen av oppgavene til sjefsoperatøren (assosiasjon og klassifikasjon) til de andre operatørene fikk han en mer akseptabel arbeidsbelastning.

SAINT er f eks også blitt brukt til å analysere arbeidsbelastningen til flygeren i et angrepshelikopter (Laughery, 1986). Fire alternative utforminger av cockpiten med ulik grad av automatisering ble simulert og sammenlignet. Modellene omfattet ca 200 deloppgaver. Arbeidsbelastningen ble karakterisert ved å se på kapasitetskrav («attentional demands») i 4 kanaler: visuell, audio, motorisk og kognitiv. Hver oppgave i nettverket krevde en viss del av disse 4 typene av kapasitet. Siden operatøren ofte utførte flere oppgaver samtidig ga simuleringen de samlede krav til kapasitet som funksjon av tid. Ved å sette terskler på operatørens kapasitet kan man simulere at operatøren ikke får utført visse oppgaver. Dette ble igjen relatert til et sårbarhetsmål (evne til å overleve trussel).



Figur 3-41 SAINT oppgave nettverk for et kampledelsessystem for undervannsbåter (Skare og Nordø, 1990)

Et annet eksempel på bruk av SAINT er modellering av snorklingsprosedyrer for en ubåt

(Breda, 1989). Snorkling er å suge inn luft via en mast til dieselmotorene for å kunne lade de elektriske batteriene som nyttes når ubåten er helt neddykket. Prosedyren ble modellert som et nettverk av operatøroppgaver og en detaljert modell av dynamikken til ubåten. Bruk av SAINTs mulighet til å inkludere en kontinuerlig modell av den fysiske prosessen var altså helt vesentlig for denne problemstillingen. Man fokuserte på måling av følgende aspekter:

1. Kvaliteten av prosedyren. Dette ble målt som maksimum høyde av snorkelmast over sjøen etter at man dykket opp. For å minimalisere muligheten for deteksjon ønsket man en høyde under en gitt verdi.
2. Tiden det tok å utføre prosedyren. Man ønsket å minimalisere denne.
3. Tiden operatøren var opptatt. Av forskjellige årsaker ønsket man at denne skulle være mindre enn 50 %.

Analysen viste bl a at kravet relatert til punkt 1 over var umulig å oppnå og under hvilke betingelser man kom nærmest kravet. Årsaken til at man ikke oppnådde kravet var forstyrrelser i båtens balast (dvs den fysiske prosess) og egentlig ikke relatert til operatørens oppførsel eller automatiseringsnivå.

Et siste eksempel på bruk av SAINT er for å studere bemanning på et marinefartøy (Wetteland *et al.*, 2000).

Beskrivelser/modeller som benyttes under system- og programvareutviklingen kan i en viss grad brukes til simulering av MMS. Vanligvis vil ikke slike språk ha eksplisitte mekanismer for modellering/simulering av kontinuerlig oppførsel. Det kan likevel være mulig å beskrive beregninger tilsvarende diskrete differenslikninger, men dette er ofte tungvint og gjør den diskrete hendelsesmodellen mer komplisert og mindre lesbar. Mulighetene til å beskrive informasjonsbehandling og beslutninger relatert til sekvenseringen av oppgaver kan også være begrenset eller tungvint. Disse aspektene er diskutert i (Bråthen, Nordø & Veum, 1994). En overordnet systemmodell er kanskje best egnet som et *rammeverk* for en simuleringsmodell som realiseres i et tradisjonelt programmeringsspråk. Rammeverket kan eventuelt genereres automatisk. En slik framgangsmåte kombinerer det å sikre konsistens mellom spesifikasjon og simuleringsmodell med den fleksibiliteten som et generelt programmeringsspråk gir.

4 SYSTEM-KRAVSETTING OG SPESIFIKASJON

> *Hvis en bedrift tar sikte på maksimal automatisering (innen lønnsomhetens grenser naturligvis), er det nødvendig å gå gjennom hele produksjons-prosessen på nytt, og da må spørsmålet være: Kan dette gjøres på en annen måte som ikke bare i seg selv er bedre, men som samtidig er bedre egnet for automatisk håndtering.*
> Kjell Holler*: Automatisering - spøkelse eller realitet (1957)*

I dette kapitlet skal vi beskrive den analytisk delen av systemarbeidsmodellen i figur 2.3 der

man spesifiserer og beskriver *hva* MMS-et skal gjøre. Denne delen omfatter etablering av krav, analyse av overordnete systemfunksjoner (oppførsel) og allokering (fordeling) til ulike hovedkomponenter (maskin og operatører). Videre beskrives analyse av operatøroppgaver.

4.1 Etablering av krav

Denne aktiviteten omfatter:

- arbeid med systemmålsetning og bestemmelse av overordnede krav til MMS
- scenarioanalyse
- operatøranalyse
- analyse av eksisterende systemer
- overordnet systemløsning (identifikasjon av de viktigste systemkomponentene og toppnivå funksjoner)

Overordnede krav. Det å komme fram til *overordnede krav* (operative krav) til MMS kan involvere en eller flere av følgende aktiviteter:

- innsamling av bakgrunnsmateriale
- innsamling av foreliggende krav
- etablering av kontakt med brukere og andre medvirkende parter
- systematisering og formalisering av krav
- sikre sporbarhet av krav for videre utvikling

I (Hsia *et al.*, 1993) defineres kravsetting («Requirement engineering») slik: «The disciplined application of proven principles, methods, tools and notations to describe a proposed system's intended behaviour and its associated constraints». I artikkelen hevdes det at man idag har en god forståelse av hvordan man skal skrive en rimelig komplett, konsistent og utvetydig kravspesifikasjon, *men* at dette generelt sett ikke blir gjort! Det hevdes at det er et gap mellom praksis og «state-of-the-art» og trekker fram følgende problemområder:

- det er vanskelig å få fram krav
- krav endres ofte underveis i utviklingen
- metoder anvendes feil og man har for stor tiltro til dataverktøy for system-/krav-beskrivelser
- opplæring i kravsetting er vanskelig
- avsatt tid til kravsetting i et prosjekt er nesten alltid for knapp
- utviklere har manglende tiltro til betydningen av kravsetting
- kommunikasjonsbarrierer mellom utviklere og brukere

Betydningen av å ha en større innsats i den initielle fasen ble påpekt i kapittel 2. Vi understreket også ganske sterkt nytteverdien av prototyping ved etablering av krav til MMS. I dette kapitlet skal vi konsentrere oss om beskrivende teknikker og utviklingen av en overordnet

systemmodell.

Scenarioanalyse. Scenarioanalyse (eng. mission/scenario analysis) kan være en svært god teknikk for å identifisere de riktige kravene. Man kan kanskje gå så langt som å si at det er den eneste praktiske måte å oppnå tilnærmet kompletthet i spesifikasjonene! Et scenario beskriver systemets oppførsel over tid og dets samspill med omverdenen for *ett spesifikt* hendelsesforløp over et gitt tidsrom. Beskrivelsen gir en god bakgrunn for å bestemme hovedfunksjonene i systemet. Scenariene kan være utgangspunkt for prototypevalueringer og akseptansetester av det endelige system.

Scenarioanalysen kan benytte samme beskrivelsesform som funksjonsanalysen (jfr neste delkapittel). Generelt kan man skille mellom en grafisk og en tekstlig scenariobeskrivelse. Eksempel på grafiske beskrivelser kan være en skisse av banen til et fly fra det tar av til det lander og hvilke eksterne hendelser som inntreffer underveis. Et annet eksempel er ruten til en båt opptegnet på et sjøkart der man også har tegnet inn viktige navigasjonshendelser pga farvann, vær eller annen trafikk. Hendelser kan også knyttes til tidsplott av fysiske variable, f.eks. fart. Under scenarioanalysen bør man prøve å identifisere typiske faser eller moder, f eks normal operering og alarm-/nød-modus.

I praksis er man nødt til å begrense omfanget av beskrivelsen, dvs antallet scenarier og deres detaljeringsgrad. Et typisk antall scenarier er i størrelsesorden 10-15. Som en generell retningslinje vil man ved evalueringer og tester av systemet sjelden ha ressurser til å benytte et særlig større antall scenarier. En forutsetning er at scenariene spenner ut rommet som inneholder alle mulige oppførsler til systemet på en fornuftig måte. Dvs at scenariene fokuserer på både normal og maksimal belastning, og både normal (feilfri) og sterkt feilbefengt oppførsel.

To problemer man ofte møter i praktisk anvendelse av scenarioanalyse er (Weidenhaupt et al. 1998).

- å finne riktig abstraksjonsnivå, dvs. hvor detaljert skal hendelser og funksjoner beskrives?
- å bevare konsistens mellom krav og scenarier, og mellom ulike scenarier som bruker de samme funksjonene. En endring i et scenario medfører ofte behov for å endre tilsvarende krav eller andre scenarier.

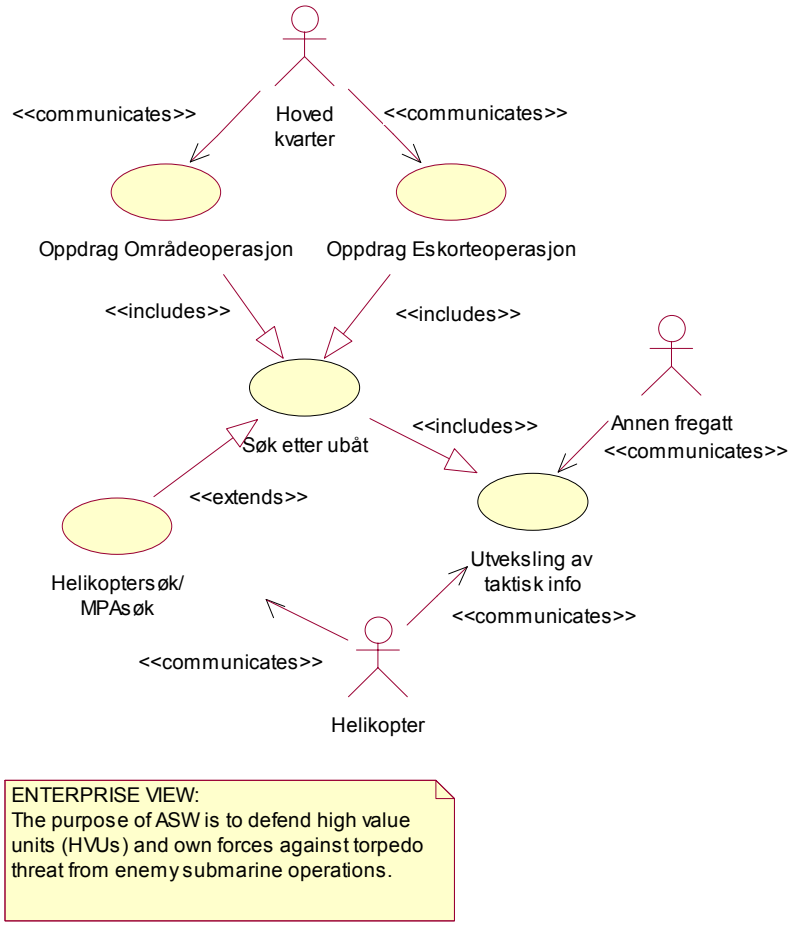
Såkalte *brukstilfeller (eng: use-cases)* er en metode for å strukturere scenarier på. De brukes ofte for å vise hvordan et system brukes i en større sammenheng med andre aktører. Hvert brukstilfelle modellerer typiske interaksjoner mellom systemet og disse aktørene. Dette er en av ni diagramtyper i modelleringsspråket Unified Modelling Language (UML) (Booch, Rumbaugh og Jacobson, 1999). Et brukstilfelle representeres av en ellipse og aktører med strekmenn (se figur 4.1). Følgende spørsmål vil typisk brukes for å identifisere brukstilfeller:

- Hvilke oppgaver har aktørene?
- Hvilke hendelser og handlinger trenger aktørene å utveksle informasjon om?

I tillegg til brukstilfellediagrammer må flyten av hendelser i hvert brukstilfelle beskrives. Denne beskrivelsen kan typisk organiseres i følgende avsnitt:

1. Betingelser som må oppfylles for at brukstilfellet skal kunne utføres.
2. Hovedsekvensen av hendelser og handlinger.
3. Subsekvenser.
4. Alternative sekvenser.

Det vil alltid være flere måter å utføre en gitt brukstilfelle. Ett spesielt gjennomløp utgjør et scenario. I UML kan et scenario vises ved hjelp av interaksjonsdiagrammer som kommer i to typer: sekvensdiagram (se figur 3.20) og samarbeidsdiagram.



Figur 4-1 Brukstilfellediagram for beskrivelse av anti-ubåtkrigføring

Operatøranalyse. Det å foreta en såkalt *operatøranalyse* helt i starten av systemutviklingen er en relativt ny idé. Formålet er å etablere en *operatørprofil* som karakteriserer *brukerpopulasjonen* som skal benytte MMS-et. Et viktig poeng i denne sammenheng er at kunden ikke nødvendigvis er representativ som sluttbruker. Som utvikler har man lett for å anta en

brukerprofil, f eks seg selv (i operatørrollen). En operatørprofil kan bestå av:

- alders- og kjønnssammensetning
- utdannelsesbakgrunn
- seleksjon og trening
- tidligere erfaring med databaserte MMS
- tidligere erfaring med tilsvarende systemer (negativ/positiv overføringseffekt)
- prosessforståelse
- kognitiv stil, personlighetsmål
- bruksfrekvens
- brukerforventning

Kognitiv stil kan f eks omfatte hvorvidt man har en analytisk/systematisk eller en mer prøve/feile-type arbeidsstil og hvorvidt man jobber mest sekvensielt/prosedyremessig eller mer parallelt/assosiativt.

Ofte kan det være stor variasjon innenfor brukergruppen. Kunnskap om denne variasjonen kan være like viktig som selve operatørprofilen. Teknikker for å utføre operatøranalyse er i stor grad kvalitative og er generelt mindre utviklet enn for scenario- og funksjonsanalyse. Hovedgrupper av slike teknikker er:

- Intervjuteknikker
- Observasjonsteknikker (høyttenkning, verbal protokollanalyse)
- Spørreskjemaer
- Simuleringsteknikker og spill (f eks vha forenklede scenarier)

Disse teknikkene benyttes typisk også ved evaluering og prototypetester, og under kunnskapsinnsamling ved utvikling av ekspertsystemer.

Analyse av eksisterende system. Det finnes nesten alltid et eksisterende system som er noenlunde sammenlignbart med det man vil utvikle. Dette bør man dra nytte av ved å spørre seg: Hva er bra og hva er dårlig med det eksisterende systemet? Toppnivåfunksjonene i samme type systemer er ofte ganske like. Det er imidlertid viktig at man ikke låser seg til gamle krav og løsninger, men at man analyserer hele konseptet på nytt. Dette sikrer at man kan få nye og mer egnete løsninger («revolusjon!») og ikke bare tilpasninger av eksisterende løsninger («evolusjon»). I denne sammenheng er det verdt å merke seg at operative krav påvirkes av forventninger til den teknisk utvikling. Derfor er det egentlig misvisende å si at tekniske krav kan avledes direkte av operative krav. En hovedutfordring ved systemkravsetting er derfor å sikre en rimelig balanse mellom behovsdrevet og teknologidrevet utvikling.

Systemløsning. I den innledende fasen er det viktig å definere grensen og omgivelsene til MMS

som nevnt i kapittel 3.1.2. Deretter utarbeides den overordnede systemløsning som omfatter toppnivåfunksjoner og de viktigste systemkomponentene. Funksjonene identifiseres typisk fra systemoppførsel knyttet til scenariene. Ideelt sett samles alle scenariene i en total oppførselsmodell av MMS (som beskriver alle mulige oppførsler). Videre beskrives de øverste nivåene i MMS-ets struktur-hierarki og den tilhørende konnektivitet i komponentmodellen. Viktige attributter til komponentene beskrives.

Systematisering av krav. Vi vil her gi en kortfattet beskrivelse av kravsetting ved materiellanskaffelser til Forsvaret i Norge iht systemarbeidsmodellen Prinsix som vi nevnte i kapittel 2. Prinsix forutsetter utarbeidelse av to dokumenter tidlig i konseptfasen:

- TOØM (Taktisk organisatorisk og økonomisk *målsetning*)
- KD 1 (Kravdokument 1, som er en foreløpig taktisk, teknisk og økonomisk *kravformulering*)

En "gjennomførbarhetsstudie" av KD 1 blir så utført og man foretar et valg av konseptuell systemløsning. Arbeidet formaliseres til slutt i et dokumentet som kalles kravdokument 2 (KD2) som er utgangspunktet for definisjonsfasen. I definisjonsfasen vil det være aktuelt å starte utvikling av prototyper og å foreta ulike typer av simuleringsstudier av systemalternativer der man ser på avveining mellom kostnader (anskaffelse og drift) og ytelse i stort.

Tekstbehandling er vanligvis et tilstrekkelig verktøy under utarbeidelsen av disse dokumentene. I Prinsix har man definert maler for dokumentene som beskriver innholdsfortegnelsen i detalj og som gir veiledning om hva som skal beskrives. For store systemer kan det være aktuelt å benytte en enkel database. Med utgangspunkt i kravene fra KD2 utarbeides en rekke forskjellige typer spesifikasjonsdokumenter som til slutt utgjør kontraktsgrunnlaget når leverandører velges.

Formatet på disse dokumentene er basert på ulike militære standarder (f eks Mil Std 490 og 2167). Beskrivelsen av dokumentene er i en viss grad formalisert og man bruker typisk en database. Dokumentene lages da vha automatisk rapportgenerering. Verktøyet kan være en enkel applikasjon for en generell database eller et mer omfattende systemutviklingsverktøy (som også kan benyttes til funksjons- og operatøroppgave-modelleringen). Typiske attributter for et krav er:

- unikt navn/nummer (identifikasjon)
- (tilnærmet) strukturert tekst (f eks konvensjoner mhp bruk av må/skal/bør)
- verifikasjonskrav (test, simulering, inspeksjon, etc)
- referanse til kildedokumenter (f eks TOØM, KD2)
- ansvarlig person

Spesfikasjonene vil kunne inneholde referanser til scenarier. Utvikling av detaljspesifikasjonene krever at man foretar en funksjonsanalyse og en allokering til komponenter. Dette er tema i neste delkapittel.

Verifikasjonskrav bør spesifisere:

1. Hvordan kravet skal verifiseres. Dette kan f.eks. gjøres ved test (eksperiment), analyse eller simulering av en matematisk modell av systemet, eller ved inspeksjon (f.eks. for å fastslå at alle komponenter er tilstede og riktig installert).
2. Miljøparametre og eksterne faktorer som påvirker resultatet av en test eller analyse.
3. Beskrivelse av eventuell behandling av måleresultater som f.eks. statistiske prosedyrer (f.eks. gjennomsnitt eller dårligste verdi av et gitt antall målinger).
4. Kriterium som må oppnås for at kravet skal godkjennes som oppfylt. Dette kan f.eks. være definert som en hard grense eller et konfidensintervall for en ytelsesparameter.

Eksempel: Kravsetting for et delsystem i et marinefartøy - Kampledelsessystem. Vi vil illustrere kravsettingsaktiviteten med et eksempel fra et sjømilitært system. En fregatt er et middels stort marinefartøy (ca 130 meter langt) som har en rekke oppgaver innen sjøforsvarsoperasjoner. Disse kan grovt deles inn i luft-, overflate- og undervannskrigføring. Alle typer operasjoner understøttes av et kampsystem som kan deles inn i fem hovedkomponenter:

1. Navigasjonssystem
2. Sensorsystem
3. Kampledelsessystem
4. Våpensystem
5. Kommunikasjonssystem

Et mulig scenario for dette systemet kan være som følger:

En norsk marinestyrke bestående av to fregatter (NF1 og NF2) med helikoptre skal avsøke et område for å sikre gjennomfarten for en konvoi. Man har etterretning om at en fiendtlig dieselselektrisk ubåt er observert 60 nm vest for det aktuelle området.

Under transitt til området planlegger Officer in Tactical Command (OTC) (på NF1) operasjonen og gir forberedende ordre til NF2. Søkeoperasjonen starter kl 0800. Kl 1109 får NF2 kontakt med en mulig ubåt på tauet antenne sonar. Kontakten befinner seg utenfor egen våpenrekkevidde. Et helikopter sendes ut for å undersøke kontakten nærmere mens NF2 informerer NF1 og prøver å opprettholde egen kontakt på sikker avstand. Helikopteret vektoreres mot kontakten basert på NF2s måldata. Kl 1137 oppnår helikopteret kontakt og klassifiserer kontakten som PROBSUB (sannsynligvis en ubåt). Dette tilfredsstiller engasjementskravene. Helikopteret heiser derfor opp sonaren umiddelbart, forflytter seg nærmere siste observerte kontakt og avfyrer en lettvektstorpedo kl 1141. Helikopteret avventer effekten av angrepet samtidig som den forbereder avfyring av neste torpedo. Kl 1144 observeres en eksplosjon og vrakrester. Helikopteret sender skadevurderingsrapport og returnerer til moderskipet.

Et eksempel på en undermengde av brukstilfellene for et fregattsystem er vist i figur 4.1. Andre brukstilfellediagrammer kan typisk vise alle brukstilfellene for en av aktørene eller et brukstilfelle med alle dets relasjoner.

For eksempel kunne brukstilfellet *SØK ETTER UBÅT* beskrives som følger (her har vi ikke tatt med detaljerte beskrivelser):

1 Betingelser.
1.1 B-1. Skipet må ikke ha fått ordre om å operere stille.
2 Hovedsekvens. Hvis det oppstår sonarinterferens utfør A-1, ellers
2.1 H-1. Manøvrering for å dekke ønsket område
2.2 H-2. Still inn sonarparametre hvis nødvendig (S-1, S-2 eller S-3).
2.3 H-3. Analyser sonarekko.
2.4 H-3. Rapporter sonarkontakt.
3 Subsekvenser.
3.1 S-1. Still inn sonareffekt.
3.2 S-2. Still inn sonarsektor.
3.3 S-3. Still inn tidsintervall for sonarping.
4 Alternative sekvenser
4.1 A-1. Velg sonarfrekvens for å unngå interferens med andre aktører.

Her skal vi konsentrere oss om kampledelsessystemet på fregatten. Dette er en systemkomponent som håndterer følgende hovedfunksjoner: informasjonsbehandling, situasjonsvurdering, beslutningsstøtte, ressurshåndtering og gjennomføring av planer.

Noen eksempler på krav til kampledelsessystemet er vist i tabell 4.1. Et eksempel på et *overordnet krav* som legger sterke begrensninger på utformingen av dette systemet er krav A. Siktemålet med dette ikke-funksjonelle kravet er bl.a. å begrense driftskostnadene over hele fartøyets levetid. De følgende kravene (B-F) er eksempler på noen av kravene som spesifiserer funksjonaliteten i kampledelsessystemet.

A	Kampledelsessystemet skal gjøre det mulig å utføre alle nødvendige funksjoner i operasjonsrommet med maksimum 15 mann.
B	Kampledelsessystemet skal ha funksjoner for editering og fremvisning av plandata i tekstlig og i grafisk form.
C	Kampledelsessystemet skal kunne utføre dekningsberegninger for hvert objekt i det taktiske bildet. Beregningene skal omfatte våpen-, sensor- og kommunikasjonsdekning.
D	Kampledelsessystemet skal kunne foreslå engasjementsplaner for eget skips våpen slik at nøytraliseringssannsynligheten for et utvalgt mål optimaliseres.
E	Kampledelsessystemet skal ha funksjoner for beregning av vektorering av helikopter og maritime overvåkningsfly.
F	Kampledelsessystemet skal ha funksjoner for automatisk generering av engasjementsordre fra godkjente engasjementsplaner i henhold til standard x.

Tabell 4.1 Eksempler på krav til et kampledelsessystem

Verifikasjonskravet for krav E kan formuleres som følger:

Verifikasjonsmetode: Et simulert helikopter-track og ubåt-track beveger seg i henhold til et

forhåndsdefinert scenario. Ved et tidspunkt valgt av en operatør beregner kampledelsessystemet en vektor for helikopter-tracket for avskjæringskurs mot ubåt-tracket.

Akseptansekriterium: Beregningsresultatet stemmer i peiling og avstand med det predefinerte scenariet.

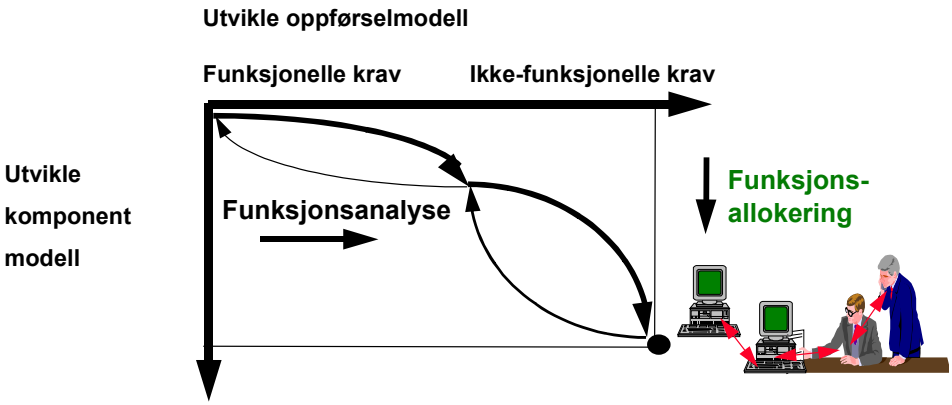
Dette kravet verifiseres ved hjelp av en simulering. Av kostnadsgrunner velger man ofte å redusere antall tester som skal utføres i en operativ sammenheng til et minimum.

4.2 Funksjonsanalyse og allokering

En introduksjon til dette temaet ble gitt i kapittel 3.1.4. Med basis i kravene og rammene som den overordnete systemløsning gir, detaljeres og beskrives oppførselen til systemet. Den tradisjonelle betegnelsen som blir brukt i systemteknikk er *funksjonsanalyse* og denne omfatter:

- detaljering og dekomponering av oppførsel (systemets funksjoner)
- fastsettelse av automatiseringsfilosofi og -nivå
- fordeling (allokering) av funksjoner mellom operatør og maskin

Figur 4.2 beskriver utviklingen (modelleringen) av et system i denne fasen.



Figur 4-2 Oppførsels- og komponentmodellen utvikles i parallell

Under funksjonsanalysen dekomponeres oppførsel og krav i en oppførselsmodell. Når systemets oppførsel analyseres og beskrives vil man implisitt også gjøre det samme for oppførselen til «objekter» i *omgivelsene* til systemet. Det kan derfor være fornuftig eksplisitt å beskrive og inkludere overordnet oppførsel til omgivelsene i oppførselsmodellen. I et MMS er det videre viktig å inkludere beskrivelser av de funksjonene som utføres manuelt av operatøren(e). Disse blir ofte ikke beskrevet i de funksjonelle kravene (Beevis, 1994). Eksempler på slike funksjoner er: overvåkning (av hverandre), rettledning, konsultering og opplæring.

I parallell med utviklingen av oppførselsmodellen detaljeres beskrivelsen av komponent-modellen. Det er ofte nødvendig ganske tidlig å ta hensyn til systemkomponentenes egenskaper for å kunne foreta en meningsfylt funksjonsanalyse. De viktigste kategoriene av systemkomponenter er operatører, maskin- og programvare.

Oppførsel (funksjoner) tilordnes komponenter, og dette betegnes som *funksjonsallokering*. I system-/programvareteknikk betraktes vanligvis dette som implisitt i konstruksjonsprosessen. Innenfor MMS omtales ofte funksjonsallokering som et av hovedstegene i systemutviklingen og en rekke teknikker har blitt foreslått. Vi skal senere beskrive noen av disse. Det vil alltid være gitt rammer for mulige allokeringer, f eks vil det som regel ikke være aktuelt å automatisere overordnete beslutninger. I militære systemer vil det typisk være rigide beslutningshierarkier som definerer hva en operatør kan gjøre og hva han ikke kan gjøre. Siden funksjoner utveksler informasjon, vil allokeringen implisitt etablere krav til grensesnitt mellom komponentene hva gjelder kapasitet, forsinkelse, osv. Hovedtypene av grensesnitt er menneske-menneske, menneske-maskin og maskin-maskin. Det vil alltid være kommunikasjon mellom operatører. Et menneske-menneske-grensesnitt (dvs ikke via dataskjermer) er ofte best egnet dersom operatører har innbyrdes kommunikasjonsbehov relatert til kunnskapsbasert oppførsel.

Evne og kapasitet til å behandle informasjon er selvsagt helt avhengig av typen komponent, men det vil alltid være visse begrensninger. Man må derfor sjekke hvorvidt ytelseskrav assosiert med allokert oppførsel kan oppfylles av komponentene og systemet som helhet. Man får derfor også nye krav mhp administrasjon av begrensete ressurser, f eks dersom belastningen i perioder er (for) høy.

Virkelige komponenter kan feile på ulike måter og dette må analyseres. Man får igjen nye krav til feildeteksjon, feilkorreksjon, redundans og vedlikehold. Betydningen av menneskelige feilhandlinger når man skal velge automatiseringsgrad og foreta allokering diskuteres nærmere i kapittel 4.2.2. Hovedformålet med feilhåndtering er å bringe systemet tilbake til «normal» oppførsel.

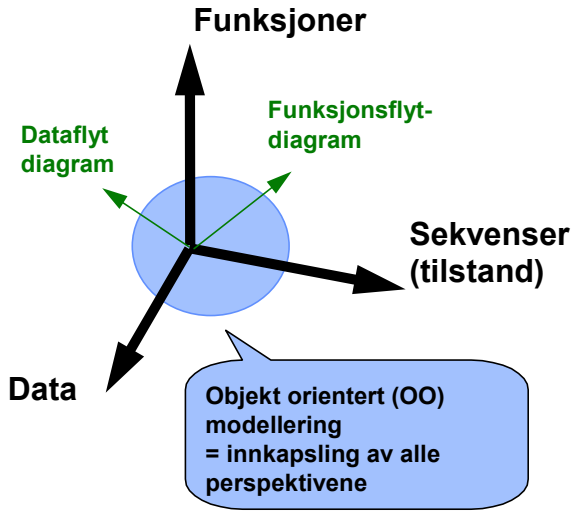
Allokering av oppførsel til operatører medfører også ulike typer nye krav, f eks opplæring og treningssystemer.

Alle disse kravene resulterer i *ny* oppførsel som må beskrives og dekomponeres, og deretter igjen allokeres til komponentene. Disse kravene er i figuren kalt *ikke-funksjonelle* (fordi de i liten grad beskriver funksjoner motivert av overordnete krav). Systemutvikling er altså en iterativ prosess som (til en viss grad) kan kontrolleres vha ulike typer mulighetsanalyser. Dersom en løsning ikke er mulig (teknisk, tidsmessig, økonomisk, osv) må man gå et skritt tilbake i modelleringen, som illustrert i figur 4.2 og prøve nye alternativer. De to viktigste teknikkene som brukes under mulighetsanalyser er simulering og prototyping (på ulike nivåer). Selv om krav til operatørgrensesnittet vanligvis klassifiseres som "funksjonelle" fordi operatøren har en helt nødvendig og selvsagt rolle i de fleste systemer, er altså slike krav et resultat av den valgte allokering og ikke nødvendigvis direkte relatert til den egentlige

systemoppførsel. Arbeid med menneske-maskin-kommunikasjon (MMK) og funksjoner relatert til ikke-funksjonelle krav, f eks feilhåndtering, kan ofte representere mer enn halvparten av innsatsen ved programvareutviklingen. Det er derfor svært viktig å analysere oppførsel relatert til MMK og ikke-funksjonelle krav. Basis for denne analysen er oppførselsmodellen. En særdeles viktig motivasjon for en slik analyse er å redusere antall kostbare endringer i konstruksjonen eller realiseringen av systemet.

4.2.1 Funksjonsanalyse

Modellering av systemoppførsel krever mange perspektiver, jfr innledende diskusjon i kapittel 3. De viktigste er illustrert i figur 4.3: funksjoner, data og sekvensering (kontroll). I praksis brukes forskjellige notasjoner/språk for forskjellige projeksjoner av systemoppførsel. F.eks. vil dataflytdiagrammer beskrive funksjoner og data, men ikke sekvenser. Notasjonene, eller bruken av disse, resulterer ofte i relativt uformelle beskrivelser som tildels også kan være inkonsistente. Uformelle beskrivelser er nødvendig i systemarbeid, men de begrenser mulighetene til analyse og tolkningsproblemer kan resultere i misforståelser. De forskjellige projeksjonene av systemoppførsel er ofte lite integrert.

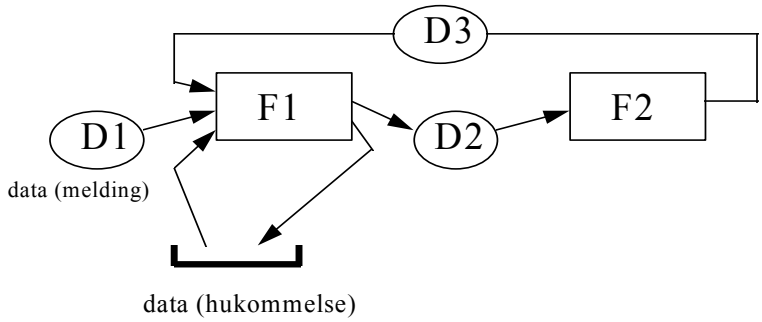


Figur 4-3 Viktige perspektiver ved oppførselsmodellering

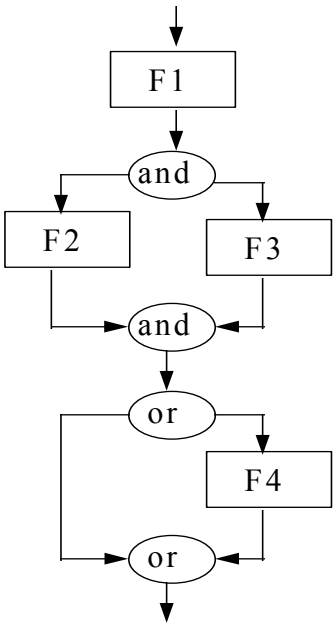
En funksjon betraktes vanligvis som en informasjonsbehandlende aktivitet. På et høyt abstraksjonsnivå fokuserer man på utdata og nødvendige inndata og funksjonen betraktes som en svart boks. Selve ryggraden i en tradisjonell funksjonsorientert oppførselsmodell er et hierarki av funksjoner som er dekomponert til et nivå systemutvikleren er fornøyd med. Antall toppnivå funksjoner er typisk ca 10. Dersom hver funksjon igjen dekomponeres i 10 funksjoner, vil man på tredje nivå i hierarkiet ha 1000 funksjoner! Bortsett fra svært store og kompliserte MMS vil man ikke operere med så mange funksjoner. Men en god og tilstrekkelig detaljert funksjonsanalyse krever en betydelig innsats.

Data-elementer/-hukommelse beskrives og assosieres med inn-/utdata som flyter mellom

funksjoner. Denne relasjonen mellom funksjoner kalles *dataflyt* og beskrives typisk i dataflytdiagrammer, se figur 4.4. Funksjoner betegnes ofte som aktiviteter, aksjoner, transformasjoner eller prosesser. Som tidligere nevnt, er det viktig å være klar over at dataflyt kun beskriver at informasjon *kan* flyte. I tillegg har man normalt behov for å beskrive *når* det vil skje og *hvem/hva* som utfører funksjonene. Den tidsmessige sekvensielle relasjon mellom funksjoner kalles kontrollflyt eller *funksjonsflyt*. Et eksempel på et funksjonsflytdiagram er vist i figur 4.5. Mange funksjoner i et sanntidssystem har som eneste (eller primære formål) å observere og/eller kontrollere sekvenseringen av andre funksjoner. Merk at man i funksjonsflytdiagrammet ikke har identifisert tilstandene eksplisitt. Systemet vil imidlertid generelt være i en tilstand før en funksjon utføres og en annen tilstand etterpå. Dersom man assosierer tilstander med kantene som forbinder funksjonene i flytdiagrammet kan man følgelig komme fram til en ekvivalent tilstandsbasert beskrivelse, jfr beskrivelsen av oppførselsgrafer i avsnitt 3.5.2 og Petri-nett i kapittel 3.5.3.



Figur 4-4 Dataflytdiagram



Figur 4-5 Funksjonsflytdiagram. OR-node angir valg og AND-node parallellitet

Det finnes språk der man kombinerer data- og funksjonsflyt i samme diagram, f eks oppførselsgrafer som illustrert i figur 4.6. Kompleksiteten av slike integrerte beskrivelser kan imidlertid fort resultere i at lesbarheten reduseres, og at man derfor i stedet (eller i tillegg) må bruke tradisjonelle projeksjoner som data- og funksjonsflyt. Funksjonsanalyse benytter i stor grad de samme modelleringsspråk som vi beskrev for prosedyrebaserte operatørmodeller (kapittel 3.5.2). Både tilstandsmaskiner og Petrinett (med nødvendige utvidelser), OMT og (O)SDL blir brukt i funksjonsanalyse.

Dersom vi ser på en spesifikk funksjon beskrives denne av:

- Relasjon til overordnet funksjon og eventuell dekomponering av funksjonen selv (f eks i data- og funksjonsflytdiagrammer)
- Inn-/utdata til funksjonen
- Start/stopp betingelser
- Ytelseskrav (tid, nøyaktighet, osv.) og eventuelle spor til kildedokumenter
- Beskrivelser av prosessering (f.eks. vha. tekst, strukturert tekst, pseudokode eller matematikk)

Det som kanskje er vanskeligst er dekomponering av krav. Forbindelsen mellom f eks krav til MMK og overordnete systemkrav er svært sjelden eksplisitt uttrykt. Dette representerer kanskje en av de viktigste begrensningene i en analytisk tilnærming til utviklingen av et MMS. Systemkrav verifiseres typisk ved at man *måler* relevante ytelsesvariable i omfattende prototyper eller mens systemet er i operativ drift.

Økende bruk av objektorientert (OO) analyse av systemer er kanskje den viktigste trenden innenfor funksjonsanalyse i dag (Coad & Yourdon, 1991 og Rumbaugh *et al*. 1991). En oversikt over metoder/teknikker finnes f eks i Monarchi & Puhr (1992). Når man snakker om formålet med OO bør man skille mellom:

- det å bidra til forståelse av den virkelige verden (modellaspektet)
- det å gi en praktisk basis for konstruksjon av programvare (realiseringsaspektet)

Det er modellaspektet som er mest relevant for vår diskusjon. Tradisjonell, funksjonsorientert analyse fokuserer på funksjoner som manipulerer passive objekter (data). I OO analyse er fokus på aktive objekter som interakterer og samarbeider. Fordi OO modellering innkapsler alle perspektivene i figur 4.3, vil dette kunne gi en mer integrert modell og derved bedre modellering av systemoppførsel. F eks er OMT som tidligere nevnt basert på tre modelltyper med hvert sitt perspektiv:

- Objektmodell (Beskrivelse av objekter, deres attributter og relasjoner, f eks aggregering og arv)
- Dynamisk modell (basert på tilstandskart)
- Funksjonsmodell representert som dataflytdiagrammer og beskrivelser av prosessering

Prioriteten er på objektmodellen, dernest den dynamiske modellen og til slutt funksjonsmodellen.

Rolle og ansvar (eng. role , resposibility) er begrep som overordnet OO analyse fokuserer på. Et system kan betraktes som en organisert struktur av samarbeidende objekter. Hvert objekt har ansvar (roller) som definerer hva som må gjøres. Et objekt tilbyr tjenester (tjenerrolle) til andre objekter og bruker selv tjenestene som andre objekter tilbyr (klientrolle). Det er i dag generell enighet angående modellering av distribuerte system som en samling av interakterende objekter med basis i klient-tjener-modellen (Nicol *et al*., 1993). Eksempler på slike OO metoder er:

- OORASS (OO Role, Analysis, Synthesis and Structuring)
- CRC (Class Responsibility Collaborator)
- RDD (Requirement Driven Design)

I avsnitt 2.2 diskuterte vi kort "filosofien" bak brukersentrert utvikling og her fokuserte vi også på begrepene rolle og ansvar. OO analyse synes å kunne få en betydelig innflytelse på oppførselsmodellering framover. Hvorvidt OO mer effektivt kan beskrive ekstern oppførsel («black box») er imidlertid diskutabelt og er hittil i liten grad dokumentert (Hsia *et al.*, 1993).

Eksempel: Funksjonsanalyse for kampledelsessystem (fortsatt) Vi fortsetter her eksempelet fra kapittel 4.1, men konsentrerer oss om funksjonene i anti-undervannsbåt (AU)-aktiviteten som foregår i operasjonsrommet. Figur 4.6 viser en oppførselsgraf som representerer en høynivå beskrivelse av denne delen av systemet.

Bildeoppbygging har som oppgave å oppdatere det taktiske undervannsbildet som inneholder informasjon om kontakter, track og andre objekter av interesse for operasjonen som for eksempel minefelt eller soner hvor man ikke kan avfyre våpen på grunn av vennlige undervannsbåter.

Kampledelsefunksjonen omfatter kontroll med bruk av sonar i tillegg til kontaktklassifisering, trusselvurdering, og kontroll med anvendelse av våpen og motmidler. Vi har valgt å splitte opp kommando og kontroll-funksjonen i en intern og en ekstern funksjon. Den eksterne delen vil bl.a. omfatte mottak og analyse av ordre fra overordnete samt ”tasking” og koordinering av underordnete enheter. Underordnete enheter kan være både fartøyer, undervannsbåter og fly. Intern kommando og kontroll innebærer koordinering mellom AU-teamet og andre enheter ombord på eget skip som for eksempel bro og vaktsjef. Dette er en forutsetning for å kunne fatte overordnete beslutninger på tvers av krigføringsområder.

Vi ser at det i tillegg kreves kommunikasjon innad i AU-teamet for å sikre en felles forståelse av situasjonen og målsetninger. Denne kommunikasjonen vil ofte fokuseres omkring nye objekter i det taktiske bildet.

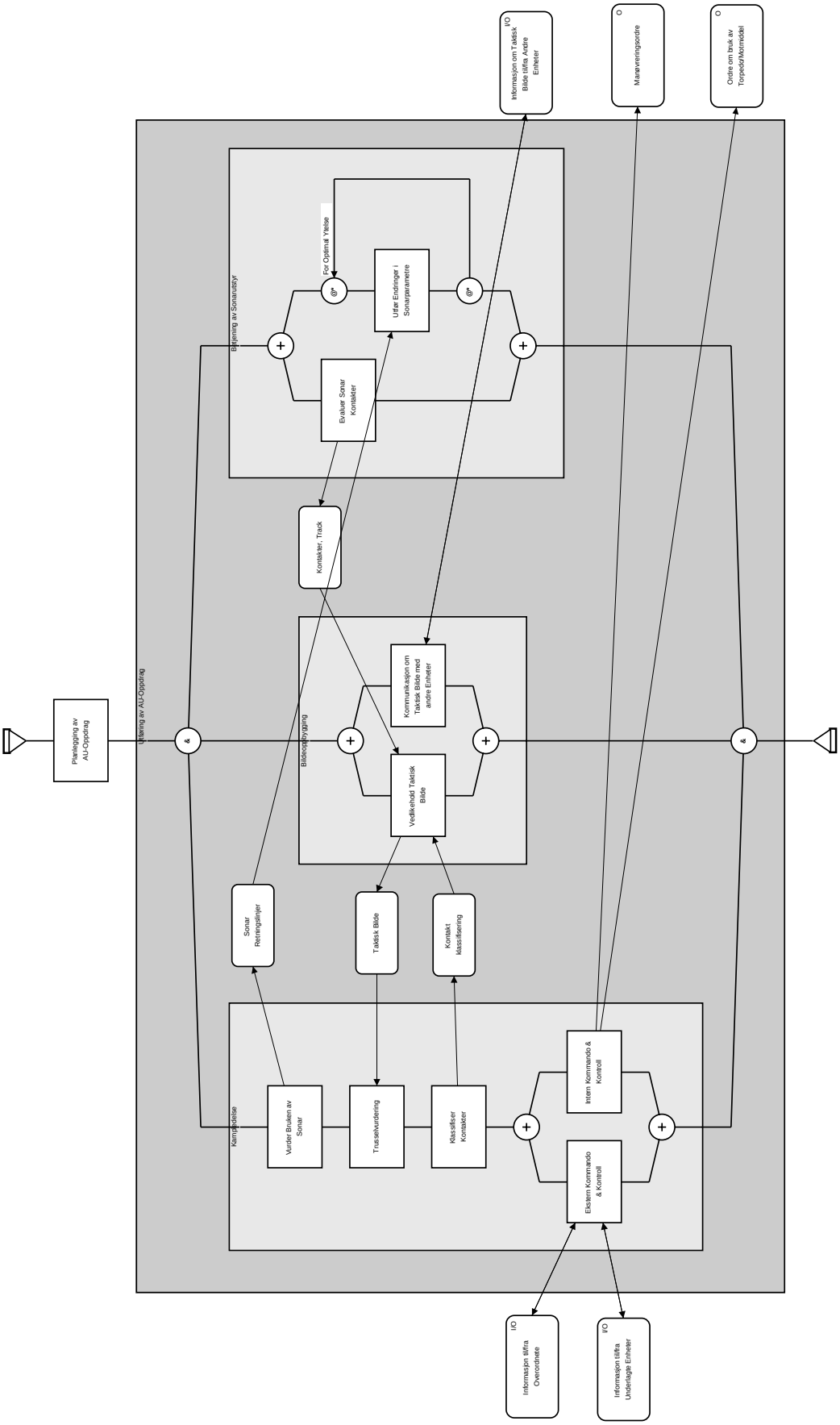
Dette eksempelet kan åpenbart føres videre med bl.a. en mer detaljert beskrivelse av

planleggingsfunksjonen, ekstern- og intern kommando og kontroll. I tillegg bør alle funksjonene og dataelement beskrives med tekst.

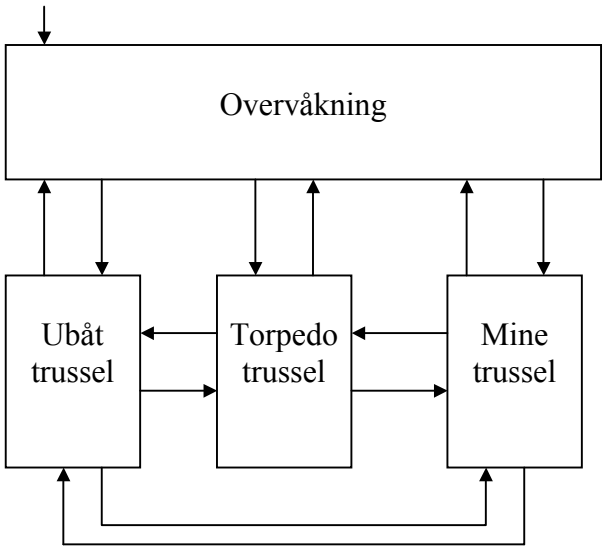
En alternativ måte å beskrive AU-funksjonaliteten på er ved hjelp av tilstandskart. Disse gir ofte et annet perspektiv på systemet og understreker systemets ulike operasjonsmodi. Innen AU-krigføring får vi et viktig modusskifte hver gang hovedtrusselen omdefineres. Et overordnet tilstandsdiagram for AU-systemet kan derfor være som vist i figur 4.7. I parallell med overvåkning og trusselhåndtering vil det være en kontinuerlig vurderingsprosess som sammenligner situasjonsutviklingen med eksisterende planer. I tilstandskart modelleres dette ved hjelp av AND-tilstander som vist i figur 4.8.

Med tilstandskart kan vi beskrive funksjonaliteten hierarkisk, vha. supertilstander akkurat som med funksjonsdiagrammer. Tilstanden for Undervannsbåttrussel kan f.eks. brytes ned i to OR-tilstander som aktiveres avhengig av om man vil unngå eller angripe ubåten. Dette er vist i figur 4.9 hvor vi har brudt disse tilstandene ned i ytterligere OR-tilstander.

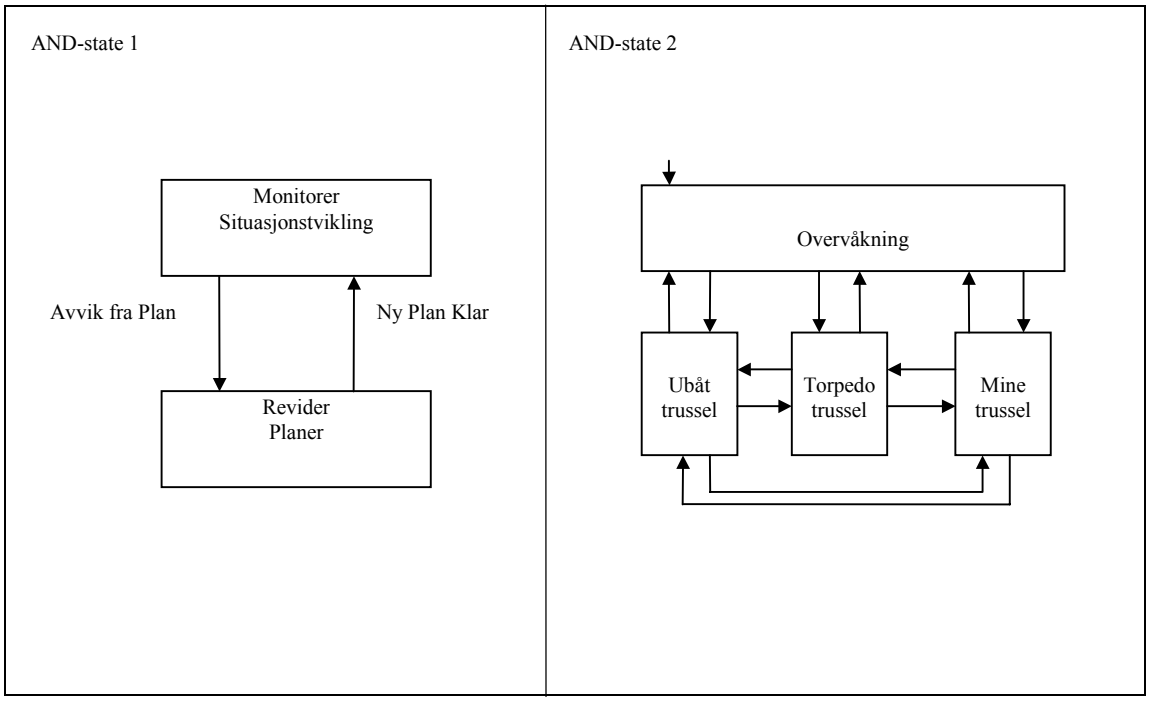
Et resultat av en mer fullstendig analyse ville være en liste av funksjoner som systemet utfører. En slik liste av 23 funksjoner er vist i tabell 4.1.



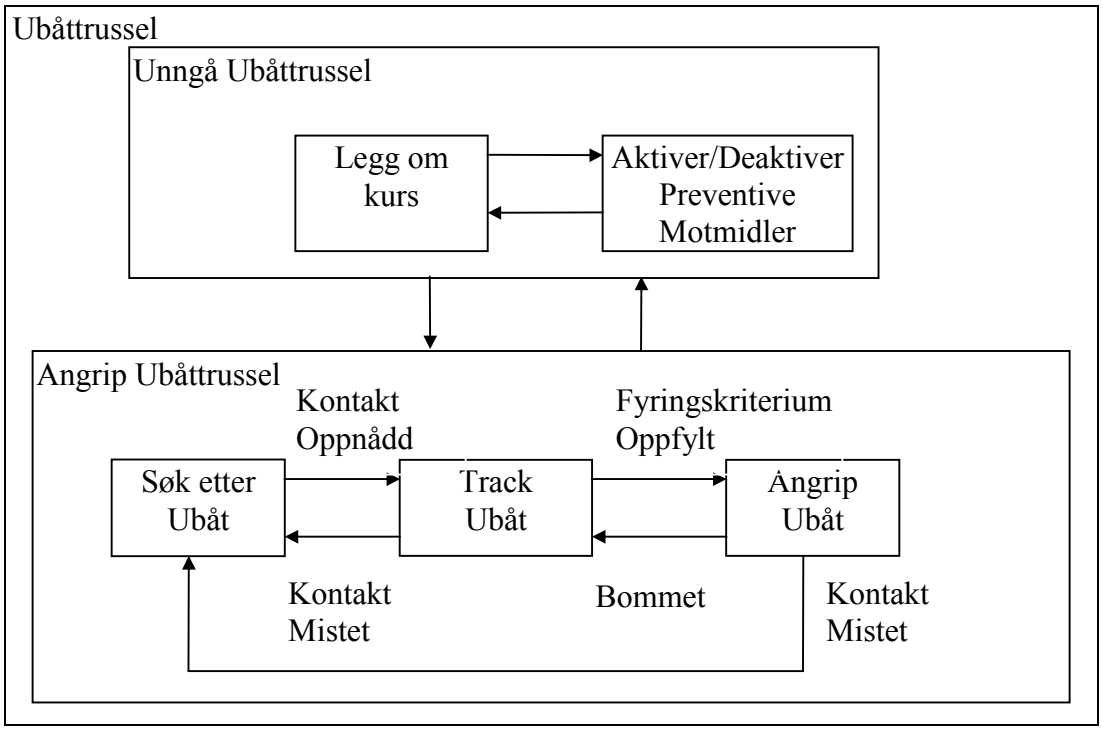
Figur 4-6 Oppførselsgraf for AU-aktiviteter i operasjonsrommet på en fregatt



Figur 4-7 Tilstandskart for alternative AU-aktiviteter



Figur 4-8 Tilstandskart for parallelle AU-aktiviteter



Figur 4-9 Ekspandert tilstandskart for Ubåttrussel-tilstand

Hovedfunksjoner	**Funksjoner**
1. Langtidsplanlegging	1.1 Situasjons- og oppdragsanalyse 1.2 Koordineringsplanlegging 1.3 Definering av faste reaksjonsmønstre 1.4 Tilordning av oppdrag
2. Korttids Kampledelse	2.1 Manøvreringskontroll 2.2 Valg av retningslinjer for sonarbruk 2.3 Kontaktklassifisering 2.4 Trusselevaluering 2.5 Kontroll av torpedomotmidler 2.6 Engasjementskontroll 2.7 Våpenkontroll 2.8 Skadevurdering
3. Ekstern kommando og kontroll	3.1 Kommunikasjon med overordnete 3.2 Taske underordnete enheter 3.3 Koordinering av underordnete enheter 3.4 Kontroll av fly og helikopter
4. Intern kommando og kontroll	4.1 Koordinering med fartøysjef 4.2 Koordinering innad i AU-teamet 4.3 Koordinering med bro
5. Undervanns bildeoppbygging	5.1 Vedlikehold av undervannsbilde 5.2 Kommunikasjon av undervannsbilde
6. Sensorbruk	6.1 Vurdering av sonarkontakter 6.2 Endring av sonarparametre

Tabell 4.2 Funksjonsnedbrytning for AU-teamet på en fregatt

4.2.2 Om automatisering

Automatisering er ikke bare et teknisk spørsmål om hva som er mulig å få til, men vel så mye et spørsmål om hvor langt det er ønskelig å gå sett fra et helhetlig systemsynspunkt. I stor grad er holdningen den at selv om vi ikke greier å utvikle et helautomatisk system, vil det være en fordel for de som skal betjene systemet at så mye som mulig er automatisert for å gjøre jobben deres lettere. Så enkelt er det imidlertid ikke. Økt automatisering fører nødvendigvis ikke til at det verken blir lettere å betjene et system eller at vi kan stille mindre krav til operatørene. Det vi i beste fall kan håpe på er at antall operatører blir mindre. Dette er på sett og vis et paradoks ved øket automatisering (Brainbridge, 1983) og er grunnen til at det ikke nødvendigvis er ønskelig å automatisere selv om det er teknisk mulig. Punktene nedenfor utdyper dette:

- Økt automatisering krever operatører med høyere kunnskapsnivå. I tillegg til at de må ha forståelse for prosessen, må de også ha kunnskap om hvordan automatiseringsutstyret virker. Operatørene må forstå automatiseringsutstyret for å ha tillit til det og bruke det. Dette gjelder også i høy grad for såkalt «intelligent» automatisering som gjør det enda vanskeligere for operatøren å bygge opp en intern modell av systemet.
- Økt automatisering fører til at det blir vanskeligere for operatører å opprettholde sin kompetanse (om prosessen og automatiseringsutstyret), da de får færre oppgaver å utføre og blir mer dekoblet fra prosessen.
- De oppgavene som systemutvikleren ikke greide å automatisere (etter lang tids tenking på et behagelig kontor) overlates til operatøren å løse i sann tid under vanskelige forhold etter å ha fratatt han muligheten til å opprettholde kompetanse og uten å tilrettelegge for at han skal kunne overta når uventede og sjeldne situasjoner oppstår.
- Økt automatisering fører til at operatøren bruker mer av sin tid på å overvåke automatikken, selv om vi vet at mennesket er en svært dårlig overvåker. Er det noe vi burde automatisere så er det overvåkningsfunksjonene. P M Fitts, som på en måte er funksjonsallokeringens far ved etablering av sin MABA-MABA liste (Men/Machines Are Better At) i 1951 (se kapittel 4.2.4), påpekte dette og behovet for maskinell støtte når mennesket skal ha en overvåkerrolle.
- Økt automatisering fører til økt kompleksitet av det tekniske utstyret. Komplekst utstyr inneholder feil. Feil fører til at operatørene mister tillit og slutter å bruke det og går tilbake til den gamle måten å løse oppgaven på. Gevinsten med innføring av automatiseringssystemet går tapt.

Så lenge vi ikke snakker om helautomatiske systemer er altså ikke målsetningen å automatisere mest mulig, men å automatisere slik at ytelsen av MMS-et blir «optimal» og slik at operatøren har kontroll og kan utøve sitt ansvar i totalsystemet.

For at det skal være mulig å automatisere må det være mulig å beskrive/formalisere/ modellere en prosess' oppførsel og styringen av den. Et motargument til punktene ovenfor er derfor at en maskin i lengden vil utføre en oppgave som lar seg formalisere bedre enn en operatør. Det kan også stilles spørsmål om det gir en verdig jobb å la mennesker utføre oppgaver og funksjoner som lar seg formalisere.

Som det fremgår av det ovenstående ser vi at hvilken filosofi som vi har rundt automatisering er

med på å bestemme operatørens *rolle* i MMS-et. Nå ser vi en klar tendens til at det går i retning av en overvåkerrolle. Spørsmålet er om det er en ønsket utvikling og hvor langt skal vi la den gå. Ved innføring av stadig mer informasjonsteknologi blir dette rollespørsmålet bare mer og mer et viktig prinsipielt spørsmål.

Automatisering er altså i stor grad en prøve- og feileprosess som det ikke er lett å sette krav til eller modellere (Hsia *et al*., 1993). Konsekvensene av automatisering mhp systemytelse og arbeidsbelastning blir ofte i liten grad analysert tidlig i utviklingen. Innføring av beslutningsstøtte kan som nevnt over føre til økt arbeidsbelastning. Selv om man forutsetter at beslutningsstøtte øker systemytelsen, vil den faktiske virkningen avhenge av tiltroen operatøren har til verktøyet og i hvilke situasjoner han bruker det. Feilaktig bruk vil kunne redusere ytelsen! Generelt sett er det viktig å huske på at automatisering ikke er noe mål i seg selv og at man bør fastholde utgangspunktet; nemlig brukerbehovene.

4.2.3 Operatørstøttesystemer

Mens automatisering har til hensikt å erstatte operatøren, har operatørstøtte til hensikt å hjelpe operatøren å gjøre sin jobb på en bedre (mer, raskere, fortere, nøyaktigere ...) måte. Operatørstøtten kan ha to ulike formål. Enten å sette han i stand til å utføre oppgaver som før ikke var mulig eller å utføre oppgaver mer effektivt. Mens automatisering gjerne fører til at oppgavene i en spesifikk jobb begrenses, kan operatørstøttesystemer, når det blir benyttet på en riktig måte, føre til at operatørens jobb kan bli utvidet til nye områder.

Med *operatørstøtte* og *operatørstøttesystemer* menes det alle former for tiltak som benyttes for å hjelpe operatøren i gjennomføringen av de oppgavene som han er blitt tildelt (jobb). Med andre ord er det et svært vidt begrep som dekker alt fra f eks å fremheve data vha fargekoding til avanserte hjelpemidler som f eks innebygd simulering. Støtten kan altså være alt fra støtte i forbindelse med basale persepsjonsoppgaver til komplekse kognitive oppgaver. *Beslutningsstøttesystemer* betegner en undergruppe av operatørstøttesystemene som gir støtte til beslutninger som operatøren må foreta. Vi kan snakke om støttesystemer for de forskjellige typer av oppgaver som operatører blir tildelt, jamfør de ulike stegene i en beslutningsprosess som vist i figur 3.11.

Operatørstøttesystemer er mange ganger slik utformet at de kommer opp med et forslag som operatøren skal godkjenne eller forkaste. Det kalles støtte, men er egentlig forkledd automatisering. Hvis «operatørstøttesystemet» er bra kommer det opp med gode forslag, som operatøren i de aller fleste tilfeller bare bør godkjenne. I hvilken grad operatøren har reell mulighet for å overprøve forslagene fra operatørstøttesystemet kan man også sette spørsmål til. Et godt operatørstøttesystem utformet etter dette prinsippet bør kutte ut operatørens medvirkning (altså automatisering). Hvis det gir så mange dårlige forslag at operatøren er nødt til å bringes inn i sløyfa for å forkaste dem, bør det vurderes om det er verdt å utvikles i det hele tatt.

Operatørstøttesystemer i sin egentlige forstand og som tar operatøren på alvor bør utformes slik at han øker sin kompetanse ved å benytte det og at det er han som bestemmer når og hvordan

støttefunksjonene skal benyttes. Dvs at operatørstøttesystemer bør utformes slik at de er mer en assistent til eller et verktøy for operatøren. Det bør legges vekt på å utnytte datamaskinens beregnings- og visualiseringskapasitet, ikke dens begrensede muligheter for å etterape menneskelige kognitive funksjoner vha såkalte kunnskapsbaserte teknikker som produksjons-regler og symbolsk databehandling. Støttesystemene bør altså hjelpe operatøren til å fremskaffe bedre beslutningsgrunnlag, men i større grad overlate vurderingen basert på datagrunnlaget og selve beslutningen til operatøren selv.

4.2.4 Allokering

Etter at den innledende funksjonsanalysen er gjennomført kan man ta fatt på første iterasjon av funksjonsallokeringen (FA) . FA kan defineres som *prosessen hvor man bestemmer hvilke systemfunksjoner som skal utføres av mennesker, av utstyr, eller av begge deler - og når mennesker skal utføre funksjoner, hvordan de fordeles på ulike operatører*. FA anses som et viktig steg under utvikling av MMS. FA har med andre ord med hvor mye automatisering som skal benyttes og hvilken rolle som operatøren(e) skal ha i systemet, hvilke oppgaver de får seg tildelt og hvordan operatørstøttesystemer utformes.

Stadig økende grad av automatisering og innføring av informasjonsteknologi har endret operatørenes oppgaver fra manuell kontinuerlig regulering («manual control») til overvåkning, beslutningstakning og problemløsning («supervisory control»). Operatøroppgavene har dermed i større grad fått et kognitivt innhold og menneskets egenskaper som informasjonsbehandler blir derfor viktig. Vi har derfor behov for kunnskap om og modeller av menneskelig informasjons-behandling for å foreta en best mulig funksjonsallokering og å finne en riktig automatiseringsgrad.

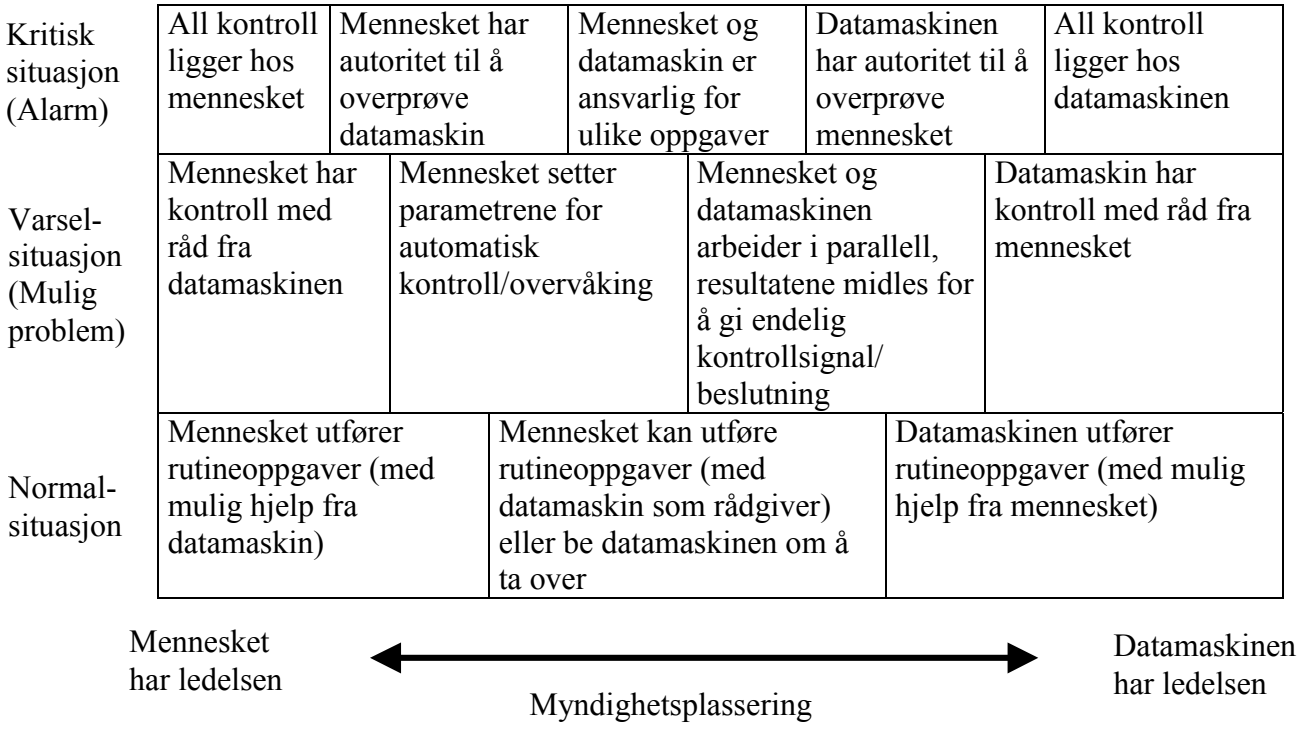
Den overordnete *målsetning* er å oppnå en funksjonsfordeling som oppfyller MMS krav og som gir en akseptabel arbeidsbelastning og jobb for operatøren(e). Funksjonsallokering er delvis et «høna og egget»-problem, dvs vi må egentlig ha en allokering og en forholdsvis detaljert konstruksjon av en løsning for å vurdere begrensninger og muligheter *før* vi kan foreslå allokeringen!

Funksjoner blir ofte allokert til både operatør og maskin, dvs man har ikke lenger to disjunkte mengder. I stadig større grad snakker man om *kooperative* systemer. Under funksjonsanalysen må man imidlertid før eller siden beslutte *hvem* som har myndighet i hvilke situasjoner; operatøren eller maskinen. Mulige alternative allokeringer er illustrert i figur 4.10 og figur 4.11. Vi legger merke til at operatøren kan delegere myndighet til maskinen i ulike situasjoner og at automatisering kan avhjelpe operatørbegrensninger. I brukersentrert utvikling mener vi at operatøren har ansvaret og derfor må ha kommando, dvs ha myndighet. Selv om en funksjon er allokert til maskinen (automatisert) vil operatøren vanligvis ha overordnet kontroll og han forventes uansett å gripe inn dersom det blir nødvendig. Det vil alltid kunne oppstå situasjoner som utviklerne av MMS ikke har forutsett og som dermed ikke kan løses med automatisering. Dette er illustrert i figur 4.11.

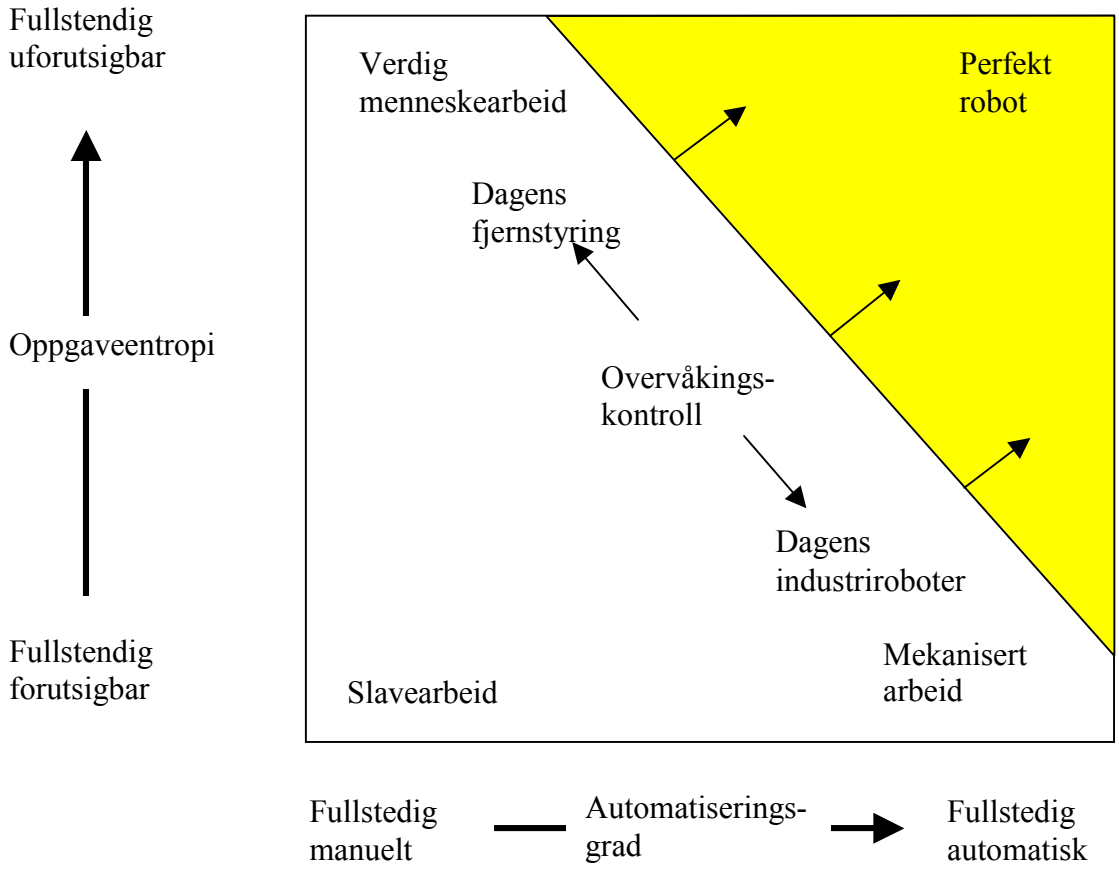
Etablering av generelle *automatiseringsnivåer* for et prosjekt kan være nyttig ved etablering av forskjellige alternative allokeringer. I et system vil man vanligvis ha ulike kombinasjoner av nivåer. Ved gjennomføring av en aksjon kan man f eks skille mellom følgende nivåer (Sheridan, 1994):

1. Operatøren evaluerer alternativer og velger aksjon. Maskinen utfører bare selve aksjonen.
2. Maskinen begrenser mulige alternativer som operatøren velger fra.
3. Maskinen foreslår et alternativ.
4. Maskinen velger et alternativ og utfører aksjonen hvis operatøren godkjenner det.
5. Maskinen velger et alternativ, utfører aksjonen og informerer operatøren.
6. Maskinen velger et alternativ, utfører aksjonen og informerer operatøren *hvis* han spør.
7. Maskinen gjør alt og velger om operatøren skal informeres.

I situasjoner med sterkt tidspress og/eller store informasjonsmengder (som må prosesseres på en kognitivt krevende måte), kan automatisering være den eneste mulige løsningen. Selv om operatører er uunværlige i de fleste systemer er de også ofte hovedårsaken til at alvorlige feil og ulykker inntreffer. Man har idag ganske mye kunnskap om menneskelige feilhandlinger. F eks er feilaktig menneskelig oppførsel viktigste årsak i ca 75 % av ulykkene i sivil lufttrafikk (Onken ,1994). Tall som det typisk referes til i litteraturen varierer mellom 20-70 %, typisk verdi er 50 %.



Figur 4-10 Alternativ allokering av myndighet mellom operatør og datamaskin som funksjon av hvor kritisk situasjonen er (Sheridan, 1994)



Figur 4-11 Systemer med ulike automatiseringsnivåer som utfører oppgaver med ulike grader av kompleksitet (entropi=mangel på forutsigbarhet), (Sheridan, 1992)

I systemsammenheng bør man egentlig snakke om *risiko* som generelt sett kan betraktes som en funksjon av *feilsannsynlighet* og *konsekvensen* av feilen. Dersom risikoen vurderes å være for høy er en rekke tiltak mulig:

- Økt innsats på opplæring
- Bedre prosedyrer
- Forbedringer i konstruksjonen av det tekniske systemet (redundans, muligheter for korrektive tiltak, støttesystemer for feildeteksjon og -diagnose, og automatisering).

Generelt sett kan man nok si at fokus har vært på teknologi og automatisering (dvs det siste punkt). Organisasjonen som opererer de tekniske systemene har ofte ikke utviklet seg i takt med teknologien. Det samme gjelder betydningen av og mulighetene for bedre operatøropplæring. Omfanget av menneskelige feil som beskrevet over må forståes på bakgrunn av dette.

En oversikt over allokeringsstrategier eller teknikker finnes i (Meister, 1985, Booher, 1990 og Rouse, 1991). En forskningsgruppe (Beevis *et al*., 1992) konkluderte med at teknikkene var mangelfulle, lite utviklet og generelt dårligere enn for andre systemutviklingsaktiviteter, f eks funksjonsanalyse og oppgaveanalyse.

Teknikker som ofte omtales i litteraturen er:

- Ad-hoc allokering (tar utgangspunkt i eksisterende systemer og informasjon om hva som er mulig å automatisere).
- Sammenligningsallokering. Sammenlignende vurdering av egenskaper og ytelse til operatører og maskin.
- Formell automatiseringsanalyse (eller avveiningsanalyse). F eks kan kriteriet være minimum kostnader (eng. economical allocation).
- «Keep-operator-in-the-loop»-allokering som fokuserer på hvordan operatøren best mulig skal kunne fungere som overvåker i kompliserte systemer. Se f eks (Booher, 1990).
- Iterativ allokeringsmetode.

Merk at man ikke vil gjøre en like grundig analyse av alle funksjonene i et system, men konsentrere innsatsen der hvor allokeringen er mest usikker og hvor den har størst betydning.

Ad-hoc allokering automatiserer så mye som mulig og lar operatøren gjøre resten (eng. leftover allocation). En slik framgangsmåte kan ha en del uheldige konsekenser dersom kompleksiteten og ansvar blir for mye redusert (kjedsomhet, understimulering) eller dersom oppgavene ikke danner en rimelig koherent operatørjobb.

Sammenligningsallokering (eng. comparison allocation). Denne teknikken sammenligner menneskets og maskinens generelle egenskaper som antas å være relevante når oppgaver skal løses, f eks oppfatning av informasjon og reaksjon på uventede hendelser. Se tabell 4.2. Teknikken ble først beskrevet av Fitts (1951). En mer moderne variant av denne teknikken er beskrevet i (Cambell & Essens, 1994). Slike teknikker er relativt enkle og lett anvendbare ved en første iterasjon av FA.

Formell automatiseringsanalyse. FA er egentlig et standard optimaliseringsproblem forutsatt at man har veldefinerte, matematiske ligninger av oppførselen til elementene som inngår i menneske-maskin-systemet og en objektfunksjon, også på matematisk form, som inkluderer alle fremtredende variabler. Problemet er selvsagt at vi pr i dag ikke har tilgjengelige fullstendige modeller av verken menneskene eller av de mer og mer komplekse utstyrselementene som inngår i et MMS. Likeledes er det heller ikke trivielt å definere en objektfunksjon som foreskriver hva en optimal fordeling av funksjoner vil være. Med andre ord, hva som er en optimal automatiseringsgrad er fremdeles vanskelig å definere.

En form for formell automatiseringsanalyse kan likevel være en nyttig teknikk å bruke ved FA og en generell framgangsmåte er:

1. Etabler evalueringskriterier, måleskala og eventuelle vekter til disse. Måleskalaen kan være rangordnet (god, middels, dårlig), intervallskala (tall uten definert nullpunkt) eller forholdsskala (tall med definert nullpunkt).
2. Etabler generelle automatiseringsnivåer.

3. Etabler ulike alternative allokeringer for hver funksjon (eller funksjonsgruppe). Bruk nivåene fra pkt 2 som en ledesnor ved utvikling av alternativene.
4. Analyser hvert av alternativene opp mot de ulike evalueringskriteriene ved at de gis en verdi på den aktuelle måleskalaen.
5. Evaluer evt. følsomheten til de ulike alternativene med hensyn på kriteriene.
6. Velg det «beste» alternativet. Dersom man har tallskala og kriteriene er vektlagt, kan dette regnes ut på ulike måter. Et regneark kan være et godt verktøy.

Egenskap	***Mennesket***	***Maskin***
Hastighet (muskler)	Tidskonstant ca 1 s	Svært mye raskere
Effekt	2 HK i ca 10 s 0.5 HK i noen minutter 0.2 HK over lengre tid	Svært mye større, konstant
Sanseegenskaper	Stort amplitydeområde (1- 10^{12}) Klarer å behandle ulike variable fra ett stimuli (f eks kan øyet sanse posisjon, farge, bevegelse og luminans) Meget god til å gjenkjenne mønster. Kan skille ut relevant informasjon fra støybefengt bakgrunn	Noen typer stimuli som ikke mennesket kjenner, f eks i det elektromagnetiske spektrum (infrarød, radiobølger), radioaktivitet o.l. Dårligere til å gjenkjenne mønster
Hukommelse	God langtidshukommelse, egnet for strategier og prinsipper	Stor korttidshukommelse, egnet for reproduksjon
Hastighet på informasjonsoverføring (sende)	Lav (5-10 bit/s)	Høy
Fleksibilitet	Fleksibel, kan utføre mange typer oppgaver	Generelt sett bare tilordnede (predefinerte) oppgaver
Repeterbarhet	Lite egnet	Ideell for rutiner og gjentakelser
Komplekse aktiviteter	I hovedsak en kanal	Flere kanaler (kan gjøre mange ting samtidig)
Vurderingsevne	God evne til å ta beslutninger fra et lite antall data (god hukommelse)	God evne til å ta enkle beslutninger fra stort antall observasjoner
Beregninger	Langsom, usikker, god evne til å korrigere grove feil (sunn fornuft)	Hurtig, sikker. Dårlig evne til å korrigere egne feil
Intelligens	Kan handle fornuftig i uforutsette situasjoner	Relativt liten (kun spesialområder)
Overvåkning	Mindre god	God

Tabell 4.3 Egenskaper til operatør og maskin

Typiske evalueringskriterier til bruk under allokering kan være:

- Ulike ytelsesmål, f eks nøyaktighet og mengde/antall enheter produsert/behandlet pr tidsenhet
- Feiltoleranse
- Sikkerhet
- Operatørbelastning
- Organisasjon/operatørakseptanse

- Opprettholdelse av kunnskap/ferdighet hos operatørene
- Operatørtrening
- Utviklingskostnader
- Driftskostnader (maskinvare og bemanning, antall operatører og opplæringskostnader)
- Vedlikeholdskostnader
- Teknologisk mulighet (begrensninger)
- Maskinbelastning
- Utvidelsesmuligheter

Rouse (1991) beskriver en *iterativ allokeringsmetode* som består av tre gjenomløp. *Den initielle konstruksjonsfasen* er basert på bruk av tradisjonelle teknikker som f eks sammenligning av egenskaper. Funksjoner som allokeres til operatøren blir deretter omdannet til operatøroppgaver ved å konstruere informasjonsgivere, betjeningsorganer og operasjonsprosedyrer. Operatørens ytelse/arbeidsbelastning predikteres for de enkelte oppgavene ved ulike tidspunkter. I *integrasjonsfasen* fokuserer man på relasjonene mellom de oppgavene som utføres samtidig. Komplementære oppgaver kan lede til mer integrert presentasjon av informasjonen, mer integrert bruk av betjeningsorganer eller mer integrerte prosedyrer. Derved kan man eventuelt bedre ytelsen og/eller redusere arbeidsbelastningen. I *den endelige konstruksjonsfasen* går man gjennom tidligere beslutninger og vurderer i tillegg bruk av dynamisk allokering. Evalueringen krever i denne fasen vanligvis prototyping.

Prinsipper ved allokering av oppførsel til maskinvare og programvare (på ulike nivåer) innenfor system- og programvareteknikk kan også nyttes (Nordø & Bråthen, 1994). Selve FA er typisk implisitt i selve konstruksjonsprosessen med oppførselsmodellering fra funksjonsanalysen som viktig basis.

Det kan være fordelaktig ikke å ha en helt fastlåst fordeling av oppgavene i et system og i stedet allokere på nytt dersom situasjonen tilsier dette. Dette kan gjøres for å få en mer balansert og konstant arbeidsbelastning for operatøren(e). Dette kalles *dynamisk allokering.* Operatøren kan ha flere forskjellige strategier for å løse en oppgave som har varierende grad av automatisering. Slik tilpasning kan gjøres på to måter:

- avhengig av operatørens tilstand, se f eks Rouse (1991)
- avhengig av systemtilstand (situasjon)

Den første typen tilpasning diskuteres innenfor MMS-forskning, men er hittil ikke benyttet i virkelige systemer. Den andre typen tilpasning bygges i en viss grad inn i en del typer moderne systemer. Slike systemer kan også inneholde en enkel brukermodell (eng. intent model) som gir en normativ beskrivelse av operatørens oppgaver og sekvenseringen av disse avhengig av viktige hendelser. Systemet kan eventuelt selv foreta tilpasningen (dvs automatisk). Manuell tilpasning basert på operatørens *egen* vurdering av arbeidsbelastning er imidlertid ofte en vel så god løsning. Men merk at dette gir en ekstra oppgave å utføre for operatøren.

Simulering er en viktig teknikk ved FA for å prediktere ytelsen for ulike automatiserings-alternativer/strategier. Dette krever egnede ytelsesmodeller av operatøren. I praksis er ofte *tid* den mest sentrale ytelsesparameteren, dvs at man kan foreta såkalte tidssimuleringer.

Man kan også prototype ulike automatiseringsalternativer. «Man-in-the-loop» simuleringer som er koblet til dynamiske scenarier er ønskelig. En viss vekt på realisme i skjermbilder, betjenings-teknikker og betjeningsorganer er nødvendig for å kunne vurdere ulike alternativer på en akseptabel måte.

For å validere funksjonsallokeringen er det stor enighet om at prototyping (eller «man-in-the-loop» simuleringer) er nødvendig. Bruk av analytiske simuleringsmodeller av MMS er som nevnt over et viktig redskap under FA-prosessen, men disse bør valideres vha prototyper (Nordø & Bråthen, 1994).

Et spørsmål er i hvor stor grad det lar seg gjøre å etablere generelle metoder og teknikker for FA som er nyttige for et bredt spekter av typer systemer. Vi tror at dette kan vise seg å være vanskelig. Vi tror at det bør legges større vekt på metoder og teknikker for det som ligger før FA, nemlig modellutvikling og det som ligger etter FA, nemlig det å verifisere og validere at den valgte allokeringen oppfyller gitte krav.

Eksempel: Funksjonsallokering i et kampledelsessystem (forts) I eksempelet i kapittel 4.2.1 startet vi på en funksjonsanalyse av systemet for undervannskrigføring på en fregatt. Det er flere av disse funksjonene som kan støttes og delvis automatiseres ved hjelp av datamaskiner, men det er antakeligvis bare en funksjon (2.5) som det vil være aktuelt å automatisere fullstendig.

Det fins flere muligheter for allokering av disse funksjonene til de ulike medlemmene i AU-teamet. Hvis vi går ut ifra at vi har fire operatører i AU-teamet, kan to mulige allokerings-alternativ settes opp som i tabell 4.4 nedenfor. I disse forslagene til arbeidsfordeling antar vi at AU-offiseren har ansvar for langsiktig planlegging, tar overordnete beslutninger og følger opp beslutningene med overvåking av de andre operatørenes arbeid.

Målsetningen med å prøve ulike alternativer er å finne en fordeling som:

1. Balanserer arbeidsbelastningen mellom operatørene på en rimelig måte.
2. Minimaliserer behovet for kommunikasjon mellom operatørene.
3. Minimaliserer ventetiden før en oppgave blir utført.
4. Maksimaliserer midlere antall oppgaver som kan utføres innen et gitt tidsrom.

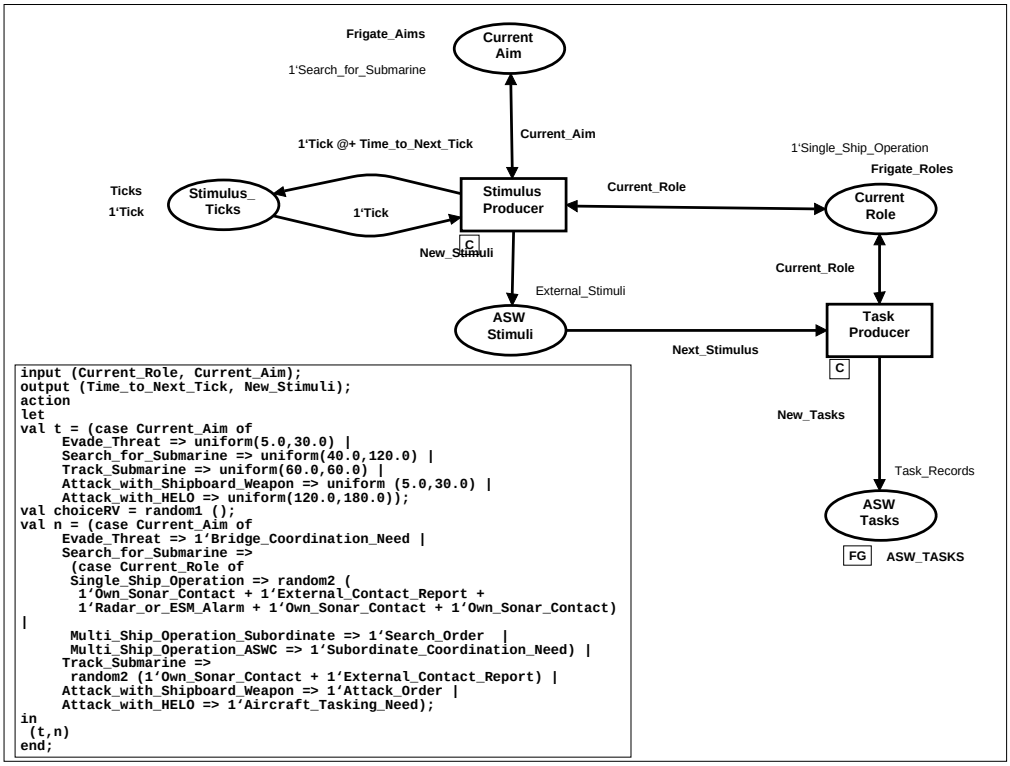
Man bør også vurdere om forutsetningen om fire operatører fører til tilfredsstillende løsninger. Vil teamet være overarbeidet, eller kunne man tvert imot klare seg med færre operatører?

I figur 4.12 til figur 4.14 viser vi Petri-nettmodeller som kan brukes for å simulere alternative

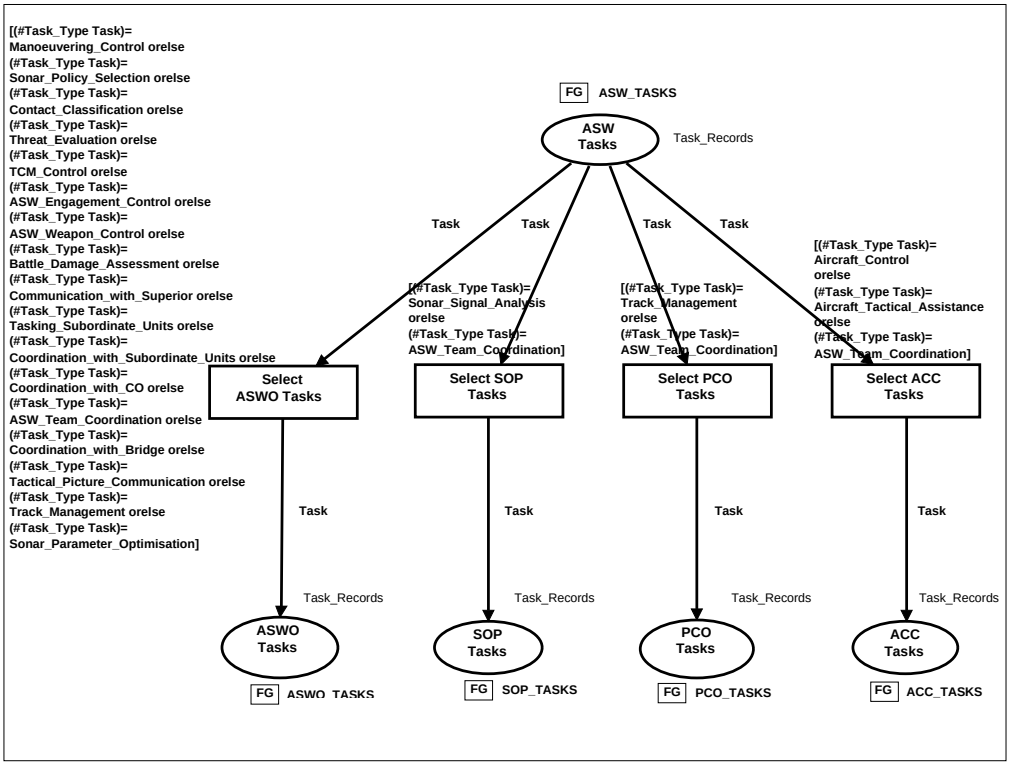
oppgavefordelinger i AU-teamet. Et eksempel på data fra en slik simulering er vist i figur 4.15. Her er arbeidsbelastningen på de ulike operatørene tegnet som en funksjon av tid.

Funksjoner	**Bemannings-alternativ 1**	**Bemannings-alternativ 2**
1.1 Situasjons- og oppdragsanalyse	AU Offiser	AU Offiser
1.2 Koordineringsplanlegging	AU Offiser	AU Offiser
1.3 Definering av faste reaksjonsmønstre	AU Offiser	AU Offiser
1.4 Tilordning av oppdrag	AU Offiser	AU Offiser
2.1 Manøvreringskontroll	BO Offiser	AU Offiser
2.1 Valg av retningslinjer for sonarbruk	BO Offiser	AU Offiser
2.2 Kontaktklassifisering	BO Offiser	AU Offiser
2.3 Trusselevaluering	BO Offiser	AU Offiser
2.4 Kontroll av torpedomotmidler	BO Offiser	AU Offiser
2.5 Engasjementskontroll	AU Offiser	AU Offiser
2.6 Våpenkontroll	BO Offiser	AU Offiser
2.7 Skadevurdering	AU Offiser	AU Offiser
3.1 Kommunikasjon med overordnete	AU Offiser	Fartøysjef
3.2 Taske underordnete enheter	AU Offiser	Fartøysjef
3.3 Koordinering av underordnete enheter	AU Offiser	Fartøysjef
3.4 Kontroll av fly og helikopter	FK Offiser	AU Offiser
4.1 Koordinering med fartøysjef	AU Offiser	AU Offiser
4.2 Koordinering innad i AU-teamet	AU Offiser	AU Offiser
4.3 Koordinering med bro	AU Offiser	AU Offiser
5.1 Vedlikehold av undervannsbilde	BO Offiser	BO Offiser
5.2 Kommunikasjon av undervannsbilde	BO Offiser	BO Offiser
6.1 Vurdering av sonarkontakter	Sonaroperatør I/II	Sonaroperatør I/II
6.2 Endring av sonarparametre	Sonaroperatør I/II	Sonaroperatør I/II

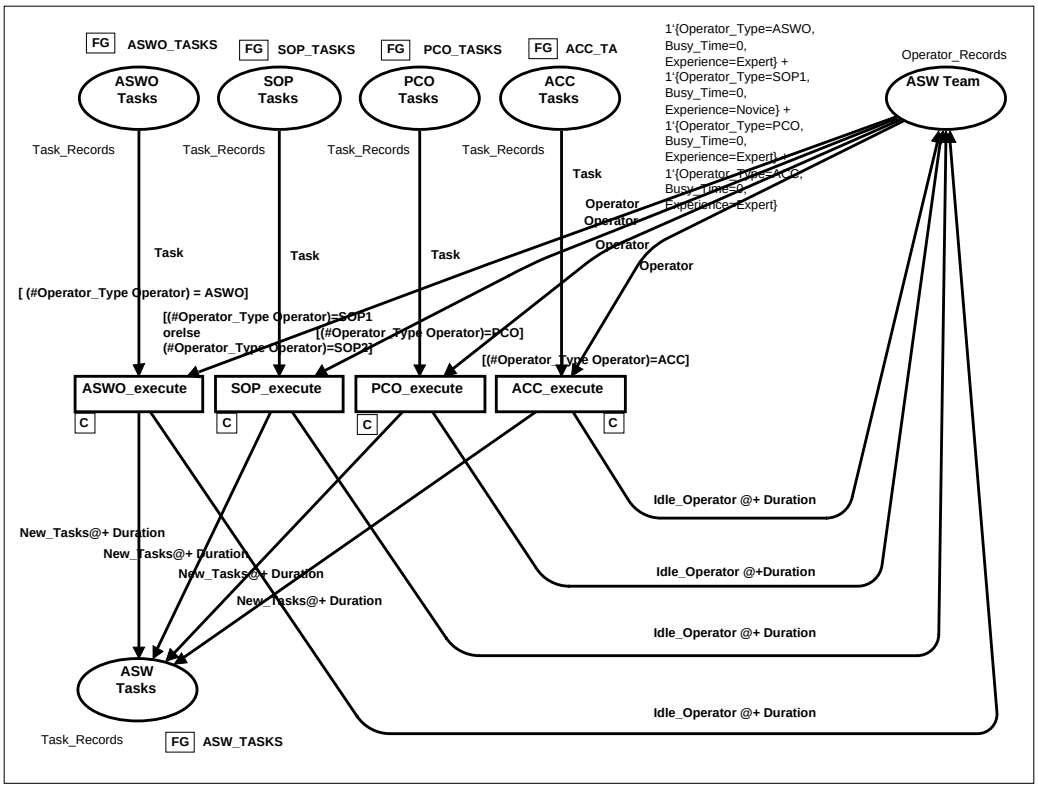
Tabell 4.4 Eksempel på mulige funksjonsfordelinger i AU-teamet på en fregatt. (AU=Anti-undervannsbåt, BO=Bildeoppbygger, FK=Flykontroll)



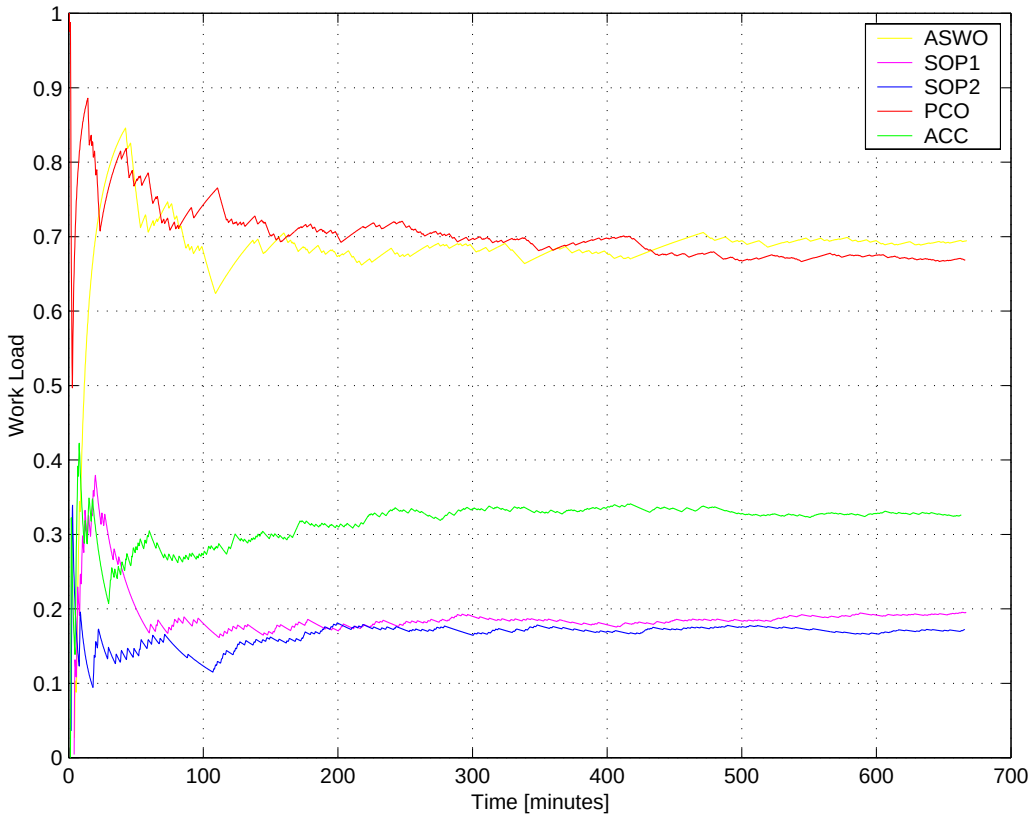
Figur 4-12 Oppgavegenerering for AU-teamet på en fregatt



Figur 4-13 Oppgavefordeling i AU-teamet på en fregatt



Figur 4-14 Tidssimulering av oppgaveutføring. Når en oppgave er utført genereres nye oppgaver som illustrert i et av kodesegmentene



Figur 4-15 Arbeidsbelastning for AU-operatører som funksjon av tid (ASWO=AU offiser, SOP=Sonaroperatør, PCO=Bildeoppbygger, ACC=Flykontrollør)

4.3 Operatør oppgaveanalyse

Heretter vil vi konsentrere oss om utviklingen av den delen av totalsystemet som har med operatører å gjøre, dvs de oppgavene som ble tildelt operatørene under funksjonsallokeringen. Vi skal altså ikke diskutere funksjoner som har blitt allokert til program- eller maskinvare og som også må analyseres på en tilsvarende måte.

Analysen av operatøroppgaver innebefatter:

- Detaljering av operatøroppgaver
- Identifikasjon og detaljering av informasjonsbehov og -flyt
- Vurdering av krav (f eks kunnskap) som stilles til operatørene og arbeidsbelastning for hver av oppgavene og operatørens samlede jobb.

Merk at en overordnet funksjonsallokering ofte bare identifiserer at en funksjon skal utføres av operatørene, dvs man har ofte ikke identifisert hvilken operatør (operatørposisjon) og eventuelt hvor mange operatører som er nødvendig. En fordeling til bestemte operatørposisjoner danner nye grensesnitt. Det er essensielt å analysere koordineringsbehovet, beslutningshierarkiet og ansvarsfordelingen mellom operatørposisjoner. Analysen av den enkelte operatørposisjon kalles *jobbanalyse*. Analyse av operatøroppgaver danner altså grunnlaget for allokeringen *mellom* operatørene og jobb-beskrivelse for hver operatørposisjon. Allokering og oppgaveanalyse er en iterativ prosess.

I (Beevis *et al.*, 1992) finner vi følgende definisjoner på oppgave (eng. task) og oppgaveanalyse (eng. task analysis):

- **Task** - A composite of related operator or maintaner activities (perceptions, decisions, and responses) performed for an immediate purpose, e.g., «insert aircraft position» (after NATO STANAG 3994/1)
- **Task analysis** - A time oriented description of personnel-equipment-software interactions brought about by an operator, controller or maintainer in accomplishing a unit of work within a system or item of equipment. It shows the sequential and simultaneous manual and intellectual activities of personnel operating, maintaining, or controlling equipment (US MIL-H-46855B).

Oppgaveanalysen fokuserer altså på *hva* som skal gjøres og ikke *hvordan* (dvs skjermbilder og betjeningsorganer). Det siste behandles under konstruksjon av operatørgrensesnitt i kapittel 5.

Først må funksjonene som er tildelt operatøren dekomponeres til et passende nivå. Ideelt sett bør man prøve å identifisere *primæroppgaver* som er «den minste logiske definerbare enhet» som en operatør (evt. operatør og maskin) må utføre for å fullføre en oppgave. Primæroppgaver identifiseres ved at det er ett enkelt tegn/signal/symbol (se kapittel 3.3) som initierer utførelsen av det. Egenskapene til de enkelte primæroppgavene beskrives med tanke på vurdering av arbeidsbelastning og ferdighets- og kunnskapskrav. Operatørens aktivitet knyttet til primær-oppgaven kan kategoriseres kvalitativt, f eks hvorvidt den er motorisk, perseptuell eller

kognitiv.

I praksis vil en oppgaveanalyse sjelden være så detaljert som beskrevet over og dette vil vel heller ikke være nødvendig for alle oppgaver. Innsatsen bør være i samsvar med informasjonsbehovet til etterfølgende analyse- og konstruksjonsaktiviteter. Typisk innhold i en oppgavebeskrivelse er:

- Hva som initierer utførelsen av oppgaven
- Avslutningsbetingelse
- Kategori av oppgave: f eks skille mellom diskrete, kontinuerlige og beslutningsoppgaver
- Informasjonskrav (for å kunne utføre oppgaven)
- Utførelsesfrekvens
- Tidsbehov (fast verdi eller stokastisk beskrivelse)
- Andre relevante ytelsesmål (f eks nøyaktighet)
- Potensielle feil
- Betingelser til grensesnitt, arbeidsplass og arbeidmiljø

En slik beskrivelse kan brukes som utgangspunkt når man skal definere ytelseskrav til de enkelte oppgavene og samlede krav til hver av operatørposisjonene. Ofte benyttes en ferdighetsskala. Eksempler på ytelseskrav er:

- Responstid
- Tidsforbruk
- Nøyaktighet
- Planleggingsevne
- Prosedyrer

Oppgavebeskrivelsen kan være basis for følgende aktiviteter:

- Detaljert simulering og prototyping
- Ytelsesprediksjon
- Utvikling av prosedyrer, opplæring og dokumentasjon
- Kravspesifikasjon til MMK
- Test og evaluering av MMK

Beskrivelsen av operatøroppgavene kan systematiseres ved å benytte en bestemt syntaks og *taksonomi* (Lenorovitz *et al*., 1984). En taksonomi er det samme som klassifikasjon eller systematisering der alle termer defineres i en egen ordliste. Den benyttes for å oppnå et ensartet detaljeringsnivå både mellom ulike MMS-beskrivelser laget av forskjellige utviklere. Det finnes egentlig ikke noen veletablert og standard taksonomi. Vanligvis er det nødvendig å tilpasse en generisk taksonomi med prosjektspesifikke termer. I tillegg kan man definere en bestemt

syntaks, dvs definere regler for sammenstilling av ord til setninger (setningslære). Bruk av taksonomi og syntaks muliggjør databasert behandling av beskrivelsen.

Følgende kan være en enkel syntaks for beskrivelse av primæroppgaver:

Oppgaveverb [adjektiv] substantiv (informasjonsenhet(er)) [kommentar]

Eksempler på bruk av syntaksen er:

- Velg alternativ
- Detekter nytt mål
- Sammenlign radar_måling_1 og radar_måling_2

Eksempel på hierarkisk oppstilling av oppgaveverb er vist i tabell 4.4 under.

Eksempler på taksonomiadjektiver er: ny, temporær, verifisert, oppdatert, fjernet, reformatert, mulig, prediktert, fremhevet osv.

Det finnes mange teknikker for oppgaveanalyse (Beevis *et al.,* 1992, Kirwan & Ainsworth, 1992):

- Tidslinjeanalyse
- Operasjonssekvensdiagram
- Tabularisk oppgaveanalyse, f eks kritisk oppgaveanalyse
- Beslutningstabeller

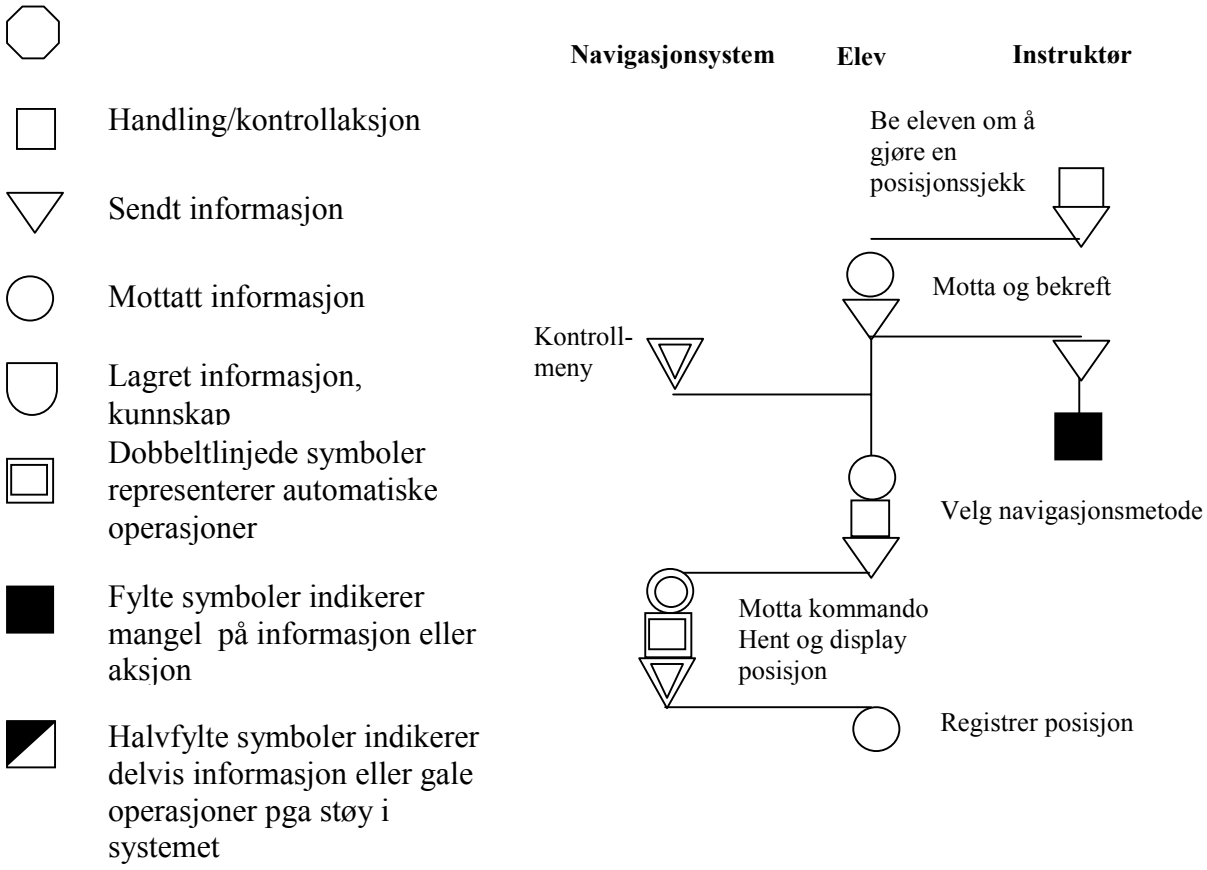
Tidslinjeanalyse (eng. timelines) kan utføres for en eller flere operatører og er en framstilling av oppgaver som funksjon av tid, som regel for ett spesifikt scenario. Diagrammet viser altså sekvenseringen av oppgavene inklusive eventuell parallellitet. Teknikken er i prinsippet enkel å bruke og kan gi utviklere kvantitativ informasjon relatert til gjennomførbarheten av tenkte operative oppgaver. Problemet med teknikken er selvfølgelig å estimere tidsforbruket, spesielt for mer kompliserte oppgaver.

- Perceptual behaviour
  - Searching and receiving information
    - Detects
    - Inspects
    - Observes
    - Reads
    - Receives
    - Scans
    - Surveys
  - Identifying objects, actions and events
    - Discriminates
    - Identifies
    - Locates
- Mediational behaviour
  - Information processing
    - .....
  - Problem solving and decision making
    - .....
- Communication behaviour
  - .....
- Motor behaviour
  - Simple/discrete
    - .....
  - Complex/continous

Tabell 4.5 Oppgavetaksonomi. Etter (Berliner et al,1964)

Operasjonssekvensdiagrammer (eng. Operational Sequence Diagrams) er en grafisk presentasjon av informasjonsflyt, beslutninger og aktiviteter i et system. Fem symboler benyttes for å representere aksjoner, beslutninger, sendt, mottatt og lagret informasjon, jfr figur 4.16. Sekvensering av oppgaver, inklusive valg og sløyfer, vises vertikalt, eventuelt med assosiert tidsinformasjon. I kapittel 3.5 kommenterte vi at OSD fort kan bli komplisert dersom man skal vise mange alternative grener i samme diagram. Representasjon av operatøroppgaver og tolkningen av symbolene kan variere mye og OSD er kanskje et (av mange) eksempel på hvordan man benytter teknikker på en uformell måte under systemutviklingen. Detaljeringsgrad og volum av analysen kan fort gjøre det vanskelig å lese/inspisere analysen. Tidslinjeanalyse og OSD kan brukes til å vurdere arbeidsbelastning og som basis for utarbeidelse av nettverksmodeller i f eks SAINT.

I såkalt *kritisk oppgaveanalyse* beskrives alle oppgaver som er klassifisert som kritiske (viktige) iht et format bestående av 17 ulike attributter (Beevis *et al.,* 1992) lignende de vi har beskrevet over.



Figur 4-16 Eksempel på OSD (Beevis et al., 1992)

Enkle teknikker som beslutningstabeller kan også brukes til å beskrive informasjonsbehovet for å ta avgjørelser. Teknikkene over kombineres ofte, f eks benyttes OSD sammen med tabulariske oppgavebeskrivelser.

En operatørorientert formell beskrivelse bør generelt ha følgende egenskaper:

- Grafisk representasjon
- Hendelser, tilstander, funksjon, inn- og utdata
- Eksplisitte beskrivelse av sekvens og tid
- Evne til å beskrive parallellitet

Teknikkene fra funksjonsanalysen og modellapparatet beskrevet i kapittel 3 kan følgelig brukes for å beskrive operatøroppgaver. Spesielt relevant er prosedyrebaserte modeller der operatøren betraktes som en hendelsessensitiv informasjonsbehandler hvor oppførselsmodellen inkluderer

informasjonsflyt ut og inn av oppgaver, sekvensering og parallellitet.

Visse typer oppførsel kan være vanskelig å beskrive, f eks stimulus som fører til at operatøren midlertidig avbryter en påbegynt oppgave eller dynamisk oppgavefordeling (Sheridan, 1994).

Mange av operatøroppgavene vil være relatert til KB oppførsel der følgende aspekter er viktige: kunnskap, oppfattelsesevne, hukommelse, forestillelsesevne, vurderingsevne og resoneringsevne. Analyse av denne typen oppgaver kalles *kognitiv oppgaveanalyse*.

Ulike teknikker for kognitiv oppgaveanalyse har blitt foreslått, se f eks diskusjonen i (Grant & Mayes, 1991). Grant & Mayes hevder imidlertid at det ikke er noen generell enighet om hvordan slike oppgaver skal beskrives. De mener at kognitiv oppgaveanalyse ihvertfall bør bidra til å beskrive *informasjonbehovene* til operatøren (for den aktuelle oppgaven) og konkluderer med at dette egentlig er alt som kognitiv oppgaveanalyse *kan* gjøres. Essens *et al.* (1994) gir en omfattende oversikt over kognitiv oppgaveanalyse. En spørreundersøkelse utført av Essens konkluderer med at det ikke finnes generelt aksepterte og brukte teknikker for analyse av kognitive aspekter knyttet til beslutningstøttesystemer og kravsetting av disse. Analysene var typisk ad-hoc og spesifikke for den enkelte applikasjon.

Det er klart at «tradisjonell» oppgaveanalyse som beskrevet tidligere vil være et nødvendig utgangspunkt for en kognitiv oppgaveanalyse. Essens *et al.* uttrykker dette slik: «It (i.e. task analysis) provides the temporal structure in which the cognitive performance operates and it gives a specification of the goal structure of tasks. Task analysis specifies the behavioural activities, aims and performance criteria». Informasjonsbehovet kan beskrives godt vha tradisjonell oppgaveanalyse.

Essens *et al*. lister opp følgende aspekter som berører kognitiv oppgaveanalyse:

- Nødvendig kunnskap for å kunne utføre oppgaven, relasjoner mellom og organisering av viktige konsepter.
- Mentale operasjoner for gjenfinning, lagring, transformasjon, integrasjon og modellering av informasjon.
- Metakognitive prosesser som kontrollerer kognitiv innsats og oppmerksomhet.
- Utvikling av kognitive ferdigheter og progresjon i kunnskapsstrukturer fra nybegynner til ekspert.

Målsetningen med en kognitiv oppgaveanalyse er i henhold til Essens *et al*. å lage en modell av oppgavene. Ved å analysere modellen kan man identifisere begrensninger i kognitiv oppførsel, og derved, mulighetene for å gi operatøren støtte. Målsetningen bør ifølge Essens *et al.* være å framskaffe kognitive krav til oppgaven som utvikleren kan forholde seg til. Modelleringen tar vanligvis utgangspunkt i en generell kognitiv modell, f eks likt det rammeverket vi beskrev i avsnitt 3.2. Denne modellen tilpasses deretter til den aktuelle anvendelsen og eventuelt de enkelte oppgavene. Kunnskapsinnsamling er nødvendig for å kunne utvikle modellen. Teknikkene som benyttes er i stor grad de samme som ble beskrevet ved operatøranalyse i

kapittel 4.1 og vanlige teknikker for funksjons- og oppgaveanalyse.

Analysen av modellen tar utgangspunkt i kunnskap innenfor kognitiv psykologi. Spesielt viktig er kunnskap om såkalte beslutningsavvik (eng. biases) som er heuristikk som fører til feilaktige resultater i forhold til normativ teori. En oversikt over slike avvik finnes i (Hogarth, 1987) og et eksempel er *konservatisme* i informasjonsbehandling: Mennesket har en tendens til *ikke* å oppdatere vurderinger i lys av ny informasjon.

Representasjon av kunnskap innenfor kognitiv psykologi benytter konsepter som har likhet med de man finner innenfor objekt-orientering. Det kan derfor være av interesse å undersøke hvorvidt objekt orientert modellering også kan bidra ved kognitiv modellering og oppgaveanalyse. Se f eks Rhine & Bargava (1992).

5 SPESIFIKASJON OG KONSTRUKSJON AV OPERATØRGRENSESNITT

There is more to a user interface than meets the eye.

Dette kapitlet omhandler konstruksjon av MMK eller operatørgrensesnittet. Det inkluderer bl a beskrivelse av informasjons- og kontrollflyt mellom maskin og operatør, koding av objekter i skjermbilder og bestemmelse av hvordan operatøren skal utføre kommandoer. Dette er altså ikke konstruksjon av programvaren som realiserer operatørgrensesnittet, men en spesifikasjon til programvarekonstruksjonen. Konstruksjon av operatørgrensesnitt tilhører altså spesifikasjons-fasen i en systemarbeidsmodell for programvareutvikling. I praksis er imidlertid ikke skillet så skarpt, da store deler av realiseringen av et operatørgrensesnitt gjøres vha større byggeklosser som begrenser og setter rammene for funksjonalitet og stil. I neste kapittel kommer vi kort inn på programvaren som realiserer operatørgrensesnittet og verktøy som benyttes.

Konstruksjonsprosessen som behandles nedenfor er en strukturert «ovenfra-ned»-prosess hvor ett av hovedpoengene er at det skilles klart mellom *hva* som kommuniseres over operatør-grensesnittet og *hvordan* denne kommunikasjonen gjennomføres. Dette har en del fellestrekk med ISOs (International Standardization Organization) OSI-modell (Open System Inter-connection) for kommunikasjon. Ved å legge an konstruksjonsprosessen på denne måten er det lettere å oppnå et mer enhetlig og konsistent grensesnitt på tvers av funksjoner og operatørposisjoner.

Før vi går nærmere inn på konstruksjonsprosessen skal vi gi noen generelle betraktninger omkring operatørgrensesnitt.

5.1 Generelle betraktninger

Det er i hovedsak to forskjellige synsvinkler som vi kan ha på hva et operatørgrensesnitt er. Det ene er kommunikasjonsorientert, mens det andre er imitasjons/simuleringsorientert.

Den kommunikasjonsorienterte betraktningsmåten ser på grensesnittet som to parter som kommuniserer, operatøren og maskinen, og at grensesnittet er språket som operatør og maskin benytter for kommunikasjonen. Ved et slikt syn vil altså konstruksjon av operatørgrensesnittet være konstruksjon av det språket som skal benyttes. Dvs at konstruksjonsprosessen bestemmer alfabet, ord og grammatikk for operatørgrensesnittet. Begrepsapparatet som benyttes bærer preg av et slik språkperspektiv, ved at vi f eks snakker om semantisk, syntaktisk og leksikalsk konstruksjon.

Den andre måten å betrakte et operatørgrensesnitt på er å si at operatørgrensesnittet "åpner et vindu" inn i en verden som operatøren kan manipulere. For våre typer MMS vil verdenen bestå av den prosessen som vi ønsker å styre og overvåke. Konstruksjon av grensesnittet er da å utarbeide måter å presentere og manipulere prosessen på som er mest mulig lettfattelig innenfor de teknologiske begrensningene som finnes. For dette benyttes metaforer og analogier som vi kjenner igjen fra den virkelige verden.

5.2 Noen viktige prinsipper for konstruksjonsprosessen

Det å konstruere operatørgrensesnitt har endel trekk som særpreger konstruksjonsprosessen og som det er viktig å være klar over når man går i gang med arbeidet.

Pr i dag er ikke konstruksjon av operatørgrensesnitt tuftet på et vitenskapelig fundament og vi kan heller ikke si at det praktiseres som en ingeniørdisiplin. Årsakene til dette kan være flere. Konstruksjon av andre typer systemer bygger i stor grad på at det er mulig å modellere og prediktere en rekke forhold ved produktet før det bygges. Dette er ikke tilfelle for konstruksjon av operatørgrensesnitt, delvis fordi en av systemkomponentene er et menneske. En annen årsak er at det er svært kort tid (15-20 år) at det har vært arbeidet med operatørgrensesnitt i tilknytning til kognitive systemfunksjoner. Imidlertid ser vi klare trekk til at det etter hvert har blitt en profesjon og vi ser også en utvikling mot en ingeniørdisiplin. Men i dag er det for mye ad hoc og alt for personavhengig og tilfeldig til at det kan sies å være godt nok. Konsekvensen av dette er at vi må ta som utgangspunkt at vi ikke helt vet hva vi holder på med. Vi må legge opp konstruksjonsprosessen slik at vi tillater å prøve og feile, eller med andre ord vi må prototype/-simulere operatørgrensesnittet og legge opp til at ting må gjøres om igjen (iterativ prosess). Vi må altså utnytte tilbakekobling under utviklingen. En annen strategi er å legge opp til en inkrementell utvikling, hvor vi først utvikler en kjernefunksjonalitet som vi er helt sikre på, og utvider denne etter hvert som vi får tilbakemelding fra brukerne/markedet.

Et annet særtrekk med konstruksjon av operatørgrensesnitt er at det er et stort antall små, detaljerte konstruksjonsavgjørelser som hver i sær kan synes ubetydelige, men som til sammen er svært så betydningsfulle for hvordan operatørgrensesnittet som helhet oppfattes. Dette gjør det viktig å løfte hodet og ha en overordnet filosofi for arbeidet. Hvis ikke får man problemer med å se skogen for bare trær og det ender gjerne opp i en ad hoc prosess. En annen lærdom som kan trekkes fra dette er at det er viktig at man har tilstrekkelig oppmerksomhet på de små detaljer og «ubetydeligheter».

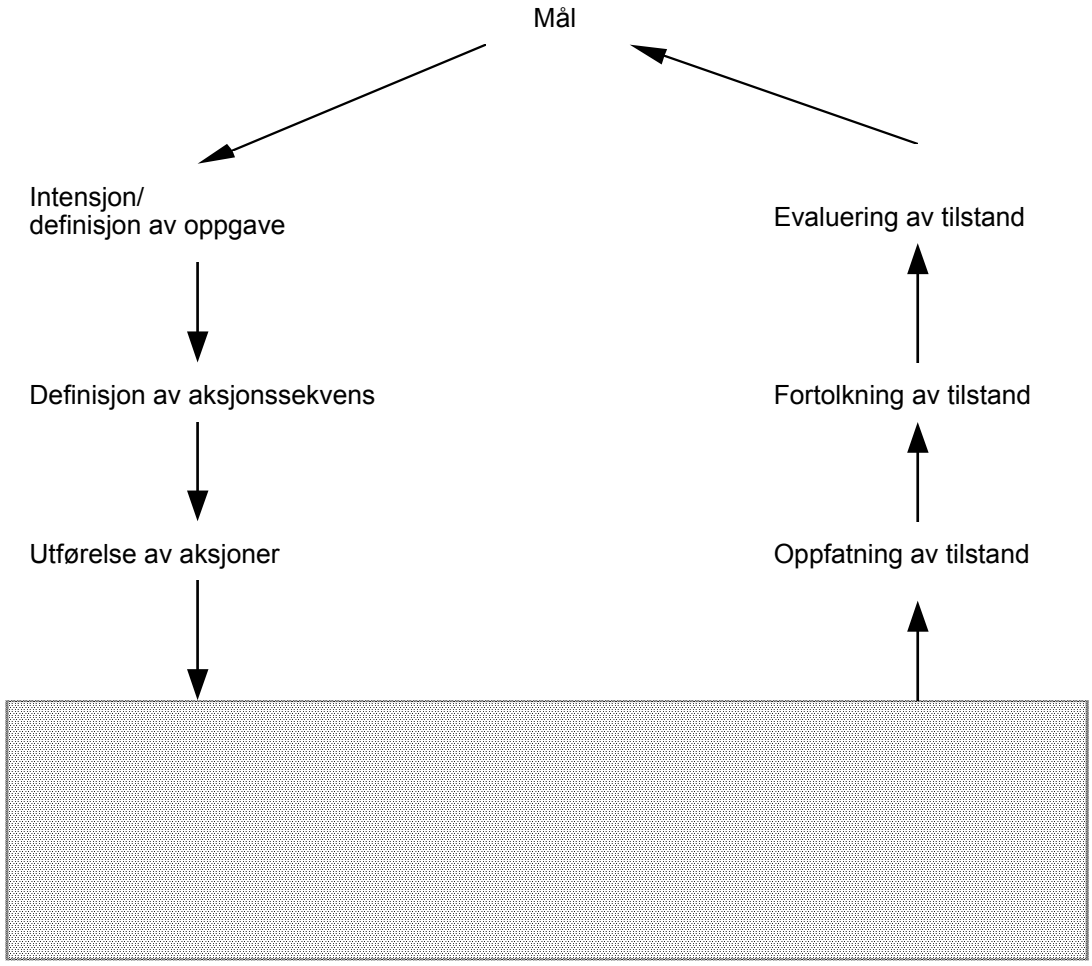
Nettopp pga alle de små detaljer, vil det *alltid* oppstå konflikter med de prinsipper og retningslinjer som etableres før konstruksjonsprosessen starter. Det er ikke noe som heter et konsistent operatørgrensesnitt, selv om et konsistent grensesnitt er et av kjennetegnene på et godt operatørgrensesnitt. Det er disse konfliktene som gjør konstruksjon av operatørgrensesnitt spesielt utfordrende.

Hva er det som karakteriserer et godt operatørgrensesnitt? Nedenfor er det trukket frem noen punkter som til stadighet blir påpekt som viktige:

1. *Konsistens*. Operatørgrensesnittet må være konsistent og enhetlig i alle sine deler.
2. *Tilbakekobling*. Operatøren må hele tiden få tilbakemelding på det han eller hun gjør. Tilbakekoblingen må gjøres på alle nivåer, dvs semantisk (funksjonelt), syntaktisk og leksikalsk. Eksempel på leksikalsk tilbakekobling er forflytning av skrivemarkør (eng. cursor) og ekko av tegn på en skjerm ved inntasting av disse. Syntaktisk tilbakekobling er f.eks. visning av at et objekt eller et alternativ i en meny er valgt, mens semantisk tilbakekobling f eks gir tilbakemelding om at en valgt funksjon virkelig er utført. Tilbakekobling (se appendiks B) reduserer usikkerhet mhp systemets respons og oppførsel og kan forbedre ytelsen.
3. *Systemtilstand*. Systemet må ikke skjule noen av sine tilstander for brukeren. Dette punktet er delvis en konsekvens og avledning av punktet ovenfor.
4. *Funksjonell utforming*. La grensesnittet ved sin utforming vise frem sin funksjonalitet. Noe av grunnen til at operatørgrensesnitt kan være vanskelig å bruke er at vi i dag har så få ting å spille på. Stort sett viser prosessen og funksjonaliteten i grensesnittet seg gjennom en liten flat skjerm. Som en motsetning kan vi tenke på hvordan fysiske objekter i en helt annen grad ved sin utforming viser hva de kan brukes til og hva de ikke kan brukes til.
5. *Mapping*. Det må være enkle overganger fra intensjon til aksjon, og fra informasjonskoding til informasjon tilrettelagt for fortolkning. Altså, enkle overganger mellom intensjon, aksjon og respons. Se figur 5.1 som viser de forskjellige stegene i en aksjonssyklus og poenget er at det ikke skal være komplekse (mentale) sprang mellom stegene. Legg merke til at denne modellen til Norman (1986) har mye til felles med stegene i en beslutningsprosess som vi omtalte i kapittel 3.
6. *Avlast korttidshukommelsen*. I sin kjente artikkel «The magic number seven, plus or minus two: some limits on our capacity for processing information» påpekte Miller (1956) at mennesker bare er i stand til å holde rede på 7±2 informasjonsenheter («chunks of information») samtidig (dvs i korttidshukommelsen) og vi glemmer fort (det som er i korttidshukommelsen). Dvs at operatørgrensesnittet må utformes slik at man til stadighet slipper å huske på enkeltheter og detaljer. Ved fornuftig bruk kan innholdet på skjermen virke som en avlastning og utvidelse av kapasiteten til korttidshukommelsen.
7. *Minimaliser feilmuligheter*. Utform operatørgrensesnittet slik at det ikke er mulig å gjøre feil. Dette er en mye bedre strategi enn først å la brukeren gjøre feil for så å ha angrefunksjoner for å gjøre om det som allerede er gjort.
8. *Muliggjør feilretting*. Hvis det fremdeles er mulig å gjøre feil selv etter at man har fulgt regelen over, skal det så og si alltid være mulig å rette feil. For alle ikke-reversible funksjoner skal det innføres spesielle mekanismer. Enkel feilretting oppnås med å forsøke å bygge opp

kommandovokabularet av små reversible kommandoer (se imidlertid neste kapittel om en slik strategi kontra et oppgaverelatert kommandovokabular).

9. *Unngå (uønskede) systemmoder*. Moder betyr at systemet bringes i en slik tilstand at operatørens muligheter begrenses eller at en viss funksjonalitet bare er tilgjengelig i visse systemtilstander. Ikke alle systemmoder er uønskede og i noen tilfeller kan det være ønskelig å benytte moder, f eks ved viktige sekvensielle oppgaver som oppstart av et prosessanlegg, men de skal benyttes med varsomhet og med en klar begrunnelse.
10. *Ortogonalitet/Uavhengighet*. Operatørens kommandovokabular må bestå av uavhengige kommandoer som ikke påvirker hverandre. Dvs, en aksjon skal være lokal og skal ikke ha konsekvenser for andre funksjoner. Det krever som regel endel arbeid og kreativitet for å få til dette.
11. *Enkelhet* ("KISS - Keep it simple, stupid"). Med datateknologi er det enkelt å foreslå funksjonalitet med et vell av muligheter og variasjoner. Dette fins det mange eksempler på bla innenfor forbrukerelektronikk. Dette har imidlertid sin pris på flere måter. En konsekvens er at funksjoner som i utgangspunktet er enkle blir mer kompliserte enn nødvendig. På en måte er det den mest kompliserte funksjonen med de fleste parametrene som setter premissene for alle de andre funksjonene. Gjør funksjonaliteten så enkel som mulig, men ikke enklere, og vær varsom med å innføre funksjonalitet som bare er «kjekt å ha».



Figur 5-1 Operatørinteraksjon, basert på Norman (1986)

5.3 Konstruksjon av menneske-maskin-kommuniksajon

Opp gjennom de senere årene har det blitt foreslått en rekke forskjellige metoder som kan benyttes når vi skal utvikle et operatørgrensesnitt. Noen eksempler er Thimbleby (1990), Shneiderman (1992), Sutcliffe (1989), Booth (1989), Laurel (1990), Hellander (1988), Winogard og Flores (1986) og Norman og Draper (1986). Den kanskje mest kjente og mest veletablerte er den metoden som ble utviklet av Foley tidlig på 80-tallet, se f eks Foley *et al.* (1990). Den tar utgangspunkt i den kommunikasjonsorienterte betraktningsmåten og bruker språkanalogi og begreper fra lingvistikk. Det er en «ovenfra-ned»-konstruksjonsmetode og gjør et skille mellom *hva* som kommuniseres og *hvordan* det kommuniseres. Konstruksjon av operatørgrensesnittet består i denne metoden av å konstruere språket som benyttes ved kommunikasjonen. Egentlig må vi konstruere *to* språk, ett språk som benyttes av operatøren for å kommunisere med maskinen og ett språk som maskinen benytter for å kommunisere med operatøren. Grunnen til at vi deler det i to språk er at maskinen og operatøren har så forskjellige virkemidler å benytte i kommunikasjonen og også selvsagt fordi at det er så ulike partnere som skal gjennomføre en «samtale».

Som nevnt er denne konstruksjonsmetoden en «ovenfra-ned»-metode med alle de ulempene slike metoder har. Konstruksjonsprosessen gjennomføres derfor sjelden på den måten som beskrevet her. Likevel vil det være en fordel å beskrive operatørgrensesnittet som om en slik strukturert metode var fulgt. Dette gir en beskrivelse som gir et godt skille mellom de forskjellige aspektene av grensesnittet. Det er altfor vanlig å se beskrivelser av operatør-grensesnitt som f eks ikke skiller mellom forhold som har med de funksjonelle sidene av grensesnittet å gjøre og selve kodingen av data. Dette gjør det vanskelig å ha en meningsfull diskusjon og samtale om operatørgrensesnitt. «Alle» har meninger og oppfatninger om operatørgrensesnitt, men uten å ha et egnet begrepsapparat til å uttrykke meningene sine med. Dette, enkelt å ha oppfatninger og mangel på begrepsapparat, lider fagfeltet av og har kanskje gjort at det ikke har fått den status som det fortjener. Vi mener at dette delvis også skyldes de som har arbeidet innenfor feltet. Altfor mye av det som er skrevet og sagt har bestått av en samling av anekdotisk kunnskap som er opphengt i den teknologien som er den mest populære for øyeblikket. Det er i alt for liten grad gjort forsøk på å danne grunnlaget for en «teori» for operatørgrensesnitt.

Den konstruksjonsmetoden som er kort beskrevet nedenfor, kan ses i et slikt lys i tillegg til å beskrive steg i en konstruksjonsprosess. Den kan ses på som et forsøk på å etablere en teori (hvor et begrepsapparat er ett element) og rammeverk for MMK. MMK-en deles opp i fire lag eller nivåer. Disse betegnes *konseptuelt, semantisk (alternativt funksjonelt), syntaktisk og leksikalsk* nivå. Hvert av disse lagene består igjen av to deler, operatør til maskin og maskin til operatør. Det har også blitt foreslått utvidelser og variasjoner på disse lagene, se f eks Nielsen (1986).

Nedenfor er det beskrevet hvert av de fire lagene og hvilke konstruksjonsavgjørelser som foreskrives i hvert av de fire tilsvarende stegene som konstruksjonsprosessen består av.

5.3.1 Konseptuelt nivå og konseptuell konstruksjon

Konstruksjonsprosessen som beskrives her ble opprinnelig utarbeidet uten å være satt inn i en systemsammenheng. Operatørgrensesnittet ble behandlet i isolasjon. Derfor faller dette nivået/steget sammen med oppgaveanalysen som ble beskrevet i forrige kapittel. I den opprinnelige formen dekket dette trinnet definisjon av de konseptene som operatøren måtte beherske. Nivået består av en definisjon av objekter og operasjoner på disse. Relasjoner mellom objekter dekkes også. Dette er det samme som ble klarlagt i oppgaveanalysen og blir ikke beskrevet videre her.

En annen oppgave som tilhører dette steget er å etablere hovedprinsipper for interaksjon og informasjonspresentasjon. Disse formuleres i *retningslinjer* som er med på å styre konstruksjonsavgjørelsene i de påfølgende stegene. Å etablere et sett med retningslinjer er viktig for å oppnå konsistente grensesnitt, spesielt hvis det er flere som er involvert i konstruksjonen. Slike retningslinjer kan gjelde på tvers av flere prosjekter og det er blitt mer og mer vanlig at slike retningslinjer blir utarbeidet på bedriftsnivå.

Retningslinjer for operatørgrensesnitt

Det er blitt utarbeidet flere generiske samlinger av retningslinjer som kan benyttes som utgangspunkt ved etablering av retningslinjer for et spesifikt prosjekt, klasse av systemer eller en bedrift. Den mest kjente og voluminøse av disse er Smith og Mosier (1986) fra MITRE Corporation. Andre eksempler er Gilmore, Gertman og Blackman (1989) med retningslinjer rettet mot prosessautomatiseringssystemer og Brown (1988) som inneholder generelle retningslinjer for datamaskin operatørgrensesnitt. Slike retningslinjer vil altså bare danne et utgangspunkt og må bearbeides for å være nyttig i et gitt prosjekt.

For det første er det nødvendig å gjøre et utvalg og skrive de om til operative retningslinjer slik at konstruksjonen senere kan testes for å se om de er oppfylt. Det bør også settes prioritet på de enkelte retningslinjene da det alltid vil oppstå konflikter (referer tilbake til konsistens i grensesnitt) og man bør legge opp til at retningslinjene vil måtte modifiseres underveis. Dagens retningslinjer danner ikke et så solid grunnlag at det bare er mulig å teste konstruksjonen opp mot disse for å validere grensesnittet. Det er i tillegg nødvendig å gjøre evalueringer av operatørgrensesnittet i en prototyp.

Noe av problemet med dagenes generelle retningslinjer er at de på den ene siden er for allmenngyldige og at de på den annen side er for spesifikke. De retningslinjene som er for generelle kan på mange måter beskrives som «sunt bondevett», mens det gjerne knytter seg usikkerhet til om de mer spesifikke har gyldighet for det spesielle systemet man arbeider med. For å komme frem til retningslinjene er det gjerne gjort laboratorieforsøk hvor alle forholdene er forenklet og lagt til rette for å få svar på ett spesielt spørsmål. Det kan derfor være tvil om retningslinjen også er gyldig i en mer sammensatt og kompleks situasjon. Et annet problem med de generelle retningslinjene som har noe av samme bakgrunn, er at de ikke er konsistente. Den ene retningslinjen slår den andre i hjel.

Altså, generelle retningslinjer må benyttes med forsiktighet og utforming av egne retningslinjer må gjøres med omhu.

En annen mulig kilde for å utarbeide egne retningslinjer kunne tenkes å være håndbøker over menneskelig yteevne, se f eks Boff *et al.* (1989). Problemet er her at man bør unngå å utforme grensesnittet slik at det kreves av operatøren at han opererer på grensen av sin yteevne. Derfor er slike håndbøker av begrenset nytteverdi, men kan gi nyttig veiledning i mer ekstreme situasjoner. Slike situasjoner er aktuelle i militære systemer.

Eksempler på retningslinjer er:

- Defaultverdier skal benyttes hvor det mulig
- Defaultverdier skal være kontekst sensitive
- Menyvalg skal settes opp etter brukshyppighet
- Antall valg på et menynivå skal ikke overskride åtte

Generiske retningslinjer og håndbøker over menneskelig yteevne har nå kommet på CD-ROM og det er laget programvaresystemer som gjør de lettere tilgjengelig og lettere å bruke.

5.3.2 Semantisk nivå og semantisk konstruksjon

Dette nivået/steget definerer *hva* som skal kommuniseres. Nivået kalles også det funksjonelle nivået da operatørgrensesnittets detaljerte funksjonalitet bestemmes. Det kalles semantisk da det er meningsinnholdet av operatørgrensesnittet som defineres.

Den delen av dette nivået som omhandler "fra operatør til maskin" består av å definere alle funksjonene/aksjonene som operatøren skal kunne gjøre og resultatene av å utføre funksjonene.

Den delen som omhandler "fra maskin til operatør" består av å definere den informasjonen som skal gjøres tilgjengelig for operatøren. Dvs at alle informasjonsenheter/objekter som har en mening i seg selv skal identifiseres. For et militært system vil dette f eks bestå av mål, sensorer, målinger, osv. En måte å strukturere beskrivelsen av det semantiske nivået av kommunikasjonen fra maskin til operatør på, er å benytte følgende notasjon:

+	–	og
{}	–	iterasjon av
[|]	–	ett av flere mulige alternativer
()	–	opsjon

For å illustrere bruken av denne notasjonen er det nedenfor gjengitt et kort utdrag av en beskrivelse av det semantiske nivået for et kartbilde:

```
Chart = own_craft + {safety_contour}no_safety_contours + ([grid1|grid|grid3])+   …
```

“Fra operatør til maskin”-delen består i å definere de funksjonen/kommandoene som operatøren skal kunne utføre, deres parametere og hva som er resultatet av dem. Også mulige feil som kan oppstå beskrives. Det er en fordel at alle kommandoer beskrives på en ensartet, standardisert måte.

Semantisk nivå beskriver altså alle funksjoner som operatøren skal kunne utføre og alle informasjonsenheter som skal kunne presenteres, men ikke hvordan operatøren skal utføre funksjonene og heller ikke hvordan informasjonsenhetene blir presentert. Disse problemstillingene dekkes av det neste nivået.

5.3.3 Syntaktisk nivå og syntaktisk konstruksjon

Det syntaktiske nivået definerer reglene for sammensetninger av de delene som hver funksjon består av. Videre defineres koding, inkludert plassering av og tidsmessige forhold (når og under hvilke betingelser blir en informasjonsenhet presentert?) knyttet til informasjonsenhetene.

"Fra operatør til maskin" består det syntaktiske nivået av å beskrive rekkefølgen/sekvensen (reglene for sammensetning) av de delene som en funksjon er bygget opp av og å knytte dette til *betjeningsteknikker og logiske betjeningsorganer*. F eks bestemmes rekkefølgen på valg av objekt og funksjon og rekkefølgen til eventuelle parametere til funksjonen. For kommandospråk (se neste kapittel) er den syntaktiske konstruksjonen klar og lett å forstå. For direkte manipulerende operatørgrensesnitt (se neste kapittel) trer de syntaktiske reglene som er benyttet ikke så klart frem, men de er der og må defineres.

"Fra maskin til operatør" består dette nivået av hvordan informasjonsenhetene som ble bestemt i det semantiske nivået kodes. Det må altså avklares hvilke kodingsteknikker som egner seg best ut fra hva som skal formidles og for hvilken oppgave operatøren skal benytte informasjonen. Utfordringen består i å finne den rette representasjon av dataene slik at informasjonsoverføringen får så stor kapasitet og går så lett som mulig. Hvis dette gjøres bra kan vi øke innformasjonsoverføringen med en størrelsesorden to i forhold til en mindre egnet koding, Fitts (1962). Representasjonen/koding må også være tilrettelagt for det dataene skal benyttes til av operatøren. Et eksempel er hvilket tallsystem som benyttes og om tall representeres numerisk eller grafisk.

5.3.4 Leksikalsk nivå og leksikalsk konstruksjon

Det leksikalske nivået definerer de konkrete elementene som benyttes for kommunikasjonen.

"Fra operatør til maskin" består det leksikalske nivået å bestemme *fysiske betjeningsorganer*.

"Fra maskin til operatør" består det leksikalske nivået av de primitivene som er nødvendige for de kodingsteknikkene som benyttes. For den visuelle presentasjonen betyr dette å avklare hvilke egenskaper det grafiske språket må ha for f eks å kunne presentere de symbolene som ble definert i det syntaktiske steget. Med andre ord, den representasjonen/kodingen som informasjonselementene har fått brytes ned i sine enkelte bestanddeler (f eks linjer, sirkler osv) og deres

attributter (f eks linjetykkelse, linjetype, farge, osv).

Normalt vil ikke leksikalsk konstruksjon bli gjennomført for å bestemme kravene til f eks den grafikkpakken som kan benyttes, men den *kan* gjennomføres for å oppnå en detaljert spesifikasjon til programvareingeniørene for å unngå misforståelser og begrense valgfriheten til den enkelte programmerer. Likeledes vil valg av fysiske betjeningsorganer i mange tilfeller være begrenset uten rom for de store variasjonene.

5.4 Betjeningsoppgaver og betjeningsteknikker, logiske og fysiske betjeningsorganer

I det semantiske nivået ble funksjonene eller aksjonene som operatøren har tilgjengelig gjennom operatørgrensesnittet definert. For å utføre disse funksjonene benyttes *betjeningsoppgaver* for å overføre de informasjonsenhetene som skal til for å utføre funksjonen. De betjeningsoppgavene som går igjen i så og si alle operatørgrensesnitt er, Foley, Wallace og Chan (1981):

- Angi posisjon
- Skrive tekst
- Velge fra et mengde av mulige alternativer (kan være objekter eller funksjoner)
- Angi tallstørrelse

Den informasjonsenheten som overføres ved de fire forskjellige betjeningsoppgavene er en posisjon (1,2 eller 3-dimensjonal), en tekststreng, en identifikasjon og en tallverdi.

For å gjennomføre disse betjeningsoppgavene kan vi benytte flere ulike *betjeningsteknikker*. Konstruksjonen består derfor av å bestemme den mest egnede betjeningsteknikk for betjeningsoppgavene. F eks kan en posisjon angis ved å peke med en mus på en linje, flate eller i et rom eller å taste posisjonen inn som tallverdier gitt i et definert koordinatsystem.

Betjeningsteknikker kan klassifiseres som *direkte* eller *indirekte* og om det benyttes en *referanse* (eller identifikator) eller ikke. Dette gjelder spesielt for betjeningsteknikker for betjeningsoppgaven *å velge*. De forskjellige typene betjeningsteknikker er:

- Direkte uten referanse - Valg ved direkte kontakt med en representasjon.
- Direkte med referanse - Valg ved direkte kontakt med referansen til en representasjon.
- Indirekte uten referanse - Valg ved nærhet av tilbakemelding, produsert av et betjeningsorgan, til en representasjon.
- Indirekte med referanse - Referanse til en representasjon velges ved nærhet av en tilbakemelding fra et betjeningsorgan.

F eks er valg av et fartøy ved å velge et symbol som representerer fartøyet i et kart en indirekte betjeningsteknikk uten referanse. Valg av et fartøy vha en trykkfølsom skjerm ved å peke i en liste over fartøysnummer er en direkte betjeningsteknikk med referanse.

En betjeningsteknikk benytter *betjeningsorganer*. Vi skiller mellom *logiske og fysiske*

betjeningsorganer for å kunne klassifisere dem og kunne sette betjeningsorganer inn i et rammeverk etter hvert som nye blir utviklet. Videre skilles det mellom *direkte og indirekte* betjeningsorganer som samsvarer med direkte og indirekte betjeningsteknikker. F eks er en trykkfølsom skjerm et direkte betjeningsorgan, mens mus og rulleball er indirekte betjeningsorganer. Følgende logiske betjeningsorganer er definert, Foley *et al.* (1981):

- Alfanumerisk tastatur - benyttes for å skrive tekst og å angi tallverdier
- Velger - benyttes for å velge objekter/funksjoner
- Posisjoneringsorgan - benyttes for å spesifisere en 1, 2 eller 3-dimensjonal posisjon
- Potensiometer - benyttes for å angi en tallstørrelse

Det skilles mellom *absolutte* og *relative* posisjoneringsorganer avhengig om de gir verdier i et definert koordinatsystem eller om de gir relative verdier. En trykkfølsom skjerm er et absolutt posisjoneringsorgan mens mus og rulleball er relative. Det skilles også mellom om posisjoneringsorganer er *kontinuerlige* eller *diskrete.* F eks er mus kontinuerlig, mens kursorkontrolltaster er diskrete.

Noen av de mest vanlige fysiske betjeningsorganene som benyttes i dag er kort beskrevet i neste kapittel.

Tabell 5.1 viser noen eksempler på sammenhengen mellom betjeningsoppgaver, betjenings-teknikker, og logiske og fysiske betjeningsorganer for betjeningsoppgaven angi en posisjon. Tilsvarende sammenhenger kan lages for de andre betjeningsoppgavene. Et slikt skjema kan man tenke seg å benytte til å utvikle alternative løsninger. Det kan også benyttes som en «veiviser» for en MMK-konstruktør.

5.5 Informasjonskoding

Når vi overfører informasjonsenheter til operatøren må vi stille oss spørsmålet *hva* vi ønsker å formidle til operatøren. Noen typiske eksempler på hva vi ønsker å formidle er:

- Skalare og vektorer av reelle tall
- Retninger
- Tidsderiverte
- Posisjoner
- Rekkefølger
- Identifikasjon
- Klassifikasjon
- Alarmer

Betjeningsoppgave	**Betjeningsteknikk** **Logisk betjeningsorgan**	**Eksempel på fysiske betjeningsorganer**
Angi posisjon	Direkte med et posisjoneringsorgan	Trykkfølsom skjerm
	Indirekte med et kontinuerlig posisjoneringsorgan	Mus, Styrespak, Rulleball
	Indirekte med et diskret posisjoneringsorgan	Kursorkontrolltaster
	Med et numerisk tastatur	Numerisk tastatur
	Direkte med en velger	Penn

Tabell 5.1 Betjeningsoppgaver, betjeningsteknikker og betjeningsorganer

Ut fra det vi ønsker å kode kan vi benytte en rekke forskjellige kodingsteknikker. Noen er rett fram: ønsker vi å formidle en posisjon viser vi den i et egnet koordinatsystem og ønsker vi å angi klasser (typer) gjøres det gjerne vha et symbolbibliotek hvor hver symboltype angir en bestemt klasse. Noe av det vi har å spille på for å oppnå en god koding/representasjon av den informasjonen vi ønsker å formidle er:

- Blinking
- Farger
- Intensitet
- Skrifttyper
- Avstand
- Linjetyper
- Linjetykkelse
- Bevegelse
- Orientering
- Posisjon
- Symboler
- Størrelse
- Lyd

Tilsvarende som tabell 5.1 viste sammenhengen mellom betjeningsoppgaver, -teknikker og -organer, kan det lages en sammenheng mellom hva som ønskes å kodes og hvilke kodingsteknikker som egner seg best. Det å utvikle slike sammenhenger kan lette arbeidet med å frembringe alternativer og kan benyttes som en «veiviser» på tilsvarende måte som tabell 5.1 indikerer.

6 DIALOG, BETJENINGSORGANER, INFORMASJONSGIVERE OG PROGRAMVARE

Dette kapitlet tar for seg noen få sider ved realisering av et operatørgrensesnitt. Mens det foregående kapitlet beskrev de forskjellige konstruksjonstrinnene ved utforming av et operatørgrensesnitt og et begrepsapparat for dette, vil vi her gå inn på de bestanddelene som et operatørgrensesnitt er bygget opp av.

Først vil vi gå gjennom noen egenskaper ved de mest benyttede typer av dialoger. Deretter gis en gjennomgang av de mest benyttede betjeningsorganer og prinsipper for koding og organisering av data i skjermbilder. Til slutt gir vi en kort beskrivelse av programvaretekniske forhold i forbindelse med realisering av operatørgrensesnitt.

Dette kapitlet gir en svært kortfattet beskrivelse og er ikke ment å dekke disse områdene fullt ut. For mer fyllestgjørende beskrivelser bør man konsultere andre deler av litteraturen. Henvisning til noe av denne litteraturen er gitt nedenfor.

Kapitlet er oppdelt slik at det skilles mellom informasjonsflyt fra operatør til maskin og fra maskin til operatør. Først beskrives hovedtyper av dialoger for datainnmating og funksjonsaktivering samt fysiske betjeningsorganer som benyttes til dette. Deretter gjennomgås egenskaper til noen kodingsteknikker og fysiske informasjonsgivere.

6.1 Utviklingstrinn

Operatørgrensesnittet bygges gjerne opp vha flere forskjellige dialogtyper. Hvilke som velges og hvordan de benyttes bestemmer operatørgrensesnittets *stil*, dvs dets utseende og oppførsel (eng. look & feel). Introduksjon av de forskjellige dialogtypene har vært nært knyttet til tilgjengelig teknologi. Frem til i dag kan det skilles mellom tre generasjoner av teknisk utstyr og vi ser fremveksten av den fjerde. De forskjellige generasjonene er:

Dedikerte informasjonsgivere og betjeningsorganer

1. Tegnbaserte og grafiske terminaler
2. Arbeidsstasjoner og PC-er
3. Kunstig virkelighet

De forskjellige generasjonene er karakterisert av egenskapene til utstyret som benyttes for informasjonsutvekslingen mellom operatør og maskin, båndbredden mellom dette utstyret og prosessor (hvis det i det hele tatt benyttes en prosessor) og prosessorkapasitet. Generasjonene er også karakterisert av hvor tett tilbakekoblingen (i tid og datamengde) er mellom operatør og maskin.

Første generasjon er karakterisert av at hver informasjonsgiver (i hovedsak viserinstrumenter) bare viste én enkelt måling og at betjeningsorganer som knapper, brytere og potensiometre bare styrte én forhåndsbestemt variabel. Denne generasjonen er preget av at operatøren må integrere

de enkelte måleverdiene for å få et mentalt "bilde" av tilstanden til prosessen. Betjeningen er karakterisert av mange betjeningsorganer med hver sin bestemte funksjon. Bruk av datateknologi er ikke-eksisterende eller svært beskjeden.

Ved innføring av terminaler får operatøren tilgang på kommandospråk, og enkle skjemaer og menyer for å styre og gi inn verdier til maskinen. Bruk av grafiske terminaler (først monokrome vektorskjermer, så farge rasterskjermer) gav mulighet for å presentere data ikke bare på alfanumerisk form. Det er imidlertid ikke særlig skille på hvordan operatøren betjener systemet om det benyttes tegnbaserte eller grafiske terminaler. Dette er på grunn av lav båndbredde til prosessor og mangel på lokal datakraft i terminalen.

Situasjonen i dag er karakterisert ved bruk av arbeidsstasjoner og PC-er, med lokal datakraft koblet til kraftige prosesser vha høyhastighets datanettverk. Denne generasjonen er preget av utstrakt bruk av grafikk for datapresentasjon, og operatøren har gjennom bruk av analogier/-metaforer direkte tilgang til grafiske representasjoner av fysiske objekter og imitasjon av tilsvarende fysiske aksjoner. Skjemaer og menyer fra den forrige generasjonen er tatt med over til denne generasjon, om enn i en noe mer visuelt tiltalende utforming. Grafiske grensesnitt, direkte manipulerende grensesnitt, objektsorienterte grensesnitt, eller WIMP (Windows, Icons, Mouse, and Pull Down Menus) benyttes som betegnelser for å beskrive denne generasjonen av operatørgrensesnitt. Mer om dette nedenfor.

Den neste generasjonen som vi ser fremveksten av i dag er karakterisert ved nye typer betjeningsorganer og informasjonsgivere, talegjenkjenning og talesyntese, samt utstrakt bruk av tredimensjonal naturtro sanntids grafikk. En kort beskrivelse av denne teknologien er gitt nedenfor.

6.2 Generelt om dialogen

Et viktig aspekt ved dialogen mellom operatøren og maskinen er hvordan den styres. Dvs, hvem det er som har initiativet i dialogen, operatøren eller maskinen.

Når det gjelder presentasjon av data fra prosessen, må disse selvsagt bli oppdatert dynamisk etter hvert som verdiene forandres. Annerledes er det for selve dialogen mellom operatør og maskin. Som en hovedregel kan man si at operatøren bør være den som initierer og styrer kommunikasjonen. I systemer hvor operatørene er godt trent og kjenner systemet godt vil det være en fordel at han styrer dialogen. På den måten vil operatøren føle at han har mer kontroll med systemet og vil ikke føle seg så bundet og styrt av maskinen som når det er den som styrer dialogen og operatøren bare responderer på dens premisser.

Men det finnes unntak. I situasjoner hvor en bestemt sekvens må følges eller hvor situasjonen kan være stressende (f eks fyring av et våpen, nedkjøring av en prosess etter at en feiltilstand har inntruffet, eller lignende) vil det kunne være riktig at maskinen initierer kommunikasjonen, og at operatøren blir ledet gjennom en bestemt sekvens vha spørsmål og svar. I systemer som har en uensartet brukergruppe og hvor brukerne bare benytter systemet sporadisk, vil maskinstyrt

dialog være å foretrekke (f eks bankautomater).

En mellomting mellom brukerstyrt og maskinstyrt dialog vil være å la operatøren ha initiativ og styre dialogen, men la maskinen overvåke dialogen og sikre at alle data blir spesifisert og at sekvenser følges på riktig måte. Maskinen vil her ikke gripe inn med mindre operatøren gjør feil eller glemmer å utføre visse kritiske operasjoner. På denne måten oppnår man fordelene med operatørstyrt dialog samtidig som man forsikrer seg om at den utføres riktig. Maskinkontrollen skjer dermed på en «mykere» og ikke så påtrengende måte.

Et annet viktig spørsmål er på hvilket nivå de tilgjengelige funksjonene eller aksjonene legges. Med andre ord, hvor nær opp til oppgavene som ble identifisert under oppgaveanalysen legges kommandoene som operatøren har tilgjengelig gjennom operatørgrensesnittet. Hvis kommandoene legges tett opp til oppgavene som er identifisert vil grensesnittet være enkelt å lære og raskt å bruke. Det forutsetter imidlertid en grundig og godt gjennomført oppgaveanalyse hvor man har greid å fange opp de forskjellige arbeidssituasjonene. Det andre alternativet er å bryte dialogen ned i små basisfunksjoner (et typisk eksempler er her kommandorepertoaret for et operativsystem). Fordelene ved en slik nedbrytning er at det oppnås et mer fleksibelt grensesnitt som har større mulighet for å fange opp uforutsette arbeidssituasjoner. Hvis man i tillegg inkluderer makroer hvor operatøren selv kan bygge opp nye kommandoer, er det mulig å oppnå et grensesnitt som kan tilpasses mer den individuelle operatøren. Ulempene vil være at det er vanskeligere å ta i bruk og kan være langsommere å bruke. MMSer vil typisk ha grensesnitt hvor kommandoene legges forholdsvis nær opp til operatøroppgavene.

6.3 Dialogtyper

Nedenfor vil vi gå gjennom de mest benyttede dialogtyper vi finner i moderne grensesnitt. Stoffet er delvis hentet fra Foley *et al.* (1990) og Shneiderman (1992).

6.3.1 Menyer

Menyer blir mye benyttet. Både «pop-up»-, rullgardin- og faste menyer. Menyer benyttes når operatøren skal gjøre et valg blant på forhånd fastlagte alternativer. Menyvalgene kan være valg av funksjoner, attributter, parametersettinger, opsjoner, osv.

Hvis det er mange valg bygges menyer opp hierarkisk. Et viktig konstruksjonsspørsmål er da antall valg på hvert nivå kontra antall nivåer i hierarkiet, med andre ord bredde kontra dybde av hierarkiet. Det «magiske» tallet 7±2 er en god ledesnor for hvor mange alternativer det bør være på ett nivå, Miller (1956). Antall nivå i meny-hierarkiet bør ikke være større enn 3-4.

Plassering av alternativene i menyen bør være fast og det mest benyttede alternativet bør plasseres lettest tilgjengelig for operatøren. Forsøk på å tilpasse valgene ut fra systemtilstanden bør gjøres med omhu. Alternativer som ikke er aktuelle i en gitt systemtilstand (kontekst-sensitive menyer) bør gjøres utilgjengelig for operatøren, men likevel være synlig for operatøren. I OSF/Motif , Open System Foundation (1991), kalles slike utilgjengelige valg for

«grey outs».

Hvilke alternativer som er ekskluderende og hvilke som er ikke-ekskluderende bør skilles fra hverandre og det må gå klart frem hvilke som er hvilke. I OSF/Motif benyttes f eks såkalte «radio-buttons» for ekskluderende valg, mens såkalte «check buttons» benyttes for ikke-ekskluderende valg.

Fordelene med bruk av menyer er at de krever liten opplæring og avhjelper operatøren med å måtte huske de forskjellige mulighetene. Ulempen er at menyer kan være langsomme å bruke.

6.3.2 Kommandospråk

Kommandospråk benyttes ikke så mye i MMS, og når det brukes er det gjerne som et supplement til en annen dialogtype.

Viktige problemstillinger ved konstruksjon av kommandospråk er valg av syntaks og kommandonavn (leksikon). Syntaksen bestemmer rekkefølge på angivelse av objekter, funksjoner og argumenter. Denne må være lik for alle deler av språket. Det må også legges stor vekt på å velge kommandonavn som er enkle å huske, og lette å skille fra hverandre. Som nevnt ovenfor må det også legges stor vekt på å finne frem til et balansert kommandosett (generalitet kontra spesialisert).

Kommandospråk kan være svært effektivt for godt øvede operatører, men dette forutsetter mye bruk. Kommandospråk er heller ikke feilrobuste. Se Shneiderman (1992) for mer detaljer rundt konstruksjon av kommandospråk.

Å lage grensesnitt basert på naturlig språk har ikke uventet vist seg å være svært vanskelig. Så langt har det vært gjort forsøk på bruk av naturlige språk dialoger i forbindelse med database applikasjoner.

6.3.3 Tale

Maskinell tolkning av kontinuerlig tale har vist seg å være minst like vanskelig som tolkning av naturlig språk i skriftlig form. Hvis man imidlertid avgrenser ordforrådet til et begrenset kommandospråk er situasjonen en helt annen. Bruk av talegjenkjenning i et operatørgrensesnitt er nyttig når f eks operatøren ikke har hendene fri. På denne måten er det mulig å oppnå en multi-modal dialog og mer parallellitet i informasjonsflyten fra operatør til maskin.

Bruk av talegjenkjenning kan være aktuelt å benytte i f eks fartøysstyring (spesielt fly og helikopter), hvor det muliggjør styring av fartøyet og betjening av f eks navigasjonssystem samtidig. Bruk av talegjenkjenning i systemer som opereres av operatørteam kan være mer problematisk med mindre disse opererer med individuelle mikrofoner. Talegjenkjenning er foreløpig ikke i utstrakt bruk, men vi vil antakeligvis se en økende anvendelse av denne dialogtypen i årene fremover.

6.3.4 Skjemaer

Dette er den skjermbaserte analogien til papirskjemaer som vi kjenner fra dagliglivet (f eks selvangivelsen). I tillegg til felt for innsetting av f eks tekst og tallverdier kan også menyer inngå i skjemaer.

De viktigste konstruksjonsspørsmålene dreier som om rekkefølgen av feltene i skjemaet, forklaringer til de enkelte feltene (ledetekster), sjekk av innsatte verdier, og semantisk tilbakekobling.

Skjemaer (også kalt «dialog panels/boxes», Apple (1987)) benyttes gjerne til oppgaver som f eks å spesifisere parametere, systemoppsett, osv.

6.3.5 Spørsmål-svar dialog

Spørsmål-svar dialog er maskinstyrt og operatøren velger mellom et sett av alternativer for hvert steg i dialogen. Dette gir en sekvensiell dialog med liten frihet for operatøren.

6.3.6 Direkte manipulerende grensesnitt

Direkte manipulerende grensesnitt kalles også grafiske grensesnitt (GUI - Graphical User Interface), objektorienterte grensesnitt eller WIMP (Windows, Icons, Mous, Pull-down menues). Disse betegnelsene blir benyttet om hverandre i litteraturen uten at noen av dem har en klar definisjon.

Det som karakteriserer et direkte manipulerende grensesnitt er at objekter, attributter og relasjoner som det kan gjøres noe med, har en visuell representasjon. Funksjoner utføres på disse representasjonene direkte gjerne vha et indirekte posisjoneringsorgan. På denne måten initieres ikke kommandoer eksplisitt ved f eks menyvalg, men gjøres implisitt ved f eks å flytte et objekts representasjon på skjermen. Altså, å flytte et objekt gjøres ved å flytte dets representasjon og ikke ved å velge en flytt-kommando.

Den visuelle representasjonen kalles gjerne et *ikon*. Et ikon er en billedlig representasjon av et objekt, en funksjon, en egenskap eller et eller annet konsept. Mest vanlig er å gi konkrete *fysiske objekter* en tilsvarende representasjon i grensesnittet, men det er også mulig å gi mer abstrakte fenomener en mer konkret representasjon. Det mest vanlige er å gi fysiske objekter, f eks båter, fly, sensorer og våpen i et militært system eller pumper, rør og ventiler i et prosessanlegg, en mer abstrakt representasjon i grensesnittet.

Dialogen er bygget opp av inkrementelle (og reversible) aksjoner hvor resultatet er umiddelbart synlig. Mapping mellom nivåene som vi beskrev i forrige kapittel er enkel og desto hurtigere man kan få til semantisk tilbakekobling, desto bedre. Dette forutsetter at de objektene som representeres i grensesnittet ikke bare er symboler, men også inneholder mest mulig informasjon om og modeller av deres oppførsel. Aksjoner gjøres ved å imitere tilsvarende fysiske aksjoner eller etablere fysiske analogier når det er snakk om mer abstrakte begreper.

6.3.7 Objektorienterte grensesnitt

Et program som er realisert vha et objektorientert programmeringsspråk blir vanligvis kalt objektorientert. Dette er ikke betydningen vi legger i et objektorientert grensesnitt. Vår betydning er at et grensesnitt er *objektorientert* i motsetning til *funksjonsorientert*. Dette går på hvordan grensesnittet er konstruert og ikke på hvilken type språk det er realisert i. Mens det er funksjonene (de kommandoene eller aksjonene som operatøren kan gjøre) som er det sentrale elementet i et funksjonsorientert grensesnitt, er det de operatørrelevante objektene som er det sentrale i et objektorientert grensesnitt. Dette betyr at syntaks for et objektorientert grensesnitt er:

Objekt+Funksjon

mens det for et funksjonsorientert grensesnitt er:

Funksjon+Objekt

Ved første øyekast synes dette ikke som noen stor forskjell, men dette vil i virkeligheten ha vidtgående konsekvenser for oppbyggingen og organiseringen av hele grensesnittet. I objekt-orienterte grensesnitt velges altså først hvilket objekt som vi ønsker å gjøre noe med, for deretter å velge hva som skal gjøres med objektet. Funksjonene er altså knyttet til objektene og ved å velge et objekt vil mulige funksjoner begrense seg til de funksjonene som er mulige for det objektet og den tilstand som objektet befinner seg i når det velges. I et funksjonsorientert grensesnitt derimot velger vi først hva vi skal gjøre for deretter å velge hvilke(t) objekt(er) vi ønsker å operere på. Påstanden er at et objektorientert grensesnitt er mer i samsvar med hvordan vi mennesker organiserer kunnskap og dermed er enklere å bruke.

Direkte manipulerende objektorienterte grensesnitt representerer «state-of-the-art» operatørgrensesnitt.

6.3.8 Kunstig og augmentert virkelighet

Kunstig virkelighet (også kalt datarom, virtuell virkelighet, syntetisk virkelighet, lissom virkelighet, osv) kan på en måte karakteriseres som direkte manipulasjon i sin ytterste konsekvens. Mens man til nå har snakket om et grensesnitt mellom operatør og maskin, er målsetningen med kunstig virkelighet at operatøren skal være fullstendig omsluttet av maskinen og inne i prosessen. Se figur 6.1.

Indirekte betjeningsorganer er byttet ut med betjeningsorganer som muliggjør en mer direkte bruk av kroppen vår (hender, hode, armer). Eksempler er hodefølgere, datahansker og dataklær.

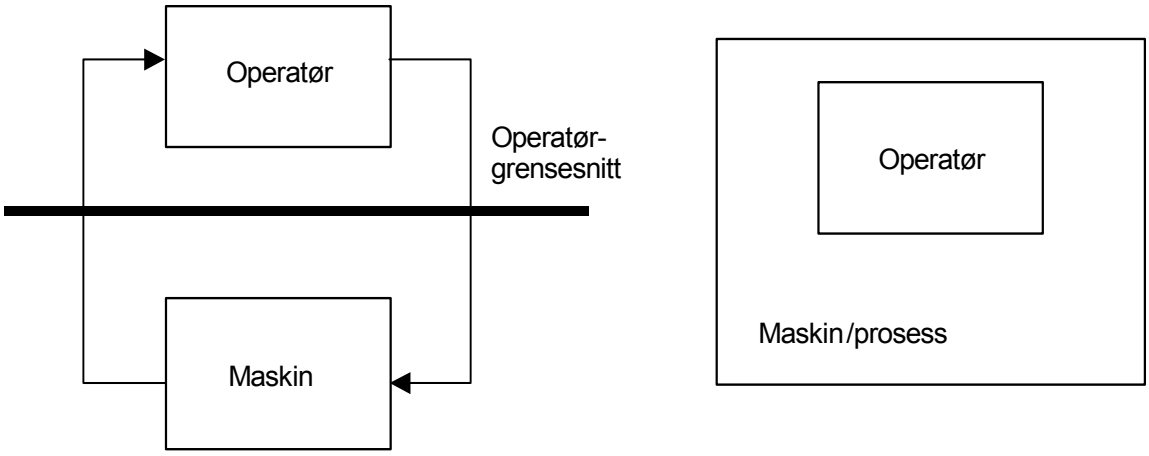
Denne dialogtypen er best egnet i de situasjoner hvor det er ønskelig å gjenskape en fysisk virkelighet, om enn ikke nødvendigvis den delen som er lett tilgjengelig for oss mennesker (f eks pga utstrekning, sikkerhetsrisiko osv). Kunstig virkelighet har en rekke anvendelses-områder for den type systemer som vi omtaler her. Ved riktig bruk kan man tenke seg å oppnå

en større grad av nærhet til prosessen selv om man benytter et kontrollrom. Teknologien har som ønskemål å imitere den fysiske virkeligheten i størst mulig grad og operatøren håndterer objekter som om de var fysiske og de gir tilbakekobling som om de var fysiske (taktil tilbakekobling).

Kunstig virkelighets teknologi har sin opplagte anvendelse hvor det er lett å representere applikasjonen i et tredimensjonalt rom. Denne teknologien vil nok også finne sin anvendelse innenfor områder hvor denne en-til-en relasjonen ikke eksisterer, men da er det nødvendig å utvikle gode analogier. Hvilke som vil passe best, gjenstår å se.

For f eks fartøysstyring vil det selvsagt ikke være ønskelig å frata operatøren mulighet for å benytte sine egne sanser for å observere verden rundt seg, slik som kunstig virkelighet gjør. Det kan imidlertid være svært nyttig å utvide operatørens egne sanser ved f eks å benytte sensorer for å dekke andre deler av det elektromagnetiske spektrum enn det øynene dekker. Dette kalles for *augmentert virkelighet*. Ved å overlagre dette på operatørens utsyn til den virkelige verden, kan man gjøre den enklere for operatøren å forstå og tolke den.

Et annet område for bruk av kunstig virkelighets teknologi er for fjernkontroll av farkoster og roboter ved å bringe operatøren der hvor farkosten/roboten befinner seg (eng: *telepresence*).



Figur 6-1 Kunstig virkelighet kontra tradisjonelt grensesnitt

6.3.9 Oppsummering

Tabell 6.1 gir en kort oppsummering av egenskaper ved forskjellige dialogtyper.

	Direkte Manipulasjon	Meny	Skjema	Kommando-språk	Spørsmål/ svar
Opplæring	lav	medium	lav	høy	lav
Hastighet	medium	medium	høy	høy	lav
Feil	lav	lav	lav	høy	lav
Utvidbarhet	lav	medium	medium	høy	høy
Skrive-ferdighet	ingen	ingen	høy	høy	høy

Tabell 6.1 Egenskaper til forskjellige dialogtyper (etter Foley et al. (1990))

6.4 Betjeningsorganer

Her er det gitt en svært kort beskrivelse av noen av de mest brukte betjeningsorganene. En grundigere beskrivelse er gitt i f eks (Sherr, 1988).

Viktige betjeningsorganer som benyttes er:

1. Alfa-numerisk tastatur:

- QWERTY-tastatur
- Dvorak-tastaur
- Akkord-tastaur

2. Indirekte posisjonseringsorganer:

- Markørtaster
- Mus (2D, 3D)
- Rulleball
- Styrespak
- «Spaceball»

3. Direkte posisjonseringsorganer

- Penn
- Trykkfølsomme skjermer
- Datahanske
- Øyefølger

Vurdering av bruk og egenskaper til betjeningsorganene er oppsummert i tabell 6.2.

Betjeningsorgan	**Betjenings-oppgaver og bruk**	**Ulemper**	**Anbefalt for**	**Uegnet for**	**Kommentar**
Trykkfølsomt skjerm	Velge	Armbelastende, utilsiktet aktivering	Lite brukte funksjoner, ikke krav til stor nøyaktighet	Kontinuerlig bruk, krav til stor nøyaktighet	Behov for armstøtte
Mus	Peke, Velge, Tegne, Dra, Flytte markør	Krever plass	Oppgaver som ikke krever tastatur	Ofte skifte mellom mus og tastatur	Kan integrere funksjonsknapper i musa
Dedikerte funksjonsknapper	Kritiske og mye brukte funksjoner	Plass setter begrensning i antallet	Tilgjengelig hele tiden, viktige funksjoner	Sjelden bruk, ikke-kritiske funksjoner	Merk knappene med funksjonsnavn
Programmerbare funksjonsknapper	Applikasjons-spesifikke funksjoner	Skifte av betydning	Hyppige brukte og viktige funskjoner	Sjelden bruk, ikke-kritiske funksjoner	Plasseres på skjermen
Lyspenn	Flytte markør, Velge, Tegne	Parallakse feil, armbelastende	Lite brukte funksjoner som ikke krever tastatur	Ofte skifte mellom lyspenn og tastatur	Behov for armstøtte, ikke så mye i bruk lenger
Tale	Tallinnmating, Starte predefinerte funksjoner	Gjenkjenning, stegvis bekreftelse på gjenkjenning	Når hendene er opptatt	Støyende omgivelser, stressende situasjoner	Bare egnet for et begrenset vokabular
Styrespak	Følge, Velge, Flytte markør	Mus kan være raskere for valg av tekst	Oppgaver med kontinuerlig styring av en variabel	Ofte skifte mellom styrespak og tastatur	Tenk på venstrehendte
Rulleball	Følge, Velge, Flytte markør	Mus kan være raskere for valg av tekst	Store nøyaktighetskrav, ombord i fartøy	Ofte skifte mellom rulleball og tastatur	Tenk på venstrehendte
Alfanumerisk tastatur	Velge, Mate inn tekst og tall	Lite egnet for grafikk og direkte manipulasjon	Generell innmating	Langsommere å velge ved å skrive enn ved å peke	Benytt standardutlegg
Numerisk tastatur	Mate inn tall	Begrenset til tallinnmating	Rask tallinnmating, beregninger	Sjelden behov for innmating av tall	Tenk på venstrehendte

Tabell 6.2 Egenskaper ved forskjellige betjeningsorganer

6.5 Presentasjon av data

Her vil vi ta opp noen få forhold rundt presentasjon og organisering av data. Mens det i kapittel 5 ble beskrevet en taksonomi for koding av data, skal vi her kort gi noen retningslinjer for noen vanlige kodingsmåter. Se Tufte (1986, 1990, 1997) for et vell av gode retningslinjer og gode/dårlige eksempler på presentasjon av både kvantitative og kvalitative data.

Forsøk har vist at for å oppnå 95% gjenkjennelse kan man ikke benytte mer enn:

-10 farger
-6 flatestørrelser
-6 lengder
-4 intensiteter
-24 vinkler
-15 geometriske figurer

Dette kan gi en pekepinn på hvordan vi bør kode data. Hovedsakelig er måten vi visualiserer og koder data svært avhengig av anvendelsen. Det er likevel mulig å gi endel generelle betraktninger. Noen av disse er beskrevet nedenfor.

Farger
For de fleste av oss er det en hovedregel som gjelder ved bruk av farger: *Unngå den store katastrofen!* («colorjunk»). Vær heller sparsommelig med bruk av farger enn å forsøke å briljere.

Farger egner seg for å inndele data i grupper, angi tilstand, imitere virkeligheten og for å forskjønne presentasjonen. Bruk av farger er mest virkningsfullt ved sparsommelig bruk av sterke, rene farger på små flater og duse, dempede farger som bakgrunn på store flater. Det er altså viktig at bakgrunnen ikke konkurrerer om oppmerksomheten til det som er informasjon i bildet, eller med andre ord signal/støy(bakgrunn)-forholdet må være stort nok. Ved f eks å benytte svake konturlinjer klarer man seg med mindre sterke farger og det kan være et viktig virkemiddel for å slippe å benytte sterke farger.

Ikke bruk mer enn 5-7 farger og pass på ikke å bryte kulturelt betingede konvensjoner for fargebruk. Farger er best egnet til å fremheve data og å vise tilstand/modus. Farger bør i hovedsak benyttes som redundant koding av data.

Hvis MMS benyttes under spesielle lysforhold må man være spesielt påpasselig. Bruk av f eks farge katodestråleskjermer i fartøyer hvor det er viktig å bevare nattsynet til operatøren er spesielt problematisk.

I tillegg til bøkene av Tufte omhandles bruk av farger i f eks Travis (1991) og Durrett (1987).

Numerisk kontra grafisk presentasjon
Bruk grafikk for å gi oversikt, for å få ned informasjonstettheten og for å fremstille kompliserte sammenhenger mellom data. Numeriske verdier egner seg best når det kreves en nøyaktig angivelse/avlesning.

Den viktigste utfordringen fremover når det gjelder å visualisere data, er etter vår mening å utnytte de mulighetene som *sanntids 3D grafikk* gir. Spesielt det å utnytte sanntidsaspektet er viktig. Vår evne til å tolke bildesekvenser med dynamikk vil dermed kunne utnyttes i en langt større grad enn det som har vært mulig opp til nå. Bruk av stereoskopiske bilder vil i endel sammenhenger kunne være nyttig for å forsterke 3D-effekten.

Blinking
Blinking for å påkalle oppmerksomheten til operatøren bør brukes sparsommelig og bare i spesielt viktige situasjoner. Antall nivåer bør bare være to og blinkfrekvensen bør ligge mellom 2-5 Hz og andel av på-tiden bør være mer enn 50%. Bruk av forskjellige blinkfrekvenser er en

ikke spesiell egnet kodingsmåte.

Tekst
Bruk både store og små bokstaver ved tekst som har en viss lengde. Forsøk å forme setninger slik at det benyttes positive utsagn istedenfor negative. Likeledes bør feilmeldinger ikke inneholde beskjed om at man har gjort feil, men heller informasjon om hvordan man kommer seg ut av situasjonen.

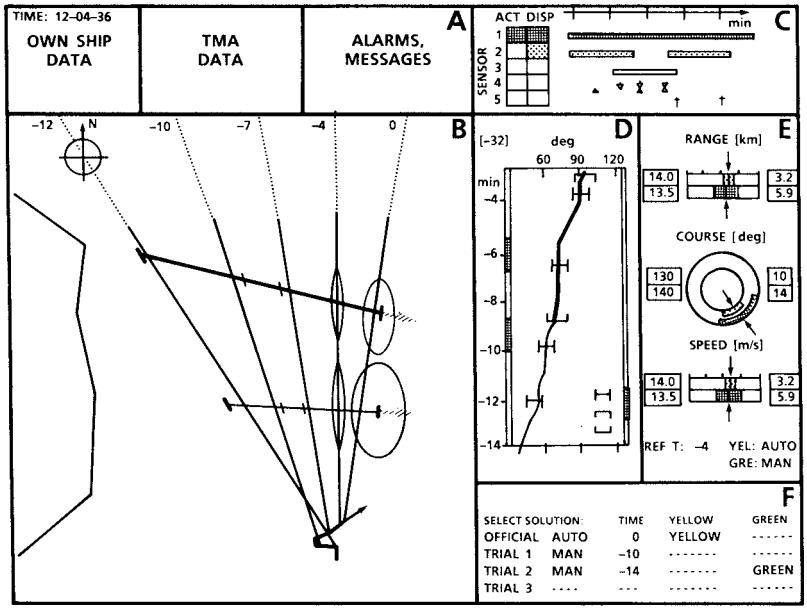
Gruppering og organisering av data
Dagens informasjonsgivere er svært begrensende mhp størrelse og oppløsning, slik at den informasjonsmengden som skal presenteres for operatøren i de langt fleste tilfellene ikke får plass på det arealet som skjermer tilbyr i dag. Informasjonen organiseres derfor gjerne i forskjellige *bilder* og *vinduer*. I store prosessanlegg har det vært vanlig å benytte opptil flere hundre bilder som presenterer prosessen på forskjellig detaljeringsnivå. Man etablerer altså bildehierarkier. Konstruksjonsspørsmål blir igjen, som for menyer, bredde og dybde av et slikt hierarki. Når hierarkiet blir stort, blir det en oppgave i seg selv å navigere i hierarkiet.

Bruk av vinduer er også vanlig å benytte for å strukturere og organisere data. I forhold til kontorautomatiseringssystemer, må man være mer forsiktig med bruk av *overlappende* vinduer, som kan forårsake at kritiske data blir skjult av mindre viktige data. Det er derfor vanlig å benytte *ikke-overlappende* vinduer og legge begrensninger på hvordan operatøren kan organisere vinduene på skjermen.

En måte å forsterke kodingen er å benytte flere vinduer til å presentere de samme dataene (redundant koding), men med forskjellig perspektiv. Det er selvsagt viktig at det er konsistens mellom de forskjellige perspektivene. Ett eksempel på dette er beskrevet i Aas et al. (1989) og er vist i figur 6.2. Dette eksemplet er fra et kampledelsessystem for ubåter hvor f eks peilemålinger fra sonarer er kodet på fire forskjellige måter, som peilinger i et plan, som en tidsserie, som tidsintervall hvor peilemålinger er tilgjengelig og peiling ved et gitt tidspunkt som numerisk verdi. I tillegg kunne man om nødvendig ha angitt peilingen på en grafisk skala.

Innenfor ett vindu eller bilde grupperes data etter viktighet, hyppighet i bruk og funksjon. Viktige data plasseres sentralt i bildet/vinduet og plasseringen bør være fast. Ved bruk av fast plassering kan informasjonstettheten i bildet økes, men bør ikke overskride 50%. Dvs ikke mer enn 50% av skjermbildearealet bør benyttes til reell informasjon. Plassering av data i et vindu/bilde bør også følge vanlige venstre-høyre- og opp-ned-konvensjoner.

Når det brukes linjer eller annen markering for å skille datagrupper, gjelder det samme som for farger. Ikke la disse bli for dominerende, slik at de konkurrerer med selve informasjons-innholdet.



Figur 6-2 Forskjellig perspektiv på data i flere vinduer

6.6 Informasjonsgivere

Så og si all informasjonsoverføring til operatøren foregår i dag visuelt. Bare en svært liten del foregår taktilt og vha lyd. Pr i dag blir lyd brukt svært primitivt selv om det potensielt er muligheter for å overføre kompleks informasjon. Tenk bare på hvilke prosesser i oss som musikk kan frembringe.

Fram til i det siste har raster katodestråleskjermer (eng: CRT, cathode ray tube) vært dominerende. Årsaken til dette er i tillegg til gode egenskaper, at de baserer seg på samme teknologi som TV-produsentene benytter, og er derfor relativt billig å produsere. Det er økende bruk av «flatskjerm»-teknologi som plasmaskjermer og flytende krystall skjermer (LCD - Liquid Crystal Display), men i anvendelser som krever store arealer (opp mot og større enn 30 tommer) og god oppløsning (opp til 2000x2000 bildeelementer) vil CRT-skjermer fremdeles dominere. Mens vi til nå i hovedsak har hatt CRT-skjermer med et aspekt-forhold på 3:4 er bruk av skjermer med aspektforhold på 9:16 i økende bruk. Tabell 6.3 angir de viktigste egenskapene til de viktigste skjermteknologiene.

I tillegg til de tradisjonelle CRT-ene er bruk av *prosjektører* aktuelt i kontrollrom for oversikts-bilder som bør være lett leselig fra flere posisjoner i kontrollrommet.

I kunstig virkelighets anvendelser benyttes to miniatyr hodemonterte CRT-skjermer/LCD-er som hver viser bilder som er forskjøvet i forhold til hverandre for å oppnå et stereoskopisk bilde. I augmentert virkelighets anvendelser benyttes gjennomsiktige visir som bildet projiseres på eller såkalte «Head-Up Display» (HUD). Bildet projiseres i uendelig fokus for at operatøren skal slippe å refokusere øynene for å avlese informasjonen og for at den lettere skal kunne

integreres sammen med det visuelle bildet som operatøren har av den fysiske omgivelsen rundt seg.

Mer informasjon om skjermteknologi finnes f eks i Foley *et al.* (1990), MacDonald og Lowe (1997) og Sherr (1998).

	CRT	LCD	Plasmaskjerm
Effektforbruk	god	meget god	god
Skjermstørrelse	meget god (>30 tommer)	god (21 tommer)	god (21 tommer)
Oppløsning	meget god (2000x2000)	meget god (1600x1200)	god
Dybde og vekt	dårlig	meget god	god
Robusthet	god	meget god	meget god
«Brightness»	meget god	god	meget god
Kontrast	meget god	god	god
Synsfelt	meget god	god	god
Farger	meget god	meget god	god
Relative kostnader	lave	lave	høye

Tabell 6.3 Egenskaper ved forskjellige skjermteknologier (basert på Foley et al. (1990))

6.7 Programvareteknikk for operatørgrensesnitt

Størsteparten av operatørgrensesnittet realiseres vha programvare. I typiske MMS vil andelen av programvare for å realisere operatørgrensesnittet være stor, typisk fra 50% og oppover. Siden dette har blitt en så stor del av den totale programvareutviklingen har det i det siste årene blitt lagt ned mye innsats for å utvikle verktøy som kan effektivisere dette arbeidet. Slike verktøy er også selvsagt viktig for raskt å kunne bygge og forandre prototyper på en kostnadseffektiv måte. På verktøysiden går utviklingen svært fort og nye produkter kommer på markedet daglig. Vi skal derfor ikke forsøke å gi en status over hvor vi står i dag. Den ville være gammel i løpet av kort tid. Vi vil heller komme med noen generelle betraktninger som forhåpentligvis har noe lenger gyldighet.

Ved å ha gjennomgått en konstruksjonsprosess som beskrevet i kapittel 5, vil vi ha en *spesifikasjon* for programvaren til operatørgrensesnittet. Ut fra denne spesifikasjonen må det gjøres en konstruksjon og koding av programvaren. Nå er ikke disse stegene så sekvensielle som vi ønsker å gi inntrykk av her. Dette fordi verktøy som benyttes ved realisering av grensesnittet til en viss grad påvirker konstruksjonen av det. Med andre ord verktøyenes muligheter bestemmer utformingen av operatørgrensesnittet. Med dagens verktøy skulle dette likevel ikke være noen ulempe da de i liten grad setter begrensninger. Første bud er derfor å benytte et eller annet verktøy for å realisere operatørgrensesnittet da det er altfor ressurskrevende å implementere et moderne grensesnitt fra bunnen av.

Programvaren for operatørgrensesnittet bør skilles ut fra den applikasjonsavhengige program-varen. Ideelt sett burde det være mulig å bygge forskjellige grensesnitt for de samme

applikasjonsfunksjonene ved bare å skifte ut programvaren for operatørgrensesnittet. Dette skillet er ikke så klart i virkeligheten, men likevel bør et slikt skille tilstrebes.

Det mest vanlige er å dele operatørgrensesnittprogramvaren inn i tre logiske deler, nemlig:

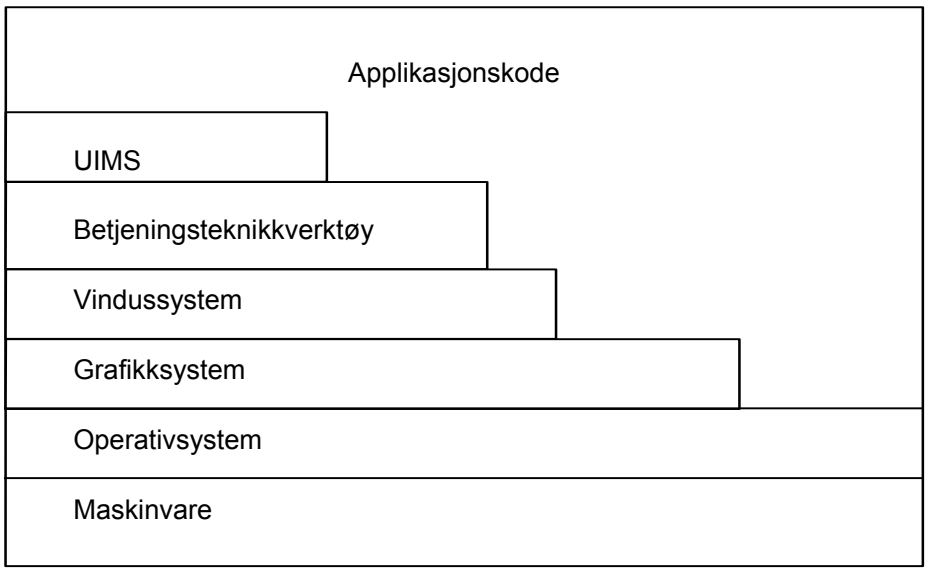
1. Applikasjonsgrensesnitt
2. Dialogkontroll
3. Presentasjonsdel

som hver tar seg av forskjellige sider ved operatørgrensesnittet. Denne modellen for oppdelingen av programvaren ble første gang foreslått under en konferanse i Seeheim og har derfor fått navnet *Seeheim-modellen* og har preget utviklingen senere, Pfaff (1985).

6.7.1 Verktøy

Det finnes en lang rekke forskjellige typer verktøy for å realisere et operatørgrensesnitt. En måte å dele de inn på er:

- Grafikk-biblioteker, f eks OpenGL, Xlib, Java 2D/3D API
- Vindussystemer, f eks X
- Betjeningsteknikk-biblioteker, f eks X/Motif, Swing
- Operatørgrensesnitt administrasjonssystemer, eller såkalte *UIMS - User Interface Management System*



Figur 6-3 Programvareteknisk oppbygging av et operatørgrensesnitt. Basert på Foley et. al. (1990)

I figur 6. 3 er de forskjellige verktøyenes plassering innenfor et programvaresystem vist.

Av de forskjellige verktøytypene er UIMS-ene de mest avanserte og omfattende. Dette er

verktøy som i tillegg til å effektivisere realiseringen i noen grad støtter konstruksjonsprosessen ved f eks å inkludere et egnet språk for å kunne beskrive dialogen i. Disse språkene er gjerne basert på tilstandsmaskin-formalismen. I tillegg inneholder UIMS-ene gjerne et interaktivt tegneprogram for å definere skjermbilder. De objektene som defineres og deres attributter kan enkelt knyttes opp til applikasjonsvariable som f eks representerer prosesstilstander som f eks avstand, fart og kurs til et fartøy. Et bibliotek av programvaremoduler er også gjerne inkludert. Noen av UIMS-ene genererer også programkode direkte fra beskrivelsen av dialogen sammen med den interaktivt definerte presentasjonsdelen.

En trend som vi ser er at UIMS blir integrert med systemteknikk- og programvareteknikk-verktøy. På denne måten letter man overgangen mellom den analytiske og eksperimentelle delen av utviklingsmodellen vår og man oppnår lettere konsistens mellom analysene og prototypene. Med denne utviklingen ser vi veien frem til målet om at «*spesifikasjonsspråket = konstruksjonsspråket = realiseringsspråket*».

7 EVALUERING

> *Hvis vi må måle grundig for å registrere noen forskjell i det hele tatt, arbeider vi kanskje med ting som ikke spiller tilstrekkelig stor rolle.*
> *Nicolas Negroponte: Leve digitalt (1995)*

I dette kapitlet skal vi gjennomgå forhold knyttet til det å evaluere MMS. Forskjellige måter å måle ytelse på ble beskrevet i innledningen til kapittel 3, jamnfør figur 3.1. Ulike typer av ytelsesparametere for operatøroppgaver ble listet opp i kapittel 4.3. Under systemutviklingen kan vi anta visse verdier på ytelsesparametrene eller vi kan, med utgangspunkt i modeller, prediktere ytelsen. Verdiene kan imidlertid ofte være temmelig usikre. Usikkerheten knyttet til enkelte parametere kan reduseres (ikke elimineres) ved å foreta laboratorie-, prototyp- eller feltforsøk med operatører. Det kan også være ønskelig å validere eller tilpasse modeller med målinger fra slike forsøk. I denne sammenheng er det viktig å ta i betraktning at avvik mellom måling og modell også kan skyldes målefeil.

Systembeskrivelser mangler ofte en eksplisitt sammenheng mellom overordnete systemkrav og krav på lavere nivåer for operatøroppgaver/MMK. Validering av overordnete systemkrav i MMS krever derfor at man har et forholdsvis komplett og tilstrekkelig realistisk prøvesystem (f eks mhp tidsrespons). En (abstrakt) modell av systemet vil implisitt inneholde koblinger mellom krav på ulike nivåer. Dersom prototypen kun omfatter visse deler av systemet, kan man derfor eventuelt prøve å inkludere tilgjengelige måleresultater i en simuleringsmodell av totalsystemet. Ofte vil man under prototyping begrense seg til å sammenligne den relative ytelsen til alternative delløsninger og *ikke* fokusere på å validere krav.

Innledningsvis i kapittel 3 understreket vi betydningen av å ha en klar målsetning med modelleringen. Det samme gjelder for prototyping. Gjennomføring av forsøk, inklusive

omfattende programvareutvikling av prototypen, koster mye, og man bør derfor kritisk vurdere sammenhengen mellom utviklingskostnad og nytteverdien av resultatene. Nytteverdien er et mål på muligheten av å integrere resultatene i systemutviklingsprosessen slik at systemet blir bedre (eventuell billigere). *Hva* vil vi vite og *hvorfor*? *Hvem* skal bruke informasjonen og *hvordan* skal den brukes?

Formålet med (hurtig) prototyping kan som nevnt over være å undersøke valg av løsninger på et overordnet nivå. Det er en viss fare for at evalueringen i stedet fokuserer på detaljer i f.eks. skjermbilder og betjening. I St. Dennis *et al.* (1990) ble et større antall prosjekter undersøkt der man benyttet tidlig prototyping. Studien konkluderte med at vurderingene stort sett var relatert til utseende av prototypen isteden for evaluering av funksjonalitet. Detaljvurderinger er selvsagt også nyttige, men de er ikke viktigst i en tidlig fase av MMS utviklingen.

7.1 Gjennomføring av forsøk

Ved utvikling av en prototyp må hensikten med denne avklares. Etter at man har etablert en klar målsetning med evalueringene må man finne ut hvordan forsøk skal gjennomføres, dvs valg av forsøkspersoner, måleteknikk, hvilke data som skal registreres og analysemetode. Man vil ofte benytte en kombinasjon av kvantitative og kvalitative registreringer:

- Logging av data under kjøring
- Observasjons- og intervjuteknikker
- Spørreskjemaer
- Protokollanalyse. Dvs operatøren «høyttenker» og forklarer resonnement bak aksjoner. Både verbal og ikke-verbal oppførsel kan registreres (f eks videofilming m/lyd). Protokollen analyseres i ettertid.

Som vi ser er disse teknikkene svært like de vi listet opp under operatøranalyse i kapittel 4.1.

Scenarier fra en kravanalyse er et godt utgangspunkt for prototypevaluering. Man vil typisk detaljere scenariene og legge inn hendelser for deretter å registrere handlingsmønstre og reaksjonstid. Det kan også være aktuelt å stanse opp i scenariet for å teste situasjonsbevissthet (engelsk: situation awareness).

Valg av forsøkspersoner er svært viktig og informasjon fra operatøranalysen nevnt over vil være nyttig. Forsøkspersonene vil ha ulike erfaringer, oppfatninger og «fordommer» med gamle systemer og/eller utviklingen av det nye systemet. De er ikke bare passive mottakere av instruksjoner fra testpersonell og vil ha, eller utvikle, egne hypoteser angående formålet med forsøket. Forsøkspersoner kan være (sterkt) opptatt av resultatene og f eks prøve å vise at de er skarpe problemløsere (Orne, 1962). En del er kritiske til det å legge for stor vekt på brukeres vurderinger ved prototypevaluering (NATO RSG.12, 1990). Årsaken til dette er bl a mulighet for negativ overføring av eksisterende ferdigheter og kunnskap. Dersom en bruker ikke kan/vil forstå eller ikke kjenner til muligheter med ny teknologi kan dette resultere i en systemløsning som inneholder unødvendig, uhensiktsmessig eller foreldet funksjonalitet. Man bør derfor prøve

å påvirke brukeren (oppdragsgiver) til å velge representanter som er tilstrekkelig åpne for nye løsninger (Beevis *et al*., 1992).

I en forsøkssituasjon er man interessert i å undersøke årsaks-virknings-relasjoner mellom parametrene (målene) X og Y. Man manipulerer (endrer) X og observerer en eventuell endring i Y. X kalles ofte for behandling (eng. treatment). Et forsøk vil typisk kunne klassifiseres som:

- «Within subject comparison»: der man sammenligner Y før og etter manipulering av X for samme forsøksperson (der X f eks representerer en endring i MMS)
- «Between subject comparison»: der man sammenligner Y til to separate grupper hvorav kun en av disse har fått behandling (f eks en gruppe for system uten endring og en for system med endring)

Intern validitet defineres slik i (Neale & Liebert, 1986): «Validity of empirical statements dealing with the question of whether X (as manipulated) causes a change in Y (as measured).» Relasjonen X→Y kan være av typen:

- Tilstrekkelig og nødvendig
- Tilstrekkelig (men ikke nødvendig, f eks X → Y eller Z → Y)
- Nødvendig (men ikke tilstrekkelig, f eks X and Z → Y)
- Statistisk. X kan øke sannsynlighet for at Y skal inntreffe.

I sistnevnte type relasjon undersøkes samvariasjonen mellom X og Y. Merk at man ut fra målt korrelasjon prinsipielt kan ha både X→Y og/eller Y→X.

I et forsøk prøver man å kontrollere alle relevante variable og isolere effekten av X alene, dvs å oppnå intern validitet. Mange typer «trusler» mot intern validitet er beskrevet i (Neale & Liebert, 1986) for de to hovedtypene av forsøk. Noen av disse er for «Within subject comparison»:

- Virkningen av å ha gjennomført tilsvarende test før
- Endringer i måleutstyr eller prosedyre (f eks bytte av testpersonell som observerer)
- Statistisk regresjon (ekstremverdier fra en fordeling har en tendens til å bevege seg mot middelverdien i fordelingen ved neste test)

For «Between subject comparison» nevnes f eks:

- Skjevheter i sammensetningen av de ulike forsøksgruppene
- Ulikheter i frafall mellom gruppene (underveis i forsøket)
- Påvirkninger (diffusjon) mellom gruppene
- Rivalisering mellom gruppene

Generelt vil f eks prestasjonene endres mellom forsøk eller i løpet av et lengre forsøk pga læring. Dersom det ikke er nettopp dette man vil måle, så bør man prøve å holde kontroll med

slike effekter. Problemer som beskrevet over er nok ikke alltid relevante for såpass teknisk betonte forsøkssituasjoner som vi er interessert i, men bør likevel ikke neglisjeres.

I et forsøk kan man skille mellom tre ulike roller eller typer av personer:

- Rolle som forsøksperson (den som skal studeres)
- Observatørrolle (testpersonell ansvarlig for gjennomføring av forsøk)
- Analyserolle (personell ansvarlig for å spesifisere forsøket, analysere data og trekke konklusjoner)

Det å bli observert, i rollen som forsøksperson, kan ofte påvirke resultatene. Et eksempel på dette er den såkalte «Hawthorne-effekten». I en fabrikk (Hawthorne) ønsket man å undersøke virkningen av ulike tiltak som påvirket arbeidsmiljøet og hvordan dette igjen påvirket produktiviteten. Temmelig overraskende fant man ut at alle typer endringer førte til økt produktivitet! En av årsakene til dette var at den spesielle interessen for de arbeiderne som var med i forsøket førte til at disse ble motivert til å jobbe hardere!

En observatør kan (ubevisst) påvirke resultatene av et forsøk på mange ulike måter, f eks ved å endre kroppspråk eller stemmeleie eller ved å feiltolke responsen til forsøkspersonen eller feilregistrere responsen (Neale & Liebert, 1986). Det kan være en stor fordel dersom observatøren ikke er den samme som skal analysere dataene. Ideelt sett burde ikke observatøren engang vite om hvilke hypoteser som skal undersøkes. Når man har nedlagt mye innsats i prototyping for å forbedre et system er det naturlig at man ønsker å få en bekreftelse på at det faktisk er blitt bedre! I en slik situasjon er det naturlig at man ubevisst kan komme til å påvirke forsøk og resultater. Ideelt sett burde ikke de som evaluerer systemet være de samme som utviklet det. Dette prinsippet er innenfor programvareteknikk kjent som IV&V (Independent Verification and Validation).

Et vesentlig problem knyttet til forsøk i en laboratoriesituasjon (prototypsituasjon) er *ekstern validitet*, dvs i hvilken grad konklusjoner kan generaliseres. Er resultatene gyldige for det endelige systemet som settes i drift under virkelige forhold og er resultatene gyldige for brukerpopulasjonen som helhet? Intern validitet er en forutsetning for ekstern validitet. Krav til intern validitet kan imidlertid resultere i såpass spesialiserte og kunstige testsituasjoner at man vanskelig kan generalisere og dermed anvende resultatene i systemutviklingen.

7.2 Måling av arbeidsbelastning

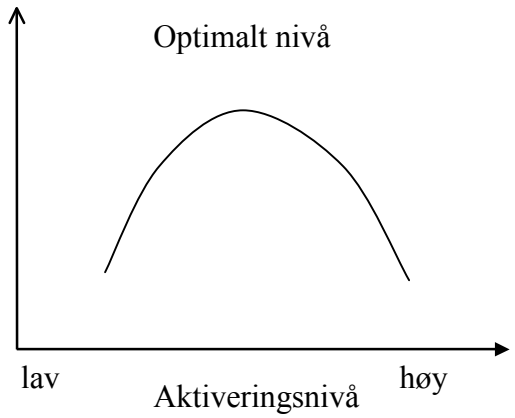
I den grad det er mulig vil man prøve å foreta en systematisk registrering av ytelsesparametre som tidsforbruk, feil, nøyaktighet. Dette er ikke alltid lett dersom systemet er et eksisterende, operativt system. Her skal vi fokusere på måling av operatørens arbeidsbelastning. Generelt vil det være store individuelle variasjoner avhengig av evner, opplæring, erfaring og motivasjon. Den vil være tidsvarierende, situasjonsavhengig og bestemt av konsekvensene relatert til ulike handlinger. Operatørens egenvurdering av arbeidsbelastningen omtales som subjektiv arbeidsbelastning. Denne er ikke nødvendigvis en direkte funksjon av ytelsesparametrene som

nevnt over.

I dagens systemer er man mest opptatt av å måle kognitiv (mental) arbeidsbelastning. Måling av belastninger relatert til persepsjon kan også være viktige, f eks belastning av syn.

Begrepet kognitiv arbeidsbelastning relateres ofte til kapasitet for informasjonsbehandling tilsvarende kanalkapasitet i et kommunikasjonssystem (Essens *et al.*, 1994). Mennesker har begrenset kapasitet, f eks er det begrenset hvor mye informasjon som kan tilbakekalles fra hukommelsen og hvor mye som kan behandles.

Den generelle arbeidsbelastning er relatert til graden av *aktivering* (eng. arousal) hos operatøren. Aktivering påvirkes av motivasjon, fare, konsekvens av mestring/ikke-mestring, mulighet for mestring, trening og personlighet (Wiik, 94). Det er ikke uten videre lett å definere hva som er optimal arbeidsbelastning, men ytelsen til operatøren vil generelt være en funksjon av aktiveringsnivå som illustrert i figur 7.1.



Figur 7-1 Ytelse som funksjon av aktivering. Etter (Neale & Liebert, 1986)

Lav aktivering betyr at operatøren er understimulert (kjeder seg). I en slik situasjon vil han kunne være uoppmerksom. Ved ekstrem aktivering er operatøren overstimulert, han er overaktiv og preget av panikk. Høye nivåer av aktivering vil resultere i uttretting og redusert ytelse. Aktivering er relatert til oppmerksomhetsnivå og en rekke fysiologiske variable. Det siste vil vi komme tilbake til senere.

Det er grovt sett tre ulike typer teknikker for måling av kognitiv arbeidsbelastning (Wilson & Eggemeier, 1994):

1. Oppførsels-/ytelsesbaserte
2. Subjektive

3. Psykofysiologiske

Oppførsel/ytelsesbaserte teknikker måler operatørens evne til å utføre oppgaver. I såkalt *primær oppgavemåling* ser man på evnen til å utføre oppgaven under ulike forhold, f eks bilkjøring under vanskelige vær/føre forhold. Man prøver å identifisere grenser for yteevne, dvs når får man en markant reduksjon eller et sammenbrudd i ytelsen? Når begynner operatøren å gjøre mange eller alvorlige feil? Denne teknikken er ikke alltid sensitiv mhp variasjoner i belastning dersom arbeidsbelastningen er liten/moderat.

I såkalt *sekundær oppgavemåling* prøver man å måle mental reservekapasitet ved å gi tilleggsoppgaver som er enklere å måle, f eks regneoppgaver. Man kan også se på operatørens evne til å monitorere informasjon som naturlig vises i den aktuelle situasjonen (f eks instrumentpanel, skjermbilde osv) i tillegg til primæroppgaven. Denne teknikken antar at ytelsen til sekundæroppgaven avtar når belastningen for primæroppgaven øker. Problemet med denne teknikken er at sekundæroppgaven kan forstyrre utførelsen av primæroppgaven.

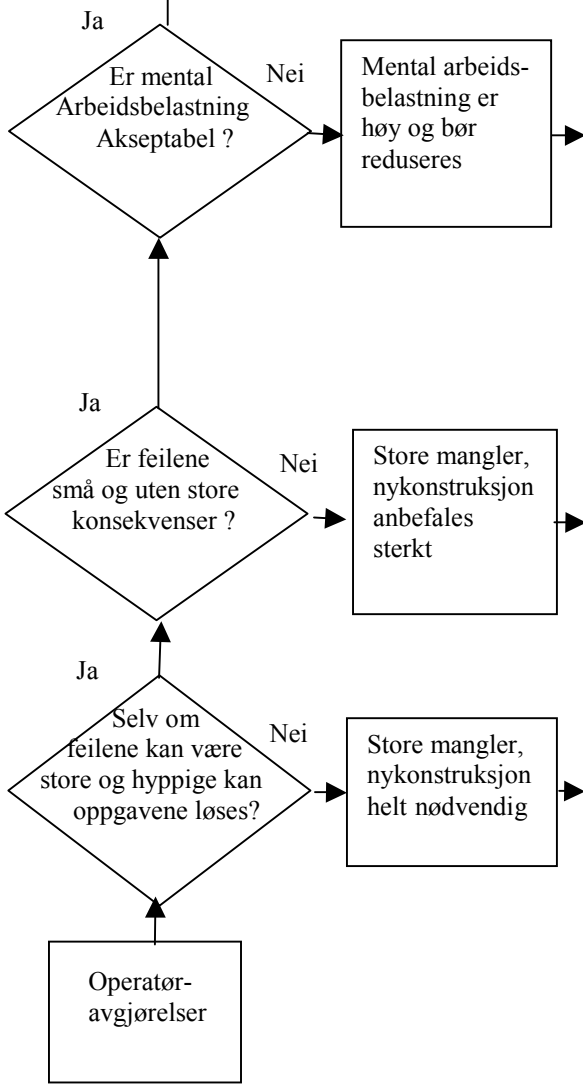
Subjektive måleteknikker består vanligvis i at operatøren besvarer ulike typer spørsmål etter at forsøket er over. Denne teknikken er kanskje den vanligste, bl a fordi den rent praktisk er svært enkel. Dermed er det ikke sagt at det er enkelt å tolke resultatene! Operatøren blir bedt om å gi sin egen vurdering av arbeidsbelastningen og ulike typer av måleskalaer blir brukt. Typiske spørsmål er vanskelighetsgrad, tidspress, egen vurdering av prestasjon, stressnivå og i hvilken grad man behersket oppgaven. Eksempler på teknikker som ofte omtales er (Wilson & Eggemeier, 1994):

- Cooper Harpers (modifiserte) test
- SWAT (Subjective Workload Assesment Technique)
- NASA-TLX (Task Load Index)

Cooper Harpers (modifiserte) test. Den kan brukes til å vurdere arbeidsbelastning og funksjonalitet for enkeltkomponenter eller et komplett system (Wiik & Bråthen, 1994). Den består av et tre av ja og nei spørsmål. Antallet spørsmål er maksimalt 3 etterfulgt av et valg mellom 3 alternativer som karakteriserer vanskelighetsgrad/krav til operatør. Resultatet er et tall i intervallet 1-10 der 1 angir "svært lett" og 10 "umulig". Dersom resultatet er 7-9 anbefales rekonstruksjon og dersom 4-6, en reduksjon av arbeidsbelastningen. Testen er svært enkel og er vist i figur 7.2.

SWAT (Subjective Worklooad Assesment Technique) måler subjektiv arbeidsbelastning som antas å være definert av tre «ortogonale» dimensjoner: tidspress, mental innsats og (psykologisk) stress (Reid & Nygren, 1968). Forsøkspersonen kan velge mellom tre nivåer som verbalt beskriver belastningen som beskrevet i tabell 7.1.

Forsøkspersonene må læres opp i teknikken. Først lages en måleskala for den aktuelle anvendelsen og brukerpopulasjonen. Denne lages ved at forsøkspersonene rangerer alle mulige svarkombinasjoner ($3 \times 3 \times 3 = 27$) som tilordnes en verdi mellom 1-100 iht rangeringen.



Vanskelighetsgrad	Krav til operatør	Måltall
Svært lett, Sterkt ønskelig	Minimal anstrengelse og ønsket prestasjon oppnås lett	1
Lett, ønskelig	Lav mental anstrengelse og ønsket prestasjon oppnås	2
Rimelig bra, Små vanskeligheter	Akseptabel mental anstrengelse er nødvendig for å oppnå akseptabel prestasjon	3
Små, men sjenerende vanskeligheter	Moderat mental anstrengelse er nødvendig	4
Moderate vanskeligheter	Høy mental anstrengelse er nødvendig for å oppnå akseptabel systemprestasjon	5
Uttalte, men tolererbare vanskeligheter	Maksimal mental anstrengelse er nødvendig for å oppnå akseptabel systemprestasjon	6
Uakseptable problemer	Maksimal mental anstrengelse er nødvendig for å bringe feil til et moderat nivå	7
Uakseptable problemer	Maksimal mental anstrengelse er nødvendig for å unngå store eller tallrike feil	8
Uakseptable problemer	Intens mental anstrengelse er nødvendig for å løse oppgaven, men hyppige feil oppstår	9
Umulig	Oppgaven kan ikke gjennomføres pålitelig	10

Figur 7-2 Coopers Harpers modifiserte test

TIME LOAD
1. Often have spare time: interruptions or overlap among activities occur infrequently, if at all. 2. Occasionally have spare time: interruptions or overlap among activities occur frequently. 3. Almost never have spare time: interruptions or overlap among activities are very frequent, or occur all the time.
MENTAL EFFORT LOAD
1. Very little conscious mental effort or concentration required: activity is almost automatic, requiring little or no attention. 2. Moderate conscious mental effort of concentration required: complexity of activity is moderately high due to uncertainty, unpredictability, or unfamiliarity; considerable attention required. 3. Extensive mental effort and concentration are necessary: very complex activity requiring total attention.
PSYCHOLOGICAL STRESS LOAD
1. Little confusion, risk, frustration, or anxiety exists and can be easily accomodated. 2. Moderate stress due to confusion, frustration, or anxiety noticeably adds to workload: sifgnificant compensation is required to maintain adequate performance. 3. High to very intense stress due to confusion, frustration, or anxiety: high to extreme determination and self-control required.

Tabell 7.1 SWAT kategorier (Beevis et al*., 1992)*

Nasa-TLX kan brukes til å få en mer spesifikk vurdering av arbeidsbelastningen enn den Cooper Harpers test gir (Wiik & Bråthen, 1994). Total arbeidsbelastning er summen av vektet verdi langs seks dimensjoner.

Psykofysiologiske teknikker har sin basis i at psykologiske tilstander som aktivering har diverse fysiologiske korrelater:

- hud-temperatur og fuktighet (svetting)
- puls og variasjoner i denne
- blodtrykk
- kroppsvæsker (f eks stresshormoner i blod)
- øyebevegelser/fokus og samsyn

Fysiologiske målinger er i prinsippet objektive og kan registreres kontinuerlig. Den teknologiske utviklingen har gjort utstyret vesentlig mindre og enklere å bruke. Rent psykologisk kan nok

likevel det å være koblet opp med alle mulige slags elektroder bidra til forsøkspersonens generelle stressnivå og dermed påvirke de resultatene man får (Neale & Liebert, 1986).

Spesielle hendelser i løpet av forsøket, som f eks feilhandlinger, kan korreleres med fysiologiske målinger og dermed bidra til å forklare oppførselen. Var operatøren overbelastet (høy aktivering), var han for trett eller var han distrahert?

Konklusjon. Fysiologiske målinger sammen med subjektive data gir best mulig grunnlag for å vurdere ytelse og arbeidsbelastning. Man får imidlertid ikke noen særlig dyptgående forklaring på hvorfor ytelseskapasiteter blir overskredet. Man fanger altså ikke opp de underliggende årsakene til overbelastning av f eks hukommelse og informasjonsbehandling. Essens *et al. (1994)* argumenterer derfor for at man i tillegg må foreta en kognitiv oppgaveanalyse der man prøver å identifisere og analysere begrensningene i den kognitive oppførsel slik som vi beskrev i kapittel 4.3.

7.3 Statistisk analyse av måledata

Alle seriøse forsøk benytter seg av statistikk for å beskrive resultater og gjøre inferens om disse. Introduksjoner finnes f eks i (Neale & Liebert, 1986, Wallø & Høyland, 1981 og Høyland, 1976).

F eks kan undersøkelse av ytelse for ulike typer grensesnitt baseres på et såkalt *randomisert blokk eksperiment* og *variansanalyse* (Skare, 1990). Variansanalyse ser på hvordan den totale variansen i et datamateriale kan deles inn i ulike komponenter. I et randomisert blokk eksperiment sammenlignes effekten av ulike behandlinger (f eks grensesnitt) på en ytelsesparameter, der de eksperimentelle enhetene er gruppert i homogene grupper som kalles blokker. Variansanalysen brukes til å undersøke hvorvidt det er en signifikant forskjell mellom middelverdien for de enkelte grensesnittene og hvorvidt det er en signifikant forskjell mellom testpersonene for de enkelte grensesnittene.

Her skal vi kun gjennomgå et lite eksempel på bruk av hypotesetesting for å illustrere visse begrep og kommentere nytteverdien av statistiske utsagn i en systemutviklingsprosess.
La f_1 representere en realisering av en funksjon i et system som er allokert til en operatør. Basert på en stor mengde med data fra ulike operatører med lang erfaring er tidsforbruket beregnet til gjennomsnittlig å være 30 sekunder. Spredningen i målingene har et standardavvik på 5 sekunder.

Anta nå at man ønsker å redusere tidsforbruket og at man har prototypet en alternativ realisering av funksjonen, f_2. Man gjennomfører en måling med n antall forsøkspersoner og beregner en gjennomsnittsverdi lik m. Hvor liten må m være før vi konkluderer med at f_2 er bedre enn f_1? Pga usikkerheten i målingene bør man selvsagt kreve at m er noe mindre enn 30 sekunder, men *hvor* mye? I utgangspunktet er man skeptisk til påstanden om at f_2 er bedre og man krever solid dokumentasjon før man er villig til å endre systemet til f_2. Vi skal nå gi et eksempel på en enkel statistisk analyse av dette forsøket.

I den statistiske modellen antas tidsforbruket å være en normalfordelt variabel med ukjent middelverdi (=m) og standardavvik lik 5 sekunder som før. Vi antar følgende hypotesetest-situasjon:

H_0 (nullhypotese): m $\geq$ 30 s (dvs f_1 best)
H_a (alternativ hypotese): m < 30 s (dvs f_2 best)

Vi ønsker å ha kontroll med sannsynligheten for feilaktig å forkaste H_0, dvs forkaste en sann H_0. Maksimal akseptabel størrelse på denne sannsynligheten kalles for *signifikansnivået, α, til testen*. Vi velger å sette denne til 0,05.

$$\alpha(k) = P[ForkasteH_0 \mid H_0 sann] = \int_{-\infty}^{k} f(m)dm = 0,05 \tag{7.1}$$

Her er *f(m)* fordelingen til middelverdien av målingene (normalfordelt). Ut fra denne ligningen kan man finne den såkalte *kritiske verdien* (k) for middelverdien som avgjør om man skal forkaste eller godta H_0. Dersom m < k vil vi godta H_a. For vårt forsøk er denne gitt ved:

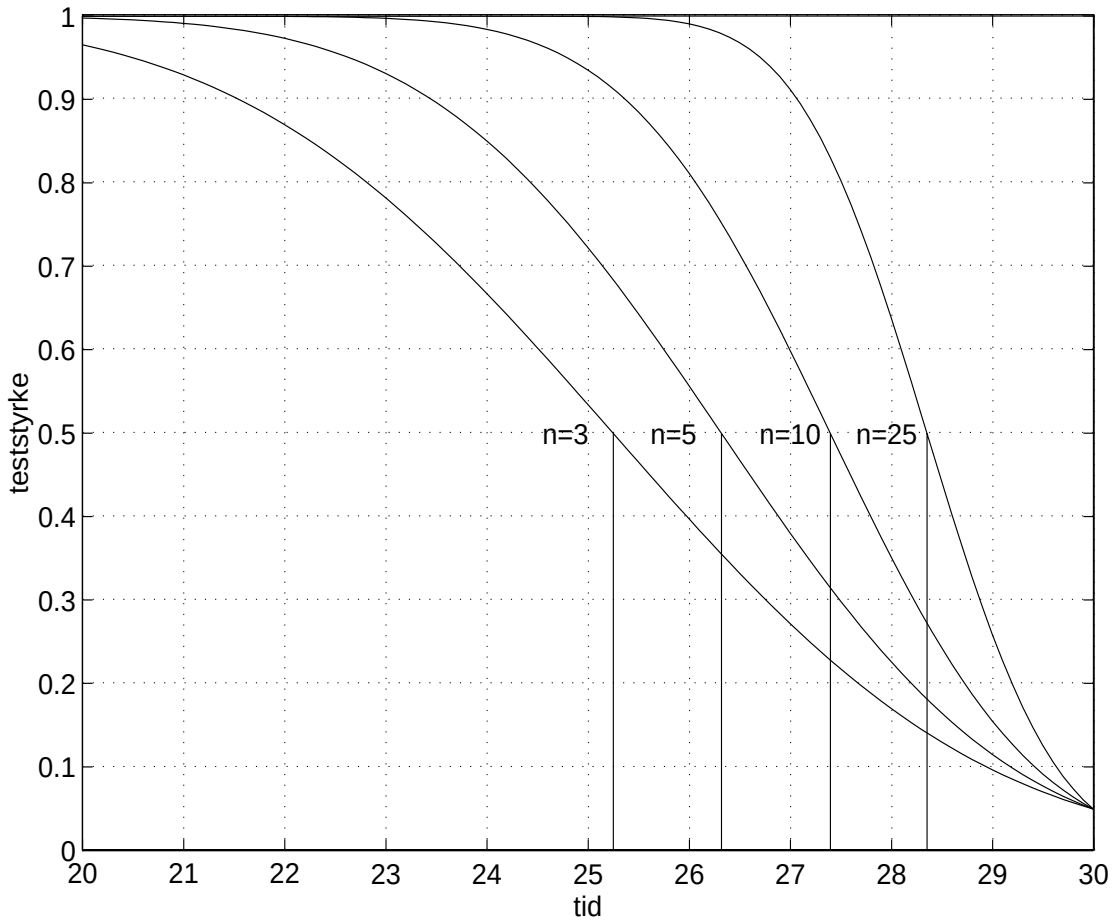
$$k = 30 - \frac{\sigma}{\sqrt{n}}u_{0,05} = 30 - \frac{5}{\sqrt{n}} \times 1,65 \tag{7.2}$$

der σ er standardavviket og $u_{0,05}$ er 5 %-kvantilen i standard normalfordelingen. Sannsynligheten for å forkaste H_0 (dvs godta H_a) når m < 30 kalles for *teststyrke* (i alternativet m). Gitt verdien på signifikansnivået, så ønsker man en størst mulig teststyrke. Sannsynligheten for å forkaste H_0 som funksjon av m kalles *styrkefunksjonen for testen* og er for forsøket vårt gitt ved:

$$\beta(m) = P[ForkasteH_0 \mid H_a sann] = \Phi\left(\frac{k-m}{\sigma}\sqrt{n}\right) = \Phi\left(\frac{30-m}{\sigma}\sqrt{n} - u_{0,05}\right) \tag{7.3}$$

der Φ er den kumulative fordelingsfunksjonen til standard normalfordelingen. Styrkefunksjonen er vist i figur 7.3 for antall forsøkspersoner, n. Vi ser at vi kan øke teststyrken ved å øke n.

Kritiske verdier for ulike valg av n er vist som vertikale linjer for en konstant teststyrke på 0,5. Når vi bruker tre forsøkspersoner (n=3) er $k \approx 25$ tilstrekkelig til å konkludere med at f_2 faktisk er bedre, dvs at forskjellen er *statistisk signifikant*. Dersom n økes til 25 er $k \approx 28,5$.



Tabell 7.2 Teststyrke som funksjon av n. Kritisk verdi angitt som vertikale linjer

Beslutningssituasjonen som er skissert ovenfor (f_1 eller f_2) er usymmetrisk, dvs vi er mer opptatt av *ikke* å forkaste H_0 når den er sann enn det å godta H_a selv om denne faktisk skulle være sann. Dette skyldes at det har større konsekvenser å gjøre den ene typen feil enn den andre. I vårt eksempel vil systemet måtte endres og dette fører bl a til utviklings-, opplærings- og driftskostnader.

I Høyland & Wallø (1981) uttrykkes dette slik: «Legg merke til at det er forhold utenfor faget statistikk som avgjør om en beslutningssituasjon er usymmetrisk eller ikke, dvs om situasjonen egner seg for behandling med en hypotesetestingsmodell».

Anta at vi estimerte m = 28. Dersom n = 25 vil vi iht analysen over kunne godta dette som en signifikant forskjell. Ved å øke n tilstrekkelig i likning (7.2) kan vi oppnå å kalle enhver verdi av m < 30 for signifikant. Slike marginale forbedringer har imidlertid ingen praktisk verdi. Ved utvikling av MMS er det derfor mer relevant å snakke om «praktisk signifikans» som innbefatter statistisk signifikans *og* at forskjellen er stor nok til å ha en effekt på systemytelsen. Dersom f eks en reduksjon til 25 vurderes som interessant, da vil 5-10 forsøkspersoner gi god teststyrke. Dersom man krever en reduksjon til f eks 20 er det tilstrekkelig med noen få forsøkspersoner (rent statistisk).

Fra likning (7.1) ser vi at dersom standardavviket reduseres med en faktor 2 kan vi redusere antallet forsøkspersoner med $2^2 = 4$. Forsøk av typen «Within subject comparison» reduserer virkningen av individuelle forskjeller og dermed dette standardavviket. Dermed reduseres også behovet for antall forsøkspersoner. Ifølge (Wiik & Bråthen, 1994) er 6-10 forsøkspersoner et fornuftig valg som gir brukbar teststyrke i de fleste tilfeller.

Anta at man i eksempelet over er i stand til å anslå økonomisk fordel som funksjon av ytelsesforbedring (dvs m) og at man kan anslå kostnadene ved å velge f_2. Forventet nytteverdi av f_2 kan da beregnes som funksjon av m. Dersom forventet verdi er positiv kan man velge å anbefale f_2.

8 OPPSUMMERING

Fagområdet menneske-maskin-systemer er svært omfattende og personer som regner seg som del av dette fagmiljøt vil ha varierende kunnskap, erfaring og faglige interesser. I dette kompendiet har vi først og fremst prøvd å gi en *oversikt* over det å utvikle menneske-maskin-systemer. Vår vinkling har vært systemteknisk med vektlegging av utviklingsmetode og hvordan man beskriver og analyser samspillet mellom operatører og tekniske delsystemer. Teknisk realisering beskrives kun overfladisk for å gi en viss fullstendighet i framstillingen. Selv om mye av det vi beskriver er gyldig for alle typer menneske-maskin-systemer, har vi fokusert på *reaktive og dynamiske* menneske-maskin-systemer der de "fysiske" prosessene som skal styres spiller en avgjørende rolle. I lys av fagfeltets mangfoldighet er det klart at mye annet stoff kunne vært inkludert. Det er imidlertid vårt håp at leseren allikevel har opparbeidet seg en representativ oversikt over utviklingsmetoder og teknikker som er relevante for analyse og konstruksjon av den typen menneske-maskin-systemer vi har fokusert på. Stoffmengden tillater ikke en inngående beskrivelse eller eksemplifisering av alle teknikker. Mer detaljert forståelse krever studier av oppgitte referanser og egenhendig erfaring med å anvende teknikkene.

På tross av omfattende forskingsresultater og teori innen fagområdet er det vår oppfatning at disse ofte er vanskelig å anvende i en konkret utviklingssituasjon. En del av forklaringen er selvsagt manglende kunnskap hos utviklerne. I norske utviklingsmiljøer finner man sjelden egne MMK-grupper med denne type spesialkompetanse. Denne "løsningen" er heller ikke uten problemer da slike grupper ofte ikke kommer inn tidlig nok i utviklingen og fordi de ikke blir tett nok integrert med det øvrige utviklingsteamet. Tettere og tidligere integrasjon av denne type spesialkompetanse er nødvendig. Kanskje like viktig er det at prosjektledere og system-ingeniører har oversikt over relevante metoder og teknikker. I en reell utviklingssituasjon vil tid, personellressurser, kompetanse og kostnadsrammer avgjøre hvorvidt de ulike aktivitetene i utviklingsprosessen kan gjennomføres. Vårt poeng er at man i alle fall bør vurdere aktiviteter, metoder og teknikker *systematisk* og, for seg selv i det minste; begrunne anvendelse/ikke anvendelse. Typisk vil man prioritere innsats på de delene av systemet som man mener har størst risiko og hvor man tror analyse kan gi en signifikant reduksjon av denne.

La oss kort gi noen eksempler på hva vi mener med en "systematisk" tilnærming til utvikling av større menneske-maskin-systemer. Man bør da bl a vurdere å gjennomføre følgende aktiviteter:

- Avgrense hva som omfattes av systemet, dvs definere systemgrensen og grensesnitt til omgivelsene. Det å inkludere operatører, menneske-menneske-kommunikasjon og funksjoner som utføres manuelt innenfor grensen er egentlig en selvfølgelighet når et menneske-maskin-system skal analyseres, men utelates ofte pga teknologisk fokusering. Ved å behandle tekniske systemer og operatørteam/enkeltoperatører på lik linje sikres en helhetlig og integrert analyse.
- Gjennomføre scenarioanalyse for å sikre at man har et representativt bilde av hva målsetningen er med systemet. Scenarier er et nødvendig utgangspunkt for simuleringer, prototyptester og akseptansetester av det endelige systemet.
- Gjennomføre operatøranalyse for å skaffe seg en oversikt over hvem de egentlige brukerne er (ikke alltid lik oppdragsgiverne!) og hva som karakteriserer disse.
- Analysere eksisterende systemer dersom slike finnes og dersom de har relevans for systemet som skal utvikles. Hva er bra og hva er dårlig? (sett fra brukernes synsvinkel og basert på egne "objektive" analyser av problemene). Det er ingen vits i å gjenta gamle feil!
- Gjennomføre funksjonsanalyse der man beskriver systemets oppførsel systematisk. I hvilken grad man vektlegger informasjonsflyt kontra kontrollflyt er avhengig av systemets natur. For den type systemer vi fokuserer på er begge nødvendige. Funksjonsanalysen er et nødvendig grunnlag for å gjennomføre simuleringsstudier og prototyputvikling. Den er også nødvendig for å kunne lage avansert operatørstøtte.
- Gjennomføre en systematisk allokering av funksjoner til menneske og maskin hvor man vurderer relevante kriterier og fordeler/ulemper til de ulike allokeringene.
- Modellere menneske-maskin-systemet. Alternative allokeringer kan i visse tilfeller analyseres vha simuleringsstudier der man f eks modellerer tidsforbruk, nøyaktighet, sannsynlighet for feil eller operatørenes arbeidsbelastning. Modelleringen er viktig fordi det gir innsikt og bør absolutt ikke kun bedømmes utifra dens evne til å gi kvantitative prediksjoner av systemytelse.
- Gjennomgå systematisk de grensesnitt som allokeringene impliserer, inklusive menneske-menneske-grensesnitt. Hvilken ny funksjonalitet kreves pga de nye grensesnitt og hvordan påvirkes f eks arbeidsbelastningen?
- Analysere systematisk kritiske operatøroppgaver der man fokuserer på informasjonsbehov og operatørrespons (hva skal han gjøre, hvor fort og hvordan skal han kommunisere resultatene!) Hvordan samvirker de ulike oppgavene som er tildelt en operatør? Utgjør de et naturlig hele eller bør man revurdere oppgavefordelingen og rollene til de ulike operatørene som inngår i systemet?
- Gjennomføre en tidlig eksperimentell prototyping av systemfunksjonalitet der man er usikker på løsningen og hvor denne i vesentlig grad vil kunne påvirke systemets effektivitet. Et hovedproblem er avgrensning og realisme i prototypen. Bruk mye tid på å analysere målsetning og gjennomførbarheten av aktiviteten. Er den kost-effektiv?
- Evaluere systematisk prototypen hvor man innhenter objektive og mer subjektive data og hvor man også *kritisk* tar hensyn til subjektivitet (vurderinger, holdninger) hos forsøkspersonene, observatør/testpersonell og konstruktører av systemet/prototypen. Forsøk med mennesker er komplisert og krever spesiell kompetanse. Krav til intern validitet

("vitenskaplig gyldighet") vil alltid komme i en viss konflikt til ekstern validitet ("generaliserbarheten av resultatene til den virkelige verden og det virkelige systemet som skal lages").

Framtidig utvikling av den typen systemer vi diskuterer vil føre til forandringer i hvordan arbeidet fordeles mellom menneske og maskin. Selv om hvert delsystem oppnår sin (begrensete) målsetning kreves det systemintegrasjon (målrettet koordinering) før systemet som helhet kan oppnå hovedmålsetningen. Mange delsystemer vil idag operere med stor grad av uavhengighet, dvs være løst integrert. Nødvendig koordinering utføres da av operatørene i systemet. Tendensen er økt systemintegrasjon vha programvare og redusert behov for operatørinteraksjon. Automatisering stiller store krav til systemintegrasjon. Kravene kan fort bli for ambisiøse; er virkelig systemet istand til å oppdage alle mulige typer problemer for deretter å reagere tilfredstillende *uten* den fleksibiliteten som en operatør tett i kontrollsløyfa utgjør?

En "kooperativ" løsning med både automatisering (ofte kalt operatørstøtte) og operatør er heller ikke uten problemer, f eks hvor realistisk er forventningene om at en overvåken operatør kan gripe inn i sjeldne situasjoner der automatikken feiler? Kanskje må man koble operatøren tettere til prosessen og la være å automatisere visse funksjoner, og dermed akseptere noe lavere ytelse.

Forsøk på å redusere antall operatører vha automatisering er ikke alltid vellykket. F eks kan man tenke seg i et overvåkningsystem å fjerne distribuerte sensoroperatører og i stedet overføre deres oppgaver til et mindre antall operatører i et sentralisert operasjonsrom. Dersom systemintegrasjonen er for dårlig vil man ikke godt nok kunne understøtte fjernkontroll av sensorene og man kan bli tvunget til å gjeninnføre de distribuerte operatørene *i tillegg* til de nye man da har innført!

Holdningen til automatisering er i stor grad at: det som lar seg automatisere (dvs der hvor systemets oppførsel og styringen av denne lar seg formalisere), det automatiserer vi. I dette kompendiet har vi både understreket betydningen av systematisk å beskrive systemoppførselen og å avklare automatiseringsgrad hvor man har et tilstrekkelig nyansert holdning til automatiseringsproblematikk.

Vi mener at det som i størst grad kan bidra til utvikling av bedre MMS er en systemteknisk angrepsmåte som fokuserer på hovedspørsmålene (f eks nødvendig informasjonsflyt mellom delsystem, antall operatører og automatiseringsgrad /arbeidsfordeling).

Vi tror at man framover vil fokusere mer på metoder og konstruksjonspraksis som lar seg *anvende* i dagens komplekse og programvareintensive systemer. I dag er det et for stort gap mellom teori og det man ser anvendt i praksis. Systemutviklere på ulike nivå trenger både mer kunnskap og positiv brukserfaring, dvs "bevis" for nytteverdien av å anvende det som omtales i kompendiet.

Betydningen av reelt samarbeid mellom tverrfaglige miljøer, f eks mellom ingeniører og industripsykologer, vil trolig bli vektlagt sterkere i framtiden. Et effektivt samarbeid krever at

deltakerne har tilstrekkelig bred forståelse av problemstillinger knyttet til utvikling av MMS.

Litteratur

Aas, J. H., Bråthen, K., Nordø, E. and Ørpen, O. Ø., *Man-machine interface in a submarine command and weapon control system: Features and design experience*, In J. Ranta, Analysis, Design and Evaluation of Man-Machine Systems 1988, IFAC Proceedings Series, London, Pergamon Press, 1989.

Alford, M. W. (1985). *A graph model based approach to specification*. In M. Paul and H. J. Siegert (Ed.), Distributed systems: Methods and tools for specification, Springer-Verlag.

Anderson, J. R., *Cognitive Psychology and its Implications,* New York, NY: W H Freeman and Company, 1990, ISBN 0-7167-2085-X

Anderson, J. R., *The Architecture of Cognition*, Cambridge, MA, Harvard University Press, 1983

Andriole, S. J., *Storyboard Prototyping. A New Approach to User Requirements Analysis*, Wellesley, Massachusetts: QED Information Sciences, Inc., 1989. ISBN 0-89435-246-6

Apple® (1987): *Human interface guidelines*: *The Apple desktop interface*, Adision-Wesley, Reading, Massachusetts

Ascent Logic Corp. (1991a). *RDD-100, Requirements Driven Design. User's Guide*. Release 3.0, August, 1991.

Ascent Logic Corp. (1991b). *RDD-100, Requirements Driven Design. Dynamic Verification Facility Manual*. Release 3.0, August 1991.

Baecker, R. M., & W. A. S. Buxton (redaktører), *Readings in Human-Computer Interaction. A Multidisciplinary Approach*, Los Altos, California: Morgan Kaufman Publishers, Inc., 1987. ISBN 0-934613-24-9

Balchen J G *Reguleringsteknikk,* bind 1, Tapir.

Baron, S. and Kleinman, D. L., *The Human as an Optimal Controller and Information Processor*, IEEE Transaction on Man-Machine Systems, MSS-10, 10, p. 9-17, 1969

Baron, S. and Levinson, W. H., *A display analysis with the optimal control model of the human operator*, Human Factors, vol 19, no 5, pp. 437-457, 1977

Baron, S. Kleinman, D. L. and Levinson, W. H., *An optimal control model of human response. Part II: Prediction of human performance in a complex task*, Automatica, Vol. 6, pp. 371-383, 19, 1970

Baron, S., Zacharias, G., Mulidharan, R. and Lancraft, R., *A model for analyzing flight crew procedures in approach to landing*, Proceedings of Eight IFAC World Congress, Tokyo, Japan, August, 1981

Beevis D et al (1992): *Analysis Techniques for Man-Machine Systems Design*, AC/243 (Panel 8) TR/7. Volume 1 and 2. Brussels: NATO Defence Research Section.

Berliner, D. C., Angell, D. and Shearer, J. W., *Behaviors, Measures, and Instruments for Performance Evaluation in Simulated Environments*, In Proceedings of the Workshop on the Quantification of Human Performance, Albuquerque, New Mexico, 1964

Birtwistle, G. M., Dahl, O-J., Myhrhaug, B., Nygaard, K., *SIMULA BEGIN*, Lund, Sverige, Studentlitteratur, 1973

Bisantz A. M., Vicente K. J. *Making the abstraction hierarchy concrete*. Int. J. Human-Computer Studies (1994) 40, 83-117.

Blanchard, B. S. & Fabrycky, W.J. (1981). *Systems Engineering and Analysis*, Prentice-Hall International Series in Industrial and Systems Engineering, Englewood Cliffs, New Jersey: Prentice-Hall, 1981, ISBN 0-13-881631-X.

Booch G., Rumbaugh J., Jacobson I., (1999). The Unified Modeling Language User Guide. Addison Wesley. ISBN 0-201-57168-4.

Boff, K.R., & Lloyd Kaufman & James P. Thomas (redaktører), *Handbook of Perception and Human Performance*, vol I og II, New York , N.Y.: John Wiley & Sons, Inc., 1986. ISBN 1-82956-0

Boff, K.R., Lincoln J.E. (1986). *Markov model for eye transitions during display monitoring*. Engineering Data Compendium. Human Perception and Performance. vol II.

Boehm, B. W. (1988). A Spiral Model of Software Development and Enhancement, IEEE Computer 21(5), 61-72.

Booher, H. R., (redaktør): *MANPRINT. An Approach to Systems Integration*, New York, N.Y.: Van Nostrand Reinhold, 1990, ISBN 0-442-00383-8

Booth, P., *An Introduction to Human-Computer Interaction*, Hillsdale, New Jersey: Lawrence Erlbaum Associates, Publishers, 1989. ISBN 0-86377-123-8

Bost, R., & Oberman, F. (1994). *Improving function allocation for integrated systems design*, In D. Beevis, P. Essens and H. Schuffel (Eds.), Improving Function Allocation for Integrated Systems Design, Wright-Patterson Air Force Base, OH, CSERIAC State-of-the-Art Report, 1996, pp. 29-43.

Brainbridge, L., *Ironies of Automation*, Automatica, Vol. 19, No. 6, pp. 775-779, 1983

Breda, L. van, *Analysis of a Procedure for Multiple Operator Task Performance on Submarines Using a Network Model*, In G R McMillan et al. (Eds.), Applications of Human Performance Models to System Design, New York, NY, Plenum Press, 1989, 231-241

Brooks F.P. Jr. (1987). *No silver bullit. Essence and accidents of software engineering*. IEEE Computer, april 87.

Brown, C.M., , *Human-Computer Interface Design Guidelines*, Norwood, New Jersey: Ablex Publishing Corporation, 1988. ISBN 0-89391-322-4

Brown, J.R., & Steve Cunnigham, S., *Programming the User Interface. Principles and Examples*, New York, NY: John Wiley and Sons, 1989. ISBN 0-471-63843-9

Brubaker, D., I. and Sheerer, C., *Fuzzy-logic system solves control problem*, Electronic Design News, June 18, 1992.

Bræk, R. og Emstad, P.J., *Telesystemering. Språk og metoder for systemarbeid innen telematikk*, Trondheim, Tapir, 2. utgave, 1986

Bræk, R., Haugen, Ø. (1993). *Engineering real time systems. An object-oriented methodology using SDL*. London: Prentice-Hall.

Bræk, R., Hygen J, Høysæter S, Johansen T, Scott P (1985): *Håndbok i systemarbeid*, TAPIR, Trondheim

Bråthen, K., *Noen utviklingstrekk innen brosystemer for hurtigbåter*, Nordisk Navigasjonsforum, 4/94, pp. 26-29, 1994

Bråthen, K., Nordø, E. and Veum, K., *An integrated framework for task analysis and systems engineering: approach, example and experience*, International Journal of Human Computer Studies, vol 41, pp. 509-526, 1994

Campbell, G. U. and Essens, P.J. M. D., *Function allocation in information systems*, In D. Beevis, P. Essens and H. Schuffel (Eds.), Improving Function Allocation for Integrated Systems Design, Wright-Patterson Air Force Base, OH, CSERIAC State-of-the-Art Report, pp. 121-136, 1996

Card, S. K., Thomas P. Moran og Allen Newell, *The Pshychology of Human-Computer Interaction*, Hillsdale, New Jersey: Lawrence Erlbaum Associates, Publishers, 1983. ISBN 0-89859-243-7, ISBN 0-89859-859-1 (paperback)

Cassandras C. G., *Discrete Event Systems, Modeling and Performace Analysis*, Boston, Massachusetts, Asken Associates Incorporated Publishers, 1993

Chalmers, B., *Work Domain Modeling to Support Shipboard Command and Control*, In, Proceedings of the 6th International Command and Control Research and Technology Symposium, Naval Academy, Annapolis, MD, USA.

Chine, W.P. and Evans S S. (1989). *Use of crew performance models and the development of complementary matematical techniques to promote efficient use of analytic resources*. In: G.R. McMillan, D. Beevis, E. Salas, M.H. Strub, R. Sutton and L. Breda (Eds.), Application of human performance models to system design, pp. 285-331. New York: Plenum Press. Defense Research Series, Volume 2.

Chubb, G. P. Laughery, K R. Jr. and Pritsker, A. A. B., *Simulating Manned Systems*, In G. Salvendy (Ed.) Handbook of Human Factors, New York, NY, John Wiley & Sons, 1987, pp. 1298-1327

Chubb, G. P., *SAINT, a digital simulation language for the study of manned systems*, In, J Moraal and K. F. Kraiss (Eds.), Manned systems design: methods, equipment and applications, New York, NY, Plenum Press, 1981

Coad, P., Yourdon, E. (1991). *Object oriented design*. New York: Prentice-Hall.

Dahl, O-J. og Belsnes, D., *Algoritmer og datastrukturer*, Lund, Sverige, Studentlitteratur, 1973

Department of Defense, *Human engineering requirements for military systems, equipment and facilities*, MIL-H-46855B, 31 January 1972

Dhillon, B. S. *Human Reliability with Human Factors*, New York, N.Y.: Pergamon Press Inc., 1986. ISBN 0-08-032774-5, ISBN 0-08-033981 (paperback)

DiStefano, J.J, Stubberud, A.R and Williams, I.J., *Feedback and Contol Systems*, McGraw Hill, 1967.

Durrett, H. D., (redaktør), *Color and the Computer*, Orlando, Florida: Academic Press, Inc., 1987. ISBN 0-12-225210-1

Elkind , J. I., et. al. (redaktører), *Human Performance Models for Computer-Aided Engineering*, Washington, D.C.: National Academy Press, 1989.

Ellis H C and Reed Hunt R (1993): *Fundamentals of cognitive psychology*, Brown & Benchmark, Madision, Wisconsin.

Enge M, Ekeland P R, Lie S, Sjøberg S (1984): *Gamle tanker i nye hoder. En empirisk undersøkelse av elevers og studenters forståelse innenfor områder i mekanikk*. UiO, Inst. for fysikk, rapport 84-21.

Essens P J M D *et al* (1994): *COADE: A framework for cognitive analysis, design and evaluation* In: AC/243(Panel 8/RSG.19), Brussels, NATO Defence Research Section.

Eysenck M W and Keane M T (1990): *Cognitive psychology*. A student´s handbook, Lawrence Erlbaum Associates, Hove, UK.

Farhoodi, F. (1993). *CADDIE - An advanced tool for organisational design and process modelling*. In Software Assistance for Business Re-Engineering. New York: Wiley Publishers.

Fishwick, P. A. and Ziegler, B. P., *A Multimodel Methodology for Qualitative Modeling Engineering*, ACM Transaction on Modeling and Computer Simulation, Vol. 2, No. 1, January 1992, pp. 52-81

Fitts, P. M., *Functions of man in complex systems*, Aerospace Engineering, January, 1962, pp. 34-39

Fitts, P. M., *Some basic questions in designing an air-navigation and traffic-control system*, In O. M. Fitts (Ed.), Human engineering for an effective air-navigation and traffic-control system, Washington, DC, National Research Council, pp. 31-56, 1951

Foley J. D, Wallace V L and Chan P (1981*): The human factors of graphic interaction tasks and techniques*, Report GWU-IIST-81-3, The George Washington University,

Foley, J.D., Andries van Dam, Steven K. Feiner og John F. Hughes, *Computer Graphics. Principles and Practice, Second Edition*, The Systems Programming Series, Reading, Massachusetts: Addision-Wesley Publishing Company, 1990. ISBN 0-201-12110-7

Gelb, A. (Ed.), *Applied Optimal Estimation*, Cambridge, Massachusetts, MIT Press, 1974

Gilmore, W. E., & David I. Gertman og Harold S. Blackman, *User-Computer Interface in Process Control. A Human Factors Engineering Handbook*, Boston, Massachusetts: Academic Press, 1989. ISBN 0-12-283965-X

Grant, G. and Mayers, T., *Cognitive task analysis?*, In G. R. S. Weir and J. L. Alty (Ed.), Human-Computer Interaction and Complex Systems, London, Academic Press, pp. 147-167, 1991

Grossman RL, Nerode A, Ravn A P and Rischel H (Eds.): *Hybrid Systems*, Berlin, Springer-Verlag, 1993

Harel, D (1992): *Biting the silver bullet. Toward a brighter future for system development*. IEEE Computer, January 1992, 8-20

Harel, D (1987): *Statecharts: A Visual Formalism for Complex Systems*, Science of Computer Programming, vol 8, no. 3, pp. 231-274.

Hastie R, Dawes RM (2001): Rational Choice in an Uncertain World, Sage Publications, Thousand Oaks, California.

Hellander, M., (redaktør), *Handbook of Human-Computer Interaction*, Amsterdam Nederland: North-Holland Elsevier Science Publishers, 1988. ISBN 0-444-70536-8

Hess, R. A., *Feedback Control Models*, In In G Salvendy (ed.), Handbook of Human Factors, New York, NY, John Wiley & Sons, 1987

Hofstad K, Løland S and Scott P (1993): *Norsk dataordbok*, Universitetsforlaget, Oslo, Norge.

Hogarth R M (1987): *Judgement and Choice* (2nd ed.), Wiley, New York.

Hsia P, Davis A, Kung D (1993): *Status report:Requirements engineering*, IEEE Software Nov 1993, 75-79.

Høyland, A., *Sannsynlighetsregning og statistisk metodelære, del I og II*, Trondheim, Tapir, 1976, 1977

Jain, R. (1991). *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation and modelling*. New York: John Wiley & Sons Inc

Kalman R. E. et al., *Topics in Mathematical System Theory*, New York: McGraw-Hill, 1969.

Jobling, C.P., et al. (1994). *Object-oriented programming in control system design: a survey*. Automatica, 30 (8), 1221-1261.

Karlin & Taylor: *A first course in stochastic processes* (2nd ed.), New york: Academic Press, 1975.

Kirwan B. and Ainsworth L.K. (1992): *A guide to task analysis*. Taylor&Francis.

Krueger, M.W., *Artificial Reality II*, Reading, Mass: Addision-Wesley Publishing Company, 1991. ISBN 0-2-1-52260-8

Laughery, K. R. & Laughery, K. R. (1987). *Analytical techniques for function analysis*. In G. Salvendy (Ed.), Handbook of human factors. Wiley-Interscience Publication, pp. 329-354.

Laughery, K. R., *Micro SAINT - A Tool for Modeling Human Performance in Systems*, In G R McMillan et al. (Eds.), Applications of Human Performance Models to System Design, New York, NY, Plenum Press, 1989, pp. 219-230

Laural B. (Ed.), *The Art of Human-Computer Interface Design*, Addision-Wesley, 1991 ISBN 0-201-51797-3

Lenorovitz, D.R:, Philips, M.D., Ardrey, R. S. and Kloster, G. V., *A taxonomic approach to characterizing human-computer interfaces*, In Proceedings of the First USA-Japan Conference on Human Factors and Computer Systems, August 1984

Loy, P.H. (1990). *A comparison of object-oriented and structured development methods*. In Thayer R.H., Dorfman M., Eds. System and software requirements engineering. Los Alamitos: IEEE Computer Society Press.

MacDonald L. W., Lowe, A., Display Systems – Design and Applications, John Wiley, Chichester, UK, 1997.

Madsen, O.L., et al. (1993). *Object-oriented programming in the BETA programming language*. Wokingham: Addison-Wesley

McMillan, G. R., et. al. (redaktører), *Applications of Human Performance Models to Systems Design*, Proceedings of NATO Research Study Group 9 Workshop on Application of Human Performance Models to Systems Design, May 9-13, 1988, Orlando, Florida, Defense Research Series, New York, NY: Plenum Press, 1989. ISBN 0-306-43242-0

McRuer, D. T., Graham, D. Krendel, E., and Reisener, W. Jr., *Human pilot dynamics in compensatory systems*, Air Force Flight Dynamics Laboratory: AFFDL-TR-65-15, 1965

Meister, D. (1985). *Behavioural analysis and measurement methods*. New York: Wiley.

Meister, D. (1991). *Psychology of Systems Design*. Amsterdam: Elsevier.

Mendel, J.M. (1995): Fuzzy logic systems for engineering: A tutorial. Proceedings IEEE, Vol. 83, No. 3, pp. 345- 377.

Miller, G. A., *The magical number seven, plus or minus two: Some limits on our capacity for processing information*, The Psychological Review, Vol. 63. No 2., pp. 81-97, March 1956

Monarchi, D.E., Puhr, G.I. (1992). *A research typology for object-oriented analysis and design*. Communications of the ACM*,* 35 (9), 35-47.

Monk, A. (redaktør), *Fundamentals of Human-Computer Interaction*, Computer and Peoples Series, London, England: Academic Press, 1984. ISBN 0-12-504580-8

Myers, B. A, *Creating User Interfaces by Demonstration*, Perspectives in Computing Series vol. 22, Boston, Massachusetts: Academic Press, Inc., 1988. ISBN 0-12-512305-1

NATO, *Final report from RSG.12 on computer-human interaction in command and control*, AC/243 (Panel8/RSG.12)D/7, Brussels: NATO Defence Research Group, 1990

Neale J. M., & Liebert, R. M., *Science and Behaviour*: *An Introduction to Methods of Research,* Prentice-Hall series in social learning theory, 1986, ISBN 0-13-795121-3.

Nicol, R., et al. (1993). *Object orientation in heterogeneous distributed systems*. IEEE Computer*,* June 1993, 57-67.

Nielsen, J., *A virtual protocol model for computer-human interaction*, International Journal of Man-Machine Studies, Vol 24, pp. 301-312, 1986

Nordø E (1983): *Operatørinformasjon i målfølging*. Hovedfagsoppgave UiO.

Nordø E, Bråthen K (1995): (U) NATO DRG Panel-8/RSG.14 Workshop "Improving function allocation for integrated systems design", FFI/REISERAPPORT-95/00642, Forsvarets forskningsinstitutt.

Nordø, E. and Bråthen, K., *The function allocation process and modern system/software engineering*, In D. Beevis, P. Essens and H. Schuffel (Eds.), Improving Function Allocation for Integrated Systems Design, Wright-Patterson Air Force Base, OH, CSERIAC State-of-the-Art Report, pp. 103-120, 1996

Norman, D. A., *Turn Signals are the Facial Expressions of Automobiles*, Reading, Massachusetts: Addision-Wesley, 1992, ISBN 0-201-58124-8

Norman, D.A., & Draper, S.W. (redaktører), *User Centered System Design. New Persepctives on Human-Computer Interaction*, Hillsdale, New Jersey: Lawrence Erlbaum Associates, Publishers, 1986. ISBN 0-89859-781-1, ISBN 0-89859-872-1 (paperback)

Onken, R., *Human-centered cockpit design through the knowledge-based cockpit assistant system (CASSY)*, In D. Beevis, P. Essens and H. Schuffel (Eds.), Improving Function Allocation for Integrated Systems Design, Wright-Patterson Air Force Base, OH, CSERIAC State-of-the-Art Report, pp. 179-198, 1996

Open Software Foundation, *OSF/Motif™ Style Guide, Revision 1.1*, Englewood Cliffs, New Jersey, Prentice Hall, 1991

Orne, M.T. (1962). *On the social psychology of the psychological experiment: With particular reference to demand characteristics and their implications*. American Psychologist*, 17*, pp. 776-783.

Papenhuijzen, R. (1985): *Simulation and model studies of the behavor of navigators*, Proceedings of the Conference on Simulations and Traffic, Delft, The Netherland, pp. 268-?

Pfaff, G. (Ed.) (1985) *User Interface Management Systems*, Berlin, Springer Verlag.

Philips L D, Hays W L, Edwards W (1966*): Conservatism in complex probabilistic inference*, IEEE Trans. on. Human Factors in Electronics.

Philips M D, Bashinski H S, Ammerman H L , Fligg C M (1988): *A task analytic approach to dialogue design* In: Handbook of Human-Computer Interaction (ed M Helander), Elsevier Science Publishers, New York, pp. 835-858.

Plous S (1993): The Psychology of Judgement and Decsision Making. McGraw-Hill, New York.

Price, H.E. (1985). *The allocation of functions in systems*. Human Factors, 27 (1), 33-45.

PRINSIX (1991). *Prosjekthåndbok for Forsvaret, del 2 Prosjektledelse*.
Prinsix koordineringssenter, Forsvarets tele- & datatjeneste, Oslo mil/Akershus,
0015 Oslo.

Rasmussen, J., *Skills, Rules, and Knowledge; Signals, Signs, and Symbols, and Other Distinctions in Human Performance Models*, IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-13, No. 3, May/June 1983, p. 257-266

Rasmussen, J., *Information Processing and Human-Machine Interaction. An Approach to Cognitive Engineering*, North-Holland Series in System Science and Engineering vol. 12, New York: North-Holland, 1986. ISBN 0-444-00987-6

Rasmussen, J, *Cognitive Engineering*, In H.-J. Bullinger and B. Shackel, Human-Computer Interaction-INTERACT'87, Amsterdam, North-Holland, 1987

Rasmussen, J., Pejtersen A M and Goodstein L P (1994): *Cognitive systems engineering*, John Wiley & Sons, New York, N.Y.

Reid, G. B. and Nygren, T. E., *The subjective workload assessment technique: A scaling procedure for measuring mental workload.* In P A Hancock and N Meshkati (Eds.), Human mental workload, Amsterdam, North-Holland,, pp. 185-218, 1988

Rheingold, H., *Virtual Reality*, New York, N.Y.: Summit Books, 1991. ISBN 0-671-69363-8

Rine D C, Bhargava B (1992): *Object-Oriented Computing*, IEEE Computer **25**, 10, 8.

Rouse, W. B., *A Theory of Human Decision Making in Stochastic Estimation Tasks*, IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-7, No. 4, pp.274-283, 1977

Rouse, W. B., *Systems Engineering Models of Human-Machine Interaction*, North Holland Series in Systems Science and Engineering vol. 6, New York: North Holland, 1980. ISBN 0-444-00366-5

Rouse, W. B., (redaktør), *Advances in Man-Machine Systems Research,. A Research Annual*, Greenwich, Connecticut, første gang 1985. ISBN 0-89232-753-7 (88)

Rouse, W. B., *Design for Success. A Human-Centered Approach to Designing Successful Products and Systems*, Wiley Series in Systems Engineering vol 2, Wiely & Sons, Inc, 1991, ISBN 0-471-52483-2

Rouse, W. B., Hammer, J.M. and Lewis, C., *On Capturing Human Skills and Knowledge: Algorithmic Approaches to Model Identification*, IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-19, No. 3, May/June 1989, pp.558-573

Rumbaugh, J., et. al., *Object-Oriented Modeling and Design*, Englewood Cliffs, New Jersey: Prentice Hall, 1991, ISBN 0-13-629841-9.

Sage, A. P. (redaktør), *System Design for Human Interaction*, New York, NY: IEEE Press, 1987. ISBN 0-87942-218-1

Senders, J.W., *The human operator as a monitor and controller of multi degree of freedom systems*, IRE Transaction on Human Factors in Electronics, vol. HFE-5, pp. 2-5, 1964

Sheridan, T. B., *Supervisory Control*, In G Salvendy (ed.), Handbook of Human Factors, New York, NY, John Wiley & Sons, 1987, pp. 1243-1268

Sheridan, T. B. (1989), *Telerobotics.* Automatica, Vol. 25, No. 4, pp. 487-507.

Sheridan, T. B., *Telerobotics, Automation and Human Supervisiory Control*, MIT Press, 1992 Press Inc., Harcourt Brace Jovanovich , Publ., 1991, ISBN:0-12-685245-6

Sheridan T. B.: *Allocating functions among humans and machines*, In D. Beevis, P. Essens and H. Schuffel (Eds.), Improving Function Allocation for Integrated Systems Design, Wright-Patterson Air Force Base, OH, CSERIAC State-of-the-Art Report, pp. 179-198, 1996

Sheridan, T. B., & William R. Ferrell, *Man-Machine Systems. Information, Control and Decision Models of Human Performance*, Cambridge, Massachusetts: The MIT Press, 1974,1981. ISBN 0 -262-19118-0, ISBN 0-262-69072-1 (paperback)

Sherr, S. (Ed.), *Input Devices*, New York, NY, Academic Press, 1988.

Sherr, S., Applications for Electronic Displays, John Wiley, New York, 1998.

Shneiderman B., *Designing the User Interface. Strategies for Effective Human-Computer Interaction*, Reading, Massachusetts: Addison-Wesley Publishing Company, 1986. ISBN 0-201-16505-8. Second Edition 1992, ISBN 0-201-57286

Skare Ø., *Modeller for operatørytelse*. Hovedfagsoppgave UiO, 1990.

Skare, Ø., Nordø, E., *Experience with Using Micro SAINT Simulation Tool for a Submarine Command and Control System Model*, FFI/NOTAT-90/7062, Forsvarets forskningsinstitutt, 1990.

Smith S. L. & Jane N. Mosier, *Guidelines for Designing User Interface Software*, ESD-TR-86-278, 1986. NTIS AD A177 198, National Technical Information Service, 5285 Port Royal Road, Springfield, Virginia 22161, USA

St. Denis, G., Bouchard, J. C. and Bergeron, G., *How to facilitate the process of translating system requirements into a virtual prototype: VPS, the design process, and human engineering*, Virtual Prototypes Inc., Montreal Toronto: DCIEM Report, 1990

Stokes, A., Wickens, C. and Kite, K., *Display Technology. Human Factors Concepts*, Warrendale, PA, Society of Automotive Engineers, Inc., 1990

Sutcliffe, A., *Human-Computer Interface Design*, New York, N.Y.: Springer-Verlag New York Inc., 1989. ISBN 0-387-91339-4

Terano, T., & Kiyoji Asai, Michio Sugeno, *Fuzzy Systems Theory and its Applications,* Boston, Academic

Thimbleby, H., *User Interface Design*, New York, N. Y.: ACM Press, 1990. ISBN 0-201-41618-2

Thomé B (Ed*.), Systems Engineering, Principles and Practice of Computer-based Systems Engineering*, Chichester, Hohn Wiley & Sons, 1993

Travis, D., *Effective Color Displays. Theory and Practice*, London, Academic Press, 1991

Treurniet W., van Delft J, and Paradis S., *Tactical Information Abstraction Framework in Maritime Command and Control*, In Modelling and Analysis of Command and Control, RTO Meeting Proceedings 38, 15-1 – 15-15, 1999.

Tufte, E. R., *The Visual Display of Quantitative Information*, Cheshire, Connecticut: Graphics Press, 1984

Tufte, E. R., *Envisioning Information*, Cheshire, Connecticut: Graphics Press, 1990.

Tufte, E. R., *Visual Explanations. Images and Quantities, Evidence and Narrative*, Cheshire, Connecticut: Graphics Press, 1997.

Veum, K. A., & Kramarics, A. M. S. (1990*). Requirement Driven Development: A system engineering method*. In Proceedings of the Norwegian Informatics Conference (NIK '90), Tapir, pp. 81-91.

Vincente, K. J, *Improving dynamic decision making in complex systems through ecological interface design: A research overview*. System Dynamics Review, Vol. 12, No. 4, pp. 251-279.

Vincente, K. J., *Cognitive Work Analysis. Toward Safe, Productive, and Healthy Computer-Based Work*, Mahwah, New Jersey: Lawrence Erlbaum Associates, 1999.

Vincente, K. J, and Rasmussen, J., *Ecological Interface Design: Theoretical Foundations*, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 22, No. 4, July/August 1992, p. 589-606

Wagner, E.,*The Computer Display Designer's Handbook*, Lund, Sverige: Studentlitteratur, 1988. ISBN 91-44-27661-3

Wetteland, C. R et. Al, *The human solution: Resolving manning issues onboard DD21.* In J.A. Joines et. al (Eds) Proceedings of the 2000 Winter Simulation Conference, pp. 1402-1406.

Weidenhaupt et al. *Scenarios in System Development: Current Practice*, IEEE Software March/April 1998.

Wiener, N., *Cybernetics: or Control and Communications in the Animal and the Machine*, Cambridge, Massachusetts, MIT Press, 1948

Wiik, P. and Bråthen, K., *Integrert bro for hurtigbåt - Fysiologiske og psykologiske forsøk i laboratorieprototyp*, FFI/NOTAT-94/00255, Forsvarets forskningsinstitutt, 1994

Wilson, G. F. and Eggemeier, F. T., *Mental Workload Assessment*, Gateway, Vol. V, No. 2, pp. 1-4, 1994

Winograd, T., & Fernando Flores, *Understanding Computers and Cognition. A New Foundation for Design*, Norwood, New Jersey: Ablex Publishing Corporation, 1986. ISBN 0-89391-050-3

Wohl, J., Serfaty, D., Entin, E., Deckert, J. and James, R., *Human Cognitive Performance in Antisubmarine Warfare: Situation Assessment and Data Fusion*, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 18, No. 5, September/October 1988, pp.777-786

Woodson, W., *Human Factors Design Handbook. Information and Guidelines for the Design of Systems, Facilities, Equipment, and Products for Human Use*, McGraw-Hill Handbook, New York, N.Y.: McGraw-Hill Book Company, 1981. ISBN 0-07-071765-6

Wortman, D. B., Duket, S. D., Seifert, D. J., Hann, R. L. Chubb, G. P. , *Simulation using SAINT: a user-oriented instruction manual*, (ARML-TR-77-61), Wright-Patterson Air Force Base, OH, Aerospace Medical Research Laboratory, 1978

Yourdon, E. (1989). *Modern structured analysis*. Englewood Cliffs, New York: Yourdon

Zadeh, L. A., *Outline of a New Approach to the Analysis of Complex Systems and Decision Processes*, IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-3, No. 1, January 1973, pp.28-44

Øgaard O. (1990). *A situation adaptive presentation system for process control*. Dr.Ing. Thesis. NTH, Trondheim.

APPENDIKS

A TILSTANDSMASKINEN

La oss først uformelt definere noen begrep:

Symbol: udefinert
Mengde: samling av elementer (medlemmer av mengden) uten gjentakelse
Streng: endelig sekvens av symboler
Alfabet: endelig mengde med symboler
Et *formelt språk* er en mengde av strenger og symboler fra et alfabet
Det *kartesisk produkt* av mengdene A og B:
$A \times B = \{\forall (a, b): a \in A \text{ og } b \in B\}$ hvor (a,b) kalles et *ordnet par.*

En endelig tilstandsmaskin kan nå formelt defineres som:

$$M = (Q, \Sigma, \Delta, \delta, \lambda, q_0) \tag{A.1}$$

hvor:

Q er et mengde med tilstander og $q_0 \in Q$ (initiell tilstand)
Σ er et mengde med inngangssymboler (inngangsalfabet)
Δ er et mengde med utgangsymboler (utgangsalfabet)

La $y \in Q$, $x \in \Sigma$ og $z \in \Delta$. Vi kan nå definere funksjonene:

δ: $Q \times \Sigma \rightarrow Q$ (dvs $\delta(y, x) \in Q$) transisjonsfunksjonen
λ: $Q \times \Sigma \rightarrow \Delta$ (dvs $\lambda(y, x) \in \Delta$) utgangsfunksjonen

Vi bruker ofte følgende notasjon for å beskrive endring av tilstand og generering av utgangssymbol som funksjon av tidsindeksen n:

$$y_{n+1} = \delta(y_n, x_n) \text{ og } z_n = \lambda(y_n, x_n) \tag{A.2}$$

Istedenfor å operere med store mengder med inngangssymboler, tilstander og utgangssymboler kan man innføre vektorvariable slik man f eks gjør i datasystemer med vektorer av binær-variable.

$$\mathbf{y}_{n+1} = \boldsymbol{\delta}(\mathbf{y}_n, \mathbf{x}_n) \text{ og } \mathbf{z}_n = \boldsymbol{\lambda}(\mathbf{y}_n, \mathbf{x}_n) \tag{A.3}$$

I en *utvidet tilstandsmaskin* innføres tilleggsvektoren $\mathbf{u}$ som er gitt ved:

$$\mathbf{u}_{n+1} = \gamma\ (\mathbf{y}_n, \mathbf{x}_n, \mathbf{u}_n) \tag{A.4}$$

Variable i (vektoren) $\mathbf{u}$ kan være heltall eller reelle tall. Et av formålene med å innføre variabelvektoren $\mathbf{u}$ er å bruke denne ved generering av utgangssymboler, f eks for å redusere antallet tilstander.

I en utvidet tilstandsmaskin er utgangsfunksjonen modifisert slik:

$$\mathbf{z}_n = \lambda\ (\mathbf{y}_n, \mathbf{x}_n, \mathbf{u}_n) \tag{A.5}$$

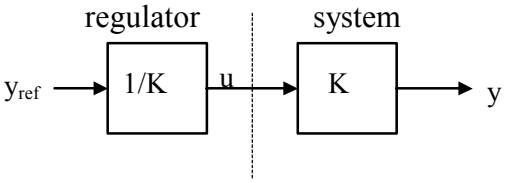
B TILBAKEKOBLING

Et *reguleringssystem* kan betraktes som en samling av (fysiske) komponenter som er sammenkoblet på en slik måte at de regulerer (styrer) et annet system (DiStefano et al, 1967). Formålet med reguleringssystemet identifiserer/definerer inngang (stimulus, pådrag) og utgang (respons). Hensikten med pådraget er vanligvis å få en spesifikk respons fra systemet. Vi kan grovt skille mellom tre typer av reguleringssystemer:

- systemer laget av mennesker (f.eks. mekaniske/elektriske)
- naturlige systemer (f.eks. biologiske)
- systemer med en blanding av de to foregående, dvs det vi vil kalle MMS

Reguleringssystemer er av typen *åpen sløyfe* eller *lukket sløyfe.* I et åpen sløyfe system er inngang (pådrag) uavhengig av utgang. I et lukket sløyfe system er inngang på en eller annen måte avhengig av utgangen Det siste kalles *tilbakekobling.*

Vi skal bruke et svært enkelt system for å illustrere forskjellen mellom disse typene og hvilke egenskaper de har. Anta gitt et system som kun multipliserer pådraget med en skalafaktor K: y = Ku, der y er respons og u er pådrag. Anta videre at vi ønsker å få responsen y til å følge en gitt referanse y_{ref} , dvs $y = y_{ref}$. Denne typen reguleringssystemer kalles for *følgesystemer* eller s*ervosystemer*. En åpen sløyfe regulator og systemet selv er vist i figuren under:

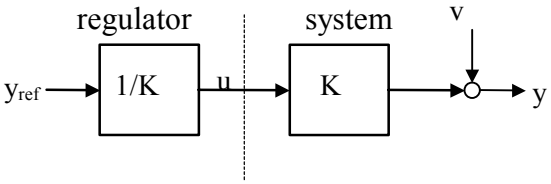


Figur B.1 Ideell åpen sløyfe

Vi ser at

$$y = K(\frac{1}{K})\, y_{ref} = y_{ref} \tag{B.1}$$

Systemet over er svært ideelt. I virkeligheten vil systemet utsettes for tilfeldige forstyrrelser og skalafaktoren vil ikke være kjent helt eksakt. F eks kan man tenke seg at K endres pga aldring. La oss først se på effekten av støy, v:



Figur B.2 Åpen sløyfe med støy

Vi ser fra figur B.2 at

$$y = y_{ref} + v \tag{B.2}$$

Dersom støyen er stor vil responsen kunne bli temmelig tilfeldig.

Dersom vår modell av systemet er feil, dvs at vi i regulatoren benytter K når systemets verdi er

$$K' = K + \Delta K \tag{B.3}$$

vil også responsen ikke bli som ønsket:

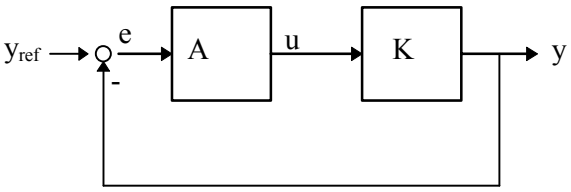
$$y = \frac{1}{K}(K + \Delta K)\, y_{ref} = (1 + \frac{\Delta K}{K})\, y_{ref} \tag{B.4}$$

Dersom vi har 10 % feil i K vil

$$\frac{y}{y_{ref}} = 1{,}1 \tag{B.5}$$

Dvs at vi ikke er i stand til å kompensere for feil i modellen vår. Nøyaktigheten er gitt av hvor nøyaktig systemet er kalibrert, dvs hvor liten ΔK kan gjøres.

Så langt kan vi altså konkludere med at et åpent sløyfe system i praksis ikke alltid vil være i stand til å oppnå det vi ønsker, nemlig at utgangen følger referansen tilstrekkelig nøyaktig. Virkningen av feilene over kan reduseres dersom vi lar pådraget være en funksjon av både referansen og utgangen. Dersom det finnes en slik lukket sløyfe av årsaks-virkningsrelasjoner i et system sies det å være *tilbakekoblet* og sløyfa kalles for *tilbakekoblingssløyfe.* Dvs y påvirker u som igjen påvirker y osv. For å få til dette kreves det at vi kan måle y. Dersom operatøren er regulatoren må han altså kunne få informasjon om y. Blokkskjemaet for et lukket sløyfe system blir nå som vist i figur B.3.



Figur B.3 Lukket sløyfe

Avviket mellom målt utgang og referanse er

$$e = y_{ref} - y \tag{B.6}$$

Vi får da at

$$y = AKe = AK(y_{ref} - y) \Rightarrow y = \frac{AK}{1 + AK} y_{ref} \tag{B.7}$$

Vi ser at

$$A \to \infty \Rightarrow y \to y_{ref} \tag{B.8}$$

La oss velge

$$A = \frac{100}{K} \Rightarrow y = \frac{100}{100 + 1} y_{ref} \approx 0{,}99\, y_{ref} \tag{B.9}$$

Vi antar at dette er godt nok. Dersom vi nå inkluderer støy på samme måte som over vil

$$y = AKe + v = AK(y_{ref} - y) + v \Rightarrow y = \frac{AK}{1 + AK} y_{ref} + \frac{1}{1 + AK} v \tag{B.10}$$

F eks, anta

$$A = \frac{100}{K} \Rightarrow y = \frac{100}{100 + 1} y_{ref} + \frac{1}{100 + 1} v \approx 0{,}99\, y_{ref} + 0{,}01 v \tag{B.11}$$

Vi ser at støyens virkning på utgangen nå er redusert med en faktor 100. Dersom vi antar modellfeil som over vil

$$y = \frac{A(K + \Delta K)}{1 + A(K + \Delta K)} y_{ref} \tag{B.12}$$

Hvis relativ feil er 10 % vil feilen i utgangen kun bli 1 %.

Systemet over er lineært, dvs y er en lineær funksjon av u. Like store relative endringer i u vil gi like store relative endringer i y. Dersom dette ikke er tilfelle, kalles systemet ulineært. Virkningen av dette vil for systemet vårt være den samme som om K varierte som funksjon av u. Som vi har sett over vil et lukket sløyfe system sterkt kunne redusere virkningen av slike ulinearitetene.

Eksemplet over var et statisk system. I praksis vil systemet være dynamisk, dvs selv ha en tilbakekobling. Tilbakekoblingen vi introduserte over er altså en kunstig tilbakekobling som vil komme i tillegg til den naturlige (dersom systemet hadde vært dynamisk).

Tilbakekobling, dvs et lukket sløyfe system, vil generelt ha følgende egenskaper relativt til et åpen sløyfe system:

- bedre nøyaktighet (evne til å følge en referanse)
- mindre følsomhet overfor støy i systemet
- reduserte effekter av ulinearitetene i systemet (se kapittel 3.4.5)
- økt båndbredde (evne til å følge raske referanseendringer)
- redusert usikkerhet
- tendens til oscillasjon og ustabilitet i systemet
- krever målinger

Tilbakekobling har som vi ser svært mange positive effekter, men disse oppnås ikke gratis. Som nevnt over må vi foreta måling av utgangen. Denne kan være flerdimensjonal og det kan også være nødvendig å estimere ikke direkte observerbare tilstander i systemet. Målinger/estimater vil generelt alltid være beheftet med feil.

Tilbakekobling kan som nevnt over føre til ustabilitet eller oscillasjoner i systemet, dvs at utgangen vokser eller svinger ukontrollert. Årsaken til dette kan ligge i systemet selv eller ved målingen av utgangen. Et typisk eksempel er transportforsinkelser i systemet, dvs at vi vil ikke måle virkningen av pådraget før etter noe tid.

FORDELINGSLISTE

FFIE **Dato:** 3 januar 2002

RAPPORTTYPE (KRYSS AV)	RAPPORT NR.	REFERANSE	RAPPORTENS DATO
X RAPP ☐ NOTAT ☐ RR	2001/04234	FFIE/730/134	3 januar 2002
RAPPORTENS BESKYTTELSESGRAD		**ANTALL EKS UTSTEDT**	**ANTALL SIDER**
UGRADERT		40	177
RAPPORTENS TITTEL		**FORFATTER(E)**	
UTVIKLING AV MENNESKE-MASKIN-SYSTEMER		BRÅTHEN Karsten, NORDØ Erik, JENSSEN Arne Cato	
FORDELING GODKJENT AV FORSKNINGSSJEF		**FORDELING GODKJENT AV AVDELINGSSJEF:**	
Vidar S Andersen		Johnny Bardal	

EKSTERN FORDELING

ANTALL	EKS NR	TIL
1		Institutt for energiteknikk
		OECD Halden Reactor Project
1		v/Angelia Sebok
		Postboks 173
		Halden
		FLO/Sjø
1		v/OK Knut Morten Hanssen
1		UNIK
1		v/Prof Oddvar Hallingstad
		NTNU
1		v/Prof Tor Onshus
1		v/Prof Arthur Aune
		KNM Tordenskjold
1		v/KL Jan Geelmuyden
		Kysteskadren
1		v/KL Ole Morten Sandquist

INTERN FORDELING

ANTALL	EKS NR	TIL
14		FFI-Bibl
1		Adm direktør/stabssjef
1		FFIE
1		FFISYS
1		FFIBM
1		FFIN
1		Karsten Bråthen, FFIE
1		Erik Nordø, FFIE
1		Arne Cato Jenssen, FFIE
1		Tore Smestad, FFIE
1		Stig Lødøen, FFIE
1		John-Mikal Størdal, FFIE
1		Vidar S Andersen, FFFIE
1		Anne Lise Bjørnstad, FFISYS
4		Arkiv, FFIE
		FFI-veven

FFI-K1 Retningslinjer for fordeling og forsendelse er gitt i Oraklet, Bind I, Bestemmelser om publikasjoner for Forsvarets forskningsinstitutt, pkt 2 og 5. Benytt ny side om nødvendig.