Systemmodellering och Simulering

med särskild inriktning mot försvarsmakten

GUNNAR HOLM

FOI-R--4457--SE

FOI

Systemmodellering och Simulering

med särskild inriktning mot försvarsmakten

GUNNAR HOLM

Systemmodellering och Simulering
med särskild inriktning mot försvarsmakten

Författare: Gunnar Holm

Bilder: Gunnar Holm där inte annat anges

Omslagsbild: Sanja Gjenero

Grafisk form: Frankfeldt Grafisk Form AB

Tryck: FOI Repro, Stockholm 2007

Första upplagan

[d^ßgßßI I JLßßhZ

ISBN 978-91-7056-124-5

Innehåll

Förord

Modellering och simulering (M&S) är ett kraftfullt verktyg inom snart sagt alla ämnesområden, för att inte säga i all mänsklig intellektuell verksamhet. Var och en har säkert en egen, mer eller mindre välgrundad, uppfattning om vad det är för någonting och kan framgångsrikt använda sig av det, men svårigheter kan uppstå i kontakten mellan olika verksamheter, vetenskaper och individer p.g.a. att man i grund och botten inte talar samma språk.

Sedan slutet av 90-talet har FOI (Totalförsvarets forskningsinstitut, tidigare FOA, Försvarets forskningsanstalt) hållit flera kurser eller kursdelar om modellering och simulering för Försvarshögskolan, där vi bl.a. försökt att ge eleverna insikter som skall hjälpa till att överbrygga dessa svårigheter och att ge dem en realistisk syn på modellers användbarhet. Behovet av ett lättillgängligt och lagom omfattande kompendium har då varit uppenbart. Föreliggande skrift är ett försök att fylla detta behov, och den har tillkommit som ett bakgrundsarbete under ett par års tid.

Målet med denna framställning är således att ge läsaren en sådan kunskap och insikt, som underlättar ett aktivt deltagande såväl i modellutvecklingsprojekt som i verksamheter där simulering med modeller utgör en viktig del. Däremot har det inte varit avsikten att här framställa något slags ”kokbok” för hur man gör modeller. För sådant finns redan god litteratur såväl generellt som specifikt för olika tillämpningsområden och vetenskaper.

Siktet har alltså varit inställt på att beskriva modellering och simulering på en inte alltför teknisk nivå men väl att skapa förståelse för hur man kan arbeta med det mycket kraftfulla verktyg som modeller och simuleringar är. Men avsikten har också varit att läsaren skall bli medveten om de svagheter verktyget har och de fallgropar som man lätt kan hamna i vid en okritisk användning. Vidare ger framställningen en översikt över olika modelltyper liksom över den begreppsapparat, som är användbar i modellerings- och simuleringsutveckling. Utgångspunkten har därför varit att förutsätta relativt komplexa modeller med inslag från fler än en enda ämnesdisciplin, s.k. systemmodeller. Det allra mesta som tas upp är dock relevant även för enklare modeller och modeller som är specifikt knutna till en enda ämnesdisciplin.

Eftersom många begrepp som används rörande modellering och simulering är allmängods men ändå, eller kanske just därför, för många är oklara till sin betydelse, läggs stor vikt vid definitioner. Ofta tas då utgångspunkten i Svenska Akademins ordbok eller Nationalencyklopedin eller, när dessa inte är tillräckligt aktuella, i andra auktoritativa verk. Dessa definitioner lyfts fram genom att presenteras i separata faktarutor. Denna presentationsform har även utnyttjats för illustrationer och annan kompletterande information.

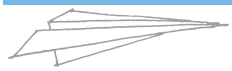
I ett par fall (rörande konceptuell modellering respektive ändliga tillståndsautomater) har dock de illustrerande exemplen blivit så omfattande, att de i stället har måst placeras i bilagor. En annan bilaga behandlar de senaste årens forskning vid FOI rörande metodik och teknik för M&S. Slutligen har det befunnits vara lämpligt med ytterligare en bilaga, som behandlar begreppet ”system”, vilket är ett mycket frekvent ord i denna skrift men för de flesta om möjligt ännu mer diffust i sin betydelse än modellering och simulering.

En förhoppning är att denna skrift skall vara användbar inte bara i Försvarshögskolans kurser utan även för forskare och andra som är intresserade av modellers användbarhet. Visserligen har varje forskare en väl fungerande arsenal av modellerings- och simuleringstekniker, som är en del av själva forskningsmetodiken inom hans eller hennes ämnesområde, men det finns ändå anledning att ibland tänka in sina modeller i ett större sammanhang. I och med att utvecklingen, inte minst inom försvarsmakten, pekar hän mot en högre grad av integration mellan olika modeller i simuleringar av större system, så blir frågor om en gemensam begreppsvärld och gemensam arkitektur betydelsefulla. Det kan då vara värt att redan då man utvecklar sina modeller se dem i ett sådant bredare perspektiv och ta vederbörlig hänsyn till att den kunskap som modellen representerar kan komma att integreras i ett större helt.

Texten är visserligen skriven av en enda person, men den skulle inte ha kunnat bli till utan ett omfattande stöd från ett stort antal kolleger. Deras samlade kunskaper, erfarenheter och undervisningsmaterial har varit till ovärderlig hjälp i arbetet. Följande har såväl ställt sitt material till förfogande som granskat den slutliga texten och lämnat värdefulla kommentarer och synpunkter: Eva Andersson, Ulla Bergsten, MajBritt Hansson, Vahid Mojtahed, Johan Pelo, Lena Sporre, Pontus Svenson och Choong-ho Yi.

Följande har därutöver granskat hela eller delar av innehållet och likaledes givit utmärkta kommentarer och uppslag till förändringar och utvidgningar: Rassul Ayani, Dirk Brade, Martin Eklöf, Marianela García Lozano, Tore Isacson, Bob Melander, Lena Sporre, Pernilla Svan, Jenny Ulriksson, Niklas Wallin och Ulla-Stina Åström.

Slutligen bör framhållas min dåvarande avdelningschefs, Monica Dahlén, och institutionschefs, Farshad Moradi, avgörande insats att tro på projektet och ställa medel till förfogande för dess förverkligande.



Modeller och simulering – vad handlar det om egentligen?

Modeller och simulering, det är begrepp som används i alla upptänkliga sammanhang. Ofta förstår vi intuitivt vad som avses, men ber man människor förklara vad de menar, så får vi nog inte en särskilt samstämmig beskrivning av vare sig modeller eller simulering.

Det är inte så konstigt, eftersom det knappast heller finns en entydig vetenskaplig definition av vad det är fråga om. Detta kapitel är till för att försöka bringa en viss ordning i begreppen och att sätta in dem i sina sammanhang.

Modell

Låt oss därför börja med följande ytterst generella definition:

En *modell* är en företeelse som representerar en annan företeelse.

Denna definition anger att en modell alltså är snarlik en **association**, ”en idémässig förbindelse mellan olika föreställningar”. Likaså finns det en likhet med begreppet **symbol**, en ”företeelse som på ett koncentrerat sätt ger uttryck för viss annan företeelse mer eller mindre tydligt”. Man kan också säga, att modeller är människans naturliga verktyg för att förstå sig själv och sin omvärld.

1

Ur Svenska Akademins ordbok (1945):

Modell

1. **BETYDELSE:** föremål som (vanl. i förminskad skala) visar utseendet (o. funktionen) hos ngt (t. ex. hos en byggnad l. en maskin)

   REDAKTIONSEXEMPEL: *En modell av, förr äv. på ngt, dels om (miniatyr)kopia av ngt, dels (vanl. i uttr. modell till ngt) om föremål avsett att tjäna ss. mönster vid framställningen av ngt; mönster, förebild; äv.: utkast, skiss.*

2. **BETYDELSE:** förebild, föredöme, mönster.
3. **BETYDELSE:** om ngn l. ngt som är föremål för avbildning.
4. **BETYDELSE:** om för en kortare tid gällande bruk i fråga om snitt, färg o. d. på kläder l. frisyr o. d.; (av gällande smak föreskrivet) sätt att kläda sig osv.

Ur Bonniers lexikon (1965):

Modell

2. Medvetet förenklad beskrivning av en komplicerad företeelse (t.ex. atommodell).

Ur Nationalencyklopedins ordbok (1996):

modell’ subst. *–en. -er*

1. förebild för framställning med större el. mindre grad av imitation
2. person som avbildas konstnärligt
3. föremål som efterbildar annat föremål vanl. i förminskad skala; för studier, lek etc.
4. förenklat, åskådligt tankeschema l. beskrivning av komplicerad abstrakt företeelse

I faktaruta 1 ges några exempel på vedertagna betydelser, och den spännvidd som i det allmänna språkbruket ligger i ordet ”modell”. Men man kan också se hur en av betydelsevarianterna har utvecklats från 1945 (alt. 1) via 1965 (alt. 2) och 1996 (alt. 3 och 4) till något som låter sig generaliseras i enlighet med vår definition.

Modeller kan vara både konkreta och abstrakta, men det gemensamma är att de utgör förenklingar av den verklighet som de representerar. Dessa förenklingar består i att modellen har ett urval av de egenskaper, som karaktäriserar den modellerade företeelsen, och utelämnar andra. Samtidigt kan modellen i sig ha egenskaper som den modellerade företeelsen inte har. Gemensamt för de utelämnade och de tillförda egenskaperna är dock att de är irrelevanta (”bakgrundsbrus”) i det sammanhang där modellen används.

Barn är tidigt duktiga på att använda modeller. En grankotte och fyra trästickor i lämplig kombination ”blir” en ko, en samling legobitar lämpligt sammansatta ”blir” ett hus, en stad, en gubbe, en helikopter eller vad som helst annat, som faller den lekande i sinnet. Själva leken som sådan är också en slags modell av vuxenlivet, så som barnen förstår det.

Kotten och stickorna är ett bra exempel på en konkret modell, där man gjort en väsentlig förenkling. De egenskaper som spelar roll är att en ko har fyra ben, långsträckt kropp och att den fordrar ett visst utrymme i kätten eller på ängen. Däremot bortser man medvetet från att en ko har fyra magar och horn, att den har ett råmande läte och är 550 kg tung. Skulle man av någon anledning behöva ta hänsyn till att det finns ett huvud, så kan man ju utvidga sin modell med hjälp av ytterligare en sticka och en tallkotte. Komodellen har emellertid inte enbart reducerat antalet egenskaper jämfört med den riktiga kon, den har också tillfört egna egenskaper, som knappast är relevanta för förståelsen av vad en ko är. Det tydligaste exemplet är att modellkroppen har en fjällig yta, vilket inte har någon motsvarighet i den riktiga kons hud. Vill man då använda modellen för att förstå hur det känns att smeka en ko, så leder modellen till helt fel slutsats.

Man kan också se detta förhållande som ett exempel på att en konkret modell ofta i sig själv kan vara något reellt med ett självständigt existensberättigande. En kotte är ju fr.a. till för fröproduktion. Nu är ju det något helt annat än en komodell, och de båda aspekterna är ju inte så svåra att skilja åt, men i många fall kan man behöva vara lite extra uppmärksam på skillnaden mellan modellobjektets reella och modellmässiga egenskaper och funktioner.

Men en modell behöver inte vara ett konkret föremål som vår grankotte med fyra stickor. Vi bär alla med oss en intuitiv föreställning om vad en ko är. Just själva symbolen (ordet) ko ger upphov till att vi aktiverar vår personliga modell. Våra egna intressen liksom situationen vi befinner oss i medför att vi gör ett urval av de egenskaper som vi förknippar med en ko, vilka bäst passar in i situationen: en fridfullt idisslande varelse på en sommaräng, en hornprydd stångande best, en produktionsenhet för mjölk, en specifik odör eller varför inte entrecôte och biff.

Men även den modellerade företeelsen kan naturligtvis vara abstrakt. Tänk t.ex. på hur marknadskrafterna påverkar priset på olika varor och tjänster, eller hur beslutsprocessen i en bataljonsstab går till. De som arbetar med sådana saker eller som påverkas av dem har säkert en intuitiv, eller rent av en välformulerad, modell av dessa fenomen. Vi har alltså en abstrakt modell av en abstrakt företeelse. Kan man då

också (för symmetrins skull) tänka sig konkreta modeller av abstrakta företeelser? Ja, faktum är att det ursprungligen var mycket vanligt och närmast önskvärt att åskådliggöra alla vetenskapliga teorier mekaniskt i form av en ”modell av trä och järn”. Exempelvis tillverkade en amerikansk ekonom i slutet av 1800-talet en hydraulisk maskin, som på ett pedagogiskt sätt åskådliggjorde just pris- och marknadsmekanismens funktion.

På tal om vetenskapliga teorier så är de naturligtvis också modeller. Ibland görs det dock en distinktion mellan teori och modell på så sätt att modeller anger att det är fråga om en förenklad representation, som inte i alla stycken ger en riktig bild av verkligheten, medan man med teori avser något heltäckande. Huruvida denna distinktion är befogad eller ej kan diskuteras. Man skulle nog kunna hävda att teori är ett specialfall av det mer generella modellbegreppet.

Andra exempel på modeller är vår bild av oss själva som individer och som grupp liksom vår bild av våra medmänniskor. Så länge vi endast har en ytlig kontakt med en annan individ har vi också en ganska enkel modell av vederbörande baserad på utseende och allmänt socialt beteende. Allteftersom man lär känna personen genom egen erfarenhet eller genom andras utsagor så modifieras modellen och blir mer mångfacetterad. Under alla stadier är det vår aktuella modell, som styr våra förväntningar på och beteende mot en annan person. Om vi inte låter modellen utvecklas, kan vi med rätta anklagas för att vara styrda av fördomar; ”om inte kartan och verkligheten stämmer överens, så gäller kartan”.

Vi skapar oss alltså modeller för att systematisera vår uppfattning om omvärlden och vår position i den. De är också viktiga instrument för att planera våra handlingar och för att prognostisera utfallet av dem.

Kontentan av vår diskussion är att modellering är en mänsklig mental aktivitet. Modellen uppstår i människans hjärna. Kottar och stickor blir inte kor förrän en människa ser analogin mellan de olika företeelserna. Utan denna mänskliga reflektion existerar helt enkelt inte modellen i egenskap av att vara just en modell.

De allra flesta modeller som vi använder är intuitiva och mer eller mindre oreflekterade, och de fungerar i allmänhet väl för sina syften; vi lyckas ju trots allt bete oss på ett någorlunda ändamålsenligt sätt i de flesta av livets situationer utan att behöva tänka så mycket på det i detalj. Men ibland räcker det inte med det, vi behöver kunna förklara vad, hur och varför vi gör eller tror vissa saker, t.ex. när man behöver skapa en samsyn med andra människor för att uppnå bättre effekter än man kan göra ensam, eller om man vill undervisa någon annan om den samlade erfarenhet man bär med sig. Då behöver vi kunna sätta ord på vår modell och dessutom se till att den (förhoppningsvis) utgör en logiskt sammanhängande tankebyggnad. Därigenom får vi en kommunicerbar modell, som andra kan ta del av, kritisera, använda och bygga vidare på.

Med en kommunicerbar modell kan man bättre förstå och beskriva hur man kommit fram till sina slutsatser. Ännu effektivare blir det om den dessutom kan uttryckas i matematiskt språk. Modellen som sådan blir inte bara mera precist uttryckt, användningen blir också mera kraftfull eftersom resultaten är beräkningsbara, vilket ger dem en avsevärt större precision. I stället för att resonemangsvis komma fram till ett tänkbart utfall, kan man nu kvantitativt bestämma utfallet och hur känsligt det är för osäkerheter i beskrivningen av utgångsläget. Ännu

bättre blir modellen om man kan uttrycka den i algoritmiskt språk, d.v.s. programmera den för en dator. Med moderna datorers stora beräknings- och minneskapacitet kan man då utnyttja modeller som är både mycket detaljrika och beräkningskomplexa och därigenom erhålla oerhört mycket mera precisa och långtgående resultat (och en djupare förståelse för det modellerade systemet) än tidigare. Detta är dock inte alltid en fördel, för det medför också att modellerna kan bli mindre överskådliga och därmed sämre redskap för att skapa förståelse av de modellerade fenomenen.

Man kan då fråga sig, om det datorprogram man utvecklar är samma sak som den ursprungliga mentala modell som vi började med. Sannolikt är det inte så. I den successiva processen fram till en algoritmisk beskrivning har många inskränkningar och preciseringar gjorts, vilket gör att vi skulle kunna säga att

en *programmerad modell* är en konkretiserad avbildning av en mental modell,

eller varför inte: en programmerad modell är en *modell* av den mentala modellen. Detta senare uttryckssätt kan dock vara en smula förvillande, eftersom den programmerade modellen inte nödvändigtvis upplevs som en *förenklad* version av den mentala modellen (jfr de definitioner som presenteras i faktaruta 1).

I fortsättningen kommer vi att, om inget annat sägs, med modell avse en programmerad modell. Det är dock lämpligt att under den fortsatta läsningen ha det mer generella resonemanget om modeller i minnet.

Valet av grundobjekt

Låt oss nu betrakta modellbegreppet ur en annan synvinkel. En modell kan ses som ett (konkret eller abstrakt) system av växelverkande objekt, t.ex. ett antal stridsvagnar invecklade i strid, eller producenter, konsumenter, råvaror och tullhinder, som styr produktion och prissättning. Men vad är egentligen dessa objekt? Jo, vid närmare betraktande inser man att de i nästan alla fall själva är system av växelverkande objekt. En producent kan t.ex. bestå av en produktionsanläggning, en styrelse, en VD, en ekonomifunktion och ett antal aktieägare. Samspelet mellan dem bestämmer hur objektet producent beter sig visavi övriga objekt på marknaden.

Men dessa mer elementära objekt är ju också system av samverkande objekt, och så kan vi fortsätta att dela upp vår modell i allt finare delar. Jämför med naturvetenskapen, där vi kan tänka oss en successivt ökande detaljrikedom genom att betrakta biologiska system som uppbyggda av kemiska system som i sin tur är uppbyggda av fysikaliska system, exempelvis djur – organ – cell – cellkärna – molekyl – atom – proton – kvark – sträng (?) – ?

Om vi i alla lägen behöver göra våra modeller så detaljerade, att ett djur beskrivs av alla dess elementarpartiklar, då har vi knappast vunnit något. Frågan är om vi ens längre kan tala om en modell, vi har snarare skapat en kopia. Någon förenkling, som bidrar till en bättre förståelse, blir det då inte.

Det gäller alltså att finna en adekvat abstraktionsnivå för sin modell. Vad den är, beror på vilka problem som modellen skall bidra till att lösa eller vilka handlingar vi använder modellen till att planera för. De grundobjekt, eller atomära objekt, som representerar den högsta detaljeringsgraden i modellen måste ha väl förstådda yttre egenskaper, medan de inre egenskaperna inte har någon betydelse för samspelet med modellens övriga objekt (eller rättare: finns sammanfattade i beskrivningen av de yttre egenskaperna).

Om vi exempelvis skall skapa en modell av vad som händer i en travtävling (ur publikens perspektiv), så är det tämligen poänglöst att beskriva varje häst i detalj vad avser muskelmassa, kroppsform, matsmältning, pendlingslängd på benen, träningsdos, beteendet visavi andra hästar och yttre störningar m.m. Vi kan förvisso tänka oss att göra en sådan modell, men den blir med nödvändighet komplex, svårgripbar och sannolikt även tidskrävande att göra beräkningar med. Vi behöver i stället en modell där hästarna snarare än deras delar är grundobjekt och där de beskrivs med en lagom mängd relevanta och begripliga egenskaper, t.ex. snabbhet, uthållighet och benägenhet att galoppera.

Dessa aggregerade egenskaper hos hästarna kan man i regel få fram med observationer och statistik, men man skulle också kunna tänka sig att man härleder dem ur en detaljerad modell av hästens anatomi och arvsanlag. Det är i själva verket mycket vanligt att man under utvecklingen av en modell bestämmer sig för vissa grundobjekt, vars yttre egenskaper man uppfattar sig ha en god uppfattning om. Medan arbetet fortskrider uppstår dock frågeställningar, som för att kunna besvaras fordrar en ännu bättre förståelse, och då tvingas man kanske att öka detaljeringsgraden. Det kan naturligtvis ske i den tilltänkta modellen som sådan, men det leder lätt till en ökad komplexitet och därmed mer svårtolkade resultat liksom ökade svårigheter att få fram tillräckligt tillförlitliga data, som beskriver objektens egenskaper. Bättre är oftast att behålla de grundobjekt man bestämt sig för (behålla modellens ”**aggregeringsnivå**”), och i stället skapa den nödvändiga kunskapen om deras egenskaper med hjälp av en fristående modell av en enstaka häst.

Simulering

Ett ord som är nära besläktat med modellering är simulering, och även här råder en viss oklarhet om vad man egentligen menar. Nationalencyklopedins ordbok beskriver det som att ”efterbilda förloppet eller funktionssättet hos ngt sammansatt skeende e.d.” (se faktaruta 2). Man skulle således också kunna säga att simulering är en modell av en process, någon företeelse som utvecklas över en tidsperiod.

Om vi erinrar oss den hydrauliska ekonomimodell, som vi nämnde tidigare, så kan vi föreställa oss att, då vi kör en sådan maskin, få se hur tillståndet i det ekonomiska systemet förändras med tiden. Kanske börjar vi då fun-

2

Ur Svenska Akademins ordbok (1967):

Simulera

1. **BETYDELSE:** gm sitt uppträdande (handlingar l. åtbörder l. minspel l. yttranden o. d.) söka framkalla ett intryck som icke är i överensstämmelse med det verkliga förhållandet.
2. **BETYDELSE:** i fråga om att (vid vetenskaplig undersökning o. d.) utbyta en i verkligheten svåråtkomlig faktor l. variabel mot en lättare tillgänglig l. billigare (i sht vid statistiska beräkningar med analogimaskin, dvs. maskin varmed förhållanden analoga med verklighetens kunna imiteras): imitera l. efterlikna (situation o. d.).

Ur Bonniers lexikon (1966):

Simulering

2. Förfarande varvid man i st.f. att observera ett verkligt förlopp studerar en modell därav med hjälp av en datamaskin. S. används för studium av kö- och lagerproblem, trafikproblem, neutronspridning m.m.

Ur Nationalencyklopedins ordbok (1996):

simule´ra verb *~de ~t*

1. försöka ge sken av visst (negativt) tillstånd, särsk. sjukdom el. skada men äv. allmännare
2. efterbilda förloppet eller funktionssättet hos ngt sammansatt skeende e.d.

dera på hur det skulle se ut om man förändrade vissa antaganden om hur olika faktorer påverkar varandra, så vi ställer om några ventiler i maskinen och kör den på nytt. Kanske kan vi finna inställningar som får maskinen att visa upp samma beteende som vi kunnat observera i levande livet. När vi gör så, kommer vi successivt att få en bättre och bättre förståelse inte bara för hur maskinen fungerar utan också hur den ekonomiska process som den modellerar fungerar. Simulering kan därför sägas vara en sorts experiment, som, oavsett om det sker på lek eller allvar, medför en ökad kunskap om och förmåga att hantera en viss företeelse och dess förändring över tiden.

Sammanfattningsvis får vi alltså följande definition:

En *simulering* är ett experiment med en modell för att följa och förstå beteenden och orsakssammanhang i den modellerade företeelsen under ett tidsförlopp.

Man kan då fråga sig om alla modeller är lämpade för simulering. Svaret bör vara nej, eftersom de modeller som inte har tiden som en viktig komponent knappast kan komma i fråga. I vissa fall kan man dock se att ordet används även för beräkningar utan tidskomponent, men vi skall här hålla oss till den angivna definitionen. De modeller som därmed kommer i fråga kallar vi **simuleringsmodeller**. Övriga modeller skulle då kunna kallas (allmänna) **beräkningsmodeller**.

En (datoriserad) simuleringsmodell kan idealt sett tillhöra en av tre klasser. Den kan vara **sluten**, vilket innebär att operatören ger indata i början, sätter igång simuleringen och sedan inte kan göra något mer förrän resultatet föreligger. Vill man följa och förstå förloppet i en sådan simulering, så bör den generera en loggfil med väsentlig information om det som skett. Detta kan kombineras med att förloppet spelas upp och illustreras på en bildskärm under simuleringens gång eller i en återuppspelning i efterhand.

Simuleringsmodellen kan också vara **brytbar**, vilket innebär att operatören kan stoppa simuleringen för att sätta om parametrar och sedan låta den fortsätta under de nya premisserna. Det förutsätter att operatören kan följa förloppet, något som vanligen presenteras i grafisk form.

Slutligen kan en simuleringsmodell vara **interaktiv**. Medan de båda tidigare formerna innebär att simuleringen kan innefatta automatiska beslut, som modellerar mänskligt beteende i någon form, så förutsätter en interaktiv simulering aktiv medverkan av en eller flera mänskliga operatörer. Det är exempelvis inte så meningsfullt med en utbildningssimulator för flygare om den körs utan aktiva piloter, där händer knappast någonting av intresse.

Liksom för slutna simuleringsmodeller gäller för både brytbara och interaktiva att man bör spela in en loggfil av allt som händer inklusive de beslut som tas. Det är nödvändigt för dokumentationen av vad man har gjort och för analysen av resultaten, vare sig den sker genom statistisk analys eller genom återuppspelning av förloppet.

Vi har nu kommit fram till ytterligare ett begrepp, som kan behöva definieras, nämligen simulator, och vi gör det på enklast tänkbara sätt:

En *simulator* är en interaktiv simuleringsmodell.

För den som är van vid stora anläggningar med hydraulstyrda och högtalarförsedda operatörsplatser med bildprojektion av omvärlden varvet

3

Ur Svenska Akademins ordbok (1967):

Simulator

1. **BETYDELSE:** motsv. **SIMULERA 1:** simulant.
2. **BETYDELSE:** (i fackspr., i sht *tekn.)* motsv. **SIMULERA 2:** apparat varmed ngt imiteras l. efterliknas; särsk.: modell l. analog l. exakt kopia av ngt (i sht av invecklad l. dyrbar maskin l. apparat), med vars hjälp en person kan försättas i situationer fullt analoga med verklighetens o. varmed (till mindre kostnad o. under betryggande former) träning l. utbildning l. experimentell forskning sker.

Ur Nationalencyklopedin (1995):

simula'tor (en modern bildning av lat. *si'mulo* 'efterhärma', 'låtsa'), apparat eller anläggning som helt eller delvis efterliknar komplicerade händelseförlopp och maskiner i samspel med människor. En simulator kan direkt upplevas och påverkas på samma sätt som sin förebild av förare eller operatör. Simulatorer används i utbildning, forskning och utveckling för att renodla delfunktioner, för att minska kostnader och skaderisker i jämförelse med sådan verksamhet med det verkliga systemet eller för att få god kontroll och repeterbarhet.

runt, så ter sig kanske denna definition lite för knapphändig. Om man tänker efter så inser man emellertid att hela anläggningen i sig är en modell som är tänkt att efterlikna ett visst system på ett sådant sätt att det för den deltagande människan ter sig så verklighetstroget som möjligt. Vi har kort sagt en modell som tar fasta på de egenskaper som är väsentliga för sinnesintrycken och att koordinationen mellan åtgärder och sinnesintryck skall återges så noggrant som möjligt. Anläggningen i sin helhet är alltså en simuleringsmodell, låt vara att den delvis är en ganska konkret modell.

Det förtjänar också att påpekas att om man i t.ex. en flygsimulator ersätter operatören med en modell av en pilot, så kan systemet kanske fungera och ge ifrån sig adekvata simuleringsresultat utan mänsklig inblandning. Detta ändrar dock inte simulatorns egenskap att vara en simulator, däremot blir *kombinationen* av simulatorn och pilotmodellen en sluten eller brytbar simuleringsmodell.

Vi kan gå ytterligare ett steg och fråga oss om en simulator alltid behöver vara avsedd för interaktion med människor. Kan begreppet rent av användas även om interaktionen avser något annat reellt objekt? Ja, det händer att det brukas i denna vidare betydelse, exempelvis kan man ibland få höra en virtuell provbänk för integrerade kretsar beskrivas som en simulator.

Men är då verkligen varje interaktiv simuleringsmodell också en simulator? Nej, inte om man i ordet simulator vill lägga in förmågan att dupera våra sinnen. Detta är dock inte alltid en väsentlig egenskap. Tänk t.ex. på en schackspelande dator! Vi är nog benägna att kalla den simulator även om man måste knappa in sina drag via ett tangentbord och avläsa datorns drag på en bildskärm, och inte inskränka ordet till den maskin som kan föra spelet även mekaniskt.

I detta sammanhang kan det finnas skäl att beröra den uppdelning av simuleringsmodeller, som är vanlig i NATO och andra USA-inspirerade kretsar. Vi talar då om *live*, *virtual* resp. *constructive simulation*. **Live simulation** (simulering i levande livet) avser utnyttjandet av verklig omvärld och materiel liksom mänskliga beslutsfattare, eventuellt understött av datorbaserade delmodeller för skottindikering och verkansavdömning, t.ex. en krigsförbandsövning. **Virtual simulation** (virtuell simulering) avser utnyttjandet av en modellerad omvärld och mänskliga beslutsfattare, d.v.s. en körning med det vi nyss kallade simulator. **Constructive simulation** (utvecklad simulering) avser utnyttjandet av en modellerad omvärld och modellerat beslutsfattande. Terminologin på svenska har tenderat till att kalla den senare

typen för ”konstruktiv” simulering enligt ett icke helt ovanligt mönster för översättningar av nya uttryck. Utvecklad, eller möjligen fulländad, simulering ligger dock närmare ordets egentliga betydelse i detta sammanhang.

Låt oss slutligen fundera över ytterligare ett par närbesläktade begrepp. Det första är **animering**, som kan betyda ”ge illusion av liv” [Nationalencyklopedins ordbok, 1995]. Det är alltså fråga om en modellering av en människas, ett djurs eller något annat objekts rörelsemönster och beteende över tiden, så att simuleringen (t.ex. uppspelningen av en tecknad film) får oss att uppfatta skeendet som ”naturligt”. Det andra är **emulering**. Därmed avses en teknik att få t.ex. en dator att efterlikna en annan med avvikande instruktionsrepertoar. Därigenom kan program som utvecklats på den ena datorn också köras på den andra. Både animering och emulering är alltså i grunden simulering, om än tillämpade på speciella områden.

Grundläggande modellbegrepp

Låt oss nu återknyta till vår diskussion om modeller som ett system av växelverkande objekt!

De **objekt**, grundobjekt eller sammansatta objekt, som behöver finnas med explicit i en modell avgörs naturligtvis av det problem som är för handen och som man behöver förstå och kunna lösa. En annan vanlig term för modellobjekt är **entitet**, som dock för det mesta förefaller vara reserverat för grundobjekt.

Objektens egenskaper beskrivs med ett eller flera **attribut**, vilka kan vara statiska eller variabla. De variabla, som alltså kan anta olika värden under en simulering med modellen, utgör objektets tillståndsvariabler. De värden som de tillsammans har vid en viss tidpunkt utgör objektets **tillstånd** vid denna tidpunkt. Man kan också tala om modellsystemets tillstånd, vilket i princip utgörs av unionen av tillstånden för alla ingående objekt. Ett annat sätt att uttrycka detta är, att modellsystemets tillstånd utgörs av all den modellinformation som behövs för att beskriva systemet vid en given tidpunkt.

Ytterligare ett begrepp behöver vi göra klart för oss, och det är **händelse**. En händelse är när systemets tillstånd förändras, d.v.s. minst en tillståndsvariabel får ett nytt värde. Detta sker ögonblickligt vid en viss tidpunkt, som brukar anges med en s.k. **tidsstämpel**. Det är därför inte brukligt att använda händelsebegreppet i samband med kontinuerliga förändringar, t.ex. positionen för en bil som färdas längs en väg. Möjligen kan man tänka sig att en hastighetsändring kan gestaltas som en händelse, eller kanske hellre en accelerationsförändring.

Klassificering av modellegenskaper

Låt oss nu börja orientera oss bland de principiella tekniker som står till buds vid utveckling av modeller. Vi kan då göra oss en ”karta” över teknikrymden genom att definiera dess dimensioner med ett antal motsatspar. Det här låter kanske ganska abstrakt och krångligt, men om vi i stället säger att vi skall skapa oss en modell av hur (datoriserade) modeller är uppbyggda, så känns det säkert mera bekant. Vi skall därför beskriva vilka objektegenskaper som är relevanta i denna modell, där objekten består av, ja just det, modeller. För att komma undan problemen som kan uppstå på grund av denna självrefererande terminologi, så kallar man ibland en sådan modell för en **metamodell**. Låt oss då titta närmare på de olika modellegenskaperna!

Analytisk eller laborativ modell?

Den första egenskapen är kanske den minst relevanta. I själva verket handlar frågan om huruvida (den matematiskt formulerade) modellen är tillräckligt enkel för att man med vanliga analytiska metoder skall kunna räkna ut svaret på de flesta (om än inte nödvändigtvis *alla*) frågor man kan ha kring modellen. Om så är fallet kan vi kalla modellen analytisk. Är å andra sidan modellen för komplex för att kunna hanteras på detta sätt, måste vi skaffa kunskapen på annat sätt. Vi börjar då experimentera med modellen, sätta in olika indata och se vilket utfall vi får. Men det är ju just det som utmärker en simuleringsmodell, åtminstone om tiden är inblandad. Därmed är dock inte sagt att man inte skulle kunna använda även den enkla, analytiska modellen på samma sätt, men det skulle i de flesta fall vara ganska poänglöst.

Låt oss illustrera detta med ett exempel! Antag att vi har ett fordon som färdas fram med jämn hastighet längs en väg, där inga störningar förekommer. Vill man modellera fordonets position, så kan man göra det genom en mycket enkel formel:

$$s(t) = s_0 + v \cdot t$$

vilket ger oss avståndet längs vägen från en given nollpunkt till den punkt fordonet befinner sig vid tiden t, då det färdats med hastigheten v från startpunkten s_0. Det är nu ingen större konst att räkna ut t.ex. var fordonet befinner sig om det startade 20 km från vägens nollpunkt och har kört med 70 km/h i 4 timmar. Man kan också lätt besvara frågor av typen: Hur fort måste man köra för att vara vid den punkt som ligger 420 km från nollpunkten inom 5 timmar? Eller: När är man framme om man i stället kör med 75 km/h?

Antag nu i stället att man inte kan färdas störningsfritt längs vägen! Där finns korsande trafik, kanske trafikljus, långsamma fordon i den egna körriktningen och tät mötande trafik, kanske en trafikolycka någonstans, som stör rytmen o.s.v. För varje sådan företeelse kan man nog införa en matematiskt beskriven delmodell, men det troliga är att modellen totalt sett blir ganska komplicerad. Att nu med analytiska metoder försöka besvara de frågor, som vi nyss så enkelt kunde lösa, blir endast i undantagsfall möjligt. Nu är det bara simulering som gäller, modellen är alltså mera laborativ än analytisk.

4

Monte Carlo-metoden

Denna kan beskrivas som en konstgjord samplingsmetod, som lämpar sig såväl för att lösa vissa komplicerade problem i analytisk form som för att genom simulering lösa statistiska problem. Metoden går ut på att man genomför en beräkning / simulering ett stort antal gånger (partier), där man låter en eller flera parametrar variera slumpmässigt från den ena gången till den andra. Resultatet av varje sådan omgång blir alltså en slumpmässig storhet, och totala mängden resultat efter samtliga partier är ett stickprov på denna storhet. Detta stickprov kan sedan behandlas på sedvanligt sätt med statistisk analys.

Matematiskt kan metoden beskrivas sålunda: Om variabeln η, vars statistiska fördelning sökes, är en funktion av ett antal stokastiska variabler $\xi_1, \xi_2, \ldots \xi_n$, så att $\eta = f(\xi_1, \xi_2, \ldots \xi_n)$ kan ett stickprov på η erhållas genom följande metod. Ett slumptal genereras för varje variabel, $r_1, r_2, \ldots r_n \in [0,1]$. Dessa betraktas som värden på fördelningsfunktionen för resp. variabel, $r_i = F_i(x_i)$, $i = 1, 2, \ldots n$. Genom att lösa dessa ekvationer får vi fram $x_1, x_2, \ldots x_n$. Därigenom har en uppsättning slumpmässigt valda värden på variablerna $\xi_1, \xi_2, \ldots \xi_n$ bestämts och det första slumpmässiga värdet på funktionen $y_1 = f(x_1, x_2, \ldots x_n)$ kan beräknas. Sedan upprepas proceduren ett stort antal gånger och vi får ett stickprov $y_1, y_2, \ldots y_m$ av $\eta = f(\xi_1, \xi_2, \ldots \xi_n)$.

Observera att metoden kan användas för såväl stokastiska problem som deterministiska. Den används mycket för att beräkna resultaten av förlopp, som också i verkligheten är stokastiska. Men även ett typiskt deterministiskt problem som bestämning av π, kan lösas genom att undersöka hur stor andel av ett stort antal slumpmässigt valda punkter i en kvadrat, som faller inom den inskrivna cirkeln. Se också integralexemplet i nästa ruta!

Det förtjänar dock i detta sammanhang att påpekas, att simulering är den minst effektiva av alla lösningsmetoder, den som man bör ta till först då analytiska och numeriska möjligheter är uttömda eller alltför ohanterliga.

Statisk eller dynamisk modell?

Nästa modellegenskap är intressantare att diskutera. Med en statisk modell menar vi en modell där tiden inte utgör en väsentlig variabel. Det sker alltså inga tillståndsförändringar över de tidsperioder för vilka beräkningar med modellen normalt utförs. En dynamisk modell är däremot en modell, där objekten ändrar tillstånd över tiden. Låt oss t.ex. göra en modell över ett fjällandskap, där vi fr.a. är intresserade av hur bergsprofilen ser ut för att kunna studera frisiktsaspekter för luftbevakningen. Några förändringar av profilen är knappast att förvänta under ett stridsförlopp, kanske inte ens under ett människoliv, såvida man inte bor i ett seismiskt mycket aktivt område. Vi gör därför en statisk modell av vår terräng. Skulle vi nu i stället vara geologer, så skulle vi kanske i stället välja att göra vår modell dynamisk för att kunna studera hur bergveckningsprocessen har förändrat landskapet under årmillionerna.

Att modellera just terrängen statiskt har varit gängse metod i de flesta modeller i militära sammanhang under lång tid. Under senare år har man dock genom den starkt ökade datorkapaciteten i högre grad kunnat utveckla och utnyttja dynamiska terrängmodeller, där stridsaktiviteterna sätter påtagliga spår i form av kratrar o.dyl. Detta visar att valet av statisk eller dynamisk modell i hög grad också är en fråga om vilken detaljeringsgrad man vill ha på modellen.

Kan en statisk modell vara en simuleringsmodell? Enligt vår definition av simulering, så skall den inte kunna vara det. Den kan dock ingå som en delmodell i en större modell, som är dynamisk. Likaså förekommer det att man ibland kallar beräkningsförfarandet med rent statiska modeller för simulering. Det gäller t.ex. då man löser en integral med Monte Carlo-teknik (se faktarutorna 4 och 5).

5

Integralberäkning m.hj.a. ”simulering”

Antag att man söker värdet av den bestämda integralen

$$I = \int_a^b g(x)dx$$

där g(x) är en reellvärd funktion, som inte är analytiskt integrerbar. Man kan då dra nytta av att I kan beräknas som ett statistiskt väntevärde:

$$I = E[(b-a)\cdot g(\xi)]$$

där ξ är en stokastisk variabel, som är likformigt fördelad i intervallet [a, b]. Genom att dra ett stort antal slumptal ($\xi_1, \xi_2, \ldots \xi_n$) ur denna fördelning kan vi få en skattning av väntevärdet enligt

$$\bar{I} = \frac{b-a}{n} \sum_{i=1}^{n} g(\xi_i)$$

Deterministisk eller stokastisk modell?

Detta leder oss in på nästa modellegenskap, som har att göra med slumpmässighet. I många fall är den verklighet som skall modelleras inte möjlig att beskriva med helt entydiga orsakssammanhang, utan där finns inslag av slumpartade inflytelser som t.ex. tillståndet i en radioaktiv atomkärna, utfallet av ett tärningskast eller antalet människor, som vid en viss tidpunkt befinner sig på Stockholms Centralstation. De kan vara genuint slumpartade som de kvantmekaniska förändringarna i atomkärnan eller biologiska mutationer (**objektiv slump**). Det kan också vara så att vår förståelse för vissa orsakssammanhang är bristfällig, t.ex. hur kursutvecklingen för en viss aktie beror av olika faktorer, eller också är sammanhanget så komplicerat att det lättare låter sig hanteras

som slumpmässigt, t.ex. vindriktning och vindstyrka på en viss plats under en viss tidsperiod (**subjektiv slump**).

I själva verket hör nog också våra exempel med tärningskast och människor på centralstationen till den senare kategorin. Om vi hade en tillräckligt förfinad modell av tärningens massfördelning och ytegenskaper, underlagets egenskaper, utgångsläget i handen och kastets impuls och riktning, så vore det kanske möjligt att med stor precision förutsäga resultatet. Till en så detaljerad modell är dock hart när omöjligt att finna tillräckligt noggranna indata, så användbarheten är obefintlig. Det är då mer givande att modellera tärningskastandet som en slumpprocess med sannolikheten 1/6 att utfallet blir t.ex. en fyra. På samma sätt skulle man kunna tänka sig en detaljerad modell (fast omöjlig att realisera), med vars hjälp man skulle kunna förutsäga exakt hur många människor som samtidigt kommer att vistas på stationen. Betydligt mer användbar blir den modell som bygger på ankomst- och avgångsfrekvenser, vilka går att beskriva statistiskt genom elementära observationer.

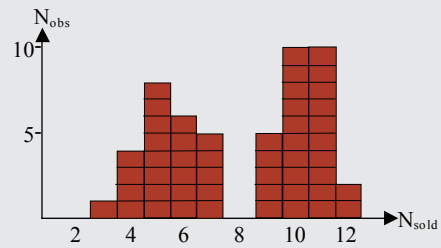
De modeller, som innehåller komponenter med slumpmässigt beteende, kallar vi stokastiska (efter det grekiska ordet för ”gissning”). I dem finns slumpmoment som antingen följer en fast inprogrammerad fördelning, t.ex. sannolikheten ½ för positivt utfall och ½ för negativt, eller en fördelningsfunktion, som styrs via indataparametrar. Utdata från sådana modeller blir alltid stokastiska. Det man får är *ett* möjligt utfall bland många, men hur representativt det är vet man egentligen inget om. För att få klarhet om detta måste man göra upprepade simuleringar med samma indata och observera hur utfallet varierar. Man gör helt enkelt en s.k. Monte Carlo-simulering för att åstadkomma en skattning av utdatafördelningen (se faktaruta 6).

Trots allt behöver man inte alltid göra sina modeller stokastiska. Många gånger är orsakssammanhangen så väl förstådda och så enkla att de låter sig beskrivas och datasättas, och vi gör då en deterministisk modell, där utdata är entydigt bestämt (”determinerat”) av indata. En sådan modell är exempelvis den enkla modell av ett fordon som kör med jämn fart längs en väg, som vi diskuterade som exempel på en analytisk modell. En deterministisk modell behöver dock inte nödvändigtvis vara så enkel. Även mycket komplicerade modeller kan komma i fråga. Det är då vanligen så att

6

Stokastiska utdata

Låt oss anta att vi i en modell beräknar hur många soldater som slås ut i en viss miljö under en given strid. Om modellen är stokastisk kan vi genom en Monte Carlo-simulering få en skattning av fördelningen över utfallsrummet i form av ett frekvensdiagram:



Vi ser då att mellan 3 och 12 soldater kommer att slås ut, och att det är hög sannolikhet att antalet blir 10 eller 11, men det är också ganska sannolikt att det bara är 5 som råkar illa ut.

I detta fall är medelvärdet 8, men det är inget typiskt värde, då detta antal utslagna knappast tycks förekomma alls. Inte heller medianvärdet, 9, är särskilt representativt för utfallet.

En enstaka simulering kan ge oss vilket utfall som helst (utom 8) i intervallet 3 – 12.

det slumpartade beteende, som ändå kan finnas, kan ”kapslas in” i grundobjekten, vilkas egenskaper återspeglar något slags medelvärde. Om spridningen i detta medelvärde är liten, så brukar denna metod fungera tillfredsställande, annars finns risk för påtagligt felaktiga resultat p.g.a. det statistiker brukar kalla ”för tidig medelvärdesbildning” (se faktaruta 7).

Slumptal

Om man nu har en stokastisk modell, så är man i behov av att kunna dra slumptal. Har man en modell, där man handräknar, så är det enkelt, för då kan man singla slant eller kasta tärning i varje läge där det fordras ett slumpmässigt värde. Men hur gör man i en programmerad modell på en dator? En sådan maskin är ju med nödvändighet utpräglat deterministisk.

Tricket är här att konstruera en algoritm som genererar en mycket lång talserie, som inte återupprepar sig i brådrasket. Den ska dessutom vara sådan att om man plockar ut ett godtyckligt avsnitt av serien, så skall dess talföljd vara till förvillelse lik en äkta slumptalsserie. Om vi preciserar detta ganska diffusa villkor, så kan man t.ex. säga att man med ett statistiskt test skulle kunna bestämma att talföljden är slumpmässig med 99 % sannolikhet. Vad som avses med återupprepningskravet är kanske inte lika lätt att kvantifiera, men det hänger samman med hur många slumptal som behövs i en typisk simulering. Börjar serien återupprepa sig uppstår oönskade korrelationer.

Det finns många algoritmer som används i slumptalsgeneratorer, men de vanligaste bygger på en mycket enkel idé, vars princip kan beskrivas med formeln

$$X_i = (a \cdot X_{i-1} + c) \bmod m$$

där alla storheter är positiva heltal eller noll (”linjär kongruensgenerator”). Man producerar alltså en talserie där varje nytt tal beräknas genom en multiplikation och addition av det föregående talet och där man sedan behåller resten efter division med basen *m*. Med förnuftiga val av parametervärdena *a, c* och *m*, liksom av frövärdet (startvärdet) X_0, kan man åstadkomma talföljder som är upp till *m* enheter långa innan de upprepas. För att få ”slumptal” likformigt fördelade i intervallet 0 – 1 delar vi så våra X_i med *m*.

För tidig medelvärdesbildning:

7

Låt oss anta att vi med det förra exemplet som grund vill göra en modell av sjukvårdsplutonens arbete och då gör det väl genomtänkta antagandet att belastningen på plutonen är proportionell mot kvadraten på antalet utslagna soldater som skall tas om hand. För enkelhetens skull sätter vi proportionalitetskonstanten till 1, d.v.s.

$$b_n = N^2_{sold,\,n}$$

och väntevärdet för belastningen

$$E(b_n) = \sum_n N^2_{sold,\,n} \cdot p_n$$

där p_n är N_{obs} för resp. antal utslagna soldater delat med totala antalet observationer (i vårt fall 50). Tillämpar vi nu denna formel får vi genom att avläsa diagrammet i föregående ruta

N^2_{sold}	N_{obs}	$50 \cdot E(b_n)$
9	1	9
16	4	64
25	8	200
36	6	216
49	5	245
81	4	324
100	10	1000
121	10	1210
144	2	288
Summa		3556

Väntevärdet får vi nu om vi dividerar totalsumman med 50, d.v.s. 71,12.

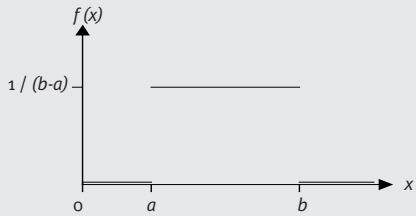
Hade vi nu i stället ersatt vår fördelning av utslagna soldater med dess medelvärde, 8, innan vi beräknat belastningsvärdet hade vi fått 8^2, d.v.s. 64, vilket alltså påtagligt underskattar belastningen på vår sjukvårdspluton.

Exempel på användbara fördelningar

Likformig fördelning

Karaktäristik: Kontinuerlig fördelning användbar för jämn men slumpmässig utspridning över ett intervall a till b.

Frekvensfunktion: $f(x) = \frac{1}{b-a} \qquad x \in [a,b]$

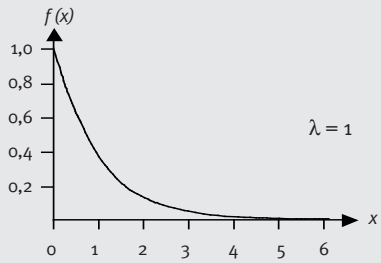


Medelvärde: $(a + b)/2$, varians: $(b - a)^2/12$

Exponentialfördelning

Karaktäristik: Kontinuerlig fördelning användbar för t.ex. tiden mellan olika händelser, som inträffar med en konstant intensitet λ.

Frekvensfunktion: $f(x) = \lambda \cdot e^{-\lambda \cdot x}$

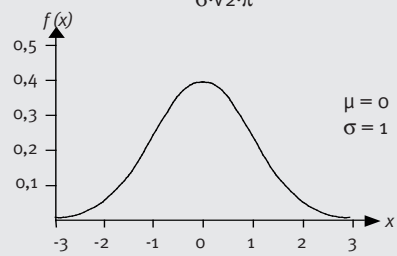


Medelvärde: $1 / \lambda$, varians: $1 / \lambda^2$

Normalfördelning

Karaktäristik: Kontinuerlig fördelning användbar för t.ex. olika typer av felvariationer eller storheter som utgör summan av en stor mängd (slumpmässiga) enheter.

Frekvensfunktion: $f(x) = \frac{1}{\sigma \cdot \sqrt{2 \cdot \pi}} \cdot e^{\frac{(x-\mu)^2}{2 \cdot \sigma^2}}$

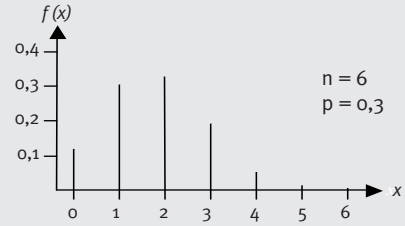


Medelvärde: μ, varians: σ^2

Binomialfördelning

Karaktäristik: Diskret fördelning användbar för t.ex. antalet felaktiga objekt i en samling av n stycken, då man vet att felsannolikheten är p.

Frekvensfunktion: $f(x) = \binom{n}{x} \cdot p^x \cdot (1-p)^{n-x} \; x \in \{0,1,\ldots,n\}$

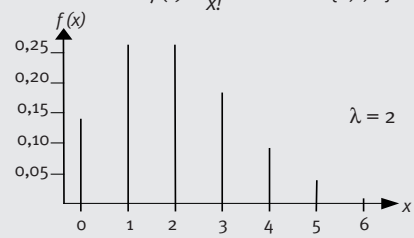


Medelvärde: $n \cdot p$, varians: $n \cdot p \cdot (1-p)$

Poissonfördelning

Karaktäristik: Diskret fördelning användbar för t.ex. antalet händelser under en tidsperiod vid genomsnittligt konstant intensitet λ.

Frekvensfunktion: $f(x) = \frac{\lambda^x}{x!} \cdot e^{-\lambda} \quad x \in \{0,1,\ldots\}$

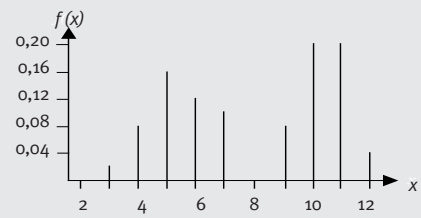


Medelvärde: λ, varians: λ

Egendefinierad fördelning

Karaktäristik: En fördelning av helt godtyckligt utseende, som vanligtvis representerar en mätserie och beskrivs i form av en tabell. Exempelvis skulle frekvensfunktionen för exemplet med de utslagna soldaterna i ruta 6 ha följande utseende.

x	f(x)	x	f(x)	x	f(x)
2	0	6	0,12	10	0,20
3	0,02	7	0,10	11	0,20
4	0,08	8	0	12	0,04
5	0,16	9	0,08	13	0



Medelvärde: 8, varians: 7,27

Normalt sett är de slumptalsgeneratorer, som tillhandahålls ganska bra. De producerar mycket långa talserier utan upprepning (m är typiskt av storleksordningen 2^{32}–1, i forskningsgeneratorer förekommer periodlängder ända upp till $2^{20\,000}$) och är i allmänhet ganska okänsliga för olika val av frövärden. De är dock ofta konstruerade med en speciell talrepresentation i åtanke, och eftersom den kan vara ganska olika mellan datorer av olika fabrikat, så bör man vara mycket försiktig om man måste föra över en slumptalsgenerator från en dator till en annan. Vill det sig illa kan man få ett helt annat beteende med oacceptabelt korta talserier med oönskade korrelationer som följd. En antydan om vad det kan handla om: Antag att vi har en generator med m av storleksordningen 100 000, att antalet objekt i simuleringen är av storleksordningen 1000, och att alla gör slumpvisa beslut. Konsekvensen blir då att efter 100 slumpade beslut är sekvensen ”förbrukad” och börjar om från början och det uppstår oönskade korrelationer.

I allmänhet tillåter slumptalsgeneratorer att användaren själv sätter frövärde. Att man därigenom kan återupprepa en redan använd slumptalsserie har påtagliga fördelar, dels då man behöver undersöka oväntade utfall lite närmare, dels när man vill jämföra två eller flera olika alternativ. Det förra fallet möjliggör avlusning av programmen. I det senare fallet kan man genom att använda samma slumptalsserie för alla alternativ få en betydligt bättre precision i jämförelsen.

Vi vet alltså nu hur vi kan åstadkomma likformigt fördelade slumptal, men därmed är inte allting gott och väl, för även om man anser att något är slumpmässigt behöver det inte vara likafördelat. Det kan i själva verket vara fördelat precis hur som helst (se exempel i faktaruta 8). Frågan blir då hur man åstadkommer slumptal i enlighet med andra fördelningar.

Här är principen den, att man genererar slumptal likformigt fördelade inom intervallet 0–1 som förut, och att man sedan kollar i den aktuella fördelningens fördelningsfunktion $F(x)$ vilka värden på x som svarar mot resp. slumptal. En sådan fördelningsfunktion kan vara given antingen analytiskt, som i de vanligast förekommande fördelningarna, eller som en tabell grundad på försöksserier.

Kontinuerlig eller diskret modell?

En fjärde klassificeringsegenskap hos modeller har att göra med hur förändringarna i tillståndsvariablerna sker. Om tillståndet ändras kontinuerligt så har vi en kontinuerlig modell. Man kan här som exempel tänka på modeller av olika slags flöden eller föremål i rörelse. Vårt enkla exempel på en analytisk modell, fordonet som färdas ostört längs en väg, är också ett exempel på en kontinuerlig modell. Ett reglersystem för styrning av t.ex. ett ventilationssystem eller en styrautomat till ett obemannat flygplan kan i allmänhet också med fördel modelleras kontinuerligt. I fysiken finns det många exempel på denna typ av modeller, och de uttrycks då ofta i form av en eller flera differentialekvationer, t.ex. ekvationen för en projektilbana:

$$\begin{cases} \ddot{x} = -c \cdot \dot{x}^2 \\ \ddot{z} = -g - c \cdot \dot{z}^2 \end{cases}$$

där c är luftmotståndet och g gravitationen.

I en diskret modell å andra sidan sker tillståndsförändringarna då och då. Exempelvis förändras kölängden på banken varje gång en ny kund kommer dit och då en kund börjar expedieras i en kassa. Men förändringarna är inte bara utspridda i tiden, de är också momentana, d.v.s. själva förändringen tar ingen tid. Denna

typ av modell, **diskreta händelse-modeller**, är mycket vanlig, och låter sig sällan uttryckas i sluten matematisk form som de kontinuerliga modellerna. Om det till äventyrs går att göra blir det vanligen i form av s.k. differens- (eller rekurrens-) ekvationer, och de är dessutom i allmänhet mycket komplicerade att lösa.

Många modeller är dock inte renodlade diskreta händelse-modeller, utan de innehåller större eller mindre delar av kontinuerlig natur. Man kan t.ex. tänka på en modell av stridsvagnar, som framrycker under eldgivning. Förflyttningen är kontinuerlig medan skotten avlossas som diskreta händelser, och dessa kan i sin tur påverka den fortsatta rörelsen. Mer allmänt kan man indela samspelet mellan kontinuerliga och diskreta komponenter i tre grupper. Den första, som vi just exemplifierat, innebär att man får en diskret förändring av en kontinuerlig variabel. Den andra innebär att en diskret händelse kan ändra själva den styrande funktionen för en kontinuerlig variabel. Ett exempel på detta kan vara då det dyker upp en polisbil i trafikflödet, vilket brukar ändra övriga förares beteende högst påtagligt. Den tredje gruppen är de händelser som utlöses därför att en eller flera kontinuerliga variabler uppnått vissa tröskelvärden (d.v.s. ”händelsevillkoren” har uppfyllts). Detta kan t.ex. vara fallet då en stridsvagn under framryckning når fram till en punkt då den blir upptäckt av fienden.

Ett problem med kontinuerliga modeller är i vad mån man kan realisera dem i en dator. Före mitten av 70-talet var det ganska vanligt med s.k. analogimaskiner, där man med hjälp av uppkopplade elektriska kretsar kunde genomföra kontinuerliga representationer av kontinuerliga förlopp. Men numera finns i regel bara digitala datorer, och hur kan man modellera något kontinuerligt på en sådan utpräglat diskret tingest? Svaret är att modellers realisering till datorprogram alltid innebär approximationer, och då är den som ersätter kontinuiteten med diskreta steg ingalunda den besvärligaste. Genom en finfördelning av tiden kan man komma godtyckligt nära. Man kan t.ex. jämföra med hur man i matematiken approximerar en integral med en summa. Egentligen är det ovan behandlade problemet med att modellera slumpmässigt beteende på en deterministisk maskin bra mycket besvärligare.

Tids- eller händelsestyrd simulering?

Vi är nu egentligen inne mera på användningen av modellen, simuleringen, än på själva modellen som sådan. En simulering kan beskrivas som händelsestyrd eller tidsstyrd beroende på mekanismen för hur tiden stegas fram. I det förra fallet förs simuleringen framåt genom att man hoppar från händelse till händelse i den ordning de skall utföras och sätter om simuleringsklockan ryckvis. I det andra fallet går tiden sin jämna lunk och händelserna inträffar allteftersom tiden är inne för dem.

I en **händelsestyrd simulering** håller man reda på alla kända framtida händelser genom att lägga dem i en tidssorterad händelselista. Ett beräkningssteg består i att ta ut första elementet i denna lista, sätta simuleringsklockan till tidsstämpeln för motsvarande händelse och utföra händelsen. Denna åtgärd kan ge upphov till en eller flera framtida händelser, vilka med vederbörliga tidsstämplar placeras in på sina rätta platser i händelselistan.

Som exempel kan vi tänka på en stridsmodell som innehåller artilleridueller. Då händelsen målidentifiering inträffar, genereras följdhändelsen eldgivning en viss tid senare, som beror på tillståndet hos artilleriförbandet. Händelsen eldgivning genererar i sin tur ett antal granat-

explosioner vid målet utspridda på lämpligt sätt över tiden. Emellertid kan det tänkas att den också genererar händelsen fientlig moteld, vilket kanske medför att vissa av de schemalagda händelserna aldrig kommer att kunna äga rum. Vi måste alltså även ha en mekanism för att plocka bort framtida händelser ur händelselistan utan att de utförs.

I en **tidsstyrd simulering** låter man alltså simuleringsklockan stega framåt med ett fixt tidsinkrement (tidssteg) varvid alla tillståndsvariabler räknas ut för den nya tidpunkten. Beroende på de nya värdena avgörs om olika händelser har inträffat under tidssteget, t.ex. att ett objekt har blivit synligt från ett annat objekt, och i så fall låter man utvecklingen ta en ny riktning. Ett problem, som är mera accentuerat med detta tillvägagångssätt än vid händelsestyrning, är att många olika händelser, som i verkligheten skulle inträffa efter varandra, skall utlösas vid samma tidpunkt. Man behöver då förse modellen med regler för i vilken ordning dessa händelser skall hanteras, för att de skall påverka varandra på ett realistiskt sätt. Att ha tillräcklig fantasi för att förutse alla nödvändig regler är dock inte lätt. Ett sätt att minska problemet är att välja kortare tidsinkrement, men då åstadkommer man i stället längre beräkningstider. För övrigt behöver inte tidsinkrementet vara konstant under hela simuleringen. Det är vanligt att man använder ett kortare steg under speciellt intensiva delar av förloppet.

Tidsstyrd simulering kan ses som ett specialfall av händelsestyrd simulering. Man kan tänka sig att man skapar en artificiell händelse vid varje tidssteg och lägger in den i händelselistan. Därigenom kan man också se en möjlig kombination av de båda typerna, d.v.s. en händelsestyrd simulering kompletterad med avstämningstidpunkter vid jämna tidsintervall (s.k. ”hjärtslag”).

Sammanfattningsvis kan man alltså påstå att händelsestyrd simulering alltid kan användas, men att det speciellt lämpar sig för de fall då händelser sker ojämnt utspridda i tiden. Tidsstyrd simulering är att föredra då man har att göra med objekt som skall ha ett synkront beteende, t.ex. digitala kretsar, artificiella neuronnät, reglersystem. Likaså kan det fungera väl med tidsstyrning när händelserna ligger någorlunda jämnt utspridda eller då tidsintervallen mellan dem är små jämfört med tidsinkrementen. I princip är det alltså modellernas detaljerings- och abstraktionsnivå som påverkar valet.

En viktig modelltyp där tidsstyrd simulering är närmast nödvändig är realtidssimulatorer. Att t.ex. en flygsimulator skulle vara händelsestyrd i strikt mening, vore minst sagt besvärande. Det skulle innebära att scenen för operatören inte skulle förändras förrän en händelse (målupptäckt, beskjutning e.dyl.) inträffar, och då skulle förändringen bli synnerligen abrupt. Här måste det kontinuerliga förloppet åskådliggöras genom att man arbetar med så små tidssteg, att operatören upplever det som kontinuerligt.

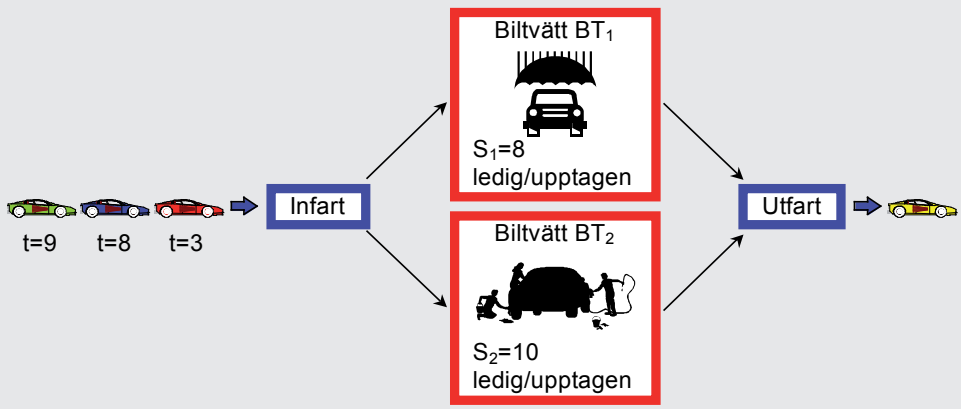
Ett annat problem med realtidssimulatorer är att beräkningarna måste synkroniseras med den verkliga tiden. Om vissa tidssteg innebär stora mängder beräkningar, kan man uppleva en onaturlig inbromsning i förloppet. Det är lika oacceptabelt som om man skulle låta simuleringen rusa iväg, då tidsstegen innebär inget eller endast lite arbete. Att hantera dessa faktorer fordrar en sofistikerad balans mellan datorernas kapacitet, tidsinkrementets storlek och utnyttjandet av pausfunktioner. Därvid har man dock en hel del hjälp av de speciella operativsystem som finns för realtidssimuleringar.

Exempel på händelsestyrning

Antag att vi modellerar en biltvättsanläggning på följande sätt. Anläggningen består av två tvättplatser med kapaciteten 8 resp. 10 min/bil. Bilar ankommer slumpmässigt. De händelsetyper som hanteras i modellen är

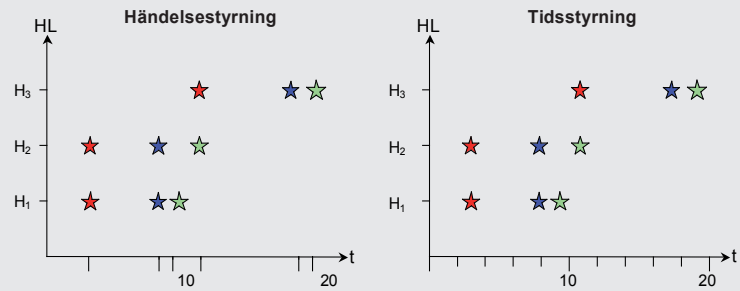
Händelsetyp / Beskrivning	Följdhändelser
H_1 : Ankomst infart	Generera nästa ankomst → ny H_1 i händelselistan Ledig biltvätt? → Skicka bil till tvättplats → ny H_2 i händelselistan
H_2 : Ankomst tvättplats BT_i	Beräkna sluttid → ny H_3 i händelselistan Ändra tillståndet till upptagen
H_3 : BT_i klar	Bil till utfart (ev. ny händelse) Ändra tillståndet till ledig

Låt oss nu göra en simulering med modellen, som vi kompletterat med informationen att BT_1 väljs före BT_2 om möjligt. Vi antar att bilarna kommer till anläggningen vid tidpunkterna 3, 8 resp. 9 minuter.



Händelselistan kommer i vårt exempel att successivt få följande innehåll.

t	H
3	Röd bil vid infart
3	Röd bil vid BT_1
8	Blå bil vid infart
8	Blå bil vid BT_2
9	Grön bil vid infart
11	Röd bil vid utfart
11	Grön bil vid BT_1
18	Blå bil vid utfart
19	Grön bil vid utfart



I diagrammen åskådliggörs hur tiden flyttas fram vid händelsestyrning resp. tidsstyrning med inkrementet 2 minuter. Varje ”tick” (exekveringstillfälle) är markerat längs tidsaxeln. Observera att i det senare fallet kommer t.ex. den röda bilen att börja tvättas först efter fyra minuter och hanteras vid utfarten först efter tolv minuter.

Programvaruteknik – en översikt

En simuleringsmodell är som vi sett ofta en modell som exekveras på datorer. Den är alltså uttryckt i först logisk, sedan matematisk och slutligen algoritmisk form.

Följaktligen är det två kunskapsområden som framför allt behövs för att stödja modellering och simulering som egen vetenskap. Det ena är **matematiken** med de flesta av dess underavdelningar som t.ex. geometri, matematisk statistik och numerisk analys. Detta kommer inte att behandlas explicit i denna skrift, det förutsätts läsaren ha stiftat bekantskap med på andra sätt.

Det andra är datalogin, och då speciellt **programvaruteknologin**, som också är en i detta sammanhang mycket viktig stödvetenskap. Den skall vi däremot nu ägna en viss uppmärksamhet.

Programvaruteknik eller *software engineering*, som det kallas på engelska, är ett begrepp, som myntades av en arbetsgrupp inom NATO 1967. Tanken med detta begrepp är att man vid all utveckling och handhavande av programvara skall använda sig av ett systematiskt, disciplinerat och mätbart tillvägagångssätt, kort sagt att arbeta ingenjörsmässigt. Dittills hade programmering i hög grad betraktats närmast som konstnärlig verksamhet med stor personlig frihet och variation bland utövarna. I fortsättningen skulle man genom att använda strikta metoder uppnå större förutsägbarhet i projekten och program, som lättare skulle kunna kontrolleras och återanvändas.

Bakom detta initiativ fanns en insikt om en begynnande kris inom programmeringen, något som säkert bromsats genom uppstramningen, men som vi ändå i stor utsträckning lever med än idag. Till stor del hänger det samman med den generella svårigheten att planera och schemalägga skapande mänsklig aktivitet. Resultatet blir att programmen alltför ofta levereras alltför sent, eller i värsta fall inte alls. Alternativt, eller kanske både och, så tenderar programmen att bli mycket dyrare än planerat att utveckla och underhålla.

Men om vi nu antar att vi lyckas producera vårt program inom utsatt tid och inom givna kostnadsramar, så är det ändå inte säkert att projektet är lyckat. Programmet kanske är instabilt, därför att det finns många kvarvarande fel i det, så användaren inte kan utföra det han eller hon vill. Det kan också vara så att den produkt vi lämnar ifrån oss kraschar vid oförsiktig användning, t.ex. om man går utan-

för definitionsområdet, och kanske förorsakar användaren förlust av mödosamt ihopsamlade indata. Värst av allt är nog ändå att många program inte visar sig motsvara användarens behov och förväntningar över huvud taget, därför att varken denne eller programutvecklaren har haft förmågan att i förväg till fullo förstå hur man egentligen skulle vilja kunna använda programmet.

I strikt mening kan man säga att om alla ovannämnda avvikelser tas i beaktande, så är det mycket få projekt som blir lyckade. Amerikanska studier pekar på att det är i storleksordningen 10 %. Självfallet upplevs det dock i verkligheten inte lika negativt. T.ex. accepterar man för det mesta ett par veckors tidsöverdrag eller att programmet inte är fullständigt uttestat utan att klassificera det som ett misslyckande. Icke förty är det blott alltför vanligt med havererade programmeringsprojekt. Sammanfattningsvis kan man nog säga att de främsta orsakerna är följande.

- Kravspecifikationerna är felaktiga eller åtminstone ofullständiga eller tvetydiga.
- Kommunikationen mellan beställaren och utvecklaren fungerar dåligt, så utvecklaren arbetar mera efter vad han tror att kunden vill ha än efter vad denne verkligen behöver.
- Kommunikationen mellan olika arbetsgrupper inom projektet kan också vara bristfällig med kompatibilitetsproblem som följd.
- Kompetent personal kan visa sig vara en bristvara. Den kanske inte finns alls eller andra projekt konkurrerar om den.
- Komplexiteten i programvaran kan bli så stor att ingen riktigt förmår överblicka den.
- Ny och oprövad teknik kan vara lockande men leda till oväntade svårigheter. Även väl beprövad teknik kan vara förledande, för när man väl ser hur ett problem skall lösas, är det lätt att underskatta svårigheterna i att genomföra lösningen.

Med en lämpligt vald programvaruteknik kan man i ganska stor utsträckning begränsa dessa skadliga inflytelser. Dock måste man komma ihåg att ingen teknik i sig löser någonting om inte utvecklarna själva är inställda på att underkasta sig en ganska stor portion av disciplin. Tekniken ger dem då hjälp och stöd att komma ihåg och förvalta sitt ansvar.

Ända sedan 60-talet har man dessutom burit med sig drömmen om att alla programvaror skall vara framtagna med en gemensam teknik och fullständig disciplin, så att man successivt kan sätta samman nya program av redan existerande programdelar, s.k. komponentbaserad programutveckling. Inom vissa delområden har man i det fallet kommit en bra bit på väg, t.ex. för grafiska användargränssnitt, ordbehandlare, webbläsare, systemprogramvara, grundläggande datastrukturer och algoritmer. Generellt sett har man dock fortfarande långt till målet. Hindren är svåra och kan indelas i fyra klasser:

- **Tekniska hinder** – det fordras en standardisering, som kanske är omöjlig att få heltäckande. Många kommer därför att känna sig alltför instängda och få alltför lite rum för den kreativa lusten och förmågan.
- **Ekonomiska hinder** – att utveckla programvaror med tanke på återanvändning kräver ett mera långsiktigt perspektiv än vad som är gängse i de flesta projekt. Därför kommer det att behövas ekonomiskt tillskott till projektmedlen för att stimulera till återanvändbara lösningar.
- **Psykologiska hinder** – förutom svårigheten att böja sig under tvånget att följa

en standard är det extra svårt att göra det om någon annan har formulerat den, ”*not invented here*”-syndromet.

- **Infrastrukturella hinder** – det behövs ett generellt system med vars hjälp man kan söka, hämta och lagra programmoduler, inte bara lokalt inom varje organisation utan globalt.

Livscykelmodeller för programvaror

I de modeller av programutvecklingsprocessen, som tillämpats sedan databehandlingens barndom är det sju faser som är genomgående, fast ibland med lite olika etiketter. I stort sett har de följande innebörd.

1. Kravfångst
   Här handlar det om att sätta sig in i användarens verkliga behov, att förstå de situationer i vilka programmet skall användas och hur det då skall kunna vara bästa möjliga stöd. Det kan t.ex. ske genom intervjuer och enkäter, genom att iaktta och analysera den verksamhet som ska stödjas, och genom att utveckla enkla prototyper som testas mot användarna.
2. Analys
   I denna fas gäller det att bestämma *vad* som skall göras med programmet och att slå fast vad som inte är relevant eller viktigt. Analysen leder fram till att man vet vad det är för slags modell som man skall programmera, vilka objekt som skall ingå, vilka relationer som skall åskådliggöras, vilka funktioner som måste tillhandahållas, vilka operationer som användaren skall kunna utföra o.s.v.
3. Planering
   Man skall nu utifrån krav och analys kunna göra en någorlunda god tids- och kostnadsuppskattning av programmeringsinsatserna. Vidare skall man schemalägga utvecklingsarbetet, vem som gör vad och när. Hur ska man t.ex. hantera konkurrensen från andra projekt om viss nyckelkompetens?
4. Design
   Medan man i analysfasen sysslade med vad som skulle göras, så är man i designfasen mera inriktad på *hur* det ska göras, d.v.s. man gör en ”planritning” över programstrukturen. Häri ingår att beskriva hur programmet skall modulariseras och hur modulernas gränssnitt skall se ut, hur samverkan med omgivande system (inklusive mänskliga operatörer) skall gå till, vilka datastrukturer man bör använda, val av algoritmer för att uppfylla effektivitets- och noggrannhetskrav o.s.v.
5. Programmering
   Nu skrivs programmet i ett lämpligt programspråk i enlighet med designen, och då måste varenda detalj vara väl förstådd och uttryckt. Datorn är därvidlag en obarmhärtig domare, eftersom den gör exakt som den blir instruerad att göra och inte ser till intentionerna, så som människor kan göra när de får lite oprecisa instruktioner.
6. Drift och underhåll
   Slutligen skall vårt program sättas i drift genom att integreras i den miljö och på det/de datorsystem, där det skall användas. Fortsättningsvis skall det antagligen också korrigeras och utvecklas vidare i enlighet med de iakttagelser som görs under användandet.
7. Validering, verifiering och testning
   Denna sjunde fas är egentligen parallell med övriga sex faser, för den innebär en fortlöpande granskning av att man gör rätt saker och att det man gör är rätt.

De olika livscykelmodeller som finns kopplar i princip ihop dessa sju faser på lite olika sätt. I den traditionella **vattenfallsmodellen** (se faktaruta 22) passeras varje fas endast en gång, även den sjunde, som genomförs först sedan de övriga är klara. I denna ursprungliga form tillämpas dock inte modellen längre (om det ens någonsin har kunnat ske), utan snarare i en modifierad form med prototyputvecklingar och iterationer mellan de olika faserna.

En variant, som därigenom fått status av egen utvecklingsmodell, är **den inkrementella modellen**. Här genomlöps faserna 1 – 3 en enda gång och i en följd. Övriga faser genomförs i flera omgångar och för olika delar av programmet, en del i taget. I den s.k. **V-modellen** går man också systematiskt igenom steg efter steg, men varje steg avslutas med en ordentlig utvärdering eller verifiering. Är man då inte nöjd, görs steget om.

Slutligen kan man nämna den s.k. **spiralmodellen**. Den kännetecknas också av ett iterativt förfarande där de olika faserna genomlöps flera gånger. Inför varje nytt steg genomförs en riskanalys utifrån den kunskap man har skaffat sig i det läge i utvecklingsprocessen där man just befinner sig. Det kan gälla problem med tillgängligheten av väsentliga medarbetare, hårdvarukomponenter eller andra resurser, att nya produkter dykt upp på marknaden, vilket gör en fortsatt nyutveckling mindre befogad, o.s.v. Skulle risken bedömas vara för stor, så avbryts eller ominriktas projektet.

Programstrukturering

Om man nu vill åstadkomma någorlunda omfattande program med en konsekvent och lättläst struktur, kod och dokumentation, med återanvändbara delar, och som dessutom är lätt att anpassa till nya krav och nya datormiljöer, så finner man snabbt att det svåraste hindret utgörs av problemens inneboende komplexitet. Lösningen måste då ligga i att införa olika abstraktionsnivåer, som gör att strukturerna framträder tydligare och inte försvinner i en skog av konkreta detaljer.

För att behärska den lokala komplexiteten i programmen väljer man ett lämpligt programspråk. Dessa har ju utvecklats från de mycket osofistikerade maskin- och assemblerspråken från datorepokens första år genom införande av fler och fler abstraktionsnivåer. Först infördes vad man kan kalla **uttrycksabstraktion** med språk som Fortran. Därefter fick vi **styrningsabstraktioner** i exempelvis Algol. Som tredje steg kom **dataabstraktioner** med språk som

Exempel på abstraktioner i programspråk 10

Uttrycksabstraktion i Fortran

```
V(I) = A/2 - B(I)
```

Styrningsabstraktion i Algol

```
IF x = y THEN s := x/2 ELSE x := x + 1
```

Dataabstraktion i Pascal

```
type  month =
      (jan,feb,mar,apr,may,jun,jul,aug,sep,oct,nov,dec);
type  date = record
      mo: month;
      day: 1..31;
      year: integer;
      end;
```

Objektabstraktion i Java

```
public class Rectangle {
      private int x, y, length, width;
      Rectangle (int x, int y, int Length, int Width) {
            setPos (x, y);
            length = l;
            width = w; }
      public void setPos (int newX, int newY) {
            x = newX;
            y = newY; }
      public int Area {
            return length * width; }
      ... }
```

Pascal, t.ex. uppräkningsbara typer och dataposter. Det fjärde steget representeras av språk som C++ och Java, som inför ytterligare en nivå, **objektabstraktioner.**

Programmens globala komplexitet bemästras dock inte enbart genom att välja ett programspråk på hög nivå. Här heter lösningen i stället programstrukturering. Programstrukturen är inte enbart en egenskap hos den färdiga koden, den är framför allt ett självständigt dokument som utarbetas innan programmet kodas och som omarbetas varje gång en programändring är aktuell. Medan programstrukturen (liksom specifikationen) beskriver vad programmet skall göra, så talar programkoden om hur programmet gör det. Liksom det är viktigt att välja ett bra programspråk för kodningen är det viktigt att välja en bra metod för att producera programstrukturen. Under årens lopp har det kommit fram åtskilliga metoder, vanligtvis inom den administrativa databehandlingens område men med mer eller mindre stor tillämpbarhet inom det tekniskt-vetenskapliga området. Generellt kan sägas att ingen metod är bra för all slags programutveckling, men att vissa metoder visat sig vara tämligen väl lämpade för de programtyper som är vanliga inom försvarsmakten. Väsentligt för dem alla är långt driven modularisering och uppbyggnad av hierarkiska strukturer.

Objektorientering

Det numera vanligaste programutvecklingsparadigmet är utan tvekan det som kallas objektorienterad programutveckling och – design. Det kännetecknas av att världen betraktas och kan beskrivas som ett system av objekt och relationer mellan dem. Följaktligen har vi här en nära koppling till vår beskrivning av en modell som ett system av växelverkande objekt. Det är därför naturligt att objektorientering blivit ett så allmänt accepterat tillvägagångssätt bland dem som programmerar simuleringsmodeller.

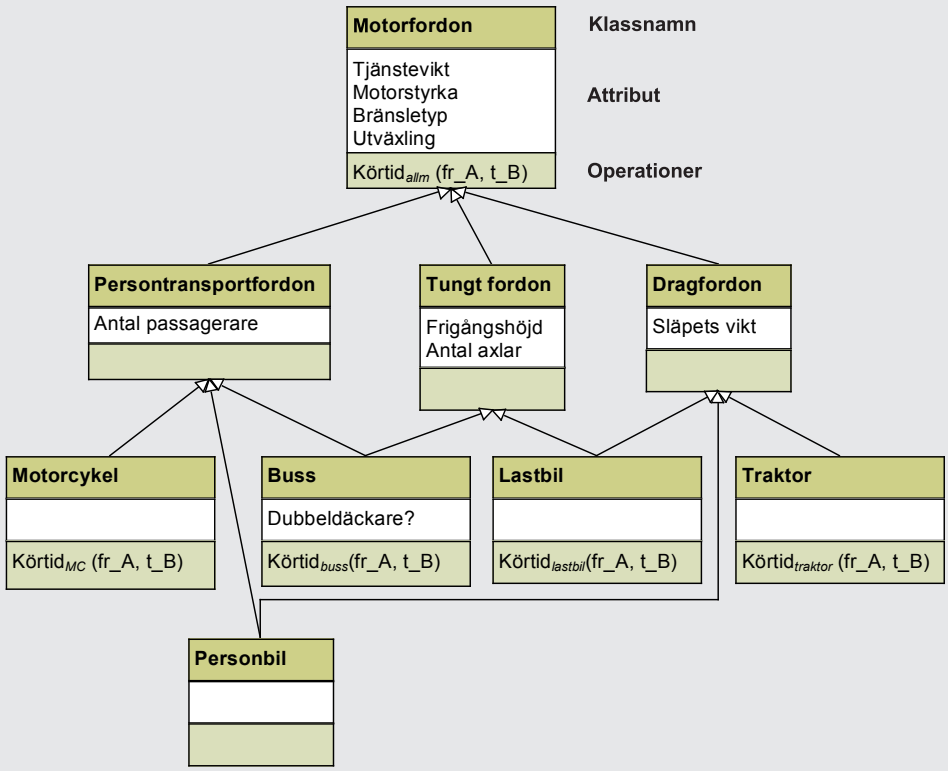
Här skall inte objektorienteringen beskrivas annat än översiktligt för att ge en allmän föreställning om dess idé och terminologi. Den som verkligen vill lära sig att utveckla program enligt paradigmet bör söka sig till någon av de utmärkta läroböcker som finns i ämnet, t.ex. *”Object-oriented Software Construction”* av Bertrand Meyer.

Vad är då ett objekt i de objektorienterade programmerarnas värld? Vi drar oss till minnes, att modeller kan uppfattas som system av samverkande objekt, konkreta eller abstrakta. Datorprogram är ju också modeller, så vi kan naturligtvis tillämpa vårt betraktelsesätt även på dem. Objekten har alltså ett antal egenskaper, som vi kallar **attribut**. Objektens samverkan modelleras genom att låta dem tillhandahålla **operationer**, vilka ibland kallas tjänster eller metoder. Ett objekt begär av ett annat objekt att det skall utföra en sådan operation genom att skicka ett **meddelande**. Det mottagande objektet reagerar på meddelandet med att köra operationen i fråga under de förutsättningar (indata) som specificeras i meddelandet. Slutligen känner vi igen att ett objekts **tillstånd** vid en viss tidpunkt bestäms av dess attributvärden vid denna tid.

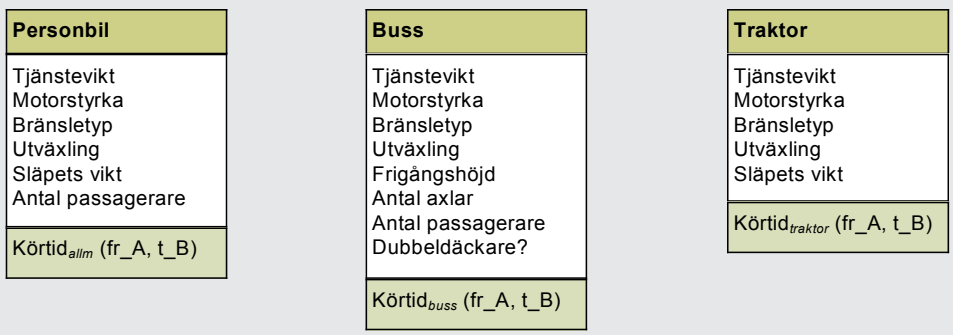
Om vi för ett ögonblick ser på objekten utifrån programmerarens synvinkel snarare än modellerarens, så kan vi beskriva dem som abstrakta (generaliserade) datatyper. En **abstrakt datatyp** är en familj av datastrukturer, som beskrivs av sina yttre egenskaper, d.v.s. vilka funktioner som kan utföras och vilka egenskaper dessa funktioner har. Specifikationen av en abstrakt datatyp är en formell, matematisk beskrivning och (i det ideala fallet) saknar sido-

11

Exempel på specialisering genom arv



Här ser vi exempel på såväl multipelt arv (t. ex. ärver personbil både persontransportfordon och dragfordon), polymorfism (t.ex. kan både lastbil och buss betraktas som tungt fordon) som dynamisk bindning (operationen Körtid har olika algoritmer i t.ex. personbil och lastbil). Arvsmekanismerna innebär att t.ex. klasserna personbil, buss och traktor de facto får följande utseende:



effekter. Den inre representationen lämnas därhän, man beskriver endast det som utåt är väsentligt. En abstrakt datatyp kan vara generisk, d.v.s. den kan ha parametrar, som definieras vid instantieringarna, vilka härigenom kan vara av vitt skilda slag. Exempelvis kan man ha en abstrakt datatyp för köer, som genom lämpliga instantieringar kan användas både till köer av bilar vid en biltvätt och till listor av händelser.

Objekten påverkar inte bara varandra, de står också i bestämda relationer till varandra. Sålunda beskrivs alla objekt av samma typ, d.v.s. som har samma attribut och operationer, som en **klass**. Denna kan alltså ses som en mall eller ett mönster för hur objekten ser ut. Varje enskilt objekt sägs vara en **instans** av sin klass.

Vissa klasser kan i sin tur stå i ett hierarkiskt förhållande till varandra på så sätt att en klass kan vara en specialisering av en annan. Detta innebär att en subklass kan definieras med hjälp av en superklass genom att den ”**ärver**” den senares alla attribut och operationer förutom att den även tillför ett antal egna. Det är också möjligt att subklassen kan modifiera definitionen av en eller flera av superklassens attribut och operationer. Vid fullt utvecklad objektorientering är det t.o.m. möjligt med arv från fler än en superklass. Se exemplet i faktaruta 11.

I klasserna visas utåt endast de egenskaper som behöver vara synliga, s.k. **inkapsling**. Interna attribut och hjälpoperationer hålls dolda för objekt av andra klasser, varigenom man kan göra gränssnitten mellan objekten tydliga och minska risken för oönskade sidoeffekter. Jämför med diskussionen om grundobjekt i modeller!

En annan typ av objektrelationer är **association**. Den representerar en koppling av något slag mellan olika objektklasser. Det kan vara kopplingar av olika styrka, t.ex. ”är allierad med”, ”äger” eller ”består av”. I det sistnämnda fallet brukar man kalla associationen för **aggregat**. Är det dessutom så att aggregatet kännetecknas av ett starkt drag av ägarskap, d.v.s. att ett subobjekt alltid ingår i ett och endast ett superobjekt och att det inte kan flyttas till ett annat superobjekt, så används beteckningen **komposition**. Alla slags associationer brukar representeras som attribut där värdet är en referens till det associerade objektet.

Två andra begrepp som förekommer i samband med objektorienterad programmering är **polymorfism** och **dynamisk bindning**. Det förra står för möjligheten att behandla objekt av olika klasser som om de vore lika. Detta sker genom att man tar fasta på det som är gemensamt och samlar det i en ny superklass, till vilken övriga klasser får bli arvingar. Dynamisk bindning har att göra med att en operation kan förekomma i olika utförande i olika subklasser till en given gemensam klass. Vilken av varianterna som skall väljas kan då inte avgöras när programmet kompileras utan först då det körs. Dessa begrepp är mycket viktiga vad gäller att skapa återanvändbara moduler. Ur struktureringssynpunkt har de fr.a. betydelse då det gäller att i programkoden bevara en god struktur från tidigare etapper i systemutvecklingsarbetet.

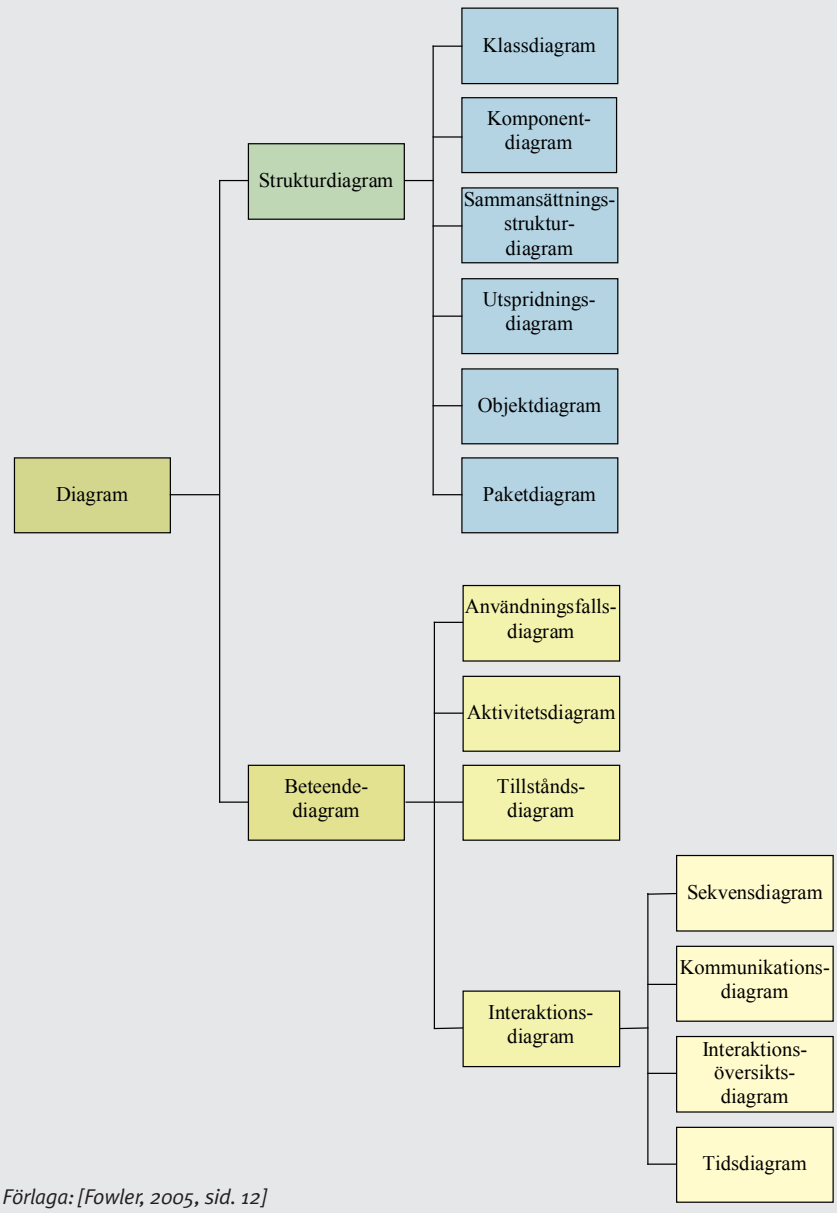
Arvsmekanismen underlättar således både återanvändning och förändring av programmoduler. Andra uppenbara fördelar med objektorientering är att ett och samma paradigm kan användas genom alla steg av programutvecklingen, analysen, designen och programmeringen. Dessutom faller den sig som sagt mycket naturlig för en systemmodellerare som skall skapa ett datorprogram av sin modell.

Även om objektorientering i sig alltså är ett i många stycken utmärkt arbetssätt, så bör

man ändå ha klart för sig att det också har sina svagheter. En är att begreppen inte alltid har en entydig definition, vilket leder till olika synsätt på hur man skall utforma metoder och verktyg. Kanske bör man också tänka på att för den som är ovan vid att tänka objektorienterat kan tröskeln för att ta till sig konceptet vara ganska hög.

12

Klassificering av UML:s diagramtyper

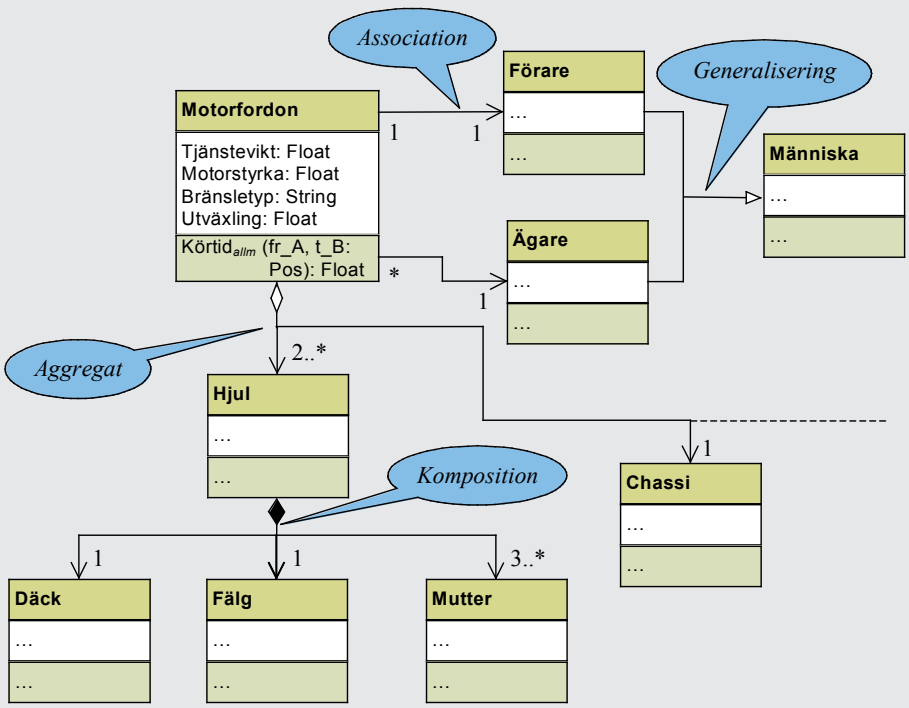


Förlaga: [Fowler, 2005, sid. 12]

UML – Unified Modeling Language

Ett generellt och vida använt modelleringsspråk – eller kanske snarare modelleringsnotation – som grundar sig på objektorienteringsparadigmet är UML – *Unified Modeling Language*. Det är resultatet av en sammanslagning och vidareutveckling av idéerna från främst tre av de mest dominerande metodansatserna inom objektorientering: OMT *(Object Modeling Technique)* av James Rumbaugh, OOSE *(Object Oriented Software Engineering)* av Ivar Jacobson och OOAD *(Object-Oriented Analysis and Design)* av Grady Booch. Det lanserades 1997 och två år senare antogs version 1.3 som standardspråk för specifikationer av OMG *(Object Management Group)*, en sammanslutning av en stor mängd företag runt om i världen. Numera är det version 2.0 som gäller. Genom industrins kraftiga uppbackning har det fått ett så kraftigt genomslag, att det är högst sannolikt att alla som sysslar med modeller kommer att i någon form komma i kontakt med det. Därför är det befogat att här ge en kort introduktion och översikt till UML. Den som vill sätta sig in mera noggrant i språket rekommenderas att studera ”*UML Distilled, Third Edition*” av Martin Fowler.

Klassdiagram 13



Ett motorfordon har 2 eller flera hjul. Ett hjul består av ett däck, en fälg och tre eller flera muttrar. Fordonet har en förare, som endast kan framföra ett fordon åt gången. Det har också en ägare, som dock kan äga fler fordon. Ägare och förare är båda specialiseringar av klassen människa.

UML är grafiskt och bygger på en stor uppsättning olika diagram. Standarden är dock deskriptiv snarare än preskriptiv, d.v.s. de regler som styr hur språkelementen används är mera en följd av användarnas sätt att förhålla sig till språket än en strikt formulerad syntax och semantik. Detta har naturligtvis vissa nackdelar med oklarheter à la naturligt språk, men också klara fördelar därigenom att det kan användas på alla nivåer i programutvecklingskedjan från de första idéutkasten till detaljerade planritningar för själva programmeringen. Det är dock inte ett programmeringsspråk, även om det kan ligga till grund för automatisk generering av vissa kodavsnitt.

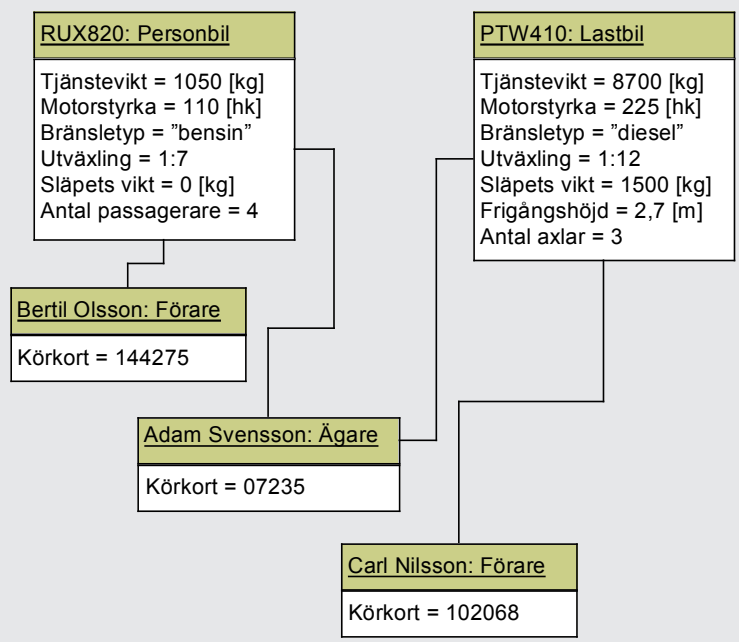
De diagram som ingår i UML är av två olika typer, strukturdiagram, som åskådliggör de statiska sambanden mellan de ingående objekten, och beteendediagram, som åskådliggör de dynamiska aspekterna i modellen. En fullständig uppräkning av de olika formerna finns i faktaruta 12. Här skall vi endast kommentera några av de vanligaste (och viktigaste).

Det allra vanligaste UML-diagrammet är utan tvekan **klassdiagrammet**. Med dess hjälp beskrivs de olika objekttyperna i ett system och de olika relationerna mellan dem. Vi har faktiskt redan stött på ett sådant diagram, i beskrivningen av motorfordon. Det var dock mycket enkelt såtillvida att den enda relation som åskådliggjordes var generalisering. Ett mera sammansatt diagram, som åskådliggör även andra relationer, visas i faktaruta 13. Där visas även attribut och operationer med typspecifikationer, vilka man vanligen brukar ta med i klassdiagrammen.

Till skillnad från klassdiagrammen visar **objektdiagrammen** enskilda objekt samt deras relationer och explicita attributvärden. De

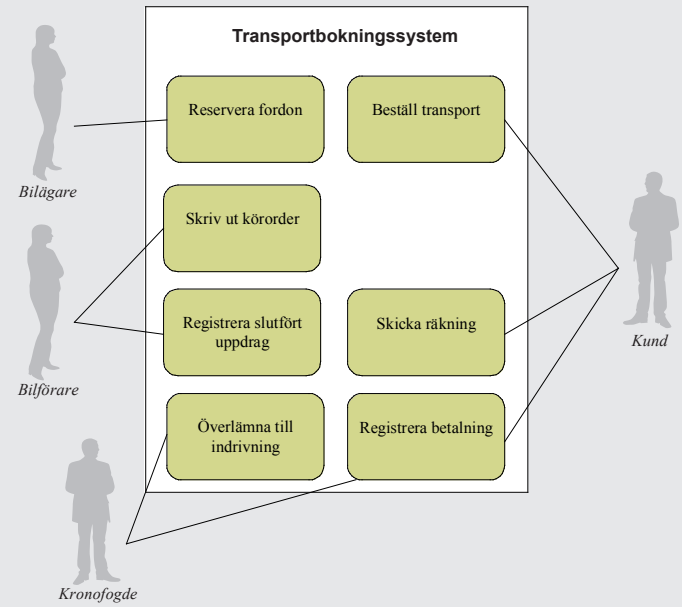
14

Objektdiagram



15

Användningsfallsdiagram



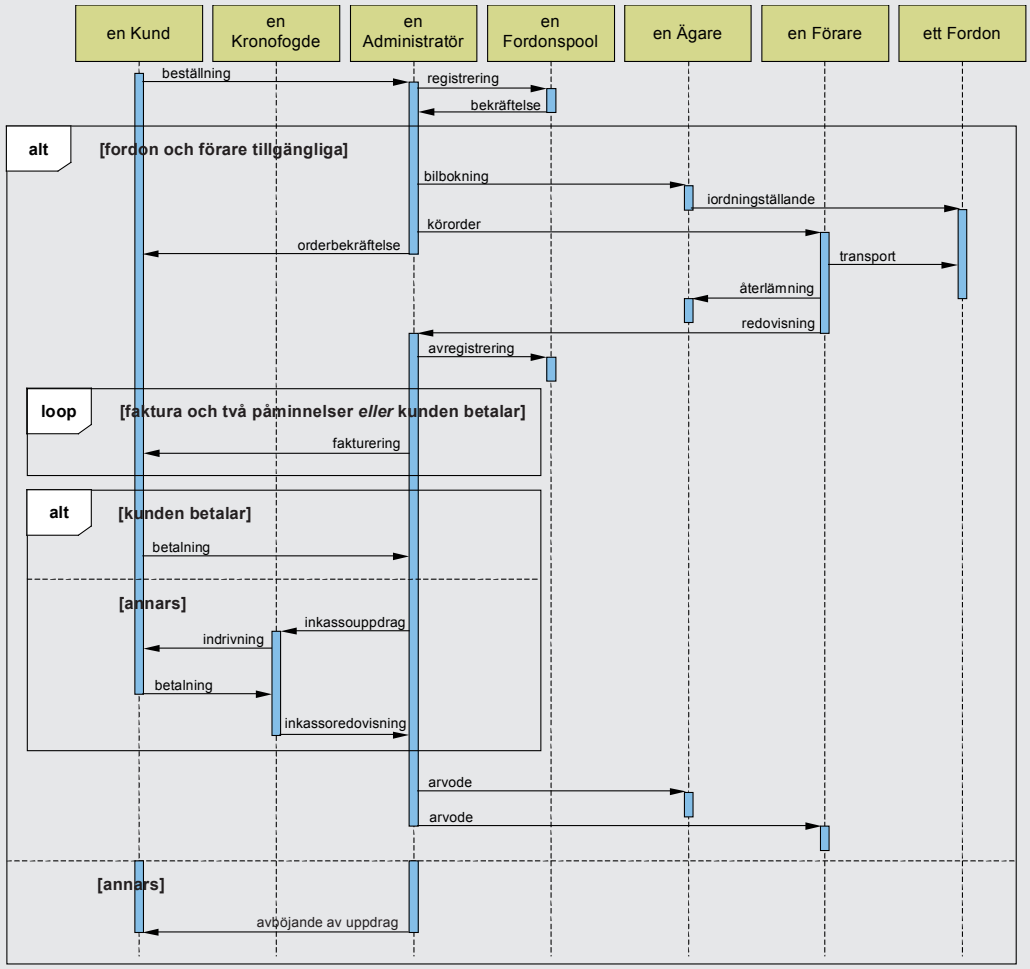
används därför fr.a. för att beskriva olika systemtillstånd eller för att ge exempel på olika möjliga konfigurationer av objekt. Varje objektnamn med sin klasstillhörighet markeras genom understrykning.

Vi lämnar nu strukturdiagrammen och tittar på några beteendediagram i stället. **Användningsfallsdiagrammen** är främst användbara då man vill åskådliggöra de funktionella kraven på programmet. De visar en helt och hållet extern bild av systemet och behöver därför inte särskilt tydligt korrelera till de interna klassdiagrammen. I stället åskådliggörs aktörerna, d.v.s. användarna i deras olika roller, i systemets omgivning och på vad sätt systemet tillgodoser deras behov. Med användare förstås då inte bara mänskliga varelser utan också andra system av skilda slag som nyttiggör sig av vad vårt program kan åstadkomma. Ett användningsfallsdiagram sammanfattar på så sätt alla de scenarier i vilka man kan tänka sig att systemet interagerar med aktörerna.

Med **sekvensdiagram** visar man meddelandeflödet mellan objekt med speciell hänsyn tagen till i vilken ordning de hanteras. Varje objekt representeras av en livslinje, där man kan utläsa när objektet är aktivt under simuleringen. Likaså blir det med detta diagram helt tydligt hur operationerna i de olika objekten kommer in i bilden. I princip kan samma sak som uttrycks i sekvensdiagrammen också visas i **kommunikationsdiagrammen**. Vilket man väljer är nog mer en fråga om tycke och smak än om faktiska skillnader. Man kan dock säga att man med den senare formen lägger mer vikt vid samarbetsrelationerna mellan objekten.

16

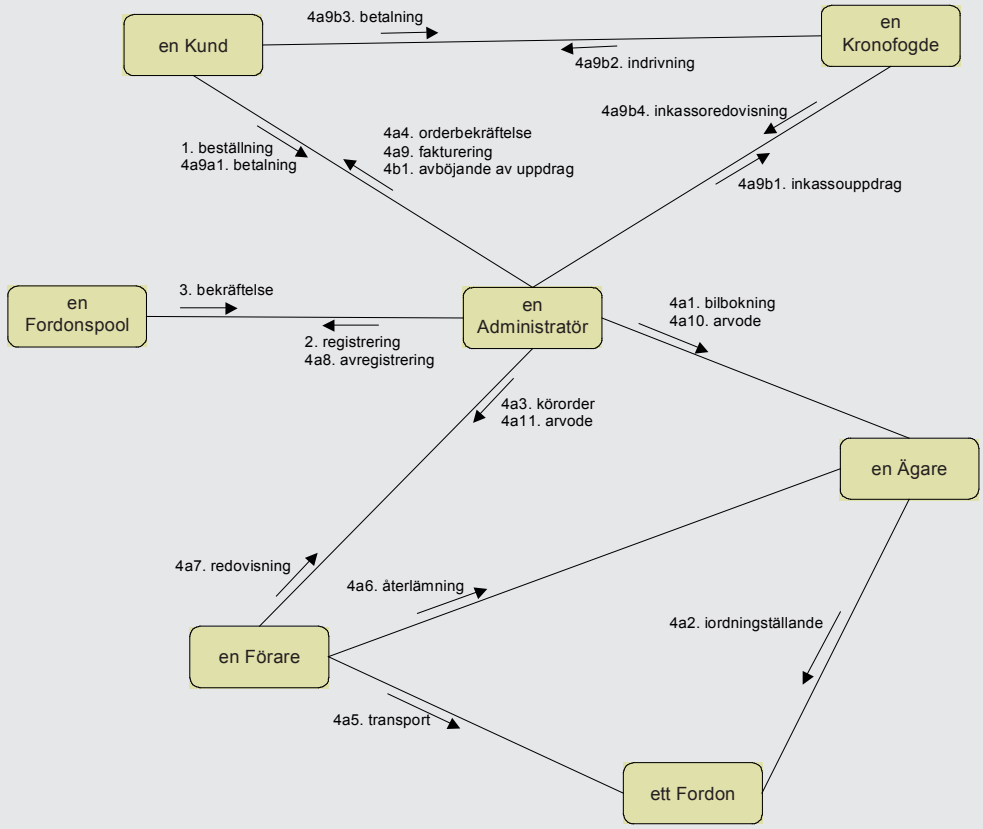
Sekvensdiagram



Här ges också exempel på s.k. interaktionsramar, som åskådliggör alternativ och loopar. I det förra fallet delas fältet inom ramen i två eller flera delar medelst streckade linjer, där varje alternativ åskådliggörs för sig.

17

Kommunikationsdiagram



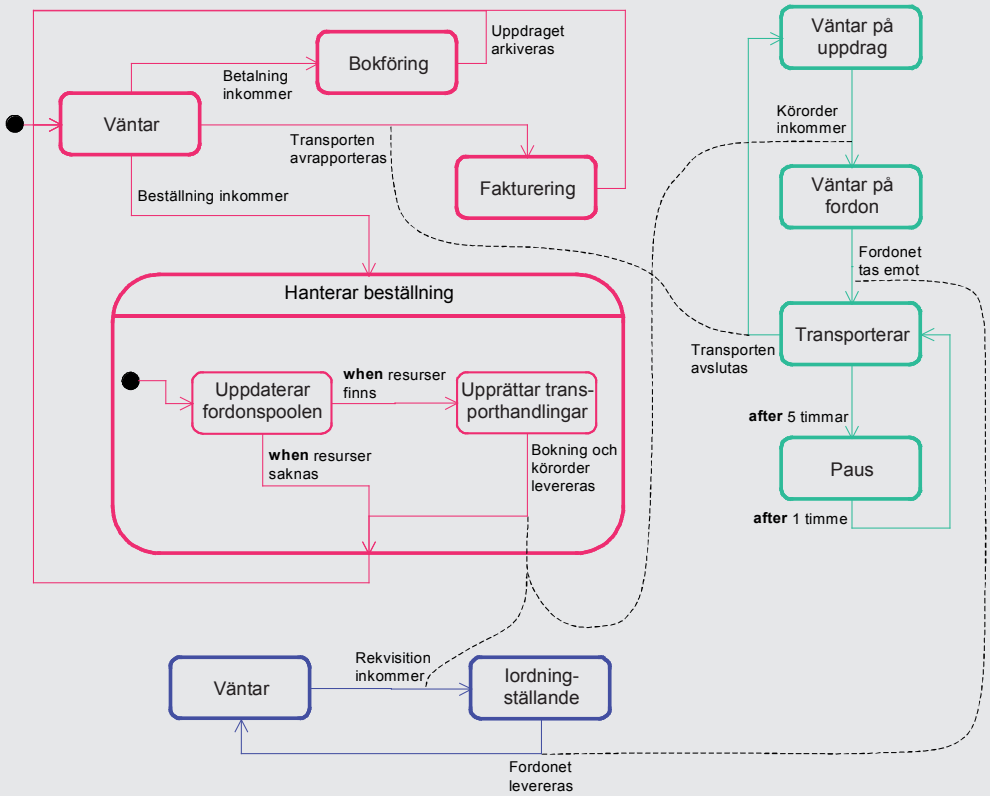
Numreringen av meddelandena kan göras på många sätt. Här används bokstäver för att ange alternativa vägar. I mera strikt UML-notation förordas annars nästade ordningsnummer (t.ex. 2.1.3.3.1....). I andra fall används rak numrering kompletterad med kommentartexter.

Ytterligare ett par beteendediagram förtjänar att lyftas fram. Det ena är **tillståndsdiagram**, som visar de tillstånd ett objekt kan genomlöpa som en reaktion på givna händelser. Det ger också möjlighet att beskriva tillstånden i hierarkiska strukturer. Om man kommer underfund med att flera tillstånd har delvis gemensamma övergångar och interna aktiviteter, så kan man därför beskriva ett ”övertillstånd”, där man lägger de gemensamma företeelserna.

Det andra diagrammet av liknande slag är **aktivitetsdiagrammet**. Det kan närmast jämföras med gamla tiders flödesscheman men med den utökningen att man här även kan representera parallella flöden. I princip visar diagrammet alltså i vilken ordning saker och ting måste

18

Tillståndsdiagram



Tillstånden visas för tre objekt: administratören (rött), fordonsägaren (blått) och föraren (grönt). Dessutom markeras med streckade linjer hur olika genererade händelser i objekten påverkar tillståndsövergångar i andra objekt. Varken färgerna eller de streckade linjerna är dock officiell UML. Vanligtvis gör man ett eget tillståndsdiagram för varje objekt, och låter de sammankopplande händelserna markeras med siffror eller kommentartext.

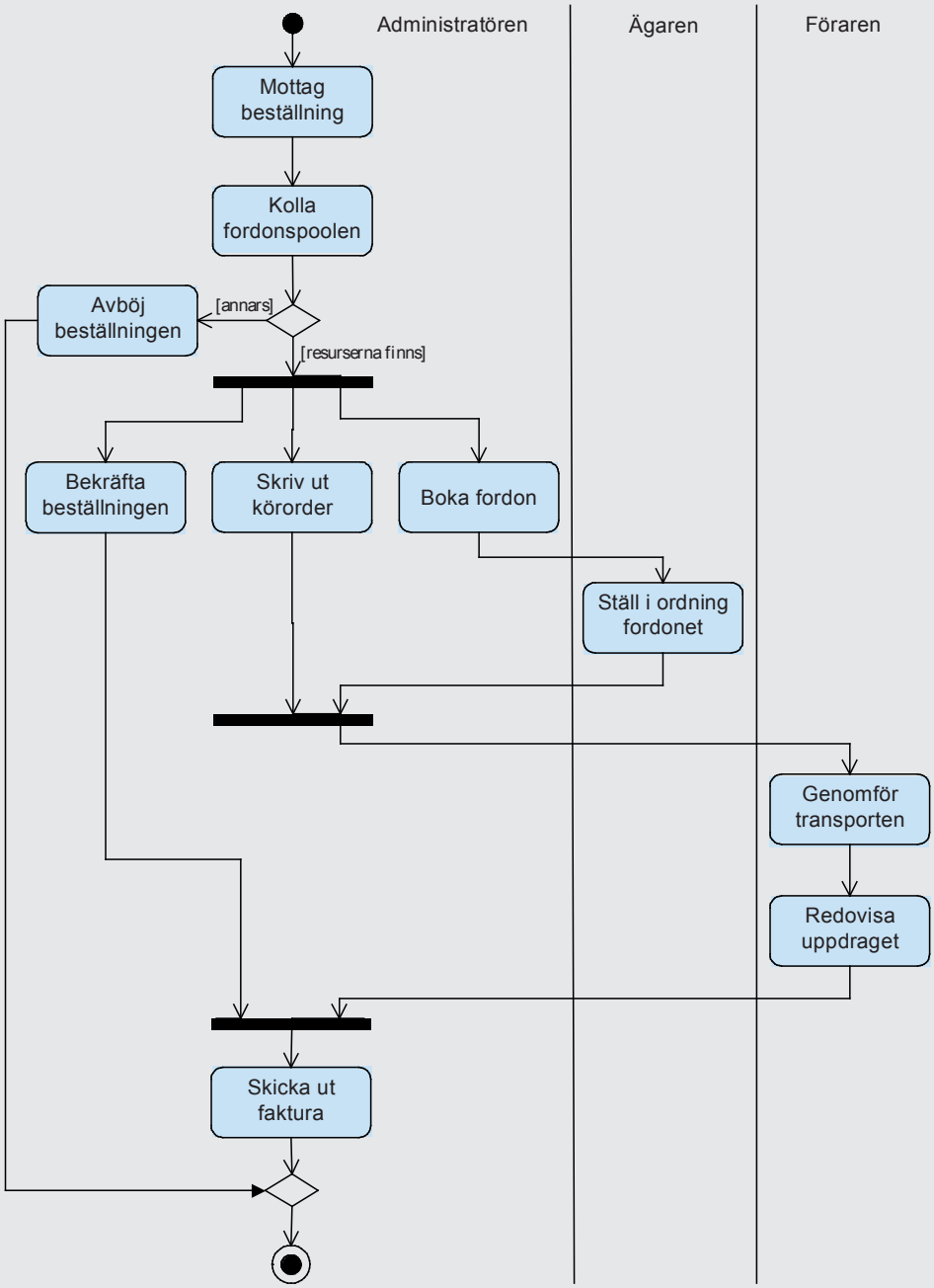
I detta diagram ser vi för övrigt även exempel på skyddsvillkor (eng. *guard conditions*; inleds med **when**) och tidsstyrda övergångar (eng. *time transitions*; inleds med **after**).

utföras i programmet och vilka objekt som är ansvariga för dem. I allmänhet kan man nog säga att även valet mellan aktivitetsdiagram och tillståndsdiagram är en smaksak, man uttrycker i stort sett samma sak med dem båda.

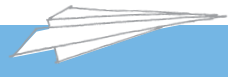
Vi kan alltså notera att UML innehåller många olika typer av diagram som ibland helt eller delvis uttrycker samma sak. Det kan ju förefalla tämligen onödigt, men man har ändå valt att införliva dem alla. Dels underlättar det för dem som är vana att tänka och uttrycka sig på ett visst sätt, dels visar det sig ibland vara värdefullt att uttrycka en och samma företeelse utifrån olika synvinklar, eftersom det kan ge nya insikter i ett problem som man trodde sig redan förstå.

19

Aktivitetsdiagram



Här visas vem som ansvarar för vad (kolumnindelningen), alternativa aktiviteter (romber) samt parallella aktiviteter (breda vågräta streck).



Konceptuell modellering

Låt oss nu återvända en stund till vår inledande diskussion om vad en modell är. Vi nämnde då att en modell som skall utnyttjas av fler än en person måste vara kommunicerbar, d.v.s. kunna uttryckas på ett naturligt språk.

En modell uttryckt i matematiskt eller algoritmiskt språk är i denna mening i allmänhet inte kommunicerbar utanför en snäv krets av experter. Men inte ens dessa experter klarar sig i längden utan en mer övergripande beskrivning av den detaljerade simuleringsmodellen.

Nackdelen med naturligt språk är ju dock att det ofta är ganska diffust. För att förstå en mening rätt, måste man vara väl förtrogen med talarens (eller skrivarens) kultur, och dessutom kanske tyda tonfall och kroppsspråk. En mening kan också uppfattas olika beroende på sammanhanget eller mottagarens referensram. Detta kan man emellertid råda bot på genom att man gemensamt kommer överens om entydiga definitioner av alla begrepp man behöver använda. Vi får då möjlighet att beskriva en modell på ett stringent sätt, och om vi dessutom undviker att inkludera allt som hör till implementeringen av simuleringsmodellen (t.ex. hur tiden stegas fram och vad som skall loggas), så har vi vad som brukar kallas en **konceptuell modell**. Man säger ibland att en sådan modell beskriver *vad* som modelleras, men inte *hur* det görs.

Konceptuell modellering är alltså modellering i termer av begrepp, d.v.s. den sker på en generisk abstraktionsnivå. Abstraktionerna består alltså mera i att man hanterar generella objekt och interaktioner än att man har atomära objekt på en hög abstraktionsnivå. Man kan därför säga att en konceptuell modell endast är *kvalitativ* till sin natur. Ett annat vanligt sätt att uttrycka det är att en konceptuell modell är den första (och högsta) abstraktionen av den företeelse som skall modelleras.

Kanske skulle man kunna förledas att tro, att konceptuella modeller huvudsakligen utvecklas sedan simuleringsmodellen är klar för att ingå i dokumentationen. Säkert sker det så ibland, men då missar man poängen. Snarare bör man först göra en konceptuell modell och sedan låta den ligga till grund för programutvecklingen. Därigenom kommer den också att vara en av de mest centrala utgångspunkterna för valideringen av simuleringsmodellen.

Detta förutsätter naturligtvis att man redan från början har klart för sig viken typ av simule-

ringsexperiment man kommer att genomföra. Det hjälper till att göra hela modelleringsprocessen mycket mer koncentrerad. ”Klassisk” fysik kan tjäna som exempel: Där har man en a priori-kunskap om det studerade systemet, lägger till en hypotes, genomför experiment samt accepterar eller förkastar tesen utifrån resultatet av experimentet. Vi ser här i hur hög grad konceptuell modellering beror av kunskapen om tillämpningsdomänen.

Genom konceptuell modellering skapas en gemensam referensram för verksamhetsföreträdare, forskare och utvecklare. Denna referensram innefattar förutom den väldefinierade begreppsapparaten en samsyn om vilka processer, objekt och interaktioner som är väsentliga inom problemområdet. Det blir även möjligt att tidigt identifiera kritiska avsnitt för vidare analys. Härigenom läggs en god grund för den fortsatta utvecklingen, då alla berörda kan referera till samma konceptuella modell. Likaså förtjänar att framhållas, att goda konceptuella modeller kan undanröja en hel del av de problem som kan uppstå på grund av de underförstådda förutsättningar som är inbyggda i alla slags modeller men som sällan eller aldrig brukar redovisas explicit i dokumentationen.

Skall man exemplifiera konceptuell modellering, blir det lätt så omfattande att det svårligen ryms inom en faktaruta på en sida. Därför har det illustrerande exemplet till detta kapitel lagts för sig. Se alltså bilaga 1, där en konceptuell modell från en tidig fas av en modellutveckling beskrivs.

Ett alternativt synsätt

Den beskrivning av konceptuell modellering, som vi nu har sett, omfattas inte av alla. Vilken innebörd man lägger i detta begrepp kan skifta ganska mycket beroende på från vilken utgångspunkt man närmar sig det. En utvecklingslinje kommer från teorin för systemanalys och systemdesign. En annan har sin rot i behovet av entydiga och auktoriserade beskrivningar av komplicerade företeelser, t.ex. militära operationer (jämför uppslagsböcker).

För att ge någon inblick i alternativa synsätt, skall vi kort presentera den något avvikande (men ganska vanliga) beskrivning av konceptuell modellering, som vi finner hos dr Dale Pace vid John Hopkins University, en av de främsta teoretikerna på konceptuell modellering för militära ändamål. Han har en något bredare syn på vad begreppet står för och anser att konceptuella modeller består av följande tre komponenter. För det första har vi **simuleringssammanhanget** (eng. *simulation context*), som är en på något sätt stadfäst information om de förhållanden som skall modelleras. Den innehåller uppgifter om relevanta enheter och processer, antaganden och beteenden, och den sätter gränser för nästa komponent, **uppdragsrymden** (eng. *mission space*). Här ligger fokus på hur de olika modellelementen representeras med avseende på tillstånd, uppdrag, beteende etc.

Så långt överensstämmer Paces beskrivning med vad som ovan sagts. Han inför emellertid ytterligare en komponent, **simuleringsrymden** (eng. *simulation space*), där man samlar information om alla praktiska ramar för modellen, t.ex. observerbarhet, tidsaspekter (t.ex. snabbare än realtid), behov av speciell hårdvara. Medan de båda första komponenterna är implementationsoberoende är alltså denna tredje komponent implementationsberoende.

Att utveckla en konceptuell modell sker enligt Pace i ett iterativt förfarande bestående av fyra steg. Först samlar man in auktoriserad information om sammanhanget för den modell som skall utvecklas. Därefter identifierar man

de enheter och processer som skall representeras samt utvecklar dem i enlighet med de representationsformer som skall användas. Slutligen identifieras och beskrivs relationerna mellan de fastställda enheterna.

Beträffande problemet att identifiera de adekvata objekten beskriver han en logisk grund för hur det skulle kunna gå till. Man rekommenderas att leta efter följande kategorier av modellelement: Sådana företeelser som finns explicit upptagna i modellkraven hör naturligtvis dit, likaså bör det finnas ett modellelement för varje fenomen som är potentiellt intressant att värdera i den företeelse som modelleras. Elementen bör i största möjliga utsträckning ha en motsvarighet i den verkliga världen, inte minst med tanke på svårigheterna att annars få fram lämpliga data. Uppdelningen i modellelement bör så långt det är praktisk möjligt följa praxis från andra modelleringssammanhang, så att jämförelser, återanvändning och samverkan underlättas. Speciella element för approximationer och andra beräkningsfrämjande åtgärder, vilka inte kan hänföras till ovan angivna elementtyper, bör användas restriktivt. Inga andra modellelement bör över huvud taget komma i fråga, för varje onödigt element är en källa till framtida problem vid simuleringarna.

Hur de karaktäristiska egenskaperna för de olika elementen sedan representeras bestämmer noggrannheten och precisionen i modellen. För att hitta rätt nivå bör man utgå från flera olika utgångspunkter för att fånga in flera olika perspektiv.

De centrala begreppen inom konceptuell modellering

Då man skall utveckla en konceptuell modell är det två faktorer som är centrala. Dels behöver man ett språk, dels behöver man ett utvecklingsmönster eller en förebild, d.v.s. ett paradigm.

Vilket språk man väljer för att uttrycka sin modell beror i hög grad på vad som skall beskrivas och i vilken utvecklingsfas man befinner sig. När man inleder projektarbetet och försöker formulera de första kraven på den framtida simuleringsmodellen, så behöver stringensen i språket inte vara lika stor som när man formulerar den konceptuella modell som skall ligga till grund för designen av simuleringsmodellen. Man kan också tänka sig att man använder olika språk för beskrivningar på olika abstraktionsnivåer. En modell på strategisk nivå av hela rikets försvar fordrar andra uttrycksmedel än en modell av en pansarplutons stridstekniska aktiviteter.

Det språk man väljer kan vara informellt eller formellt. I båda fallen gäller dock kravet att alla begrepp måste vara väl definierade. Ett informellt eller ”naturligt” språk har fördelen att det är relativt lätt att använda och förstå för en människa. Det kan dessutom stödjas med olika former av grafisk framställning, diagram, stiliserade teckningar, foton etc. Nackdelen är att syntax och semantik inte är entydigt definierad, vilket kan vara en fördel för vitsaren men lätt orsakar missförstånd mellan deltagare i ett modellutvecklingsprojekt. Denna egenskap gör det också svårt eller rent av omöjligt att utveckla datorverktyg som editorer, syntaxkontrollanter, programkodsgeneratorer o.dyl.

Formella språk har å andra sidan en väldefinierad syntax och semantik, så med dem kan tveksamheter och missförstånd lättare undvikas. Likaså kan de relativt lätt stödjas med datorbaserade verktyg. Nackdelen i detta fall är att sådana språk har en begränsad uttryckskraft och att de är svåra att förstå och använda för den ovane.

Vad gäller valet av utvecklingsparadigm styrs man i hög grad av var man har sitt fokus. Ser man den modellerade företeelsen som en samling samverkande objekt, så faller det sig naturligt att tillämpa ett **objektorienterat tillvägagångssätt**. Om man i stället ser företeelsen som ett spel mellan självständiga och beräknande individer får man naturligtvis i stället en **aktörorientering**. Fokuserar man i stället på att företeelsen utgör ett flöde eller en process ligger det närmare till hands att tillgripa **processorientering**, såvida det inte handlar om en process med ett väl definierat slutresultat, då kan **målorientering** vara det adekvata paradigmet.

CMMS

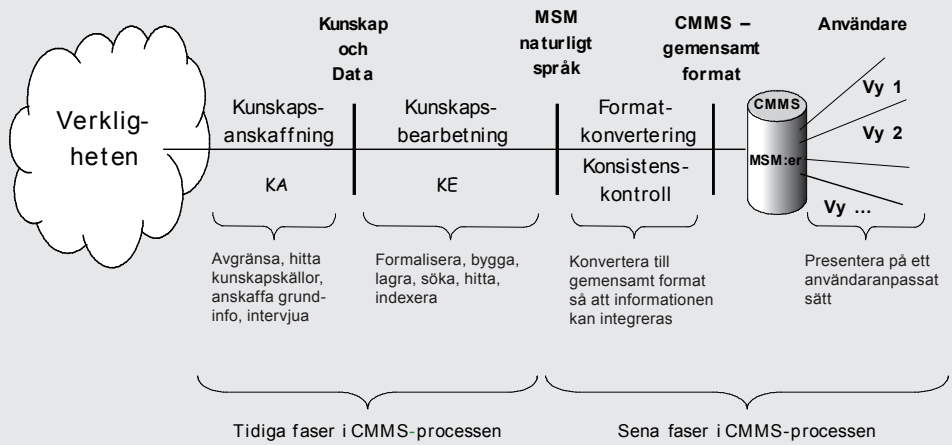
Att utveckla konceptuella modeller är som sagt av stort värde, men det är också förenat med stora problem. Ett sådant är att trovärdiga eller auktoriserade informationskällor ofta är svåra att finna. Det leder vanligen till att olika modellutvecklingsprojekt förlitar sig på olika källor av skiftande kvalitet (kanske bara ett allmänt tyckande) för samma information, vilket i sin tur lätt leder till förvirring. Ett annat problem är att även om den meddelade kunskapen kommer från en auktoritativ källa, så kan den vara ofullständig eller tvetydig. Detta leder då till att modellerarna fyller i med gissningar och förutfattade meningar eller omedvetet vantolkar informationen. Ett tredje vanligt problem är att den insamlade informationen, som ofta fordrat avsevärda insatser, sällan bevaras för framtida användning. Följden blir att vissa senare projekt aldrig kommer till stånd på grund av onödigtvis alltför höga kostnader, eller att icke önskvärt dubbelarbete måste utföras.

År 1995 fastställde amerikanska försvarsdepartementet en övergripande plan för modellerings- och simuleringsverksamheten inom sitt ansvarsområde (*Modeling and Simulation Master Plan*). Målet var att råda bot på ovan nämnda problem bland många andra genom att skapa en struktur och ett gemensamt ramverk för allt sådant arbete. I detta ramverk utgör CMMS (*Conceptual Models of Mission Space*) en viktig komponent. Idén är att man skall utveckla konceptuella modeller av all väsentlig militär verksamhet, vilka skall utgöra en gemensam utgångspunkt vid byggandet av konsistenta och auktoriserade simuleringsmodeller. De skulle också underlätta interoperabilitet mellan och återanvändning av simuleringsmodeller. Dessa tankar har sedermera trängt ut i världen, fr.a. inom NATO, och även i Sverige har man nu insett nyttan av dem.

CMMS innebär alltså att man skapar och underhåller ett bibliotek med konceptuella modeller, MSM:er (*Mission Space Models*), av alla militära operationer och processer. Dessa modeller skall vara konsistenta, funktionella och strukturerade beskrivningar av verkliga eller tänkta företeelser av intresse. För att detta skall vara möjligt behövs ett gemensamt tekniskt ramverk – standarder för att samla in kunskap (KA, *Knowledge Acquisition*) och för att bearbeta och integrera kunskap (KE, *Knowledge Engineering*) – där en syntax och semantik för enhetlig beskrivning utgör de viktigaste komponenterna. Vidare innehåller detta ramverk en processbeskrivning för hur man skapar och underhåller konceptuella modeller, liksom standarder för datautbyte. För att detta skall fungera behövs även ett databassystem för registrering, lagring och publicering av modellerna och deras förutsättningar. Likaså behövs gemensamma verktyg för att man skall kunna överblicka, hitta i, hämta från, skapa och leverera nya MSM:er med tillhörande information till databassystemet.

20

De olika stegen vid transformering av kunskap och data i CMMS-processen



Förlaga: [Mojtahed et al., 2003, sid. 11]

Ett väl underhållet och innehållsrikt CMMS byggt på auktoriserad information kan då fungera som en brygga mellan de militära experterna och modellerarna. Det fordrar dock att försvarsmakten har en organisation med befogenhet och kraft att driva igenom att alla modelleringsprojekt skall leverera bidrag till biblioteket och dessutom använda sig av det som redan finns där. Biblioteket måste hela tiden utvecklas för att vara en levande kunskapsbas, och inte bli en historisk artefakt, som endast intresserar ur ett musealt perspektiv.

En central del i CMMS är alltså KA/KE, så låt oss därför titta lite närmare på dessa begrepp. Kunskapsinhämtningen (KA) sker i princip dels från statiska källor (d.v.s. litteratur), dels från levande källor (d.v.s. experter). De senare är vanligen de svåraste att nyttiggöra, eftersom en människas kunskaper hela tiden förändras med nya erfarenheter. Dessutom är det få människor som verkligen vet vad de vet. Mycket av kunskapen har med tiden blivit så självklar att den ligger i det omedvetna. (Försök att förklara hur man gör när man cyklar!) Här fordras en öppen dialog, som får ta tid, och det är mycket sannolikt att den behöver återupptas många gånger för att förklara otydligheter eller rena luckor, som man under det fortsatta modelleringsarbetet blir uppmärksammad på.

Andra påtagliga problem med kunskapen är att den kan vara enorm och utvecklad ”distribuerat”, varför den kan vara ytterst svår att överblicka ens för en grupp av experter. Dessutom kan auktoritativa källor, som normalt är bra och värdefulla, bli konserverande och skymma möjligheten till ett utvecklande paradigmskifte, om de inte riktigt hänger med i utvecklingen. Därför behövs även ”vildhjärnor” som Nicolaus Copernicus eller Galileo Galilei, som ifrågasatte sin tids ”auktoritativa källor”, men de skall tas med vid rätt tid och i lagom mängd.

En väl genomförd kunskapsinhämtning bör bestå av följande fyra steg:

- Bestäm det sammanhang som avses, d.v.s. bestäm gränser och omfattning för arbetet (jfr användningsfall i UML). Utgångspunkten bör då vara vilka förväntade användarvyer som kan tänkas bli aktuella (jfr figuren i faktaruta 20), annars är risken stor att man missar väsentliga aspekter.
- Bestäm abstraktionsnivån för att därigenom avgöra vilken kunskap som måste inhämtas om de aktuella aktiviteterna.
- Identifiera vilka källor som finns att tillgå. När det gäller expertmedverkan, är det en god regel att företrädesvis försöka engagera sådana personer som har aktuell erfarenhet på rätt operativ nivå. Undvik alltså generalen och tala med sergeanten om modellen handlar om pansarbandvagnars och besättningars beteende vid olika typer av terränghinder! Man talar ibland om **auktoriserad** resp. **auktoritativ** källa. Det förra är en av en myndighet utpekad normsättande skrift (eller kanske person), medan det senare är en person (eller kanske skrift) som är allmänt erkänd som kunnig.
- Samla in kunskapen. Detta kan bli en ganska tidskrävande process, där deltagarna tvingas penetrera informationen i grunden tills alla är säkra på att de förstår vad alla begrepp står för. Det är då lätt gjort att man hittar många nya trådar att nysta i, men skall man komma till ett lyckligt slut med denna aktivitet, är det av största vikt att man behåller fokus på det mål man definierat i de föregående punkterna.

I kunskapsbearbetningen (KE) formaliseras den insamlade informationen, man rensar den från adiafora, strukturerar den och dokumenterar den på ett formellt sätt. Därvid framträder säkerligen diverse brister och tvetydigheter, vilka fordrar omtagningar av KA-steget. Det är även nyttigt att man tillåter olika perspektiv på den modellerade verkligheten att påverka utformningen av vår konceptuella modell. Ett flygplan kan t.ex. beskrivas på ganska olika sätt av en attackpilot, en luftvärnsoperatör och en jaktledare. Att utnyttja flera olika utgångspunkter leder till dels att man lättare kan uppdaga begreppsförvirringar mellan olika verksamhetsområden (vad menas t.ex. med ”låg höjd” – höjden över marken som för flygaren eller låg siktvinkel som för luftvärnsskytten?), dels att modellen blir användbar som underlag för ett bredare spektrum av simuleringsmodeller.

En svensk studie (se [Mojtahed et al., 2003]) har visat att den produkt som skall framställas måste möta höga krav på både flexibilitet och beständighet. Därför måste KE-steget understödjas av en gemensam syntax och semantik grundad på

- en militär ontologi,
- ett lexikon med gemensamma militära, naturvetenskapliga, samhällsvetenskapliga m.fl. begrepp liksom en uppsättning med modelleringsbegrepp (generella fackuttryck för modellering och simulering, t.ex. objekt, verksamhet, uppgift, interaktion, tillstånd) samt
- ett modelleringsspråk (som med fördel skulle kunna vara UML).

CMMS är som idé väl etablerad, men endast i ringa utsträckning genomförd. En svårighet är att ett bibliotek med konceptuella modeller som täcker stora delar av ett lands militära verksamhet naturligtvis bör hållas hemligt. Även om informationsutbytet därigenom blir

begränsat vet vi dock att USA har genomfört en del av dessa idéer och att en del andra länder har viss verksamhet på gång. Men även om vi i Sverige ännu inte har etablerat den formella grunden för CMMS, så bör de bakomliggande idéerna kunna vara vägledande i det militära modelleringsarbetet. Visionen är att inför varje ny simuleringstillämpning skall man med en måttlig arbetsinsats kunna sätta ihop en ny skräddarsydd modell genom återanvändning av de konceptuella modeller som redan finns i biblioteket. Det skall alltså vara möjligt för olika personer i olika roller och från olika verksamhetsområden att arbeta med samma konceptuella modell men att implementationerna kommer att vara olika beroende på syftet i varje särskilt fall.

21

Exempel på olika användningar av en konceptuell modell

Utgående från exemplet i bilaga 1 kan någon tänka sig att t.ex. utveckla en modell med vars hjälp man skulle kunna analysera hur skadebilden i befolkningen ser ut vid olika typer av explosionsolyckor. Det skulle kunna vara ett bra hjälpmedel för sjukvårdsplaneringen inom regionen. I detta fall väljer man kanske att implementera en enda explosionspunkt, vars läge användaren får peka ut i en kartbild, medan man lägger stor vikt vid att placera ut människorna enligt en mycket verklighetstrogen modell. Resultatpresentationen utformas med fokus på hur fördelningen av skadade och döda ser ut (t.v.).

Någon annan kanske utnyttjar samma konceptuella modell men med avsikten att skapa ett instrument för analys av hur en stadsmiljö förändras under en militär strid och hur soldaterna på bästa sätt skall utnyttja denna miljö. I detta fall lägger man större vikt vid att få en realistisk utspridning av stridsdelarna, medan människorna kanske endast är intressanta indirekt, t.ex. genom att man kanske vill veta hur stor risken är att befinna sig på vissa bestämda platser. Man implementerar alltså människoutplaceringen genom att låta användaren peka ut dessa intressanta punkter. Resultatpresentationen utformas i detta fall med fokus på byggnadsstatus och rasmassor (t.h.).

Modellerings- och simuleringsprocessen – ett projektperspektiv

Att behärska alla verktyg och metoder för utveckling av användbara modeller är bra men knappast tillräckligt. Många modeller är resultatet av ett lagarbete, där människor med olika specialistkunskaper samverkar till ett gemensamt resultat.

Man genomför helt enkelt ett projekt tillsammans, och kommer nu alla deltagare med var sin uppsättning verktyg och metoder eller uppfattning om vad som skall modelleras, så blir det lätt lite kaotiskt. Och vad händer när man väl har lyckats åstadkomma en gemensam modell? Genomför man några simuleringar för att sedan lägga den åt sidan i tro på att kanske fortsätta arbeta med den vid något senare tillfälle? Hur mycket minns man då av hur modellen var tänkt?

Som vi har sagt redan tidigare är modeller kunskap. Att utveckla kunskap och förståelse är ofta ett intrikat arbete. Att förvalta kunskap kan också vara besvärligt. Skall detta lyckas fordras att alla deltagare har en medvetenhet om den process som modellutveckling och modellanvändning innebär.

En (datoriserad) modell är till stor del **programvara**. Samtidigt är varje programvara en modell av något. Exempelvis är ett ordbehandlingsprogram en modell av en skrivmaskin, fast sådan program har med tiden blivit så sofistikerade att de snarare är att betrakta som modeller av skrivmaskin + en del av författarens, grafikerns, sättarens m.fl. verksamhet.

En (datoriserad) modell är naturligtvis också ett **system**, ett begrepp som är lika flitigt använt som det är mångtydigt. Vi kan här nöja oss med att använda en beskrivning från [Nationalencyklopedins ordbok, 1996], enligt vilken ett system är en ”sammanhängande helhet med delar som står i vissa relationer till varandra och som fungerar efter vissa principer”. Det kan vara naturligt förekommande eller konstruerat, konkret eller abstrakt. Delarna kan ju sedan i sin tur vara system, så vi känner igen situationen från vår tidigare diskussion om grundobjekt. Se vidare bilaga 3!

Dessutom har vi ju konstaterat att modellerings- och simuleringsverksamheten är ett **projekt**. Tar man nu fasta på modellarbetets programvaru-, system- och projektegenskaper, så kan man utnyttja att det genom åren har formulerats många beskrivningar (eller egentligen modeller) för hur man bedriver ett framgångsrikt projektarbete eller en lyckad system- eller programvaruutveckling. Särskilt

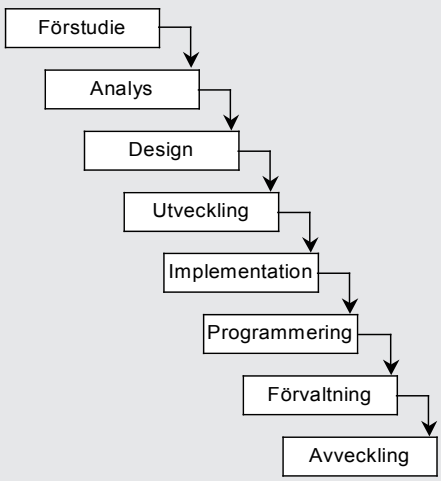
har modellerna av programvarors hela livscykel från behovsidentifiering till avveckling varit näraliggande till vårt ämnesområde. Under senare år har det kommit fram flera nya beskrivningar med ett tydligare fokus på just modellerings- och simuleringsaspekterna.

Från att från början ha varit ganska enkla och sekventiella (t.ex. ”vattenfallsmodellen”, se faktaruta 22; jfr avsnittet ”Livscykelmodeller för programvaror”, sid. 27) har modellerna för system-, program- och modellutveckling blivit mer och mer detaljerade och iterativa. Egentligen är de alla ganska lika, då de ju bara är en formalisering av allmän erfarenhet och ”sunt förnuft”. De bör därför närmast betraktas som minneslistor över vad som bör beaktas och inte en absolut mall som skall tillämpas i alla väder. Men använda just med ”sunt förnuft” är de utmärkta hjälpmedel för att ge stadga och framgång i ett modellerings- och simuleringsprojekt.

22

Vattenfallsmodellen

beskriver utvecklingen av programvara som en metodisk process, där varje steg skall vara helt klart och kvalitetssäkrat innan man tar nästa steg.



Denna modell har dock fått mycket kritik för att vara oflexibel då det trots allt kan bli fel på vägen och det då blir omständigt att korrigera felet.

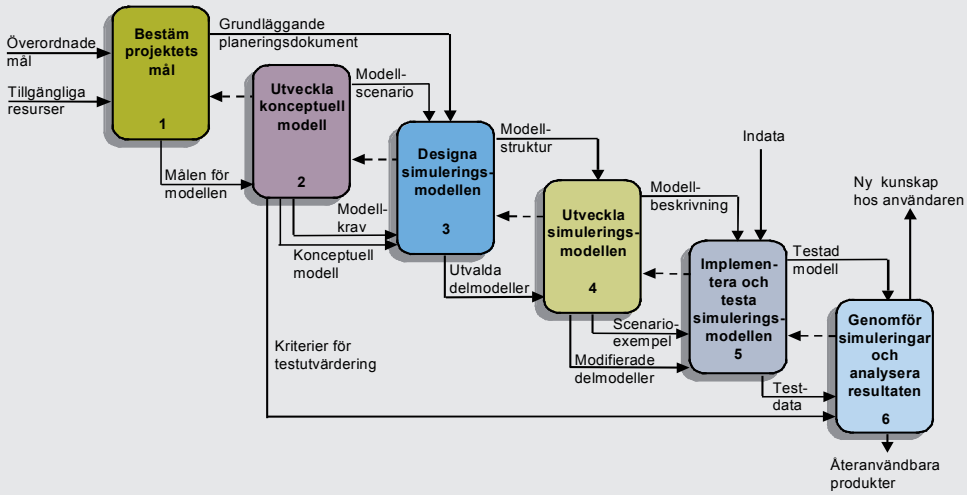
Låt oss nu fördjupa oss i modellerings- och simuleringsprocessen med hjälp av en lämplig modellformulering, som närmast är inspirerad av FEDEP (*Federation Development and Execution Process*). Den åskådliggörs i faktaruta 23. Vi skall strax diskutera de sex stegen mera ingående, men först skall vi notera de streckade pilar som pekar bakåt i bilden. De representerar de delprocesser, som gör att denna modell inte är en renodlad vattenfallsmodell. ”Vattnet” kan s.a.s. även rinna uppströms. De säger oss alltså, att då man i ett steg stöter på en felaktighet, tvetydighet eller annan oklarhet, så är det ofta förnuftigt att gå tillbaka till föregående steg (eller kanske något ännu tidigare steg) och lösa problemet där. Att exempelvis i utvecklingssteget försöka kompensera för en ofullständighet i designen eller i den konceptuella modellen leder nästan alltid till sämre lösningar än om man går tillbaka till respektive steg och gör en omarbetning där.

Bestäm projektets mål

Låt oss anta att vi skall dra igång ett stort projekt, där en väsentlig del består i att utveckla och använda en modell över ett ganska sammansatt problemområde. Det kan kanske gälla en träningssimulator för stridsvagnsbesättningar, eller kanske är det fråga om planering av logistiken för ett pansarskytteförband i en fredsbevarande utlandsinsats. I det förra fallet är det själva modellen som är projektresultatet, i det senare är det snarare en rekommendation om organisation och handlingssätt, d.v.s. modellen är här bara ett hjälpmedel för att uppnå den eftersökta kunskapen.

23

Modell av modellerings- och simuleringsprocessen



Ett tredje exempel kan vara utforskandet av rent fysikaliska fenomen som akustisk vågutbredning i grunda skärgårdsvatten. Här kan modellen vara både ett medel för att få fram den levererade kunskapen *och* själva resultatet. I så fall är det en modell som på ett lättanvänt sätt sammanfattar kunskapen om området, ungefär som gamla tiders tabellverk.

Under alla förhållanden är det nödvändigt att redan från början av projektet komma fram till en hållbar **problemformulering**. Om den inte finns formulerad redan före projektstarten, så är det viktigt att ägna denna fråga så mycket tid och uppmärksamhet som behövs för att alla inblandade skall vara ense om och förstå den. Det kan vara svårare än det låter, speciellt då projektet innehåller experter på olika ämnesområden. De har från början sannolikt ganska olikartade uppfattningar om hur helheten skall se ut beroende på olikartad modelleringspraxis inom resp. ämnesområde, olika uppfattningar om vad som är viktigt och bristande förståelse för de andras problemområden.

En problemformulering måste besvara frågan om vad som är problemdomänen, d.v.s. staka ut gränserna för vad modellen egentligen skall användas till. Handlar det om en modell av indirekt eld mot tätbebyggt område, så måste man bestämma sig för att strunta i att inkludera direktskjutande vapens verkan, hur lockande eller enkelt det än kan tyckas vara. Kommer man så småningom fram till att man egentligen vill ha med även dessa vapen, får man återvända till problemformuleringsfasen och omdefiniera problemdomänen.

Inte bara bredden utan också djupet måste definieras under denna fas. Vi måste alltså fråga oss vilket system det egentligen är, som vi skall simulera, och vilka dess minsta delar (de atomära objekten) skall vara, d.v.s. hur omfattande verklighet vill vi kunna hantera, och vilken grund anser vi oss kunna utgå ifrån.

Till denna fas hör också frågan om vad modellen skall användas till, vilka frågor som man hoppas kunna besvara med dess hjälp, eller vilken typ av utbildning och träning, som skall kunna genomföras med modellens hjälp, o.s.v. Är t.ex. målet en analys av fördelar och nackdelar med olika taktiska dispositioner, så är det kanske inte nödvändigt att göra modellen lika naturtrogen som om målet är att använda den för träning av t.ex. artilleriets eldledare.

Slutligen bör vi kanske t.o.m. fråga oss om det över huvud taget är bra eller ens meningsfullt att använda simulering som verktyg för att finna en lösning på den aktuella problemformuleringen. Kanske är det rent av bättre att låta några insiktsfulla personer resonemangsvis komma fram till en lösning, eller kanske ett manuellt krigsspel ger tillräckliga insikter för att man skall kunna identifiera en tillräckligt bra lösning. Allt blir inte bättre här i världen när man gör det med datorer.

Ytterligare en utgångspunkt för vårt projektarbete är de **ramar och förutsättningar**, som vi måste anpassa oss till. Det är en dygd att ”rätta mun efter matsäcken” och inte göra arbetet vidlyftigare än att man kan vara någorlunda säker på att klara av det. Man bör alltså redan från början skaffa sig en uppfattning om vilka kompetenser som behövs och i vad mån dessa finns tillgängliga över huvud taget. Och om nu alla de personer, som har de efterfrågade kompetenserna, verkligen finns, så måste man fråga sig, om de finns tillgängliga just när de behövs i detta projekt.

Det senare hänger ihop med att ett projekt vanligtvis har givna tidsramar, ett slutdatum, då modellen/kunskapen avses att användas till något, och kanske några andra avtappningstidpunkter. Problemet är, att all planering av modellerings- och simuleringsprojekt är planering under stor osäkerhet. Vissa delar kan fordra djupare analyser och forskning, vars omfattning kan vara svår att i förväg uppskatta. Det finns även en stor risk att underskatta tidsåtgången för de delmodeller, vilka man från början tycker sig ha en god förståelse för. Det är mycket vanligt att tro att när man ser de stora dragen så är problemet mycket nära sin lösning. Erfarenheten säger i stället att när man preciserar sin modell så upptäcker man luckor och obalanser som fordrar betydligt djupare analyser än man från början trodde. Det är alltså klokt att planera in marginaler för sådana oförutsedda belastningar. Mindre lyckat är att under pågående projekt upptäcka att man inte kan hålla tiden och att då försöka engagera fler människor i arbetet. Dels tar det tid att få de nya medarbetarna delaktiga i projektarbetets kultur och ramar, dels ökar den administrativa bördan.

En tredje väsentlig förutsättning är vilka ekonomiska resurser som står till buds. Att missbedöma hur omfattande modell man kan få för en viss summa pengar leder mycket lätt till att hela projektet misslyckas. Att utveckla, formalisera och kvalitetssäkra kunskap, som ju modellering och simulering egentligen är, är i allmänhet mer resurskrävande än man kan tro. Att i efterhand tillföra mer pengar, när man inser behovet, är kanske inte lika problematiskt som att tillföra mer personal, under förutsättning att man även kan tänja på tidsramen. Svårigheten är ju emellertid att tidsramen definieras av leveransåtaganden och sannolikt även av nyckelpersoners inbokning i andra projekt.

Förutom dessa tunga resursbegränsningar kan man behöva ta hänsyn till tillgången på utrustning, fr.a. datorer och programvaror. De datorer man måste använda kanske inte har tillräcklig kapacitet för att klara en så detalje-

rad och beräkningstung modell som man först tänkt sig, och en uppgradering av maskinerna är ekonomiskt omöjlig. Man kan också ha programvaruverktyg, som kanske inte är allra bäst lämpade för projektet, men där marginalkostnaden att använda dem är liten jämfört med att skaffa nya programvarulicenser och lära upp personal på nya verktyg. Allt detta kan i någon mån påverka hur projektmålen utformas. Det kan f.ö. också vara så att beställaren förväntar sig att viss programvara eller vissa delmodeller skall användas i projektet, därför att han redan investerat stora summor i dem.

Har vi nu gjort klart för oss vilket problem projektet skall lösa och inom vilka ramar detta skall ske, så är det möjligt att precisera **målformuleringarna**. Det är då lämpligt att relatera dessa till de överordnade mål, som finns i de sammanhang där modellen, eller de nya kunskaper som kan utvinnas genom simuleringar med den, ska användas. Det är tämligen meningslöst att i ett projekt för försvarsmakten sätta upp som mål, att man med den nya modellen skall kunna värdera skogsavverkning på distans och risknivån för bärplockare i anslutning till detta. Troligen är det mera i linje med beställarens eget uppdrag att förhindra fientligt intrång i landet, om vi i stället sätter målet att kunna värdera artilleribekämpning av fientliga nästen i betäckt terräng. I stora drag kanske vi kommer fram till samma modell i båda fallen, men det kommer säkert att finnas detaljer som gör dem avsevärt mer lämpade för den ena eller den andra målsättningen.

Det är också viktigt att de mål man formulerar är tillräckligt konkreta för att vara ”mätbara”, annars vet varken vi eller våra beställare när vi är färdiga. Att bara säga att man skall göra en heltäckande modell av t.ex. ett sensorsystem är inte en god målformulering. Ordet ”heltäckande” måste relateras till något eller några användningsområden, t.ex. för jämförande utvärdering mot andra system av upplösningsförmågan i en viss typ av terräng, eller för att ge en pilot en realistisk upplevelse av informationsflödet i en flygsimulator. Den mätbara målformuleringen skulle i det förra fallet kunna vara, att simuleringar med modellen korrekt skall återge och förklara (den empiriskt kända) skillnaden mellan två eller flera existerande system. I det senare fallet skulle man kunna ha som mål att modellen skall klara ett Turing-test, d.v.s. att ett antal försökspersoner inte skall kunna skilja mellan verkliga sensorgenererade data och modellgenererade data.

Till målformuleringen hör också att fastlägga i vilken operationell miljö modellen skall användas. Det är olämpligt att utveckla en modell i Windows-miljö om man vet att den kommer att användas i Unix-miljö. Och skall den användas i båda miljöerna bör man ha också detta klart för sig redan från början för att inte införa besvärande låsningar till det ena eller andra systemet. Det kan även vara så att modellen skall användas i en miljö, som har olämpliga ljusförhållanden, är extremt bullrig eller som är allmänt stressig. Också detta kan ha en påtaglig betydelse för hur modellen, eller åtminstone dess gränsyta mot användaren, skall utformas.

Till driftmiljön bör man även lägga de eventuella säkerhetskrav, som kan finnas på modellen som sådan och på dess användning. Om modellen eller dess indata är sekretessbelagda eller om de förhållanden under vilka simuleringarna utförs är det, så bör detta framgå redan i målformuleringen. Därigenom kan man undvika att onödigtvis utveckla lösningar som exempelvis bygger på öppen nätverkskommunikation. Att i efterhand täppa till säkerhetsluckor blir ofta både besvärligt och kostsamt.

Har man nu klart för sig hur problemformuleringen ser ut, vilka yttre förutsättningar som gäller, hur målen skall formuleras, i vilken miljö modellen skall användas och vilka säkerhetsmässiga restriktioner som gäller, så är det ”bara” att sammanfatta detta i en projektplan med budget, olika delmål, leveransplan o.s.v. Planen bör vara så välavvägd, att den inte ger alltför strikt styrning långt in i framtiden, men samtidigt så strikt att man snabbt kan upptäcka om någon verksamhet är på väg åt galet håll. Detta kan åstadkommas med detaljerade delmål i närtid men mera svepande delmål i ett längre perspektiv. Allteftersom projektet framskrider kan delmålen preciseras och göras mer detaljrika.

Utveckla konceptuell modell

När man nu har klart för sig vad man vill uppnå i sitt projekt, är det dags att närma sig frågan hur man skall uppnå detta. Första steget blir att utveckla en konceptuell modell av den företeelse man vill kunna simulera. För det första är det viktigt att alla projektdeltagare har samma mentala modell av vad som skall åstadkommas och att man talar samma språk. För det andra behövs en formellt oantastlig beskrivning som en grund för framtida valideringsinsatser och för modelldokumentationen. En väl utarbetad konceptuell modell är dessutom det bästa hjälpmedlet för den som vill ta till sig och förstå någon annans modell.

Det första steget i denna del av arbetet består i att visa hur och i vilket sammanhang situationer uppstår, som man vill studera. Utifrån de mål man har satt upp för modellen utvecklar man lämpliga scenarier, som illustrerar de aktuella problemställningarna och de aktuella systemens funktion och beteende. Scenarierna behöver inte vara onödigt omfattande, men de bör tydligt och klart fånga in de situationer i vilka de aktuella problemen förväntas förekomma. Vi får härigenom en första uppfattning om vilka objekt, som spelar en viktig roll och hur många de är. Likaså illustreras vilka relationer, som finns mellan de olika objekten, vilka interaktioner som kan ske mellan dem, och vilka beteenden de i övrigt har. Ur denna beskrivning kan man sedan sluta sig till vilken detaljeringsgrad, som är lämplig för den kommande modellen.

Till scenariot hör också den miljö i vilken de olika objekten skall agera. Detta kan innefatta väderleksförhållanden, markbeskaffenhet, vattenströmmar, ljusförhållanden o.s.v. Även här får vi genom scenariot en indikation på lämplig abstraktionsnivå.

Slutligen beskriver scenariot förloppets start- och sluttillstånd, varigenom man kan

24

Ur Svenska Akademins ordbok (1965):

Scenario

BETYDELSE: i fråga om scenverk l. film: skriftlig plan som anger handlingens förlopp scen för scen o. de förhållanden varunder denna tänkes utspelad; dels om sådan plan avsedd att (utan ytterligare bearbetning av författaren) tjänstgöra ss. underlag för skådespelares improvisationer vid föreställning, t. ex. i det italienska maskspelet (commedia dell’arte), dels om sådan plan avsedd att kompletteras av författaren med dialog o. scenanvisningar, dels om sceneri

Ur Nationalencyklopedins ordbok (1996):

scena´rio subst. *~t, ~n* el. *scenarier*
• (film- eller teaterregissörs) samlade praktiska anvisningar för ett verks inspelning eller framförande: *filmscenario*; *tänka ut ett ~ till kortfilmen*; *göra ändringar i ~t*

BET.NYANS: överfört om (antagen) möjlig händelseutveckling på visst område: *framtidsscenario*; *krigsscenario*; *försvarsstabens ~ för en konflikt i Centraleuropa*

HIST.: sedan 1889; 1969 i bet. ’tänkbart händelseförlopp’

hämta andra viktiga begränsningar för den tilltänkta modellen.

Skall scenarioutvecklingen bli lyckosam fordras att man redan från början engagerar människor med god erfarenhet av och insikt i den verksamhet som skall modelleras. Sådana personer brukar ibland benämnas SME:er efter engelskans *Subject Matter Expert*. Är det en militär verksamhet, som skall modelleras, fordras alltså medverkan av militärt kunniga personer. Vanligtvis är det inte heller endast en enstaka person det gäller utan flera. Om problemområdet t.ex. innefattar duellen mellan luftvärn och flyg, bör man engagera officerare från både flygvapnet och luftvärnet. Kanske behövs det även personer, som är specialister på de olika tekniska systemen.

Det kan rent av finnas behov att engagera flera personer med ungefär samma kompetens, speciellt om problemområdet är ganska illa förstått. Man kan då få en allsidigare belysning fast till priset av att någon måste väga ihop kanske oförenliga synpunkter.

Att få till stånd en stor SME-grupp är dock i praktiken knappast möjligt. Dels lägger kostnadsramarna vissa hinder i vägen, dels är det i allmänhet ont om lämpliga och (samtidigt) disponibla experter. Dessutom är det av vikt att de tillfrågade SME-kandidaterna har ett påtagligt intresse för modellering och de approximationer, som måste göras, annars finns det risk för ett ganska ofruktsamt ”käbblande”. Å andra sidan bör den som tillfrågas om att ställa upp som SME se detta som ett hedersuppdrag och ett sätt att inte bara föra sina kunskaper och erfarenheter vidare på ett ganska trevligt sätt utan också fördjupa och systematisera dem.

Är det fråga om modeller av ”enklare” slag (mera renodlat hemmahörande inom en enda ämnesdisciplin), t.ex. en modell av en kemisk reaktion, kan det kanske räcka med de kunskaper, som modellutvecklaren själv besitter. Men självklart lönar det sig även i detta fall att rådfråga andra experter för att få en så allsidig belysning av ämnet som möjligt.

Har vi nu gjort vår läxa ordentligt, så har vi alltså nu en tydlig målformulering för vårt projekt och ett eller flera väl genomtänkta och förankrade scenarier. Utgångsläget för att utveckla en konceptuell modell, som utgör grunden till en funktionell och beteendemässig specifikation av simuleringsmiljön och de aktiva objekten, är då mycket lovande. Det gäller dock att inte förlora fokus, utan begränsa sig till att beskriva de objekt, interaktioner och beteenden, som verkligen upplevs som väsentliga utifrån målsättning och scenarier. Gör man inte det, utan börjar intressera sig för andra egenskaper hos det modellerade systemet, är det stor risk att man inte kommer att kunna uppfylla de ursprungliga önskemålen med modellbygget.

Det är alltså av vikt att man redan i detta skede av arbetet gör klart för sig vilka restriktioner som gäller, d.v.s. vilka situationer modellen inte skall användas för, och vilka objektegenskaper den således inte tar hänsyn till. Likaså bör man låta fantasin flöda, då det gäller att söka alla upptänkliga extremfall som kan uppstå inom problemdomänen och som modellen bör kunna hantera. Och anser man att den inte skall hantera ett visst extremfall, så inför man i stället en restriktion.

Låt oss för en stund återvända till frågan om vilken detaljeringsgrad modellen skall ha. De faktorer som framför allt styr valet är

- **Projektets mål.** Det finns ingen anledning att satsa på en mycket detaljerad beskrivning av ett flygplans yttre, om man främst är intresserad av ifall och i så fall hur skenmål kan användas för att reducera

hotet från IR-målsökande luftvärnsstridsdelar. Om modellen däremot skall användas för bedömning av hotet från radarstyrda vapen, behöver man en ytmodell som ger en trovärdig bild av kanter och andra detaljer som en radar lätt kan urskilja. Och skulle modellen användas i en simulatoranläggning för att åskådliggöra ett fientligt företag, då behöver ytbeskrivning och aerodynamiskt beteende vara så trovärdiga att de som bemannar simulatorn får en realistisk upplevelse av striden. Detta understryks av nästa punkt:

- **Trovärdigheten i beteendet.** Oavsett vilken modell vi arbetar med, så måste den upplevas som trovärdig, inte bara till objektens yttre egenskaper utan också hur de interagerar med varandra. Den behöver dock inte ha en mer detaljerad beskrivning än att den uppfyller detta kriterium. Om man kan generera en missilbana med hjälp av en ekvation, vars resultat ingen kan klaga på, så finns det ingen poäng med att detaljmodellera motordelarna i missilen. När det gäller bedömningen av vad som är trovärdigt, så är det fr.a. ett kriterium som väger tungt:
- **SME:ernas uppfattning.** Det finns dock en viss risk om man har flera SME:er inblandade, speciellt om de representerar olika ämnesområden. De kan då hamna i mycket olika uppfattningar beroende på att de har olika perspektiv. Det är mänskligt att övervärdera sitt eget expertområde på bekostnad av andra, som man kanske bara har ett allmänt hum om. Modellerarens uppgift är då att försöka avväga de olika uppfattningarna mot varandra och eventuellt försöka få de ivrigaste förespråkarna för hög detaljeringsgrad att modifiera sina åsikter och tänka mera i aggregerade termer. Men även när SME:erna har likartad expertis kan det hända att de drar olika slutsatser beroende på personliga ”käpphästar”.
- **Tillgängligheten av data.** Att exempelvis modellera en strid mellan egna och fientliga sjöstridskrafter, när man vet att ett eller ett par av de ingående objekten utgörs av sekretessbelagda främmande koncept, är ett äventyr i sig. Försöker man då göra en detaljerad modell, där dessa objekts prestanda måste parametersättas med stor noggrannhet, så är man illa ute. Endast ett lyckat spionage mot den främmande statens utvecklingscentra skulle då kunna rädda indatabeskrivningen. I annat fall är man hänvisad till mer eller mindre intelligenta gissningar, och simuleringsresultaten är då inte säkrare än gissningarna varit, snarare mindre än så. Det är då bättre att göra en mer översiktlig modell där de specifika (och i huvudsak okända) detaljerna ”bäddas in” i mer aggregerade objekt och egenskaper. Det är förmodligen enklare att då göra vettiga gissningar om parametrarna, vilket ger en större säkerhet i simuleringsresultaten.
- **Datorkapaciteten.** En mera prosaisk men icke desto mindre viktig aspekt beträffande detaljrikedomen är vilka prestanda man förväntar sig av den färdiga simuleringsmodellen. Vill man t.ex. åstadkomma en realistisk realtidssimulering måste man kanske dra ner på realismen i vissa avseenden för att inte målet med realtid skall drunkna i den flod av beräkningar som brukar bli följden av alltför många objekt och interaktioner. Ett exempel på detta kan vara en flygsimulator i domutförande,

där man kanske måste inskränka detaljrikedomen i de delar av synfältet, som piloten ändå inte har i fokus.

- **Tid och pengar.** En annan resurs, som brukar vara gränssättande mot önskemål om stor detaljrikedom, är personalens tillgänglighet och projektets ekonomi. Med oändliga resurser kan man naturligtvis komma godtyckligt nära en exakt kopia av den modellerade verkligheten, men som vi tidigare sagt, så är det knappast eftersträvansvärt. Däremot kan det med ändliga resurser uppstå besvärliga avvägningsproblem då det gäller att fördela projektets personal och pengar på modellering av diverse detaljer eller på andra väsentliga insatser, t.ex. verifiering och validering.

Vi bör även ägna frågan om tillgängligheten av data en ordentlig eftertanke. Förutom problemet om data över huvud taget finns tillgängliga eller kan utvinnas med en rimlig arbetsinsats, finns det anledning att i ett tidigt skede även reflektera över kvaliteten på de data man tror sig besitta. Är de egentligen intressanta, och om de är det, är de relevanta? Detta behandlas i kapitlet ”Dataförsörjning” (sid. 93).

Arbetssteget med konceptuell modellering leder slutligen fram till en specifikation av kraven på simuleringsmodellen. I detta läge skall man eftersträva att undvika detaljerade pekpinnar om hur modellen skall se ut, snarare beskriver man vilka uppgifter, som den och dess objekt skall ha. Därigenom undviker man att hamna i en situation, där modellutvecklarna kanske ser en effektiv och framkomlig modellutformning men som förhindras av en alltför rigid specifikation. Det är dessutom fördelaktigt att inte skriva allt på en gång utan arbeta etappvis mot allt större grad av detaljrikedom. Skriver man allt från början upptäcker man snart att man inte har förmått tänka på alla aspekter, vilket lätt kan leda till ett antal mindre lämpliga ad hoc-lösningar allteftersom arbetet fortskrider.

Designa simuleringsmodellen

Vi har nu en konceptuell modell, som alla inblandade anser sig vara överens om, och ett antal välformulerade modellkrav. Då är det dags att börja fundera över hur man skall lägga upp vår datoriserade modell, så att den skall kunna användas för de simuleringar, som behöver göras för att möta våra problemformuleringar.

Betraktar vi de krav som i specifikationen ställs på simuleringsmodellen, så ger det oss vägledning beträffande hur exakta våra objekt- och interaktionsrepresentationer behöver vara, vilken detaljeringsnivå som fordras, vad som är lämpliga atomära objekt. Detta ger oss även en indikation på hur vi bör utforma våra approximationer och algoritmer, beräkningsnoggrannheten behöver inte vara högre än att den matchar objektnoggrannheten (men naturligtvis inte heller lägre).

Likaså finns det ingen anledning att göra modellen mer komplicerad än nödvändigt, även om det kan vara lockande att bygga in ”vackra” och ”trendiga” strukturer i den. Enkelhet och transparens är honnörsord i sammanhanget. Det som är lättförståeligt för en icke specialist, som betraktar den slutliga strukturen, det är i allmänhet också det som i längden är mest hållbart.

Till enkelheten och transparensen hör att man tar tillvara de möjligheter som finns att dela in modellen i logiska delsystem. Det är även tillrådligt att tänka igenom vilka tillstånd som det modellerade systemet kan anta, när

det kan ske tillståndsförändringar, d.v.s. händelser, och låta det påverka strukturen. Annars är det lätt hänt att det uppstår relationer mellan vitt skilda programdelar, med otydlighet och svårighet att underhålla som följd. Liksom alla andra lärosatser skall emellertid inte heller denna betraktas som en ofelbar dogm. Drivs den in absurdum kommer den till slut att motverka sitt eget syfte därigenom att koden blir ”snuttifierad” eller formen blir viktigare än innehållet.

Komponentbaserad modellutveckling

Har vi nu skaffat oss en uppfattning om hur vi vill dela in vår modell i delmodeller kan det löna sig att se sig om ifall det redan finns färdiga modeller att tillgå. I så fall sparar vi mycket tid. Här har man stor nytta av om det finns väl administrerade bibliotek med moduler, med en standardiserad, väl genomförd och sökbar dokumentation. Om dessutom biblioteket är uppbyggt med tanke på att de ingående modellerna skall vara sammansättningsbara talar man om **komponentbaserad modellutveckling** (se faktaruta 25).

Om vi antar att vi hittar en modell, som skulle kunna passa som delmodell i vår nya simuleringsmodell, så är det emellertid inte bara att okritiskt och tacksamt inkorporera den. Även om den utger sig för att representera samma objekt och interaktioner som vi önskar, så måste vi noggrant pröva om den gör det i alla detaljer, som är viktiga för oss. Kom ihåg att en modell alltid är en avskalad avbildning av verkligheten, så kanske saknar den erbjudna modellen representation för någon för oss viktig objektegenskap!

En motsatt aspekt, som man också bör tänka på, är att välja delmodeller som inte är mer detaljerade än nödvändigt. En generellt användbar delmodell kan ibland bli mer av en belastning än en tillgång, t.ex. genom att den fordrar fler indata och räknar noggrannare än vad den aktuella problemlösningen egentligen kräver. Här fordras en avvägning mellan enkelheten i utvecklingen och effektiviteten vid användningen av modellen. Lösningen kan ibland vara en medelväg, att ta den generella delmodellen och modifiera (begränsa) den för det aktuella syftet.

25

Vad är komponentbaserad modellering?

”Utveckling av simuleringsmodeller genom återanvändning av existerande delmodeller är ett effektivt sätt att minska utvecklingskostnader, samt öka kvaliteten hos dessa simuleringsmodeller. Liknande tillvägagångssätt förekommer i tillverkningsindustrin, där färdiga komponenter används för att sätta samman nya produkter. Det är dock ingen trivial uppgift att utveckla modeller på ett komponentbaserat sätt.

Ett nyckelord inom komponentbaserad modellutveckling är *composability*. *Composability* är förmågan att sätta samman återanvändbara komponenter i olika kombinationer och utveckla sammansatta simuleringar som möter användarnas krav och behov. Det finns två typer av *composability*, syntaktisk och semantisk. Om implementeringsrelaterade detaljer såsom mekanismer för parameterutbyte och dataöverföring, och tidshantering för olika simuleringskomponenter är kompatibla råder syntaktisk *composability* mellan dessa. Semantisk *composability* handlar däremot om meningsfullheten av den sammansatta modellen. Genom semantisk *composability* kan giltigheten hos den sammansatta modellen förstärkas.”

För att komponentbaserad modellering skall vara genomförbar fordras därför ”en gemensam ontologi och informationsmodell, strukturerade metamodeller, ett semantiskt sökbart modellbibliotek, metoder för matchning av simuleringsmodeller och verktyg för ihopsättning av simuleringsmodeller.”

Ur [Eklöf et al., 2005]

Andra frågor att beakta, då man vill utnyttja en redan befintlig delmodell, är om modellen är tillgänglig i den utsträckning som vi behöver den. Kan vi komma åt källkoden för att göra eventuella modifieringar, eller har leverantören reserverat den möjligheten för sig själv? Kan vi över huvud taget integrera modellkoden, eller är den endast tillgänglig genom anrop i en distribuerad simulering? Hur samarbetsvillig är i så fall modellägaren? Vilken status har modellen avseende validering, verifiering och ackreditering, och hur väl är detta dokumenterat? Kan man vara säker på att modellkoden inte innehåller några säkerhetsrisker i form av trojanska hästar eller andra läckage- eller sabotagemöjligheter?

När vi så har delat in vår modell i delmodeller och kanske också hittat lämpliga moduler till en del av dem, så skall vi se till att få en bra och tydlig fördelning av funktionaliteten mellan dem. Det är t.ex. mindre lämpligt att siktberäkningar görs på flera olika ställen i modellen. Bättre är i allmänhet att lägga ansvaret för dessa på en bestämd delmodell. Det kan också lätt vålla kaos om det i ett fall är den delmodell som innehåller det skjutande objektet som beräknar om det blir träff eller inte, och i andra fall är delmodellen med målobjektet som gör det. Bestäm och dokumentera därför en strategi för hur denna fördelning skall ske redan på ett tidigt stadium.

Har vi nu valt en eller flera redan existerande modeller att ingå i vår nya modell, så är det med till visshet gränsande sannolikhet så, att de inte kommer att kunna integreras exakt i det utförande de har. Det är snarare så att det behöver ske vissa justeringar för att möta de krav som den specifika situationen i det aktuella projektet har, t.ex. beträffande ansvaret för olika funktioner, som vi just diskuterat. Se också diskussionen ovan om användning av generella men alltför detaljerade delmodeller. Resultatet är att vi på denna nivå i vårt projekt står med en lista med några delmodeller som behöver nyutvecklas och några andra som behöver genomgå större eller mindre anpassningsåtgärder.

Utifrån denna kunskap upprättas en implementeringsplan, som förutom tids- och resurstilldelning till de olika delmodellerna också innehåller val av utvecklingsmetod och -verktyg. Här kommer ofta objektorienterade metoder och UML-baserade verktyg väl till pass. Det finns även anledning att fundera över hur exekveringarna av modellen skall göras och i vad mån man är betjänt av att använda olika hjälpmedel för dessa, t.ex. databashanterare, simuleringsmotorer, visualiserare och dataloggare. Kanske är det rent av förnuftigt att göra utvecklingsarbetet och körningarna i ett simuleringsramverk (se sid. 81), eller med något annat hjälpmedel som t.ex. Mathlab.

Utveckla simuleringsmodellen

Det är nu dags att med hjälp av de valda utvecklingsverktygen omvandla modellen till programkod. Vilket programspråk man skall välja är en fråga som åtminstone förr kunde få närmast religiösa övertoner. Generellt kan man säga att det allra mesta går att uttrycka i de allra flesta programspråk men med något större eller något mindre möda. Den som har ett favoritspråk brukar hävda att det är effektivast att använda det, därför att vinsterna med att gå över till ett bättre anpassat språk inte uppvägs av kostnaderna för att lära in det nya och kanske ändå inte kunna utnyttja dess finesser optimalt. Det kan vara sant i fråga om små modeller, men när det gäller stora och sammansatta modeller bör man nog ändå tänka efter och investera i upplärningen i ett lämpligt språk. Mo-

derna språk har ofta konstruktioner som stöder ganska avancerade data- och programstrukturer, t.ex. är det enklare och tydligare att utveckla en objektorienterat specificerad modell i ett språk som stöder objektorienterade konstruktioner som att ärva och att gömma information.

Under programmeringen får man ytterligare ett tillfälle att fundera över noggrannheten i olika delmodeller och i de valda algoritmerna. När delsystemen verkligen kommer att påverka varandra, upptäcker man ofta nya ojämlikheter i beskrivningen av objekten och interaktionerna. Det kan vara harmlösa skillnader, men de kan också innebära att vissa beräkningar drar ut på tiden alldeles i onödan och snedbelastar hela simuleringen.

En annan viktig aspekt som förtjänar uppmärksamhet även under denna utvecklingsfas är konsistensen mellan de valda algoritmerna och de data, som skall nyttjas. Det finns åtskilliga klassiska fallgropar, som man annars lätt ramlar i. Tänk bara på vad som händer om man tilllämpar olika koordinatsystem. Att läsa av polära koordinater som om de vore kartesiska, ger definitivt fel läge. Eller vad händer om man inte observerar att en kraft anges i kilopond och inte i newton? Det behöver ju inte gå så illa som när en Marslandare misslyckades därför att konstruktörerna av styrmodellen inte observerat skillnaden mellan längder givna i fot och i meter, men simuleringsresultaten kommer i alla fall inte att bli särskilt trovärdiga. Och har man en delmodell som går på en annan dator med ett annat operativsystem, kan de data som genereras till resten av systemet misstolkas därför att talen representeras med olika ”byte”-ordningar i de olika systemen. Ett heltal i form av en ström av ettor och nollor tolkas då som ett helt annat tal i den mottagande datorn än i den sändande.

Likaså är det viktigt att sköta synkroniseringen mellan delmodeller som löper i parallella processer. För att det skall kunna ske måste förloppet styras av en gemensam ”klocka”, mot vilken alla delar avstäms. Detta ställer också krav på att de olika delarna initieras på ett sätt, som ger ett korrekt uppstartsförfarande.

Man kan också fundera över hur indata till simuleringarna skall organiseras. Skall varje delmodell ha separat indatainläsning eller skall den ske centralt? Bör man preprocessa vissa delar av data, så att beräkningarna i modellen kan göras snabbare (d.v.s. man lyfter ut en bit av modellen utanför den programmerade versionen)?

Till datahanteringsfrågorna hör även hur resultaten skall sparas. Även om simuleringarna huvudsakligen är tänkta för grafisk presentation är det ofta viktigt att kunna spara data som genereras under körningens gång för efterbearbetning eller för att kunna analysera varför resultatet blev så obegripligt som det blev. Även sådana mer administrativa aspekter bör planeras in i programmet från början.

Implementera och testa simuleringsmodellen

Det är nu dags att integrera alla delar av systemet till en gemensam simuleringsmodell. Har utvecklingsarbetet bedrivits helt och hållet inom ett och samma simuleringsramverk fordrar detta steg inte så stora extra insatser. Mer påtagligt blir det om man skall genomföra simuleringarna i en distribuerad miljö. Då kan sammanlänkningen av delmodellerna innebära ett mycket arbetsintensivt skede med samtidiga insatser vid flera datorer och kanske på flera orter. Det kan också finnas situationer då utvecklingsarbetet skett på vissa datorer men sedan skall modellen sättas samman på en annan

maskin. Då kan våra designansatser sättas på ett och annat hårt prov. Denna situation är ganska vanlig då simuleringarna använder hemliga data och därför endast får utföras i RÖS-skyddad miljö (RÖS = röjande strålning).

Det gäller nu också att trimma in de parametrar som påverkar exekveringshastighet och minnesutnyttjande, så att simuleringarna uppfyller de prestationskrav som uppställts. Många kommersiella programvaror har så många parametrar att ”skruva på” att problemet att hitta en för aktuell tillämpning optimal inställning ibland kan vara ett forskningsprojekt i sig. Det är också viktigt att man samlar alla sina erfarenheter och rekommendationer i en handhavandemanual i vilken framtida modellanvändare (inkl. utvecklarna själva) kan hämta ledning hur parametrarna skall sättas för nya tillämpningar. I en sådan manual bör också redovisas vilka hårdvarukrav och eventuella nätverkskrav som finns för att utnyttja modellen i framgångsrika simuleringar.

Då hela systemet av delmodeller skall testas är det naturligt att varje delmodell först testas för sig, något som kan förväntas ha skett redan då de utvecklades. Ingen programkonstruktör vill väl lämna ifrån sig något som han eller hon inte känner sig någorlunda säker på är riktigt. I sådana tester använder man en attrappmodell i stället för resten av den verkliga modellen. Dess enda syfte är att förse vår delmodell med ett simulerat indataflöde.

Nästa steg är att testa delmodellerna mot varandra parvis och sedan successivt i allt större grupper. Man undersöker då om dataflödet mellan dem är korrekt och om de sinsemellan uppfattar varandras data på ett riktigt sätt. Även här kommer attrapper till användning.

Slutligen testar man modellen i dess helhet med påhittade data och så småningom med riktiga data. Testverksamheten kan dock sägas vara en del av verifierings- och valideringsprocessen, vilken behandlas mer ingående i nästa kapitel.

Genomför simuleringar och analysera resultaten

När man nu äntligen har en simuleringsmodell, testad och godkänd enligt konstens alla regler, då är det dags att börja använda den till det den var avsedd till. Vi skall alltså utföra simuleringar, experiment med modellen, i ett bestämt syfte, t.ex. utbildning, träning, studier, o.s.v. En datoriserad modell kan vara så enkel att använda att man med lite sakkunskap i botten kan pröva sig fram ganska planlöst och därmed vinna insikt i den verklighet modellen beskriver. Men liksom vid alla typer av experiment får man ut mycket mera om man i förväg gör upp en noggrann plan.

En sådan plan kan exempelvis innehålla uppgifter om scenarier, vilka beteenden som särskilt skall beaktas och vilka variabler som skall studeras. Vidare kan man här beskriva vilka parameterintervall som är intressanta och hur variationerna inom intervallen skall göras (jämn fördelning, större tyngd kring ena ändpunkten, likformigt stokastiskt o.s.v.). Är det en stokastisk modell bör man också överväga hur många replikeringar (partier) som bör göras. Det kan anges som ett absolut tal eller t.ex. som ett krav på en största tillåten bredd på ett 95-procentigt konfidensintervall.

En annan sak, som man måste ta hänsyn till vid simuleringsplaneringen, är hur initialtillståndet påverkar utfallet. Om vi återgår till vårt biltvättsexempel kan vi tänka oss att vi vill studera hur väntetiden ser ut för de bilar som kommer under perioden 18:00 – 20:00 en vardagskväll. Om vi då startar vår simulering

klockan 18 med en tom kö, får vi säkert ett lägre värde på medelväntetiden än om vi utgår från att det redan står några bilar i kön. Ett sätt att undvika denna osäkerhet är att starta simuleringen redan från kl. 09:00 när biltvätten öppnar för dagen och köra simuleringen fram till 20:00, men endast samla statistik för den aktuella perioden. Man får då en s.k. **uppvärmningsperiod** (eng. *warm up period*), som omfattar 9 timmar. Även om denna teknik ger realistiska initialtillstånd för det studerade tidsintervallet, kan det ibland kännas som ett onödigt slöseri med beräkningskraft. Ett alternativ är då att anta (utifrån observationer av det verkliga systemet eller på annat sätt) att initialtillståndet har en viss statistisk fördelning, och slumpa fram en lämplig startkö ur denna fördelning.

Under exekveringen av en simuleringsmodell kan det behövas någon form av övervakning. På 50- och 60-talet kunde duktiga datoroperatörer se på lamporna på datorns kontrollpanel var i programmet den befann sig och dess aktuella tillstånd. Numera finns det mer sofistikerade övervakningsverktyg, som visar grafiskt på en skärm, eventuellt på en speciell övervakningskonsol vad som sker. Likaså kan man med vissa verktyg styra vilka data som ska samlas in för senare bearbetning. I många simuleringar är det dessutom önskvärt att från övervakarens sida kunna göra tillfälliga avbrott i exekveringen för att sätta om vissa parametrar och styra spelet i en önskvärd riktning. Vi behöver alltså brytbara simuleringar enligt definitionen i avsnittet ”Simulering” (sid. 12).

En simulering kan naturligtvis vara i sig en tillräcklig kunskapskälla, men dess värde ökar avsevärt om man efteråt kan reflektera över förloppet och resultatet. Man kan t.ex. identifiera vilka faktorer som har haft det största inflytandet, och vilka som kanske har haft ett försumbart inflytande. Det leder till en bättre förståelse av det studerade systemet och därigenom ger det underlag till modellförbättringar. Detta förutsätter dock att man har ett väl genomtänkt system för insamling av observationsdata i sitt program.

När man arbetar med stokastiska modeller eller med deterministiska modeller där man slumpat indata så får man ett stokastiskt utfall. Vad man då måste komma ihåg är att en genomkörning ger ett *möjligt* utfall bland en stor mängd. Hur representativt detta utfall är vet vi strängt taget inget om, men genom att köra en stor mängd upprepningar med varierat slumputfall så får vi en utdatafördelning som berättar mycket om modellens beteende, om tröskelvärden m.m. Även om man alltså avser att endast köra ett enda parti, därför att man använder modellen som simulator t.ex., så skapar det en avsevärt större förståelse om man dessutom gör en sådan s.k. Monte Carlosimulering.

Ett vanligt fall vid användning av modeller är att jämföra två alternativ med varandra, t.ex. en artillerigranat med en annan. Det är då onödigt att försöka finna de absoluta värdemåtten med hög noggrannhet, i allmänhet är det lättare att få hög konfidens på skillnaden mellan alternativen än att beräkna absolutvärdena var för sig och sedan beräkna skillnaden mellan dem. Det kan t.ex. vara fallet vid Monte Carlosimulering, där man utnyttjar samma slumptalsserie för båda alternativen.

Efter en genomförd simuleringsomgång bör man alltid ställa några frågor, som till råga på allt även bör besvaras så bra som möjligt. Den första och helt självklara frågan är om målen med simuleringarna har uppnåtts. Har vi vunnit den insikt som vi eftersträvade? Har vi

kunnat göra ett rimligt val mellan alternativen som var för handen? Har vi uppnått den träningseffekt som eftersträvades?

Den andra frågan vi bör ställa är kanske inte lika självklar: Det resultat vi uppnått, avspeglar det verkligheten eller är det en följd av modellens specifika egenskaper? Med andra ord, har vi lärt oss att det är behagligare att stryka kor framåt än bakåt därför att vi använt kottmodellen, som vi talade om i det inledande kapitlet?

Den tredje frågan har att göra med om de scenarier vi använt verkligen fångat det väsentliga i frågeställningen och om modellen har kunnat åskådliggöra de centrala aspekterna i scenarierna. Till denna fråga återkommer vi i ett senare kapitel. Här skall vi endast peka på att det är viktigt att vara uppmärksam på att små och till synes oväsentliga förändringar i initialtillstånden kan ge oväntat stora effekter på utfallet.

Ytterligare några frågor om modellens beteende: Hur var insvängningsförloppet till det erhållna resultatet, var konvergensen stabil, eller skedde det kraftiga pendlingar på vägen? Kan modellen vara sådan att den fastnar i stationära tillstånd och därmed har missat en del av det möjliga utfallet? Finns det tröskelvärden, små slumpstyrda variationer som ger stora skillnader i beteendet? För att ha verklig nytta av sin modell bör man vara så väl förtrogen med den och med den verklighet som modelleras, att man kan inse om och när sådana frågor är relevanta. Och om de bejakas bör man noga analysera i vad mån detta har betydelse för simuleringsprojektets resultat eller ej.

Dokumentera

Under hela modellerings- och simuleringsprocessen skall man dokumentera alla transformationer man gör och alla beslut man tar. Ledordet är **spårbarhet**, det ska vara möjligt att från den slutliga simuleringsmodellen kunna följa upp varför varje konstruktion är som den är. Detta är naturligtvis viktigt för att andra personer i framtiden skall kunna ta hand om och utnyttja modellen, men det är minst lika viktigt för dem som aktivt deltar i utvecklingsprojektet. Den som någon gång har försökt att gå tillbaka till ett datorprogram som man utvecklat för något år sedan för att inför ändringar eller för att förklara ett oväntat beteende, vet hur svårt det är att minnas alla detaljer. Då är en väl genomförd dokumentation guld värd.

En speciell svårighet är därvidlag att förstå varför äldre testkörningar ledde till de resultat som de gjorde. Programmet har med all sannolikhet utvecklats mycket sedan dess, så det är i princip omöjligt att återskapa den körning som gjordes den gången. En hjälp därvidlag är att använda sig av en versionshanterare, som sparar tidigare programversioner som använts i olika exekveringar.

Även när det gäller att hantera de textmassor som dokumentationen ger upphov till finns det många hjälpmedel i form av mallar och regelverk. Sådana system kan ibland verka avskräckande omfångsrika, och de kan också mera upplevas som ett hinder än en hjälp om de inte används med förnuft. Det övergripande målet är fortfarande att få en spårbarhet genom hela utvecklings- och användningsprocessen, sedan är formerna av underordnad betydelse.

Modellens användbarhet beror på dokumentationen. Utan god dokumentation kommer den inte att överleva det aktuella projektet. Arbetsbördan blir helt enkelt för stor för att man skall kunna ta till sig innehållet.

Då är det ju tur att det är så roligt att dokumentera! Det är ju först när man försöker berätta om det man gör och sätter det på pränt som man verkligen förstår det man gör.

Validering, verifiering och ackreditering av modeller

Att utveckla och använda en simuleringsmodell behöver inte vara särskilt komplicerat och inte heller särskilt svårt att förstå. De ballistiska modeller som på 50-talet utvecklades vid FOA och kördes på BESK-datorn och andra av den tidens maskiner var naturligtvis ganska komplicerade för sin tid, men de kunde greppas av varje människa, som gav sig tid att sätta sig in i dem.

Sådana modeller gör man också idag, men datorernas utveckling har möjliggjort så omfattande modeller att ingen enskild längre till fullo kan överblicka dem. Samtidigt medför utvecklingen att modellerna spelar en mer aktiv roll än de gjorde för 50 år sedan. Då resulterade simuleringarna i ett underlag som hjälpte människor att fatta beslut. Nu är det ofta så att simuleringsresultatet automatiskt medför ett beslut. Därmed har ett korrektivt filter försvunnit eller givits mindre tid att verka, en människa som skulle kunna upptäcka och rensa bort orimligheter. Vikten av att kunna lita på våra modeller är alltså större.

Vi står därför inför problemet att värdera våra M&S-produkter, även då vi i många fall inte förmår göra det ensamma utan måste förlita oss på insatser från ett större kollektiv. Vår modellerings- och simuleringsprocess behöver alltså en väl förstådd och genomförd delprocess med kontrollåtgärder för att vi skall kunna se till att de modeller som skall användas inom försvarsmakten är tillförlitliga och användbara för sina syften. Denna delprocess brukar betecknas som VV&A, d.v.s. validering, verifiering och ackreditering. Vad detta innebär framgår av följande definitioner:

Validering **är en process där man avgör om modellen är en riktig representation av den företeelse som modelleras i det sammanhang där den är tänkt att användas.**

Man kan också säga att valideringen besvarar frågan om det i Turings anda är omöjligt att skilja modellen från det verkliga systemet i den tilltänkta tillämpningen. Fångar alltså modellen systemets beteende i den utsträckning som behövs (men inte heller mer än så)?

Verifiering **är en process där man avgör om modellimplementationen överensstämmer med utvecklarens idébeskrivning och kravspecifikation.**

M.a.o. består verifieringen i att visa att en modell har en riktig representation och att trans-

formationerna mellan olika representationsformer under utvecklingen har varit korrekta.

Ett populärt sätt att uttrycka skillnaden mellan validering och verifiering är att säga att validering besvarar frågan: ”Bygger vi rätt modell?”, medan verifiering ger svar på: ”Bygger vi modellen rätt?”.

V&V kan inte garantera absolut korrekthet. Graden av kvarstående osäkerhet beror på den övertygande styrkan i den genomförda V&V-processen och den samlade beviskraften i de tillgängliga indicierna. Absolut säkerhet kan endast vinnas om att en modell är felaktig, men inte ens det är alltid möjligt.

Ackreditering **är ett officiellt bemyndigande att en modell eller simulering är acceptabel för en viss typ av användning.**

Naturligtvis förutsätter detta att det inom försvarsmakten finns någon organisatorisk enhet med befogenhet att utföra ackreditering eller som åtminstone kan peka ut och övervaka dem som skall göra det i varje enskilt fall. För övrigt vore nog **acceptans** ett bättre ord än ackreditering, då det senare snarast brukar förknippas med personer eller organisationer. Som vi ser finns här två olika typer av acceptans:

- Acceptans av modeller med hänsyn till ett specificerat användningsområde. Denna är dock till sin natur endast provisorisk, d.v.s. den måste omprövas allteftersom kunskapen växer och de verkliga systemen utvecklas.
- Acceptans av simuleringsresultat, som inkluderar en värdering av använda indata och de genomförda simuleringarnas omfattning och relevans.

Vad är det då som kan gå snett i ett modellerings- och simuleringsprojekt, och som VV&A skall kunna motverka? I stora drag kan man indela felen i tre typer:

- Simuleringsresultaten förkastas fastän de är tillräckligt trovärdiga. Denna situation brukar kallas **modellbyggarens risk** (eng. *Model Builder's Risk*), och innebär vanligtvis en onödig ökning av modellutvecklingsinsatserna.
- Ogiltiga simuleringsresultat accepteras som om de var tillräckligt trovärdiga. Här har vi i stället **modellanvändarens risk** (eng. *Model User's Risk*), vilken kan få katastrofala följder, om kritiska beslut skall baseras på resultaten.
- Simuleringsmodellen löser fel problem. Detta kan betraktas som en risk för både modellbyggaren och modellanvändaren, och den uppstår då förståelsen för det verkliga problemet inte är fullgod. Den leder till att simuleringsresultaten blir irrelevanta trots att genomförandet kanske har varit exemplariskt. Ett dåligt definierat problem eller en bristfällig förståelse för problemet är ofta den enskilt viktigaste orsaken till att många simuleringsprojekt inte ger trovärdiga resultat.

Fel kan som bekant uppstå i alla faser av utvecklingsarbetet och vid användningen av den färdiga modellen. Att upptäcka dessa fel och motverka dem så tidigt som möjligt i utvecklingskedjan, det är målet med VV&A. Ju tidigare ett misstag upptäcks, desto billigare är det att korrigera det.

En kort översikt över vad det handlar om: Naturligtvis är redan den konceptuella modelleringen en kritisk fas. Det gäller här att de teorier och antaganden som man bygger på är relevanta, att man har en tydlig och av alla förstådd

begreppsapparat, och att experternas kunskap tas till vara på ett sätt som inte förvränger informationen. När sedan simuleringsmodellen specificeras och designas gäller det att den konceptuella modellen transformeras på ett korrekt sätt, så att inte innehållet förvanskas. Och då man så småningom skall implementera modellen i ett programspråk så kan som bekant allehanda programmeringsfel inträffa.

När sedan modellen skall användas vid simulering, så gäller det att använda den på rätt sätt, så att scenarierna inte går utanför ramen av vad modellen är ämnad för. Likaså är det viktigt att se till att de indata man stoppar in i körningen är riktiga och relevanta för den specifika tilllämpningen. Och slutligen måste utfallet av simuleringen kunna tolkas i relation till modellens förutsättningar.

Alla dessa felkällor naggar naturligtvis modellens trovärdighet i kanten. Ordet **trovärdighet** används för resten för två ganska olika egenskaper hos en modell, vilka bör hållas isär. I det ena fallet avses med vilken noggrannhet (d.v.s. detaljeringsgrad) som modellen avbildar verkligheten. Denna egenskap kallas ibland ”**fidelitet**” (efter engelskans *fidelity*), och den bestäms i och med att den konceptuella modellen fastställs. Den andra aspekten är hur väl simuleringsmodellen realiserar intentionerna, sådana de fastställts i den konceptuella modellen. Är approximationer, algoritmer och antaganden lämpliga? Finns det programmeringsfel? Medan vi har ganska god kontroll över den förra aspekten (även om vi naturligtvis kan göra missbedömningar vid valet av deltaljeringsgrad), så är läget betydligt svårare då det gäller den senare.

Det är alltså fr.a. i det senare fallet som VV&A-insatserna är av avgörande betydelse. När vi valt detaljeringsnivå för en specifik tilllämpning ger de oss möjlighet att utifrån denna föresats uttala oss om modellens trovärdighet. En modell med låg detaljeringsnivå kan således ha hög trovärdighet för ett specifikt syfte. Visar det sig sedan att man gjort en missbedömning av lämplig detaljeringsgrad påverkar förstås även detta modellens trovärdighet.

Att VV&A-arbete behövs är det knappast någon tvekan om, men hur stora insatser behövs? Det är inte underligt om en projektledare med knappa ekonomiska ramar, eller en projektbeställare med kortsiktigt perspektiv kan låta denna del av projektet komma i strykklass i förhållande till mer ”produktiva” delar. Av någon anledning upplevs det inte lika glamoröst att kritiskt granska och kanske rent av döma ut programdelar som man har producerat jämfört med att skriva programkoden. Men i stället för att fråga vad VV&A-processen kostar borde man ställa sig frågan vad det kostar att låta bli.

Ett bra sätt att tackla denna frågeställning är att redan från början göra en regelrätt riskanalys av projektet. Det innebär följande:

- Identifiera möjliga riskhändelser och vad de kan leda till, d.v.s. alla de negativa konsekvenser som olika fel i modellutvecklingen och modellanvändningen kan få. Exempelvis kan vissa fel i en simulator leda till att ett olämpligt handlingsmönster lärs in av dem som utnyttjar den, vilket ökar risken för nederlag i en framtida strid. Eller kanske kommer man p.g.a. simuleringar, som modellen egentligen inte var konstruerad för, att satsa på ett sämre vapenalternativ än man egentligen borde ha gjort. Andra fel kanske inte är så allvarliga utan endast leder till irritation, något som dock kan ge simuleringsmodellen ett oförtjänt dåligt rykte.

Vad menas med risk?

”I dagligt tal förknippas risker vanligen med händelser som av en individ eller grupp uppfattas ha ett negativt värde. Ibland används riskbegreppet i betydelsen sannolikheten för att en negativ händelse kommer att inträffa. För att skilja dessa tolkningar åt används i detta dokument begreppen riskhändelse och risktal.

Med *riskhändelser* avses slumpmässiga eller osäkra händelser, vars konsekvenser är av negativt värde.

Med *risktal* avses sannolikheten för att en riskhändelse skall inträffa.

En *riskkälla* är en specificerad omständighet som leder fram till en specificerad oönskad händelse eller effekt.

Riskbärare är den individ eller grupp av individer som upplever risken.

Om den oönskade konsekvensen kan ha varierande storlek brukar ’risk’ ange den på något sätt sammanvägda sluteffekten.

...

Med riskhändelser för en simuleringsmodell avses de negativa konsekvenser som användning av en modell kan leda till på grund av brister och felaktigheter i modell eller data samt på grund av felaktig användning av modellen. Risker med själva modellutvecklingsprojektet i form av att projektet blir försenat eller inte håller sig inom tilldelade resurser tas inte upp här.”

Ur [Bergsten, 2002]

26

- Värdera och rangordna riskhändelserna.
- Avgör risktalen för varje händelse, d.v.s. hur troligt det är att felaktigheterna i modellen leder till felaktiga beslut, olämplig inlärning o.s.v. Det är ofta svårt att sätta några precisa värden på dessa risktal, men man bör åtminstone kunna värdera dem i storleksklasser, t.ex. låg, medium och hög.
- Bestäm vilka risktal som är oacceptabelt höga sett ur olika riskbärares perspektiv och baserat på syftet med modellen.
- Upprätta åtgärdslista: Avgör riskkällorna, d.v.s. möjliga orsaker, till de riskhändelser, som har för höga risktal, och rangordna dem efter svårigheten att åtgärda eller undvika dem.

Åtgärdslistan, liksom övrigt analysmaterial, ligger sedan till grund för inriktningen av V&V-insatserna. De viktigaste principerna är:

- Arbetet skall inriktas på att rätta uppkomna fel så tidigt som möjligt i M&S-processen, ju senare de upptäcks desto dyrare att rätta. Det är därför av central betydelse att lägga stor vikt vid V&V av den konceptuella modellen.
- Vissa delar av en simuleringsmodell kan vara av större vikt än andra. Dessa kritiska delar bör röna större uppmärksamhet än andra eftersom resurserna i allmänhet är begränsade.
- Vissa typer av fel kräver små resurser att upptäcka och åtgärda, varför de under alla omständigheter bör elimineras. Annars kommer de för all framtid att utgöra onödiga irritationsmoment vid modellanvändningen.

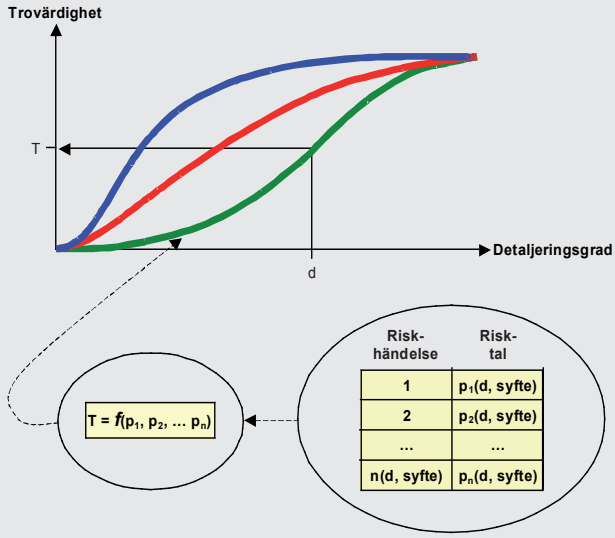
Riskanalysen bör dessutom fortleva under resten av projektet. Dels kan det finnas en tidsdimension med i bilden genom att projektdeltagarnas riskmedvetande ökar under arbetets gång, och dels kan en total riskanalys inte slutföras förrän modellen är färdigutvecklad.

Låt oss i detta sammanhang återvända till frågan om tillgängligheten av data och deras kvalitet. Vi vet att skall man utföra experiment, d.v.s. simulera, med en modell, så fordras det

27

Trovärdighet kontra risk

”Låt oss titta lite på bilden nedan för att koppla ihop trovärdighets- och riskaspekter. Vi antar att vi betraktar en speciell modell, t.ex. en modell av en bro.”



”Här har vi på x-axeln använt begreppet detaljeringsgrad och avser då ett mått på hur detaljerat modellen avbildar verkligheten. De olikfärgade kurvorna står för tre olika syften med modellen. Syftet med den övre kurvan kan t.ex. vara att använda bromodellen i en annan modell för att ge en verklighetstrogen bakgrundsmiljö (d.v.s. bron är sekundär i sammanhanget). Syftet med den undre kan vara att man vid konstruktionen av en ny bro vill använda en modell för att testa egenskaper hos bron vid olika former av belastning och påfrestning. I detta fall är bron huvudfrågan.

Om vi först avgör modellens detaljeringsgrad och därefter betraktar syfteskurvorna kommer vi att för den övre kunna få en god trovärdighet medan vi för den undre får en betydligt lägre trovärdighet för samma modell.”

Trovärdigheten kan här betraktas som en funktion av risktalen, vilka i sin tur är funktioner av modellens syfte och detaljeringsgrad. I fallet med den gröna kurvan kan vi tänka oss att en riskhändelse är att bron rasar vid extrem trafikmängd. Risktalet för detta är högt om detaljeringsgraden är låg, men blir successivt mindre ju högre detaljeringsgrad vi har. Modellens trovärdighet ökar i motsvarande grad som risktalet minskar.

Fritt efter [Bergsten, 2002]

indata. Naturligtvis behövs de även för validering och verifiering av modellen.

Om man nu har tillgång till alla de data som behövs, så är det givetvis gott och väl. Men det vanligaste är ändå att någon av följande situationer råder:

- Data är till största delen ej tillgängliga p.g.a. hemlighetsmakeri eller stora risker för människa och miljö att skaffa fram dem.
- Man har en viss mängd data, men de måste och kan kompletteras med fältstudier eller experiment.
- Det råder ett överflöd på data, ofta inte riktigt samstämda, vilket medför att det kan vara svårt att inse vad som är relevant information.

28

VV&A-processens steg

Definition av VV&A-krav
När väl ett beslut tagits huruvida ärvd modell, modifierad modell eller helt ny modell skall utnyttjas för aktuell tillämpning måste krav definieras mot vilka VV&A-arbetet skall kunna värderas.

Initiering av VV&A-planering
De planer som skall tas fram fokuserar på att identifiera VV&A-aktiviteter som passar ihop med M&S-plan, krav och resurser.

VV&A av problemkrav
Det är viktigt att validera de krav som ställts upp vid problemdefinitionen, så att simuleringen kommer att behandla rätt problem.

V&V av den konceptuella modellen
Detta är ett mycket viktigt steg och såväl den konceptuella modellen som V&V-arbetet måste dokumenteras. Dokumentationen visar varför (eller varför inte) antaganden, algoritmer, tillgängliga data o.s.v. kan förväntas vara en acceptabel representation av systemet för den tänkta applikationen.

Verifiering av designen
M&S-designen verifieras mot den konceptuella modellen.

Verifiering av implementationen
När väl implementeringen av designen har resulterat i kod skall denna kontrolleras för att minska risken för programmeringsfel.

V&V för hela tillämpningen
För att kontrollera att den kompletta modellen är tillräckligt noggrann för den aktuella tillämpningen jämförs den med känt eller förväntat uppträdande hos det system den skall representera.

Acceptansvärdering
Acceptansvärdering är det slutliga steget innan beslut kan tas om ackreditering av modellen för ett givet syfte. I detta steg granskas informationen som samlats upp under V&V-arbetet med modellen.

Ur [Bergsten, 2002]

I alla tre fallen är risken för felaktiga data påtaglig. Datafel kan orsakas av att man

- gissar och gör bedömningar utifrån analogier som inte håller i verkligheten, eller utnyttjar data från studier som egentligen inte är relevanta för tillämpningen,
- gör observationsfel eller registrerar något annat än man tror i en fältstudie,
- tillämpar en felaktig samplingsstrategi, t.ex. ett cykliskt förfarande som råkar överensstämma med en cyklisk egenskap i datamaterialet.

Om man nu råkar ut för felaktiga indata så kan det i bästa fall ge meningslösa simuleringsresultat, som gör det lätt att upptäcka att något är fel. Värre är det om felen resulterar i ett resultat, som inte är uppenbart orimligt, då kan det vara mycket svårt att upptäcka att det över huvud taget föreligger ett fel. Till råga på eländet kan det också hända, att de fel som introduceras på ett ställe i modellen ger upphov till märkbara avvikelser först i en helt annan del av simuleringen, och att då inse var felet ligger är långt ifrån enkelt. Det kan då vara fråga om ganska obetydliga fel, som så småningom fått en förstorad effekt under simuleringens gång.

Ett sätt att komma till rätta med dessa problem är att införa ett ackrediteringsförfarande även för data. Skillnaden mellan att ackreditera modeller och data är naturligtvis hårfin, eftersom data utan koppling till någon form av modell är obegripliga, och modeller utan data knappast kan utnyttjas för simuleringar. Ändå brukar man ofta använda en annan beteckning, nämligen **certifiering** av data.

Grunden för en rimlig certifieringsverksamhet är åtminstone trefaldig. För det första måste data förses med metadata, d.v.s. data om data, som alltså beskriver datas egenskaper, tillkomst-

historia etc. För det andra måste man så långt det är möjligt sträva efter att minska behovet av datatransformationer, vilka alltid utgör en risk för förvrängningar eller förluster i precisionen. Detta gör man genom att använda datastandarder med väletablerade transformationsprinciper. Slutligen måste man stödja sina anspråk på att ha data av rätt kvalitet genom tester och fältförsök. Det är därför riskabelt, om man inom försvarsmakten minskar utrymmet för sådan verksamhet.

Kärnfrågan för VV&A är att välja metoder baserade på vetenskaplig grund som på ett mångsidigt och korrekt sätt belyser modellens giltighet och utesluter så många fel som möjligt. För att nå upp till detta måste VV&A-processen löpa parallellt och integrerat med modellerings- och simuleringsprocessen. Det är således viktigt att redan vid projektstarten planlägga sina insatser avseende VV&A men det är minst lika viktigt att planen inte blir statisk, utan att den kan modifieras allteftersom nya insikter vinns medan projektet fortlöper.

Modellutvecklarna utför naturligtvis själva åtskilliga tester för att övertyga sig om att den produkt de arbetar fram blir riktig, och för mindre modeller som huvudsakligen skall användas i den egna forskningen är det vanligen till fyllest. Vid större och mer sammansatta modeller är det däremot fördelaktigt med från modellutvecklarna fristående VV&A-insatser. Det är i allmänhet svårt att upptäcka egna tanke- eller slarvfel, och då är det bra med en oberoende granskningsprocess. Denna process beskrivs i faktaruta 28.

Ackrediteringsbeslutet, som är resultatet av acceptansvärderingen, kan i princip få fyra olika utfall. För det första kan modellen få en fullständig ackreditering, d.v.s. simuleringarnas resultat är tillräckligt trovärdiga för att stödja tillämpningen. Om det inte är möjligt, kan de få en begränsad eller villkorlig ackreditering, d.v.s. man inför regler och begränsningar för hur simuleringarna kan och får stödja tillämpningen. En tredje nivå är att modellen troligen har förutsättningar att kunna användas i avsett syfte, men det inte är tillräckligt väl visat, varför man begär ytterligare V&V-insatser innan ackreditering kan ske. Det fjärde utfallet är naturligtvis att ingen ackreditering beviljas, därför att resultaten visar att simuleringarna inte är lämpliga för att stödja den aktuella tillämpningen.

Metoder och tekniker

De verifierings- och valideringstekniker vi har till vårt förfogande indelar vi i informella, statiska, dynamiska och formella. Användningen av matematik och logik ökar från informell till formell teknik och därmed också komplexiteten.

De informella teknikerna grundas fr.a. på sunt förnuft och subjektiva upplevelser. Bland de vanligaste är s.k. **skrivbordskontroll** (eng. *desk checking*), som inte är en utvärdering av arbetsplatsen, utan avser en granskning av de dokument (inkl. programkoden) som producerats i akt och mening att bedöma hur korrekt modellen är. Detta kan naturligtvis utföras av modellutvecklarna själva, men det brukar vara effektivare om någon fristående person får göra det. En annan vanlig metod är ”**utseendevalidering**” eller ”validering av synbarlig giltighet” (eng. *face validation*), som innebär att modellutvecklare, framtida användare och övriga inblandade experter gör en subjektiv bedömning av om modellens uppträdande verkar rimligt. Denna metod är främst lämpad för tidiga faser av utvecklingsarbetet, då man med prototyper vill visa hur den framtida modellen kan kom-

ma att te sig. En mer stringent form av denna metod är **Turing-testet**. Där låter man experter granska två uppsättningar data, den ena från en simulering med modellen, den andra från ett försök under samma förutsättningar med det verkliga systemet. Om experterna inte lyckas peka ut vilken uppsättning som kommer från simuleringen, eller inte kan förklara varför de väljer den ena framför den andra, så ökar modellens trovärdighet.

Statiska tekniker används fr.a. för att kontrollera modelldesign och källkod. För dem finns många maskinella hjälpmedel, varav programspråkets kompilator är ett exempel. Utmärkande för dessa tekniker är att modellen själv inte behöver exekveras i dator. Som exempel kan nämnas "**åskådliggörande av orsak och verkan**" (eng. *cause-effect graphing*), där man identifierar orsakssammanhang i det verkliga systemet och sedan granskar hur dessa representeras i modellen. En annan användbar metod är **gränssnittsanalys** (eng. *interface analysis*), då man försöker avgöra om struktur och beteende hos gränssnitten mellan olika delmodeller kan leda till fel vid informationsöverföringen. **Semantisk analys** för kontrollen av att specifikationerna blivit rätt tolkade vid programkodningen och **syntaktisk analys** för kontroll av att programspråkets mekanismer utnyttjas rätt är ganska självklara tekniker för alla programutvecklare. **Spårbarhetsutvärdering** (eng. *traceability assessment*), används för att spåra vilken motsvarighet t.ex. ett element i kravspecifikationen har i modellen under de olika utvecklingsstegen. Element som saknar motsvarighet kommer på detta sätt att avslöja icke uppfyllda krav.

Dynamiska tekniker används då man validerar modellens uppträdande under exekvering, och de har generellt tre steg:

1. Simuleringsmodellen instrumenteras, d.v.s. man inför ytterligare kod i modellen för att samla in information om förloppet under exekveringen.
2. Den instrumenterade modellen exekveras.
3. Utdata från modellen analyseras och det dynamiska uppförandet utvärderas.

Några vanliga metoder och tekniker för dynamisk V&V bör omnämnas. Först har vi **avlusning** (eng. *debugging*), som vi känner igen som det vanliga sättet att följa beräkningsgången i ett program i avsikt att finna felaktigheter. Ett mera provokativt sätt är **testning med infört fel** (eng. *fault insertion testing*). Med denna teknik för man medvetet in en felaktighet i modellen och observerar hur och när simuleringen spårar ur i ett felaktigt uppträdande. En tredje teknik är **känslighetsanalys** (eng. *sensitivity analysis*), med vars hjälp man identifierar vilka indatavariabler som har det största genomslaget på utdata. Därigenom kan man koncentrera sina vidare V&V-insatser på de faktorer som är mest känsliga. Även **statistiska metoder** som t.ex. hypotesprövning är användbara för att göra troligt att modellen är korrekt. En teknik som **objektflödestestning** (eng. *object flow testing*) ger en möjlighet att under exekveringen följa de modellerade objektens livscykler, vanligtvis i grafisk form. Detta ger ofta en god inblick i att modellen beter sig som förväntat.

En dynamisk metod som kan vara av stort värde är **jämförande testning** (eng. *comparison testing*). Den förutsätter att det finns åtminstone en annan modell, som behandlar centrala delar av vad vår nya modell täcker in. Kan man

då finna scenarier som kan simuleras i båda modellerna, kan de ge underlag till en jämförelse mellan modellernas beteende och resultat, som kan ge en djupare förståelse för resp. modells egenskaper och eventuella brister.

Ser vi slutligen på de formella metoderna, så är de baserade på matematisk och logisk formalism. I regel fordras att modellspecifikationerna skrivs i ett formellt språk för att man sedan skall kunna med exempelvis logisk deduktion göra utsagor om simuleringsmodellen följer av specifikationerna och därmed är korrekt. Vi har alltså här en metod för **formell korrekthetsprövning** (eng. *proof of correctness*). Den i grunden enkla principen för dessa logiska slutledningar är det som kallas ”modus ponens”, d.v.s. om vi vet att påståendet A är sant och att påståendet B följer av A, så måste även B vara sant. I de flesta praktiska fall blir emellertid den totala komplexiteten mycket stor, varför dessa i och för sig mycket kraftfulla verifieringsinstrument blir teoretiskt krävande och därför ganska tungarbetade. I allmänhet kan man därför inte använda metoden på stora modeller, däremot är den ofta lämplig att tillämpa på säkerhetskritiska mindre delmodeller.

Ansvarsroller

I ett stort M&S-projekt framträder mer eller mindre tydligt ett antal arbetsuppgifter. Ansvaret för deras genomförande bör man explicit lägga på utpekade personer. Fördelningen av dessa roller följer i allmänhet ingen strikt mall och kan se olika ut beroende på projektets syfte. För mindre modellutvecklingsprojekt eller projekt med lägre krav på garantier beträffande korrektheten hamnar ofta flera olika roller hos en och samma person. Även om det utåt sett då inte råder en klar bild av hur fördelningen ser ut, så är det viktigt att alla inblandade själva har klart för sig när de går in i och ut ur de olika rollerna. Här ges en kort beskrivning av de tydligaste rollerna.

Kunden kallar vi den som äger problemet och som har ett syfte med att det utvecklas en modell. Denna rollfigur lägger ut ett uppdrag till en *problemanalytiker*. Om uppdraget explicit avser utveckling av en ny modell, så utser kunden också en *ackrediteringsagent* samt fattar det slutliga ackrediteringsbeslutet på grundval av agentens rekommendation.

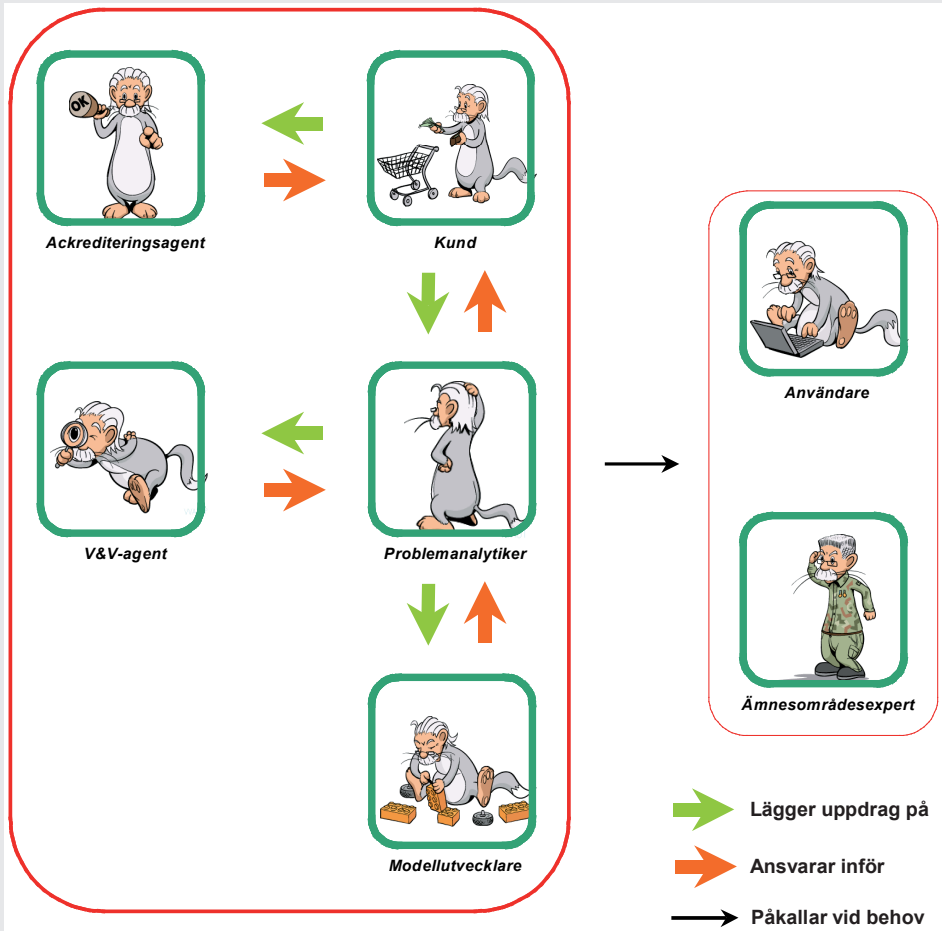
Den organisation som får kundens beställning utser en projektledare, som ikläder sig rollen som **problemanalytiker**. Problemanalytikern ansvarar inför kunden och har till uppgift att se till att det givna problemet löses på ett tillfredsställande sätt i tid och inom ramen för överenskommen budget. Detta innefattar att ta fram en kravspecifikation för simuleringsmodellen, utreda om M&S över huvud taget är ett lämpligt lösningsalternativ, och i så fall vilken modelleringsteknik som bör komma i fråga samt lägga ett uppdrag på en *modellutvecklare*. Han eller hon utser även en *V&V-agent*. Om kundens uppdrag är att få en fråga besvarad och problemanalytikern väljer att använda sig av M&S för att kunna besvara frågan, så är det denne och inte kunden som också utser *ackrediteringsagenten* samt fattar ackrediteringsbeslutet.

Den organisation som får uppdraget att utveckla simuleringsmodellen tillsätter en projektledare, som då ikläder sig rollen som **modellutvecklare** (sannolikt tillsammans med ett antal medarbetare). Denna person ansvarar gentemot problemanalytikern för leveransen av en modell i enlighet med givna krav och ramar. Modellutvecklaren skall även ge *V&V-agenten* erforderligt underlag.

29

Rollbesättningen i VV&A-spelet

I denna schematiska framställning förutsätts att kunden har beställt en modell, d.v.s. han eller hon är *intressent* i modellen.



Teckningar: Martin Ek

V&V-agent är den eller de som har ansvaret för validerings- och verifieringsarbetet. De tillsätts av problemanalytikern och ansvarar inför denne för att validerings- och verifieringsåtgärderna är relevanta och tillräckliga med avseende på överenskommen nivå. Agenten lämnar en V&V-rapport som underlag för *ackrediteringsagentens* värdering.

Ackrediteringsagenten utses av kunden eller problemanalytikern och fungerar som dennes advokat för att försäkra att kraven för acceptans är uppfyllda. Agenten ansvarar inledningsvis för att definiera de kriterier som utgör utgångspunkten för värderingsarbetet, utför det utifrån V&V-agentens underlag samt utfärdar slutligen en ackrediteringsrapport.

Ytterligare två rollfigurer har betydelse för VV&A-processen, men kallas snarare in vid behov än tilldelas ett specifikt ansvar. Det är **användaren**, som så småningom skall utnyttja den färdiga modellen, och **ämnesområdesexperten** (SME:n, se avsnittet ”Utveckla konceptuell modell”, sid. 55), som besitter sakkunskapen om den modellerade verksamheten eller delar därav.

Sammanfattning av de centrala lärdomarna om VV&A

De kriterier vi har för en god tillförlitlighet hos en modell är att vi

- tillämpat en god M&S-process,
- gjort en kvalificerad riskanalys,
- har en väl genomförd VV&A-process,
- har gjort en noggrann dokumentation,
- har skapat en hög kompetens för att använda modellen.

Konsekvenserna av en bristfällig VV&A-process är fr.a. att

- användning av modellen resulterar i felaktiga beslut,
- onödiga kontroverser om modellens korrekthet blossar upp,
- modellen inte används och våra satsade resurser är bortkastade.

De viktigaste principerna i VV&A-verksamheten (enligt [Bergsten et al., 2001]):

- Det är inte möjligt att garantera fullständig säkerhet för en modell. En modell utvecklas med avseende på ett specifikt syfte och dess trovärdighet skall bedömas utifrån detta syfte. Att en modell är ackrediterad för en tillämpning betyder inte att det är fritt fram att använda modellen i andra sammanhang.
- VV&A-processen måste pågå under modellens hela livscykel. Det är inte något man genomför efter en modellutveckling, eller en gång för alla. VV&A-processen startar samtidigt med M&S-processen, och om modellen senare modifieras skall VV&A-processen åter sättas igång.
- Man kan endast uttala sig om en modells trovärdighet för de villkor under vilka den testats. Exempelvis kan trafikmodeller som testats med avseende på morgontrafiken uppvisa helt felaktiga resultat beträffande kvällstrafiken.
- Att de ingående delmodellerna i en modell bedömts tillräckligt trovärdiga var och en för sig, är inte tillräckligt för att modellen själv ska vara tillräckligt trovärdig. Hela modellen måste vara ordentligt testad.
- En lyckad VV&A-process kräver data som har verifierats, validerats och certifierats.
- VV&A-aktiviteterna måste noggrant planeras och dokumenteras. Dokumentationen är viktig både för att den ger kunskap om modellens styrka och brister för kommande användare, och för att underlätta kommande VV&A-processer vid modifieringar av modellen.

Fallgropar vid användning av simuleringsmodeller

När man nu har en efter konstens alla regler verifierad, validerad och ackrediterad modell och dessutom god tillgång till certifierade data, då är det väl bara att sätta igång och simulera?

Nej, tyvärr finns det ingen garanti för ett godtagbart resultat (d.v.s. ett svar på den ställda frågan) ens om man har en ackrediterad modell, många olika scenarier och en klar och entydig frågeställning. I faktarutorna 31 och 32 ges ett par exempel på bekymmer, som uppstått vid användningen av den väl validerade och verifierade luftvärnsmodellen SILVIA.

Det som kan vara svårt att komma ihåg är att de slutsatser man drar från en simulering i hög grad beror på vilka scenarier man har valt. Det är lätt gjort att man vid resultatanalysen lägger allt fokus på de situationer som simulerats men glömmer bort de situationer som man inte simulerat därför att de kanske var alldeles för självklara. Vi vill ju t.ex. gärna att det ska hända något dramatiskt i simuleringarna och glömmer gärna bort att i verkliga livet är det allra vanligast att ingenting alls händer. Ett exempel är luftvärnet, som genom sin blotta existens kan avskräcka från ett tänkt luftangrepp och således i stort sett förbli på en låg verksamhetsnivå.

Frågan är då hur långt man kan dra slutsatser om tekniska detaljer utifrån simuleringar av så osannolika händelser. Det kan bli helt vilseledande. I fallet med luftvärn tenderar exempelvis stridssimuleringar att överdriva betydelsen av skjutkapacitetsparametrar. I stället borde man modellera hur fienden modifierar sin taktik utifrån det aktuella hotet. Speciellt i en många-mot-många-modell måste man tänka sig in i situationen från fiendens sida. Då kanske effektmåttet inte i första hand är hur många fientliga vapenplattformar som försätts ur stridbart skick, utan snarare hur länge ett system förmår att *överleva som ett hot*.

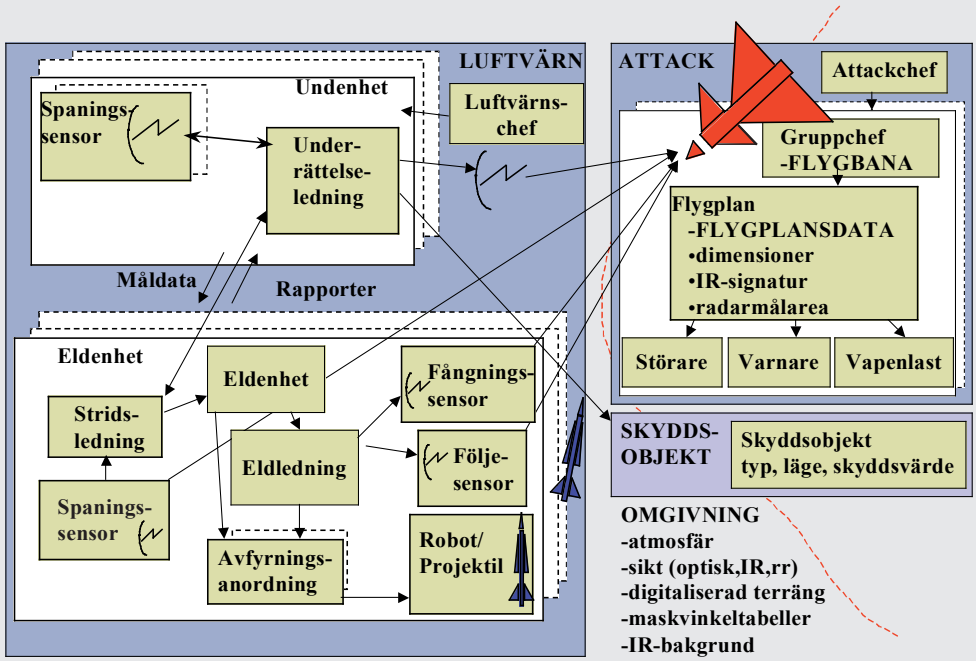
Analysarbetet begränsas alltså inte primärt av modellens detaljeringsnivå eller av datorkraften, utan av analytikerns förmåga att välja scenarier och effektmått som återspeglar det verkliga aktuella problemet.

Ser vi på våra luftvärnsexempel, så kan vi konstatera att bekämpningssekvensen är den faktor som påverkar utfallet mest. Den påverkas i sin tur av såväl små som stora förändringar i såväl tekniska som taktiska faktorer. En förändring av bekämpningssekvensen i ett tidigt skede kan medföra stora förändringar i utfallet i ett senare skede. En till synes obetydlig på-

30

LV-modellen SILVIA

I [Hansson 1997] ges några tydliga exempel på vad som kan inträffa om man inte rätt förstår hur scenario och modell samverkar. För att kunna inse situationen måste vi först ge några karaktäristika om den använda modellen. Den är deterministisk och händelsestyrd. De ingående objekten, aggregerade och atomära, samt interaktionerna mellan dem kan åskådliggöras med följande figur.



verkan kan därför få stora konsekvenser. Vid en systemjämförelse kan således en liten scenarioändring medföra en jämförelse med två helt olika förutsättningar, eftersom man efter första sekvensändringen inte längre jämför med samma förlopp. Fallgropsexemplet 1 visar just hur enkelt bekämpningssekvensen kan påverkas.

En studie med en simuleringsmodell fordrar djupa kunskaper inom området. Det är därför tillrådligt att börja i liten skala med enkla scenarier, enkla modeller och problemställningar. Kunskapen som man då får om systembeteendet är till mycket stor hjälp då man sedan analyserar simuleringarna i en större skala. Analysen bör dessutom alltid ske gemensamt med problemägaren och modellexperten för att i möjligaste mån motverka att man hamnar på ett sidospår.

Även om stora modeller kan ge möjlighet till att spela upp stora scenarier, så gör även de sig bäst med små scenarier. I ett scenario med många aktörer blir det lätt alltför många samverkande parametrar för att det ska vara lätt att analysera resultatet.

Det är när man använder komplexa modeller dessutom lätt att se just det som man vill eller förväntar sig att se och att förklara underliga resultat på fel sätt. Ett visst fenomen kan bero på modellen, på scenariot, på det verkliga systemet eller på en kombination av dessa. Är

Fallgropsexempel 1

Med ett antal simuleringar med SILVIA ville man bl.a. studera ”om det var bättre att ha 6 robotar i stället för 4 på en eldenhet ...

Med de körningar vi gjorde kom vi fram till att det spelade ingen roll att höja antalet robotar till 6 i stället för 4. Resultatet berodde på att de scenarier som kördes hade en angripare vars separationstid mellan grupper var väl anpassad till omladdningstiden (eller snarare tvärtom). Vi fick då ett fenomen att om man hade 2 extra robotar så hamnade omladdningen efter 3 rotar, det vill säga mitt i en grupp. Eftersom tiden mellan rotar var kortare hann man inte ladda om utan missade resten av gruppen under omladdningen. När omladdningen hamnade mellan grupperna på grund av att man bara hade 4 robotar hann man ladda om inför nästa grupp. Svaret blev att med en sån angripare så räckte det med 4 robotar. Det var till och med en nackdel att ha 6 robotar.

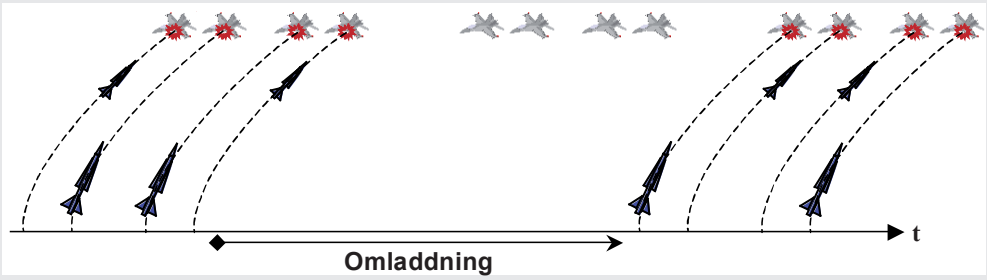
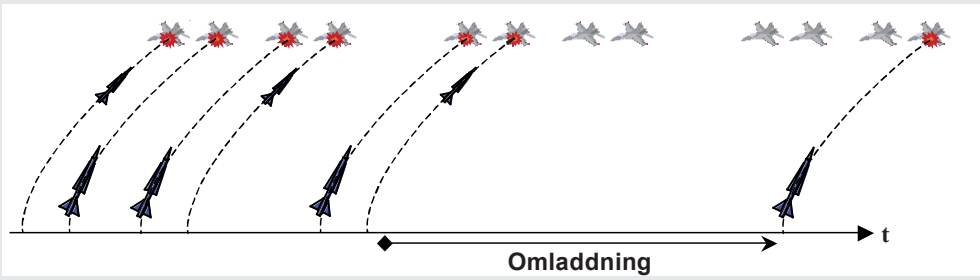


Fig. 1: Fyra robotar per eldenhet. 8 av 12 flygplan skjuts ner.



Figur 2: Sex robotar per eldenhet. 7 av 12 flygplan skjuts ner.

Det är viktigt att notera att detta resultat fick vi beroende på att omladdningstiden passade så väl med den antagna tiden mellan grupper i angriparscenariot. För en angripare är det lätt att anpassa sig efter systemet och att i stället komma in och mätta det till exempel genom att i detta fall komma in med fler rotar per grupp eller att välja en annan separationstid mellan grupperna. En annan viktig poäng är att resultatet beror på att SILVIA bekämpade allt och att modellen är gjord så att omladdning inte kan göras innan robotarna på lavetten är slut.

Exemplet visar både att modellkännedom är viktig och att scenariot har stor betydelse för resultatet. En lärdom är att vara försiktig med att dra slutsatser av bara några scenarier om man inte tar hänsyn till att en angripare lätt kan ändra sitt beteende.”

[Hansson 1997]

det därigenom svårbegripligt, kan det vara lätt att låta bli att tränga in i den verkliga orsaken och avfärda resultatet med att ”det är fel på modellen”.

Sammanfattningsvis säger alltså erfarenheten att det fordras flera scenarier i varje värdering eller studie, och att varje scenario måste varieras med små parameterskillnader. Det senare

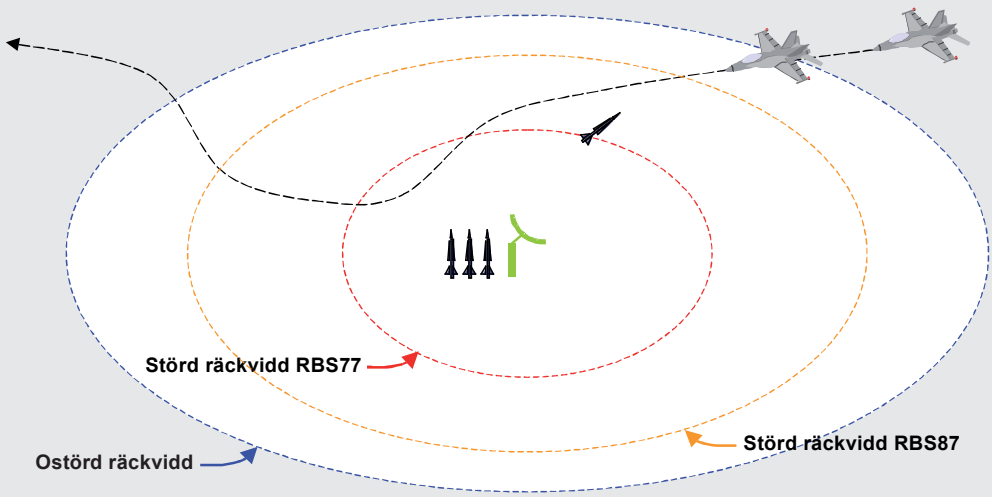
32

Fallgropsexempel 2

Med ett antal simuleringar med SILVIA ville man jämföra två varianter av Hawk-robotsystemet, det ena med den vanliga spaningsradarn, RBS77, det andra med en mycket bättre och mindre störkänslig radar, RBS87. Flera scenarier med varierat angriparbeteende och såväl med som utan fientlig störning simulerades. Resultatet i form av andelen nedskjutna flygplan blev

	Utan störning	Med störning
RBS77	59 %	53 %
RBS87	58 %	42 %

Slutsatsen blev således att det äldre systemet var bättre än det nya, speciellt i det störda fallet. Någonting är uppenbarligen galet, men vad? Det visade sig att resultatet berodde på att luftvärnet i de använda scenarierna grupperats ganska nära skyddsobjektet och de flesta anfallande flygplanen flög nära eldenheten. I de fall då man inte hade störning, sköt luftvärnet i allmänhet på långt avstånd, eftersom planen upptäcktes tidigt. Den långa skottiden innebar att vissa flygplan hann försvinna bakom terrängmasken innan roboten nådde fram. Med störning, vilken i modellen reducerar upptäcktsavståndet, minskade denna möjlighet påtagligt. Dessutom blev eldhastigheten större eftersom robotarnas flygtid blev kortare.



Den korrekta slutsatsen av simuleringen var alltså, att man inte alltid skall skjuta då målet upptäcks på maximalt avstånd utan vänta en stund. Men detta ger ju absolut ingen vägledning då det gäller att besvara den ursprungliga frågan.

[Hansson 1997]

kan man t.ex. åstadkomma med Monte Carlo-körningar (om modellen är stokastisk) på varje scenario, varigenom man kan få en god uppfattning om utfallsrummets omfattning och oregelbundenheter.

Modellering och simulering är ett utomordentligt kraftfullt verktyg för att skapa förståelse, men *bara om det används med förstånd.*

Ramverktyg för modellering och simulering

Allteftersom datorerna har blivit kraftfullare har nya och områden kunnat mutas in för modellering och simulering. Vi kan därför numera utveckla simuleringsmodeller som täcker ett enormt spektrum av såväl komplexitet som tillämpningsområden.

Problemet är dock att vi inte har förmåga att utnyttja hela denna väldiga potential, därför att det också kräver väldiga modellutvecklingsinsatser av oss.

En del av detta problem härrör från det faktum att vi varje gång, som vi utvecklar en större modell, lägger ner en hel del tid på att konstruera delar, som inte är tillämpningsrelaterade utan snarare kan betraktas som allmängods som i större eller mindre utsträckning förekommer i alla simuleringar. Det förefaller ju då vara förnuftigt att utveckla dessa gemensamma komponenter en gång för alla och så generellt användbara att de enkelt skall kunna utnyttjas för varje ny tillämpning. Ännu bättre vore det naturligtvis om man kombinerade detta med verktyg som underlättade själva modellerandet och hopkopplandet av olika delmodeller, liksom hjälpmedel för att konstruera lämpliga scenarier för simuleringarna. Lyckas vi åstadkomma detta, har vi något som brukar kallas för ett **simuleringsramverk**.

Flera sådana ramverk har utvecklats på skilda håll. I allmänhet är de inte helt generella, för de som utvecklar dem har i regel fokus på ett begränsat antal tillämningsområden. I gengäld kan man säga att inom de områden ramverket är tänkt, så bidrar det mycket positivt till de aktuella projekten med den mångåriga kunskap och erfarenhet som finns inbyggt i det.

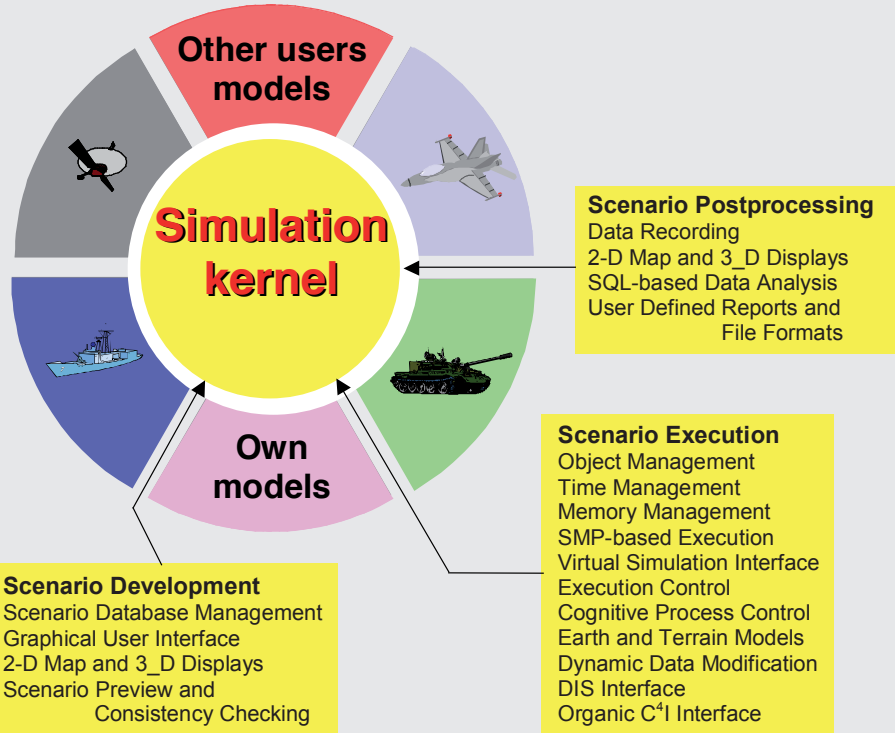
Ett väldesignat, välfungerande simuleringsramverk tillsammans med ett ackrediterat modellbibliotek skulle alltså kunna erbjuda en stabil plattform för att snabbt och effektivt prova idéer, generera beslutsunderlag, se resultat eller snabbt skaffa sig en uppfattning om en viss situation. Ledtider för studier och modellutveckling blir avsevärt bättre än med traditionell metodik och återanvändbarheten underlättas.

En definition skulle alltså kunna formuleras på följande sätt:

Ett *simuleringsramverk* är en infrastruktur som tillhandahåller tjänster för modellutveckling, scenariouppbyggnad, exekvering, analys och visualisering.

33

FLAMES – ett exempel på simuleringsramverk



FLAMES (*Flexible Analysis Modeling and Exercise System*) är ett simuleringsramverk utvecklat av Ternion Inc. i USA och levereras med en uppsättning exempelmodeller inom många olika tillämpningsområden (här illustrerat med de fyra domänerna mark, sjö, luft och rymd). Det har använts i Sverige sedan 1997 för såväl luft- som markstridssimuleringar.

Som exempel på vilka tjänster som bör förekomma kan nämnas tidsstyrning, databashantering, minnesallokering, schemaläggning, terrängrepresentation, dataloggning, återuppspelning och grafik för resultatpresentation. Vissa ramverk stödjer även komponentbaserad modelleringsmetodik, d.v.s. att större modeller byggs upp av mindre, som kommunicerar med varandra.

På denna modelloberoende kärna finns ett modellbibliotek, som stöder en eller flera tillämpningsdomäner. Idealt sett så bör det innehålla modeller på flera olika abstraktionsnivåer och med flera olka grader av verklighetstrohet (fidelitet). I allmänhet förser ramverkskonstruktören redan från början sin produkt med ett enkelt bibliotek av detta slag. Med tiden växer det sedan till genom bidrag såväl från den egna projektgruppen som från andra användare.

Ett bra ramverk levereras också med standardapplikationer för scenariogenerering,

exekvering, efterbearbetning och visualisering. Om inte annat så kommer man själv med tiden att få en sådan uppsättning med egenutvecklade applikationer, vilka kan tjäna som utgångspunkt för nya tillämpningar. Dessutom bör ramverket innehålla en god utvecklingsmiljö för den som skall utveckla egna delmodeller eller anpassa någon redan befintlig.

Med ett bra simuleringsramverk bör man således direkt efter anskaffningen (och upplärningen) komma igång med modellutvecklingen, som kan koncentreras till de områden som är av primärt intresse för den aktuella tillämpningen. För den som är någorlunda förtrogen med verktyget, så ger det därför goda möjligheter att snabbt och effektivt nå fram till önskat resultat. En öppen fråga i sammanhanget är dock om det är möjligt och framför allt eftersträvansvärt att ha ett enda generellt simuleringsramverk för försvaret och i så fall i vilken utsträckning det skall användas.

Fördelarna med simuleringsramverk kan sammanfattas i följande punkter:

- De ger möjlighet till snabb scenario- och modellframtagning.
- De är tids- och kostnadsbesparande i jämförelse med egen programutveckling.
- De kan underlätta samverkan mellan olika tillämpningsområden.
- Det är enkelt att återanvända gamla modeller, men bara inom det egna ramverket.
- Programvaran är i allmänhet relativt väl avlusad.
- Merparten av kostnaderna för underhåll och förvaltning drabbar tillverkaren av ramverket och delas därigenom av alla kunder.

Modellarkitektur och distribuerad simulering

Antag nu att vi skulle vilja skapa en modell av ett ganska komplext system med många olika komponenter av väsentligt olika slag (t.ex. människor, fordon, vapensystem, kommunikationssystem, ledningssystem, sensorsystem och terrängformationer).

Därvid vill man utnyttja redan befintliga modeller av de olika delsystemen och av deras interaktiva beteende, modeller som framtagits av de bästa experterna inom respektive område. Det vore ju då ganska bra om vi kunde plocka ihop alla dessa delmodeller i ett simuleringsramverk för att utnyttja alla dess fördelar. Kanske detta skulle kunna ske i några lyckliga fall, men i allmänhet kommer verkligheten att ställa oss inför nästan oöverstigliga problem.

Med största säkerhet kommer flera av delmodellerna att vara utvecklade i programspråk och i datormiljöer, som inte alls stämmer överens med vårt simuleringsramverk. Således måste vi konvertera programmen, men det kan vara en ganska omfattande process. Dessutom är det inte alls säkert att programägaren vill släppa ifrån sig sin källkod eller ens kontrollen över hur den exekverbara koden hanteras.

För att ändå kunna utnyttja varandras modeller i gemensamma simuleringar skulle man alltså behöva något som är mindre slutet än vad ett simuleringsramverk är. En framkomlig väg vore att be programkonstruktörerna att alltid se till att varje modellprogram tilldelas förmågan att kommunicera med andra program på ett standardiserat och enkelt sätt, men utan att kräva något speciellt beträffande programmens inre egenskaper. Det vi då behöver är förutom en infrastruktur för själva kommunikationen också en struktur och vissa allmänt accepterade principer för hur modeller skall utvecklas. I detta sammanhang talar man dock snarare om det något vidare begreppet **arkitektur**.

Detta begrepp har, som vi ser i faktaruta 34, med tiden fått en vidgad innebörd. Från att från början ha varit reserverat för byggnader och andra mänskliga artefakter, har ordet så småningom kommit att användas även för föremål i naturen (t.ex. ”trädets arkitektur”) och för abstraktioner som musikstycken eller programvara och modeller. Standarden IEEE 1471 definierar ”*Architecture of Software-Intensive Systems*” som

den grundläggande *strukturen* för systemet, sådan den gestaltas i dess komponenter, deras relationer till varandra och omgivningen, samt de *principer* som styr dess konstruktion och vidareutveckling.

Man kan alltså se arkitektur som ett vidare begrepp för det som förr benämndes

34

Ur Nationalencyklopedin (1989):

arkitektur, byggnadskonst. Termen avser i vidsträckt betydelse allt mänskligt byggande... Bildligt kan arkitektur beteckna uppbyggnaden även hos t.ex. naturobjekt, musik eller poesi. ... Kunskapen [har] kunnat uppfattas som delbar i konstruktion, funktion och form.

Ur Nationalencyklopedins ordbok (1996):

arkitektu´r subst. *~en ~er*

läran om samspelet mellan tekniska och konstnärliga faktorer vid byggande {SYN. **byggnadskonst**}: *~ens historia*

BET.NYANSER: **a)** konkret konstnärlig och teknisk utformning av byggnad: *kyrkan har en märklig ~* **b)** utvidgat: *trädets ~*

in`frastruktur subst. *~en ~er*

ekonomiskt bassystem som utgör grunden för ett (industri)lands försörjning omfattande fasta anläggningar, transportleder, telesystem, utbildningsväsen m.m.: *de nyrika oljeländerna håller på att bygga upp en ~*

Ur Oxford English Dictionary (2005):

architecture

4. The special method or 'style' in accordance with which the details of the structure and ornamentation of a building are arranged.

5. *transf.* or *fig.* Construction or structure generally; both *abstr.* and *concr.*

6. *Computing*. The conceptual structure and overall logical organization of a computer or computer-based system from the point of view of its use or design; a particular realization of this.

infrastructure

A collective term for the subordinate parts of an undertaking; substructure, foundation; *spec.* the permanent installations forming a basis for military operations, as airfields, naval bases, training establishments, etc.

programstruktur, där man har tillfört tydligare styrande regler för att underlätta återanvändning i allmänhet och samutnyttjande i synnerhet. Ett simuleringsramverk kan därför sägas ange en sådan arkitektur, vilken modellskaparen måste anpassa sig till. Den är dock ganska sluten, och det finns många arkitekturer som är mindre tvingande för modelleraren. En öppnare form erbjuds t.ex. i distribuerad simulering med de arkitekturer som man då använder. Vi ska nu titta på ett par exempel på sådana mer eller mindre öppna arkitekturer.

Man kan säga att ***distribuerad simulering*** **är en verksamhet, där modeller av olika typer, byggda av olika leverantörer vid olika tillfällen, körda på olika datorer, eventuellt tillsammans med verkliga objekt och beslutsfattare, samverkar i en simulering under ömsesidig påverkan.**

Ett näraliggande begrepp, men ändå inte synonymt, är **parallell simulering**. I det fallet handlar det om en modell som har programmerats så att den utnytjar flera processer samtidigt. Fokus ligger då på att optimera beräkningshastigheten. I fallet med distribuerad simulering ligger tonvikten inte på samma sätt på prestanda utan på att genom samutnyttjande av resurser åstadkomma mer omfattande simuleringar än vad som annars vore möjligt.

Sin bakgrund har begreppet distribuerad simulering i ett projekt, som 1984 iscensattes av DARPA (*Defense Advanced Research Projects Agency*) vid amerikanska försvarsdepartementet. Det fanns då i USA ett stort antal simulatorer i vilka man kunde träna befattningshavare i olika fordon och vapenplattformar i det rena handhavandet. Det saknades dock möjligheter till samverkansträning och träning i grupp. Lösningen blev att koppla samman flera simulatorer så att deras operatörer kunde agera i en gemensam simulerad omvärld och dessutom se bilden (eller åtminstone en symbol) av varandra på sina skärmar.

Resultatet av denna ansats blev SIMNET (*Simulator Networking*), med vilket man i maj 1986 första gången lyckades koppla ihop två stridsvagnssimulatorer i en gemensam körning.

Då projektet avslutades i april 1988 hade man kommit så långt att man kunde knyta samman flera hundra simulatorer över hela USA i gemensamma simuleringar. Det var då även möjligt att låta verkliga fysiska enheter, utrustade med lämplig kommunikations- och bildskärmsutrustning, delta i spelet på samma premisser som de modellerade förbanden. Medan de verkliga enheterna rörde sig i verklig terräng, simulerades de modellerade förbanden i motsvarande modellterräng. Efter de första framgångarna insågs snart nog behovet av att standardisera hur modellerna i en distribuerad simulering skall samverka med varandra, en process som pågått alltsedan slutet av 80-talet.

Den första standarden, DIS (*Distributed Interactive Simulation*, IEEE 1278), kännetecknas av att alla deltagande simuleringsmodeller (noder) är autonoma och när som helst kan ansluta sig till eller lämna en pågående simulering. Vidare förutsätts alla noder ha en gemensam tidsskala och signalera alla sina tillståndsförändringar till alla andra noder med hjälp av PDU:er (*Protocol Data Units*). Denna efter hand växande mängd av standardiserade PDU:er utgörs av formaterade dataenheter, som utväxlas mellan simuleringsnoderna för att förmedla budskap om objekt och händelser.

Kravet på gemensam tidsskala innebär de facto att det är realtid som gäller, vilket i sin tur innebär att denna första DIS-standard huvudsakligen var avsedd för tränings- och utbildningssimulatorer. Man insåg dock att konceptet även var användbart för andra modeller som var ämnade för analyser och studier. Där simulerades vanligen enheter på högre nivå än enskilda plattformar. Som ett komplement till DIS-arkitekturen utvecklades därför ALSP (*Aggregate Level Simulation Protocol*) för att möjliggöra distribuerade simuleringar av denna typ.

Varken DIS eller ALSP visade sig emellertid vara till fyllest för att möjliggöra distribuerad simulering fullt ut enligt definitionen, som ibland går under beteckningen ADS (*Advanced Distributed Simulation*). Därför initierade DMSO (*Defense Modelling and Simulation Office*) 1994 ett arbete som bl.a. ledde fram till HLA (*High Level Architecture*).

HLA beskriver den strukturella uppbyggnaden hos det programsystem och den infrastruktur som krävs för att olika typer av modeller ska kunna kopplas ihop i gemensamma simuleringar för att kunna samverka med varandra samt även kunna återanvändas som komponenter i framtida gemensamma simuleringsstudier och övningar. En mängd modeller som samverkar med varandra i enlighet med HLA och som har sammanförts för en simulering i ett specifikt syfte kallas en **federation**. De deltagande modellerna benämns **federater**.

I sig själv består HLA av tre delar. För det första har den en **beskrivningsmall för objektmodeller**, OMT – *Object Model Template*. Denna mall består av ett antal tabeller med givna rubriker, men där inget övrigt sägs om hur tabellerna skall implementeras. Det kan alltså ske i XML, HTML, Excel, Word eller något annat system. Med hjälp av denna mall kan man beskriva objektens egenskaper i form av attribut, d.v.s. variabler som kan åsättas värden. Vidare kan man ange objektens statiska relationer i form av klasshierarkier och deras dynamiska relationer i form av interaktioner. Även interaktionerna kan ordnas i klasshierarkier och de kan tilldelas egenskaper, som i detta fall kallas parametrar. Vidare kan attribut och parametrar förses med information om dimensioner, sorter, noggrannhet, uppdateringsfrekvens m.m. Varje begrepp, objekt, attribut eller

parameter, skall dessutom ha en beskrivande text, som gör det möjligt att förstå dess precisa innebörd i modellen.

Denna mall används för tre olika beskrivningar:

- Varje potentiell federat ska beskrivas med en SOM (*Simulation Object Model*), där man redovisar alla de objekt och interaktioner, som federaten modellerar och *vill göra synliga* för andra modeller, eller som den *kan se och reagera på* i andra modeller. Det är således endast de förhållanden som federaten vill använda för framtida kommunikation med andra federater som redovisas. Denna beskrivning kan sägas vara en slags varudeklarationsmärkning av modellen, med vars hjälp en presumtiv federationsbyggare kan avgöra om modellen passar in i hans bygge eller ej. Ett exempel på hur en SOM kan se ut visas i faktaruta 35.
- Till varje federation måste finnas en FOM (*Federation Object Model*), som beskriver alla de objekt och interaktioner som används i kommunikationen mellan federaterna i den aktuella tillämpningen. Den kan sägas utgöra unionen av de deltagande modellernas SOM:ar, eller kanske rentav bara en delmängd av denna. Man kan betrakta denna objektmodell som det protokoll enligt vilket kommunikationen mellan federaterna sker.
- Mindre av principiella men mer av praktiska skäl behöver man även en MOM (*Management Object Model*). I den beskrivs det som behövs för styrning och övervakning av federationen under simuleringen, t.ex. vilka federater som är anslutna, vilka objekt och interaktioner de ansvarar för och förändringar i tidsstyrningen.

Den andra huvudingrediensen i HLA är en **gränssnittsspecifikation**. Den beskriver vilka tjänster som ingår i den infrastruktur som knyter samman federaterna till en federation. Dessa tjänster uppgår till c:a 130 stycken och implementeras som funktioner / procedurer i en programvara som benämns RTI (*Runtime Infrastructure*). Varje potentiell federat skall vara förberedd för kommunikation med andra federater via RTI:n. Praktiskt sker detta genom att federaterna, förutom den egentliga modellen, även innehåller en s.k. RTI-ambassadör och en s.k. federatambassadör. I dessa konverteras information från modellen till anrop till RTI:n resp. anrop från RTI:n till information i modellen. På sätt och vis kan man likna RTI:n vid ett distribuerat operativsystem som betjänar ett antal applikationsprogram (i detta fall simuleringar) och gör det tekniskt möjligt för dem att samverka. Se faktaruta 36!

Slutligen består HLA av **tio regler**, som formulerar förutsättningar och villkor för hur en simulering skall genomföras, vilket ansvar som åvilar de deltagande federaterna och hur interaktionen skall ske. Dessa regler går ut på att varje federation måste ha en FOM och varje federat en SOM, samt att allt det som utlovas i dem också skall uppfyllas via ambassadörerna. Dessutom stadgas att ingen utväxling av information, som finns beskriven i FOM:en, får ske utanför RTI:n, och att endast en federat i taget har rätt att uppdatera ett och samma attribut.

En arkitektur som HLA ger en relativt god grund för interoperabilitet mellan modeller. Man måste dock komma ihåg, att detta bara gäller den tekniska förmågan att samverka. Om modellerna verkligen förstår varandras meddelanden på rätt sätt och förmår utnyttja dem på avsett sätt, det som brukar kallas substantiell, eller semantisk, interoperabilitet, är fortfarande en fråga som måste ägnas stor uppmärksamhet.

Ett ganska vanligt förhållande är att modellerna skiljer sig mer eller mindre beträffande abstraktionsnivån. Aggregering och disaggregering mellan sådana samverkande modeller

Exempel ur en enkel SOM i HLA version 1.3NG i html-format

En *Objcet Model Identification Table* hör också till men utgör endast en ”identifikationsetikett” till modellen.

Objekt och interaktioner

Object Class Structure Table	
Target_Element (N)	Tank (P)
	Armoured_Vehicle (P)
	Soldier (P)

En objektklass *Target_Element* specialiseras till tre olika publicerbara (P) klasser. Den här aktuella modellen behöver/kan inte subskribera (S) på andra objekt.

I detta fall kan modellen initiera (I) en interaktion och själv reagera (R) på en som initieras utifrån.

Interaction Class Structure Table
Firing (R)
Brisade (I)

Attribut och parametrar

Attribute Table												
Object	Attribute	Datatype	Cardinality	Units	Resolution	Accuracy	Accuracy Condition	Update Type	Update Condition	T/A	U/R	Routing Space
Target_Element	Is_Dead	boolean	1	N/A	true, false	N/A	N/A	Conditional	After explosion	N	UR	N/A
	Position	vector	1	m	N/A	N/A	N/A	Static	N/A	N	UR	N/A

Attributen beskrivs med typ, antal, sort, upplösning, maximalt tillåten avvikelse, uppdateringsfrekvens, ägarskap (T= transferable, A = acceptable), om det kan publiceras (U = updateable) eller tas emot (R = reflectable) samt ev. filtrering vid sändning / mottagning (N/A = not applicable).

Parameter Table								
Interaction	Parameter	Datatype	Cardinality	Units	Resolution	Accuracy	Accuracy Condition	Routing Space
Firing	Ammunition	ammunition_type	1	N/A	N/A	N/A	N/A	N/A
	Firing_direction	vector	1	rad	N/A	N/A	N/A	
	Firing_point	vector	1	m	N/A	N/A	N/A	
	Firing_speed	double	1	m/s	N/A	N/A	N/A	
	Target_point	vector	1	m	N/A	N/A	N/A	
Brisade	BurstPoint	vector	1	m	N/A	N/A	N/A	N/A

Interaktionsparametrarna beskrivs i tillämpliga delar på samma sätt som objektattributen.

Enumerated Datatype Table		
Identifier	Enumerator	Representation
Ammunition_Type	m77	1
	slufa	2
	granat_p	3
	m483sgr	4
	bonus	5
	sub	6
	rb63	7
	strix	8
	rb63rak	9
	GPbomb250	10
	new_am	11

Egendefinierade datatyper

Det finns ett antal enkla datatyper definierade i HLA, men man kan också definiera egna datatyper, antingen som uppräkningsbara eller som komplexa.

Complex Datatype Table							
Complex Datatype	Field Name	Datatype	Cardinality	Units	Resolution	Accuracy	Accuracy Condition
Vector	Z	double	1	N/A	N/A	N/A	N/A
	Y	double	1	N/A	N/A	N/A	N/A
	X	double	1	N/A	N/A	N/A	N/A

FOM/SOM-lexikonet

Object Class Definitions	
Term	Definition
Target_Element	An elementary target that could be either a soldier, a tank or an armoured vehicle.

Interaction Class Definitions	
Term	Definition
Firing	The firing unit aims and fires the number of shots required
Brisade	The warhead will thow out fragments or bullets or activate a shaped charge

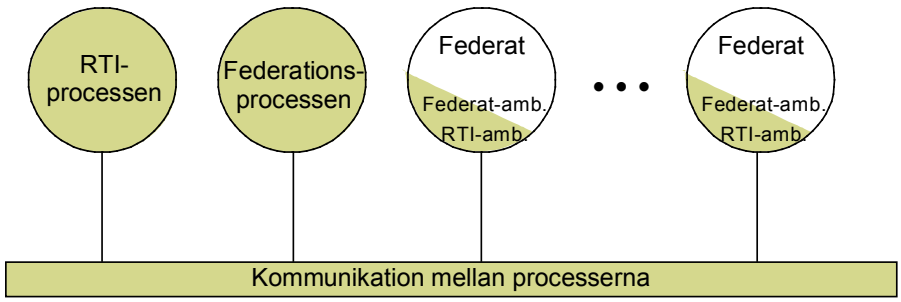
FOM/SOM-lexikonet innehåller en verbal beskrivning av alla objekt, interaktioner, attribut och parametrar, som förekommer i övriga tabeller.

Attribute Definitions		
Class	Term	Definition
Target_Element	Is_Dead	Indicates whether the target element is still functioning in its task or not.
Target_Element	Position	The geographical position of the target element.

Parameter Definitions		
Class	Term	Definition
Firing	Ammunition	Tells what kind of ammunition that is used
Firing	Firing_direction	The firing direction given in three dimensions
Firing	Firing_point	The position of the firing device
Firing	Firing_speed	The velocity of the warhead when leaving the firing device
Firing	Target_point	The aiming point in mind
Brisade	BurstPoint	The actual point of the brisade

36

Gränssnittet i HLA



Gröna delar representerar det som RTI:n tillhandahåller, vita det som utgör själva modellerna. Varje federat skall implementera en federatambassadör och en RTI-ambassadör med hjälp av gränssnittstjänsterna.

Gränssnittstjänster finns för sex olika behov:

Federationshantering	*Deklarationshantering*	*Objekthantering*
Tidshantering	*Ägarskapshantering*	*Datadistributionshantering*

Låt oss till exempel titta på de tjänster som hör hemma under federationshanteringen. Dem skall man använda för att starta och avsluta en simulering med en federation, ansluta till och lämna en simulering, ta paus, spara aktuellt tillstånd, starta om etc. Tjänsterna är följande (blått innebär att federaten anropar sin RTI-ambassadör, rött att RTI:n anropar federatambassadören):

Create Federation Execution	Initiate Federate Save
Destroy Federation Execution	Federate Save Begun
Join Federation Execution	Federate Save Complete
Resign Federation Execution	Federation Saved
Register Federation Synchronization Point	Request Federation Restore
Confirm Synchronization Point Registration	Confirm Federation Restoration Request
Announce Synchronization Point	Federation Restore Begun
Synchronization Point Achived	Initiate Federate Restore
Federation Synchronized	Federate Restore Complete
Request Federation Save	Federation Restored

Exempel 1: **Uppstartsprocessen:**

1. Användaren startar RTI-processen
2. Användaren startar sin federat och definierar federationen genom att anropa funktionen Create Federation Execution i sin RTI-ambassadör. Därvid reserveras ett unikt namn hos RTI-processen, federationsprocessen startar, registrerar sin kommunikationsadress hos RTI-processen samt läser in FOM:en från en fil i standardiserat utförande.
3. Olika operatörer ansluter sina modeller till federationen. Deras RTI-ambassadörer registrerar sin egen adress och frågar RTI-processen om adressen till federationsprocessen för att meddela den senare om sin existens genom att anropa funktionen Join Federation Execution.

Exempel 2: **Processen att spara ett aktuellt tillstånd:**

1. En federat begär att tillståndet sparas genom att anropa Request Federation Save.
2. RTI:n beordrar samtliga federater att börja spara genom anropet Initiate Federate Save.
3. Varje federat kvitterar ordern genom Federate Save Begun.
4. Då en federat är klar, meddelar den detta genom Federate Save Complete.
5. Då samtliga federater rapporterat att de är klara, meddelar RTI:n detta till den federat som ursprungligen begärde operationen genom anropet Federation Saved.

är i sig en form av modellering och simulering som kräver en stor sakkunskap i den disciplin inom vilken den utförs. Skall det lyckas är det av stor vikt att den utförs i nära samarbete med sakkunniga. Det är troligtvis inte heller ovanligt med oväntade skillnader i ”fidelitet” (graden av överensstämmelse med ”verkligheten”) mellan modeller. Beroende på hur de ursprungligen var tänkta att användas har man i modellerna infört olika approximationer av samma företeelse. Speciellt om en sådan företeelse är tämligen perifer i en modell är det inte heller säkert att detta uppmärksammas särskilt mycket i dokumentationen. Interoperabilitet är alltså bra, men allomfattande interoperabilitet (*universal interoperability*) är inte ett rimligt mål.

HLA, som ursprungligen utvecklades som en standard inom amerikanska försvaret (DMSO 1.3NG), har sedermera även blivit en IEEE-standard med nummer 1516, som i vissa avseenden blivit bättre och klarare i sin struktur. Tyvärr används dock fortfarande den gamla standarden parallellt med den nya, vilket medför en viss förvirring, eftersom de inte är helt kompatibla. Ändå kan nog HLA sägas ha blivit ganska etablerad inom västvärldens försvarsmakter och försvarsindustrier. Den har dock svagheter, något som driver utvecklingen vidare mot nya framtida arkitekturstandarder.

En sådan svaghet är att OMT inte är fullt ut utan bara nästan i överensstämmelse med objektorienteringsparadigmet. En annan HLA-kritik riktar in sig mot att infrastrukturen är alltför statisk, att man måste dra med sig en stor mängd tjänster som man inte utnyttjar, samtidigt som man kanske skulle vilja lägga till egna tjänster någon gång.

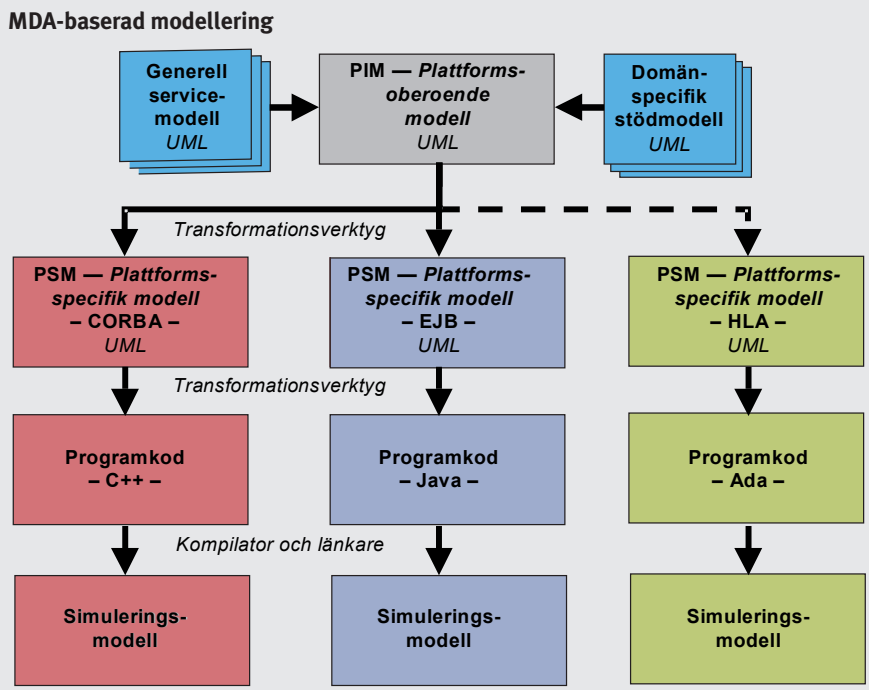
Av större betydelse är nog de tendenser som finns att utveckla arkitekturen som sådan. Därvid kommer impulser från olika håll, som ger löften om möjligheter och utmaningar i ett större sammanhang. Kopplingen mellan M&S och ledningssystem (C^4I, *Command, Control, Communications, Computing and Intelligence*) är en sådan drivkraft. Å ena sidan skulle den kunna leda till att befintliga arkitekturer på ömse sidor utökas med ett gemensamt gränssnittstillägg. Å andra sidan finns även idéer om att betrakta de båda domänerna som en enda med en sammansatt funktionalitet och gemensam informationsmodell.

Inom den civila M&S-marknaden finns också intressanta trender. Sedan länge har där funnits flera olika etablerade infrastrukturer, t.ex. CORBA (*Common Object Request Broker Architecture*), SOAP (*Simple Object Access Protocol*) och Java/EJB (*Enterprise Java Beans*). Somliga har också stark uppbackning från t.ex. OMG (*Object Management Group*), en sammanslutning av en stor mängd företag runt om i världen. Många sådana strukturer är dessutom rikt representerade inom försvarssektorns olika system.

Ur interoperabilitets- och återanvändningssynpunkt är det naturligtvis inte så bra med många olika och delvis konkurrerande arkitekturer. OMG har därför på senare år gjort ett försök att angripa detta Babels torn-fenomen genom att betrakta modellarkitekturen från en högre generisk nivå, en som är oberoende av infrastrukturerna. Resultatet skulle då kunna bli att man får stabila och generellt återanvändbara metamodeller, som sedan vid behov kan förverkligas med olika infrastrukturer med hjälp av standardiserade transformationer. Detta koncept har blivit känt under namnet MDA (*Model Driven Architecture*).

Utvecklingen av en MDA-baserad simuleringsmodell skulle kunna se ut på följande sätt (se faktaruta 37). Man utvecklar först en PIM (*Platform Independent Model*) i UML med hjälp av standardiserade UML-profiler ur två

37



Principiell bild över transformationsstegen i MDA. Infrastrukturer och programspråk är endast exempel. Kopplingen till HLA är endast föreslagen.

bibliotek. Det ena är domänspecifikt och består i princip av konceptuella modeller för domänen. Det andra består av generellt användbara komponenter som tidshantering, visualisering, terrängrepresentation o.s.v. För varje typ av infrastruktur (CORBA:s, EJB:s o.s.v.) skall det finnas ett transformationsverktyg, med vars hjälp man från PIM:en konstruerar en PSM (*Platform Specific Model*), även den i UML, där implementationsberoende delar inkorporerats. Med ytterligare transformationsverktyg skall man sedan kunna konvertera UML-beskrivningen till kod i något lämpligt programspråk (Java, C++, Ada o.s.v.), som sedan kan kompileras och länkas till den eftersträvade simuleringsmodellen

MDA-konceptet är tilltalande, inte minst därför att det tycks erbjuda goda möjligheter att koppla ihop modeller på ett sätt som underlättar substantiell interoperabilitet. Hopkopplingen sker här redan på PIM-nivån, varigenom man får en ny PIM, som kan verifieras och valideras innan transformationen sker till den valda infrastrukturen. Speciellt intressant blir det om man utvecklar transformationsverktyg från PIM till PSM för HLA. Detta har också föreslagits, och det skulle fr.a. innebära en klar förbättring för försvarsmaktens distribuerade simuleringar.

Ytterligare ett arkitekturexempel, s.k. **komponentbaserad arkitektur**, har vi f.ö. sett en antydan till redan i det sista stycket i faktaruta 25 i samband med diskussionen om komponentbaserad modellutveckling.

Dataförsörjning

Gång efter annan har vi i den föregående texten berört de problem som kan uppstå beträffande hur modeller skall förses med adekvata data.

Eftersom detta är centralt vid all modellanvändning skall vi här än en gång ta upp och sammanfatta vad man måste tänka på därvidlag.

Först bör vi göra klart för oss vad data egentligen är. Som vi ser i faktaruta 38 var ordet förr i tiden liktydigt med fakta i största allmänhet. Under tidens gång och kanske främst genom datorernas (datamaskinernas) ankomst, har det fått en mera inskränkt betydelse: kvantifierbara fakta.

Data är alltså i vårt sammanhang liktydigt med information i sifferform. Tillsammans med kunskap om de förutsättningar under vilka siffrorna tillkommit gör data vår egen kunskap om ett visst förhållande mera stringent. Men för att det skall bli riktig och värdefull kunskap får man inte slarva med de nämnda förutsättningarna. Data existerar bara i sitt sammanhang – i sin modell. Tappar man bort den modell, enligt vilken data togs fram, är de bara en samling numeriska värden.

För att illustrera detta kan vi betrakta talserien 5560, 2808, 1196 och 990. Den säger absolut ingenting förrän man vet att det handlar om kokpunkten i °C för grundämnena molybden, guld, ytterbium resp. tellur. Fast å andra sidan skulle det också kunna vara be-

38

Ur Svenska Akademins ordbok (1908):

Datum

1. **BETYDELSE:** (i sht i vetenskapligt språk) ngt givet; uppgift (som kan läggas till grund för ett resonemang osv.); faktum; vanl. i pl.

Ur Nationalencyklopedins ordbok:

kun`skap subst. *~en ~er* (1996)
välbestämd föreställning om (visst) förhållande eller sakläge som ngn har lagrad i minnet etc., ofta som resultat av studier e.d.

information subst. *~en ~er* (1996)
(meddelad) mängd fakta vanl. av mer el. mindre exakt slag

fak`tum subst. ~ el. *~et, fakta* el. *~,* best. pl. *fakta* el. *~en* (1995)
sakförhållande som inte kan bestridas

da`ta subst., plur. (1995)
uppgifter (om ngt) av förhållandevis exakt slag gärna resultat av mätningar e.d.; ofta i sifferform

tal subst. *~et,* plur. *~,* best. plur. *~en* (1996)
en grundläggande matematisk storhet som i sin enklaste form primärt anger antal

folkningen år 1995 i tätorterna Surte (i Västergötland), Skinnskatteberg (i Västmanland), Koskullskulle (i Lappland) resp. Färlöv (i Skåne). Eller varför inte arealen i km^2 av öarna Bali (i Indonesien), Kupreanof (i Alaska), Lifou (i Nya Caledonien) resp. Dagö (i Estland). Tal i sig är alltså inte data, det fordras ett sammanhang att relatera talen till. Som vi kan se av faktaruta 38 är kunskap i sin tur en integration och bearbetning av tillgänglig information i ens eget medvetande och föreställningsvärld. Om alltså de data som man vill använda inte är i överensstämmelse med den kunskap, som får sitt uttryck i den modell som utvecklas, så kan det bli ganska fel.

Nu är ju i allmänhet inte tolkningsskillnaderna så stora som i exemplet ovan, men det skulle kunna vara så att det experiment som givit aktuella data om exempelvis skador på ett militärt radiokommunikationssystem förutsatte att förbandet befann sig i en försvarssituation medan den nya modell vi skall bygga huvudsakligen handlar om anfall. Man bör då fråga sig om det kanske finns anledning att förkasta de data som erbjuds och i stället genomföra nya experiment.

Att använda existerande data, eventuellt efter viss mer eller mindre välgrundad modifiering eller att ta fram nya är en kostnadsfråga men fr.a. en trovärdighetsfråga. Det är ofta lätt gjort att man för att klara av sin studie inom budgetramen utnyttjar data som man egentligen vet inte riktigt återspeglar den situation man vill studera, men nästan. I bästa fall gör man då vissa korrigeringar grundade på mer eller mindre intelligenta gissningar. I sämsta fall ändrar man sin frågeställning för att passa data.

Av intresse är också hur väl underbyggda data är. Är de framtagna i ett enda försök, så representerar de *en* möjlig värdeuppsättning, men hur typisk denna är kanske man inte vet så mycket om. En fördel är om de värden man vill använda sig av är försedda med någon slags osäkerhetsspecifikation, t.ex. 3,72 ± 0,18. Allra bäst är om man kan få en detaljerad frekvensfunktion för värdenas fördelning. Vi kan då välja att göra vår modell stokastisk och därigenom få möjligheter att studera i princip hela utfallsrummet på ett någorlunda enkelt och uttömmande sätt. Under alla omständigheter är det viktigt att det finns en trovärdig dokumentation, som förklarar hur man har fått fram dessa värden och deras eventuella spridning.

Sårigheterna att experimentellt bestämma vissa parametrar kan ibland vara betydande. Det kan vara fråga om förstörande tester av dyrbara system, tester som är omöjliga därför att de är för farliga att genomföra eller tester som är etiskt eller juridiskt omöjliga. Systemet i fråga kan dessutom vara otillgängligt för experiment och kanske inte ens vara möjligt att direkt observera, t.ex. hemliga fientliga vapensystem eller egna system på idéstadiet. Kunskap och data kanske då måste härledas ur mer grundläggande och detaljerade modeller. I detta fall får vi dessutom effekten att sambandet mellan data och deras bakomliggande modell blir explicit tydligt, i det generella fallet finns det ju kanske bara en mental modell hos den som utförde experimentet eller på annat sätt härledde data.

Vill man bygga mycket detaljerade modeller kan det å andra sidan vara mycket svårt att få fram relevanta data. För sådana modeller kan dessutom även de intelligentaste gissningar slå mycket fel. Man kan t.ex. tänka på vad som händer om man skall modellera den inre strukturen hos en fientlig farkost, som man kanske bara har ett exteriörfoto av, och som dessutom

kan vara av dålig kvalitet. I detta fall kan våra gissningar placera olika komponenter i fel del av farkosten och därmed göra sårbarhetssimuleringar helt otillförlitliga. Det är då ofta lättare att ”gissa” värden på en aggregerad nivå.

Att det kan vara svårt att komma åt främmande makters och organisationers data är naturligtvis inget att förundras över. Men sekretessproblematiken kan ställa till problem även när det gäller att få tag på relevanta data inom den egna nationen eller organisationen, och får man tillgång till dem kan de vara förknippade med så starka restriktioner att det blir svårt eller omöjligt att använda dem i simuleringar, åtminstone i den situation och miljö de är tänkta att köras i.

Om nu inte experimentella eller sekretessmässiga hinder föreligger, så kan en icke väl genomtänkt modell ändå bli ganska svår att parametersätta. Exempelvis kan man råka modellera objekt, som är mer eller mindre esoteriska eller åtminstone ogripbara i den verkliga världen. I allmänhet blir det enklare att finna lämpliga data om modellobjekten i möjligaste mån överensstämmer med verkliga objekt om vilka man har en tillräcklig kunskap.

Ibland är inte knappheten på data det stora problemet, utan snarare motsatsen, ett överflöd av data som kan vara mer eller mindre motstridiga. Att då avgöra vilka man skall använda är en fråga som närmast kan liknas vid de humanistiska vetenskapernas källkritik. Fast här beskriver vi det snarare som att vi måste analysera de bakomliggande modellerna: vilken eller vilka stämmer bäst överens med den modell vi vill använda data till? Finns det en fungerande certifieringsprocess inom den verksamhet där modellen skall användas, så har man inte minst i detta fall mycket lättare att bemästra problemen.

Data är ofta färskvara, efter en tid är de data som togs fram i ett i och för sig väl anpassat experiment inte längre säkert användbara. Själva den kunskap som experimentet genererade kan ha påverkat förutsättningarna i stort eller i en och annan detalj. Teknik- och taktikutvecklingen bidrar också till denna åldrandeprocess. Detta kan vara detaljer som lätt kan förbises i den dokumentation av den bakomliggande datamodellen, men som man som användare bör vara observant på. En speciell form av åldrande kan dessutom bestå i att det datamedium som använts inte längre kan läsas med modern utrustning, eller att transformationen av data från en gammal databas till en ny kan medföra ökad onoggrannhet.

Är man nu osäker beträffande vilken kvalitet de valda indata har, så kan ibland problemet desarmeras genom att man utför känslighetsanalyser. Visar det sig då att utdata är relativt okänsliga för variationer i en viss parameter, så behöver man kanske inte bekymra sig alltför mycket över att man inte har fullgod kunskap om dess sanna värde.

Rent allmänt kan sägas att data som vi redan från början vet att vi kommer att variera i våra studier behöver man kanske inte vara så helt säker på. Det kan räcka att ha ett allmänt hum om i vilket område variabelvärdena kan röra sig och ungefär hur de fördelar sig. Sedan är det upp till modellanvändaren att se till att man genomför simuleringar med ett antal olika indatauppsättningar på ett sådant sätt att man fördjupar förståelsen för hur dessa variabler påverkar utfallet.

Krigsspel och M&S

Hittills har vi i denna skrift huvudsakligen ägnat oss åt generella aspekter på modellering och simulering. I fortsättningen skall vi diskutera några tillämpningsområden med speciell relevans för militära verksamheter.

I det inledande kapitlet har vi berört den speciella typ av simulering i levande livet (*live simulation*) som brukar kallas krigsförbandsövning (det som förr kallades ”simulaker”, ett i M&S-sammanhang passande ord, som dock kom helt ur bruk efter första världskriget). En sådan simulering är ju till för att träna alla nivåer i organisationen för sina uppgifter i ett eventuellt krig.

Även om regelbundna krigsförbandsövningar är tillfyllest för att upprätthålla och utveckla kompetensen hos de lägre nivåerna i den militära hierarkin, så fordras något mer för de högre beslutsfattarnas träning, för att inte tala om planering, taktikutveckling m.m. Att för dessa ändamål dra samman stora förband, är naturligtvis praktiskt och ekonomiskt inte möjligt. Här behövs alltså någon form av modellering och simulering, det som vi kallar krigsspel.

Idén till krigsspel, sådana vi känner dem idag, uppkom i Preussen i början av 1800-talet och började användas i Sverige 1830. I det ursprungliga upplägget genomför två grupper ett spel mot varandra med hjälp av spelpjäser på en karta eller i en terrängmodell. Varje drag omfattar en viss tidsrymd och vissa händelser. Spelledaren, som har en liknande men mera omfattande funktion som stridsdomarna i krigsförbandsövningar, avdömer vilka konsekvenser speldraget får och meddelar det till deltagarna, som därefter får resonera sig fram till nästa drag.

När datorerna så småningom blev allmänt tillgängliga, kunde man åtminstone delvis börja ersätta den mänskliga spelledarens subjektiva bedömningar med mera objektiva datormodeller.

Krigsspel kan med fördel användas i fr.a. tre sammanhang, som studiehjälpmedel, som planerings- och som övningsinstrument. I ett studiespel analyseras och värderas framtida system, strategier och taktiker. Då betonas inte uppdelningen av deltagarna i olika lag så starkt, utan spelgruppen, som består av experter inom olika delområden, resonerar sig ofta gemen-

39

Ur Svenska Akademins ordbok:

Manöver (1942)

ETYMOLOGI: [jfr t. *manöver*, eng. *manœuvre; av* fr. *manœuvre,* av vulgärlat. *manuopera*], eg.*: arbete med handen,* av lat. *manus, hand,* o. *opus* (pl. *opera), verksamhet]*

3.b. BETYDELSE: större fälttjänstövning, fältmanöver; äv. om större stridsmanöver vid flottan l. flyget.

Fältmanöver (1926)

(numera icke i officiellt spr.) större fälttjänstövning.

Fälttjänstövning (1926)

BETYDELSE: övning i fälttjänst.

Simulaker (1967)

ETYMOLOGI: [av fr. *simulacre*, av lat. *simulacrum*, avbild, skuggbild, sken(bild), till *simulare*]

1. BRUK: (numera br. mera tillf., ålderdomligt)

BETYDELSE: låtsad sammandrabbning l. strid mellan truppstyrkor l. soldater; krigsövning l. fälttjänstövning; i sht förr äv. [jfr fr. *faire simulacre*] i uttr. göra simulaker, ha militär övning varvid man fingerar strid.

Krigsspel (1937)

BETYDELSE: på karta l. terrängmodell l. i terrängen utan trupp utförd övning i taktiska l. strategiska rörelser l. operationer.

Ur Nationalencyklopedin (1993):

Försvarsmaktens personal inkallas regelbundet till ***krigsförbandsövning*** för att vidmakthålla krigsdugligheten. Därvid övas förbanden i de uppgifter och den (krigs)miljö som de beräknas användas i.

krigsspel, simulering av krigs- eller stridsförlopp i studie- eller utbildningssyfte eller för att pröva krigsplanläggning.

samt fram till en uppfattning om hur de båda sidorna kan tänkas agera. Här är det en fördel om den stödjande datormodellen är så flexibel, att man snabbt kan ändra förutsättningarna för att möjliggöra studiet av alternativ, som uppkommer under diskussionerna.

I planeringsspelen, som ibland ses som en speciell variant av studiespelen, modelleras vanligen befintliga förband och system med det primära målet att utveckla organisationens operativa planering. En variant, där man oftast spelar med flera självständigt agerande parter, är politisk-militära spel, där också civila aspekter och aktörer förekommer. Här är målet att förstå motsättningar och samspel vid politiska, sociala och etniska konflikter och kriser.

I övningsspel ligger tonvikten på att utbilda och träna större eller mindre grupper. Själva spelet bestämmer då den omvärld och de angränsande verksamheter som har betydelse för den situation som skall övas. Deltagarna fattar sina beslut vid varje drag och får feedback på dem. Man kan i detta fall ha en enda grupp som spelar mot en planerad motståndare, som följer en i förväg uppgjord plan, eller två eller flera grupper som spelar mot varandra. I det senare fallet kan man låta alla få tillgång till all information och låta spelarna själva bedöma vilken information de i verkligheten skulle ha haft tillgänglig att grunda besluten på. Ett trovärdigare, men mera tids- och resurskrävande, alternativ är att hålla spelgrupperna isolerade från varandra och bara förmedla selektiv information till dem. I så fall kan man också låta dem agera i sina verkliga miljöer och med de normala informationskanalerna, så att situationen upplevs likadant som i ett skarpt läge. I detta fall ser deltagarna ingenting av själva spelet, det gör bara spelledningen.

Ett krigsspel kan sammanfattningsvis karaktäriseras som

- ett sätt att skapa en **konkret ram** för diskussioner inom en vidare krets än vad t.ex. en simuleringsmodell medger,
- ett sätt att **behandla olika** verkligheter och **uppfattningar** om dessa för att uppnå ett visst (studie-) syfte, samt
- en samling **regler** om resurser, vinster och förluster samt olika slags **speldrag**.

I faktaruta 40 återges några sätt att kategorisera krigsspel. Liksom i alla andra modelleringssammanhang gäller även här att det är viktigt att välja rätt komplexitetsgrad. Ett realistiskt och komplext spel är mycket resurskrävande och ofta svårt att genomskåda vad gäller orsak och verkan i utfallet.

Ett krigsspel har i allmänhet många fördelar som studie- och utbildningsinstrument:

- Det är till sin natur kunskapsuppbyggande, man lär genom praktiska erfarenheter och genom gemensamma diskussioner
- Det är flexibelt, det kan anpassas så att tiden utnyttjas till de väsentliga problemen medan man snabbt kan spela förbi de mera rutinartade situationerna.
- Genom att spelet engagerar många experter och sakkunniga uppstår det ofta värdefulla synergieffekter under det gemensamma arbetet.
- Med ett krigsspel kan man simulera beslutsfattandet på olika nivåer på ett mycket verklighetstroget sätt.
- Det kan vara ett bra sätt att skaffa fram underlag för utveckling av konceptuella och simuleringsmodeller.

40

Klassificering av spel

Begreppet *spel* saknar en allmänt erkänd definition. Det är dock nära besläktat med både *simulering* och *övning*. Här redovisas ett klassifikationsschema enl. [Dreborg, 1993].

Indelning efter syfte

1. Underhållningsspel
2. ”Seriösa” spel
   - 2.1. Spel som utbildningsmetod
   - 2.2. Spel som undersökningsmetod

Indelning efter funktion

1. Spel för kunskapsöverföring
2. Spel för kunskapsgenerering

Speltyper efter funktion och syfte:

Funktion / Syfte	Kunskaps-överföring	Kunskaps-generering
Spel som utbildningsmetod	Undervisnings-spel	Övningsspel
Spel som undersökningsmetod	Expertspel	Värderingsspel Testspel

Med expertspel menas här ”spel där spelupplaggaren med spelets hjälp försöker komma åt delvis tyst kunskap hos en grupp experter.”

Indelning efter komplexitet

1. Enkla, idealiserade spel
2. Komplexa, realistiska

Användbarhet hos spel med varierande komplexitetsgrad och syfte:

Komplexitet / Syfte	Enkelt och idealiserat	Komplext och realistiskt
Spel som utbildningsmetod	Lär ut övergripande principer	Ger förståelse för helhet och roller
Spel som undersökningsmetod	Stödjer forskning	Indikerar problemområden

Observera att komplexitetsindelningen till skillnad från de andra indelningsgrunderna har ett kontinuerligt spektrum av mellanlägen.

Det finns naturligtvis också påtagliga nackdelar med ett spel:

- Det är tids- och personalkrävande.
- Det är svårt att reproducera resultaten.
- Man måste acceptera även sådana resultat, som inte kunnat genomlysas så väl som de borde.
- Det kan vara svårt att inse vilka faktorer som verkligen är betydelsefulla.

Att spel är resurskrävande kan illustreras med faktaruta 41, som visar storleksordningen av nödvändig framförhållning vid spel på operativ nivå. Den visar också tydligt att själva spelet sällan är det som tar längst tid, det är snarare förberedelserna och efterarbetet.

Till viss del kan krigsspelens nackdelar reduceras med hjälp av simuleringsmodeller. Det är fr.a. följande fördelar, som gör simuleringar med hjälp av dator till ett attraktivt hjälpmedel:

- Det krävs förhållandevis liten tid att genomföra en simulering, åtminstone i det fall då modell och data redan finns framtagna.
- Delproblemen är genom modelleringen väl strukturerade och analyserade.
- Det finns möjlighet att studera många variationer.
- Datorerna tillhandahåller en stor minneskapacitet, vilket medger såväl mer detaljerade studier som en mer omfattande bokföring av vad som händer i spelen.

Som vi vet har ju simuleringsmodellerna också nackdelar, t.ex. att

- de kan vara så komplexa att de inte är transparenta,
- ofta bara experter kan hantera dem, och
- de lätt kan inge obefogat stort förtroende (eller tvärtom).

Med en väl genomtänkt användning kan emellertid kombinationen av krigsspel och simuleringsmodeller vara ett utomordentligt effektivt verktyg för allehanda studie- och utbildningsproblem. Man talar här om **blandade spel** till skillnad från **manuella spel**, där man inte använder datorstöd.

Spel kan också karaktäriseras med avseende på andra parametrar. Spelet kan vara **styrt** eller **icke styrt** (ibland också kallat **fritt**). I det förra fallet finns det bestämda ramar och definierade regler redan från början. I det senare tillåts spelet utveckla sig fritt utan hänsyn till (alltför strikta) ramar, och reglerna kan ändras under spelets gång.

Ett spel kan också vara **öppet** eller **slutet**, något vi redan berört i diskussionen om studie- och övningsspel. I det öppna spelet sitter alla deltagare i samma rum, vilket har fördelen att det kräver mindre resurser och är enklare att administrera. Dessutom kan tillgängliga

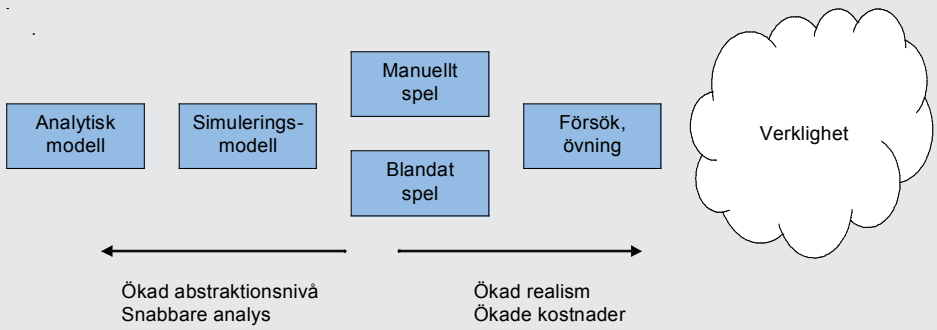
41

Storleksordningar för tidsåtgången vid operativt krigsspel

Typisk tidsram	Aktiviteter
År	Framtagning av underlag som teknisk prognos, hotverk, avdömningsunderlag och simuleringsmodeller
Månad	Framtagning av strukturer, ekonomi, scenarier, ambitioner, syften, lokaler, specialister, stridsplaner
Vecka	Framtagning av kartunderlag, formulär m.m., inläsning av underlag, allokering av hårdvarustöd
Vecka	Grundspel
Dag	Variationsspel
Vecka	Analys, dokumentation

42

Modeller – spel – verklighet



I vilken ordning de båda spelalternativen skall sättas är inte givet. Om man jämför ett manuellt och ett blandat spel med samma nivå på realism, så kräver det förra större resurser, samtidigt som det ändå ofta blir billigare och snabbare eftersom handlingsfriheten är större.

Observera att relationerna i bilden endast gäller då simuleringsmodellen redan existerar. Behöver man utveckla en helt ny modell eller modifiera en existerande kan ordningen avseende tid och kostnad kastas om.

experter utnyttjas effektivare och diskussionerna blir mer allsidiga. Risken med öppna spel är dock uppenbar, besluten kan bli alltför rationella, eftersom det kan vara svårt att bortse från information, som man normalt inte skulle ha haft tillgång till. Om det alltså är viktigt att beslutsfattandet blir realistiskt representerat, bör man utnyttja slutna spel, där de olika aktörerna sitter i skilda rum.

Ytterligare ett sätt att beskriva spel är som **fria** eller **skriptade**. Detta är fr.a. aktuellt vid spel vars främsta syfte är övning. I fria spel agerar alla parter efter bästa förstånd. I skriptade spel måste den ena sidan (den icke övande) följa en i förväg uppgjord plan med bestämda inspel. Det förekommer även blandformer. Om t.ex. övningen i ett fritt spel är på väg att ta en vändning som inte främjar övningsändamålet kan spelledningen ålägga den icke övande sidan vissa restriktioner.

I studiespel är valet av speltyp ofta beroende av var man befinner sig i studieprocessen. Det är alltså viktigt att man fattar medvetna beslut i denna fråga. I en förstudie där man vill

- få ett bättre grepp på problemformuleringen,
- skaffa sig en uppfattning om studiealternativens egenskaper och
- ta fram beslutsunderlag om spel- och annan metodik för huvudstudien,

är det exempelvis oftast fördelaktigt att genomföra ett icke styrt spel.

Under huvudstudien kan man tänka sig flera olika speltyper. Gäller studien analys av olika alternativ är ofta öppna, styrda spel lämpliga. Då man varierar spelet för olika alternativ är det nämligen viktigt att få en ordentlig förståelse för varför olika händelser inträffar. Därigenom kan man minimera risken att få resultat som beror på vissa aktörers oskicklighet eller på ren slump och får bättre förutsättningar att bedöma vilka utfall som verkligen är systemskiljande.

Ett annat exempel är när man före en omgång med styrda spel vill genomföra ett översiktsspel. Meningen är då att skaffa sig en uppfattning om vilka de viktiga händelserna är, så att huvudstudien kan koncentreras till dem. Det kan också vara fråga om att undersöka om det valda scenariot kommer att belysa de frågeställningar som huvudstudien skall besvara. I sådana fall är icke styrda spel utan tvekan bäst lämpade.

Ett krigsspel kan som andra simuleringar genomföras **händelsestyrt** eller **tidsstyrt**. Vilket man i varje enskild situation väljer beror naturligtvis på hur komplex och svåröverblickbar situationen är. Ju enklare att överblicka, ju naturligare är det med ett händelsestyrt spel (-avsnitt), och vice versa. Skulle den stödjande datorsimuleringen använda en realtidsmodell blir naturligtvis även krigsspelet tidsstyrt.

I ett händelsestyrt spel kan det ibland vara praktiskt att inte hantera händelserna i strikt tidsföljd. Det kan t.ex. vara lämpligare att klara av en hel händelsekedja inom ett visst geografiskt område innan man tar itu med händelser som berör andra områden. Här måste man dock se upp, så att inte en viss händelse inom ett område faktiskt påverkar vad som händer i ett annat. I ett sådant fall måste man synkronisera händelsekedjorna vid motsvarande tidpunkt.

Man bör även beakta att i en och samma studie kan man ha många syften med sina spel. Det är då inte säkert att alla syften uppnås med spel på en och samma nivå. Genom att växla mellan spel på högre nivå och spel av mer begränsade delsituationer kan tiden utnyttjas bättre och dessutom vinner man kanske en djupare förståelse för orsakssammanhangen.

För att alla aktörer i ett spel skall agera utifrån samma förutsättningar använder man s.k. **spelkort**. Ett sådant beskriver allt som den spelande behöver veta om ett visst förband (dess innehåll, organisation, avsedda användning och förmåga) eller om en viss utrustning (prestanda, rörlighet, verkan, etc.). Dessa kort är speciellt viktiga i spel med scenarier långt in i framtiden där deltagarna inte kan ha någon personlig erfarenhet av de objekt de hanterar. Då alla aktörer i ett spel är väl förtrogna med de förband och den materiel som används, vilket vanligen är fallet i övningsspel, behöver man däremot i allmänhet inte några spelkort.

Avdömningarna i ett spel är naturligtvis centrala för att föra spelet framåt genom att de genererar nya händelsekedjor. Vi har redan sett att de kan grunda sig på spelledarens bedömningar eller på resultatet av simuleringar med datormodeller. Man kan ju också tänka sig att de sker som resultat av mer detaljerade spel eller rent av förbandsövningar, fältförsök och tekniska prov. Under alla förhållanden gäller det hela tiden att dokumentera vilka avdömningar som görs och varför resultaten blir som de blir.

Ledningssystem och M&S

Ett andra tillämpningsområde med stor såväl militär som civil relevans, och som därför förtjänar att kommenteras lite extra, rör ledningssystem.

Som vi ser av vidstående ordboksdefinitioner står verbet **leda**, i betydelsen anföra, för att någon genom beslut och åtgärder samordnar och genom olika former av direktiv styr någon form av mänsklig aktivitet. Vad gäller ledning av militära verksamheter (eng. *Command and Control, C²*), så kan man uppfatta den som en process som omfattar en följd av aktiviteter, fr.a. insamling av information, tolkning av informationen, analys av hotbilden, beslutsfattande, ordergivning samt övervakning av genomförandet (i sin idealiserade form kallad OODA-loopen efter engelskans *Observe, Orient, Decide and Act*). Detta skall utföras i komplexa och dynamiska situationer och ofta under stor osäkerhet och tidspress.

För att klara dessa uppgifter behöver de som leder ett **ledningssystem**. Det kan ses som en samtrimmad organisation av människor (ofta räknas ledaren själv med i systemet) och teknisk utrustning, vilken handlar enligt en uttalad ledningsdoktrin. Sådana system behövs på alla nivåer från ÖB ner till den

43

Ur Svenska Akademins ordbok (1939):

Leda

BETYDELSE: som anförare l. chef o.d. styra l. dirigera (en grupp personer, ett affärsföretag o.d. resp. en av en grupp personer o.s.v. i gemenskap utförd verksamhet); vara anförare l. chef för (ngt); ofta i fråga om gymnastik, sång, dans, gudstjänst, förhandlingar o.d.; äv. bildl., med saksubj.

REDAKTIONSEXEMPEL: *Leda en armé, en trupp (till seger). Leda ett anfall, en strid. Leda ett affärsföretag, en expedition. Leda en orkester. Leda ett barns uppfostran. NN utsågs att som ordförande leda förhandlingarna. Leda samtalet, diskussionen.*

Ur Nationalencyklopedins ordbok:

leda verb *ledde lett,* pres. *leder* (1996)
ha bestämmande inflytande över organisation, verksamhet e.d. {se anföra}: *~ ett företag*; *projektet leddes av professor NN*; *en ung korpral ledde styrkan*.

anföra verb *anförde anfört,* pres. *anför* (1995)
fungera som ledare för ordnad grupp, t.ex. mil. styrka, procession el. kör; vanl. på ett synligt sätt {se leda}: *radiosymfonikerna, anförda av NN.*
BET.NYANS: utvidgat och överfört vara talesman för: *Unga Sverige, till en början anfört av NN.*
KONSTR.: *~ ngn* el. *ngt*
HIST.: sedan 1559; av ty. *anführen* 'föra fram; kommendera; citera'

enskilda stridspiloten eller arméplutonen, och systemen bör även kommunicera med varandra på ett smidigt sätt.

Ett bra ledningssystem skall förutom väl fungerande kommunikationsvägar, dokumenthantering, grafiska presentationsmöjligheter och andra administrativa funktioner erbjuda respektive befälhavare ett kraftfullt **beslutsstöd** avpassat för vars och ens ansvarsnivå. Stödet kan naturligtvis bestå av rådgivare i den stab som omger henne eller honom, men med hel- eller halvautomatiserade hjälpmedel kan man påtagligt höja den kvalitativa effektiviteten och kanske även i någon mån den kvantitativa prestationen i beslutsprocessen. Man måste dock komma ihåg att det här är fråga om ett stöd till och inte en ersättning för befälhavarens eget omdöme.

De olika stödformerna integreras lämpligen i ett sammanhållet **beslutsstödssystem** (eng. *Decision Support System, DSS*). Här kan man t.ex. finna olika typer av databaser med verktyg för databrytning (eng. *data mining*, d.v.s. automatisk sökning i stora datamängder efter mönster, som innebär ny kunskap), hypotesprövning och visualisering. Och här förekommer flera olika typer av modeller, så låt oss nu titta närmare på de viktigaste användningsområdena.

Modeller i ledningssystem

Stöd för informationstolkning

Indata till systemet kommer från olika sensorsystem, från rapporter m.m. Dessa skall sammanställas till en trovärdig lägesbild. Problemet är dock dels att informationsflödet kan vara mycket intensivt, dels att en hel del av informationen kan vara vag för att inte säga motsägelsefull. För en människa kan det därför vara mycket svårt att tolka och förstå vad informationen egentligen säger, i synnerhet som det normalt måste ske under mer eller mindre stor tidspress. Ett sätt att påtagligt öka effektiviteten i tolkningsprocessen är att genom informationsfusion automatisera vissa delar av stabsarbetet, fr.a. beträffande genereringen av en lägesbild.

Informationsfusion har sin bakgrund i datafusion, ett begrepp som började användas på 1980-talet. De första tillämpningarna bestod i att skapa en gemensam bild av data från flera likadana sensorer, t.ex. flera radarstationer som följer samma mål. Denna ansats vidgades snart till fusionering av data från sensorer av olika typer, t.ex. för effektivisering av navigationssystem. Nästa steg blev att tillämpa fusionsbegreppet på information i en vidare mening, sensordata blandat med rapporter av skiftande slag, och då etablerades termen informationsfusion.

Förutom att snabbare bygga upp lägesbilden bidrar informationsfusionstekniken till att göra det möjligt för befälhavare på olika nivåer att påverka varandras lägesbilder. Den högre befälhavarens intentioner sprids neråt i organisationen och påverkar bedömningsunderlaget där. Från de lägre nivåerna förs i gengäld information om aktuell status inom förbanden uppåt i hierarkin tillsammans med övrig insamlad information.

Informationsfusion innebär de facto en simulering i realtid av det aktuella händelseförloppet. Simuleringsmodellen baseras på all tillgänglig sensor- och rapporteringsinformation samt på den förhandskunskap man har om fiendens förbandssammansättning, utrustning, signaturer och beteende. Därtill lägger man relevant information om den omgivande miljön, egen underrättelseledning och tillgängliga sensorer.

Stöd för beslutsfattande

Då nu befälhavaren har en så tillförlitlig lägesbild som möjligt måste han eller hon fatta kloka beslut om vilka åtgärder som behöver vidtagas. Även då kan modeller integrerade i ledningssystemet ge värdefullt bistånd. En typ av sådana stödmodeller kan vara s.k. expertsystem, vilka reproducerar resonemangsprocessen hos skickligt fackfolk. Med detta bidrag från AI-området (artificiell intelligens) kan besluten fattas snabbare av en erfaren beslutsfattare och säkrare av en mindre erfaren. Fördelen med en sådan modell är att man inte glömmer bort någon relevant information, då man fattar sitt beslut. Nackdelen med att alltför mycket förlita sig på ett expertsystem är dock att ens beteende kan bli lite stereotypt och förutsägbart, eftersom det inte i sig promoverar nytänkande. Om man ger modellen återkoppling om utfallet av en rekommendation, så att den successivt kan ”lära sig”, så kan man till en viss del motverka denna nackdel.

Minst lika användbara som expertsystem är modeller med vars hjälp beslutsfattaren kan simulera effekten av olika beslut innan han eller hon måste ta definitiv ställning. På en sådan simuleringsmodell måste man ställa ganska omfattande krav. För det första måste den hämta och förstå data som beskrivs i lägesbilden. För det andra måste den ha en hög grad av trovärdighet, d.v.s. den måste ha genomgått en långt driven VV&A-process. För det tredje måste den vara snabb, mycket snabbare än realtid, vilket kräver en förhållandevis hög grad av abstraktion. För det fjärde måste den ha ett mycket flexibelt men också lättanvänt gränssnitt mot användaren, för att befälhavaren snabbt och enkelt skall kunna testa mer eller mindre okonventionella lösningar. Dessutom fordras av befälhavaren att han eller hon förstår modellen väl.

Mycket av beslutsstödsmodellerna måste som synes fungera integrerat i den löpande processen: upprättande av lägesbild, identifiering av objekt, utvärdering av alternativ, taktisk bedömning m.m. Man kan också tänka sig, speciellt på lite högre beslutsnivåer, att operationsanalytiker, som arbetar mera långsiktigt, kan utföra en hel del analysarbete lite mera oberoende av det sekundaktuella förloppet. Naturligtvis innebär det att kraven på deras modeller blir lite annorlunda. Å andra sidan kan de ha behov av simuleringsverktyg som Matlab [Etter, 2003] eller Extend [Law & Kelton, 2000, sid. 219ff] för att snabbt konstruera nya enkla modeller.

På lägre beslutsnivåer, t.ex. för enskilda flygförare, är beslutstiderna mycket korta men beslutsalternativen färre och enklare. Det ger möjlighet till en ökad automatisering, d.v.s. modellerna är inte längre enbart stöd till en mänsklig beslutsfattare utan de samverkar direkt med systemet. Man talar här om inbäddade modeller, d.v.s. simuleringsmodeller som utgör autonoma delar i ett fysiskt system. Ett enkelt och välkänt exempel är en autopilot, som under vissa givna betingelser och utan mänskligt deltagande förmår ge själva flygplanet sådana styrsignaler att flygningen kan genomföras på ett funktionellt och säkert sätt. Principiellt fungerar en sådan modell så, att den kontinuerligt tar emot och bearbetar inkommande sensordata, resonerar som ett expertsystem, utför kanske simuleringar av några svårbedömda alternativa lösningar, väljer lämplig åtgärd samt omformar den till styrsignaler till det fysiska systemet. Vi kan alltså säga, att vi här har ett mycket avancerat reglersystem.

Inbäddade system kan naturligtvis fungera mycket bra under lågintensiva delar av ett uppdrag, men dess största styrka är nog egentligen när situationen är mycket dynamisk. Då kan inflödet av data vara extremt stort och göra en människa stressad och förvirrad, medan modellen snabbare och utan att utmattas kan initiera tillräckligt förnuftiga åtgärder. Men för säkerhets skull är det självklart viktigt för en mänsklig beslutsfattare att kunna ta ifrån modellen kommandot om något är på väg att gå snett eller åtminstone i oönskad riktning. Modellen måste alltså vara brytbar, väl validerad och snabb.

Arkitekturer och ontologier

Som vi sett är ett ledningssystem oftast sammansatt av många delar. I militära ledningssystem ingår vanligen inte bara tekniska produkter utan de inkluderar även doktrinära, organisatoriska och personella aspekter. Skall allt detta fungera som en helhet behöver man en välgenomtänkt arkitektur som beskriver hur delarna skall hänga ihop och samverka; varje komponent behöver utvecklas enligt givna designregler (enl. terminologin i försvarsmaktens LedsystT-arbete). En arkitektur av detta slag måste vara mycket flexibel för att medge ständigt växande och anpassade system.

I ett modellbaserat ledningssystem, sådant som vi här ovan antytt det, måste även modellerna ses som likaberättigade systemdelar tillsammans med sensorer, kommunikationsutrustning, presentationsutrustning, organisation o.s.v. Som vi antydde i kapitlet om arkitektur (sid. 85), så uppstår då ett möte mellan två traditioner utvecklade inom lite olika kulturer, och som därför inte passar så bra ihop. Medan M&S-kulturen närmar sig interoperabilitetsproblemet utifrån sin HLA-tradition, så närmar sig ledningskulturen utifrån sin tradition med rapporter och ordergivning. En arkitektur baserad huvudsakligen på textmeddelanden, i många fall i naturligt språk, lämpar sig illa för automatisk kommunikation mellan olika enheter inom systemet.

I stället bör ledningssystemen byggas som objektmodeller, där varje komponent inte bara skall ingå fysiskt utan även i form av en motsvarande delmodell. Alla fysiska resurser och all relevant information av betydelse för ledningsfunktionens utövande kan då representeras i en distribuerad realtidsdatabas med en gemensam informationsmodell. Som vi ser av definitionen i faktaruta 44 är informationsmodell väsentligen detsamma som konceptuell modell. Det förra är dock gängse språkbruk då man beskriver systemkommunikation medan det senare, som vi sett, hör hemma i M&S-världen.

44

Ur Wikipedia (2006):

Information model

An information model is an abstract but formal representation of entities including their properties, relationships and the operations that can be performed on them.

The entities being modeled may be real-world, such as devices on a network, or they may themselves be abstract, such as the entities used in a billing system. Typically, though, they are used to model a constrained domain that can be completely described by a closed set of entities, properties, relationships and operations.

The main driving force behind the definition of an information model is to provide formalism to the description of a problem domain without constraining how that description is mapped to an actual implementation in software. There may be many mappings of the information model. Such mappings are called Data Models irrespective of whether they are Object Models (e.g. using UML), Entity Relationship Models or XML schemas.

45

Ur Svenska Akademins ordbok (1950):

Ontologi

BETYDELSE: läran om det varandes väsen o. tingens allmännaste väsentliga bestämningar o. sammanhang; den del av den teoretiska filosofien som behandlar det varande ss. sådant.

Tom Gruber (1993):

”In the context of knowledge sharing, I use the term **ontology** to mean a *specification of a conceptualization*. That is, an ontology is a description (like a formal specification of a program) of the concepts and relationships that can exist for an agent or a community of agents. This definition is consistent with the usage of ontology as set-of-concept-definitions, but more general. And it is certainly a different sense of the word than its use in philosophy.

What is important is what an ontology is *for*. My colleagues and I have been designing ontologies for the purpose of enabling knowledge sharing and reuse. In that context, an ontology is a specification used for making ontological commitments. The formal definition of ontological commitment is given below. For pragmatic reasons, we choose to write an ontology as a set of definitions of formal vocabulary. Although this isn’t the only way to specify a conceptualization, it has some nice properties for knowledge sharing among AI software (e.g., semantics independent of reader and context). Practically, an ontological commitment is an agreement to use a vocabulary (i.e., ask queries and make assertions) in a way that is consistent (but not complete) with respect to the theory specified by an ontology. We build agents that commit to ontologies. We design ontologies so we can share knowledge with and among these agents.”

Ur [Gruber, 2006]

Även om man skulle komma överens om en lösning i linje med den ovan beskrivna, är det dock mycket långt kvar innan existerande delsystem kan tänkas ha ändrats och anpassats för att uppnå önskvärd interoperabilitet efter en sådan linje. I stället för att utveckla en ny standard, som skall åläggas alla producenter av de olika komponenterna i ledningssystemet, har det föreslagits att man skall låta var och en så långt som möjligt behålla de standarder som de är nöjda med. Vad som behövs är då att befintliga arkitekturer utökas med ett gemensamt gränssnittstillägg.

Denna överbyggnad, som till sin idé i hög grad anknyter till MDA (*Model Driven Architecture*), som vi stiftade bekantskap med i arkitekturkapitlet (sid. 91-92), förutsätter att man utvecklar en metamodell. En sådan innebär i sin tur en standardisering av alla adekvata begrepp som förekommer i de olika delsystemen och i deras arkitekturer, d.v.s. ett gemensamt språk. Mot denna metamodell skall alltså varje annan informationsmodell för delsystemen (M&S-komponenterna inbegripna) ha en entydig mappning, d.v.s. det gemensamma gränssnittet.

I själva verket är metamodellen en formaliserad ontologi för ledningssystem, och man hör ofta detta begrepp användas för denna högsta modellnivå. Med utgångspunkt från Tom Grubers definition här intill, kan man konstatera att även ontologi praktiskt taget är en synonym till konceptuell modell. Förslagsvis bör kanske detta begrepp helst användas för mycket övergripande konceptuella modeller, närmast vokabulärer, medan man annars kan använda begreppet informationsmodell eller konceptuell modell.

Mycket talar för att det är detta senare alternativ med en hierarkisk överbyggnad på befintliga standarder och inte utveckling av en ny gemensam standard som kommer att bli riktlinjen för framtida ledningssystemutveckling, men vägen dit förefaller ännu vara mycket lång och mödosam att vandra.

Modeller av ledningssystem

Ett ledningssystem innehåller som vi sett modeller av skilda slag. Men ett ledningssystem kan naturligtvis också låta sig modelleras, precis som alla andra system. Exempelvis kan man föreställa sig en ledningssystemssimulator för utbildning och träning av den personal, som i ett skarpt läge skall kunna dra nytta av det verkliga systemet. Det förmodligen vanligaste sättet att åstadkomma detta är att utnyttja systemet självt men med simulerade stimuli. Därigenom uppnår man en mycket hög grad av verklighetskänsla i övningarna och dessutom kan dödtid i ordinarie tjänst utnyttjas till träning.

Det är naturligtvis även möjligt att utveckla en fristående ledningssystemssimulator, som innehåller en mer eller mindre detaljerad modell av systemet med en simulerad omgivning. Beroende på detaljeringsgrad kan de modeller som verkligen skall ingå i det studerade systemet användas även i modellen. Annars bör de lämpligen representeras på en högre abstraktionsnivå.

Sådana virtuella ledningssystem kan också användas för planering, studier och forskning, men i dessa fall kan det ofta vara befogat att modellera hela supersystemet, där även en modell av befälhavaren ingår som en komponent. Vi får då en utvecklad (eng. *constructive*, se sid. 13) simulering, vilket ofta är att föredra i studiesammanhang, då den inte fordrar en människa som aktiv aktör i förloppet.

Det utmärkande för den senare typen av simuleringsmodeller är att de utifrån en given målbild, doktrin, strategi samt signaler från en (simulerad) omvärld fattar ”beslut” om lämpliga åtgärder på ett mer eller mindre mänskligt sätt. Sådana modeller kan även med fördel användas som komponenter i mer omfattande systemmodeller, t.ex. då man i en simulator vill representera fientliga aktörer och egna stödförband med ett trovärdigt beteende. Vi talar då om **datorgenererade förband** (CGF:er, eng. *Computer Generated Forces*). Sådan delmodeller används även för att beskriva de förband som står i fokus i analysmodeller av sluten typ, där operatörsingripanden inte är möjliga under gång. Detta är det vanliga vid Monte Carlo-simuleringar av t.ex. stridsförlopp.

44

Ur Space & Electronic Warfare Lexicon (2006):

COMPUTER GENERATED FORCE (CGF) –

A computer system that generates agents in a military simulation; it can be used with trainees in simulators fighting virtual battles, or can be run autonomously for planning of tactics.

AGENT – (1) In virtual reality: a software program that can carry out fairly sophisticated tasks in unknown networked environments without human intervention; in other words a ”smart” bot.

(2) A software component that performs one or more common tasks by acting in a preset manner. Agents may be classified as to characteristics (mobility – stationary or mobile); response method (deliberative or reactive); autonomy; learning

M&S i studier och forskning

Vi har i denna skrift inte gått närmare in på de mer konkreta modelleringstekniker som finns utan hållit oss till de mera allmängiltiga metoder som är generellt användbara, speciellt vid arbete med systemmodeller.

Detta är en naturlig begränsning, eftersom det finns utmärkta läroböcker som ger en god insikt i de flesta av de tekniker som tillämpas. Dessutom får man vanligen vid studier i ett speciellt ämne på köpet den för området gängse metodiken och tekniken för att modellera denna speciella kunskap.

Hur modeller kommer in i studier för långsiktig planering har vi redan i viss mån berört i kapitlet om krigsspel (sid. 97). Den militära studieverksamheten handlar i allt väsentligt om att ta fram underlag för långsiktiga beslut. Det gäller perspektivstudier, funktionsstudier, system- och förbandsstudier. Här är (dator-) modellerna endast en liten del av arbetet och kan närmast ses som ett sätt att strukturera befintlig kunskap. Det går rätt bra att utföra studierna utan dessa modeller, däremot skulle det vara omöjligt att låta modellerna ensamma göra jobbet. Det väsentliga är själva processen med tänkande och diskussion, varför organisation och personalens kompetens och kreativitet spelar en betydligt större roll. De modeller som används är, som vi såg, fr.a. sådana som stöder krigsförloppsspelen, men även översiktliga ekonomiska modeller förekommer.

I natur- och samhällsvetenskaplig forskning utnyttjas simuleringsmodeller som ett centralt hjälpmedel. De olika ämnesområdenas karaktär återspeglas då gärna i att man utnyttjar olika modelleringstekniker. Exempelvis är jämviktsmodeller (eng. *constraint models*) vanliga inom fysiken, variabelbaserade funktionsmodeller i kemin, cellulära automater i biologiska och ekologiska vetenskaper eller stokastiska deklarativa modeller i ekonomin. De fungerar alla väl för sina ändamål, men kan ibland bidra till att representanter för olika ämnesdiscipliner får svårt att förstå och uppskatta varandras metoder. Att beskriva alla förekommande modelltyper vore dock att gå långt utöver ramen för detta arbete. Här nöjer vi oss med en översiktlig beskrivning av de vanligaste modelleringsteknikerna. Den som vill studera dem närmare kan göra det i de utmärkta läroböcker som finns i ämnet, t.ex. i [Fishwick, 1995].

När man betraktar de olika teknikerna kan man urskilja fyra mera renodlade typer, vardera

med två undertyper. För systemmodeller gäller dock ofta att ingen av dessa ensam är tillfyllest för att ge oss en välfungerande och transparent modell, utan här är det i allmänhet fråga om att välja en för de olika delmodellerna lämplig mix.

Deklarativa modeller

Den modellerade världen betraktas här fr.a. som en följd av tillståndsförändringar, och där modellerandet väsentligen går ut på att föreskriva regler för hur man förflyttar sig mellan olika möjliga tillstånd. De primära modellkomponenterna är därför (ett uppräkningsbart antal) tillstånd och händelser. Det finns så att säga en linjär ordning i tillståndsrymden

Indelningen i undergrupper bygger i detta fall på om det är tillstånden eller händelserna som står i fokus.

Tillståndsbaserade deklarativa modeller brukar vanligen beskrivas som *ändliga tillståndsautomater* (se exempel i bilaga 2). Om man vill beskriva exempelvis en fyrtaktsmotor eller ett trafikljussystem är det mycket lämpligt att göra med sådana modeller. Om tillståndsövergångarna är slumpartade enligt bestämda fördelningsfunktioner som i radioaktiva processer modelleras detta vanligen enligt teorin för *Markov-processer*. I det enklaste fallet av dessa tillståndsautomater föreskriver man för varje enskilt tillstånd hur man tar sig till nästa tillstånd, men detta är naturligtvis mycket opraktiskt då tillståndsrymden är stor, vilken den naturligtvis lätt blir då de modellerade systemen blir lite mer komplexa. Då utnyttjar man i stället produktionsregler med vars hjälp man kan föreskriva övergångarna för hela grupper av tillstånd. Dessa regler uttrycks som generella funktioner, vars algoritmer kan beskrivas med *predikatlogik*.

Händelsebaserade deklarativa modeller skiljer sig från de föregående så till vida att fokus ligger på händelser i stället för tillstånd. Vi får då det som vi tidigare kallat händelsestyrda simuleringar. I de enkla fallen kan de jämföras med script för en animerad filmsekvens, i de mer avancerade genererar vissa händelser schemaläggning av nya händelser i framtiden (motsvarande produktionsbaserad modellering i det tillståndsbaserade fallet). Naturligtvis kan man även här tänka sig både deterministiska och stokastiska alternativ.

Det finns även hybridformer, t.ex. *Petrinät*, där tillstånd och händelser hanteras som jämbördiga storheter.

Tjänstebaserade modeller

Om den modellerade världen karaktäriseras av klart åtskilda unika fysiska objekt som är logiskt sammanlänkade i en tydlig tidsordning faller det sig ofta naturligt att utforma modellen med fokus på funktioner och variabler. Vi kan dra oss till minnes att i objektorienterad mening så består objekt av operationer och attribut, vilka ju direkt svarar mot funktioner och variabler, som kan utnyttjas för att utföra tjänster åt varandra. I dessa modeller finns en explicit riktning i den meningen att de kan uppfattas som ett flöde från den ena funktionen till den andra.

De två underklasserna till tjänstemodeller anknyter till om tonvikten ligger på funktioner eller variabler.

En **funktionsbaserad tjänstemodell** karaktäriseras av att funktionerna, som transformerar indata till utdata utgör modellens komponenter. Ett bra exempel är en modell av ett servicesystem som en bank med flera funktioner och betjäningsstationer, eller en biltvättanläggning som exemplifierades i faktaruta 9. Även

modeller som ligger till grund för reglersystem, t.ex. en farthållare i en bil, modelleras ofta på detta sätt.

I en **variabelbaserad tjänstemodell** är å andra sidan funktionsaspekten nedtonad och det är i stället de tillståndsbeskrivande variablerna som utgör de modellbyggande komponenterna. I allmänhet är en speciell funktionsstruktur underförstådd, fr.a. när det gäller linjära system. Modeller av kemiska reaktioner är vanligen av denna karaktär, eftersom de fr.a. behandlar mängder av olika ämnen (d.v.s. tillståndsvariablerna) och hur dessa förändras över tiden. Ett annat exempel är flödesgrafer där man endast är intresserad av nivåerna i de olika ackumulatorerna.

Spatiala modeller

Detta är modeller som är användbara när vi kan betrakta vår värld som sammansatt av många små och likartade delar, och när vi därtill har anledning att tro, att om vi känner beteendet hos de enskilda delarna, så kan vi också förstå beteendet hos det betraktade systemet som helhet. Ofta innebär spatial modellering att man arbetar mycket detaljerat.

Även i detta fall kan vi finna två undertyper: rumsbaserade resp. entitetsbaserade.

I **rumsbaserade spatiala modeller** är det själva rummet som delas upp i regelbundna smådelar. Det är då dessa delars egenskaper som successivt ändrar egenskaper, d.v.s. innehåll och kanske t.o.m. form, från operation till operation. Då enbart innehållet förändras talar vi ofta om *cellulära automater*, t.ex. det välkända *Game of Life* (faktaruta 47). Ett exempel på när även formen ändras är *finita elementmodeller* (FEM), som används fr.a. i hållfasthetsberäkningar.

I **entitetsbaserade spatiala modeller** är det i stället objekten som sådana och deras dynamik som står i fokus mer än själva rummet. Objekten inkluderar därför egenskaper som position och rörelseriktning. Typiska exempel är gravitationsstyrda mångkropparsproblem eller *random walk*-tillämpningar. En annan tillämpning brukar vara simulering av växters tillväxt genom successiv förgrening. Fraktala modeller hör också ofta till denna kategori, men kan lika gärna vara rumsbaserade.

Jämviktsmodeller

(eller kanske **bundna modeller**, eng. *constraint models*), tar man till för att t.ex. beskriva naturlagarna. Här ser man världen som ett system med balans mellan olika företeelser och med vissa kvantiteter (t.ex. energin) bevarade. Till skillnad mot deklarativa och tjänstebaserade modeller indikerar kopplingen mellan modellkomponenterna ingen tidsmässig à priori-riktning.

Här urskiljer man också två undertyper: ekvationsbaserade resp. grafbaserade.

Ekvationsbaserade jämviktsmodeller uttrycks i differensekvationer, då tillståndsförändrinarna sker över diskreta tidsintervall. Ett typiskt exempel är modellering av naturlig tillväxt i populationer av växter eller djur. Sker däremot förändringarna kontinuerligt över tiden uttrycks dessa modeller i stället i form av differentialekvationer. Oerhört många fysiska fenomen låter sig som bekant beskrivas på detta sätt, alltifrån harmoniska svängningar till kaossystem.

Grafbaserade jämviktsmodeller kan exemplifieras med elektriska kretsscheman, eller mer generellt med s.k. *Bond graphs* (faktaruta 48), där man med ett slutet schema kan beskriva t.ex. anspänningen (eng. *effort*, t.ex. spänning i elektriska grafer) och flödet

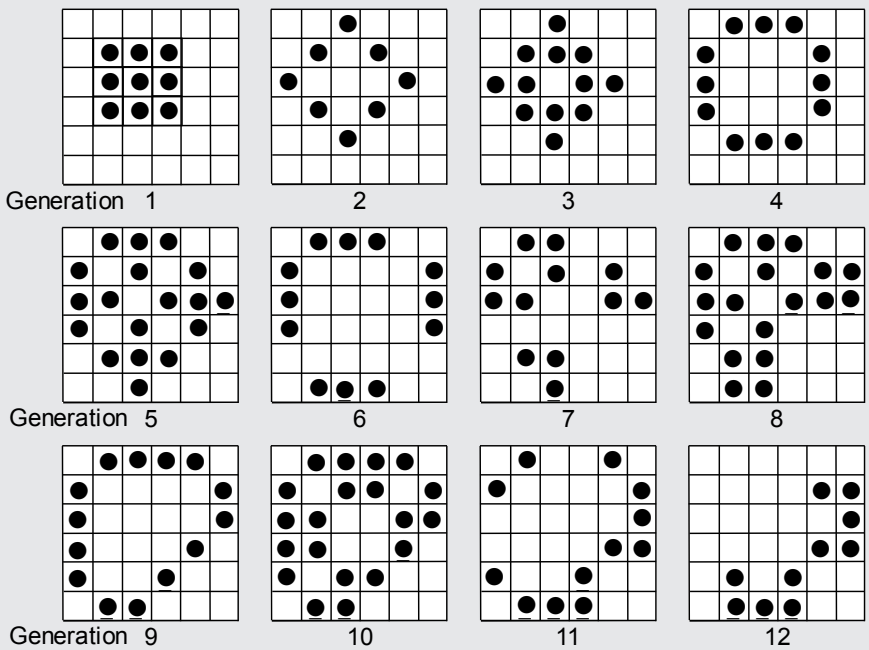
47

Exempel på cellulär automat – Game of life

Denna simuleringsmodell utvecklades omkring 1970 av John Conway, matematiker vid Cambridge universitet. Den består av en rektangulär matris av celler, i vårt exempel 6 x 6 stycken, där varje cell påverkas av alla sina åtta närmaste grannar. En cell lever, dör och kommer till liv igen beroende på tre enkla regler:

1. Varje levande cell med högst en levande granne dör av ensamhet.
2. Varje levande cell med minst fyra levande grannar dör av överbefolkning.
3. Varje död cell med tre levande grannar blir levande.

Utgångsmönstret utgör den första generationen av systemet. Den andra generationen skapas genom att tillämpa de tre reglerna på varje cell i den första generationen.



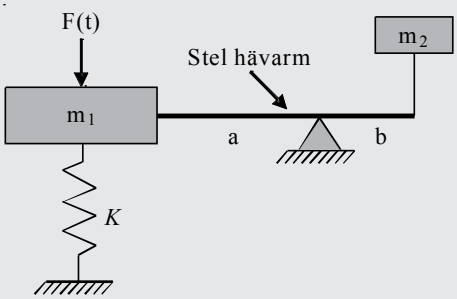
(eng. *flow*, t.ex. strömstyrkan i elektriska grafer) vid varje nod. Men genom sin generella natur lämpar sig sådana modeller även för exempelvis mekaniska, hydrauliska och termiska system likaväl som för kombinationer av komponenter av flera olika slag. Grafspråket kan genom standardiserade transformationer konverteras till exekverbar kod.

Forskning om M&S-metodik

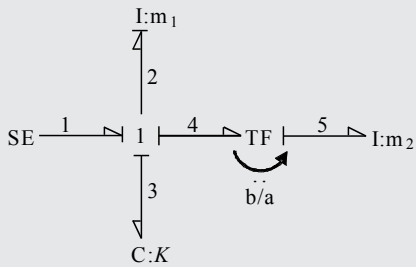
Nu är det ju inte bara så att modellering och simulering är ett hjälpmedel i forskningen inom allehanda ämnesdiscipliner. För att stödet som ges skall bli så effektivt som möjligt fordras också att hjälpmedlet i sig utvecklas, anpassas och förfinas. Kort sagt, det behövs forskning även inom ämneskategorin modellering och simulering för att skapa bättre förutsättningar för utveckling och användning av modeller.

Exempel på grafbaserade jämviktsmodeller – Bond graphs

1. En mekanisk modell, t.ex. en enkel bromspedal, kan beskrivas med följande stiliserade figur:



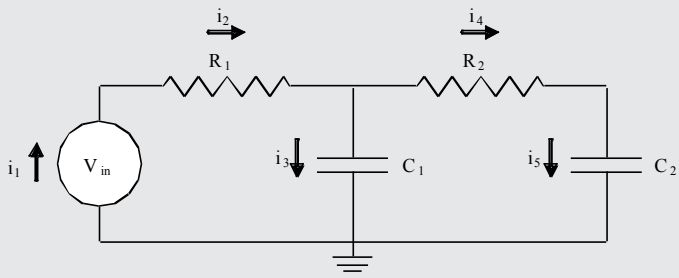
Uttryckt som en Bond graph-modell får den följande utseende



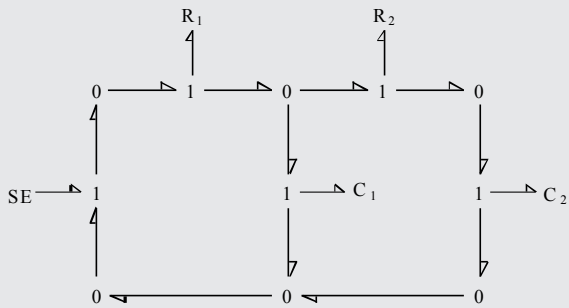
SE = *Effort source* (den anbringade kraften)
C = *Capacitive storage element* (fjädern)
TF = *Transformer* (hävarmen)
I = *Inertance storage element* (vikterna)

Förgreningspunkterna kan vara s.k. *o-junctions* eller *1-junctions*. I de förra summeras flödena till noll, i de senare summeras krafterna till noll.

2. En elektrisk modell, exempelvis



kan på samma sätt uttryckas som en Bond graph



Å ena sidan gäller det alltså att finna metoder som garanterar en hög och jämn kvalitet i modellerna, å den andra att modellutveckling och simulering kan ske så kostnadseffektivt som möjligt.

Forskningen inom detta område är till sin natur i huvudsak tillämpningsdriven. Nya behov, t.ex. simulering av mer komplexa och heterogena system, och frågeställningar som behöver lösas leder till nya krav på metoder, tekniker och verktyg. Ett skolexempel är historien om hur distribuerad simulering utvecklades (se sid. 86-87). Detta exempel visar också hur lösningen till ett konkret problem i en bestämd tillämpning genom fortsatt forskning generaliserades, så att metoden blev användbar för ett mycket bredare spektrum av tillämpningar. Detta är kännetecknande för M&S-forskningen, att gå från det specifika till det generella. Man kan jämföra med matematiken, där problem inom tillämpad matematik kan ge upphov till en ny teoribildning inom den ”rena” matematiken.

Medan tillämpningsorienterad problemlösning förekommer överallt där modellering och simulering utövas, så bedrivs därutöver M&S-forskning i denna generaliserande mening fr.a. vid universitet och högskolor, inom vissa företag och inte minst inom försvarsorganisationerna världen över. Inom Sveriges försvar är det främst Totalförsvarets forskningsinstitut (FOI), men även i viss mån Försvarshögskolan (FHS), Försvarets Materielverk (FMV) och försvarsindustrin, som står för denna verksamhet.

För att ge en uppfattning om metodforskningens inriktning och betydelse finns som exempel en översiktlig presentation i bilaga 4 av de forskningsprojekt som bedrivits 1996 - 2007 vid FOI inom försvarsmaktens forsknings- och teknikutvecklingsprogram (FoT). Målet med just denna forskning är att generera kunskap om sådana generella metoder och tekniker som behövs för utveckling och användning av militära modeller på ett kostnadseffektivt sätt. Därigenom bereds försvarsmakten möjlighet att definiera den metodmässiga och tekniska basen för hela sin M&S-användning. Forskningsresultaten är alltså en viktig förutsättning dels för den eftersträvade gemensamma modellarkitekturen och infrastrukturen, dels för att man inom all försvarsrelaterad verksamhet skall ha en realistisk förståelse av möjligheter och begränsningar med M&S.

Sammanfattar vi forskningsinsatserna enligt förteckningen i bilagan, kan vi se att de berör ett ganska brett spektrum kring några centrala teman. För det första studeras hur modeller förhåller sig till varandra och hur de skall kunna samarbeta sinsemellan och med andra system. För det andra utforskas arkitekturers och infrastrukturers möjligheter och begränsningar. Som ett tredje forskningsområde framträder konceptuell modellering med dithörande problem med inhämtande och representation av kunskap. Verifiering och validering är det fjärde framträdande temat. Femte temat handlar om miljömodellering med tillhörande dataförsörjning och datamodifiering. Slutligen finns några mer tillämpningsorienterade projekt (rörande träning och utbildning samt modellering av mänskligt beteende), som emellertid har stor potentiell betydelse inom betydligt vidare områden.

Litteratur

Fördjupningslitteratur

Banks, Jerry (ed.): *Handbook of Simulation. Principles, Methodology, Advances, Applications, and Practice*. John Wiley & Sons, Inc. (1998)

Bergsten, Ulla: *Ett riskperspektiv på verifiering, validering och ackreditering av simuleringsmodeller*. FOI-R--0738--SE december 2002

Blilie, Charles: *The Promise and Limits of Computer Modeling*. World Scientific (2007)

Carlsson, Christer: *Om krigsspel*. Ingår i serien ”Spel för ökad kunskap”. Försvarets Krigsspelscentrum 1995:02

Dockery, J T & Woodcock, A E R: *The Military Landscape. Mathematical Models of Combat*. Woodhead Publishing Limited (1993)

Dreborg, Karl-Henrik: *Spela för att lära. Om spel och övningar i civil beredskap*. FOA rapport C 10356-1.2 december 1993

Fjällström, Per-Olof; Neider, Göran; Persson, Mats; Risch, Tore & Svensson, Per: *Arkitekturprinciper för informationsöverlägsenhet i framtidens ledningsstödsystem*. FOA-rapport FOA-R--00-01435-505--SE, februari 2000

FOA-tidningen nr 1, februari 1998 (temanummer om modellering och simulering)

Fishwick, Paul A: *Simulation Model Design and Execution. Building Digital Worlds.* Prentice Hall (1995)

Fowler, Martin: *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Third Edition. Addison-Wesley (2005)

Holm, Gunnar & Moradi, Farshad: *Distribuerad interaktiv simulering och värderingsmodeller*. FOA-rapport FOA-R--97-00496-202--SE, maj 1997

Law, Averill M & Kelton, W David: *Simulation Modeling and Analysis*. Third edition. McGraw-Hill international series (2000)

Meyer, Bertrand: *Object-oriented Software Construction*. Prentice Hall (1988)

Mojtahed, Vahid; García Lozano, Marianela; Svan, Pernilla; Andersson, Birger & Kabilan, Vandana: *DCMF – Defence Conceptual Modelling Framework*. FOI-R--1754--SE november 2005

Övrig refererad litteratur

Bergsten, Ulla; André, Tommy & Sporre, Lena: *Verifiering, validering och ackreditering av simuleringsmodeller*. http://www.va.foi.se/reports/VV&A-RapportInternet.pdf (2001)

Bonniers lexikon, band 10 (1965), band 12 (1966). AB Nordiska Uppslagsböcker

Eklöf, Martin; García Lozano, Marianela; Moradi, Farshad; Nordqvist, Dan & Ulriksson, Jenny: *Nätverksbaserad modellering och simulering – prototypen*. FOI-R--1767--SE oktober 2005

Etter, Delores M: *Introduction to MATLAB for Engineers and Scientists*. Prentice Hall (2003)

Gruber, Tom: *What is an Ontology?* http://www-ksl.stanford.edu/kst/what-is-an-ontology.html (2006)

Hansson, MajBritt: *Erfarenheter från användningen av värderingsmodellen SILVIA*. FOA-R--97-00559-202--SE oktober 1997

Mojtahed, Vahid; García Lozano, Marianela; Ulriksson, Jenny & Lundgren, Mårten: *Analys av CMMS-konceptet med fokus på Kunskapsanskaffning (KA) och Kunskapsbearbetning (KE)*. FOI-R--1049--SE december 2003

Nationalencyklopedin, band 1 (1989), band 11 (1993), band 16 (1995), band 18 (1995). Bokförlaget Bra Böcker

Nationalencyklopedins ordbok, band 1 (1995), band 2 (1996), band 3 (1996). Bokförlaget Bra Böcker

Oxford English Dictionary. http://dictionary.oed.com/entrance.dt (2005)

*Space & Electronic Warfare Lexicon***.** http://www.sew-lexicon.com/index.html (2006)

Svenska Akademins ordbok, band 6 (1908), band 9 (1926), band 15 (1937-39), band 16 (1942), band 17 (1945), band 19 (1950), band 24 (1965), band 25 (1967), band 33 (2002). http://g3.spraakdata.gu.se/saob/ (2006)

Wikipedia. http://en.wikipedia.org/wiki/Information_model (2006)

Exempel på en enkel konceptuell modell
Bombning av ett bostadsområde

Låt oss anta att vi skall skapa en modell för att kunna simulera effekterna på byggnader och människor vid bombning av ett bostadsområde.

Ett första steg blir då att företrädare för olika kompetenser kommer samman och skapar en konceptuell modell av fenomenet som sådant. Syftet i denna tidiga utvecklingsfas är alltså att nå fram till en gemensam förståelse för det problemområde som skall fokuseras. I detta skede är det lämpligt med en informell konceptuell modell i ett ”naturligt” språk. Den skall beskriva vilka objekt och interaktioner som kan vara aktuella samt definiera de begrepp som kommer till användning. Vi kan då tänka oss att man kommer fram till något liknande följande konceptuella modell.

Modellerade objekt	**Modellerade interaktioner**		
Marken	Stridsdel	↔	Mark
Byggnader	Stridsdel	↔	Byggnad
Människor	Skadad byggnad	↔	Skadad byggnad
Stridsdelar	Stridsdel	→	Människa
	Skadad byggnad	→	Människa

(→ betecknar ensidig påverkan och ↔ ömsesidig påverkan)

Objektet mark

har följande egenskaper:

1. Det är plant och horisontellt över hela spelområdet.
2. Det har konstant geologi över hela spelområdet.
3. Dess motståndsförmåga mot inträngande stridsdelar beskrivs enl. [FortH 1, 1987], t.ex. som i grov sand eller grusbemängd åkerjord.
4. Dess tryckförmedlande förmåga beskrivs enl. [Kristiansen et al., 1973], t.ex. som i grushaltig jord.

Objektet byggnad

har följande egenskaper:

1. Dess basyta har en kontur i form av 1 - 3 sammanhängande rektanglar i ett vinkelrätt arrangemang: I-form, L-form eller U-form.
2. Det har ett antal våningar ovan jord, minst 1.
3. Det kan ha eller sakna källare. Om det har källare antas denna ha endast en våning vars höjd är densamma som våningshöjden.
4. Det kan ha eller sakna vind.
5. Dess konstruktionsklass anges med en typkod. Byggnaden får en schablonindelning genom att motsvarande typhus' egenskaper anpassas till den aktuella byggnadens dimensioner.
6. Det har ett antal stomenheter (område inom ett våningsplan som begränsas av bärande väggar) längs långsidan resp. kortsidan i varje byggnadsdel (= rektangel enl. punkt 1).
7. Dess byggnadsmaterial i bjälklag och bärande väggar antas genomgående vara betong med konstant kubhållfasthet. I den mån det verkliga byggnadsmaterialet är annat, beaktas detta genom en modifiering av väggtjockleken.

Följande typhuskoder förekommer:

B3MN Platsgjutet betonghus med bärande tvärväggar och utfackade längsgående ytterväggar.
B4TN Platsgjutet cellhus i betong.
M1LS Småhus med murad stomme.
M1TN Murat stenhus med betongbjälklag.
M2TN Murat stenhus med fyllningsbjälklag.
P2TN Monterat elementhus i betong.
T1MN Plank- och timmerhus.
T3MS Småhus med träregelstomme.

Motsvarande typhus karaktäriseras av följande egenskaper:

1. Typisk längd och bredd.
2. Typisk våningshöjd.
3. Typisk vindshöjd.
4. Typisk sockelhöjd.
5. Typisk vägg- och bjälklagstjocklek.
6. Typisk andel av fasad- resp. gavelytan som utgörs av fönster.
7. Typiskt antal stomenheter.
8. Sårbarhet avseende olika skadetyper (golvras, väggras, sprickbildning, sättningar, glaskross) på grund av tryck- och impulsverkan.
9. En uppräkningsordning för stomenheter som anger hur eventuella skador fördelar sig över ett våningsplan.
10. Antal bärande väggar som behöver raseras i en stomenhet för att ovanliggande golvbjälklag skall kollapsa.

Objektet människa

har följande egenskaper:

1. Det har en position i markplanet, som kan vara utomhus eller inomhus. I det senare fallet har det även en "adress", som identifierar byggnad, våningsplan och stomenhet.
2. Det står alltid upp.
3. Det har en sårbarhet för splitter enl. [Ahlers, 1969] med kriteriet för buk och lemmar utvidgat att gälla för hela kroppen.
4. Det har en sårbarhet för direkt övertryck vid markdetonationer med fri utbredning enl. [White et al., 1971].

Objektet stridsdel

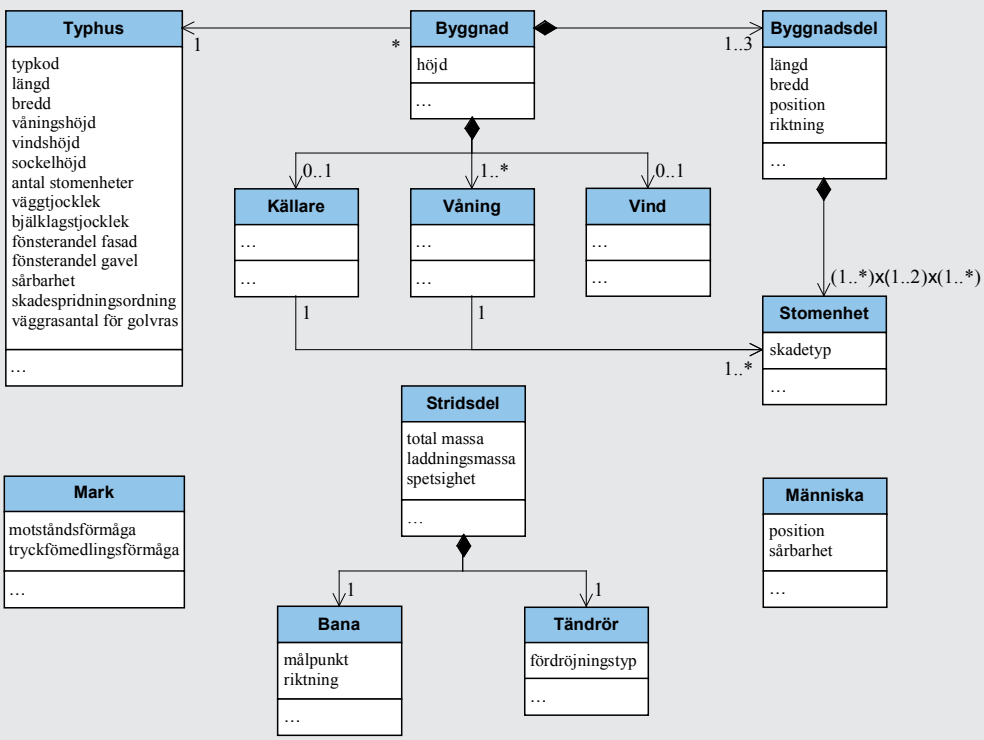
har följande egenskaper:

1. Det har en rätlinjig bana definierad av en punkt i markplanet och en hastighetsvektor, som i och för sig kan vara $\vec{0}$.
2. Det har en given totalmassa.
3. Det har en laddning med given massa.
4. Det har en nos som är mer eller mindre spetsig i enlighet med Petrys formel för markinträngning i [FortH 1, 1987].
5. Det har en viss tröghet vid inträngning i byggnadsmaterialet enl. [Bergman, 1952].
6. Som källa till tryckvågor antas det sakna utsträckning.
7. Det genererar dödande resp. sårande splitter enl. [Lindqvist et al., 1991].
8. Det har ett tändrör med anslagsfunktion med olika grad av fördröjning, vilken anges i tre klasser:

   1 => Ingen fördröjning, d.v.s. brisaden sker vid första kontakt med byggnad eller mark, varför ingen inträngning sker.
   2 => Kort fördröjning, d.v.s. brisaden utlöses efter genomslag av ett bjälklag eller en yttervägg resp. stridsdelen hinner tränga ner i marken.
   3 => Lång fördröjning, d.v.s. stridsdelen hinner stanna innan brisaden sker.
   Tändrörsfunktionen är felfri, d.v.s. det blir inga blindgångare.

B1:1

Beskrivning med klassdiagram i UML: associationer och kompositioner



Interaktionen mellan stridsdel och mark

1. Om stridsdelen når marken och tändrörsinställningen är 1 (ingen fördröjning), eller hastigheten vid islaget är låg, sker brisaden vid träffpunkten.

   Ingen krater bildas.

2. Om tändröret däremot har fördröjning (2 eller 3) tränger stridsdelen ner i marken i enlighet med Petrys formel i [FortH 1, 1987]. Sträckan är lika för båda tändrörsinställningarna. En eventuell källarvägg stoppar stridsdelen totalt.

   Kraterdiametern beror av laddningsmängd och markbeskaffenhet. Runt kratern uppkastas en vall, vars ytterdiameter är dubbelt så stor som själva kraterns.

Interaktionen mellan stridsdel och byggnad

1. En byggnad kan träffas och penetreras av stridsdelar som inte dessförinnan passerat genom marken. Yttertak och icke bärande innerväggar (d.v.s. väggar som inte skiljer stomenheter) anses inte utgöra något hinder för stridsdelen.
2. Stridsdelar som penetrerar det nedersta golvet i byggnaden, vare sig det är källargolv eller första våningens golv, antas stanna ovanpå detta golv. Också stridsdelar som passerar ut genom en källaryttervägg under marknivån antas brisera innanför denna vägg. Likaså antas stridsdelar som briserar under en källarlös byggnad efter att först passerat in genom hussockeln ha sin brisadpunkt på golvet rakt ovanför.
3. Vid penetrationen av ett bjälklag eller en bärande vägg bromsas stridsdelen enligt formel angiven i [Bergman, 1952]. Stomenheterna drabbas också av en skada i form av stora sprickor i väggar och bjälklag i spåret efter stridsdelen.
4. Vid tändrörsinställning 1 antages brisaden ske då stridsdelen första gången träffar på ett bjälklag eller en bärande vägg. Den kan alltså dessförinnan ha passerat t.ex. ett yttertak.

   Vid tändrörsinställning 2 antages brisaden ske invid den andra väggen / bjälklaget i banriktningen, förutsatt att stridsdelen över huvud taget förmår slå igenom det första hindret.

   Vid tändrörsinställning 3 antages brisaden ske omedelbart före inträngandet i den vägg / bjälklag, där stridsdelens kinetiska energi inte är tillräcklig för genomträngning. Det kan naturligtvis också hända att stridsdelen förmår slå igenom alla hinder i byggnaden och fortsätta på andra sidan tills den når marken eller penetrerar en annan byggnad.
5. Som en följd av stridsdelsbrisader uppstår skador av olika slag i byggnaderna, som beskrivs med ett antal zoner:

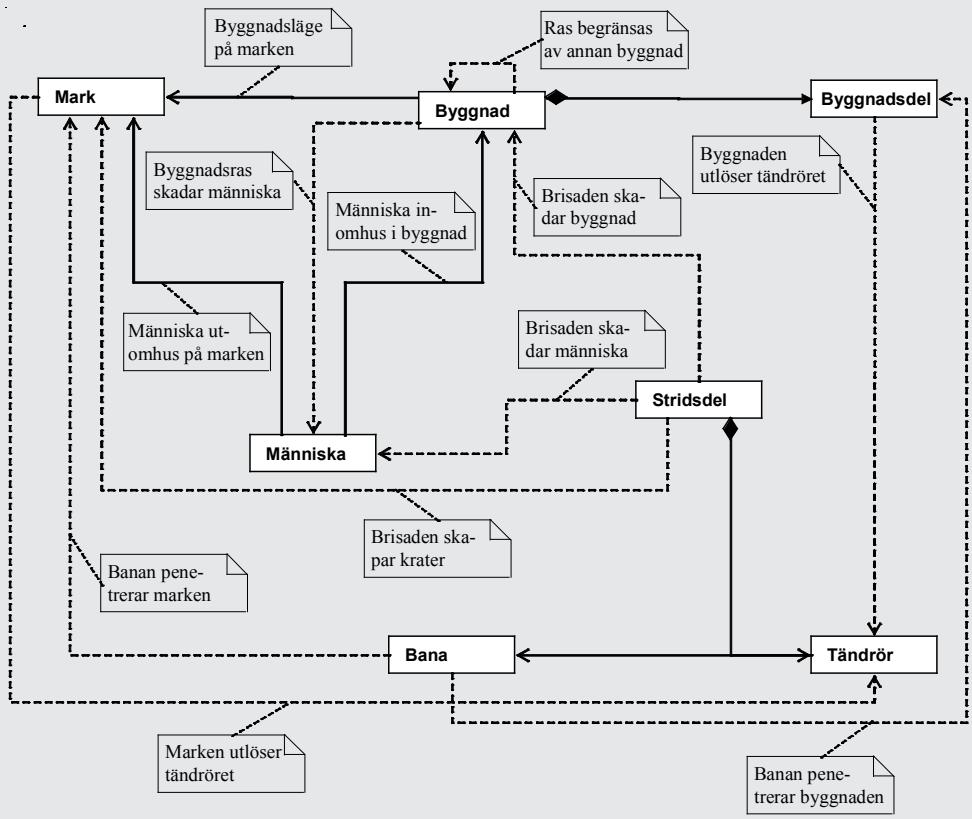
Golvraszon	De stomenheter på varje våningsplan där golvbjälklag ej längre bär.
Väggraszon	De stomenheter på varje våningsplan där minst en bärande vägg har kollapsat.
Sprickzon	De stomenheter där det i bärande byggnadsdelar förekommer stora genomgående sprickor. Alla fönster antas vara utslagna och alla dörrar öppna.
Skakzon	De stomenheter där bärande väggar erhållit en elastisk deformation. Alla fönster antas vara utslagna och merparten av dörrarna öppna. Lätta mellanväggar är skadade eller raserade.
Fönsterkrosszon	Det område där minst hälften av alla fönster är utslagna.
Fallzon	Det område utanför byggnadens ursprungliga planyta som ingår i rasområdet. Häri inräknas dock inte kastzonen.

Kastzon Det område utanför byggnadens ursprungliga planyta och utanför fallzonen dit byggnadsdelar, hela eller sönderbrutna, har kastats iväg av explosionen.

6. Då en stomenhet påverkas av mer än en brisad tas ingen hänsyn till additiva effekter utan skadan i området blir den värsta skada som någon av de enskilda brisaderna åstadkommer.
7. Tryckverkan mot en vägg blir densamma oavsett om väggen är skymd eller ej.
8. Golvras kan ej uppstå i byggnadens nedersta våning, oavsett om denna är en källare eller bottenvåningen i ett källarlöst hus. Dessutom kan skakzon endast förekomma om det dessutom uppstått minst sprickzon i anslutning till brisaden. Vindar behandlas över huvud taget inte vad avser skadeutfallet.
9. Fönsterkrosszonen är en oändlig, rät, cirkulär cylinder, vars axel går lodrätt genom brisadpunkten, och vars radie beror på stridsdelens laddnings- och höljesvikt. De stomenheter som till någon del ligger inom denna cylinder tillhör zonen.
10. Uppräkningsordningen för stomenheterna utgår i varje våning från den stomenhet (händelsecentrum) som ligger rakt över eller under brisadpunkten.
11. Fallzonen uppträder där väggras- eller golvraszonen når en yttervägg.
12. Kastzonen antas uppträda mitt för den stomenhet vari brisaden sker.
13. Vid brisad utomhus ovan jord beräknas verkan av tryck och impulstäthet för väggras med hjälp av formler enl. [Granström, 1958] och [Kingery et al., 1984]. Om någon del av en stomenhets yttervägg uppfyller förutsättningarna för väggras så anses stomenheten tillhöra väggraszonen. Om hela väggen till stomenheten uppfyller förutsättningarna kan det uppstå ras i ovanliggande våningar. Om så sker beror dels på hur många av stomenhetens väggar som raserats, dels vilken hustyp det är fråga om.
14. Sprickzonen beräknas vid utomhusbrisad ovan jord på samma sätt som väggraszonen under antagandet att den uppstår vid utböjningar som är hälften så stora som vid väggras.
15. Skakzon beräknas inte vid utomhusbrisad ovan jord. Inte heller är kastzon aktuellt. Fönsterkrosszon och fallzon beräknas som vid inomhusbrisad.
16. Vid brisad under marken bestäms det största brisadavstånd som kan ge väggras utgående från modellen i [Kristiansen et al., 1973]. En stomenhet vars yttervägg (under jord) till någon del ligger inom angivet avstånd från brisaden räknas till väggraszonen.
17. Sprickzon uppstår i hela byggnaden vid brisader under mark om kortaste avståndet mellan byggnad och brisad understiger en sträcka, som beror på laddningsvikten.
18. Skakzon uppstår i hela byggnaden vid brisader under mark om kortaste avståndet mellan byggnad och brisad är mindre än dubbla gränsavståndet för sprickzon.
19. Inga kast- eller fallzoner förekommer vid brisader under marken.

B1:2

Beskrivning med klassdiagram i UML: associationer och beroenden (streckad pil)



Interaktionen mellan skadade byggnader

1. Fall- och kastzoner omkring raserade byggnader begränsas av omkringliggande byggnader. Dock tas ingen hänsyn till att omkringliggande byggnader kan vara så låga att fallzonen ev. skulle kunna täcka dem eller att fragmenten i kastzonen skulle kunna flyga över dem.

Interaktionen mellan stridsdel och människa

1. Stridsdelar skadar människor antingen genom direkt övertryck eller genom splitter. Däremot tas ingen hänsyn till att en stridsdel i sin bana kan träffa en människa, och inte heller till att en sådan träff kan orsaka en brisad.
2. Splittrens förmåga att döda resp. skada beräknas utifrån människans sårbarhet och hennes avstånd från brisaden. En förutsättning för att några splitter över huvud taget skall nå fram är att det är fri sikt mellan brisadpunkten och människan. Om människan dessutom befinner sig längre bort än 200 m antas alla splitter vara verkningslösa.

3. Skador av direkt övertryck antas endast ske vid brisader ovan marken. Alla människor som befinner sig innanför gränsen för dödande tryck antas dö, däremot ingen som befinner sig utanför. Alla som befinner sig mellan gränserna för skadande tryck och dödande tryck antas likaså bli skadade, däremot ingen som befinner sig utanför.
4. Vid inomhusbrisader gäller samma villkor för både splitter och tryck med tillägg av att människan måste befinna sig i den stomenhet där brisaden sker.
5. Vid brisader under marken kommer markmaterialet att dämpa tryck- och splitterverkan. Endast människor som befinner sig mycket nära den krater som bildas kan då skadas eller dö.

Interaktionen mellan skadad byggnad och människa

1. Den skada som drabbar en människa på grund av att en byggnad rasar beror på inom vilken skadezon hon befinner sig.

Referenser

Ahlers, Edward B: *Fragment Hazard Study*. Minutes of the Eleventh Explosives Safety Seminar. Memphis, Tennessee (september 1969)

Bergman, Sten G.A.: *Tekniska meddelande från Fortifikationsförvaltningen*. Befästningsbyrån nr B8 (oktober 1952)

FortH 1 – *Fortifikationshandbok*, del 1. M7747-706111 (1987)

Granström, S.A.: *Beräkningsmetod för stötvågsbelastade konstruktioner*. Fortifikationsförvaltningen, forskningsbyrån, rapport nr 103:18 (1958)

Kingery & Bulmach: *Airblast Parameters from TNT Spherical Air Burst and Hemispherical Surface Burst*. Ballistic Research, Technical Report ARBRL-TR-02555 (april 1984)

Kristiansen, G. & Fuglaas, J.: *Fortifikasjonshåndbok – 1973. Våpenvirkninger*. Direktoratet for sivilt beredskap, Norge (1973)

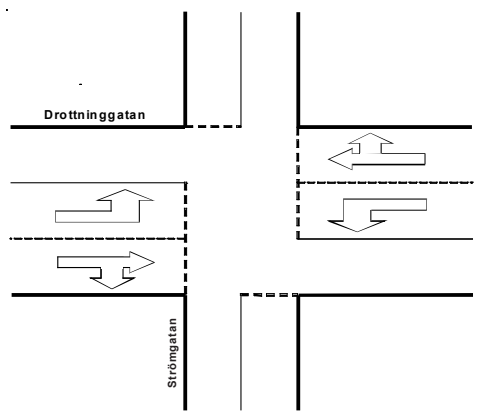
Lindqvist, Stig; Forsén, Rickard; Holm, Gunnar; Hägglund, Bengt & Onnermark, Bengt: *VEBE – Datoriserad dynamisk modell för konventionell vapenverkan i bebyggelse. Arbetslägesrapport 1992/1993*. FOA rapport C 20971-2.5(2.6,2.7) (juni 1994)

White, C.S.; Jones, R.K.; Damon, E.G.; Fletcher, E.R. & Richmond, D.: *The Biodynamics of Airblast*. Technical Report to Defence Nuclear Agency, DNA 2738T, Lovelace Foundation for Medical Education and Research (1971)

Exempel på en enkel ändlig tillståndsautomat

Signalreglerad gatukorsning

Låt oss anta att vi skall skapa en modell för regleringen av trafikströmmarna i en signalreglerad korsning mellan en större gata, Drottninggatan, i öst-västlig riktning och en mindre, Strömgatan, i nord-sydlig. Det finns sex trafikljus, varav två är synkroniserade:



D_{Vr} Drottninggatan från väster rakt fram eller högersväng

D_{Vv} Drottninggatan från väster vänstersväng

$D_{Ör}$ Drottninggatan från öster rakt fram eller högersväng

$D_{Öv}$ Drottninggatan från öster vänstersväng

S Strömgatan från både norr och söder

Vardera signalen kan han fyra olika tillstånd (ha något av följande värden):

G Grön Y Gul R Röd O Röd och gul

Det finns även sex sensorer som triggar ljusväxling och som kan ha två värden, antingen 1 (bil kommer) eller 0 (ingen bil i antågande). Två sensorer (i Strömgatans båda riktningar) är kopplade till varandra:

T_{dvr} Bil på Drottninggatan från väster skall rakt fram

T_{dvv} Bil på Drottninggatan från väster skall svänga vänster

$T_{dör}$ Bil på Drottninggatan från öster skall rakt fram

$T_{döv}$ Bil på Drottninggatan från öster skall svänga vänster

T_{s} Bil på Strömgatan

Om vi numrerar alla kombinationer av triggerinställningar {T_{dvr}, T_{dvv}, $T_{dör}$, $T_{döv}$, T_s} får vi

0	{0, 0, 0, 0, 0}	7	{0, 0, 1, 1, 1}	14	{0, 1, 1, 1, 0}	21	{1, 0, 1, 0, 1}	28	{1, 1, 1, 0, 0}
1	{0, 0, 0, 0, 1}	8	{0, 1, 0, 0, 0}	15	{0, 1, 1, 1, 1}	22	{1, 0, 1, 1, 0}	29	{1, 1, 1, 0, 1}
2	{0, 0, 0, 1, 0}	9	{0, 1, 0, 0, 1}	16	{1, 0, 0, 0, 0}	23	{1, 0, 1, 1, 1}	30	{1, 1, 1, 1, 0}
3	{0, 0, 0, 1, 1}	10	{0, 1, 0, 1, 0}	17	{1, 0, 0, 0, 1}	24	{1, 1, 0, 0, 0}	31	{1, 1, 1, 1, 1}
4	{0, 0, 1, 0, 0}	11	{0, 1, 0, 1, 1}	18	{1, 0, 0, 1, 0}	25	{1, 1, 0, 0, 1}		
5	{0, 0, 1, 0, 1}	12	{0, 1, 1, 0, 0}	19	{1, 0, 0, 1, 1}	26	{1, 1, 0, 1, 0}		
6	{0, 0, 1, 1, 0}	13	{0, 1, 1, 0, 1}	20	{1, 0, 1, 0, 0}	27	{1, 1, 0, 1, 1}		

Vi kan nu formulera i en tabell vilka triggerinställningar som orsakar de olika tillståndsövergångarna. För enkelhetens skull bortser vi t.v. från det gula, Y, och det röd-gula, O, tillståndet.

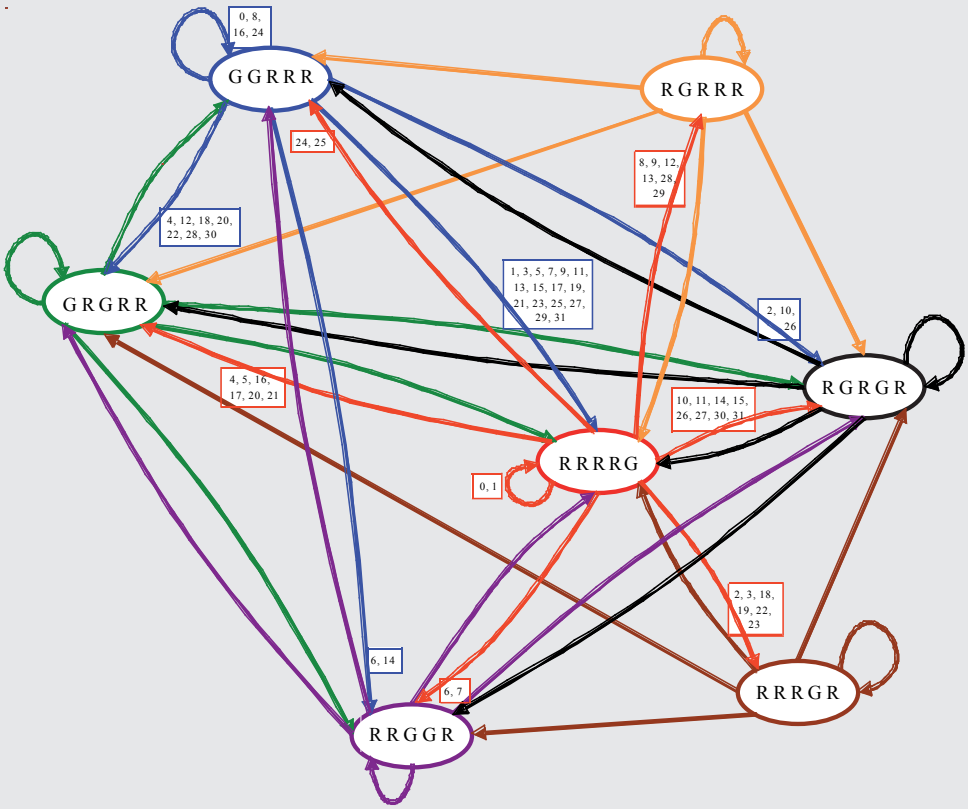
Start / Mål	RRRRG	RRRGR	RGRGR	RRGGR	RGRRR	GGRRR	GRGRR
RRRRG	0, 1	1, 3, 9, 11	1, 3, 9, 11	1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31	1, 3, 9, 11	1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31	1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31
RRRGR	2, 3, 18, 19, 22, 23	0, 2					
RGRGR	10, 11, 14, 15, 26, 27, 30, 31	8, 10	0, 2, 8, 10	8, 10, 14	2, 10	2, 10, 26	10, 14, 26, 30
RRGGR	6, 7	4, 6	4, 6	0, 2, 4, 6		6, 14	2, 6, 18, 22
RGRRR	8, 9, 12, 13, 28, 29				0, 8		
GGRRR	24, 25		16, 24	24, 26	16, 24	0, 8, 16, 24	8, 12, 24, 28
GRGRR	4, 5, 16, 17, 20, 21	5, 7, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31	5, 7, 12, 13, 14, 15, 17, 18, 19, 20, 21, 22, 23, 25, 26, 27, 28, 29, 30, 31	12, 16, 18, 20, 22, 28, 30	4, 5, 6, 7, 12, 13, 14, 15, 17, 18, 19, 20, 21, 22, 23, 25, 26, 27, 28, 29, 30, 31	4, 12, 18, 20, 22, 28, 30	0, 4, 16, 20

Vi kan också gestalta detta i grafisk form. Systemet {D_{Vr}, D_{Vv}, $D_{Ör}$, $D_{Öv}$, S} tänker vi oss då ha ett starttillstånd {R,R,R,R,G}, där det befinner sig till dess det kommer någon bil på Drottninggatan. Se faktaruta B 2:1 för en förenklad tillståndsgraf.

B2:1

Tillståndsövergångarna i trafikljusexemplet

Här redovisas de stora dragen, d.v.s. växlingarna mellan grönt och rött. Dessutom har endast triggervärderna för tillståndsövergångar från två av tillstånden, RRRRG och GGRRR, angivits för att bilden inte skall bli alltför rörig.

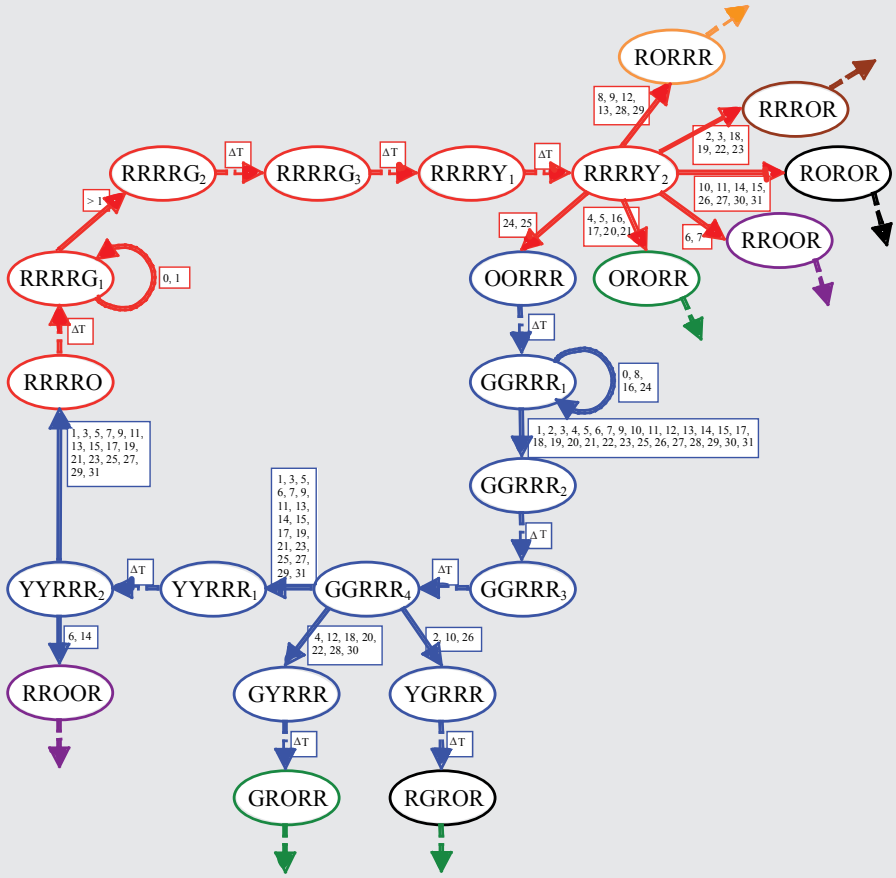


I en mera verklighetstrogen modell måste man naturligtvis även ta med de gula och röd-gula tillstånden. För att undvika att ljussignalerna slår om stup i ett inför vi dessutom en klocka som startar då triggervärdena indikerar att ljusväxling bör ske. Efter två perioder ($2 \cdot \Delta T$) släcks det gröna ljuset och i stället tänds det gula, som i sin tur lyser under en period (ΔT), innan det blir rött och något annat ljus blir röd-gult. I faktaruta B 2:2 har tillståndsdiagrammet delvis utvecklats för att illustrera detta.

B2:2

Tillståndsövergångarna med tillägg av tidsstyrning

Här redovisas den mer detaljerade modellen, d.v.s. även tillstånden gult och röd-gult sken är med. Dessutom har vi lagt till interna händelser i form av ”klocktick”, ΔT . Endast övergångarna mellan RRRRG och GGRRR har utarbetats fullständigt för att bilden inte skall bli alltför rörig.



Vad är ett system?

Begreppet system är lika vanligt förekommande, inte minst i denna skrift, som det för de flesta är diffust till sin innebörd. Det kan därför vara på sin plats att diskutera dess betydelse. I faktaruta B 3:1 ser vi hur Svenska Akademin definierar ordet på sex olika sätt (och då har vi ändå inte tagit med företaget som sköter det svenska alkoholmonopolet). Den betydelsevariant, som det i vårt fall närmast handlar om är nummer två, men definitionen kan behöva utvecklas något, och vi gör det utifrån professor Harold Lawsons tankar i ämnet.

Enligt honom är ett system dels ett föremål eller en företeelse som används för att uppnå ett visst syfte eller mål, att kunna utföra ett visst uppdrag. Dels är det en produkt eller en tjänst som tillhandahålls åt den som skall uppnå målet eller utföra uppdraget. Utifrån detta synsätt kan ett system inte existera i absolut mening utan endast i relation till något som skall utföras. Man kan jämföra med diskussionen om grankottemodellen av en ko (se avsnittet ”Modell”, sid. 8). Kottar och stickor är inget annat än just kottar och stickor förrän en människa ser analogin med en ko och kan utnyttja den för t.ex. lek.

Ytterligare en parallell mellan modeller och system finner vi om vi funderar på hur ett system är sammansatt. Vi nämnde i avsnittet ”Valet av grundobjekt” (sid.10) att en modell består av samverkande objekt, som i sin tur är modeller, som består av samverkande objekt o.s.v. På motsvarande sätt består ett system av komponenter, som är till för att lösa vissa deluppdrag inom systemet. Men det innebär att komponenten också kan betraktas som ett system, som i sin tur består av komponenter o.s.v. Denna uppdelning i system av system kan naturligtvis drivas hur långt som helst, men i det praktiska livet finns det alltid en undre gräns för uppdelningen. Vi betraktar vissa komponenter som så elementära och tillförlitliga att vi inte mentalt uppfattar dem som system. Exempelvis kan vi uppfatta en IKEA-hylla som ett system av gavlar, hyllplan, bakstycke, skruvar o.s.v., men vi uppfattar inte skruvarna som ett system av järn- och kolmolekyler.

Vad kan då systemkomponenter bestå av annat än skruvar? De kan naturligtvis vara andra mekaniska eller elektroniska prylar av allehanda slag, d.v.s. hårdvara. De kan också vara människor i olika roller, t.ex. som ope-

B3:1

Ur Svenska Akademins ordbok (2002):

System

ETYMOLOGI: [liksom dan. o. nor. *system*, t. *system*, eng. *system*, fr. *système*, ä. fr. äv. *sistème*, senlat. *Systema*, ytterst av gr. σύστημα, något sammanställt, sammanställning, till συνιστυναι, sätta samman, ordna, av σύν, samman.

1. om i naturen l. kroppen förekommande enskilda element l. delar l. dyl. som hänger samman med varandra så att de bildar en ordnad helhet; särsk. (i fackspr.) om en sådan enhet bestående av sammanhängande l. sammanhörande naturformationer (t. ex. sjöar l. bergskedjor); särsk. (*geol.*) om en stratigrafisk enhet omfattande alla bildningar från en viss geologisk period.
2. om sammanfattningen av enskilda sammanhängande l. sammanhörande l. i förbindelse bragta föremål l. komponenter l. delar l. element l. processer o.d. som bildar en samverkande enhet l. ordnad helhet o. fungerar efter bestämda principer.
3. (i fackspr., i sht *filos.*) om samlad enhet av iakttagelser l. erfarenheter l. idéer l. kunskaper l. förutsättningar l. handlingar o. d. förbundna o. logiskt ordnade efter (en) bestämd(a) princip(er) (o. formulerade i definitioner l. hypoteser o. d.); om ngt sammanhängande helt av begrepp; äv. om systematisk uppställning av satser l. regler som tillsammans utgör en lära, lärobyggnad l. lärosystem l. tankebyggnad.
4. (i fackspr.) om en systematisk sammanställning l. indelning av ngt; särsk. (*naturv.*) om en efter vetenskapliga principer (likhet l. släktskap o. d.) gjord sammanställning l. indelning (i arter, släkter, familjer, ordningar o. d.) över växter, djur o. mineral; äv. om sådan sammanställning osv. av grundämnen; äv. om princip(er) som ligger till grund för sådan sammanställning osv.
5. om sammanfattningen av allt som hör till viss företeelse l. verksamhet l. visst företag o. d., ofta liktydigt med: väsen(de) l. skick o. d. -- särsk. om social l. samhällsekonomisk l. politisk ordning l. form l. modell l. organisation, särsk. med tanke på de principer som den är (l. uppfattas vara) bestämd av; i sg. best. äv. dels om sammanfattningen av en stats o. d. olika organ l. institutioner o. d. med särskild tanke på dessas inbördes samverkan, dels om den rådande politiska uppfattningen; stundom närmande sig bet.: dels regim, dels statsform l. regeringsform.
6. metod l. plan l. planmässighet o. d.; ordning (o. reda). Bringa system i ngt.

ratörer. Men man behöver inte inskränka sig till enbart konkreta fysiska ting. Det bereder inga svårigheter att tänka sig mjukvara i form av datorprogram som en systemkomponent. Komponenter kan t.o.m. tänkas vara ännu mer abstrakta. Sålunda kan en föreskriven process (t.ex. tillverkningsprocesser, politiska och administrativa processer eller beslutsprocesser) också utgöra en komponent i ett system. Likaså kan då även procedurer, t.ex. handhavandeinstruktioner, uppfattas som systemkomponenter. Det centrala är att varje komponent har ett syfte, en produkt eller tjänst, som bidrar till att systemet i stort kan fullgöra sitt syfte, tillhandahålla den tilltänkta produkten eller tjänsten.

Sammanfattningsvis kan alltså ett system bestå av konkreta komponenter som maskiner, programvara för att styra dem och övervakande mänskliga operatörer, liksom av lagar, patent, kontrakt och överenskommelser. De sammanhålls av en mer eller mindre tydlig struktur och de bidrar till det för systemet gemensamma syftet.

Ett i vårt sammanhang centralt systembegrepp är **informationssystem**. I allmän mening har ett sådant system syftet att på något sätt behandla information för ett visst ändamål. Det tillhandahåller därför tjänster för insamling, bearbetning, lagring, sökning och distribution av information. I enlighet med vad som sagts generellt om system, så gäller alltså att ett informationssystem omfattar såväl teknisk utrustning som mänskliga aktiviteter och rutiner. Ett typiskt och ganska omfattande exempel är ett beslutsstödsystem, så som det beskrivits i kapitlet ”Ledningssystem och M&S” (sid. 103).

Ytterligare några intressanta avledningar till ordet system kan vara värda att notera i detta sammanhang, varför de presenteras i faktarutorna B 3:2 och B 3:3.

B3:2

Ur Nationalencyklopedin (1995):

systemanalys, analys av komplexa system som underlag för beslut, ofta med matematiska hjälpmedel. De system som är aktuella kan vara industriella tillverkningssystem, transportsystem etc. men även t.ex. ekologiska system. Avsikten med en systemanalys är att fastställa hur de resurser man förfogar över ska utnyttjas för att målen efter ingrepp ska uppfyllas på bästa möjliga sätt. T.ex. kan man vilja minimera väntetiderna i ett betjäningssystem, avgöra vilket resursuttag från ett ekologiskt system (virke, fisk) som är långsiktigt hållbart eller minimera risken för felfunktioner i komplexa tekniska system.

En systemanalys förutsätter att vissa allmänna förutsättningar och prestationsmått för systemet specificeras. Vidare krävs att man gör en systemavgränsning, dvs. klargör vad som kan påverkas (resurser, styrvariabler) resp. inte kan påverkas (omgivningen). De matematiska metoder som i första hand kommer till användning avser beskrivning av dynamiska system (differential- och differensekvationer, köteori; se simulering). Eftersom prestationsmåtten normalt är relaterade till effektivitet utnyttjas därjämte ofta optimeringsteori i analysen.

Historiskt har systemanalysen vuxit fram ur teorin för dynamiska system (Leibniz och Newtons differentialkalkyl), men en utvecklad teori kom först under senare delen av 1800-talet (James Maxwell, ryssen Ivan Vysjnegradskij, 1831-95). Under 1900-talet har tillkommit spelteorin för analys av system med flera beslutsfattare som kan ha sinsemellan motstridiga mål. Den under andra världskriget utvecklade operationsanalysen (OA) är en direkt föregångare till systemanalysen. Inom OA var uppgiften att utnyttja befintliga system på effektivast möjliga sätt (t.ex. att minimera förlusterna i de transatlantiska fartygskonvojerna genom lämpligt val av konvojstorlek), medan systemanalys i större utsträckning avser utveckling av nya system, vilket ger större friheter. För stora system med flera delvis motstridiga mål används ibland termen *policyanalys*. En teoribildning som har vissa inslag gemensamma med systemanalysen är systemteorin.

B3:2

Ur Svenska Akademins ordbok (2002):

Systematik

ETYMOLOGI: om (läran l. vetenskapen om) systematisk indelning l. framställning o.d., vetenskaplig klassificering; särsk. om efter visst system (släktskapsförhållanden o.d.) gjord sådan indelning osv. av växt- o. djurriket i arter, släkter, familjer osv.; stundom äv. närmande sig l. övergående i bet.: systematiserande synsätt o.d.; äv. allmännare: systematiskt tillvägagångssätt, planmässighet; systematisering.

Systematisera

ETYMOLOGI: sätta l. bringa l. ordna (ngt) i system; indela (ngt) i system; systematiskt l. vetenskapligt ordna l. sammanställa (ngt);

BETYDELSE: stundom liktydigt med: systematik; äv. allmännare, ofta liktydigt med: (metodiskt) ordna (ngt).

Systematisk

1. med sakligt huvudord: som hör till l. har sammanhang med l. har att göra med l. ingår i l. hänför sig till l. är i överensstämmelse med l. utföres efter visst system l. viss(a) metod(er) (som anges l. antyds av sammanhanget); planmässig l. metodisk l. ordnad efter ett bestämt system l. bestämd(a) metod(er); om vetenskap o.d. äv.: uppställd l. indelad l. ordnad i (ett vetenskapligt) system; äv. övergående i bet.: regelbunden l. konsekvent, stundom: grundlig l. noggrann; äv. allmännare, närmande sig bet.: medveten l. utstuderad o.d. *Arbeta systematiskt. Gå systematiskt tillväga. Ett systematiskt insamlande av material. En systematisk inventering, undersökning. Systematisk biologi, zoologi, växtsociologi.*

I sådana uttr. som *systematiskt* (*sett*), *i systematiskt hänseende, ur systematisk synpunkt*, om man ser till systemet. *Indelningen är ur systematisk synpunkt olämplig.*

2. om person: som tänker l. handlar l. arbetar o.d. efter ett genomtänkt system l. en genomtänkt metod; som utmärkes av ett systematiskt tillvägagångssätt;

BETYDELSE: metodisk; i sht förr äv.: som sysslar med systematisk vetenskap; äv. allmännare, övergående i bet.: grundlig l. noggrann o.d.; stundom allmännare, närmande sig bet.: utstuderad.

Systemera

ETYMOLOGI: (i fackspr., i sht *datal.*) konstruera l. analysera l. utveckla (databehandlings)system. ”Systemering” är en ny språklig term som används för verksamhet avseende analys, konstruktion och evaluering av informations- och databehandlingssystem.

FOI forskningsprojekt inom FoT rörande M&S

Distribuerad interaktiv simulering var den gemensamma nämnaren för ett antal projekt som bedrevs med lite varierande inriktning under åren 1996 – 2002. Bland de frågor som där ställdes och fick mer eller mindre uttömmande svar var:

- Vad krävs för att anpassa existerande modeller för värdering av militära system så att de blir lämpade att ingå i en distribuerad miljö i samverkan med andra modeller?
- Hur ska konceptet HLA utvecklas för att samtidig simulering på såväl operativ, taktisk som stridsteknisk nivå ska kunna ske (hantering av aggregering / disaggregering)?
- I vilka sammanhang är HLA-baserad distribuerad simulering lämplig för försvarsmaktens behov och när är monolitiska modeller (t.ex. baserade på något simuleringsramverk) bättre?
- Hur skall man genom utnyttjande av webben kunna bygga HLA-federationer och användargränssnitt för befintliga simuleringsprogram?

Modellsamordning drevs under fyra år från 1997 och behandlade mer övergripande frågor, t.ex.:

- Hur kan man för olika användningsområden förbättra kontinuiteten mellan modeller på olika nivåer – från teknisk till operativ nivå?
- Vad kan göras för att underlätta utveckling, återbruk och samordning av simuleringsmodeller?
- Hur kan FOI på ett effektivare sätt bidra till försvarets behov av modeller för olika tilllämpningar?

Verifiering, validering och ackreditering av simuleringsmodeller är ett projekt som inleddes 1998, och som ännu pågår fast nu som en del i ett internationellt samarbete under EDA (*European Defence Agency*). Några exempel på frågeställningar som behandlats inom detta projekt är:

- Hur ska vi se till att de modeller som används inom försvaret är bra och användbara för sina syften?

- Vad avses med en modells trovärdighet, hur mäts trovärdighet, hur relateras detta till VV&A och hur trovärdig behöver en modell vara?
- Hur kan VV&A-tekniker i omvärlden tillämpas vid modellering och simulering inom försvarsmakten?
- Hur ser den VV&A-metodik ut som kan understödja validerings- och ackrediteringsprocesser och som är användbar för försvarets behov?

Konceptuella modeller av militära operationer är ett sedan 2003 under olika namn pågående projekt. Under arbetets gång och alltefter som kunskapen växt har fokus successivt flyttats från CMMS-konceptets struktur till ontologiproblem. Några exempel på frågeställningar som belyser detta:

- I vad mån kan CMMS-konceptet tillämpas inom det svenska försvaret? Vilka riktlinjer och vilket underlag krävs då?
- Hur bör konceptuella modeller av militära operationer framställas och underhållas för att en praktisk återanvändning av och en verklig samverkan mellan framtida simuleringsmodeller ska vara möjlig?
- Vilka metoder, tekniker och verktyg behövs för att analysera, representera och modellera kunskap?
- Hur bör ett ramverk (DCMF, *Defence Conceptual Modelling Framework*) se ut, som kan bli en standard för försvarsmaktens framtida modellutveckling?
- Hur skall en arkitektur se ut som skall stödja semantisk interoperabilitet mellan system av olika slag? Hur kan man mäta graden av interoperabilitet i ett sammankopplat system?

Omvärldsmodellering, som påbörjades 2000, handlar om metod- och teknikutveckling för dataförsörjning och framställning av detaljerade terrängmodeller för användning i simuleringsmodeller. Några problemområden som behandlats är:

- Hur kan man skapa detaljerat dataunderlag från olika typer av spaningsdata?
- Hur kan försvarsmaktens förmåga att hantera omgivningsmodeller förbättras med ökad samordning och nyttjande av standarder?
- Hur kan det amerikanska SEDRIS-konceptet (ursprunglig betydelse: *Synthetic Environment Data Representation and Interchange Specification*, numera står akronymen för sig själv) tillämpas inom den svenska försvarsmakten?
- Vilka metoder, tekniker, standarder och verktyg är lämpliga för att framställa, hantera och nyttja syntetiska naturliga omgivningar för försvarsmaktens behov?
- Vilka metoder, tekniker, standarder och verktyg är lämpliga för att genomföra sensorsimuleringar i syntetiska naturliga omgivningar?
- Hur kan dynamisk terräng hanteras i försvarsrelaterad M&S med tonvikt på urban miljö?

M&S–ramverk för uppföljning, utbildning och träning av sammansatta insatsstyrkor pågick 2001 – 2004 och hade fokus på frågor av typen:

- Vilka metoder och tekniker kan effektivt stödja uppföljning av sammansatta insatsstyrkor inom totalförsvaret?

- Hur kan systematiskt applicerade M&S-metoder stödja lärande och utveckling i en insatsorganisation?
- Vilka modeller, metoder och teknik krävs för att stödja ovanstående frågor?

Modellering och simulering av mänskligt beteende, som startade 2002, kan sägas vara nära knutet till beteendeforskningen. Följande frågeställningar exemplifierar vad detta projekt sysslar med:

- Vilka möjligheter och förutsättningar finns att bygga upp ett försvarsmaktsgemensamt bibliotek med realistiska och återanvändbara modeller av datorgenererade styrkor för träning, utbildning och analys?
- Vilka riktlinjer behövs för ett sådant referensbibliotek med modeller på olika nivåer och för olika ändamål?
- Vilka krav ställs vid insamling av data för att de ska vara användbara i referensbiblioteket?
- Hur kommunicerar människor med datorgenererade aktörer och datorgenererade aktörer sinsemellan på bästa sätt?
- Hur valideras modeller av mänskligt beteende?
- Hur ser processen ut från behovsställning och målsättning för träning till utveckling av CGF:er, användning, utvärdering och vidmakthållande?

Nätverksbaserad modellering och simulering pågick 2004 – 2006 med inriktningen att få en djupare insikt i arkitekturer som bygger på komponentbaserad utvecklings- och exekveringsmetodik och där simuleringskomponenterna kommunicerar via ett nätverk. Bland frågeställningar som behandlas i detta projekt märks:

- Vilka metoder och tekniker är lämpliga för att realisera nätverksbaserad M&S på olika nivåer?
- Hur kan nätverksbaserad M&S utnyttjas för att
  - öka återanvändningen, få effektivare utveckling och enklare åtkomst av simuleringsmodeller,
  - öka samarbetet mellan utvecklare och användare,
  - få effektivare utnyttjande av befintliga datorresurser?
- Vilka metoder, tekniker och verktyg för nätverksbaserad M&S finns tillgängliga idag? Hur kan de anpassas till försvarsmaktens behov? Vad behöver tas fram därutöver?

Semantiskt baserat distribuerat resursbibliotek inleddes 2007 för att följa upp ett av spåren inom föregående projekt. Här avser man att besvara bl.a. följande frågor:

- Hur kan man samarbeta över organisationsgränser och dela med sig av resurser på ett säkert sätt?
- Hur skall resurserna beskrivas och hanteras?
- Hur anpassas resursbiblioteket till försvarsmaktens behov?

Simuleringsbaserade inbyggda system (SBIS) inom det nätverksbaserade försvaret påbörjades 2005 och planeras att avslutas under 2007. Här studeras simuleringar som samverkar i realtid med komplexa fysiska distribuerade system i syfte att styra eller optimera deras beteende eller för att skapa beslutsunderlag. Projektet har bl.a. tagit sig an följande frågor:

- Vilka lämpliga metoder finns eller behöver tas fram för utveckling av SBIS?

- Hur kan SBIS utnyttjas i det nätverksbaserade försvaret?
- Hur ser kopplingen ut mellan SBIS och andra områden t.ex. datafusion?

Dataspel och försvar påbörjades 2006 med inriktningen att nyttiggöra kommersiella datorspel i försvarsmakten. Bland frågeställningar som behandlas i detta projekt märks:

- Vilka krav bör ställas på interoperabilitet och integration? Vilka krav måste ställas på möjligheten att bygga, övervaka och utvärdera scenarier?
- Vilken typ av träning och utbildning kan genomföras med kommersiella datorspel?
- Hur förhåller sig mänsklig prestation i dataspel vid jämförelse med prestationerna i traditionella simulatorer eller på fältet?

Modellering och simulering som beslutsstöd på strategisk och operativ nivå startade 2006 med inriktningen att inventera vilka modeller och stödverktyg som finns i förekommande modellbibliotek inom NATO. De frågeställningar som skall besvaras är:

- Vilka av bibliotekets modeller korresponderar med det modellbehov som föreligger inom försvarsmakten för studier, analys och planläggning på olika nivåer?
- Vilka av dessa verktyg är tillgängliga inom gällande avtal?
- Hur har validering, verifiering och ackreditering hanterats för dessa modeller?
- Vilka förändringar kan modellerna behöva genomgå för att motsvara svenska behov och krav?

Under 2008 kommer ytterligare ett par forskningsprojekt att starta:

M&S infrastruktur för *rapid prototyping* avser att behandla frågeställningar av typen

- Hur kan befintliga kommersiella produkter bidra till att skapa en infrastruktur för att relativt enkelt kunna simulera komplexa förlopp?
- Hur utformas och driftsätts en sådan infrastruktur så att man med dess hjälp snabbt och billigt skall kunna
  - prova idéer och minska osäkerheten i genererat beslutsunderlag,
  - skapa demonstrationsscenarier och presentationer av nya koncept?

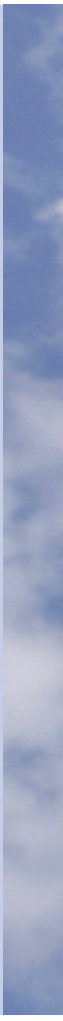
Realtidssimulering som stöd för effektbaserad planering kommer att leda till bättre kunskaper rörande följande frågor:

- På vilka nivåer är det meningsfullt att utnyttja realtidssimulering som stöd för effektbaserad planering?
- Vilka typer av simuleringar är meningsfulla på dessa nivåer? Vilka krav ställs på simuleringarna?
- Hur kan sådana simuleringar kopplas till ledningssystem?
- Hur bör resultaten visualiseras?

Människan har en inneboende strävan att förstå, påverka och förutsäga sin omvärld. Exemplen är oräkneliga. Det kan handla om hur morgondagens väder ska bli. Det kan gälla att inse effekterna av en föreslagen ny väg runt en storstad eller att förstå hur olika samhällsfunktioner bör samverka vid katastrofhantering. Att skapa modeller av verkligheten och simulera händelser i denna modellvärld är då ett mycket kraftfullt hjälpmedel.

Denna bok vänder sig i första hand till dem som vill modellera och simulera relativt komplexa sammanhang, men är även relevant för modeller av enklare slag. Grundläggande teori blandad med konkreta exempel ger läsaren kunskaper som underlättar ett aktivt deltagande i verksamheter där modellutveckling eller modellanvändning utgör en viktig del. Styrkor, svagheter och vanliga fallgropar förklaras på ett tydligt sätt. Dessutom ges en översikt över olika modelltyper och den begreppsapparat, som är användbar vid modellutveckling och simulering.

Boken har visserligen sin utgångspunkt inom den militära domänen, men de principer och begrepp som behandlas är generella och lika viktiga inom alla tillämpningsområden.



FOI
Totalförsvarets forskningsinstitut
164 90 Stockholm

Tel 08-555 030 00
Fax 08-555 031 00

www.foi.se
ISBN 978-91-7056-124-5