FFI RAPPORT

SIMULERINGSMETODER INNEN OPERASJONSANALYSE - En oversiktsstudie

KRÅKENES Tony, LJØGODT Håkon, MALERUD Stein

FFI/RAPPORT-2007/00297

SIMULERINGSMETODER INNEN OPERASJONSANALYSE - En oversiktsstudie

KRÅKENES Tony, LJØGODT Håkon, MALERUD Stein

FFI/RAPPORT-2007/00297

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-2007/00297	UNCLASSIFIED	55
1a) PROJECT REFERENCE	**2a) DECLASSIFICATION/DOWNGRADING SCHEDULE**	
FFI-I/1004/161.4	-	

4) TITLE

SIMULERINGSMETODER INNEN OPERASJONSANALYSE -
En oversiktsstudie

SIMULATION METHODS IN OPERATIONAL RESEARCH – A GOAL Survey

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

KRÅKENES Tony, LJØGODT Håkon, MALERUD Stein

6) DISTRIBUTION STATEMENT

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

7) INDEXING TERMS

IN ENGLISH:	IN NORWEGIAN:
a) Simulation	a) Simulering
b) Modelling	b) Modellering
c) Operational Analysis	c) Operasjonsanalyse
d)	d)
e)	e)

THESAURUS REFERENCE:

8) ABSTRACT

Simulation is an important method in the field of operational research (OR). This report gives an introduction to simulation, explaining the terminology in use. Four main categories of methods are described: a) Static Monte Carlo simulation; b) System dynamics; c) Discrete event simulation; d) Agent-based simulation. For each of these methods, the main principles are described, examples of use are given and tools are listed.

The report also discusses *when* it is appropriate to chose simulation as the solution method when faced with a problem. Analytical methods are generally preferable, but not always available, and simulation is then often the best option. Having chosen simulation as the analysis approach, the report gives guidelines on *how* to choose the appropriate simulation method. This decision largely depends on three main factors: a) the role of time; b) the presence of stochastic events; c) the level of system abstraction.

The report describes and employs a simple military scenario for the purpose of illustrating examples of problems that can be addressed with the different simulation methods.

This report is part of a series of surveys of OA methods conducted by project GOAL at FFI.

9) DATE	AUTHORIZED BY This page only	POSITION
2007-01-15	Jan Erik Torp	Director

ISBN 978-82-464-1143-9

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE
(when data entered)

INNHOLD

SIMULERINGSMETODER INNEN OPERASJONSANALYSE - En oversiktsstudie

1 INNLEDNING

Denne oversiktsrapporten om simuleringsmetoder innen OA er utarbeidet av prosjekt 1004 "Grunnlagsforskning operasjonsanalyse" (GOAL), og inngår i en serie med breddestudier av sentrale operasjonsanalytiske metoder. Andre rapporten i denne serien omhandler problem-strukturerende metoder (1) og flermålsanalysemetoder (2).

1.1 Bakgrunn

FFI har en lang tradisjon for å bygge og bruke simuleringsmodeller, både til operasjonsanalyse (OA) og andre analyseformål. Denne tradisjonen ser ut til å fortsette, om enn i en annen form enn tidligere. Under Den kalde krigen og til langt ut på 90-tallet kunne OA-modellene karakteriseres som omfattende, lite fleksible og orientert mot storskala, symmetrisk krig. Etter Den kalde krigens slutt har en ny og mye mer uklar trusselsituasjon etablert seg, med asymmetriske lavintensitetskonflikter og krisehåndtering som de mest fremtredende oppgavene for Forsvaret. Spekteret av mulige OA-problemstillinger synes dermed mye bredere enn før.

Tidligere var det en ambisjon å ha en nasjonal balansert forsvarsstruktur, som var optimalisert for å møte en invasjonstrussel. Denne problemstillingen medførte behov for store, integrerende simuleringsmodeller, både på forsvarsgrensnivå og på fellesoperativt nivå. De senere års fokus på styrkebidrag til internasjonale operasjoner har medført en redusert nyutvikling og bruk av store simuleringsmodeller innen OA. Til gjengjeld ser man en trend mot mindre, enklere og mer problemfokuserte modeller. Disse modellene er gjerne hurtige å lage og ikke nødvendigvis beregnet for gjenbruk.[1] Man kan gjerne si at før var fokuset på å *modellere systemet*, og deretter løse problemer så godt som mulig med modellene ("system før problem"), mens fokuset i dag er rettet mer mot å *modellere problemet*, og definere systemet ut ifra dette ("problem før system").

Parallelt med dette kommer en gryende bevissthet rundt fordelene ved å ha operasjons-analytikere som et fast stabsinnslag i operative hovedkvarter. Denne arbeidssituasjonen karakteriseres av hurtig oppdukkende analyseoppgaver og korte tidsfrister, noe som setter krav til både kunnskapen og verktøykassen av metoder en analytiker kan og bør ha.

Alle de vestlige lands militære styrker transformeres nå for å møte den nye, dynamiske trussel-situasjonen på en adekvat og robust måte. Det planlegges ut ifra et større spektrum scenarier, og asymmetriske trusler og hurtig oppdukkende oppgaver er sentrale stikkord. I tråd med denne utviklingen er en parallell transformasjon av analysemetodene også nødvendig. De endrede

[1] Gjennomslagskraften til slike ad hoc-modeller er i stor grad muliggjort av utviklingen av spesialiserte simuleringsverktøy (denne rapporten omtaler flere slike verktøy)

rammebetingelsene taler for en ny tilnærming til modellering og simulering innen OA. Det er vår hypotese at trenden mot mindre, enklere og mer problemfokuserte OA-modeller vil fortsette. Det er likevel grunn til å tro at etter hvert som man vinner erfaring med dagens konflikttyper og problemstillinger, vil behovet for større og mer detaljerte modeller vokse frem. Likeledes kan man forvente at etter hvert som skogen av mindre ad-hoc-modeller vokser seg stor, vil det bli stadig lettere å finne eldre modeller for gjenbruk.

Simulering som metode vil fremdeles være relevant og spille en viktig rolle for å analysere de problemstillingene som Forsvaret står overfor i dag og i fremtiden. Men det er noe uklart hvordan simulering kan bidra, og hvilke simuleringsmetoder som vil være viktige.

1.2 Formål med rapporten

Det er flere formål med denne rapporten. For det første skal rapproten gi en rask og lavterskel introduksjon til feltet simulering innen operasjonsanalyse og beskrive de mest sentrale simuleringsmetodene. Rapporten inneholder henvisninger til lærebøker, simuleringsverktøy og nettressurser. En slik oversikt over metoder anses som nyttig, både for erfarne brukere av simuleringsmetoder og nykommere som ikke kjenner feltet nevneverdig fra før. Den primære målgruppen er forskere i Faggruppe OA ved FFI.[2]

I tillegg til å beskrive de mest sentrale hovedklasser av simuleringsmetoder, søker rapporten å diskutere hvilke metoder som egner seg best for ulike typer problemstillinger. En sentral ambisjon er å kunne si noe om hvilke metoder det vil være behov for fremover. I drøftingen av dette vil behovene til Faggruppe OA vil være styrende.

Det presiseres at studien kun ser på simulering som verktøy innen feltet *operasjonsanalyse*. Studien har ikke i særlig grad behandlet simulering der det primære formålet er visualisering, simulatorutvikling, trening m.m. Grensene mellom de ulike simuleringsregimene er uklare og til dels miljøskapte, og det finnes flere likheter enn ulikheter.

Rapporten er å anse som en delvis oppdatering og utvidelse av FFI-rapporten ”Introduksjon til simulering” fra 1994 (3). En annen sentral litteraturkilde for dette arbeidet er læreboken ”Simulation Modeling and Analysis” av Law & Kelton (4).

1.3 Rapportens innhold

I tillegg til å gi en kortfattet og generell innføring i feltet simulering for operasjonsanalyse, beskriver rapporten spesielt fire viktige klasser av simuleringsmetoder:

- Statisk Monte Carlo-simulering (SMC)
- Systemdynamikk (SD)

[2] Ved FFI er det i dag Faggruppe OA som faglig organiserer de personene og miljøene som i utgangspunktet vil bygge og bruke simuleringsmodeller av den typen det her er snakk om. Denne faggruppen er en forholdsvis ny konstruksjon, og kunnskaps- og erfaringsnivået blant medlemmene i gruppen varierer en del

- Diskret hendelsesstyrt simulering (DHS)
- Agentbasert modellering og simulering (ABMS)

Dette synes å være de fire viktigste hovedklassene av simulering, både i litteraturen og i bruk ved FFI. Det tre første er tradisjonelle og velkjente metoder. Den siste, ABMS, er en relativt ny tilnærming til modellering og simulering innen OA som er mye i vinden for tiden. GOAL utgir en egen rapport om ABMS og metodens anvendelighet for OA ved FFI (5).

Rapporten beskriver og benytter seg av et stilistisk scenario for å illustrere disse fire hovedtypene simulering. Scenariet tjener som rammeverk for eksempler på problemstillinger som de ulike metodene er egnet til å løse. Erfaringer med dette scenariet blir også brukt som grunnlag for en grov sammenligning av metodenes styrker, svakheter og mulige anvendelsesområder.

Rapporten er disponert som følger. Kapittel 2 gir en generell innføring i feltet simulering for operasjonsanalyse. Kapitlet beskriver simuleringens plass innen OA og inneholder en fyldig avklaring av begreper knyttet til simulering. Sammenhengen mellom problem, system, modell og simulering er sentral her. Utvikling av simuleringsmodeller og gjennomføring av simulering blir også viet plass. Scenariet som brukes for å illustrere metoder introduseres i kapittel 3, mens selve metodene beskrives separat i kapitlene 4–7. Struktur og innhold i disse metodekapitlene varierer Kapittel 8 inneholder en diskusjon og studiens konklusjoner.

2 HVA ER SIMULERING?

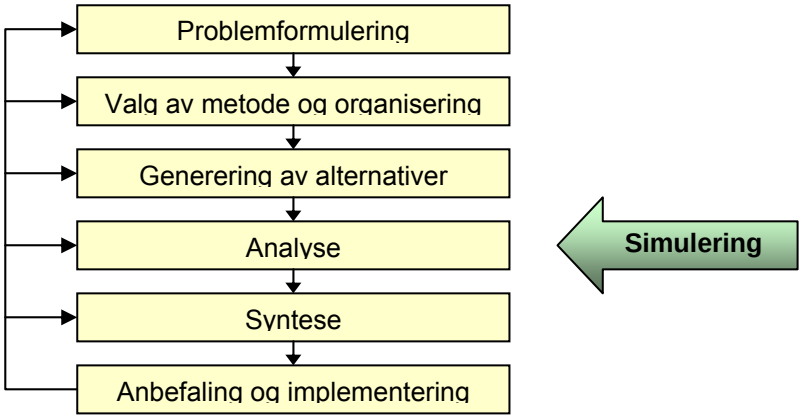
Begrepet simulering rommer mye, og er dermed upresist. I denne sammenhengen begrenser vi oss til å omtale simulering som en metode innen operasjonsanalyse (OA). Simulering i OA-sammenheng er å gjennomføre eksperimenter med en modell av et system. En nærmere beskrivelse av simulering krever derfor kjennskap til begrepene ”operasjonsanalyse”, ”system” og ”modell”.

2.1 Hva er operasjonsanalyse?

Operasjonsanalyse er et ullent fagfelt for veldig mange, og grensene mellom OA og relaterte fagfelt er uklare. Det finnes mange definisjoner av hva operasjonsanalyse er, alt etter hvilket aspekt ved OA man velger å vektlegge. En klassisk definisjon fra (6) er ”*Operations research is a scientific method of providing executive departments with a quantitative basis for decisions regarding the operations under their control*”. En mer moderne definisjon som noen sentrale OA-organisasjoner[3] opererer med, er ”*OR is the discipline of applying advanced analytical methods to help make better decisions”*, med slagordet ”*OR is the science of better*”. Felles for de fleste definisjoner er at OA skal bidra til å gi bedre beslutninger ved å fremskaffe et godt beslutningsgrunnlag. Tidlig OA – fra andre verdenskrig og årene etter – la vekt på det kvantitative aspektet ved beslutningsstøtten. OA i dag er imidlertid mye bredere og gjør utstrakt bruk av også mer kvalitative metoder (ofte kalt ”soft OA”) (1).

[3] Bl.a. britiske OR Society (www.theorsociety.com) og amerikanske INFORMS (www.informs.org)

Figur 2.1 viser den overordnede arbeidsprosessen i en operasjonsanalyse. Simulering er en av mange metoder/teknikker som benyttes innen OA, og da primært på analysesteget i arbeids-prosessen. Eksempler på andre viktige metoder er optimering, problemstrukturering, risiko-analyse, sannsynlighetsregning/statistikk, beslutningsteori m.m. For en generell innføring i operasjonsanalyse, anbefales f.eks. (7), (8), (9). En god innføring i "soft OA" finnes i (10).



Figur 2.1 Figur av OA-arbeidsprosessen og simuleringens plass i denne

2.2 Hva er et system?

Et *system* kan enkelt fortalt forstås som den avgrensede delen av virkeligheten som angår det foreliggende problemet. Et system kan ses på som en helhet, bestående av ulike elementer som står i relasjon til hverandre (ikke nødvendigvis alle-til-alle relasjoner) og påvirker hverandre gjensidig. Elementer som ikke har en relasjon til *noen* andre elementer i et system, er følgelig ikke en del av systemet. Hvordan denne avgrensingen av et system gjøres, avhenger av problemet man er interessert i å studere, og hvilken grad av detalj som er ønskelig i systembeskrivelsen. Et system er altså en mer eller mindre subjektiv konstruksjon.

Videre er det vanlig å skille et system fra et *nettverk* eller en *struktur*– som også er en samling elementer med innbyrdes relasjoner – ved å tillegge systemet en eller annen funksjon eller mening. Dette er ingen mystisk sak; ved å beskrive og avgrense et system, har man i praksis også sagt hva dette systemet er til for.

Den verdensanskuelsen der alt kan ses på og beskrives som systemer, blir kalt *systemtenkning*. Fagfelt som har sin direkte bakgrunn i systemtenkning er bl.a. kybernetikk/kontrollteori, katastrofe- og kaosteori, samt "komplekse adaptive systemer". Simuleringsmetoden system-dynamikk, som beskrives i kapittel 5, bør også nevnes spesielt i denne sammenhengen. Det meste av moderne operasjonsanalyse kan også sies å basere seg i betydelig grad på system-tenkning.

2.3 Hva er en modell?

Når man ønsker å studere et system, kan dette i prinsippet gjøres på to måter: man kan eksperimentere med selve systemet, eller man kan eksperimentere med en *modell* av systemet. En modell kan i denne sammenheng defineres som en forenklet representasjon av et system, som innehar de samme relevante egenskapene som systemet. En modell er således en abstraksjon, og detaljeringsgraden i modellen avhenger av hvilke aspekter av systemet som skal beskrives, og hvor nøye man ønsker å være i beskrivelsen. Dette betyr at modeller – som systemer – langt på vei er subjektive konstruksjoner. Dette innebærer også at modeller i prinsippet bør tilpasses det foreliggende problemet, og at det ikke finnes universelle modeller som kan tjene alle tenkelige analyseformål.

Direkte eksperimentering med selve systemet er selvsagt i utgangspunktet den beste og mest realistiske måten å studere et system på. Imidlertid er direkte eksperimentering ofte umulig eller lite hensiktsmessig i praksis. Årsaker til dette kan være at det er ødeleggende, dyrt, tidkrevende eller at systemet rett og slett ikke eksisterer ennå. Man er da nødt til å ty til modellering.

Modellering har mange fordeler. For det første tvinger det oss til å tenke strukturert omkring problemet, og gir dermed økt forståelse av systemets oppbygging og virkemåte. Modeller krever gjerne at vi beskriver og tallfester sammenhenger, noe som bidrar ytterligere til konkretisering av et system. Videre vil en modell støtte kommunikasjon gjennom en presis og utvetydig beskrivelse av systemet. Sist, men viktigst i denne sammenhengen, kan modeller hjelpe oss til å løse problemer i tilknytning til systemet vi studerer.

Ytterligere fordeler med modeller er at man lettere kan

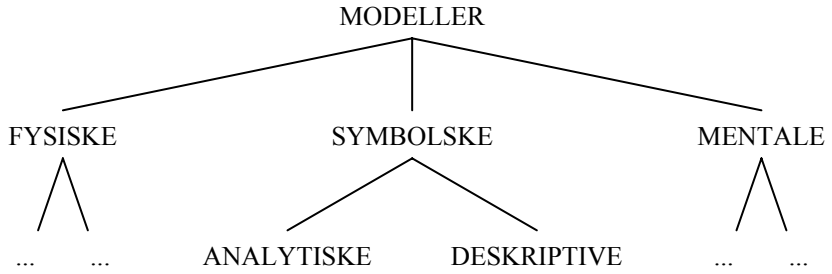
- evaluere et system under endrede forutsetninger, og dermed undersøke konsekvensene av forskjellige beslutninger (”hva hvis?”-spørsmål)
- sammenligne alternative systemdesign for å velge den beste kandidaten eller vurdere en utskifting av et eksisterende system
- kartlegge sannsynligheter for hendelser
- studere systemer med lang endringstid i komprimert tid, og dermed muliggjøre prediksjon av systemets utvikling i fremtiden
- støtte opplæring og øving av de som skal operere selve systemet eller i tilknytning til det

Modeller kan kategoriseres på flere måter. En vanlig inndeling er i fysiske, symbolske og mentale modeller. En globus er et eksempel på en *fysisk* modell av jordoverflaten, mens en persons oppfatning av bilmotorens oppbygging og virkemåte er et eksempel på en *mental* modell. I OA-sammenheng er man mest interessert i *symbolske* (også kalt *matematiske* eller *formelle*) modeller. Symbolske modeller kan beskrives på en generell matematisk form med ligninger og logiske sammenhenger.

Symbolske modeller kan videre deles inn i analytiske og deskriptive modeller. I en *analytisk* modell kan løsninger på beslutningsproblemer utledes matematisk. Et ligningssett – vel å merke

et som lar seg løse analytisk – er et eksempel på en analytisk modell. Som oftest er det bare avgrensede og enkle problemer som lar seg løse analytisk. En symbolsk modell som *ikke* lar seg løse analytisk, kan kalles en *deskriptiv* modell fordi den nøyer seg med å kun *beskrive* sammenhengene i systemet. [4] En slik beskrivelse angir ikke hvordan problemet skal løses, men modellen gjør det mulig å studere systemets oppførsel i ulike situasjoner.

Simulering er én måte å utnytte deskriptive modeller på. En *simuleringsmodell* er altså i denne terminologien å forstå som en symbolsk, deskriptiv modell som man gjør eksperimenter med, dvs. eksekverer numerisk. Figur 2.2 viser den beskrevne kategoriseringen av modeller.



Figur 2.2 Kategorisering av modeller. En simuleringsmodell er en symbolsk, deskriptiv modell som man gjør eksperimenter med

Deskriptive modeller implementeres som regel i et programmeringsspråk eller et dertil egnet dataverktøy (se kapittel 2.5). Implementasjonen av modellen omtales ofte for enkelhets skyld som ”modellen”, men konseptuelt er modellen og dens implementasjon to forskjellige ting, og disse begrepene bør holdes atskilt.

2.4 Definisjoner av simulering

”Simulering” er et begrep med mange betydninger i dagligtalen og i fagspråket. Ordet stammer fra det latinske verbet ”simulare” som betyr å late som, imitere, gjøre seg lik. I operasjonsanalysesammenheng er simulering en fremgangsmåte og et verktøy for å kunne studere systemer. Å simulere et system innebærer altså å etterape systemet, og dette gjøres ved hjelp av en modell av systemet. I denne studien avgrenser vi oss til kun symbolske, deskriptive modeller.

Det er vanskelig å komme med en kortfattet og fullstendig dekkende definisjon av hva simulering er. Innledningsvis ”definerte” vi simulering til å være ”eksperimentering med en modell av et system”. Avhengig av hva man legger vekt på ved simuleringen, og ut i fra faglig ståsted, kan man utvide eller spisse denne definisjonen i ønsket retning. Mange litteraturkilder legger f.eks. vekt på at simulering skal være koblet til et tidsforløp:

> Simulation: The imitation of the operation of a real-world process or system over time. (Banks et al.) (11)

[4] ”Deskriptiv” i denne sammenhengen skal altså ikke forstås som motsatsen til ”normativ”, men som ”ikke-analytisk” eller ”(rent) beskrivende”

> Simulering er experiment med en modell för att följa och förstå beteenden och orsakssammanhang under ett tidsförlopp. (G Holm, FOI) (12)

Andre igjen fremhever at simulering er et begrep forbeholdt stokastiske modeller. Fra statistikk-hold er simulering tett koblet til stokastiske modeller og prosesser:

> Simulation: The artificial generation of random processes (usually by the means of pseudorandom numbers and/or computers) to imitate the behaviour of particular statistical models. (The Cambridge Dictionary of Statistics) (13)

Noen definisjoner av mer generell karakter:

> Simulate: Imitate the conditions of (a situation or process), esp. for the purpose of training etc. (New Shorter Oxford English Dictionary) (14)
>
> Simulation: Imitation of some real thing, state of affairs or process. The act of simulating something generally entails representing certain key characteristics or behaviours of a selected physical or abstract system. (Wikipedia)[5]

Denne opplistingen av definisjoner viser at simulering langt i fra er noe entydig begrep, noe som kan skape grunnlag for misforstått kommunikasjon mellom ulike miljøer. Det er derfor viktig at en leser som eksponeres for begrepet, setter seg inn i hva forfatteren egentlig mener, og er åpen for å akseptere andre tolkninger enn sin egen.

2.5 Simulering på datamaskin

Simulering kan anvendes på alle typer modeller – både fysiske, mentale og symbolske, og krever ikke nødvendigvis bruk av datamaskiner. Når datamaskin benyttes (datamaskinbasert simulering), brukes imidlertid kun deskriptive modeller. I praksis er all simulering i dag datamaskinbasert, både for OA og andre formål. I resten av denne rapporten er det derfor underforstått at simulering er ensbetydende med datamaskinbasert simulering.

Datamaskinbasert simulering krever at modellene beskrives som dataprogrammer vha. et programmeringsspråk. De fleste programmeringsspråk kan i prinsippet brukes til simulering, men noen språk er mer velegnet enn andre.[6] I de senere år har spesialiserte dataverktøy for flere typer simulering dukket opp, der et velutbygd brukergrensesnitt og ferdig funksjonalitet i stor grad skjermer brukeren for ubekvemmelighetene ved selve kodegenereringen. Flere av disse verktøyene vil bli nevnt i beskrivelsen av de ulike simuleringsmetodene i kapitlene 4–7.

Den viktigste grunnen til å bruke datamaskiner til simulering er maskinenes evne til å simulere hurtig og korrekt. De feil som eventuelt oppstår skyldes feil i implementasjonen eller sågar i modellen (se kapittel 2.9 om verifikasjon og validering av modeller). Store modeller med flere tusentalls linjer programkode kan eksekveres på sekunder med dagens datamaskiner. Denne

[5] http://en.wikipedia.org/wiki/Simulation
[6] Programmeringsspråk med stavelsen ”sim” i navnet er som regel beregnet for simulering

hurtigheten er avgjørende for å kunne gjennomføre det store antall eksperimenter som må til for å kunne finne gode løsninger på problemer knyttet til store og komplekse systemer. Manuelle eksperimenter med tilsvarende modeller eller selve systemet kan aldri gjennomføres på en tilsvarende komprimert og konsistent måte.

Det finnes to hovedperspektiver på simuleringsmodeller innen programmeringsverdenen: diskret hendelsesstyrt simulering og kvasi-kontinuerlig (tidsdiskret) simulering. Forskjellen ligger i hvordan tidsdimensjonen håndteres (se kapittel 2.7.7).

2.6 Fordeler med simulering

Svaret på spørsmålet ”hvorfor simulere?” henger nøye sammen med svaret på spørsmålet ”hvorfor *modellere*?”. Alle fordelene knyttet til modellering (se kapittel 2.3) gjelder således også for simulering. Det er spesielt tiltalende at det er enkelt å kunne studere modifiserte systemer, samt at det er mulig å komprimere eller utvide systemets endringstid. I tillegg kan det fremheves at simulering på datamaskin er billig, nøyaktig og tillater mange eksperimenter på kort tid, noe som er spesielt viktig for analyse av stokastiske systemer.

Usikkerhet er en fundamental egenskap i de fleste typer beslutningsproblemer i den virkelige verden. Et godt beslutningsgrunnlag vil være ufullstendig uten en diskusjon av usikkerheten i de resultatene som presenteres. Tilgang på datakraft tillater grundig og automatisk utforsking av mulighetsrommet for et system, og simulering er derfor en velegnet metode for å studere usikkerhet (vel å merke usikkerhet som er ”kjent” i form av sannsynlighetsfordelinger).

Hvorfor skal man velge simulering som analysemetode fremfor andre tilgjengelige metoder? Det er viktig å understreke at simulering ikke er et mål i seg selv; det man egentlig er ute etter er forenklingen som modellen representerer og svarene modellen kan gi. Simulering er bare en måte å komme frem til disse svarene i de tilfellene man ikke kan finne en analytisk løsning. Matematiske systemmodeller blir fort for komplekse til å kunne løses analytisk. Faktorer som bidrar til økt kompleksitet er ikke-linearitet, stokastikk, store parameterrom m.m. I praksis kan det forekomme at modeller som faktisk *har* en analytisk løsning, likevel blir simulert på grunn av mangel på tid eller kompetanse til å finne den analytiske løsningen.

Simulering benyttes altså til å studere oppførselen til og trekke slutninger om et system. Det presisertes at eventuelle slutninger baserer seg på estimater, og er således ingen løsning i matematisk forstand.

2.7 Klassifisering av simuleringsmetoder

Man opererer med svært mange ulike begreper for å beskrive og karakterisere modeller og simuleringsteknikker, og det er lett å miste oversikten og blande disse begrepene sammen. Ulike miljøer har i tillegg sprikende oppfatninger av hvordan begrepene bør brukes. I det følgende listes og forklares de viktigste sorteringsdimensjonene for simulering.

2.7.1 Tidsaspektet: **statisk – dynamisk**

De fleste systemer man studerer forandrer seg over tid. En *statisk* simuleringsmodell er en representasjon av et system på et gitt tidspunkt – eller et system der tidsaspektet er fraværende. Et eventuelt tidsforløp er altså uvedkommende for statiske simuleringer. Statiske Monte Carlo-modeller (se kapittel 4 og 2.8) er eksempler på simulering der tidsaspektet er fraværende. I en *dynamisk* simulering følges utviklingen i et system over tid.[7] Langt de fleste OA-modeller for simulering er dynamiske.

Noen miljøer mener at kun dynamiske modeller meritterer betegnelsen ”simulering”, mens statiske modeller bør kalles *beregningsmodeller*.

2.7.2 Usikkerhetshåndtering: **deterministisk – stokastisk**

En *stokastisk* simulering inneholder tilfeldige trekninger ved hjelp av slumptall, mens en *deterministisk* simulering ikke har tilfeldige trekninger. En stokastisk simulering gjennomløper én av virkelighetens mange variasjonsmuligheter, og gir følgelig forskjellige utfall hver gang den utføres. Man må derfor gjøre mange gjentakelser (kalt replikasjoner) for å få et godt bilde av usikkerheten i resultatene, typisk i form av gjennomsnittsverdier og spredningsmål.[8] Dette tar tid, men simuleringen blir også mer naturtro. En deterministisk simulering gir derimot samme svar hver gang, og trenger derfor bare å utføres én gang. En deterministisk simulering baserer seg gjerne på gjennomsnittsverdier som inputdata, noe som undertrykker virkelighetens variasjonsmuligheter. Eventuell usikkerhet må derfor håndteres på annet vis eksternt.

En simuleringsmodell kan ofte være i hovedsak deterministisk med innslag av stokastiske elementer, eller vice versa, og da blir det en smakssak hvilket av de to begrepene som passer best for hybriden. For å være på den sikre siden, tar man som regel den fundamentalistiske tilnærmelsen å kalle en modell stokastisk såfremt den inneholder minst én tilfeldig trekning.

Noen miljøer mener at betegnelsen ”simulering” bør forbeholdes stokastiske modeller, mens man for deterministiske modeller bør bruke betegnelsen *kalkulering*.[9]

2.7.3 Operasjonsmodus: **lukket – åpen**

En *åpen* modell styres manuelt under simuleringen vha. kommandoer, menyer e.l. Dette betyr at simuleringen påvirkes av de beslutninger som brukeren tar underveis. Brukerinteraksjonen kan arte seg på to måter: *interaktive* modeller krever brukerkontroll, mens *avbrytbare* modeller tillater brukerkontroll. Betegnelsen *simulator* brukes ofte om interaktive simuleringsmodeller. En åpen simulering er nødvendig når modellen inneholder menneskelige beslutninger som

---

[7] I denne rapporten brukes begrepet *dynamisk* i betydningen ”med tidsforløp” om problemer, systemer, modeller og simulering. Det gjøres oppmerksom på at innen matematikk/systemteori har et *dynamisk system* en presis betydning som er mer restriktiv enn den som legges til grunn her

[8] Dette kalles også *Monte Carlo-simulering*, se kapittel 2.8

[9] Om man skulle ta både denne meningen og meningen i kapittel 2.7.1 til følge, ville betegnelsen simulering kun anvendes på stokastiske simuleringer med tidsforløp. Svært mange modeller er nettopp slik, men langt fra alle. Bl.a. vil både statisk Monte Carlo-simulering og systemdynamisk simulering (oftest deterministisk) falle utenfor

vanskelig lar seg automatisere. Et vanlig anvendelsesområde for åpne simuleringsmodeller er for trening av beslutningstakere på ulike nivåer i en militær organisasjon.

I en *lukket* modell går simuleringen fra start til slutt uten avbrudd, og brukeren må avvente resultatene til slutt. All informasjonen som trengs for simuleringen må være matet inn på forhånd. Fordelen med en lukket simulering er at den kan kjøres raskere og tillater dermed flere replikasjoner. Lukkede modeller krever automatisering av menneskelig beslutningsfatning, og teknikker for kunstig intelligens står gjerne sentralt. Et prinsipielt spørsmål i denne sammenheng er om man skal forsøke å representere menneskelig oppførsel så naturtro som mulig, dvs. hvordan et menneske faktisk vil reagere i forskjellige situasjoner. Eller om man skal benytte mer normative regler for oppførsel, dvs. at mennesket handler rasjonelt iht. fastsatte mål. Den første tilnærmingen er klart den mest utfordrende og stiller store krav til modellering av menneskelig beslutningsfatning.

2.7.4 Oppløsning: **finkornet – grov**

Detaljeringsnivået på en modell bør stå i forhold til problemet man ønsker å studere. *Finkornede* modeller beskriver mange av virkelighetens detaljer, og fordelen med dette er at man oppnår en relativt realistisk simulering. Ulempen er at det er ressurskrevende å modellere et system i alle dets detaljer. Dessuten gir økt størrelse og kompleksitet større sjanse for feil underveis i modelleringen eller implementeringen av modellen. Motsatt vil *grovkornede* modeller i stor grad forenkle og abstrahere virkeligheten. Dermed er det en fare for at vesentlige faktorer og sammenhenger utelates. Fordelen med grove modeller er at de er lette og raske å utvikle og bruke. Hvilken plass en modell skal finne i spennet mellom fin og grov, er avhengig av kompleksiteten til problemet som skal løses.

2.7.5 Kompleksitet: **statistisk – sammensatt**

Ved modellering av komplekse systemer lager man gjerne mindre, finkornede delmodeller som hver tar for seg for snevrere problemer. Løsningene fra delmodellene mates inn i en overordnet, grovere modell. På denne måten dannes et hierarki av modeller som fanger opp de fleste faktorer og sammenhenger som finnes i virkeligheten, uten at dette resulterer i én enkeltstående, stor og kompleks modell. En slik modultenkning gir bedre modelloversikt, og åpner dessuten for effektiv gjenbruk av modeller.

Slike aggregerte – eller *sammensatte* – modeller har altså mange parametere som estimeres fra forskjellige kilder. Output fra modellen kan brukes til validering (se kapittel 2.9.4), men ikke til parameterestimering i delmodellene. Delmodellene i sammensatte modeller er som nevnt mindre og mer rettet mot spesifikke delproblemer. Slike delmodeller kjennetegnes ofte ved at de har relativt få parametere som kan estimeres fra empiriske data, og kalles av den grunn gjerne *statistiske* modeller.[10]

[10] Bruken av ordet ”statistisk” kan innby til forvirring. Poenget her er at de enklere delmodellene kalles statistiske i egenskap av å være *parametriserbare* (i form av kjente sannsynlighetsfordelinger, regresjonsmodeller m.m.). Aggregerte modeller kan selvfølgelig også anses som statistiske i betydning *stokastiske* (hvis de er det)

Et eksempel på en statistisk modell kan være en regresjonsmodell for sammenhengen mellom parametrene flytimer med jagerfly og reparasjonsbehov. Et eksempel på en tilhørende aggregert modell kan være en modell for tilgjengelighet av jagerfly i en internasjonal operasjon. Her vil delmodellen for reparasjonsbehov være en av mange "leverandører" til den aggregerte modellen for beregning av tilgjengelighet på jagerfly.[11]

2.7.6 Tilstandsendring: **diskret – kontinuerlig**

For dynamisk simulering, altså simulering med et tidsforløp, har man to hovedtyper simulering. I en *diskret* simulering endrer systemet tilstand kun på diskrete tidspunkt på tidsaksen, og man ser på det som eventuelt skjer mellom disse hendelsene som irrelevant. En *kø* er et eksempel på et diskret system, hvor man kun er interessert i makronivåhendelser som at et element ankommer systemet, behandling starter og behandling slutter. Elementenes liv mellom disse hendelsene er uinteressant, og har ingen innvirkning på andre systemvariabler. Tilstanden til systemet beskrives av variabler som antall elementer i kø, antall til behandling, ledig behandlingskapasitet m.m.

I en *kontinuerlig* simulering endres systemets tilstand kontinuerlig med tiden. Slik simulering brukes når systemets oppførsel under hele simuleringens tidsforløp er interessant. I kontinuerlige modeller er endringene som regel beskrevet med differensialligninger. Mange prosesser av ulik art modelleres gjerne på denne måten. Fysiske strømningsprosesser (varme, luft, væske o.l.) må opplagt modelleres kontinuerlig; det ville vært meningsløst (og umulig) å f.eks. følge enkeltmolekyler i en væskestrøm (vha. diskret modellering) når det man er interessert i kan beskrives av differensiallikninger for endringer på makronivå.

Systemer som er kontinuerlige av natur, kan ofte med fordel modelleres diskret. Et eksempel kan være et system med en beholder som fylles med vann med en viss fyllingsrate. Dersom den løpende vannstanden i beholderen er noe man er interessert i, og noe som andre systemvariabler er avhengig av, må systemet modelleres kontinuerlig og tilstanden oppdateres hyppig med små tidsskritt. Hvis man derimot kun er interessert i å vite når beholderen er full, og vannstanden underveis er irrelevant for andre systemvariabler, kan man modellere systemet diskret. Dette innebærer at man regner ut når beholderen vil være full, og hopper direkte til dette tidspunktet i simuleringen. En forutsetning er selvsagt at dette tidspunktet lar seg beregne. Dersom fyllingsraten er variabel og komplekst avhengig av andre systemvariabler, må man kanskje likevel modellere kontinuerlig og hele tiden teste for om beholderen er full. Det innebærer altså en abstraksjon, en bortluking av detaljer, å lage en diskret modell av et kontinuerlig system på denne måten.

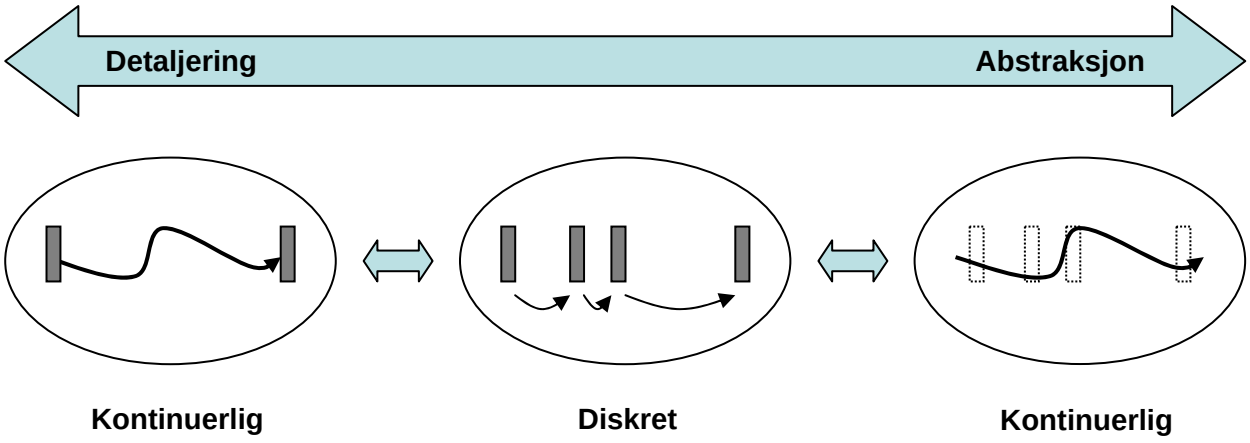
Motsatt kan systemer som er diskrete av natur, ofte med fordel modelleres kontinuerlig. Dette gjelder spesielt når systemet blir stort med svært mange elementer. Et eksempel kan være en kassakø i en butikk. Dersom hver enkelt kundes ankomst, behandling og avgang er interessant for å finne f.eks. en fordeling over ventetider, må systemet modelleres diskret. Men det er gjerne

[11] Dette eksempelet er ikke tatt helt ut av lufta, men er en relevant problemstilling som FFI nylig har behandlet med modellen FLYT2 (se f.eks. (15))

tilstrekkelig og hensiktsmessig å studere *flyten* av kunder på makronivå, dvs. ankomst- og behandlings*rater* o.l., for å kunne svare på overordnede spørsmål om betjeningen. Slike rater tilsvarer gjerne gjennomsnittsverdier fra en tilsvarende diskret simulering, og man ser dermed bort fra variansen som resulterer fra en diskret simulering. Det innebærer altså en abstraksjon å lage en kontinuerlig modell av et diskret system på denne måten.

Moralen blir at en modell må ha et detaljeringsnivå tilpasset problemet man ønsker å studere, og at problemanskuelsen dikterer hvilken modelleringsform – diskret eller kontinuerlig – som er best egnet.

Videre kan en simuleringsmodell gjerne være en hybrid mellom diskret og kontinuerlig i den forstand at tidsaksen deles opp i diskrete og kontinuerlige partier. Overganger fra det ene regimet til det andre kommer som regel av at man vil studere deler av forløpet i større grad av detalj. Som diskutert over, er det ikke i utgangspunktet gitt om det er diskret eller kontinuerlig modellering som tillater størst grad av detaljering. Om man i utgangspunktet ser på kontinuerlig flyt av mange elementer, vil det være de diskrete partiene som tillater detaljstudier, siden man kan gå ned til å følge enkeltelementer. Men diskret følging av enkeltelementer kan også detaljeres ytterligere ved å følge kontinuerlig forløpet mellom tilstandsendringer (figur 2.3).



Figur 2.3 *Diskrete systemer kan modelleres kontinuerlig og vice versa, avhengig av problemanskuelse og ønsket detaljeringsgrad. En simuleringsmodell kan ha både diskrete og kontinuerlige partier*

2.7.7 Tidshåndtering: **hendelsesstyrt – tidsstyrt**

For datamaskinstøttet, dynamisk simulering opererer man også med begrepene hendelsesstyrt og tidsstyrt. I *hendelsesstyrte* simuleringer avanserer simuleringsklokka fra en (diskret) hendelse til den neste, og hopper over tiden i mellom. Diskrete modeller er som oftest hendelsesstyrte, og man snakker om *diskret hendelsesstyrt simulering* (DHS). DHS er et av hovedområdene i denne rapporten, og omtales i kapittel 6.

I *tidsstyrte* simuleringer avanserer simuleringsklokka med passende små tidsskritt. Lengden på

tidsskrittene bør tilpasses systemets tidskonstanter. I praksis er imidlertid tidsskrittene som regel av fast lengde. Tidsstyring er nødvendig for kontinuerlige modeller, siden tilstanden hele tiden er i endring. Kontinuerlige modeller er i realiteten ”kvasi-kontinuerlige”, siden bruk av data-maskin krever diskretisering av tidsaksen og numerisk løsning av differensialligninger. Dette reduserer presisjonen i løsningen, noe man må kompensere for ved å ha et så lite tidsskritt som mulig. Ved fastsettelse av lengden på tidsskrittet må kravet til nøyaktighet avveies mot krav til kjøretid, siden små tidsskritt betyr hyppige oppdateringer og beregninger, noe som øker kjøretiden.

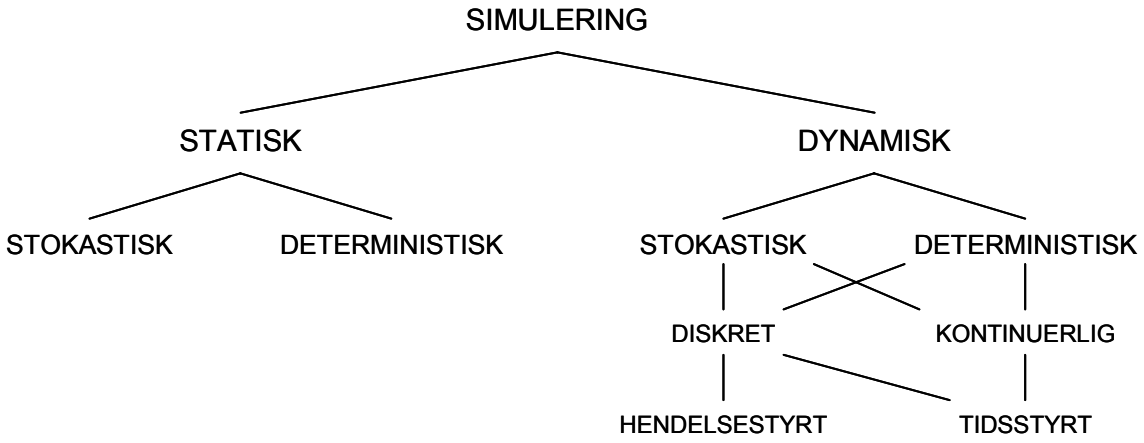
Tidsstyring kan også benyttes i forbindelse med diskrete modeller, men dette er mindre vanlig. Simuleringen hopper da fra et tidspunkt til det neste i passende små steg, og undersøker om det har skjedd endringer i systemet siden forrige tidspunkt. Hvis så er tilfelle, oppdaterer man systemtilstanden før man går til neste tidspunkt; hvis ikke forblir systemtilstanden uendret. Dette gjør det mulig å kombinere tids- og hendelsesstyrtsimulering i en modell. En diskret hendelse kan medføre en endring i en kontinuerlig tilstandsvariabel eller en kontinuerlig tilstandsvariabel kan nå en terskelverdi som kan føre til at en hendelse oppstår.

2.7.8 Andre dimensjoner

I tillegg til dimensjonene som er omtalt hittil, kan man velge å beskrive simuleringsmodeller langs en rekke andre dimensjoner man finner hensiktsmessig, eksempelvis datamaskinstøttet eller ikke, innslag av kunstig intelligens, høy eller lav brukerterskel, osv.

Som indikert flere steder, trenger ikke en modell/simulering være rendyrket det ene eller det andre langs de angitte dimensjonene. Kombinasjoner kan tenkes, og forekommer ofte i praksis.

Det er vanskelig å fremstille visuelt det mangedimensjonale rommet som spennes ut av simuleringsdimensjonene beskrevet i dette kapittelet. Hierarkiet i figur 2.4 tar for seg de kanskje mest sentrale dimensjonene i vår sammenheng – dimensjonene som bidrar mest til å beskrive og skille ad de fire ulike simuleringsmetodene som diskuteres spesielt i denne rapporten. Dette er én av mange mulige måter å grovinndele og skape en viss grad av orden i begrepene på.



Figur 2.4 Klassifisering av simuleringsmetoder. Dette er én av mange mulige måter å strukturere de viktigste begrepene knyttet til simulering

2.8 Monte Carlo-simulering

Stokastisk simulering er som nevnt simulering som inneholder tilfeldige trekninger, og som derfor gir forskjellige resultater hver gang den utføres. Informasjonen fra en enkeltstående kjøring av en stokastisk simulering er usikker, så i reelle eksperimenter utføres simuleringen mange ganger for å få dannet et godt nok bilde av usikkerheten i resultatene. En populær benevnelse på slik repetitiv, stokastisk simulering er *Monte Carlo (MC)-simulering*. Resultater fra en MC-simulering er typisk gjennomsnittsverdier og spredningsmål for variablene man studerer.

Den odde betegnelsen *Monte Carlo* for denne typen simulering henspiller på det berømte kasinoet i Monaco og dets kjennetegn, nemlig tilfeldigheter og repetisjoner. MC-betegnelsen ble først tatt i bruk av de moderne pionerene innen feltet på 30-tallet for å hedre en spilleglad onkel. Den mest kjente tidlige anvendelsen av MC-simulering var under Manhattan-prosjektet i USA under Den andre verdenskrig. Både da og senere har MC-simulering tradisjonelt vært forbundet med *statiske* problemer, dvs. problemer uten tidsforløp (se kapittel 2.7.1). En mer generell betegnelse for MC-simulering er derfor *statistisk sampling*.

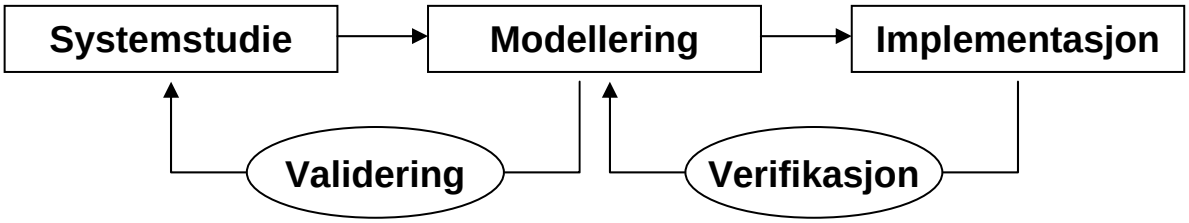
I tråd med historien mener mange miljøer at betegnelsen *Monte Carlo* skal forbeholdes statiske modeller, mens dynamiske modeller bør gis andre navn. I dag brukes derimot MC-betegnelsen i praksis om all stokastisk simulering, både med og uten tidsforløp. Dette gjelder spesielt for simulering i OA-sammenheng. I prinsippet kan derfor MC-begrepet komme til anvendelse for alle hovedklassene simulering som beskrives i denne rapporten.

Statisk MC-simulering, dvs. MC-simulering i klassisk forstand, er et av hovedsimuleringsområdene beskrevet i denne rapporten (se kapittel 4). Det er interessant å merke seg at klassisk MC-simulering er en stokastisk tilnærming til å løse *deterministiske* problemer – eksempler på dette gis i kapittel 4.

2.9 Utvikling av simuleringsmodeller

Utviklingen av en simuleringsmodell vil i grove trekk gjennomløpe fire faser, som i praksis inngår i en iterativ prosess (figur 2.5):

1. Systemstudie
2. Modellutvikling
3. Implementasjon
4. Verifikasjon og validering



Figur 2.5 Faser i utviklingen av en simuleringsmodell[12]

Simuleringsprosessen som helhet og som metode for problemløsning er mye videre enn dette, og følger i prinsipp OA-arbeidsprosessen (figur 2.1). *Før* de angitte utviklingstrinnene må man selvsagt ha en operasjonell problemformulering for å kunne gjennomføre systemstudien. *Etter* disse trinnene kommer aktiviteter som eksekvering av modellen, analyse av resultater og eventuelt anbefalinger. Vi begrenser oss i dette kapitlet til selve modellutviklingen, og de fire trinnene beskrives nærmere i det følgende.

2.9.1 Systemstudie

Begrepet *system* ble beskrevet i kapittel 2.2, og er kort fortalt den delen av virkeligheten som angår problemet man skal løse. Alle faktorer og forhold som påvirker systemets oppførsel bør være med i systembeskrivelsen. Dette krever inngående kjennskap til og forståelse av sammenhengene i systemet. Videre kan et system alltid ses på som et delsystem i et større system. Det er analytikerens (eller oppdragsgivers) perspektiv som avgjør hvilke av virkelighetens aspekter som er relevante eller ikke for problemet som skal løses. Systemstudien skal definere grensene mellom systemet og resten av virkeligheten (*systemgrensene*). De komponenter som ligger utenfor det spesifiserte systemet, men likevel har en relasjon til systemet, kalles systemets *omgivelser* eller *miljø*.

2.9.2 Modellering

Konseptet *modell* ble generelt beskrevet i kapittel 2.3, og er kort fortalt en forenklet representasjon av et system. Simulering krever en formell (matematisk) modell, der systemets

[12] Beslutningen om å velge simulering eller ikke som løsningsmetode bør i prinsippet tas *etter* at systemstudien og modelleringen er gjennomført. Først da vil det vise seg om problemet er enklere enn antatt og kan løses analytisk (eller motsatt). I praksis er det likevel gjerne slik at løsningsmetode bestemmes tidlig i prosessen, og at modellen utformes med tanke på valgt metode

struktur og oppførsel beskrives av komponenter, variabler, parametere og funksjonelle sammenhenger.

Komponenter (eller *elementer*) representerer systemets bestanddeler. Komponentene har målbare egenskaper som beskrives vha. variabler og parametere. *Variabler* kan endres av modellen og dermed anta ulike verdier i løpet av en simulering, mens *parametere* er størrelser som spesifiseres før simuleringsstart og som ikke endres under simuleringen. Variablenes verdi på et gitt tidspunkt under simuleringen beskriver systemets *tilstand* på dette tidspunktet. *Funksjonelle sammenhenger* beskriver hvordan komponentene i modellen påvirker hverandre. Sammenhengene er enten *deterministiske* der tilstandsovergangene er entydige, eller *stokastiske* der en gitt tilstand kan lede til flere alternative tilstander som følge av usikkerhet.

De ulike komponentene i systemet modelleres som regel hver for seg, for så å kombineres til en helhet. Ressursrammene definerer hvor mye arbeid man kan legge ned i modelleringen av komponentene, men det vil uansett alltid være nødvendig med en større eller mindre grad av forenkling. Det er gjerne fornuftig med en grov modellering av de delene av systemet som er minst betydningsfulle, og en finere modellering av de viktigste komponentene. Imidlertid er det et godt prinsipp å balansere detaljeringsgraden i de ulike delmodellene; det er uhensiktsmessig å modellere noen komponenter i stor detalj dersom man samtidig modellerer grovt andre komponenter *som er like viktige*.

Det finnes ingen standard oppskrift for hvordan en modell skal utformes. Det er en utfordring å bestemme hvilke deler og sammenhenger som skal være med, og hvilke som kan utelates uten at resultatene blir merkbart dårligere. Et system kan derfor gi opphav til mange mer eller mindre gode modeller. Modellering bygger på kunnskap, ideer og intuisjon opparbeidet under systemstudien, samt kunnskap om modellbygging generelt. Dette arbeidet er like mye en kunst som et håndverk.

Modellering innebærer også å fremskaffe inputdata for variabler og sammenhenger i modellen. Gode inputdata er avgjørende for troverdigheten til resultatene fra modellen. Det kan være en utfordring både å *finne* data og å *tolke* data inn i et format modellen kan arbeide med. En modell bør ikke utformes på et detaljeringsnivå finere enn det tilgangen på data realistisk sett åpner opp for. Kapittel 2.10.1 nedenfor diskuterer inputmodellering fra en mer praktisk synsvinkel.

Modelleringens betydning kan knapt overvurderes. En god modell med gode inputdata er avgjørende for å finne gode og troverdige løsninger på et problem gjennom simulering.

2.9.3 Implementasjon

Med *implementasjonen* av en modell mener vi et dataprogram som realiserer de formelle sammenhengene i modellen. Dette krever programmeringskunnskap, men gitt en grundig modelleringsjobb på forhånd, er implementasjonen mest en teknisk øvelse. Programmeringen kan likevel være krevende; simuleringsmodeller gir ofte opphav til store og kompliserte dataprogrammer, og det er lett å miste oversikten og trå feil under implementeringen. Et

gjennomtenkt design, strukturert koding og god dokumentasjon er viktig for å håndtere denne kompleksiteten. I praksis utvikles gjerne modellen og implementasjonen hånd i hånd, og disse to oppgavene glir dermed over i hverandre.

Som nevnt tidligere omtales implementasjonen av modellen for enkelhets skyld som ”modellen”, men konseptuelt er modellen og dens implementasjon to forskjellige ting, og bør holdes atskilt.

2.9.4 Verifikasjon og validering

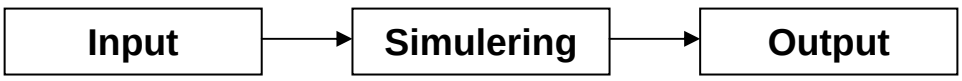
Når implementeringen er ferdig, må simuleringsmodellen gjennomgå verifikasjon og validering (figur 2.5). *Verifikasjon* innebærer å forsikre seg om at implementasjonen av modellen er korrekt, dvs. at det er overensstemmelse mellom modellen og dens representasjon (data-programmet). Det oppstår lett både tastefeil, språkfeil og logiske feil under programmering av store modeller, og mye tid går vanligvis med til feilsøking. Det er også vanlig å teste programmet med varierende inputdata for å studere oppførselen i ulike settinger.

Validering innebærer å forvisse seg om at modellen faktisk gir en rimelig beskrivelse av virkeligheten, dvs. at det er overensstemmelse mellom systemet og dets representasjon (modellen). Validering er i prinsippet uavhengig av dataprogrammet, og bør i størst mulig grad gjennomføres før programmeringen starter. I praksis er det vanskelig å teste en modells validitet før den er realisert og kan kjøres på en datamaskin.

I praktisk bruk av simulering er det viktig å oppnå troverdighet til modellen. Troverdighet vil avhenge av en subjektiv oppfattelse av modellens korrekthet (verifikasjon) og anvendbarhet (validering). Verifisering og validering bør derfor etterstrebes i alle deler av utviklingsprosessen.

2.10 Gjennomføring av simulering

Selve gjennomføringen av en simulering kan deles inn i tre faser: inputmodellering, kjøring av modellen og analyse av resultater (figur 2.6). Teksten i dette kapitlet er mest relevant for stokastisk simulering.



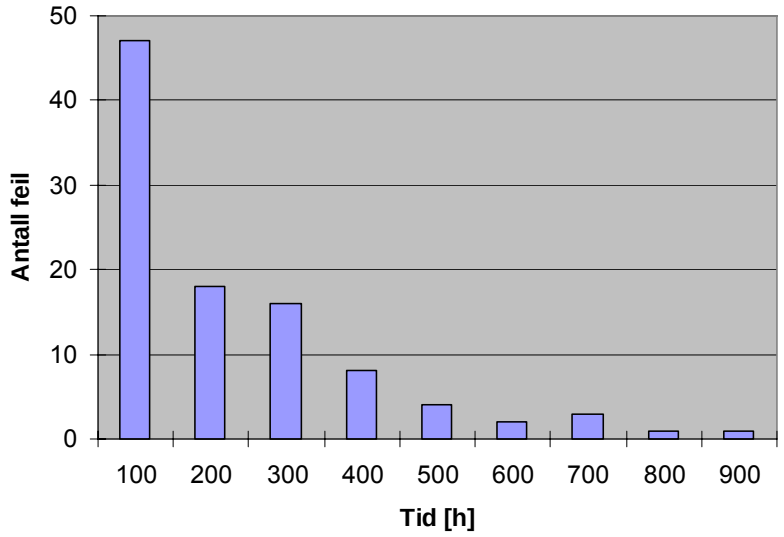
Figur 2.6 Tre faser i gjennomføringen av en simulering: inputmodellering, kjøring av modellen og analyse av resultater

2.10.1 Inputmodellering

En vesentlig egenskap ved simuleringer er at styrken til resultatene er direkte avhengig av hvor gode inngangsdataene til simuleringen er (”garbage in = garbage out”). Dette gjelder spesielt for stokastiske modeller, der utfordringen ligger i å beskrive usikkerheten i inngangsverdiene på en god nok måte. I Banks et al. (11) beskrives ”Input Modeling” i kapittel 9 over ca. 40 sider. Law

& Kelton (4) har en fyldigere fremstilling på drøyt 100 sider i kapittel 6 kalt ”Selecting Input Probability Distributions”. Eksempelet nedenfor belyser litt av det som omhandles.

I scenariobeskrivelsen i appendiks B spesifiseres det at tiden til neste feil inntreffer for en lastebil er eksponentialfordelt. Bakgrunnen for en slik spesifikasjon vil variere. For modelleringen vil det beste være om man i en periode har observert bruken av mange lastebiler i det aktuelle terrenget eller i et tilsvarende terreng, og for hvert kjøretøy notert tiden mellom hver feil. Disse observasjonene vil det være vanlig å fremstille i et histogram. Nedenfor er det vist et slikt histogram basert på 100 observasjoner.



Figur 2.7 *Fordeling av kjøretid i timer mellom feilforekomster for lastebil. Figuren er basert på 100 observasjoner. Tallene på x-aksen representerer intervaller: Tallet ”100” betyr intervallet 0–100; ”200” betyr 100–200; etc.*

Første trinn i inputmodelleringen består i å foreslå en type sannsynlighetsfordeling for å representere de aktuelle observasjonene. Det finnes gode metoder for å bedømme samsvaret mellom en empirisk fordeling og ulike parametriske fordelinger (se ”trinn tre” nedenfor). Av figur 2.7 går det klart fram at kjøretiden mellom feilforekomster ikke er symmetrisk fordelt. Formen på histogrammet tilsier at det kan være naturlig å prøve med en eksponentialfordeling for å representere observasjonene.

Trinn to i inputmodelleringen består i å estimere parameterne i den valgte fordelingen. Eksponentialfordelingen har bare én parameter, feilraten λ (lambda). Sannsynlighetstettheten er $f(x;\lambda)= \lambda e^{-\lambda x}$, der $x \geq 0$ og $\lambda>0$, og forventningsverdien er lik $1/\lambda$. Den vanlige estimatoren for $1/\lambda$ er gjennomsnittet av observasjonene. I vårt eksempel er gjennomsnittlig ventetid til neste feil inntreffer 206 timer, og estimatoren for feilraten λ blir da 1/206 feil per time.

Trinn tre i inputmodelleringen består i å vurdere om eksponentialfordelingen med estimert parameter representerer kjøretid mellom feilforekomster på en god nok måte. Dette gjøres ofte ved å gjennomføre en kjikvadratføyningstest. Dette er beskrevet i Law & Kelton (4) på sidene 356–363. En alternativ test for å vurdere en fordelings egnethet til å representere et datasett, er

Kolmogorov-Smirnov tester. Law & Kelton (4) nevner på side 370 software som blant et antall kandidater plukker ut den fordelingen som best representerer et aktuelt datasett. Det er imidlertid viktig å utvise kritisk sans ved valg av fordeling, og ikke automatisk gå for den som samsvarer best. Som regel innehar man noe kunnskap om fenomenet som modelleres, og ved å inkludere denne i vurderingen, kan man ofte utelukke eller anbefale enkelte fordelinger.

Hvis man bruker simulering for å vurdere alternative spesifikasjoner for et system, vil man ofte mangle data. Law & Kelton (4) nevner på side 386 et par heuristiske prosedyrer som kan brukes ved valg av fordelinger i slike situasjoner. Trekantfordelingen er mye brukt, også i casen som beskrives i denne rapporten.

2.10.2 Kjøring av simulering

Stokastisk simulering gir resultater med usikkerhet; man ønsker gjerne en viss nøyaktighet i resultatene, og denne nøyaktigheten øker med antall replikasjoner. Ved å planlegge analysen på forhånd, vil man ofte kunne spare seg mange replikasjoner. En annen effekt kan være at man slipper å starte opp igjen simuleringene i analysefasen. Uten en gjennomtenkt analyseplan kan man i analysefasen oppdage at man har gjennomført for få simuleringer til å kunne konkludere.

For å illustrere tankegangen kan man se på et problem fra utpostscenariet (kapittel 3 og appendiks B). Med de gitte betingelsene ønsker en å estimere sannsynligheten p for at utposten skal gå tom for forsyninger i løpet av en bestemt periode. En ønsker en viss nøyaktighet for estimatoren $\hat{p}$. Dette uttrykkes ofte ved et krav om at $P\left(\left|\hat{p}-p\right|>\delta\right)\leq\varepsilon.$ Vanlige valg for ε er 0,005, 0,01, 0,05 eller 0,1. Spørsmålet er hvor mange replikasjoner n vi trenger for å oppnå denne nøyaktigheten for ulike verdier av δ og ε. Hvis Y er antall replikasjoner av totalt n hvor utposten går tom for forsyninger, er $\hat{p}=\frac{Y}{n}$ en forventningsrett estimator for p. Y er binomisk fordelt, og med en tilnærming til normalfordelingen kan det vises at nøyaktighetskravet er oppfylt hvis n velges slik at $n\geq\frac{z_{\frac{\varepsilon}{2}}^{\ 2}}{4\delta^2}$, der $z_{\frac{\varepsilon}{2}}$ er øvre $\frac{\varepsilon}{2}$-fraktil i standard normalfordelingen.[13] Velges δ=0,05 og ε=0,05 blir $n\geq\frac{1,96^2}{4\left(0,05\right)^2}=384,16$, altså kreves 385 replikasjoner. Hvis man har sikker forhåndsinformasjon om sannsynligheten p – eksempelvis at p < 0,2 – trenger man færre replikasjoner for å oppnå den ønskede nøyaktighet.

I kapittel 10 i Law & Kelton (4) drøftes opplegg for sammenligning og rangering av to eller flere systemer. I kapittel 10 er de aktuelle systemalternativene spesifisert på forhånd, mens man i kapittel 12 studerer situasjoner hvor man ønsker å finne ut hvordan variasjonen i ulike parametere påvirker ytelsen til systemet som studeres.

[13] Se hvilken som helst grunnleggende lærebok i statistikk, f.eks. (16)

2.10.3 Vurdering av output

Resultater fra en simulering må vurderes nøye av flere årsaker. For det første er det viktig å ikke glemme at simuleringsresultatene baserer seg på en forenklet modell av virkeligheten, og at det som regel er tatt mange og til dels grove antakelser for å få etablert denne modellen. Dette betyr at man aldri helt kan stole 100 % på resultatene man får ut – de er omtrentlige, men kan likevel være nyttige (”All models are wrong. Some are useful”).

For det andre er resultatene kun estimater som er beheftet med usikkerhet. Det er derfor viktig å gjennomføre sensitivitetsanalyser i etterkant for å anslå følsomheten i de endelige svarene. Systematisk variasjon av parametere viser hvor robuste resultatene er overfor små og store endringer i initialbetingelsene. Dersom små endringer i enkelte parametere gir store utslag, kan det være lurt å modellere disse delene av systemet ekstra nøyaktig.

For det tredje er resultatene kun gyldige i det scenariet som er lagt til grunn, og man kan få helt andre svar når man bruker andre scenarier. Det er derfor viktig å forsikre seg om at scenariet er tilstrekkelig representativt, eller være bevisst den potensielle begrensningen som ligger i det å kun bruke ett scenario. Det beste er å gjennomføre simulering over et bredt spekter av scenarier, og dra konklusjoner om systemets ytelse basert på en helhetsvurdering av enkeltresultatene.

For det fjerde er det gjerne slik at det finnes overordnede effektivitetsmål i analyseprosessen som ikke alltid kan knyttes direkte til simuleringens resultater. Simuleringen tar seg av sin begrensede bit av en større oppgave, og resultatene fra simuleringsstudien er kun innspill til den overordnede prosessen. Dette gjelder for så vidt all OA for beslutningsstøtte – det er viktig å være klar over at OA-resultater alltid kun vil være et beslutnings*grunnlag*, og sjelden i seg selv diktere beslutningen.

3 SCENARIO: UTPOST

For å kunne belyse de ulike simuleringsmetodene og -teknikkene med eksempler, er det skissert et felles, generisk scenario. Scenariet tjener som rammeverk for eksempler på problemstillinger som de ulike metodene er egnet til å løse. En ytterligere gevinst ved scenariet er at det har gitt forfatterne verdifull praktisk innsikt i metodene som beskrives, og kjennskap til flere ulike verktøy for simulering. Resultater fra analyser av scenariet oppnådd med de ulike verktøyene beskrives ikke i denne rapporten. Erfaringer med dette scenariet kan likevel brukes som grunnlag for en grov sammenligning av metodenes styrker, svakheter og mulige anvendelses-områder.

Nedenfor introduseres scenariet kort og overordnet. Eksempler med utgangspunkt i dette scenariet finnes under beskrivelsene av de enkelte simuleringsmetodene i kapitlene 4–7. En mer detaljert scenariobeskrivelse finnes i appendiks B. Denne er primært utviklet med tanke på diskret modellering, og inneholder en god del mer informasjon enn nødvendig for mange av eksemplene. Videre vil mange av eksemplene også gjøre antakelser om forhold som *ikke* står i

den detaljerte beskrivelsen, der dette er opportunt for poengene som eksemplene skal illustrere.

3.1 Overordnet scenariobeskrivelse

Norge stiller med militære styrker i en internasjonal operasjon i et land med mangelfull infrastruktur. Operasjonen har FN-mandat og er fredsbevarende. Situasjonen i landet er for tiden stort sett rolig, men den er spent og ustabil etter mange års krig. FN-styrken skal støtte landets myndigheter i arbeidet med å bygge opp en sivil administrasjon og infrastruktur.

Total FN-styrke er på ca. 10 000 mann og Norge stiller med ca. 400. Den internasjonale styrken har en hovedbase ved landets hovedstad. Det er ellers utplassert mindre styrker 10 steder spredt rundt om i landet. De norske styrkene befinner seg på hovedbasen og på en av utpostene. Utposten må få sine forsyninger fra hovedbasen. Avstanden mellom hovedbasen og utposten er ca. 500 km.

De disponible helikoptrene må ha en etterfylling av drivstoff underveis for å nå fra hovedbasen til utposten med vanlig nyttelast. Med sterkt redusert nyttelast er det mulig å nå fram uten etterfylling av drivstoff. Flytiden er mellom to og tre timer.

Veiene er stedvis svært dårlige. I kortere perioder av året er det i praksis umulige å kjøre mellom hovedbasen og utposten med de disponible kjøretøyene. Det er særlig veiene lengst fra hovedbasen som stenges. Under vanlige forhold tar kjøreturen omtrent to døgn. Transportene til utposten må eskorteres av pansrede personellkjøretøy eller stormpanservogner. Angrep mot forsyningskolonnen forekommer.

4 STATISK MONTE CARLO-SIMULERING

Som nevnt tidligere er *Monte Carlo* (MC) simulering en repetitiv, stokastisk simulering som gir sannsynlighetsfordelinger over resultater for størrelsene man studerer (kapittel 2.8). *Statisk* simulering er simulering der tidsaspektet er fraværende eller underordnet (kapittel 2.7.1). MC-simulering i klassisk forstand befattet seg nettopp med stasjonære, stokastiske prosesser, og dette kapittelet ser nærmere på denne simuleringsmetoden. Det er underforstått i teksten at ”MC” i utgangspunktet begrenser seg til *statisk* MC. To referanseverk innen MC-metoden er (17), (18).

4.1 Aggregering av usikkerhet

En MC-simulering tar eksplisitt hensyn til usikkerheten i inngangsvariablene i modellen. Usikkerheten representeres ved sannsynlighetsfordelinger, og er dermed å regne som ”kjent usikkerhet”.[14] Når sannsynlighetsfordelingene er kjent, kan man i noen tilfeller finne analytiske løsninger for størrelser man er interessert i, men dette blir fort uhensiktsmessig – og ofte umulig

[14] Noen miljøer bruker betegnelsen *risiko* på kjent usikkerhet (og enda flere miljøer bruker risiko kun om nedsiden av kjent usikkerhet (oppsiden kalles da *muligheter* e.l.))

– med mange variabler og økende kompleksitet. Også simuleringsresultatene vil foreligge i form av sannsynlighetsfordelinger. Fra disse fordelingene rapporteres typisk gjennomsnitts-verdier og spredningsmål.

Et enkelt eksempel på det ovenstående kan være som følger: Noen varer skal kjøpes inn; alle varene har ukjent pris, men sannsynlighetsfordelingene for prisene er kjent (og gjerne forskjellige). Sluttsummen for kjøpet blir da også ukjent, men vil ha en sannsynlighetsfordeling som kan regnes ut på bakgrunn av varenes kjente prisfordelinger. Denne summen vil ha en forventningsverdi lik summen av varenes forventningsverdier og en varians lik summen av varenes varians.[15] Usikkerheten i summen er alltid større enn usikkerheten i enkeltprisene, og en kan si at man i svaret aggregerer usikkerheten i inngangsvariablene.

I dette enkle eksempelet har de få variablene lik måleenhet (penger) og en enkel aggregerings-funksjon (summering), noe som gjør problemet oversiktlig og løsningen intuitiv. I mer praktiske tilfeller har man gjerne mange variabler med ulike måleenheter og en komplisert aggregerings-funksjon, og gjerne også avhengigheter mellom inngangsvariablene. Dette gjør det vanskeligere å forutse resultatfordelingen, men utgjør ikke noe problem for simuleringen, som takler komplekse avhengighetsforhold mellom inngangsvariabler svært godt (forutsatt at simultan-fordelingen for de avhengige variablene er kjent).

Et litt mer komplisert eksempel kan være utfallet av en stridssimulering, der en serie tilfeldige trekninger om mange ulike fenomener (treffsannsynligheter, værforhold, soldatmoral etc.) over et tidsforløp gir et sluttresultat som kan være f.eks. tap av liv eller materiell. Dette er dog et *dynamisk* problem og dermed utenfor fokus for dette kapittelet, men det illustrerer noe av bredden i bruken av MC-simulering.

Eksemplene ovenfor viser MC-simulering brukt til å løse ”stokastiske problemer”, dvs. problemer der svaret er en fordeling av mulige resultater. MC-simulering er en stokastisk metode, men kan også brukes til å løse *deterministiske* problemer. Dette kan høres paradoksalt ut, men faktum er at ”klassisk” MC-simulering var viet nettopp deterministiske problemer. Eksempler på deterministiske problemer er numerisk beregning av (vanskelige) integraler og estimering av tallet pi – oppgaver som opplagt har en bestemt løsning. Eksempler på slike anvendelser er gitt i kapittel 4.3.1.

4.2 Generering av tilfeldig utvalg

En nødvendig forutsetning for MC-simuleringer – og for alle stokastiske simuleringer – er tilgang på tilfeldige tallverdier (slumptall, eng: random numbers). I gamle dager hentet man slumptall fra ferdigproduserte tabellverk, mens de i dag genereres av datamaskiner. En slumptallgenerator trekker uniformt fordelte tilfeldige tallverdier, typisk i intervallet (0, 1). Disse tallverdiene har ingen interesse i seg selv, men benyttes til å trekke tilfeldige observasjoner fra sannsynlighetsfordelinger for de variablene man er interessert i.

[15] Varians-regnestykket gjelder kun når prisene er statistisk uavhengige, noe som ikke alltid er tilfelle

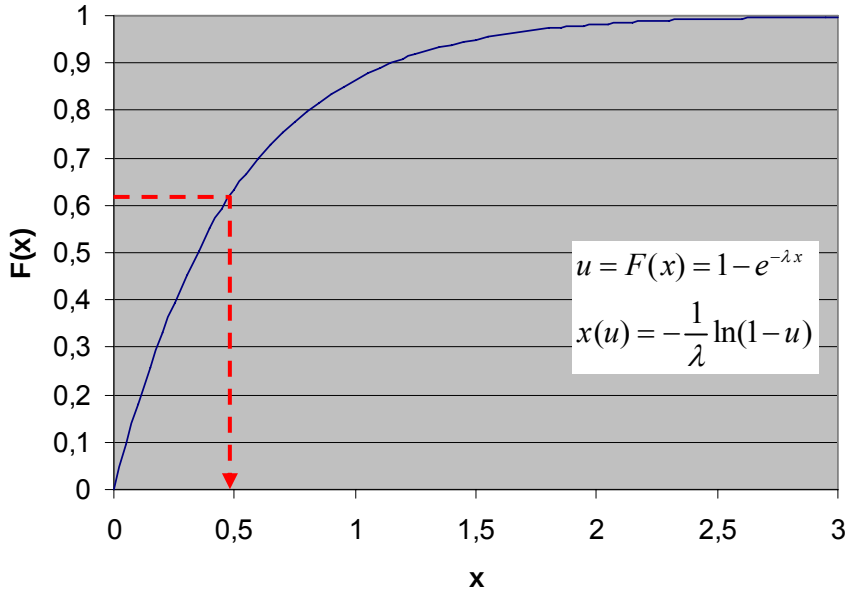
Sannsynlighetsfordelingene kan anta mange former, fra vanlige, parametriske fordelingene som normal-, eksponential- og Poissonfordelingen, til mer uvanlige eller empiriske fordelinger. De fleste programmeringsspråk og simuleringsverktøy har innebygde funksjoner for å trekke uavhengige og identisk fordelte tallverdier/observasjoner fra de mest kjente fordelingene.

Noen sannsynlighetsfordelinger er lette å trekke tilfeldige observasjoner fra mens andre er mer utfordrende. Hvis man kjenner uttrykket for den kumulative sannsynlighetsfordelingen $F(x)$, kan man i noen tilfeller benytte en invers transformasjon i trekningen (19). Nedenfor gis et eksempel på dette for trekninger fra eksponentialfordelingen.

I eksponentialfordelingen er $F(x) = 1 - e^{-\lambda x}$. $F(x)$ er en stokastisk variabel i seg selv (avhengig av x), og F er uniformt fordelt i intervallet (0, 1). Dette gjelder for alle kumulative fordelinger, ikke bare eksponentialfordelingen. Et slumptall u som trekkes uniformt i intervallet (0, 1), kan derfor ses på som et utfall for variabelen F. Man kan deretter regne ut hvilken verdi x fra den underliggende fordelingen (her: eksponentialfordelingen) som ville gitt denne verdien av F. Dette forutsetter at uttrykket for den inverse funksjonen $F^{-1}(u)$ – eller $x(u)$ – kan utledes, noe som er tilfelle for eksponentialfordelingen: $x(u) = -\frac{1}{\lambda}\ln(1-u)$. Utvalgsprosedyren er illustrert i figur 4.1.

Dette eksempelet viste utvalgsprosedyren for en kontinuerlig variabel, men tilsvarende prosedyre med bruk av en invers transformasjon kan også benyttes for diskrete variabler (se f.eks. side 444 i (4)). Imidlertid vil de fleste fordelinger – kontinuerlige og diskrete – være såpass kompliserte at man ikke kan finne den inverse funksjonen $F^{-1}(u)$. For noen av de vanligste fordelingene, som f.eks. normalfordelingen, har det blitt utviklet egne, gode algoritmer for å trekke uavhengige, identisk fordelte tallverdier.[16] For mer ukurante fordelinger vil det normalt være vanskelig å generere tilfeldige utvalg av observasjoner. For dette formål finnes det en rekke metoder som kan benyttes. Den mest vanlige metoden er ”Rejection sampling” som benytter en enkel “testfordeling” som grunnlag for utvalgsprosessen.[17] En annen kraftig metode er Markov Chain Monte Carlo (MCMC) og Metropolis-algoritmen (20), som kan benyttes til å generere tilfeldige utvalg fra stort sett alle typer fordelinger uavhengig av kompleksitet og dimensjonalitet.[18]

[16] Se f.eks. kapittel 8.3 i (4)
[17] Se f.eks. side 91 i (19)
[18] MCMC genererer ikke uavhengige utvalg, dvs. at observasjonene er korrelerte



Figur 4.1 *Utvalgsprosedyre ved bruk av invers transformasjon av eksponentialfordelingen (λ = 2). Variabelen F trekkes uniformt i intervallet (0, 1), og den tilhørende eksponentialfordelte variabelen x regnes ut*

Kvaliteten på slumptallsgeneratorer er et forskningsfelt i seg selv. Slumptall genereres i realiteten ofte etter deterministisk algoritmer basert på datamaskinens systemklokke e.l. Tallene er derfor ikke uavhengige, noe som avsløres av repetisjonsmønstre i lange nok sekvenser. Slike tall kalles ”pseudo-tilfeldige” i motsetning til ”sanne” tilfeldige tall.[19] Dette er i prinsippet en alvorlig feil, men i praksis er deterministisk genererte slumptall sjelden et problem. For de fleste simuleringsformål genererer man slumptallssekvenser av begrenset lengde, og det er tilstrekkelig at tallene *virker* tilfeldige og ikke avsløres i statistiske tester av tilfeldighet. Det at det er mulig å reprodusere slumptallsekvenser er også en fordel i det at man kan gjenskape bestemte hendelsesforløp.

4.3 Eksempler på anvendelse av MC-simuleringer

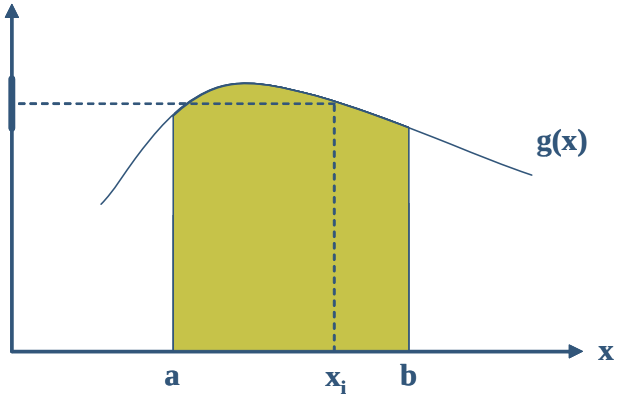
I dette kapittelet beskrives noen enkle eksempler som illustrerer ulike anvendelser av statisk MC-simulering.

4.3.1 Estimere verdien av et integral

Mange problemer involverer integraler som ikke har en analytisk løsning. Det finnes i dag mange gode numeriske metoder for å løse integraler, og disse er generelt å foretrekke fremfor å bruke MC-simulering. Men simuleringstilnærmingen kan være fruktbar når problemet blir mangedimensjonalt og mindre håndterbart med standard numeriske integrasjonsmetoder.[20] En

[19] Det finnes metoder for å fremskaffe ”sanne” tilfeldige tall basert på fysiske prosesser som f.eks. termisk støy i elektriske kabler eller andre fenomener på kvantenivå
[20] Se f.eks. (20)

klassisk og informativ anvendelse av MC-simulering er nettopp å løse slike integraler, derfor inkluderes et forenkelt eksempel her. Funksjonen $g(x)$ skal integreres over intervallet (a,b): $I = \int_a^b g(x)dx$. Man trekker uniformt N verdier x_i fra intervallet (a, b), og finner funksjonsverdien $g(x_i)$ for hver av disse (figur 4.2). En estimator for integralet vil da være $\hat{I} = \frac{b-a}{N}\sum_{i=1}^{N} g(x_i)$.



Figur 4.2 *Funksjonen g(x) skal integreres over intervallet (a, b). I MC-simuleringen trekkes x-verdier uniformt i (a, b)*

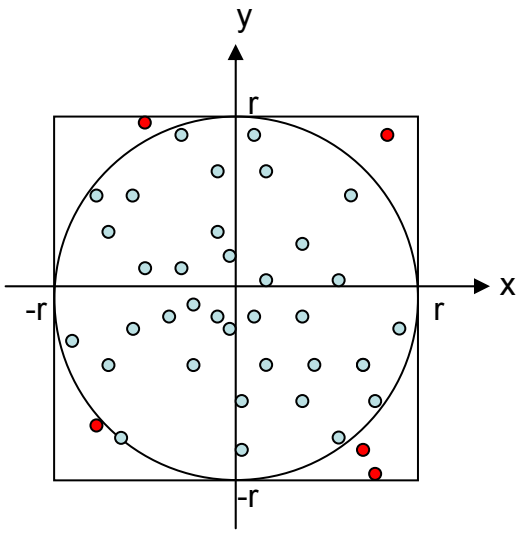
Dette er et intuitivt riktig resultat fra to synsvinkler. For det første kan gjennomsnittet $\frac{1}{N}\sum_{i=1}^{N} g(x_i)$ betraktes som en representativ høyde under grafen, og multiplisert med grunnlinjen $(b - a)$ får man arealet, dvs. verdien av integralet. Den andre måten å se det på er å tenke seg en oppdeling av intervallet (a, b) i N like partier som alle har bredde $\frac{b-a}{N}$, og at integralet blir en sum av N små arealbidrag med grunnlinje $\frac{b-a}{N}$ og høyde $g(x_i)$.[21]

Hvor mange trekninger man gjør vil være en avveining mellom ønsket presisjon og tilgjengelige ressurser. I mangedimensjonale problemer lønner det seg å trekke ”smart” ved å ta ekstra mange observasjoner i områder der gradienten til integranden $g(\vec{x})$ er stor (her er $\vec{x}$ en vektor). I praksis gjøres dette ved å trekke fra en sannsynlighetsfordeling som i form ligner integranden mest mulig. Å finne en slik sannsynlighetsfordeling $\pi(\vec{x})$ er langt fra trivielt. Det generelle uttrykket for estimatoren for integralet blir da $\hat{I} = \frac{1}{N}\sum_{i=1}^{N} \frac{g(\vec{x}_i)}{\pi(\vec{x}_i)}$, som reduseres til vårt tidligere estimat $\hat{I} = \frac{b-a}{N}\sum_{i=1}^{N} g(x_i)$ når $\pi(x)$ er uniformfordelingen i en dimensjon.[22]

[21] Dette kalles en Riemann-tilnærming, men har egentlig ingenting med simulering og tilfeldige trekninger å gjøre
[22] Se f.eks. (20)

4.3.2 Estimering av π

Bruk av MC-simulering for å estimere tallet π er et gammelt eksempel.[23] Man kan tenke seg en sirkel med radius *r* innskrevet i et kvadrat med sidekant 2*r*.[24] Formene plasseres i et koordinatsystem med to akser x og y, og tilfeldige koordinater trekkes innenfor kvadratet. Dette gjøres ved å trekke *x*-verdier og *y*-verdier uniformt i intervallet (-*r*, *r*) (se figur 4.3). En god analog er en dartskive på en vegg der pilkastene er obsevasjoner.[25] Man trekker *N* koordinater i kvadratet, og teller opp antallet *n* som også havner innenfor sirkelen. Andelen trekninger innenfor sirkelen vil da være lik forholdet mellom sirkelens og kvadratets arealer, dvs. $\frac{n}{N} \approx \frac{\pi r^2}{4r^2}$, og π finnes lett fra dette. I figur 4.3 er $N = 40$ og $n = 35$, noe som gir $\pi \approx 3{,}5$. Feilen i estimatet vil minke med økende antall replikasjoner (jf. kapittel 2.10.2).



Figur 4.3 Estimering av π med pilkastmetoden

Denne fremgangsmåten kan åpenbart også benyttes til å løse det beslektede problemet med å estimere arealet av en vilkårlig form som er inneholdt i en annen form med kjent areal.

4.3.3 Eksempler fra utpostscenariet

Dette eksemplet tar utgangspunkt i scenariet beskrevet i kapittel 3 og appendiks B. Problemstillingen som studeres er kjøretid fra basen til utposten for kolonner av ulik størrelse, både uten angrep og medregnet angrep. Forsyningsaspektet blir ergo ikke studert, kun kjøretiden. De aspektene i scenariet som har noe å si for denne analysen, er alle de som har knyttet en sannsynlighetsfordeling til seg, nemlig kjørehastighet, tid mellom feil, reparasjonstid, angrep og resultat av angrep. Eksempel på spørsmål man kan svare på er:

---

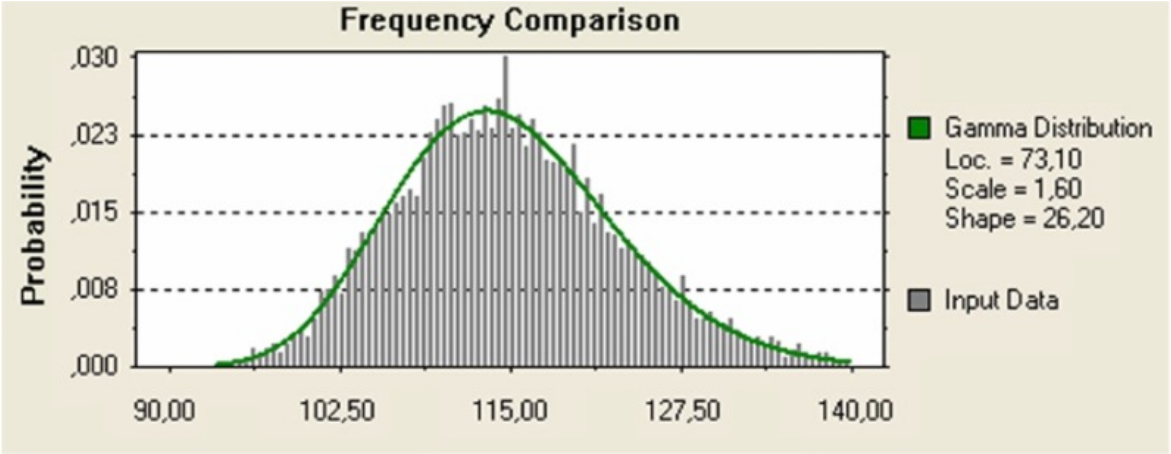
[23] I 1800-tallsproblemet "Buffon's Needle" beregnes π ved å kaste en nål på et plankegulv e.l. og telle antall ganger nålen krysser to planker (se f.eks. http://en.wikipedia.org/wiki/Buffon%27s_needle)

[24] De geometriske formene trenger ikke å være som beskrevet, det eneste kravet er at arealene er kjent og at π inngår i et av dem

[25] For å sikre tilfeldig kasting, krever eksperimentet at kasteren har bind for øynene eller er overordentlig evneløs i sporten

- Hva slags betydning har ulike kolonnestørrelser for kjøretiden?
- Hva er sannsynligheten for at kjøretiden blir lengre enn en gitt grense?
- Hvor store forsinkelser medfører et angrep?

Figur 4.4 viser et eksempel på et histogram over simuleringsresultater for kjøretid. Formen på histogrammet avslører hvilke kjøretider som er mest vanlige og i hvor stor grad kjøretiden kan bli *betydelig* større eller mindre enn gjennomsnittsresultatet. I praksis blir resultatfordelingen ofte seende noenlunde ut som i figur 4.4, nemlig normalfordelingsaktig med en slagside en av veiene. Men det er også lett å konstruere eksempler der resultatfordelingen kan anta de villeste former. I dette eksempelet har simuleringsverktøyet Crystal Ball for Excel blitt benyttet (se kapittel 4.4), og resultatene er basert på 5000 replikasjoner.



Figur 4.4 Histogram over kjøretider og en tilpasning av gammafordelingen til dataene. Resultater fra 5000 replikasjoner med verktøyet Crystal Ball

Resultatfordelingen er empirisk og "rufsete", men når man skal bruke fordelingen til å gjøre beregninger, er det ønskelig med en kjent, parametrisk sannsynlighetsfordeling som representerer datasettet tilstrekkelig godt. I dette eksempelet har man forsøkt å tilpasse en gammafordeling med parametere som angitt i figuren. Som nevnt i kapittel 2.10.1, er det viktig å være kritisk og påpasselig når man skal tilpasse fordelinger til datasett.

4.4 Verktøy

MC-simulering kan gjøres med mange ulike slags verktøy. De fleste statistikkpakker vil i prinsippet kunne anvendes til slik simulering. Stokastiske modeller implementert i et standard programmeringsspråk kan alltid uten problemer MC-simuleres – alt som trengs er en ytre repetisjonsløkke og en egnet struktur (f.eks. en tabell) for å logge resultater fra hver replikasjon. Dette gjelder også "enklere" programmeringsomgivelser som f.eks. Matlab og Visual Basic.

Det finnes også en rekke verktøy spesialutviklet for MC-simulering. *Crystal Ball*[26] (CB) for Excel er et eksempel på et slikt verktøy. CB har blitt benyttet en del ved FFI; et veldokumentert

[26] Decisioneering Inc., www.decisioneering.com/crystal_ball

eksempel er en kostnadsanalyse for langtrekkende presisjonsstyrte våpen (21). Andre verktøy som kan nevnes er risikoanalyseverktøyet *@Risk*[27] – også det for Excel – og *iDecide*[28].

Hvilket verktøy man skal bruke til et gitt problem, vil være mest avhengig av hva man er komfortabel med fra før. Statistikkpakker og verktøy som MatLab har generelt en litt høyere startterskel enn de mer skreddersydde verktøyene, men på den annen side gir disse større fleksibilitet. Hvis modellen allerede eksisterer i et programmeringsspråk, er selvsagt det enkleste å utvide programmet med noen kodelinjer. Hvis man derimot begynner en analyse fra scratch, og ikke har bestemte verktøypreferanser, kan det være bryet verdt å legge litt mer innsats ned i valget av verktøy.

5 SYSTEMDYNAMIKK

Mye av innholdet i dette kapittelet er basert på læreboken "System Dynamics Modeling" av Coyle (22). Denne boken, sammen med (23), vil være gode utgangspunkt for en videre innføring i feltet systemdynamikk. En mer kortfattet introduksjon til denne og andre simuleringsmetoder kan finnes i (7).

5.1 Innledning

Som nevnt i kapittel 2.7 kalles systemer som endrer tilstand under et tidsforløp for *dynamiske* systemer. Kapittel 6 tar for seg *diskrete* dynamiske systemer, dvs. systemer som endrer tilstand på diskrete/isolerte tidspunkt. Det herværende kapittelet tar for seg *kontinuerlig* dynamisk simulering, dvs. simulering av systemer som endrer tilstand kontinuerlig med tiden. Hvorvidt et system skal karakteriseres (og modelleres) som diskret eller kontinuerlig, er avhengig av formålet med studien av systemet (dette ble diskutert i kapittel 2.7.6).

Kontinuerlige dynamiske systemer finnes overalt i naturen og i samfunnet. Dette kan være relativt enkle, deterministiske systemer som f.eks. en pendel i bevegelse, eller mer komplekse systemer som boligmarkedet i Oslo. Komplekse systemer karakteriseres ofte ved en stor grad av interaksjon mellom enhetene, slik at endringer i en del av systemet kan få flere og uforutsigbare ringvirkninger i andre deler av systemet. Denne uforutsigbarheten i oppførsel trenger ikke kun å komme av at systemet i seg selv er stort og uoversiktlig. I dynamiske systemer er det spesielt to fenomener som øker kompleksiteten, nemlig tilbakekoblingseffekter og tidsforsinkelser. Dynamiske systemer med komplekse tilbakekoblingseffekter og tidsforsinkelser modelleres gjerne vha. systemdynamikk.

5.2 Systemdynamikk og kontrollproblemer

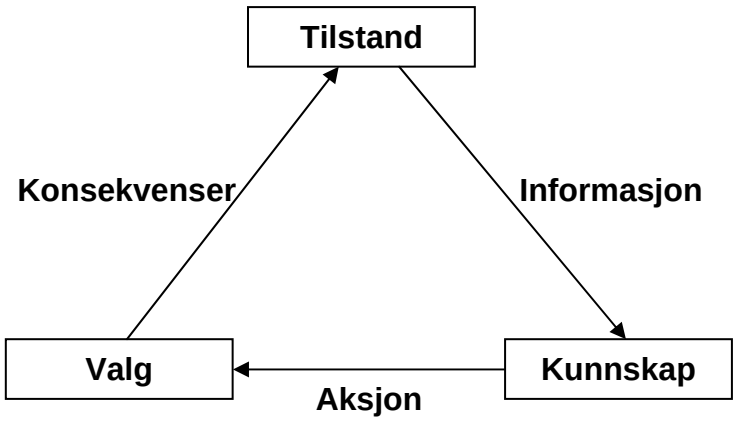
Fagfeltet systemdynamikk (SD) ble opprinnelig utviklet på 1950-tallet av Jay Forrester ved MIT

[27] Palisade Corporation, www.palisade.com/risk
[28] Decisive Tools LLC, www.decisivetools.com. *iDecide* er en videreutvikling av det tidligere verktøyet *Definitive Scenario*

(22). SD har sine røtter i fagfeltet kontrollteori[29], og er en overføring av prinsipper og tankegang fra kontrollteori til andre typer dynamiske systemer, hovedsaklig av sosial og økonomisk art. De fleste assosierer derfor SD primært med business-anvendelser, godt hjulpet av det som ser ut til å være det moderne referanseverket av Sterman innen området: "Business Dynamics: Systems Thinking and Modelling for a Complex World" (23). En treffende definisjon av SD fra dette perspektivet er (fra (22)):

> Systemdynamikk er den delen av kontrollteori som omhandler sosio-økonomiske systemer, og den delen av Management Science som omhandler kontrollproblemer.

Figur 5.1 illustrerer den overordnede situasjonen i et generelt kontrollproblem. Formålet er å kontinuerlig observere og korrigere et system slik at relevante variabler holdes innenfor gitte grenser. Systemet har til enhver tid en *tilstand* som kommer av *valg* man gjør som kontrollør. På bakgrunn av *kunnskap* om tilstanden, gjør kontrolløren nye valg for å justere og styre tilstanden i ønsket retning. Det *dynamiske* i denne kontrollsyklusen ligger i troikaen *informasjon*, *aksjon* og *konsekvenser*: Informasjon utløser aksjoner som har konsekvenser som gir ny informasjon. Hver av disse tre "prosessene" er gjerne forbundet med en tidsforsinkelse: det tar tid før konsekvensene av valg tar effekt; det tar tid før informasjonen om systemets tilstand når kontrolløren; det tar tid å vurdere informasjonen og treffe nye valg.



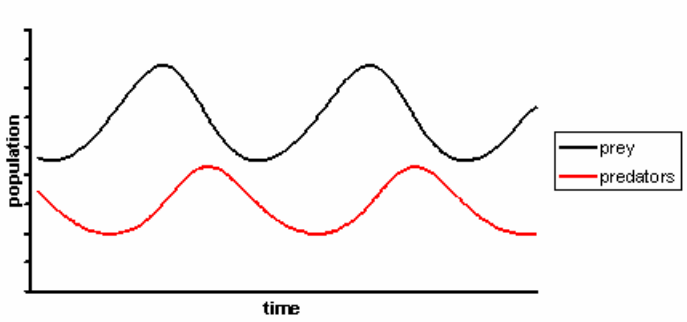
Figur 5.1 Syklusen informasjon–aksjon–konsekvens i kontrollmodeller (fra (22))

Den tidsforsinkelsen som gjør seg mest gjeldende i praksis, er den knyttet til konsekvenser. Konsekvensene av en aksjon kommer til uttrykk først etter en viss tidsforsinkelse, og aksjonens effekt har da blandet seg med andre dynamiske effekter – deriblant andre aksjoner som er tatt før og etterpå – slik at det blir vanskelig å isolere effekten av aksjoner. Det at man ikke lett klarer å koble handling og konsekvens, kan fort føre til overkorrigering. Et enkelt eksempel er temperaturregulering i dusjen: Dersom man ønsker en litt varmere vannstråle, justeres blandebatteriet så mye man tror man trenger. Hvis effekten ikke raskt melder seg, kan man lett tro at justeringen var effektløs, og dermed øke endringen enda mer. Når effekten så melder seg, blir man brent, og må hurtig korrigere tilbake. Folk flest har systemkunnskap nok til å være både

[29] Begrepene kontrollteori, styring, reguleringsteknikk/-teori og kybernetikk brukes mye om hverandre når man snakker om kontrollproblemer

forsiktige og tålmodige i dusjen, men ikke alle systemer er like enkle og oversiktelige som dette.

SD er ikke begrenset til kontrollproblemer og sosioøkonomiske systemer, men kan brukes til å modellere egnede kontinuerlige problemer innen alle domener. Et godt og klassisk eksempel på dette fra naturens verden er rovdyr – byttedyrmodeller, som ikke er et kontrollproblem. I den enkle Lotka-Volterra-modellen av slike systemer er endringsratene i bestandene proporsjonal med størrelsen på bestandene, dvs. $\dot{R} = -c_1 R + c_2 RB$ og $\dot{B} = c_3 B - c_4 RB$ for henholdsvis rovdyr (R) og byttedyr (B) (c_i er positive proporsjonalitetskonstanter). Slike systemer vil utvise en oscillerende oppførsel rundt en likevektløsning.[30]



Figur 5.2 Periodisk løsning av Lotka–Volterra rovdyr–byttedyrmodell (kilde: Wikipedia)

5.3 Influensdiagram

Systemdynamisk modellering gjøres som regel ved hjelp av influensdiagram (også kalt *kausale løkkediagram*). Slike diagram beskriver hvilke variabler som inngår i systemet og hvordan de påvirker hverandre. Figur 5.1 er et eksempel på et enkelt, nedstrippet influensdiagram for et generisk system, som selvfølgelig må fylles med spesifikt innhold for å være anvendelig. I reelle problemer blir slike diagram gjerne omfattende og intrikate. Sammenhenger mellom variabler illustreres ved bruk av piler og beskrives formelt av differensialligninger for hvordan variablene endres. Disse ligningene løses vanligvis numerisk med tilstrekkelig små tidsskritt. Simulering med systemdynamikk er altså alltid tidsstyrt (se kapittel 2.7.7).

Variablene som inngår i et influensdiagram vil typisk danne flere tilbakekoblingsløkker (”feedback loops”). Disse løkkene kan deles i inn i negative (også kalt balanserende eller målsøkende) og positive (også kalt forsterkende) løkker. *Negative* løkker søker vha. kontroll-signaler å eliminere en registrert forskjell mellom faktisk og ønsket tilstand – det finnes et ”mål” å styre etter. Temperaturregulering i dusjen er et eksempel på en negativ løkke. *Positive* løkker derimot, er vekstmekanismer der variablene øker (eller minker) uten bremser. Penger i banken (eller gjeld) som forrenter seg er et eksempel på en positiv løkke. Et annet eksempel man kan tenke seg er biler på en vei, der hastigheten reduseres med antall biler på veien (rushtrafikk). Man kan da få den situasjonen at bilflyten forbi et kontrollpunkt blir *redusert* dersom man *øker* antall biler på veien. Slike effekter der en korrigering i retning av en ønsket effekt faktisk gir

[30] Se f.eks. Wikipedia: http://en.wikipedia.org/wiki/Lotka_Volterra-equation

den *motsatte* effekten, er svært vanskelig å forholde seg til. Systemdynamikk er spesielt egnet til å modellere slike effekter.

5.4 Anvendelser av systemdynamikk

Systemdynamikk er velegnet for å modellere og studere de store sammenhengene i kompliserte dynamiske systemer. Formålet er gjerne å *forstå* oppførselen til systemet og forklare en observert dynamikk, og ikke primært gjøre kvantitative beregninger.

SD kan med fordel brukes til innledende, aggregert eksperimentering med et diskret system, der de diskrete fenomenene gjøres om til overordnede flytrater. Senere kan man bruke mer detaljerte simuleringsmetoder som f.eks. diskret hendelsesstyrt simulering (omtalt i kapittel 6) for å gjøre kvantitative beregninger.

Når dette er sagt, må det likevel presiseres at SD langt ifra er *uegnet* til å produsere kvantitative resultater. Et eksempel på en kvantitativ analyse fra FFI er en utredning om mulige fremtidige befalsordninger (24).

Som beskrevet er SD også skreddersydd til modellering av kontrollproblemer der en kontrollør prøver å styre en prosess i ønsket retning eller holde prosessen stabil innenfor gitte beskrankninger. Noen eksempler fra FFI på bruk av SD på kontrollproblemer i opplærings-sammenheng, er arbeid i prosjektene P802 ”Beslutningstrening på høyere nivå” og oppfølgeren P846 ”Implementering av beslutningstrener” (25).

Fokuset på å utforske og forstå systemer gjør at SD i OA-sammenheng som regel anvendes på *deterministiske* problemer. Det er imidlertid ikke noe i veien for å introdusere stokastiske fenomener i modellene. Det er åpenbart at stokastiske hendelser – eller deterministiske hendelser for den saks skyld – vil være av *diskret* natur, og dermed å anse som påtrykk på systemet på visse tidspunkt som forrykker balansen i et ellers deterministisk forløp.

I motsetning til diskrete metoder med fokus på *antall* og *hendelser*, er SD rettet inn mot *nivåer* og *flytrater* knyttet til variablene i systemet. Begrepsbruken her varierer. Innen økonomi brukes begrepene ”stock” (beholdning) og ”flow” (strøm/flyt), mens innen SD brukes begrepene ”level” (nivå) og ”rate” (rate). Innen kontrollteori bruker man ”state variables” (tildstands-variabler) i stedet for ”levels”.

5.5 Eksempler fra utpostscenariet

Det tas utgangspunkt i casen beskrevet i kapittel 3 og appendiks B, hvor problemstillingen er å bringe forsyninger fra en base til en utpost gjennom fiendtlig terreng. For å kle denne casen i en systemdynamisk drakt, må man kunne løsrive seg en god del fra detaljeringsgraden i scenario-beskrivelsen. Alle diskrete fenomener som man ønsker å ta med, må omsettes til kontinuerlige variabler. Man må for det første tenke i rater i stedet for hendelser, og nivå i stedet for

forsendelsespakker. Det kan tenkes flere problemstillinger egnet for simulering med system-dynamikk; i det følgende beskrives nærmere et overordnet problem og et kontrollproblem.

5.5.1 Et overordnet problem

Det antas en fastlagt rutine for etterforsyning av utposten, og man ønsker å studere mengden forsyninger som til enhver tid befinner seg på utpostens lager gitt en viss forbruksprofil. Formålet er å kunne si noe om hvor ofte lagerbeholdningen kommer under gitte nivå. Diskrete (og gjerne stokastiske) hendelser som at lastebiler sendes av gårde, kommer frem, bryter sammen eller blir angrepet modelleres som deterministiske rater. Eksempelvis vil sannsynligheten 10 % for at en bil blir slått ut under angrep, modelleres med en deterministisk 10 % tapsrate. Denne tapsraten er forventningen av den stokastiske variabelen ”andel biler slått ut”.

Dette problemet har i utgangspunktet ingen tilbakekoblingseffekter. Slik kompleksitet kan imidlertid lett inkluderes, eksempelvis ved å tenke seg at kjøretøyene kjører fortere når lagerbeholdningen er liten, og at informasjon om dette ikke når fem før etter en viss tid.

5.5.2 Et kontrollproblem

Det antas at utposten har en viss forbruksprofil av forsyninger, men ingen lagerkapasitet. Utposten er da avhengig av kontinuerlige leveranser av forsyninger. Med forbruks*profil* menes at forsyningsbehovet varierer over tid på en forutsigbar måte (f.eks. størst behov mot slutten av hver måned). Utfordringen er da å finne en forsyningspolicy som best mulig matcher forbruksprofilen. En passende oppløsning for dette problemet vil være dager, dvs. at utpostens behov er daglige behov og at det tas en beslutning om utsendelse per dag. Kontrollproblemet er altså: Hvilke kontinuerlige (daglige) beslutninger må tas for å møte et gitt, variabelt forsyningsbehov? Her må man anta at det innebærer en kostnad å sende ut forsyninger som ikke blir brukt, ellers blir løsningen trivielt å sende ut mest mulig hele tiden.

For å gjøre problemet interessant, kan man innføre tilbakekoblingseffekter og tidsforsinkelser ved å anta at lastebilene må gjennomgå en eller annen klargjøringsprosess på basen før de kan begynne å kjøre (dieselfylling, papirutfylling, verkstedsjekk m.m.) eller at de må gjennom kontrollposter og sjekkpunkt underveis. Disse ”postene” som lastebilene må gjennom, har alle begrenset kapasitet, og medfører fort ventetider når utsendelsestakten er for stor, i tillegg til ev. behandlingstid. Man kan videre tenke seg at behandlingstiden på postene er avhengig av hvor mange som venter på tur: det kan gå fortere fordi det haster mer, eller saktere fordi operatørene på postene blir stresset. Hendelser som angrep og sammenbrudd av lastebiler kan også modelleres som poster lastebilene må gjennom.

5.6 Verktøy

Man kan i prinsippet utvikle gode SD-modeller vha. et standard programmeringsspråk, men dette innebærer gjerne mye arbeid og lite oversiktlighet. Det er mer hensiktsmessig å benytte et kommersielt verktøy spesialutviklet for SD-modellering. Tidlige SD-verktøy baserte seg på ligningsskriving via en tekseditor. Moderne SD-verktøy har gjerne et grafisk brukergrensesnitt

som tillater tegning av influens- og flytdiagrammer og automatisk generering av ligninger basert på diagrammene. Dette forenkler modelleringsprosessen veldig, både når det gjelder visualisering, utvikling og feilsøking.

Det eksisterer flere verktøy for utvikling av SD-modeller. Noen av de mest kjente er:

- **Ithink** og **Stella**, isee systems inc. (www.iseesystems.com)
  Disse programmene finnes også i sterkt reduserte og tidsbegrensede gratisversjoner for utprøving. isee systems er tidl. kjent som High Performance Systems.
- **Vensim**, Ventana Systems Inc. (www.vensim.com)
  Dette programmet finnes også i en enklere gratisversjon.
- **Studio**-familien, Powersim Software AS (www.powersim.com)
  Dette programmet finnes i en tidsbegrenset gratis fullversjon. Bergensbaserte Powersim er tidl. kjent som ModellData AS, og produktfamilien Studio bygger på det tidl. verktøyet Powersim.

Flere av disse verktøyene er relativt like i oppbygging og virkemåte. Små forskjeller i brukerterskel, simuleringskraft o.l. forekommer, men hvilket verktøy man velger vil i første rekke være en smakssak. Verktøyene Ithink og Vensim har blitt brukt en del på FFI.

6 DISKRET HENDELSESSTYRT SIMULERING

Diskret hendelsesstyrt simulering (DHS) er simulering av systemer hvor tiden er en viktig parameter (dynamisk simulering, jf. kapittel 2.7.1). DHS beskriver et systems utvikling ved hjelp av tilstandsvariabler som kan endre verdi på et endelig antall diskrete tidspunkt. Dette betyr at DHS er et aktuelt verktøy for å studere systemer som ikke kontinuerlig endrer sin tilstand når det gjelder interessante egenskaper.

I motsetning til systemdynamisk modellering er DHS best egnet til å modellere systemer med en *moderat* grad av interaksjon mellom enheter. DHS kan med fordel brukes til å gjøre detalj-simuleringer etter en innledende, overordnet simulering med andre metoder (som f.eks. SD).

DHS er en utbredt simuleringsmetode i OA-sammenheng. DHS blir særlig anvendt for å studere trafikk- og produksjonssystemer hvor man blant annet ønsker å finne flaskehalser og redusere køer. DHS anvendes på mange forskjellige problemtyper innen militær sektor. Angrep på og reparasjon av rullebaner for fly er aktiviteter som lar seg studere detaljert ved hjelp av DHS. Strid mellom en hangarskipsgruppe og angripende missiler, fly og ubåter egner seg også for DHS, men her må man nøye vurdere detaljeringsnivået.

Læreboken til Banks et al. (11) er et godt utgangspunkt for en videre innføring i DHS. En mer kortfattet introduksjon til denne og andre simuleringsmetoder kan finnes i (7).

6.1 Terminologi

For å kunne gi en kort beskrivelse av hvordan DHS foregår, trengs det en del begrep og definisjoner. Det er naturlig å starte med de fire grunnleggende begrepene entitet, attributt, aktivitet og hendelse.

- En **entitet** er et objekt eller en komponent i systemet som eksplisitt blir representert i modellen på et eller annet tidspunkt. Lastebiler og jagerfly kan være eksempler på entiteter.
- Et **attributt** er en egenskap ved en entitet. Attributter for en lastebil kan være maksimal lasteevne, maksimalt akseltrykk og drivstofforbruk.
- En **aktivitet** karakteriseres i denne sammenheng ved sin varighet. Drivstoffylling og hjulskifte er eksempler på aktiviteter.
- En **hendelse** er noe som får en eller flere av tilstandsvariablene som beskriver systemet til å endre seg. Eksempelvis vil hendelsen ”Drivstoffylling er ferdig” medføre at variabelen ”Antall lastebiler til vedlikehold” får sin verdi redusert med én mens verdien av variabelen ”Antall lastebiler klar til innsats” øker med én. Hendelser har ingen varighet.

Med denne terminologien vil drivstoffylling beskrives som hendelsen ”Drivstoffylling starter”, aktiviteten ”Drivstoffylling” og hendelsen ”Drivstoffylling er ferdig”.

6.2 Input

Før man kan starte en simulering er det en del input som kreves. For det første trengs en prosessbeskrivelse av aktørene i systemet. En kunde vil f.eks. ankomme, stille seg i kø, vente, forlate køen, bli betjent og forlate systemet. Mulig samspill med andre aktører må beskrives. Køtypene må spesifiseres, f.eks. FIFO (First In, First Out). Videre må sannsynlighets-fordelingene for ankomsttider og betjeningstider spesifiseres. Når det gjelder ankomsttider, er det vanlig å beskrive tiden mellom ankomster som eksponentielt fordelt. Avhengig av formålet med simuleringen må parameteren i eksponentialfordelingen velges eller estimeres. Fordelingen av betjeningstider vil variere sterkt med type tjeneste som tilbys. Basert på observasjon av betjeningstider, kan det være aktuelt å undersøke om f.eks. normalfordeling, trekantfordeling eller Weibullfordeling kan beskrive betjeningstiden tilstrekkelig godt.

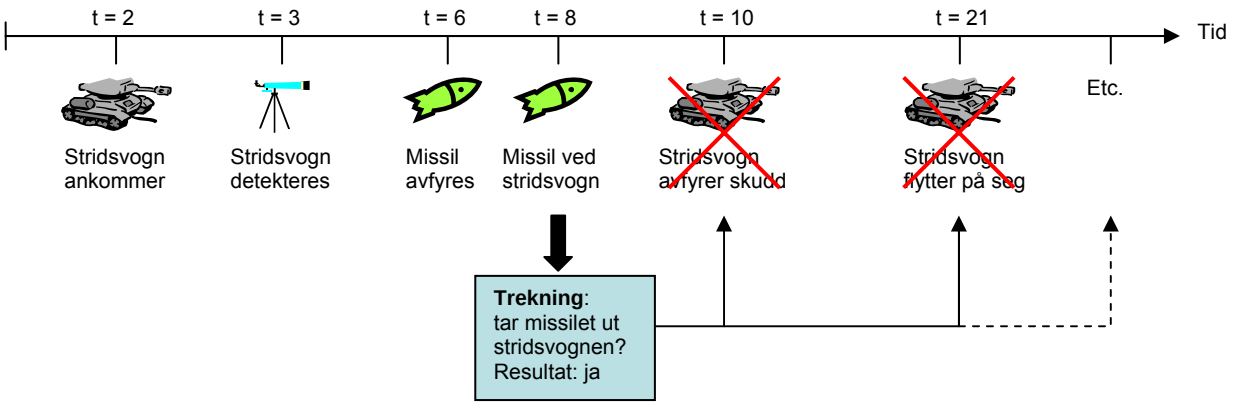
6.3 Gjennomføring

Sentralt i gjennomføringen står simuleringsklokken og tidsaksen. Simuleringsklokken måler intern tid i simuleringen. Denne tiden er forskjellig fra den reelle tiden maskinen bruker på å gjennomføre simuleringen.

Ved initialiseringen trekkes tidspunktene for en del hendelser og disse henges opp på tidsaksen. En trekning foregår ved at maskinens slumptallsgenerator brukes til å gi et pseudotilfeldig tall u uniformt fordelt mellom 0 og 1. Hvis F er den kumulative sannsynlighetsfordelingen for ventetiden til neste hendelse, vil denne inntreffe ved tid $F^{-1}(u)$.

Når initialiseringen er gjennomført ved tid $t = 0$, flyttes simuleringen til neste hendelse på tidsaksen og denne gjennomføres. Gjennomføringen kan f.eks. innbære at nye hendelser blir satt opp på tidsaksen og at andre hendelser blir fjernet. Et eksempel på fjerning av hendelse kan være: En stridsvogn er hengt opp på tidsaksen med hendelsen ”skudd avfyres” ved tid $t = 10$. Ved $t = 8$ kan stridsvognen bli truffet av et missil og bli ødelagt. Ved $t = 8$ blir da hendelsen ”skudd avfyres” ved $t = 10$ fjernet fra tidsaksen (denne prosessen er illustrert i figur 6.1). Ved hver hendelse blir altså alle impliserte variabler endret. Variabelen ”antall personer i kø C” kan f.eks. bli økt med én (ny kunde ankommer) eller redusert med én (kunde er ferdig betjent).

Enhver simulering må ha et stoppkriterium. Dette kan f.eks. være at simulert tid når en fastsatt grense eller at gitte variable når et fastsatt nivå (f.eks. antall kunder ankommet er lik 50, eller fienden er desimert til mindre enn 10 % av opprinnelig styrke).



Figur 6.1 *Ved initialiseringen av en DHS trekkes en del hendelser som henges opp på tidslinjen. Under simuleringens gang sørger stokastiske fenomener for at planlagte hendelser blir oppdatert/fjernet og nye hendelser blir generert*

6.4 Output

Formålet med simuleringen bestemmer hvilke variabler som registreres. I noen tilfeller kan man være interessert i å finne ut hvilke ekstremsituasjoner som kan oppstå og hvilke kombinasjoner av hendelser som førte til dette. I slike tilfeller vil det være nødvendig med detaljert registrering av mange variabler. I andre tilfeller kan man være interessert i hvordan systemet ”gjennomsnittlig” oppfører seg. Da gjennomføres et antall replikasjoner av simuleringer med identisk utgangspunkt, men med ulike strenger av pseudotilfeldige tall. På det grunnlag kan man så beregne f.eks. gjennomsnittlig kølengde og lage et histogram over de observerte kølengdene for å få et inntrykk av hvordan kølengden fordeler seg.

6.5 Eksempler fra utpostscenariet

Nedenfor følger noen eksempler på problemer og spørsmål basert på utpostscenariet beskrevet i kapittel 3 og appendiks B, som helt eller delvis kan løses ved hjelp av simulering.

- Hva er best for å forsyne utposten: store kolonner sjelden eller små kolonner ofte?

- Anta gitt en bestemt rutine for etterforsyning av utposten. Hvor stor lageroppbygging trengs da på utposten for at sannsynligheten for å slippe opp for forsyninger skal være mindre enn 10 % i en gitt tidsperiode?
- Finn fordelingen for kjøretid for en kolonne fra hovedbasen til utposten.
- Beregn evakueringstid fra utposten hvis bare helikoptre kan brukes.
- Beregn evakueringstid fra utposten hvis både helikoptre og kjøretøy kan brukes.

For at man skal kunne løse problemer ved hjelp av simulering, er det en rekke data som må spesifiseres. Noen verdier vil ligge fast i hele simuleringen, mens andre verdier varieres for å se hvilke utslag det gir (jf. beskrivelsen av parametere og variabler i kapittel 2.9.2). Nedenfor følger eksempler på data som trengs for at en simuleringsmodell skal kunne besvare spørsmål av typen som ble stilt ovenfor.

- Lasteevne for biler og helikoptre
- Drivstofforbruk for biler og helikoptre
- Mean Time Between Failures (MTBF) for biler og helikoptre
- Forholdsregler ved bilskader. Repareres den på stedet eller etterlates den? Er det muligheter for omlastning?
- Forventet antall dager med stengte veier for de ulike månedene
- Maksimal hastighet for lastebilene på ulike veistrekninger
- Antall disponible lastebiler, pansrede personellkjøretøy, stormpanservogner og helikoptre
- Sannsynlighet for at en kolonne blir angrepet
- Sannsynlighetsfordelingen for den tidsforsinkelsen et angrep medfører
- Forventet antall vedlikeholdstimer per brukstime for biler og helikoptre
- Hyppighet og varighet av vær som umuliggjør bruk av helikopter

For en første versjon av en slik simuleringsmodell vil det være naturlig å bruke data basert på erfaringer fra Norge med noen skjønnsmessige modifikasjoner. Etter hvert som man høster erfaring på stedet, vil man få mer realistiske data, eksempelvis for forventet antall vedlikeholds-timer per brukstime.

6.6 Verktøy

En DHS kan i prinsippet gjennomføres manuelt, men det vil ta lang tid å gjennomføre selv enkle eksempler. Siden man ofte ønsker å studere hvordan et system ”gjennomsnittlig” oppfører seg, vil det være behov for å gjenta simuleringen mange ganger med samme startbetingelser, men med ulike sekvenser av tilfeldige tall for bestemmelse av enkeltutfallene. DHS forgår derfor ved hjelp av ulike typer programvare på datamaskiner. Felles for all slik programvare er at det foreligger moduler som tar seg av listehåndtering, generering av pseudotilfeldige tall og andre grunnleggende operasjoner.

Det fins mange måter å karakterisere og klassifisere slik programvare på. En måte er vist i Banks et al. (11) på side 95 og en annen i Law & Kelton (4) på side 204. Banks et al. bruker tre

kategorier. For det første er det mulig å lage simuleringsprogrammer ved bruk av vanlige programmeringsspråk som f.eks. C++ og Java. Dette var nok vanligere før enn det er nå. For det andre finnes det spesielle simuleringsspråk som GPSS/H og MODSIM III. Det objektorienterte språket SIMULA var i sin tid mye brukt til simulering ved FFI, men utviklingen har nå løpt fra SIMULA. I den tredje kategorien kommer generelle simuleringsomgivelser som gjerne har grafisk brukergrensesnitt, animering, innebygde statistikkpakker og støtte for dokumentasjon. Noen eksempler på slik programvare er Arena, Extend, Simul8 og AnyLogic[31]. Se (11) for en beskrivelse av disse og andre verktøy.

7 AGENTBASERT MODELLERING OG SIMULERING

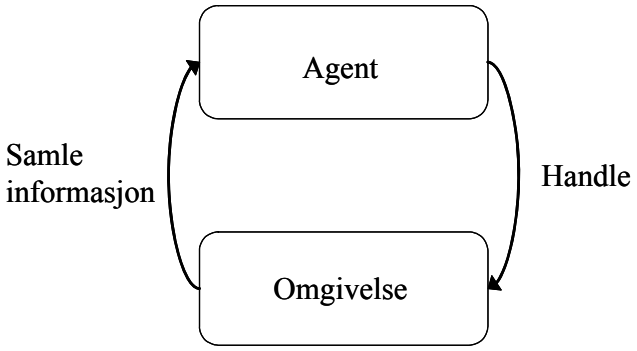
Hensikten med dette kapitlet er å gi en kortfattet introduksjon til agentbasert simulering (ABMS) for å kunne sammenlike denne simuleringsmetoden med andre relevante simuleringsmetoder innen OA. En mer detaljert fremstilling av ABMS med vurderinger av denne metodens egnethet opp mot typiske OA-problemstillinger, finnes i (5). For en mer fyldig presentasjon av ABMS henvises det til boken ”An Introduction to MultiAgent Systems” av Wooldridge (26).

7.1 Hva er ABMS?

ABMS skiller seg fra andre simuleringsmetoder ved at alle eller deler av de simulerte entitetene er modellert som agenter. En agent kan ses på som et autonomt softwareobjekt som tar beslutninger og handler basert på dets oppfattelse av sine omgivelser (27). Det at en agent kan operere autonomt i en omgivelse hvor den interagerer med andre agenter, er kanskje den mest sentrale egenskapen ved en agent. For at agenten skal kunne tilpasse sin oppførsel til omgivelsene, må den kunne oppfatte sine omgivelser ved å gjøre bruk av informasjon fra egne sensorer og informasjon fra andre agenter som opererer i samme nettverk. Den må videre kunne sammenstille og tolke denne informasjon, og benytte denne ”kunnskapen” til å velge, planlegge og utføre handlinger som i størst mulig grad samsvarer med dens egne målsettinger (figur 7.1).[32]

Agentbaserte modeller bygges normalt etter et ”bottom-up”-prinsipp[33] hvor det tas utgangs-punkt i autonome individer eller grupper av individer som settes sammen i en simulerings-omgivelse hvor de kan interagere. Det at agentene kan interagere er helt sentralt i ABMS, og fører til at det kan oppstå interessante mønstre når man betrakter agentenes kollektive oppførsel på et aggregert nivå. Ulineære interaksjoner mellom agentene på liten skala kan resultere i en oppdukkende kollektiv oppførsel på stor skala (eng: emergent behaviour), som f.eks. selv

[31] Se f.eks. www.xjtek.com
[32] Agenter som innehar disse egenskapene refereres ofte til som *intelligente* agenter
[33] ABMS har sitt opphav bl.a. i studier av komplekse adaptive systemer (complex adaptive systems – CAS), se f.eks. (27)



Figur 7.1 Agent som interagerer med sine omgivelser

organisering[34] og dynamisk reorganisering for å bedre evnen til overlevelse og evnen til å utnytte/dominere sine omgivelser. Denne type oppførsel vil man normalt ikke finne i andre typer simuleringer, og nettopp derfor fremheves ABMS som godt egnet til å simulere menneskelig oppførsel i sosiale sammenhenger. I forbindelse med studier av freds- og lavintensitets-operasjoner er fokus i større grad rettet mot å representere menneskelig oppførsel og interaksjon mellom individer og grupper av individer enn mot det å representere mange fysiske entiteter på en slagmark. ABMS kan i så måte være en egnet simuleringsmetode.

Hva er så hovedforskjellene mellom agenter og objekter? Hovedforskjellen er å finne i graden av autonomi. Agenter kan på selvstendig grunnlag beslutte å endre sin oppførsel basert på hvordan de oppfatter sine omgivelser. Wooldridge (26) fremhever tre viktige egenskaper ved agenter: de kan reagere på endringer i omgivelsene, de kan ta initiativ til å endre sin oppførsel og de har sosiale evner i den forstand at de interagerer med andre agenter. Dette i motsetning til objekter, som typisk er passive, dvs. at de ikke gjør noe før de får beskjed om å aktiveres for å utføre en funksjon/tjeneste. Men til tross for klare forskjeller mellom objekter og agenter, er det ikke er noe skarpt skille mellom objektorientert og agentbasert modellering og simulering. Objektorienterte modeller vil i ulik grad ha egenskaper som kan assosieres med agentbaserte modeller. Av denne grunn vil også objektorient modellering være en naturlig måte å modellere agenter på.

7.2 Hva kan ABMS brukes til?

ABMS favner vidt – fra detaljerte simuleringsmodeller og kunstig intelligens til mer aggregerte og grovkornede simuleringsmodeller hvor man ønsker å studere sentrale egenskaper ved et system. Denne rapporten avgrenser seg til å se på anvendelser av aggregerte/grovkornede simuleringsmodeller, fordi disse anses å være mest relevante i OA-sammenheng. ABMS-modeller innenfor dette domenet refereres ofte til som *destillasjoner*, som betyr at tekniske detaljer er abstrahert for å kunne studere de essensielle delene av et problem eller en situasjon.

[34] Fra Wikipedia: "Self-organization is a process in which the internal organization of a system, normally an open system, increases in complexity without being guided or managed by an outside source" (http://en.wikipedia.org/wiki/Self-organization)

Agentbaserte destillasjoner kombineres ofte med ”data farming”[35] for å studere systemers respons på ulike kombinasjoner av parameterverdier. Project Albert[36], som ble gjennomført i regi av US Marine Corps, benyttet denne kombinasjonen i en rekke studier, se f.eks. (28), (29). Hensikten er å utforske store deler av parameter- og verdirommet for å kartlegge landskapet av mulige utfall. Fremgangsmåten benyttes ofte i forbindelse med innledende utforskende studier for bl.a. å identifisere hvilke parametere som har størst betydning for ulike utfall/konsekvenser, og for å studere ulike handlemåter i forbindelse med beslutningsstøtte. I denne typen utforskende studier vil det være interessant å finne de hyppigst forekommende utfallene, men også å studere sjeldne utfall som kan medføre store konsekvenser. ABMS og ”data farming” er m.a.o. egnet til å utforske usikre faktorer, samt til å identifisere viktige avveininger mellom parametrene i systemet.

7.3 Hva kan ABMS benyttes til i OA-sammenheng?

I forbindelse med problemstillingene knyttet til utpostscenariet, beskrevet i kapittel 3, vil ABMS ha begrenset nytteverdi fordi agentbaserte destillasjoner ikke er spesielt godt egnet til å kvantifisere ytelse og effektivitet. Men, metoden har sin styrke i å representere menneskelig faktorer, og således vil den være egnet for problemstillinger hvor menneskelig oppførsel er viktig. Under er det listet opp noen typiske problemstillinger relatert til utpostscenariet hvor ABMS kan bidra til å styrke analysene:

- Hva er kritiske suksessfaktorer for at transportene skal komme frem til riktig tid og uten tap?
- Hvilke parametere er det som har størst betydning for overlevelsen til kolonnen, og hvilke verdier på disse parameterne er det som f.eks. fører til svært dårlige resultater og svært gode resultater?
- Hvordan vil soldater og ev. sivilbefolkning reagere hvis en uønsket hendelse finner sted?
- Hvordan bør egne styrker handle hvis de blir utsatt for ulike typer angrep?
- Hvordan bør egne styrker være utrustet?
- Hva er nytten av å få informasjon om mulige motstandere/terrorister som opererer langs ruten, og når er nytten av denne informasjonen størst?

Sentralt i disse problemstillingene står agentenes kognitive prosesser, som naturlig nok er vanskelig å representere i stor grad av detalj. En av styrkene med ABMS er at man kan få frem interessant kollektiv oppførsel uten å modellere enkeltindivider i stor grad av detalj. I denne typen ulineære systemer kan kun små justeringer av verdier på de viktigste parametrene (kontrollparametrene) føre til store endringer i resultatet. Hva skal til for at resultatet skal bli vellykket, og hvilke betingelser kan lede til en katastrofe? Ved å utforske systemet vha. ABMS kan man dermed støtte opp under robuste beslutninger. På den annen side bør man også være forsiktig med å benytte output fra en ABMS til å kvantifisere ytelse og effektivitet – her bør

[35] Fra Wikipedia: ”Data farming is the process of using a high performance computer or computing grid to run a simulation thousands or millions of times accross a large parameter and value space. The result of Data Farming is a ”landscape” of output that can be analyzed for trends, anomalies, and insights in multiple parameter dimensions” (http://en.wikipedia.org/wiki/Data_farming)

[36] www.projectalbert.org

analytiske modeller eller andre simuleringsteknikker benyttes.

7.4 Verktøy

I det følgende er det gitt en kortfattet oversikt over noen verktøy som støtter ABMS.

- **MANA** (Map Aware Non-Uniform Automata) er utviklet av Defence Technology Agency, New Zealand. MANA er en simuleringsomgivelse hvor man enkelt kan sette opp scenarier og definere agenter og interaksjon mellom agenter. Oppførsel, sensorer, våpen og nettverk er representert ganske overordnet, slik at MANA er å anse som en destillasjon.
- **GAMMA** (Global Aggregated Model for Military Assessment) er utviklet for NC3A, og benyttes i planleggingsøyemed.
- **NetLogo** er en modelleringsomgivelse for å simulere komplekse naturlige og sosiale systemer. Programmet er gratis og kan lastes ned fra http://ccl.northwestern.edu/netlogo
- **Repast** (Recursive Porous Agent Simulation Toolkit) er en programmeringsomgivelse i Java. Programmet kan lastes ned fra http://repast.sourceforge.net
- **AnyLogic** er en simuleringsomgivelse som gir mulighet for å benytte flere simuleringsteknikker, deriblant ABMS, DHS og SD. Mer informasjon om AnyLogic kan finnes på http://www.xjtek.com

8 DISKUSJON

Denne rapporten gir en introduksjon til temaet simulering innen operasjonsanalyse og beskriver spesielt fire hovedtyper simuleringsmetoder, som samlet dekker et bredt spekter av metoder:

- Statisk Monte Carlo-simulering
- Diskret hendelsesstyrt simulering
- Systemdynamikk
- Agentbasert simulering

Simulering er, og vil fortsette å være, en sentral metode innen operasjonsanalyse. Imidlertid synes utviklingen innenfor militære OA-miljøer tilsvarende det vi har på FFI å gå i retning av mindre, enklere og mer problemfokuserte modeller på bekostning av mer tradisjonelle, omfattende stridsmodeller. Hovedårsaken til dette er den nye trusselsituasjonen og det brede spekteret av problemstillinger som følger med, som gjør det vanskeligere og mer ressurskrevende å lage store systemorienterte modeller med tilstrekkelig fleksibilitet til at de kan anvendes på ulike problemstillinger.[37] I tillegg er det slik at analysestøtte til reelle operasjoner og øvelser – med hurtig oppdukkende analysemål og korte tidsfrister – blir vektlagt mer i takt med økende internasjonalt engasjement for mange lands forsvar. Det vil imidlertid fremdeles

[37] Selv om gjenbruk av større simuleringsmodeller anses som mindre sannsynlig vil komponenter fra disse kunne gjenbrukes i andre simuleringsmodeller. Eksempelvis vil ferdigutviklede komponenter, som f.eks. radarer, kjøretøy og fly, kunne gjenbrukes i flere simuleringsmodeller.

være behov for større og grundigere modeller også i slike sammenhenger, og utfordringen blir å finne en god balanse og bredde i metoder og modeller som kan tilbys.

8.1 Hvorfor velge simulering?

Hvorfor velge simulering som analysemetode fremfor andre tilgjengelige metoder? Som nevnt tidligere står simulering i motsetning til (i utgangspunktet) mer nøyaktige metoder som å eksperimentere direkte med systemet eller å løse analytiske modeller av systemet. Når direkte eksperimentering med systemet er umulig eller uønsket, og matematiske modeller blir for komplekse til å løses analytisk, er simulering ofte en velegnet metode.

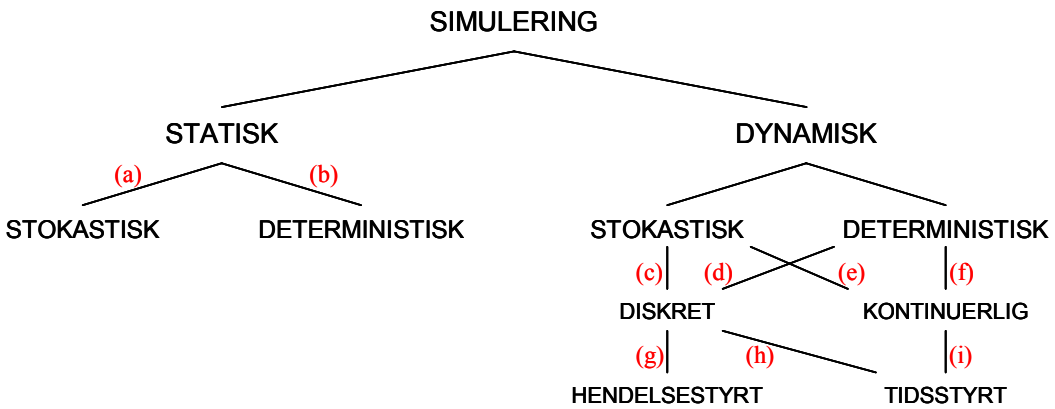
Som nevnt i kapittel 2.6 kan en annen grunn til å velge simulering rett og slett være mangel på tid eller kompetanse til å finne en eksisterende analytisk løsning. Slik pragmatisme er selvfølgelig en svakhet, men det kan gi gode nok svar på kort tid når ressursene ikke strekker til. I analysesituasjoner med korte tidsfrister – eksempelvis ved OA-støtte til pågående operasjoner – kan simulering ofte være å foretrekke foran andre og i utgangspunktet bedre metoder.

8.2 Hvordan velge simuleringsmetode?

8.2.1 Systemet velger metoden

Når man har bestemt seg for å løse et problem ved hjelp av modellering og simulering, kan det være en utfordring å velge en hensiktsmessig simuleringsmetode. Valg av metode må i første rekke være knyttet til hvordan det aktuelle problemet ser ut – eller mer korrekt: hvordan man velger å se på problemet. Problemanskuelsen definerer det aktuelle systemet, og dette systemet må beskrives langs de dimensjonene som betyr noe for valg av metode; disse er gjennomgått i kapittel 2.7, og de viktigste ble vist i figur 2.4 og gjentas her:

- Tidsaspektet: statisk – dynamisk
- Usikkerhetshåndtering: stokastisk – deterministisk
- Tilstandsendring: diskret – kontinuerlig



Figur 8.1 Sorteringsdimensjoner for simulering. Bokstavene i parentes refererer til kombinasjoner som omtales i teksten

De mest sentrale spørsmålene ved valg av metode er således:

1. Forholder systemet seg til tiden eller ikke?
2. Har systemet innslag av tilfeldigheter?
3. Krever problemet at enkeltkomponenter følges, eller er det mulig å aggregere?

Disse spørsmålene har blitt belyst fra ulike vinkler flere steder i rapporten, og nedenfor gjøres et sammendrag av diskusjonen. I det følgende refererer bokstavene i parentes til kombinasjoner i figur 8.1.

Statiske systemer som er stokastiske (*a*), studeres vha. Monte Carlo-simulering. Statiske systemer som er deterministiske (*b*), studeres sjelden ved hjelp av simulering; disse løses heller vha. andre analytiske eller numeriske metoder, f.eks. optimering.

Dynamiske systemer er det mest vanlige når det kommer til simulering. Dynamiske systemer blir ofte modellert som *enten* stokastiske og diskrete (*c*) *eller* deterministiske og kontinuerlige (*f*). Andre konstellasjoner enn disse forekommer, men er mer eller mindre uvanlig. En kort drøfting av de mulige konstellasjonene gis i det følgende.

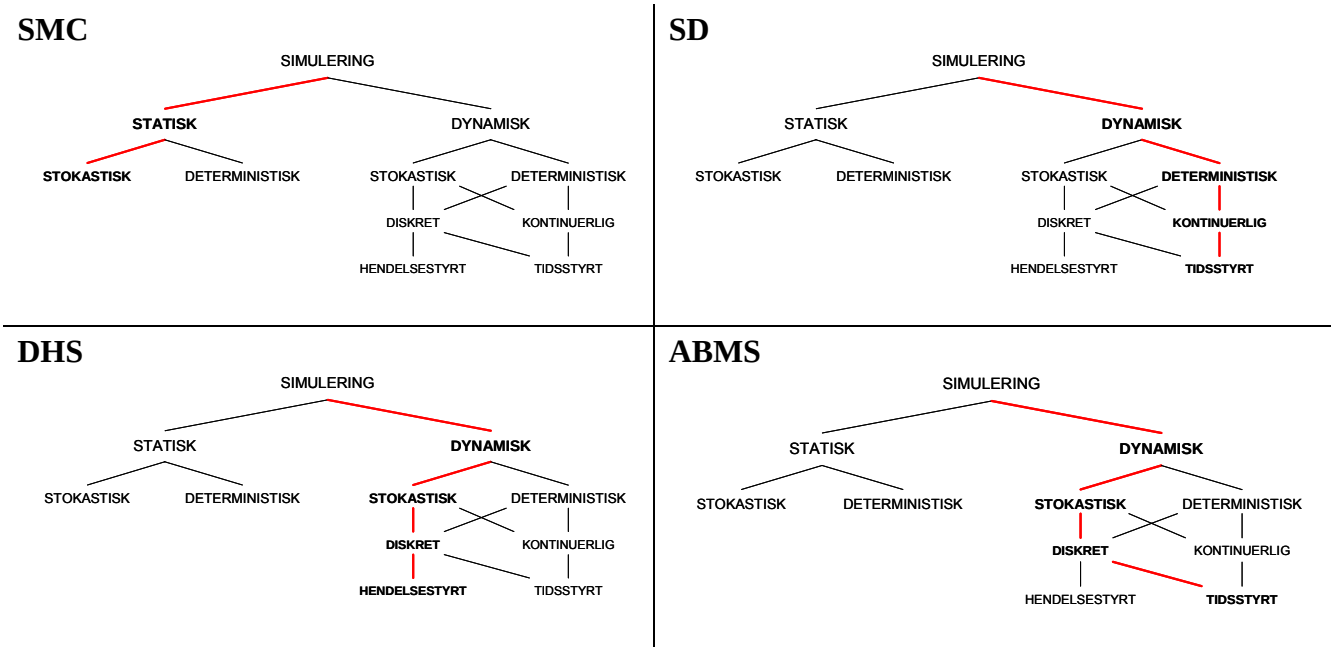
Diskrete modeller er hensiktsmessig når man ønsker å studere enkeltkomponenter i systemet, og stokastikk (*c*) er en naturlig del av enkeltkomponenters liv. Deterministiske, diskrete systemer (*d*) er også vanlig, men her gjelder det samme som for deterministiske, statiske systemer (*b*), nemlig at det gjerne finnes andre og bedre metoder enn simulering tilgjengelig.

I kontinuerlige modeller har man gjerne en overordnet tilnærming til systemet, der enkelt-elementene ses på som en helhet eller en strøm. Kontinuerlige modeller er som regel deterministiske (*f*). Årsaken til dette er at virkelighetens stokastikk oftest er koblet til hendelser på enkeltelementnivå, og representerer således en detaljgrad som ikke er nødvendig for å studere et større kollektiv av elementer. Stokastikken oppheves ved å bruke forventningsverdier for hendelsene, og dette blir uttrykt som endringsrater på makroskopisk nivå. Modellering av rater krever naturlig nok en kontinuerlig tilnærming.

Man kan selvfølgelig lage kontinuerlige modeller som også er stokastiske (*e*), men stokastiske innslag vil da av natur være diskrete, og fremstår gjerne som ”forstyrrelser” i det ellers kontinuerlige forløpet.

Den fjerde dimensjonen i figur 8.1 – tidshåndtering – er et mer simuleringsteknisk valg som avhenger mest av om systemet er diskret eller kontinuerlig. Diskrete systemer simuleres som regel med hendelsesstyring (*g*), som i DHS, mens kontinuerlige systemer simuleres med tidsstyring (*i*), som i systemdynamisk simulering. Imidlertid kan diskrete simuleringer også ha tidsstyring (*h*), noe som er tilfelle for ABMS.

De fire hovedklassene simuleringsmetoder som presenteres i denne rapporten er tegnet inn i treet med sorteringsdimensjoner i figur 8.2.



Figur 8.2 Klassifisering av de fire hovedtypene simuleringsmetoder. SMC = Statisk Monte Carlo-simulering; SD = systemdynamikk; DHS = diskret hendelsesstyrt simulering; ABMS = agentbasert simulering

8.2.2 Andre momenter

Vi har til nå skissert en fremgangsmåte for valg av metode på bakgrunn av karakteristika ved systemet. Systemet defineres som nevnt i tråd med problemanskuelsen, og det er flere andre momenter som påvirker denne problemanskuelsen, og derigjennom indirekte valget av metode. To viktige momenter er:

- **Formålet med studien**. Er problemet å utforske og forstå et fenomen, eller er man mer ute etter å gjøre kvantitative analyser av systemet? For å utforske og forstå er metoder som systemdynamikk og ABMS velegnet, mens DHS og statisk simulering er mer rettet inn mot å gjøre beregninger av et på forhånd forstått system.
- **Kjennskap (og vennskap) til simuleringsmetoder og verktøy.** Det er et kjent fenomen at mennesker prøver å løse nye problemer med verktøy de allerede kjenner, og dermed risikerer å løse problemene på en suboptimal måte. Har man kun en hammer, ser alle problemer ut som spikre – man kan riktignok hamre ned et tre, men det hadde vært bedre å bruke en sag.

Ofte bruker man grove og overordnede simuleringsmetoder til forstudier, for så å detaljsimulere med mer detaljerte metoder. Eksempelvis kan systemdynamikk eller ABMS først benyttes for å utforske og forstå kompliserte sammenhenger i et system, hvorpå DHS kan anvendes for å gjøre mer detaljerte kvantitative analyser når systemet er mer kjent og forutsigbart. Innledende simuleringsstudier kan også ha som misjon å bestemme hvilke deler av systemet man ønsker å

studere i mer detalj.

De ulike simuleringsmetodene som er i bruk ble opprinnelig utviklet for å studere spesielle problemområder. Eksempelvis ble systemdynamikk utviklet for å forstå kompliserte kausale *sammenhenger* i et system, mens DHS på sin side er mer rettet inn mot å belyse og forstå *usikkerhet* i en prosess. Begge metodene kan benyttes til begge problemstillingene, men det kan være nyttig å kjenne til metodenes opprinnelige rasjonale når man skal velge metode.

8.2.3 Erfaringer fra utpostscenariet

Den viktigste erfaringen fra utpostscenariet er at de fire simuleringsmetodene som omtales i rapporten synes å være egnet til å belyse ulike sider ved det samme problemet, men i liten grad egnet til å svare på akkurat de samme spørsmålene. Ergo utfyller metodene hverandre mer enn å være ”konkurrenter” – noe som vel rettferdiggjør både at de eksisterer som selvstendige metoder og at de beskrives individuelt i denne rapporten. Alle metodene har sin naturlige plass innen OA, og det er ingenting som tyder på at dette vil endre seg. Kjennskap til disse metodene bør derfor inngå i verktøykassen til alle operasjonsanalytikere.

Agentbasert simulering var den minst kjente av de fire metodene i forkant av denne studien. ABMS kan ses på som en avart av DHS, men også disse to metodene synes å være komplementære snarere enn overlappende. Det virker som det er en økende interesse for å benytte ABMS i analyser av militære operasjoner. ABMS oppfattes å være spesielt egnet til å studere lavintensitetskonflikter og krisehåndtering hvor det ikke benyttes store styrker og hvor målsettingen ikke nødvendigvis er å ødelegge flest mulige fiender. Utfordringene i slike operasjoner er derimot ofte knyttet til menneskelige faktorer og gruppeoppførsel i dynamiske omgivelser.

APPENDIKS

A FORKORTELSER

ABMS	-	Agentbasert modellering og simulering
CAS	-	Complex Adaptive Systems
CB	-	Crystal Ball
DHS	-	Diskret hendelsesstyrt simulering
FFI	-	Forsvarets forskningsinstitutt
FIFO	-	First In, First Out
FLYT2	-	Flytilgjengelighet- og timeproduksjonsmodell
FN	-	Forente nasjoner
FOI	-	Totalförsvarets forskningsinstitut
GAMMA	-	Global Aggregated Model for Military Assessment
GOAL	-	Grunnlagsforskning operasjonsanalyse
GUI	-	Graphical User Interface
INFORMS	-	Institute for Operations Research and the Management Sciences
MANA	-	Map Aware Non-Uniform Automata
MC	-	Monte Carlo
MCMC	-	Markov Chain Monte Carlo
MIT	-	Massachusetts Institute of Technology
MTBF	-	Mean Time Between Failures
NATO	-	North Atlantic Treaty Organisation
NC3A	-	NATO Consultation, Command and Control Agency
OA	-	Operasjonsanalyse
OR	-	Operations Research / Operational Research
Repast	-	Recursive Porous Agent Simulation Toolkit
SD	-	Systemdynamikk
SMC	-	Statisk Monte Carlo
US	-	United States

B DETALJERT SCENARIO

Nedenfor følger en spesifisering av scenariet som ble skissert i kapittel 3. Spesifikasjonen er utarbeidet hovedsakelig for diskret modellering, men blir brukt som utgangspunkt for alle simuleringsmetodene beskrevet i denne rapporten. Tallene som angis er ikke basert på konkret inputmodellering.

Generelt

- Avstanden til utposten er 1000 km
- Utposten trenger 2 tonn/døgn
- Hver lastebil tar 4 tonn nyttelast
- Det er 6 lastebiler og 4 eskortekjøretøy tilgjengelig
- Lastebiler krever eskorte; 1 lastebil trenger 1 eskortebil, mens 2, 3 eller 4 lastebiler trenger 2 eskortekjøretøy

Feil- og reparasjonsrater

Kjøretøy	MTBF [h]	MTTR [h]
Lastebil	200	5
Eskortekjøretøy	150	8

Tiden til neste feil inntreffer, er eksponentielt fordelt med parameter 1/MTBF. Tiden her gjelder kjøretid, ikke løpende tid. Det forutsettes vanlig vedlikehold når kjøretøyene er innom hovedbasen. Reparasjonstiden er eksponentielt fordelt med parameter 1/MTTR.

Lastebil

Dersom trukket reparasjonstid er mindre enn 3 h, utføres reparasjonen på stedet mens kolonnen venter. Hvis trukket reparasjonstid er større eller lik 3 h, etterlates kjøretøyet med eventuell last. Dette forsinker kolonnen en tid som er uniformt fordelt R[0,5, 1,5]. Tid for bilberging og reparasjon er eksponentielt fordelt med parameter 1/150. Etter denne tiden er lastebilen på plass i reservepoolen på hovedbasen.

Eskorte

Hvis trukket reparasjonstid er mindre enn 4 t, utføres reparasjonen på stedet mens kolonnen venter. Hvis trukket reparasjonstid er større enn 4 t etterlates eskortekjøretøyet. Dette forsinker kolonnen en tid DT som er uniformt fordelt R[0.5, 1]. Tid for berging og reparasjon er eksponentielt fordelt med parameter 1/200. Etter denne tiden er eskortekjøretøyet på plass i reservepoolen på hovedbasen.

Kjøretid

- Kolonnens kjøretid til utposten er trekantfordelt med parametere (40, 48, 72)
- Kolonnens oppholdstid på utposten er trekantfordelt med parametere (12, 16, 24)
- Kolonnens kjøretid tilbake til hovedbasen er trekantfordelt med parametere (36, 48, 60)
- Samlet hviletid, vedlikeholdstid og lastetid før neste tur er trekantfordelt med parametere (48, 60, 72)

Angrep

Med sannsynlighet p = 0,1 blir kolonnen angrepet på vei til utposten. Angrepstidspunktet er R[20, 30] etter start. Hvis minst et kjøretøy blir slått ut, vender overlevende deler av kolonnen tilbake til hovedbasen etter en konsolideringspause som er R[2, 6]. Kjøretid tilbake er den samme som på vei ut. Gitt at angrep finner sted, er sannsynligheten for at en lastebil eller et eskortekjøretøy blir slått ut hhv. 0,1 og 0,05. Utslagning av et kjøretøy er uavhengig av utslagning av de andre kjøretøyene. Hvis ingen kjøretøy blir slått ut, fortsetter alle til utposten etter en konsolideringspause som er R[2, 6].

Startbetingelser

Startsituasjonen for simuleringen: Utposten har 14 tonn forsyninger. 4 lastebiler (hver bil har 4 tonn last) og 2 eskortekjøretøy er klare til å starte turen til utposten. 2 lastebiler og 2 eskortekjøretøy er i reserve på hovedbasen.

Litteratur

(1) Gilljam M, Ljøgodt H (2006): Problem structuring methods – A survey and a case study, FFI/RAPPORT-2005/00852, Forsvarets forskningsinstitutt

(2) Malerud S, Kråkenes T (2006): Metoder for flermålsanalyse – En oversiktsstudie fra GOAL, FFI/RAPPORT-2005/03041, Forsvarets forskningsinstitutt

(3) Skjelland N E, Feet E H, Frihagen J (1994): Introduksjon til simulering, FFI/RAPPORT-94/03698, Forsvarets forskningsinstitutt

(4) Law A M, Kelton W D (2000): Simulation Modelling and Analysis, 3rd ed., McGraw-Hill

(5) Sendstad O J, Malerud S (2007): Agent Based Modelling and Simulation – Applicability within OR at FFI, FFI/RAPPORT-2007/00164, Forsvarets forskningsinstitutt

(6) Morse P M, Kimball G E (1950): Methods of Operations Research, Technology Press of MiT and John Wiley & Sons

(7) Pidd M (2003): Tools for Thinking: Modelling in Management Science, 2nd ed., John Wiley & Sons

(8) Taha, H A (2007): Operations Research – An Introduction, 8th ed., Prentice Hall

(9) Jaiswal N K (1997): Military Operations Research – Quantitative Decision Making, Kluwer Academic Publishers

(10) Rosenhead J, Mingers J (2001): Rational Analysis for a Problematic World Revisited, John Wiley & Sons

(11) Banks J, Carson II J S, Nelson B L, Nicol D M (2005): Discrete-Event System Simulation, 4th ed., Pearson Prentice Hall

(12) Holm G (2005): Modeller och simulering – Vad handlar det om egentligen? Presentasjon på FFI-seminar om modellering og simulering, 27–28 september 2005, Jeløy Radio

(13) Everitt B S (2002): The Cambridge Dictionary of Statistics, 2nd ed., Cambridge University Press

(14) Brown L (Ed.) (1993): The New Shorter Oxford English Dictionary, Clarendon Press

(15) Graasvoll O (2005): Sustainability simulations for fighter aircraft in peace and at war – A paper presented at the NMSG conference October 2005, FFI/NOTAT-2005/03849, Forsvarets forskningsinstitutt

(16) Larsen R J, Marx M L (1986): An Introduction to Mathematical Statistics and Its Applications, 2nd ed., Prentice-Hall

(17) Fishman G S (1996): Monte Carlo – Concepts, Algorithms and Applications, Springer-Verlag

(18) Rubinstein R Y (1981): Simulation and the Monte Carlo Method, Wiley

(19) Rice J A (1995): Mathematical Statistics and Data Analysis, 2nd ed., Duxbury Press

(20) Liu J S (2001): Monte Carlo Strategies in Scientific Computing, Springer-Verlag

(21) Hoff E Ø, Hennum A C (2005): Kostnader for langtrekkende presisjonsstyrte våpen – Metode og usikkerhetsbetraktninger, FFI/RAPPORT-2004/03654, Forsvarets forskningsinstitutt, Begrenset

(22) Coyle R G (1996): System Dynamics Modelling – A Practical Approach, Chapmann & Hall

(23) Sterman J D (2000): Business Dynamics – Systems Thinking and Modeling for a Complex World, McGraw Hill

(24) Østbye P R (2004): Støtte til militærfaglig utredning 2003 (MFU 03) – Utredning om befalsordning, FFI/RAPPORT-2003/01485, Forsvarets forskningsinstitutt

(25) Gilljam M (2004): Implementering av beslutningstrener – Sluttrapport for prosjekt 846, FFI/RAPPORT-2003/01446, Forsvarets forskningsinstitutt

(26) Wooldridge M (2002): An Introduction to MultiAgent Systems, John Wiley & Sons

(27) Macal C M, North M J (2005): Tutorial on Agent-Based Modeling and Simulation, Proceedings of the 2005 Winter Simulation Conference (http://www.wintersim.org)

(28) Forsyth A J, Horne G E, Upton S C (2005): Marine Corps Applications of Data Farming, Proceedings of the 2005 Winter Simulation Conference (http://www.wintersim.org)

(29) Barry P, Koehler M (2004): Simulation in Context: Using Data Farming for Decision Support, Proceedings of the 2004 Winter Simulation Conference (http://www.wintersim.org)