FOA-R--00-01435-505-SE
Februari 2000
ISSN 1104-9154

Användarrapport

Per-Olof Fjällström, Göran Neider, Mats Persson, Tore Risch
och Per Svensson

Arkitekturprinciper för informationsöverlägsenhet i framtidens ledningsstödsystem

Avdelningen för
Ledningssystemteknik
SE-581 11 LINKÖPING

FÖRSVARETS FORSKNINGSANSTALT
Avdelningen för Ledningssystemteknik
SE-581 11 LINKÖPING

FOA-R--00-01435-505--SE
Februari 2000
ISSN 1104-9154

Per-Olof Fjällström, Göran Neider, Mats Persson, Tore Risch
och Per Svensson

Arkitekturprinciper för informationsöverlägsenhet i framtidens ledningsstödsystem

Avdelningen för
Ledningssystemteknik
SE-581 11 LINKÖPING

Dokumentets utgivare	Dokumentbeteckning, ISRN	
Försvarets forskningsanstalt Avdelningen för Ledningssystemteknik SE-581 11 LINKÖPING	FOA-R--00-01435-505-SE	
	Dokumentets datum Februari 2000	**Uppdragsnummer** E1433
	Projektnamn (ev förkortat) Systemarkitektur	

Upphovsman(män)	Uppdragsgivare
	FM
Per-Olof Fjällström, Göran Neider, Mats Persson, Tore Risch, Per Svensson	**Projektansvarig** Per Svensson
	Fackansvarig Per Svensson

Dokumentets titel

Arkitekturprinciper för informationsöverlägsenhet i framtidens ledningsstödsystem

Sammanfattning

Detta är den avslutande rapporten för projektet Systemarkitektur. Projektets tidsperspektiv är långsiktigt och avsikten har varit att utreda förutsättningar för successiv övergång till en integrerad, nätverksbaserad teknisk ledningssystemstruktur under de två första decennierna av 2000-talet.

I denna rapport, liksom i förstudierapporten från 1998, används begreppet *evolutionär systemutveckling* för att beteckna en metodik och teknik som gör det möjligt att först pröva nya idéer i praktiken och därefter bygga ut och modifiera ledningssystemets funktioner i takt med förändrade behov och tillgång till nya tekniska lösningar. Rapporten redovisar metodmässiga och systemtekniska lösningar som vi bedömer som särskilt viktiga för att åstadkomma denna flexibilitet. I rapporten hävdas också att begreppet *modellbaserad ledning* bör läggas till grund för en nödvändig gradvis integrering av ledningssystem, sensorsystem och simuleringsmodeller för ledning (*ledningssimulatorer*).

En av utgångspunkterna för våra resonemang är det ofta uttryckta behovet av en *gemensam lägesbild.* Vi påpekar att detta begrepp bör uppfattas, inte som en bild, utan som en metafor som omfattar all potentiellt användbar och insamlingsbar tillståndsinformation om egna och fientliga förband, och som också kan innehålla bedömda eller beräknade hot och möjligheter förknippade med en given situation.

Den metodik och teknik som presenteras i rapporten har valts ut därför att vi tror att den erbjuder goda möjligheter att bygga sådana komplexa system av system som det framtida ledningssystemet kommer att behöva bestå av. En del av de tekniska lösningar som vi beskriver, bland dem medlartekniken, finns ännu inte som mogna produkter. Vi beskriver dock enbart sådan teknik som existerar antingen som etablerade produkter och standarder eller som väl utprovade forskningsprototyper.

Nyckelord

systemarkitektur

Övriga bibliografiska uppgifter	**Språk** Svenska

ISSN 1104-9154	ISBN	
	Omfång 106 sidor	**Pris** Enl. prislista

Distributör (om annan än ovan)

Issuing organization

Defence Research Establishment
Division of Command and Control Warfare Technology
SE-581 11 LINKÖPING

SWEDEN

Document ref. No., ISRN

FOA-R--00-01435-505-SE

Date of issue

February 2000

Project No.

E1433

Project name (abbrev. if necessary)

System architecture

Author(s)

Per-Olof Fjällström, Göran Neider, Mats Persson, Tore Risch, Per Svensson

Initiator or sponsoring organization

Project manager

Per Svensson

Scientifically and technically responsible

Per Svensson

Document title in translation

Architecture Principles for Information Superiority in Future Command and Control Systems

Abstract

This is the final report from the System Architecture project. The project takes a long-range perspective and its purpose is to investigate prerequisites for a successive change of focus towards a network-based technical and system architecture for future Swedish Command and Control systems during the next two decades.

In this report, as in the prestudy report from 1998, the concept of *evolutionary system development* is used to denote the vision of a methodological and technological approach. This would enable the gradual extension and modification of the functions of the command and control system, in step with changes in requirements and as new technical solutions become available. The report presents methodological and software technology approaches which we consider particularly important to achieve the required flexibility. The report also claims that the concept of Model Based Battle Command should be adopted as the basis for a necessary gradual integration of command and control, sensor, and C2 simulation systems.

A starting point for much of our discussion is the frequently expressed need for a *common operational picture*. We point out that this concept should be understood, not as a veritable picture, but as a metaphor which encompasses all potentially useful and collectible state information about own and enemy forces, as well as estimated or calculated threats and opportunities associated with a given situation.

The methods and the technology presented were selected because they should offer good opportunities to create such a complex system of systems which will form the backbone of the future C2 system. Some of the technical solutions we present, among them the mediator technology, do not yet exist as mature products. We describe, however, only such technology which exists either as established products and standards or as well-tested research prototypes.

Key words

system architecture

Further bibliographic information

Language Swedish

ISSN 1104-9154

ISBN

Pages 106

Price Acc.to pricelist

Distributor (if not issuing organization)

INNEHÅLLSFÖRTECKNING

Figurförteckning

Förord

Det ursprungligen angivna målet för projektet Systemarkitektur var ”att formulera en övergripande arkitektur för framtida datorstödda ledningssystem samt att inom denna arkitektur utveckla och demonstrera metoder som möjliggör en dynamisk och synkroniserad teknik- och kunskapsutveckling”.

Efter att ha studerat arkitekturfrågor för ledningssystem under projektets treåriga existens, uppfattar vi idag vårt uppdrag något annorlunda, nämligen som att identifiera och översiktligt beskriva grundförutsättningar beträffande främst teknik, men med nödvändighet också beträffande metodik, utbildning och organisation, som vi anser måste uppfyllas för att Försvarsmaktens ledningssystemutveckling skall bli framgångsrik.

Projektets tidsperspektiv är långsiktigt och avsikten är att utreda förutsättningar för successiv övergång till en integrerad, nätverksbaserad teknisk ledningssystemstruktur under de två första decennierna av 2000-talet. De informationstekniska utvecklings-tendenser som vi pekar på i denna rapport är dock i huvudsak sådana som redan nu någorlunda kan överblickas.

Vi tror inte att det är nödvändigt eller ens önskvärt, att i förväg i detalj fastslå en specifik arkitektur för det framtida ledningssystemet och föreslår istället att Försvarsmakten satsar på att, med stöd av särskilt FMV och FOA, bygga upp den organisationsstruktur och den kompetens som krävs för att upphandla, specificera och genomföra detta krävande och långsiktiga projekt. Vi föreslår också att konstruktion och utveckling bedrivs i nätverksform, där en systemleverantör ansvarar för leveranser gentemot beställaren Försvarsmakten, men där olika kompetenscentra, och inte minst tillgänglig informationsteknisk kompetens, ges tillfälle att bidra till forsknings- och utvecklingsarbetet i olika roller. Vi anser att forsknings- och utvecklingsarbetet bör bedrivas i relativt korta etapper med demonstration av ett konkret resultat som avslutning på varje etapp. Vi anser inte att det är möjligt att i förväg planera hela utvecklingsarbetet som ett enda projekt, utan föreslår att man istället bedriver delvis parallella studier och utvecklingsaktiviteter baserade på i förväg överenskomna arkitektoniska standarder, på ett sådant sätt att de olika aktiviteternas resultat efterhand kan sättas samman till ett gemensamt ”system av system”.

I denna avslutande rapport, liksom i förstudierapporten från 1998 [1], används begreppet *evolutionär systemutveckling* för att beteckna en metodik och teknik som gör det möjligt att först pröva nya idéer i praktiken och därefter bygga ut och modifiera ledningssystemets funktioner i takt med förändrade behov och tillgång till nya tekniska lösningar. Rapporten redovisar metodmässiga och systemtekniska lösningar som vi bedömer som särskilt viktiga för att åstadkomma denna flexibilitet. Förutom att ge synpunkter på sättet att strukturera själva systemen vill författarna också påverka debatten om valet av målsättningar och sättet att leda och organisera utvecklingsarbete och därmed bidra till arbetet att skapa en effektiv struktur för Försvarsmaktens ledningssystemutveckling. För att åskådligt kunna anknyta våra resonemang till den problemanalys som gjordes i förstudien har vi ”återanvänt” delar av förstudierapporten, särskilt i texten till kap. 1 och 2.

I rapporten hävdas att begreppet *modellbaserad ledning* bör läggas till grund för en nödvändig gradvis integrering av ledningssystem, sensorsystem och simuleringsmodeller för ledning (*ledningssimulatorer*). Funktioner och delsystem för *beslutsstöd* och *informationsfusion* kommer att vara två av de viktigaste komponenterna i det framtida ledningssystemet; det svåra arbetet att analysera och evolutionärt utveckla dessa komplexa funktioner behöver påbörjas nu och alla tillgängliga, relevanta kompetenser från användare till teoretiska dataloger behöver involveras i detta samarbete.

Även om texten nedan huvudsakligen består av referat och analyser av forskning som utförts av andra än författarna, är de idéer vi presenterar inte enbart läsefrukter. En av författarna till denna rapport, Tore Risch, är professor i databasteknik vid Uppsala universitet och en av upphovsmännen till den s k *medlarteknik* som vi tror kommer att spela en viktig roll för evolutionär systemutveckling i framtiden. Mats Persson är forskare i datasäkerhet för nätverksbaserade system vid FOAs institution för systemanalys och IT-säkerhet och Per-Olof Fjällström forskar om simuleringsmetodik för ledningssystemtillämpningar vid FOAs institution för modellering och simulering. Projektledaren Per Svensson forskar om informationsfusion för taktisk underrättelseanalys vid FOAs institution för data- och informationsfusion samt om integration av spatiella data och bilddata i objektdatabaser vid KTHs institution för numerisk analys och datalogi. Göran Neider är forskare och systemutvecklare vid samma FOA-institution.

Den metodik och teknik som presenteras i rapporten har valts ut därför att vi tror att den erbjuder goda möjligheter att bygga sådana komplexa system av system som det framtida ledningssystemet kommer att behöva bestå av. Tillflödet av nya idéer och ny teknik på IT-området är enormt, och varken författare eller läsare skulle ha tålamod att ta till sig allt som eventuellt kan vara av värde. Vårt urval är därför subjektivt och partiellt, men om vi som IT-forskare ska kunna bidra till Försvarsmaktens ledningssystemutveckling är det snarare genom att peka på nya möjligheter än genom att bekräfta redan etablerade teknologiska trender. En del av de tekniska lösningar som vi beskriver, bland dem medlartekniken, finns ännu inte som mogna produkter, och vi kan inte lämna någon garanti för att de inom kort kommer att erbjudas på marknaden, än mindre att de kommer att överleva i konkurrensen. Vi har ändå ansträngt oss att vara konkreta och realistiska och beskriver därför främst sådan teknik som existerar antingen som etablerade produkter och standarder eller som väl utprovade forskningsprototyper.

Avsnittet om metoder innehåller en text som beskriver Metria GISCentrums utvecklingsmetodik *Symmetria*. Skälet till att vi valt att presentera just denna metod framför alla andra är delvis tillfälligt: GISCentrum, som sedan sin tillkomst varit Försvarsmaktens expertkonsulter inom geoinformatikområdet, har på min förfrågan generöst ställt en beskrivning av sin metod till förfogande. Bakgrunden är att vi velat presentera en metodik som bygger på *Unified Modeling Language*, UML, en relativt ny industristandard som sannolikt kommer att bli dominerande.

Per Svensson
projektledare

Sammanfattning för beslutsfattare

Rapporten har byggts upp kring ett antal teman som tillsammans bildar en vision av hur det framtida ledningsstödsystemet bör vara uppbyggt, liksom av hur den organisation kan vara beskaffad som ska leda denna utveckling. Vi presenterar vare sig en komplett teknisk design eller konkreta organisationsförslag, utan analyserar problem, ställer krav på och ger förslag till principlösningar i både den tekniska och organisatoriska dimensionen.

Eftersom författarna är tekniker och inte organisationsexperter ligger rapportens tyngdpunkt i tekniken, och våra synpunkter på utvecklingsorganisation kan förmodligen uppfattas som amatörmässiga. Men som iakttagare och ”avnämare” av den hittills existerande organisationsstrukturen och dess konsekvenser, liksom av egen verksamhet inom avancerad systemutveckling, har vi fått konkreta erfarenheter av organisationsförhållandenas avgörande inflytande på utvecklingen av metoder och teknik.

De teser som rapporten för fram är:

1. begreppet *modellbaserad ledning* inklusive informationsfusion och beslutsstöd som en fundamental vision för en nödvändig gradvis integrering av ledningssystem, sensorsystem och simuleringsmodeller för ledning (*ledningssimulatorer*)
2. delsystem för *beslutsstöd* och *informationsfusion* kommer att vara två av de viktigaste komponenterna i framtida ledningssystemet; det svåra arbetet att analysera och evolutionärt utveckla dessa komplexa funktioner behöver påbörjas nu och alla tillgängliga, relevanta kompetenser från användare till teoretiska dataloger behöver involveras i detta samarbete
3. fungerande metodik för *evolutionär systemutveckling* och *förvaltning* behöver skapas, organiseras och tillämpas
4. *databaser* kommer även framgent att spela en central roll i ledningsstödsystem, men framtidens databaser kommer att innehålla dynamiskt föränderliga sensor- och bilddata, målspårsinformation, terrängdata och annan omgivningsinformation, organisations-, materiel- och doktrindatabaser och skall i varje tidpunkt och även historiskt representera och därefter arkivera den mångfasetterade *gemensamma lägesbilden;* databaserna blir *objektorienterade, dynamiska, multimediala, distribuerade* och *heterogena*
5. *interoperabilitet* mellan distribuerade, heterogena ledningssystem, databaser och ledningssimulatorer behöver efterhand åstadkommas och många informationsmodeller för systemets olika databaser måste då kunna samexistera
6. ledningssystemet kommer att utvecklas mot ett *nätverksbaserat system* med *semiautonoma mobila noder*; för detta krävs tillämpning och kanske utveckling av *distribuerade replikeringsalgoritmer* för att systemets databaser skall kunna ge en så välsynkroniserad, distribuerad lägesbild som möjligt
7. *medlartekniken* tillhandahåller verktyg för att skapa gemensamma vyer av heterogena databaser och åstadkomma ökat *dataoberoende*, så att strukturen hos databaser kan ändras utan att högre systemnivåer (”applikationer”) behöver ändras; dessa verktyg kan göras robusta i förhållande till den snabba teknikutvecklingen

8. medlarteknik framstår av detta och andra skäl som ett mycket viktigt hjälpmedel för att åstadkomma evolutionär systemutveckling och modellbaserad ledning; vi ser denna teknik som central och föreslår att den *utvärderas systematiskt* i ledningssystemdemonstratorer
9. stränga krav på *informationssäkerhet* ställs på ledningssystemen men man behöver finna vägar som gör det möjligt att se dessa krav ur ett risk-nyttoperspektiv; dagens absoluta, lagstiftningsbundna säkerhetskrav tenderar att i onödan låsa utvecklingen och förhindra adekvat användning av den nya tekniken
10. *integrerad* säkerhetsdesign, säkerhetsdokumentation och kvalitetskontroll krävs för att åstadkomma acceptabel informationssäkerhet; adekvata säkerhets-överväganden skall ligga till grund för systemutvecklingen, inte läggas på när systemen redan skapats
11. Försvarsmakten bör minimera antalet olika *utvecklingsmetoder* och *utvecklings-miljöer* som används för prototyputveckling och riktlinjer för val av sådana metoder och miljöer bör utformas; trots detta måste osäkra och osystematiska prototyper och demonstratorer även i fortsättningen få förekomma i rimlig utsträckning, så att den evolutionära utvecklingen inte hindras av byråkratiska direktiv
12. avancerad *4D-visualisering* har länge tillämpats för främst flygsimulatorer även i Sverige; sådan teknik förtjänar att få en central plats i ledningssystemet men måste då integreras med ledningssystemens distribuerade, dynamiska lägesdatabaser och simuleringsmodeller till en enhetlig arkitektur för modellbaserat ledningsstöd

1. Möjligheter och problem kring framtidens ledningssystem

Krigshistorien har många exempel på tekniskt-taktiska ”revolutioner” eller paradigmskiften. Den nazityska arméns taktiskt djärva och skickliga utnyttjande av det på den tiden snabbt föränderliga pansarvapnet är ett åskådligt om än inte alldeles aktuellt exempel. En insikt vi idag kan behöva erinra oss med anledning av detta är att investeringar i försvarsförmåga måste göras långsiktigt, framsynt och fördomsfritt, om resultatet ska bli av mer bestående värde än vilseledning av den egna opinionen och konstgjord andning åt förlegade organisationsstrukturer. Ett av få tänkbara alternativ vore att öppet avstå från att vidmakthålla ett militärt försvar, men då skulle vi å andra sidan tvingas svika våra internationella åtaganden och skyldigheter som självständig stat, alldeles bortsett från hur man bedömer riskerna för och konsekvenserna av att bli militärt angripen och kanske ockuperad.

Nästa tekniskt-taktiska paradigmskifte kan nu skönjas. Det militära systemet kommer denna gång att vara ännu mer beroende av den civila teknikens utveckling än den var då pansarvapnet utvecklades. Processen har fått namn redan i förväg, *”A Revolution in Military Affairs*”, RMA [2]. En av förutsättningarna för RMA – de övriga är avancerade sensorer, precisionsvapen och modern logistik –, och den som tydligast representerar det nya tänkesättet, är utvecklingen av ledningssystemet – och i första hand den starkt förändringsdrivande tekniska delen av ledningssystemet. I likhet med t ex den civila marknadens finans- och företagssystem är de militära ledningssystemen i färd med att integreras till heltäckande *nätverk*. Denna utveckling sker i växelspel med framväxten av ett nytt, globalt kommunikationsnätverk baserat på Internet, World Wide Web och mobila terminaler (mobiltelefonernas multimediala efterföljare).

Särskilt omvälvande kan man förvänta att den nya teknikens förmåga till *full konnektivitet, mobilitet* och *omedelbar respons* kommer att vara, i kombination med de nya möjligheterna att skapa *distribuerade intelligenta system*. Ett nätverk kommer att bli tillgängligt där tekniken inte längre lägger hinder i vägen för omedelbar kontakt mellan människor likaväl som datorer, en-till-en, en-till-många eller många-till-många, över hela organisationen och även utanför den, oberoende av var i världen man befinner sig.

Detta betyder för de militära tillämpningarnas del bl a att konferenser, förfrågningar, order och t o m organisationsförändringar kan initieras lättare än vi idag ringer ett telefonsamtal, låt vara att tillgången till kommunikationsmedlen även i framtiden kan komma att behöva begränsas med administrativa medel.

Det militära begreppet *ledningssystem* har i Sverige varit, och är fortfarande, reserverat för hela den organisatoriska, personella och materiella struktur som försvarsmakten byggt upp för att på grundval av ett mer eller mindre detaljerat och precist, i förväg fastställt regelverk, *ledningsdoktrinen* [3], kunna leda försvarets förband i krig, kris och fred. Eftersom de operativa uppgifterna i fred åtminstone hittills inskränkt sig till att hålla en viss beredskap, i vilket ingår att samla in och analysera underrättelser, skydda vår territoriella integritet (dvs i tid upptäcka och avvisa inkräktare) samt efter beslut av regering och riksdag i begränsad omfattning delta i internationella fredsbevarande eller fredsframtvingande operationer, har ledningssystemets fredstida roller och uppgifter varit mycket olika de krigstida.

Detta är ett av de problem som måste hanteras i samband med förändring och utveckling av ledningssystemet. Direktiv och attityder i detta avseende har varierat under de gångna 20 åren[1], från en stark betoning av krigsuppgiften – invasionsförsvaret – under det kalla kriget ända in på 80-talet, via olika satsningar på datoriserat, administrativt verksamhetsstöd främst för fredsverksamhetens behov, till att åter betona och prioritera krigets krav. I dessa doktrinförändringar ingick att hela organisationen skulle krigsanpassas, med ledningssystemet som en viktig del. Den uppfattningen gäller fortfarande, men kan sägas ha modifierats av de starkt ökade krav på förmåga till snabba fredstida internationella operationer i samverkan med andra länder, inte minst NATO-länderna, som ställs idag.

För att inte hamna i en situation där tekniken styr verksamhetsutvecklingen i stället för tvärtom, har man inom försvarsmakten, inte minst i styrdokumentet *ÖB Grundsyn Ledning* från 1994, betonat att utveckling av ledningssystemet inte är en teknisk fråga, utan istället handlar om att skapa och vidmakthålla en komplex social struktur, som har såväl auktoritet som reell förmåga att leda landets försvarsmakt till framgång om vi någon gång i framtiden skulle bli militärt angripna. Sett ur en teknikers synvinkel ger dock detta synsätt inte mycket hjälp när det gäller att analysera behov av ny metodik och utveckla nya funktioner hos de tekniska delarna av ledningssystemet. En annan brist är att vi saknar ett entydigt begrepp för dessa IT-system, som avgränsar dem från de ”mjuka” delarna av det som officiellt är ledningssystemet. I fortsättningen kommer vi därför att använda termen *ledningsstödsystem* i den vanligt förekommande men inte officiellt accepterade innebörden IT-system till stöd för ledning av operationer.

1.1 Det framtida ledningsstödsystemet

Det framtida ledningsstödsystemet är ett kommunikations- och bearbetningssystem som har till uppgift att knyta samman både mänskliga aktörer och “intelligenta” datorsystem med sensorsystem och styrda eller autonoma plattformar.

Förband, sensorer, vapen och samband kommer att behöva kunna kraftsamla snabbt för att möta olika hot, liksom för att genomföra överraskande och improviserade insatser. Detta ställer nya och stora krav på organisation, utbildning och övning. Det innebär också att nya metoder kommer att behöva utvecklas och utnyttjas för att planera och leda underhållsoperationer med kortare ledtider än som nu är normalt. I ett tjugoårsperspektiv kommer detta sannolikt att kunna göras på ett sådant sätt att informationsöverföring, informationskorrelering, situationsanalys, taktiskt beslutsfattande och vapeninsats kan ske med dramatiskt mindre fördröjning än idag.

Det är troligt att antalet högre befälhavare och antalet beslutsnivåer båda kommer att kunna minskas därför att ny informationsteknologi möjliggör att en enskild befälhavare har både överblick och detaljinformation om ett större ansvarsområde. Organisationen kommer att kunna decentraliseras men tekniken möjliggör samtidigt att högre nivåer tar befälet i kritiska situationer som kräver maximal samordning.

---

[1] Två decennier ter sig som ett rimligt tidsperspektiv, å ena sidan därför att Försvarsmakten i sin perspektivplanering nu behöver få tillgång till förutsättningar för ledningssystemutvecklingen fram till år 2020, å den andra därför att Försvarsmakten för 20 år sedan började installera sitt första egentliga ledningsstödsystem, LEO, som då varit under utveckling under större delen av 70-talet.

Nätverksuppbyggnaden ger möjlighet till flexibel vertikal eller horisontell integrering av stabsarbetet och möjliggör införande av distribuerade staber, decentraliserad lokalisering, skadetålig gruppering, adaptiv omgruppering och flexibel sammansättning av ledningsgrupper.

Användaren kommer att se ett starkt ökat tjänsteutbud, men behöver inte veta var i nätet tjänsten finns. Geografiskt avlägsen expertkunskap kommer att kunna utnyttjas för att tolka svårbegriplig eller flertydig information. Information kommer snabbt att kunna sammanställas eller disaggregeras för att passa till den aktuella ledningssituationen. I många fall kommer dock grundläggande tjänster även i fortsättningen att vara knutna till den egna organisationsenheten, bl a därför att sambandet under krig med nödvändighet kommer att vara intermittent och förmåga till autonomt uppträdande därför förblir ett krav.

Ledningsstödsystemet blir också en allt viktigare faktor för att uppnå ökad uthållighet, mindre resursförbrukning och kortare reaktionstider. Användning av datafusionsteknik tillsammans med förmåga att simulera stridssituationer och utnyttjande av beslutsstödsystem och avancerade presentationsformer (t ex *virtuell verklighet*, VR) ger efterhand allt snabbare och större tillgång till adekvat beslutsunderlag. Den gemensamma lägesbilden baserad på modern informationsteknik ger möjlighet till distribuerade staber med hög uthållighet. Genom att utnyttja sådan teknik kan man också öka möjligheterna till realistisk träning i nya situationer.

Information kommer att kunna vara distribuerad i nätverket, och lagras med användning av teknik för distribuerade databaser som medger begränsad men korrekt dataåtkomst trots tillfälliga eller i värsta fall permanenta avbrott i sambandet. Databaserna måste också kunna hantera och presentera multimedia, geografisk information och en mängd andra komplexa datatyper.

I vissa fall ställer tillämpningen krav på nära realtidsöverföring, dvs överföring av information med mycket kort och garanterad maximal fördröjning. Så kan t ex vara fallet vid överföring av sensorinformation från sensor till stridsledningscentral. I andra tillämpningar, t ex vid överföring av bildbaserad underrättelseinformation, kan realtidskraven vara lägre men behovet av att kunna överföra stora datamängder avgörande. Videokonferenser ställer höga krav på såväl realtidsöverföring som överföringskapacitet.

1.2 Den militära beslutsprocessen och den nya informationstekniken

Militära beslutsfattare på taktisk nivå skall ha förmåga att fatta beslut snabbt, så att läget inte hinner förändras i avgörande grad innan en beslutad åtgärd får effekt [4]. Dessutom skall besluten vara effektiva i relation till det aktuella läget likaväl som till målen på längre sikt. Beslutsfattaren måste därför också snabbt kunna skaffa sig den information som krävs för att hantera situationen, och utnyttja den innan läget hinner ändra karaktär. Samtidigt måste förmågan att kritiskt analysera såväl aktuella planer som redan fattade beslut bevaras och helst stärkas.

Moderna ledningsstödsystem innehåller viktiga automatiska funktioner till stöd inte bara för hantering av information utan också för filtrering, sammanställning och tolkning [5].

> ".. Som en följd av detta kan en beslutsfattares datorstöd bli omfattande och det kan ha förmåga att leverera starkt specialiserade och komplexa kunskapsprodukter... Resonemangsstöd är ett område som växer i betydelse i takt med att beslutsfattare ger sig i kast med alltmer komplicerade och tidskritiska beslutssituationer och får tillgång till stora mängder information som kan ge stöd i dessa situationer." [6]

Arbetsmängden och komplexiteten som är förknippad med hantering av inmatad information kan idag vara lika viktiga problem som brist på information. Forskning visar [5] att de tre belastningsfaktorerna komplexitet, osäkerhet och tidspress har likartade effekter på beslutsprocessen: vid måttligt ökad belastning uppstår ofta en förbättring av prestationsförmågan, mer information/ökad komplexitet ger möjlighet till en mer realistisk omvärldsuppfattning, en viss grad av osäkerhet i informationen kan påminna om behovet av att disponera sina resurser för att kunna möta okända risker och krav på beslut inom en given tid kan driva fram ett mer systematiskt och fokuserat arbetssätt. Men över en viss nivå av belastning inträder alltid en försämring av beslutens kvalitet, större ju högre belastningen är. Beslutsfattarens uppmärksamhet riktas alltmer mot själva belastningen och dess negativa effekter på honom själv, andra individer och grupper, och frågan om hur den ska kunna minskas tar över uppmärksamhet från den egentliga beslutssituationen. Kvaliteten på besluten sjunker, och man får mindre självreflektion, mindre planering, mer stereotypa beslut, sämre kontroll över verkställandet, större risktagande, fler brott mot regler och större tendens att undvika eller fly från problem i stället för att ta itu med dem.

Informationsmängden kommer i framtiden att fortsätta att öka, tidspressen kommer att bli än större och osäkerheten kommer att finnas kvar men anta andra former. Istället för brist på kunskap om fiendens gruppering kommer osynliga hot från smygfarkoster och intelligenta vapen, hot från taktiska missiler med extremt kort anflygningstid osv.

Det man måste göra för att motverka dessa hot mot ledningsfunktionen är att intensifiera arbetet med att integrera systemen, att skapa system av system, att anlägga ett mer integrerat synsätt av typ *sensor-to-shooter* på hur system ska byggas upp, samt att för ledningssystemen konstruera representationer som visar operatörens eller beslutsfattarens möjligheter, snarare än sedvanliga enkla lägesavbildningar. Sådana traditionella representationer degenererar lätt till ett myller av fientliga och egna målangivelser, utlagda på en karta vars underliggande informationsskikt snart skyms av den snabbt ökande rapportmängden. I dagens ledningsstödsystem på hög nivå saknas dessutom i regel automatisk målföljning och lägesuppdatering, vilket bland annat innebär att informationens kvalitet merendels är mer eller mindre okänd.

1.3 Informationskvalitet

Eftersom det framtida ledningsstödsystemet kommer att behöva hantera en stor mängd information som är resultat av beräkningar baserade på delvis osäkra grunddata, kommer det att bli allt viktigare att kvaliteten på beslutsunderlaget kan utvärderas av systemet och presenteras så att det korrekt kan uppfattas av användaren [5].

Om uppgiften enbart är att beskriva kvaliteten hos innehållet i en databas med grunddata är lösningen förhållandevis enkel: man får kräva att dataleverantören ger ett kvalitetsmått av specificerat slag tillsammans med sina data samt att han dokumenterar den process som har använts för att producera kvalitetsmåttet. Detta kvalitetsmått måste sedan presenteras tillsammans med den efterfrågade informationen på ett

begripligt sätt. Det sistnämnda är inte alltid lätt att åstadkomma, men en rad s k visualiseringsmetoder för kvalificerad grafisk presentation utvecklas i snabb takt och kan komma till nytta här.

När man vill tillgodose rimliga kvalitetskrav vid utnyttjande av beräkningsresultat är problemet betydligt svårare: dels behöver man förse alla indata till den aktuella beräkningsmodellen med tolerans- och andra kvalitetsmått, dels måste man konstruera modellen på ett mer sofistikerat sätt än som idag är normalt. Den måste från början förses med inbyggda, konsekvent genomförda, feluppskattningar och bygga på noga genomtänkta giltighetskriterier. Sådan djupgående och rigorös analys av beräknings-modellers noggrannhet och precision har länge varit idealet (om än inte alltid praktisk verklighet) inom metodvetenskaper för vetenskapliga beräkningar men den har sällan tillämpats i militära simuleringar. Utvecklingen har dock visat att systematisk veri-fiering och validering är en nödvändig metodik för kvalitetssäkring.

Svårast är att korrekt värdera och kombinera genuint osäker information, t ex underrättelseinformation, så att användaren får hjälp att dra väl underbyggda slutsatser. Det ligger i sakens natur att sådana slutsatser sällan är entydiga, och ett exempel på forskningsproblem är att korrekt hantera hela uppsättningar av möjliga alternativ så att de maskinella systemen kan erbjuda sina användare ett besluts-underlag som är både realistiskt och begripligt. Inte minst militära behov har bidragit till att den internationella forskningen på detta område idag är intensiv, och sådan metodik studeras nu i ökande grad med sikte på tillämpning i ledningssystem.

För att användarna skall kunna förstå och snabbt uppfatta analysresultaten trots de komplicerade mått och metoder som kommer att behöva användas i många fall, kommer det också att krävas förändringar i utbildningen av ledningspersonal, mot ett större inslag av formell, matematisk metodik för analys av osäker, vag och mot-sägelsefull information. Även om människor genom den biologiska evolutionen utrustats med mycket effektiva mentala processer för att kunna fatta beslut under osäkerhet, är denna förmåga begränsad till kvalitativ bedömning av ett förhållandevis litet antal faktorer. Utvecklingen av formella kvantitativa metoder öppnar därför successivt nya möjligheter, som det gäller att lära sig utnyttja och behärska om man vill bli vinnare på framtidens slagfält.

1.4 Hoten mot ledningssystemet

De civila kommunikationssystemen har stor redundans, men är ofta sårbara och måste kompletteras med militärt motiverade säkerhetsskydd. Ekonomiska transaktioner i nätverk kräver hög säkerhet, och civil nätverksteknik bör till stor del kunna utnyttjas även för militära kommunikationsbehov. Men t ex kvalificerat sabotage har i regel inte ansetts vara ett tillräckligt stort civilt hot för att motivera att den civila tekniken förses med ett skydd mot detta, som är tillräckligt säkert även ur militärt perspektiv.

Samtidigt som den informationsteknologiska utvecklingen ger upphov till nya möjligheter kommer ett allt djupare beroende av störningsfri tillgång till den nya tekniken att uppstå. Det faktum att en mängd nya påverkanstyper är möjliga innebär dock inte nödvändigtvis att de är oundvikliga. I stället har vi som vanligt en duell-situation mellan verkan och motverkan, där den sida som har de bästa tekniska resurserna och de mest robusta lösningarna bör dra det längsta strået.

Talesmän för USAs krigsmakt har i olika sammanhang framhållit att man vill utnyttja sitt informationstekniska övertag över ekonomiskt och tekniskt mindre utvecklade motståndare och samtidigt gardera sig mot de nya sårbarheter IT-utvecklingen skapar. Begreppen *ledningskrigföring* (*command and control warfare, C2W*) och *informationskrigföring* (*information warfare, IW*) har utvecklats för att täcka denna doktrin.

Ledningskrigföring handlar om hur man kan angripa och slå ut motståndarens ledningsfunktioner på alla plan, respektive skydda sina egna. Informationskrigföring är ett samlingsnamn för taktik och metodik för att störa motståndarens, respektive skydda den egna sidans, verklighetsuppfattning, inklusive de informationstekniska komponenter som är avsedda att stödja denna, som informationslager, informationsöverförings- och informationsbehandlingssystem. I båda fallen rör det sig om metoder och teoribildningar som spänner över mycket stora områden och som inkluderar såväl tekniska som doktrinmässiga, sociologiska och psykologiska frågor.

1.5 Ledningssystemet och den tekniska utvecklingen inom IT-området

För att kunna övergå till en nätverksbaserad systemarkitektur som möjliggör evolutionär utveckling kommer försvaret att behöva ta den civila teknologiutvecklingen till hjälp, inte bara när det gäller programsystem- och datorutveckling, utan även när det gäller att åstadkomma ett integrerat, rörligt samband samt ett avancerat åtkomstskydd. Den civila tekniken erbjuder ramverk och vissa generella komponenter när det gäller såväl maskinvara som programvara, men dessa måste kombineras med speciella programvarulösningar, dels för att skapa de tillämpningsfunktioner som är specifika för försvarets behov, dels för att uppnå erforderlig säkerhet och skyddseffekt. När det gäller de mobila kommunikationslänkarna i ledningssystemet kommer kanske även maskinvaran delvis att behöva vara specialkonstruerad, men civila komponenter och delsystem bör kunna utnyttjas i betydande omfattning även här.

Det finns naturligtvis vissa skillnader mellan kravbilderna för militära och civila lednings- och informationssystem. För att vara till nytta i komplexa stridssituationer behöver de militära systemen:

- stor flexibilitet
- stor dynamik, extrem snabbhet vid behov
- god funktion även i situationer där användarna upplever stark stress
- förmåga att hantera osäkra och falska indata
- hög datasäkerhet
- skydd mot medveten förstörelse och störning
- förmåga till gradvis degradering (graceful degradation)

De flesta civila ledningssystemtillämpningar ställer väsentligt lägre krav i dessa avseenden, och inga civila system uppvisar, såvitt vi känner till, alla dessa krav samtidigt.

IT-utvecklingen sker mycket snabbt, dels på grund av de stora ekonomiska vinster som bättre "generisk", dvs brett och generellt användbar, teknik kan ge, dels på grund

av den enastående utvecklingsbarhet som mikroelektroniken, baserad på integrerad-krets-teknik, visat sig besitta.

Prestandaförbättringen hos kommunikationssystemen och då framförallt datornät är en annan stark drivkraft, men här har utvecklingen bara börjat. Börsutvecklingen under 1999 har varit explosionsartad för många s k IT- eller webföretag och även om den enligt många bedömare varit kraftigt överdriven, visar den att förväntan är enorm på informationsteknikens praktiska och ekonomiska effekter i en nära framtid.

Denna rapport kommer inte att vidare beröra den tekniska utvecklingen på kommunikationsområdet, dels därför att vi anser att beskrivningen i förstudierapporten [1] i allt väsentligt fortfarande är giltig, dels därför att FOA numera driver specialiserade tekniska idéprojekt inom mobila datanät [7], och dels slutligen därför att de genomgripande konsekvenserna av ett globalt och homogent nätverk för kommunikation under de senaste två åren blivit det troligen mest omskrivna och omtalade tekniska fenomenet i industrisamhället och vi därför kan ta mycket i denna utveckling för givet när vi diskuterar arkitektur för framtidens ledningssystem.

1.6 Problem i tidigare utvecklingsprojekt

Försvarsmakten har arbetat länge med utvecklingen av ett nytt ledningssystem. Problemen har varit stora, inte minst då det gäller att etablera en fungerande utvecklingsorganisation och att finna former för rationellt kompetensutnyttjande. F n förbereds en omstart och vi vill hävda att Försvarsmakten därmed har ett unikt tillfälle att utnyttja pågående paradigmskiften inom såväl civil informationsteknologi som militär ledning.

I en rapport från 1996 [8] pekade Riksrevisionsverket (RRV) ut en rad problem som behövde lösas för att utvecklingen av det nya ledningssystemet skall bli framgångsrikt:

- skapa *gemensamma tekniska förutsättningar*: det saknas för närvarande en tydlig informationsarkitektur som visar hur informationsutbyte mellan olika system ska fungera, en gemensam arkitektur för att kunna styra och följa den komplexa samordningsprocessen, en säkerhetsarkitektur som omfattar regelverket för hur skyddsfunktioner ska specificeras och konstrueras, en klargörande beskrivning av grunddataförsörjningen för att säkerställa att korrekta data skapas, vidmakthålls och distribueras på rätt sätt samt en gemensam begreppskatalog och gemensamma standarder, nödvändiga för att Försvarsmaktens ledningssystem ska kunna fungera
- *förvaltningsstrategi och systemförvaltningsmodell*: utan en gemensam förvaltningsmodell är projekten tvingade att leva kvar som förvaltnings-organisation, vilket medför större kostnader än som egentligen vore nödvändigt
- *utvecklingsmodell*: en för Försvarsmakten gemensam utvecklingsmodell skulle göra utvecklingsprojekten jämförbara; gemensamma begreppsdefinitioner skulle möjliggöra övergripande bedömningar av utvecklingsläget
- *samverkan mellan projekten*: projekten har till största delen löst egna problem på egna sätt; samverkan avseende en gemensam begreppskatalog och gemensamma meddelandetyper för att underlätta utvecklingen av verksamhetsbaserade system och informationsutbyte via s k CAMA-avtal har förekommit i mycket liten

omfattning; däremot har frågeställningar kring Försvarsmaktens gemensamma grundsystem varit föremål för diskussioner som tagit mycket tid och kraft

Vi ger i avsnitt 3.2.1 exempel och synpunkter på val av utvecklingsmetodik och utvecklingsmodell men tror för övrigt inte att utarbetande av sådana nya standarder eller ”modeller” är en lämplig uppgift för FOA. Däremot anser vi att FOAs informationstekniska och operationsanalytiska kompetens bör utnyttjas för att värdera förslag till nya utvecklings- och förvaltningsmodeller. Det klassiska svenska remissförfarandet vore utan tvekan värt en renässans på detta område.

Vi anser också att utvecklingen har gett dem rätt som gentemot en tidigare dominerande trend av användarfokusering, inte bara i form av delaktighet utan i form av totalt utvecklingsansvar, hävdat att ledningssystemutveckling är en utpräglat högteknologisk aktivitet, som kräver professionell teknisk-vetenskaplig kompetens på hög nivå för att lyckas. Att tillgång till och inflytande för sådan kompetens är ett nödvändigt men inte tillräckligt villkor för framgångsrik systemutveckling borde inte behöva sägas. Varaktig framgång kommer endast den organisation till del som förmår att integrera alla de kompetenser och erfarenheter som behövs för denna komplexa verksamhet. Hit måste, förutom den militära ledningspersonalen själv, också räknas försvarsanalytiker, kognitions- och beslutsforskare, organisationsexperter, dataloger och systemarkitekter och inte minst tillämpade matematiker med inriktning mot informationsfusion, beslutsstödsmetodik och planeringsmetodik.

1.7 Arkitektur för ledningssystem

Det har visat sig att tolkning och användning av begreppet systemarkitektur i samband med forskning om försvarets framtida ledningssystem är både omtvistad och flertydig. Detta kan tyckas vara naturligt eftersom begreppet ledningssystem inom försvarsmakten inte enbart avser tekniska system, än mindre enbart programsystem, utan också innefattar doktrinmässiga, organisatoriska och personella aspekter på ledning av stridskrafter.

En del forskare (se t ex [9]) anser att informationssystemarkitektur egentligen inte handlar om processen att bygga system utan om att ställa krav på hur systemen skall vara beskaffade. De centrala egenskaperna är enligt denna uppfattning: struktur och avgränsning, innehåll (funktion och data), relation mellan system, relation till verksamheten och migration/förändring.

Men man kan ju inte ställa rimliga krav om man inte vet vad som är möjligt och rimligt att kräva. Detta är redan ett stort problem i en situation där inte ens specialister alltid kan följa med den snabba utvecklingen. En konstruktion blir naturligtvis inte heller bra om den enbart utgör ett konglomerat av delar som olika användargrupper i en organisation ansett nödvändiga. Konstruktörens eller systemarkitektens uppgift kommer att vara lika viktig som någonsin, men förskjuts från att lägga fast systemets hela utformning en gång för alla, till att på användarnas uppdrag initiera, planera och underhålla en ständigt växande struktur.

Eftersom ett informationssystem omfattar inte bara programvara utan också datalager och dataflöden, maskinell utrustning, och framför allt, verksamhetsprocesser, behöver arkitekturen hos ett informationssystem i regel beskrivas och diskuteras på flera

nivåer och från flera olika perspektiv. Frågeställningarna på de olika nivåerna är mycket olikartade men är ändå i ett effektivt fungerande, verksamhetskritiskt informationssystem starkt kopplade till varandra.

Inom *US Department of Defense* (US DoD) talar man om tre huvudnivåer för begreppet arkitektur, som har beskrivits på följande sätt:

Operational Architecture (OA) - is the total aggregation of missions, functions, tasks, information requirements, and business rules.

Technical Architecture (TA) - is the 'building code' upon which systems are based.

Systems Architecture (SA) - is the physical implementation of the OA based on the TA, and also the layout and relationship of systems and communications.

Den operationella arkitekturen, eller *verksamhetsarkitekturen* som det väl hellre borde heta på svenska, ligger alltså närmast det begrepp vi nyss diskuterat, medan vi i denna rapport i fortsättningen huvudsakligen behandlar sådana principiella aspekter på *systemarkitektur* och *teknisk arkitektur*, som är nödvändiga förutsättningar för att kunna uppnå verksamhetsmålen. I definitionerna ovan tänker man sig ett specifikt system som beskrivs ur tre olika perspektiv. Systemarkitektur som framväxande kunskapsområde handlar snarare om olika metoders och teknologiers möjligheter och begränsningar i relation till identifierade behov av generella funktioner och kravbilder.

En risk med denna uppdelning i tre olika skikt är att tillräcklig informationsteknisk grundkompetens kanske inte får påverka valet av verksamhetsprocesser. I verkligheten är det en nära samverkan mellan verksamhetskompetens och teknisk modelleringskompetens som kan leda till nya, effektiva tillämpningar av informationstekniken.

I en *nätverksbaserad arkitektur* är sambandstekniken helt integrerad i systemarkitekturen. Detta innebär att sambandet bildar en separat funktionsnivå, så att olika sätt att utföra sambandet kan utnyttjas samtidigt, utan att högre nivåer - och mänskliga användare - ser någon annan skillnad än att överföringshastigheten kan variera med uppgiften och över tiden. Ett avancerat åtkomstskydd baserat på moderna krypterings- och autentiseringsmetoder, behörighetsklassning och identifieringsteknik är också en nödvändig del av ett ledningsstödsystem. Slutligen behöver noderna i nätverket ha tillräcklig autonom kapacitet för att en meningsfull aktivitet skall kunna upprätthållas även om sambandet med högre staber skärs av, fast ledningsförmågan i så fall naturligtvis reduceras.

En sådan arkitektur har förutsättningar för att kunna utvecklas successivt och evolutionärt utan att förlora kontakten med den mycket snabba utvecklingen av informationstekniken. Den bör också kunna göras tillräckligt flexibel för att överleva övergång till framtida ledningskoncept.

1.8 Omfattande utvecklingsinsatser kommer att krävas

Utveckling av avancerade tillämpningsorienterade funktioner i ledningsstödsystemen kommer även i framtiden att vara komplex och resurskrävande. Teknikutvecklingen medför därför inte ett minskat investeringsbehov för IT-system, däremot högre krav

på *modularisering* och *standardisering* av framförallt gränssnitt mellan komponenter och delsystem. Den medför sannolikt också att utvecklings- och införandearbetet behöver bedrivas mer eller mindre kontinuerligt, dvs i form av mindre, delvis parallella och varandra avlösande projekt, i stället för som hittills i stora, tidsmässigt avgränsade projekt med relativt långa tidsmellanrum.

Även om militär användning av informationstekniken i hög grad kommer att baseras på allmänt tillgänglig civil teknik, kan man förvänta ett betydande "konkurrenstryck" beträffande utveckling av såväl försvarsanpassad arkitektur som militärspecifika tillämpningar, och de försvarsmakter som väljer att stå vid sidan om denna förvandling tar en uppenbar risk att hamna i underläge om en militär konflikt skulle uppstå.

Den svenska forskningen om ledningssystemteknik står inför stora och förändrade krav. Fastän vi ännu inte sett stora nysatsningar eller omfattande FoU-projekt (med ett viktigt undantag, den uppbyggnad av IT-säkerhetsforskning som skett under de senaste åren vid FOA) sprider sig nu insikten att ett tekniskt avancerat ledningssystem kommer att vara en fundamental del av framtidens militära infrastruktur. Dessutom har det visat sig vara en del som det är svårt att anskaffa genom direkt upphandling på den civila IT-marknaden.

2. Funktioner i ledningsstödsystemet

I den pågående FOA-ledda studien LUST (*LedningsUtvecklingsSTudie*) har man studerat en framtida struktur för decentraliserad ledning över landet med en centraliserad underrättelsebataljon i varje militärdistrikt.

Enligt LUST skall insatsstyrkans underrättelsebataljon inom sitt ansvarsområde understödja flera parallella ledningsfunktioner (staber) och verkanssystem. Ett stort regionalt sambandsbehov föreligger i samband med överföring av sensorinformation. Övrigt sambandsbehov för utväxling av lägesinformation och order bedöms bli förhållandevis måttligt.

Underrättelsebehoven ställs dels från Högkvarteret (främst i fred) och dels från de ledningsstaber som underrättelsebataljonen betjänar. Underrättelsebataljonen svarar för sammanställning av en aktuell inhämtningsplan där områdets spaningsresurser allokeras så effektivt som möjligt. Den svarar också för kontinuerlig multisensor- och informationsfusion som resulterar i lägesbilder i nära realtid, för bearbetning av specifikt inhämtad information mot bakgrund av aktuell, predikterad eller historisk lägesbild och information i underrättelsedatabaser samt för delgivning av generella bedömningar och svar på specifika underrättelsefrågor.

Förutom underrättelsefunktionen, som skall svara för informationen om motståndaren och därmed en stor del av den gemensamma lägesbilden (se avsnitt 2.2), måste ledningsstödet också betjäna en rad andra funktioner, inte minst lägesrapportering och planering av alla den egna sidans resurser, men också bl a beslutsstöd för planering av egna operationer (avsnitt 2.5).

I fortsättningen av detta kapitel ger vi en översikt över koncept och metodik som behövs för att uppnå dessa målsättningar för det framtida ledningsstödsystemet.

2.1 Modellbaserad ledning: en vision för framtidens ledningsstödsystem

I ett bidrag till *1995 Symposium on Command and Control Research and Technology* presenterar Sam Chamberlain vid *US Army Research Laboratory* konceptet *Model-Based Battle Command* (MBBC) [10]. Författaren beskriver och kritiserar den rådande paradigmen för ledningsstödsystem som *meddelandebaserad* och *kommunikationsintensiv*, och hävdar att det är dags att övergå till *modellbaserad* ledning som grundfilosofi för konstruktion av ledningsstödsystem. Fastän man vid utveckling av ledningsstödsystem länge har utnyttjat modellering och simulering som en del av anskaffningsprocessen, görs fortfarande en fundamental åtskillnad mellan sättet att *utföra* och sättet att *modellera* ledning. Fastän ett stort antal modeller och system för simulering av ledning har utvecklats, har dessa system endast rollen av studie- och stödsystem och utnyttjas inte som integrerade delar av ledningsstödet.

Med modellbaserad ledning menar Chamberlain ett ledningsstöd som utnyttjar koncepten modellering och simulering för att *representera* och *påverka* läget på slagfältet, så att utövande av ledning blir i tekniskt avseende identiskt med simulering av ledning, fastän med utnyttjande av verkliga data i realtid. Skillnaden mellan simulering av ledning och utövande av ledning kommer efterhand att minska, för att till slut enbart bestå av att man vid bemannade ledningsövningar använder sig av såväl

verkliga som simulerade styrkor, medan man i en krigssituation modellerar verkliga styrkor när man vill representera det faktiska läget, men ”skarvlöst” kan övergå till simulerade styrkor när man vill studera konsekvenser av möjliga framtida händelser och hypotetiska beslut.

Författaren hävdar att man fortfarande (alltså 1995) huvudsakligen utformar kommunikationen mellan noderna i ett ledningssystem efter principer som har tillkommit för att åstadkomma standardiserad och robust kommunikation mellan människor. Han nämner som exempel att ett nytt format, kallat *Variable Message Format* (VMF), nyligen utvecklades av US Army för användning i de s k *Force XXI*-övningarna. Dessa textmeddelanden är direkt överförda från standardprocedurer för t ex begäran av eldtillstånd, och i huvudsak samma textformat används för att överföra dessa meddelanden mellan olika ledningssystemnoders databaser, trots att för människor utvecklade meddelandeformat inte lämpar sig särskilt väl för detta ändamål. Orsaken till att man inte utvecklat ett mer databasanpassat format är att utformning av datorabstraktioner för komplexa mänskliga begrepp är en både arbetskrävande och intellektuellt svår uppgift. Konsekvensen blir att ledningsstödsystemen inte byggs upp som objektmodeller utan enbart utnyttjas som en sorts avancerade telegrafimaskiner, tycks Chamberlain mena.

I vilken grad denna brist gäller även andra länders arméledningssystem vet vi inte, men vi noterar att t ex det nya holländska ISIS-systemet har byggts upp kring en informationsmodell baserad på *ontologin* (begreppsapparaten) ATCCIS, som utvecklats av NATO (se avsnitt 3.2.3), och ISIS bör därmed ha en betydligt bättre informationsmodell än ett system som enbart lagrar textmeddelanden i sina databaser. Som framgår av avsnitt 4.1.2 har även US Army numera beslutat övergå till en gemensam informationsmodell för sina ledningsstödsystem, *Joint Tactical Data Model* (JTDM).

Chamberlain vill alltså se ledningsstödsystemen uppbyggda kring informationsmodeller i en distribuerad databas (i exemplen tänker han sig en relationsdatabas), med vars hjälp en sammanhängande logisk struktur åtminstone i princip kan uppnås. Kommunikation mellan olika noder kan då skötas automatiskt av det distribuerade databassystemet genom s k *replikering* (se avsnitt 4.3.2). Frågan om val av format för meddelanden mellan noder i systemet övergår till frågor om hur databasreplikering skall gå till, ett utpräglat tekniskt problem på databassystemnivå. Om databasleverantören har en bra generell lösning på replikeringsproblemet så behöver konstruktören av ledningsstödsystemet inte bekymra sig om hur replikeringen ska gå till. Men om ledningsstödet förutsätter kommunikation mellan olika typer eller fabrikat av databassystem så kommer replikeringsmekanismen med dagens kommersiellt tillgängliga teknik inte att fungera.

Denna modellbaserade syn på ledningsstödsystem kan tyckas elementär och självklar, men är i själva verket revolutionerande om den dras ut till sina yttersta konsekvenser, vilket Chamberlain själv gör när han säger att målet är att ledning skall ske modellbaserat. Visionen innebär nämligen att all materiel och all information som är av betydelse för ledningen på slagfältet i realtid skall representeras i en distribuerad realtidsdatabas med en homogen informationsmodell. I avsnittet om medlarteknik beskriver vi hur detta i en nära framtid kan göras utan att hela systemet måste konstrueras samtidigt eller med samma databasteknologi för varje datakälla.

Visionen om modellbaserad ledning påminner om de tankar på modellbaserad målrepresentation som uttryckts av DARPA-projektet *Dynamic DataBase* (DDB) [5]. Tre mål har ställts upp för DDB-programmet:

1. att effektivt kunna lagra all väsentlig stridsområdesinformation och ge tillgång till alla sensorobservationer
2. att använda de resulterande sensorhistorikerna för att identifiera och fokusera användarens uppmärksamhet på väsentliga händelser i stridsområdet
3. att kunna skapa och över hela det distribuerade stridsområdet synkronisera lokala situationsestimat.

För att uppnå dessa mål krävs enligt DARPA utveckling av databasteknik och databastjänster som kan hantera, ställa frågor till, underhålla och bearbeta sensordatahistoriken. Både ny och redan existerande databasteknik skall studeras och prövas ur detta perspektiv. Till de tekniskt mest avancerade systemkraven i denna tillämpning hör förmåga att besvara *oskarpa spatiotemporala* databasfrågor.

Den dynamiska situationsmodellen skall ge en konsistent representation av viktiga, ur sensorsynpunkt relevanta, aspekter av läget på stridsfältet. Innehållet i denna modell kommer att organiseras i flera nivåer som omfattar utvalda sensorrådata, sensormeta-data, attribut och probabilistisk attributinformation härledd från sensordata, terräng-data, omgivningstillstånd, objektlokaliseringsdata, hypoteser om objektens identitet och tillstånd, hypoteser om egna och fientliga luft-, land- och sjöstyrkor samt "släkt-träd" för data.

I projektet skall utvecklas metoder för att upprätthålla logisk konsistens mellan situa-tionsnivåerna, sådana att konflikter mellan hypoteser beträffande terräng, objekt och styrkor minimeras. Slutmålet är att åstadkomma ett gemensamt ramverk för samman-ställning och integration av dynamisk mållägesinformation från distribuerade källor.

Eftersom denna tekniskt avancerade plan för ett integrerat sensorfusions- och läges-bildsystem motsvarar ett enda av flera komplexa delsystem i Chamberlains MBBC-vision, ser vi att denna är ett mål på lång sikt. Bland annat ISIS-projektet visar att system redan idag kan byggas på principer som är förenliga med MBBC-visionen, även om det alltså finns många och stora steg att ta innan den är fullt realiserad. MBBC bör dock enligt vår uppfattning vara ett centralt långsiktigt arkitekturmål, och viktiga delar av den teknik vi kommer att beröra har valts ut därför att den har poten-tial att underlätta evolutionär utveckling av ledningsstödsystem som efterhand kan närma sig idealen i denna vision.

2.2 Den gemensamma lägesbilden

Många bedömare anser att striden i framtiden kommer att föras i avgränsade men inte nödvändigtvis sammanhängande områden över hela det berörda territoriet. Varje stridsenhet måste då kunna ges en bild av stridsläget som är förenlig med alla andra enheters. Målbekämpning skall kunna överlåtas från en styrka till en annan under pågående stridsförlopp.

Ett starkt behov kommer att finnas av system som medger att läget på marken kan detaljföljas och distribueras till de stridande enheterna på motsvarande sätt som sker

idag i luften och till sjöss. Därmed skapas förutsättningar för att koppla samman hittills separata ledningssystem som utvecklats för att betjäna olika delar av organisationen med varandra, och därmed uppnå en länge eftersträvad förmåga till integrerade operationer, där mark-, sjö- och luftstridskrafter deltar i nära samverkan. Denna vision brukar kallas *den gemensamma lägesbilden*. Exakt vad detta begrepp skall innebära är inte klarlagt, men med lite eftertanke inser man att innebörden är långt mer komplex än namnet ger intryck av.

Många sensorer av olika slag kommer att kunna samverka för att möjliggöra bättre målföljning och identifiering. De metoder som framför allt behöver utvecklas och tillämpas för att åstadkomma detta brukar gemensamt kallas *multisensorfusionsmetoder* [11]. Dessa metoder kan ge en noggrann beskrivning av egna och fientliga objekts position och rörelsevektor i stridsrummet, tillsammans med en precis och tillförlitlig klassificering och, åtminstone när det gäller egna objekt, identifiering. En ögonblicksbild av denna dynamiska beskrivning av objekten på stridsfältet kommer vi att kalla *mållägesbild*. Självfallet är mållägesinformationen egentligen inte en bild utan snarare en dynamisk, geografisk (*spatio-temporal*) informationsmodell eller databas.

Den gemensamma lägesbilden innehåller emellertid betydligt mer information än mållägesbilden. För det första måste lägesbilden anpassas till sina användare på olika nivåer i organisationen, eftersom en beslutsfattare på hög nivå vanligtvis behöver se läget över stora geografiska områden med symboler som identifierar förband på hög nivå och vice versa för en chef på låg nivå. Förbandsinformationen måste alltså sammanställas eller *aggregeras* för att passa till olika användares behov. På motsvarande sätt behöver den geografiska informationen naturligtvis anpassas till den skala som användaren behöver, och i moderna geografiska informationssystem strävar man efter att göra detta *dynamiskt* allteftersom skalan förändras. Kartinformationen skall kunna *generaliseras* på ett sätt som är anpassat till presentationsskalan, vilket kräver utveckling av automatiska, ”intelligenta” generaliseringsprogram.

För det andra är lägesbilden egentligen en metafor som dels omfattar all potentiellt användbar och insamlingsbar tillståndsinformation om egna och fientliga förband, dels också kan innehålla bedömda eller beräknade hot och möjligheter förknippade med en given situation.

För det tredje kan lägesbilden också omfatta miljöbeskrivningar, väderläge, demografisk information om eventuell civilbefolkning, läget för andra relevanta civila resurser etc, kort sagt all information om stridsfältets omgivning som anses relevant för ledning av striden och som kan anskaffas och hållas tillräckligt aktuell. Åtminstone då det gäller det egna territoriet är det med all säkerhet inte tillgången till data som begränsar användbarheten, utan ledningsstödsystemets förmåga att modellera och presentera situationen på ett sätt som svarar mot användarnas behov.

Sist men inte minst önskar man antagligen att all denna information skall hålla tillräcklig kvalitet i alla situationer, något som dock förefaller omöjligt att åstadkomma. Istället måste man nöja sig med att insamlad information är av känd kvalitet och att användning av grundinformationen sker med hänsyn till denna kvalitet, så att inga slutsatser dras och inga beräkningsresultat presenteras för vilka resultatkvaliteten är okänd eller alltför låg för att vara meningsfull. Redan denna målsättning måste

betraktas som mycket hög och för att uppnå den kommer ett omfattande forsknings- och utvecklingsarbete att krävas. En intressant fråga är därför om och i så fall när ett system kan användas fast det inte uppfyller ett så högt kvalitetskrav. Det uppstår behov av en användarutbildning som tar sikte på att användare ska kunna ”genomskåda” systemet när det presenterar ett analysresultat som inte håller måttet. Tyvärr är det inte lätt att säga hur en sådan utbildning skall vara beskaffad, men säkert är att den måste byggas på matematisk-naturvetenskaplig grund.

2.2.1 Datafusion

Den snabbt ökande förmågan att dynamiskt mäta/observera, samla in och lagra information leder till allt större behov av automatiska system som kan extrahera och kombinera relevanta data från multipla och disparata källor till konsistent och så fullständig information som möjligt. Denna information skall ge stöd för, eller i vissa fall ersätta, mänskliga beslut.

Datafusion är den metodmässiga basteknologin för sådana system. Det är ett multidisciplinärt ämnesområde som omfattar en hierarki av processer, till vilka hör mätning/observation, signalbehandling, egenskapsextrahering, mönsteranalys, situationsvärdering, beslutsfattande, återkopplade styrprocesser till stöd för olika delprocesser som t ex aktiv datafångst eller nätverksstyrning. Till området räknas idag en rad tekniker som tidigare betraktats som separata forskningsområden, även om det naturligtvis inte råder någon fullständig enighet om hur gränserna ska dras: detektion, målföljning, objektigenkänning, situationsanalys, hotanalys, sensorplanering m fl.

2.2.2 Taktisk informationsfusion

Informationsteknik (i första hand programvara) som stöder uppgiften att i realtid skapa en korrekt och för den operativt/taktiska beslutsfattaren adekvat lägesuppfattning, brukar kallas system för *informationsfusion* [5]. Begreppet täcker de delar av datafusionsprocessen som är avsedda att ge beslutsfattare och analytiker stöd i arbetet med planering och genomförande av underrättelseinhämtning samt analys av underrättelser från alla källor, dvs processer som ligger närmare underrättelsebearbetning än sensordatabehandling och mållägesberäkning.

Den allt snabbare takten i modern krigföring kräver att informationsinhämtning, underrättelseanalys och fördelning av spaningsresurser sker snabbt och integrerat. Resultaten av inhämtning och analys måste kunna göras omedelbart tillgängligt till de stridande enheter som behöver dem. Mängden sensordata har redan nått den punkt där automatisering blir nödvändig, eftersom enbart mänskliga operatörer inte längre räcker till för att utvärdera all den information som finns tillgänglig på slagfältet.

Spanings- och analysresurser, liksom resultat i form av läges- och prognosinformation, måste kunna efterfrågas från alla nivåer. I dialog med sina användare behöver systemet kunna föreslå goda lösningar på de resurskonflikter som kommer att uppstå. Det skall fungera även då sambandet har låg kapacitet eller är intermittent, om än nödvändigtvis med reducerad kvalitet hos resultatet. Möjligheterna till informationsfusion är alltså nära förbundna med tillgång till ett heltäckande informationsnätverk. Utan ett sådant nätverk kan varken indata till eller resultat från fusionsprocessen i tid nå de adressater som behöver den.

Informationsfusionen syftar till att utnyttja samspelet mellan sensorinformation, omgivningsinformation och kända egenskaper hos motståndarens plattformar, förband, organisation och doktrin för att underlätta och höja kvalitén i arbetet med att värdera och prognosticera fiendens förmåga och avsikter på kortare och längre sikt.

Karakteristiskt för fusionsprocessen på taktisk nivå är att målet eller mätobjektet i princip är hela den observerade mängden fientliga förband, alltså en mycket komplex och föränderlig verklighet. Resultatet skall levereras i form av korrekt och kvalitetsmärkt tillstånds- och prognosinformation som kan uppfattas av en beslutsfattare eller i vissa fall av ett automatiskt system, och som är avpassad för att hjälpa denne eller detta att i tid lösa vissa givna typer av problem, vanligen något eller några av uppgifterna identifiering, lokalisering och positionsprognosticering, hotvärdering och val av motåtgärder.

För att detta ska kunna ske behövs:

- mätvärden av känd och tillräckligt hög kvalitet (med avseende på bl a mätnoggrannhet, risk för felaktig detektion, korrekt karakterisering eller identifiering) och möjliga att uttrycka i ett gemensamt referenssystem
- adekvat information om den omgivande miljön, som påverkar mätvärdena och deras tolkning, liksom information om målets möjligheter att förflytta sig eller på annat sätt ändra tillstånd
- modeller av alla relevanta ingående processer, som kan göra det möjligt att med mätvärden och aprioriinformation som indata, i realtid räkna ut det man vill och kan veta om målets nuvarande och framtida tillstånd
- sist men inte minst, åtgärdsval med ledning av den vunna insikten.

Informationsfusion baseras på tillgång till en gemensam mållägesbild, i form av etiketterade målspår eller liknande, som är resultat av tidigare steg i fusionsprocessen. Med utgångspunkt från mållägesbilden genomförs modellering, simulering och resultatanalys i realtid av sammansättning, beteende, målstyrka och signaturer hos fientliga förband.

För att göra detta behöver systemet känna till relevanta egenskaper hos egen underrättelseledning, tillgängliga sensorer och den omgivande miljön, liksom växelverkan mellan dessa faktorer.

De viktigaste delprocesserna är:

Situationsanalys
Resultatet från lägre nivåers datafusion överlämnas till ett stödsystem för situations- och hotanalys. Vid situationsanalys identifierar man den situation som har orsakat observerade data och händelser. Här genererar man också en mängd alternativa hypoteser angående den aktuella situationen. Hypoteserna utvärderas med hänsyn till observationerna och tilldelas sannolikheter. Målet för analysen är att finna, karakterisera och rangordna de mest sannolika hypoteserna. Situationsanalysen är en kronologiskt ordnad process där data anländer över tiden och analysen successivt förfinas mot en allt mer detaljerad och säker bästa hypotes.

Hotanalys
Hotanalys är en process med flera olika perspektiv. Här analyseras risker och möjligheter för de egna styrkorna att möta motståndaren på ett effektivt sätt. Resultatet av den tidigare situationsanalysen kombineras med erhållna indikationer om motståndarens avsikter samt förhandslagrad information från tekniska och doktrinbeskrivande databaser för att leverera en fullständig hotanalys.

Adaption
Adaption betecknar en kategori av processer, som syftar till att förbättra resultatet av en datafusionsprocess genom styrning av insamlings- och tolkningsprocesser, med utgångspunkt från identifierade brister i det producerade resultatet, dvs en generell form av återkoppling. En sensorfusionsprocess kan t ex ha resulterat i alltför vida felgränser för positionen hos ett mål. I en adaptionsprocess identifieras detta problem och tillräckliga sensorresurser för att bringa ned felet till godtagbar nivå tilldelas.

Beslutsfattande
Som resultat av situations- och hotanalyserna levereras ett antal tänkbara handlingsalternativ för de egna styrkorna. Beslutsfattaren kan direkt observera både den genomförda situations- och hotanalysen samt studera de mest lovande föreslagna handlingsalternativen inför sitt beslutsfattande.

2.2.3 Miljöbeskrivningar och geografisk informationsbehandling
De mätningar som ligger till grund för den maskinella verklighetstolkning som informationsfusion är, kommer fusionsprocessen tillgodo på i princip två vägar: i första hand direkt i realtid från de sensorer som den militära styrkan själv förfogar över, men i växande utsträckning också i form av datorlagrad, jämförelsevis statisk information som kunnat mätas upp eller samlas in på annat sätt och därefter lagrats i förväg i databaser. Sådana databaser kan beskriva terrängens utseende och beskaffenhet, men också annan relevant miljöinformation om vägar, nätverk, demografi, bebyggelse m m kan vara av stor betydelse. Vidare har ledningssystemen länge innehållit detaljerade databaser över tänkbara motståndares organisation, taktik och utrustning. Hittills har dessa miljö- och hotdatabaser främst utnyttjats av mänskliga operatörer, och på senare tid som stöd för tolkning av sensorinformation för målidentifiering (bl a databaser över målsignaturer, *signaturbibliotek*, som beskriver hur olika måltyper i detalj ter sig eller “ser ut” för ett visst slags sensor). Vidareutvecklade sådana databaser har stor potential som informationskälla vid utveckling av metoder för informationsfusion.

2.3 Evolutionär systemutveckling och system av system
Den tekniska utvecklingen på IT-området kommer att göra många av dagens och gårdagens arkitekturlösningar föråldrade. Men den tycks alltså också erbjuda nya möjligheter till gradvis anpassning av "systemarvet", varför visioner av *evolutionär systemutveckling* ter sig mer realistiska idag än tidigare. Med denna term menar vi en utvecklingsprocess för lednings- och informationssystem som gör det möjligt och effektivt att utveckla komplexa system successivt och i takt med förändringar av verksamhetens behov.

Det finns ofta anledning att se alla system inom en verksamhet som ett supersystem (*system of systems* [2][12]), och med den synen blir behovet av att kunna hantera och förstå en dynamiskt föränderlig, icke-homogen arkitektur tydlig. Det vore visserligen mycket riskabelt och ofta omöjligt att försöka bygga detta supersystem som ett enda

projekt, men när det småningom visar sig att de i en verksamhet ingående olika systemen allt mer måste integreras och därvid behöver kunna utbyta information med varandra, måste först systemens begreppsvärldar ensas eller automatisk översättning mellan dem ske.

Detta förutsätter att teknik som bygger upp något av de många olika skikt som systemet består av, kan bytas ut helt eller delvis utan att systemet upphör att fungera väl. Inte ens väl genomtänkt och samordnad systemarkitektur av konventionellt slag har som regel kunnat förhindra att viss teknik, viss organisation etc byggts in i systemet på ett sätt som kraftigt begränsar möjligheterna till vidareutveckling och underhåll. Det är alltså inte tillräckligt att skapa ett system som är en perfekt maskin när det sätts i drift (i och för sig en som regel ouppnåelig dröm), utan det krävs framför allt att de många delar systemet består av är sammankopplade på ett så flexibelt sätt att de kan bytas ut utan att andra delsystem eller helheten påverkas negativt.

Ett intressant och aktuellt forskningsfält inom informationssystemområdet, som vi tror kan spela en viktig roll för att underlätta evolutionär systemutveckling, är framväxten av s k *medlarsystem* (avsnitt 3.2.6), som är avsedda att möjliggöra automatisk översättning mellan olika databasers begreppsvärldar eller *scheman*, dvs namn och andra karakteristika hos deras datatermer och de semantiska sambanden datatermerna emellan. Då och då ställer förändringar i verksamheten krav på att nya, genomgripande begreppsvärldar skapas. Ett argument för evolutionär systemutveckling är att man kan och bör pröva dessa nya begrepp i praktiken vid sidan av de traditionella, och därvid utnyttja de möjligheter som finns att genom medlarsystem låta olika systemgenerationer samexistera och samverka. Detta är dock i allmänhet inte möjligt om systemen från början konstruerats helt utan gemensam begreppsmässig grund: medlaren kan inte skapa en dataterm om den inte redan finns i någon form eller kan beräknas ur annan tillgänglig information. Detta är ett exempel på en allmän begränsning av vad som kan åstadkommas med en rent evolutionär attityd till systemutveckling.

2.4 Simulering

Modellering och simulering (MoS) är begrepp som har en utomordentligt vidsträckt innebörd. I en grundläggande mening är allt mänskligt tänkande baserat på modeller, tillfälligt skapade eller mer permanenta. Att simulera är att aktivera en modell samt att iaktta och registrera det förlopp som utspelar sig när modellens tillstånd utvecklas i enlighet med de regler eller lagar som vi tilldelat den. Modellering och simulering har fått en utomordentligt stor praktisk betydelse. Några viktiga orsaker till detta är [4]:

- verktyg och metodik i växelverkan möjliggör i datorns tidsålder modellering och simulering av processer som kan fånga det förutsägbara inslaget, trenden, i naturliga fenomen så komplexa att människan tidigare stått helt utan effektiva prognosinstrument
- det har i många fall visat sig möjligt att särskilja metodiken från det enskilda problemet och därvid utveckla generiska verktyg som med en begränsad och relativt rutinbetonad arbetsinsats kan appliceras på problem inom en vid klass; i dessa fall kan det alltså vara både vetenskapligt och ekonomiskt lönsamt att satsa mycket stora och kvalificerade resurser på att analysera och finslipa metodiken och utgående från detta skapa de generiska verktygen. Det är uppenbart att denna väg

endast kan vara framgångsrik vid modellering av system som är vetenskapligt mycket väl förstådda och beskrivna, och där precisa och allmängiltiga delmodeller kunnat utvecklas för varje aspekt som har betydelse för den valda klassen av tillämpningar

- i takt med att allt fler sådana generiska modellerings- och simuleringsverktyg tagits i bruk inom olika tillämpningsområden, har dessas verklighetsuppfattning påverkats starkt, tidigare alltför komplexa eller till och med okända samband har kunnat beskrivas i detalj och utnyttjas praktiskt, forskare och analytiker har fått helt nya språk och resultat av gigantiskt ekonomiskt värde har gradvis kommit fram
- processer som tidigare varit omöjliga att studera experimentellt, i försvars-tillämpningar exempelvis destruktionsförlopp och effekt av vapen mot komplexa och oersättliga mål eller utfallet av en strid mellan stora förband med olika utrustning och ledningsförmåga, blir med MoS möjliga att beskriva och i viss mening förstå och analysera; det allestädes närvarande inslaget av osäkerhet och slump kan åtminstone i princip behärskas genom statistiska modeller samt registrering och analys av resultat från upprepade försök, även om svåra och fundamentala problem som *kaos* och *kombinatorisk explosion* inte sällan reser oöverstigliga hinder mot långtgående förutsägelser och exakt lösning av komplexa problem.

I militärtekniska sammanhang har modellering och simulering ofta kommit att användas för att beteckna stora spel- eller utbildningsmodeller, som simulerat ett krig eller någon delprocess som ansetts viktig för utvecklingen av en situation som har betydelse i krig. Modellerna är då avbildande och beskrivande, och deras syfte är att en analytiker ska få insikter som hjälper honom att bättre förstå en process, eller att en "soldat" tränas för att bättre kunna hantera en svår situation.

Väl utformade simuleringsmetoder erbjuder möjligheter att förbättra förmågan att förstå och hantera dynamiska förlopp hos beslutsfattare och deras medarbetare. Genom simulering av stridssituationer kan personalen ges ökad stridsträning redan under fredsförhållanden.

All simulering kräver att en adekvat modellering av verksamheten har genomförts, vilket kan medföra höga kostnader för systemutveckling. När ett adekvat simuleringssystem väl finns att tillgå är det dock regel att det medför både förhöjd kompetens och minskade utbildningskostnader jämfört med en organisation som saknar simuleringsmöjligheter.

Distribuerad interaktiv simulering (DIS) är en teknologi som utvecklats långt inom USAs krigsmakt. Med dess hjälp kan stora, geografiskt distribuerade grupper eller förband träna tillsammans under tillräckligt realistiska former för att deras strids-duglighet skall kunna höjas väsentligt. Upprinnelsen till DIS var *SIMulator NETworking* (SIMNET)-projektet (1983-89), som initierades av DARPA. I SIMNET utvecklades en distribuerad realtidssimulator bestående av hundratals ihopkopplade simulatorer för pansarvagnar, helikoptrar etc. En av visionerna bakom DIS som nämns i US DoDs *Modeling and Simulation Master Plan* [13] är att åstadkomma en infrastruktur som möjliggör att simuleringsmodeller av olika typer, som kan vara belägna på olika platser och vara framtagna vid olika tillfällen av olika leverantörer, ska kunna länkas samman. Denna teknik innebär även att *skarpa* simuleringar (dvs

verkliga objekt), *virtuella* simuleringar (dvs bemannade simulatorer), *konstruerade* simule-ringar (t ex förbandsmodeller) och mänskliga beslutsfattare alla kan växelverka i en realistisk och komplex virtuell värld.

För att realisera ovanstående vision driver DoD utvecklingen av ett gemensamt tekniskt ramverk för MoS (*M&S Common Technical Framework*). Detta består *av High Level Architecture* (HLA), *Conceptual Models of Mission Space* (CMMS), och *Data Standards* (DS). HLA är ett standardiserat kommunikationsgränssnitt för informationsutbyte mellan simuleringsmodeller (se avsnitt 4.1.1). 1996 beslöts att alla simuleringsprogramvaror finansierade av US DoD ska följa HLA-standarden. CMMS består bl a av bibliotek innehållande simuleringsoberoende beskrivningar av militärt relevanta processer, objekt och omgivningar. DS syftar till att ta fram auktoriserade data om militära förband, vapensystem etc.

Inom ledningssystemområdet kan vi urskilja åtminstone tre användningsområden för simulering: *träning av handhavande* av ledningsstödssystem, *utvärdering* av ledningssystem, och *simulering som beslutsstöd* i skarpa ledningsstödssystem. Det sistnämnda området behandlas i avsnitt 2.5.2.

I synnerhet vad beträffar handhavandeträning är det fördelaktigt om man enkelt kan länka samman ledningsstödssystemet med simulatorer och konstruerade simuleringar. Idag krävs dock ofta utveckling av komplicerade program för att översätta mellan olika systems datarepresentationer. I avsnitt 3.2.3 redogörs för några internationella ansatser för att hantera detta problem. Sammanfattningsvis kan sägas, att även om det är önskvärt att de informationsmodeller som används inom ledningsstödssystem och simuleringar blir alltmer likartade, så kommer detta troligen att ta tid. Ett mer realistiskt utvecklingsscenario på medellång sikt är att man enas om en gemensam modell för den information som behöver utbytas mellan system, och att översättningar görs mellan systemspecifika representationer och den gemensamma modellen.

2.5 Beslutsstöd

Inom ramen för ett ledningsstödsystem vill vi skilja mellan delsystem för kommunikation, informationsinhämtning och (direkt) stöd i beslutsfattandet, och då avser vi med beslutsstöd sådan programvara som kan hjälpa beslutsfattaren att analysera beslutsalternativ baserat på ett underlag som levereras av andra delsystem, t ex system för informationsfusion eller en databas med terränginformation. Men man får komma ihåg att sådana system också själva kan innehålla beslutsstödsfunktioner, t ex stöd för optimalt sensorutnyttjande i ett fusionssystem och stöd för gruppering av luftvärn eller fältartilleri i en terrängdatabas, varför det vore missvisande att se beslutsstödet som ett sammanhängande och avgränsat delsystem i ledningsstödsystemet. Här kommer vi främst att diskutera programvara som kan stödja militära beslutsfattare vid planering eller genomförande av militära operationer.

Låt oss först peka på vissa allmänna förutsättningar för att bra beslut ska kunna fattas. En förutsättning som kan tyckas självklar, men inte alltid är uppfylld, är att beslutsfattaren måste ha en klar uppfattning om målet för det beslut som ska fattas. En annan uppenbar förutsättning är tillgång till aktuella och relevanta fakta. Dessutom måste beslutsfattaren ha en omvärldsmodell (naturvetenskaplig, teknisk, doktrinär, psykologisk, sociologisk etc) som gör det möjligt att tolka osäkra eller ofullständiga data,

bedöma sannolikheten av att en viss händelse kan inträffa eller att en viss handling kan genomföras, förutsäga konsekvenserna av en händelse eller handling m m.

Den tekniska utvecklingen har gjort det lättare att uppfylla flera av de ovanstående förutsättningarna. Utvecklingen av satelliter, sensorer, datanät m m ger nya möjligheter till kommunikation och övervakning. Ökningen av datorernas lagrings- och beräkningskapacitet har gjort det möjligt att lagra och utföra beräkningar på stora datamängder. I den mån våra omvärldsmodeller kan utryckas i en logisk-matematisk modell, så kan datorerna också hjälpa oss med tolkning av data, att förutsäga händelseutvecklingen m m.

I återstoden av detta avsnitt ger vi först några exempel på beslutsstöd, sedan beskriver vi några nyckelteknologier som dessa bygger på. Avslutningsvis diskuterar vi utvecklingsmetoder för beslutsstöd.

2.5.1 Exempel på militärt beslutsstöd

Beslutsstöd för militära beslutsfattare behövs inom flera områden, t ex prognosticering av förluster och grupperingstider, analys av egna och angripares handlingsalternativ m m.

Nedan ger vi några mer detaljerade exempel på militära beslutsstöd.

För marina tillämpningsområden har exempelvis följande beslutsstöd utvecklats [5]:

- Taktisk hotvärdering: Taktisk hotvärdering söker känna igen vilken plan fienden följer. Modeller av möjliga fientliga planer kan konstrueras. De levererar uppgifter om vilken information som är att förvänta medan planen fullföljs. Rapporter om fienden används för att uppskatta sannolikheten för att han följer en viss plan samt i vilket stadium av planen han befinner sig. Planigenkänning har demonstrerats t ex i fallet att en landstigningsstyrkas chef observerar en försvarande styrka. Nästa steg är att använda denna information för att förutsäga fiendens nästa observerbara åtgärd.
- Tidskritisk planering mot ytattack: För att stödja planering av åtgärder mot ytattack har metoder utvecklats för att planera optimala fartygsmanövrer. Tidskritiska metoder har utvecklats för att beräkna siktområden, för att finna kortaste användbara vägen mellan två punkter och för att beräkna vart ett fientligt fartyg kan ha nått vid en given tidpunkt i framtiden. Nyckelidén här har varit att utveckla tekniker som kan ge mer noggranna resultat, ju längre de tillåts exekvera, för att möjliggöra optimalt utnyttjande av tillgängliga beräkningsresurser i varje situation.
- Planering av koordinerat anfall: Planeringen av koordinerat anfall kan t ex ske utifrån perspektivet hos chefen för en försvarande fartygsstyrka. Stöd ges för att generera en koordinerad anfallsplan som tilldelar egna fartyg och robotar till de olika målen i en fientlig angreppsstyrka.

För armétillämpningar har följande beslutsstöd demonstrerats i ett utländskt forskningsprojekt [5]:

- Beräkning av framkomlighetskorridorer: Digitala terrängdata lagras i en databas och kan presenteras med utbytbara överläggsbilder. Överläggsbilderna används för att visa områden med olika framkomlighet. Ur denna information beräknas korridorer av bestämd bredd, tillräcklig för passage med fordon av en viss typ.

Korridorerna ger sedan möjliga framryckningsvägar och noder mellan sådana, vilket ger en utgångspunkt för konstruktion av Course of Action (CoA). Det sistnämnda betraktas som en kooperativ process, och stöd för sådant grupparbete kan ingå i systemet.

- CoA-jämförelser: Stöd för att jämföra egna och fientliga CoAs, t ex med hjälp av vapeneffektindex och vägda förbandsvärden, kan också byggas in. Dessa metoder kombinerar subjektiva värderingar av relativ vapeneffekt med sannolikhetsanalys av utfallen.
- Planering av manövrering och eldunderstöd: Delvis automatiserad konstruktion av en detaljerad manöverplan kan ske i realtid. Komplexa temporala, spatiella och taktiska villkor och kriterier kan beaktas vid planeringen.

2.5.2 Exempel på teknologier som används för beslutsstöd

Nedan ger vi korta beskrivningar av några teknikområden som används vid utveckling av militära beslutsstöd.

Geografiska informationssystem (GIS, se avsnitt 4.3.4): För bl a logistikprogram är geografiska informationssystem ett viktigt grundläggande hjälpmedel, men avancerad transportplanering kräver metodik som går utöver de enkla nätverks- och framkomlighetsberäkningar som avancerade GIS idag tillhandahåller.

Simulering: Simuleringar av stridsförlopp har hittills huvudsakligen använts i analys, studier och utbildning. Det finns idag ett ökande intresse av att använda simulering som beslutsstöd i ledningsstödssystem, men för detta krävs simuleringsprogram som uppfyller följande villkor:

- Alla relevanta processer (ledning, spaning m fl) måste modelleras på ett adekvat sätt.
- *Computer generated forces* (CGF) behövs för att underlätta hanterandet av simuleringsprogrammet; naturligtvis måste CGF också uppträda på ett intelligent sätt.
- Indata till simuleringsprogrammet måste kunna genereras automatiskt från information i ledningsstödsystemet, som tyvärr ofta är vag, osäker, ofullständig och motsägelsefull.
- Simuleringsprogrammet behöver validerade data beträffande terräng, vapensystem, doktriner mm.

Det är dock tveksamt om existerande simuleringsprogram uppfyller dessa villkor; ofta är t ex tidskrävande förberedelser och specialistkompetens nödvändiga för att de ska kunna användas.

Ett naturligt användningsområde för simulering är utveckling och analys av planer (CoA). I ett nyligen avslutat projekt finansierat av *Defense Advanced Research Projects Agency* (DARPA) och genomfört av bl a *Science Applications International Corporation* (SAIC), utvärderades metoder för analys, utveckling och jämförelse av olika handlingsalternativ. Målet var att mer robusta handlingsalternativ ska kunna utvecklas snabbare och att militära planerare ska få djupare förståelse för olika handlingsalternativ. För att uppnå detta, såg man ett behov av följande typer av stödfunktioner:

- *Initial Feasibility Filter*: ett interaktivt verktyg som kan användas för att testa genomförbarheten hos en föreslagen CoA. Man kom fram till att ett sådant verktyg lämpligen använder s k constraint programming, som beskrivs närmare nedan.

- *Range-of-Outcomes Generator*: ett program som generar ett stort antal av de möjliga konsekvenserna av en CoA.
- *Representative Outcome Generator*: ett simuleringsprogram som gör det möjligt att mer i detalj studera effekterna av en viss CoA.
- *Animering*: Verktyg som hjälper planläggaren att visualisera en CoA.

Prognosmodeller är nödvändiga om man ska kunna genomföra situations- och hotanalyser. De är också värdefulla i situationer då sambandet inte fungerar. Då kan en "avskuren" stab prediktera händelseutvecklingen i ett stridsförlopp, följa upp egna resurser etc, och trots det avskurna sambandet under en tid bibehålla en acceptabel uppfattning om situationen. När sambandet återkommer uppdateras modellerna med verkliga data. En alternativ stab skall snabbt kunna överta ledningsansvaret vid bortfall av ordinarie stab. Detta innebär att alternativstaber skall vara uppdaterade även med utgångsvärden för och resultat av genomförda simuleringar.

Constraint programming (CP): CP behandlar tekniker för formulera och lösa s k *constraint satisfaction* (CS) problem, som kan formuleras enligt följande: givet ett antal logiska villkor som involverar ett antal variabler, bestäm värden till variablerna så att samtliga villkor är samtidigt satisfierade (eller avgör att detta är omöjligt). En mängd olika problem kan formuleras som CS problem. Emellertid är CS-problem nära besläktade med det s k satisfierbarhetsproblemet, vilket var det första problem som bevisades vara *NP-fullständigt*, en stark indikation på att generellt effektiva lösningsmetoder saknas. I praktiken kan dock många CS problem ändå lösas effektivt.

Ovan nämndes att CP kan användas för analys av CoA. I korthet fungerar det som följer. Varje militärt uppdrag beskrivs med en uppsättning villkor och variabler. T ex kan den doktrin en trupp följer vid attack beskrivas på detta sätt. När planeraren utvecklar en specifik handlingsplan väljer han en eller flera uppsättningar av villkor samt fixerar värdet på vissa variabler. Systemet avgör sedan om villkoren kan satisfieras, dvs om det är möjligt att hitta värden för de fria variablerna så att samtliga villkor är uppfyllda. Om villkoren inte kan uppfyllas måste planeraren modifiera handlingsplanen.

2.5.3 Utveckling av beslutsstöd

För utveckling av beslutsstöd finns en mängd mer eller mindre generella hjälpmedel. Dessa kan vara "icke domänspecifika", dvs avser antingen generella system-utvecklingshjälpmedel eller hjälpmedel för att hantera en viss, noga avgränsad metodik, t ex hantering av s k Bayesianska nätverk för resonemang under osäkerhet eller programsystem för tids- och resursplanering av projekt. De kan också vara tillämpningsorienterade, och på senare tid har det kommit fram flera miljöer för specifikt militära analys- och planeringsuppgifter, i regel baserade på en plattform av generella GIS- och kontorsstödprogram.

För att beslutsstödsmetoder skall kunna tillämpas i den utsträckning som är möjligt och önskvärt i en framtida försvarsmakt, krävs att betydande forsknings- och utvecklingsinsatser görs. Centralt är att välja, utveckla och anpassa metoder och algoritmer för hantering av osäker information, för situations- och plananalys, beräkning av styrkeförhållanden, simulering av alternativa handlingsalternativ, prediktion och prognosticering, modellering och kunskapsrepresentation. Som grund för alla dessa metoder krävs väsentligt förbättrad metodik för kommunikationsnätverk,

databaser, datafusion, simulering, användargränssnitt m m. Men viktigast är att skapa och utnyttja förmåga att identifiera de situationer där beslutsstöd är mest kostnads-effektiva. Sådan förmåga tror vi enbart kan skapas genom kombination av relevant taktisk och teknisk-vetenskaplig kompetens.

2.5.4 Exempel: logistiksystem för övervakning och styrning av fordonsflottor

Ett område där datoriserat beslutsstöd visat sig kunna göra stor nytta är logistik, dvs planering av resursförsörjning och annat underhåll, med hänsyn till begränsningar i exempelvis transportkapacitet, omhändertaganderesurser och lagringsmöjligheter. *Fleet management systems* (FMS) är t ex en typ av system som är nära besläktade med logistiksystem och (militära eller civila) ledningsstödsystem.

FMS är kombinerade informations- och kommunikationssystem med vars hjälp en mängd rörliga enheter kan kontrolleras och övervakas. ”Enheterna” kan vara förarstyrda (t ex taxibilar, bussar, lastbilar), autonoma eller eventuellt fjärrstyrda. De kan också vara objekt som färdas ombord en följd av fordon, t ex mänskliga passagerare eller gods som fraktas styckevis eller i containers.
Integration av avancerade FMS och logistikplaneringssystem kan komma att leda till betydelsefulla innovationer. Exempelvis kommer det att bli möjligt, inte bara att dagligen planera rutten för varje fordon i flottan, utan också känna till varje kollis position i varje ögonblick, antingen det befinner sig ombord på ett fordon, väntar i en godsterminal, eller just har levererats till en kund. Det kommer också att bli möjligt att utföra omplanering av leveransschemat så snart tillräckligt stora förändringar i logistiksituationen motiverar det. Man kan då ta hänsyn till såväl trafikbestämmelser som gäller för varje typ fordon längs olika delar av vägsträckan, som till sammansättningen av den aktuella lasten och restriktioner som kan betingas av detta, såväl som till de rådande och förväntade trafikförhållandena. Sådana system kommer att behöva integreras med väginformationssystem, med vars hjälp en förare kan finna, visualisera och följa sin väg genom ett komplext, okänt vägnät.

2.6 Informationssäkerhetsproblemet

Grundproblemen inom datasäkerhet är sekretess, integritet och behörighetskontroll. Till dessa tillkommer de problem som finns inom informationssäkerhetsområdet, främst tillförlitlighet och nyckelhantering. Det finns dessutom några problem som är av intresse inom försvaret som ökad mobilitet, heterogena system (se avsnitt 3) och fysiska skydd.

2.6.1 Datasäkerhet

Datasäkerhetsområdet rör problem med kontroll över data, vid lagring och överföring. Problemen gäller *sekretess* och *integritet*. Sekretess handlar om att enbart behörig aktör ska kunna läsa data. Aktör kan vara antingen en person eller en datorresurs. Integritet handlar om att förhindra att data förvanskas. Ett flertal metoder för att lösa problemen finns. En vanlig princip är att fysiskt skydda data och överföringskanaler. Detta är ofta inte möjligt utan skyddet åstadkoms på systemmässig väg genom en systemutformning där ett behörighetssystem byggts in samt genom kryptering av data.

När system knyts samman över större datornätverk ökar säkerhetsproblemen ytterligare. Nätverkstekniken med ett stort antal aktiva datorer som sköter transport av

data kräver även att datorresurser kan identifiera sig och nödvändiggör ett behörighetssystem som sträcker sig globalt över nätet. Nätet förenar ofta system uppbyggda med olika säkerhetsfilosofier. Detta kan gälla skilda metoder för kryptering eller autentisering vilket ställer krav på metoder för att hantera dessa olikheter.

2.6.2 Trust - Förtroende - Tilltro - Tillförlitlighet

Det engelska begreppet trust är ett vitt begrepp med tillämplighet i många sammanhang av mänskliga aktiviteter. Det handlar om tilltro, tillit, förtroende mellan människor men också mellan människor och maskiner och mellan människor och system. Ursprungligen baserades "trust" på direkta personliga relationer mellan människor, men med den moderna tekniken hjälp förmedlas relationer ofta med hjälp av svåröverskådliga tekniska system.

Den allmänna betydelsen av ordet är viktig att hålla i minnet. Man kan aldrig uppnå en absolut tilltro till ett system eller någon verksamhet över huvudtaget. Tilltron beror på ett stort antal faktorer varav vissa kan kontrolleras relativt väl med tekniska lösningar medan andra är svåra att både beskriva och hantera. Metoder att lösa problemen kan falla under rubrikerna datasäkerhet, systemsäkerhet och användningsrelaterade problem.

2.6.3 Nyckelhantering

För identifiering av systemresurser måste digital nycklar lagras i systemet. Distribution och generering av nycklar kräver speciella metoder. Existensen av ett flertal system med olika egenskaper både beträffande skyddsnivåer och hanterbarhet kan göra metodvalet komplicerat. Olika lösningar i olika sammanhang ställer dessutom krav på kompatibilitet. Det måste betonas att själva krypteringens styrka inte är den enda faktorn som avgör datasäkerheten. Möjligheten att hemlighålla nycklar, att säkert distribuera nycklar och att certifiera på ett säkert sätt är viktiga faktorer.

2.6.4 Distribuerade system och mobila noder

Teknikutvecklingen har lett till att alltmer av den information som transporteras i näten är programkod. Den kommande tekniken med s k agenter, dvs autonoma program som flyttar sig mellan noder i näten, ökar ytterligare kraven på effektiva behörighetssystem och autentiseringsmekanismer. En lösning av säkerhetsproblemen är i själva verket avgörande för möjligheterna att använda dessa tekniker. Den civila utvecklingen mot användning av Internet för kommersiella syften har ökat tempot i utvecklingen av säkerhetsstandarder för olika typer av transaktioner i öppna heterogena nät.

Behovet för en användare att kunna koppla in sig via en godtycklig nod i nätet och ändå direkt kunna arbeta i samma miljö som på sin hemnod har lett till begreppet *nomadicity* och forskning som stöder detta. Mobila nät, dvs i första hand nät med trådlös kommunikation, har extra svåra säkerhetsproblem. I dessa nät förändras topologin ständigt vilket bl a kräver att noderna i nätet identifierar sig för varandra kontinuerligt.

2.6.5 Fysiska skydd

Samtidigt som små och bärbara datorutrustningar gör det möjligt för användarna att bli mer rörliga innebär det också att det ställs nya säkerhetskrav på datorutrustningen och den information som därigenom kan göras åtkomlig.

Förutom att fysiskt skydda informationen i den personliga utrustningen kan man även ställa följande krav:

- att utrustningen utför exakt vad som förväntas och inget annat. Exempelvis får inte information förvanskas eller styras mot felaktiga destinationer vid korrekt användning. Detta kräver att alla steg från tillverkare till användare (utveckling, produktion, montering, leverans) inte har tillfört något medvetet eller omedvetet fel eller någon s k "bakdörr". Dessutom måste användargränssnittet vara utformat korrekt och entydigt för att garantera att användaren interagerar med systemet på ett förväntat och önskat sätt.
- att förlorad utrustning med öppna säkerhetsspärrar inte kan utnyttjas för falsk signering av data och information. Detta kan åstadkommas genom automatisk avstängning av utrustning som inte använts under en viss tid kombinerat med krav på kontinuerligt återkommande verifiering av användarens identitet.
- att lagring och verifiering av PIN-koder sker distribuerat så att dessa koder inte kan återskapas via en enskild datorenhet.
- att utrustningen består av skyddade och säkra gränssnitt, t ex tangentbord och bildskärm, som inte går att manipulera eller avlyssna.
- att utrustningen innehåller en tillförlitlig klocka som mäter tidsintervall för användaridentifiering och kontroll av tidsstämplade meddelanden samt en fysikalisk slumptalsgenerator istället för en slumptalsgenerator baserad på programvara. En programvarubaserad slumptalsgenerator ger nämligen inte en verkligt slumpmässig talföljd, utan en s k pseudoslumptalssekvens, som eventuellt skulle kunna förutses.

3. Evolutionär utveckling av ledningsstödsystem

Den tekniska utvecklingen gör det möjligt att integrera allt fler olika slags informationssystem i ledningsstödsystemen. Man kan se en trend bort från centrala informationssystem mot *distribuerade* system som har en allt större *heterogenitet* där många olika typer av informationssystem ingår, och där en allt större mängd mer eller mindre välstrukturerad information görs tillgänglig. Inte minst utvecklingen av *inter-* och *intranät* ökar radikalt tillgängligheten till information, men därmed ökar också heterogeniteten. Informationen blir samtidigt alltmer *dynamisk* och förändras kontinuerligt.

Framtida militära ledningsstödsystem kommer således att arbeta i en heterogen datormiljö där datorerna är sammankopplade genom höghastighetsnät och där både data och programvara är *distribuerade, heterogena och dynamiska*. Infrastrukturer för framtida ledningssystem måste också stödja såväl informationshantering för många olika typer av aktiviteter, som samordning av olika, mer eller mindre beroende, aktiviteter.

En för den hittillsvarande ledningssystemutvecklingen viktig distinktion är den som gällt mellan de egentliga ledningssystemen, avsedda som stöd främst till de operativa och taktiska ledningsnivåerna, dvs chefer och staber på divisions- eller åtminstone brigadnivå och däröver, och s k *stridsledningssystem* och andra datorbaserade stödsystem på lägre nivå i organisationen. Medan stridsledningssystemen från början betraktats som ”tekniska” system, avsedda att stödja förhållandevis rutinmässiga och i regel starkt tidspressade uppgifter, har de högre ledningsstödsystemen uppfattats som en samling hjälpfunktioner som dels underlättar framtagande av ett relevant beslutsunderlag för högre chefer, dels stöder uppgiften att korrekt och säkert sköta utsändning och registrering av skrivelser, order och meddelanden till och från de högre staberna.

Ytterligare en betydelsefull faktor som påverkat utvecklingen av ledningssystem i Sverige har varit vapenslagens olika behov, och kanske också deras traditionella attityder till behovet, av avancerad teknisk utrustning. Medan man i marinen och särskilt flygvapnet tidigt upplevde ett allt starkare behov av olika slags datorsystem för att kunna övervaka, styra och hantera dessa vapenslags komplicerade och snabbrörliga tekniska materiel, igångsatte man inom armén visserligen vid mitten av 80-talet ett ganska omfattande arbete för att utveckla ett taktiskt ledningssystem (*Datal*), men detta projekt lades ner efter några års specificerings- och provnings-arbete. Flygvapnet och senare även marinen har alltsedan Stril 60-systemets införande däremot använt stridsledningssystem driftsmässigt och även upphandlat nya sådana (framför allt flygvapnets *STRIC* och marinens *STRIMA*). Flera av dessa stridsledningssystem har utvecklats av *CelsiusTech Systems* (CTS), som levererat sådana system såväl till det svenska försvaret som till utländska försvarsmakter.

3.1 Öppna arkitekturer för evolutionär systemutveckling

Vi menar att utveckling av unika informationssystem, avsiktligt och systematiskt, bör läggas upp som en serie aktiviteter av begränsad omfattning och syftning. Varje aktivitet skall resultera i ökad konkret kunskap om hur det avsedda verksamhetsstödet

ska se ut och fungera, dvs ska leda till en fungerande och väl beskriven modell eller prototyp av någon aspekt av det slutliga systemet. Modelleringens nytta för verksamheten ska, där så är möjligt, prövas och värderas med systematiska metoder. Nya prototyper med stegvis ökad eller breddad förmåga utvecklas sedan, varvid man utnyttjar de resultat som tidigare nåtts. När man kommit så långt att de tvingande systemkraven ("skallkraven") uppnåtts, kan prototypen tas i bruk som ett driftssystem. Utvecklingsprocessen bromsas när man efter noggrann utvärdering bedömer att det inte längre är kostnadseffektivt att sträva efter ytterligare förbättring. Man bör dock räkna med att varje ledningsstödsystem kommer att behöva förändras och vidareutvecklas mer eller mindre fortlöpande.

Förenklat kan man beskriva *evolutionär systemutveckling* som en metodik där stora utvecklingsprojekt bryts ner till väldefinierade etapper med mål som beställaren själv kan verifiera. Om ett etappmål inte uppnåtts så får man välja mellan att satsa mer på etappen, minska ambitionen för etappen, eller lägga ner hela projektet. Det måste finnas en plan för hur etapperna leder till slutmålet och den ska kunna ändras i samband med utvärderingen av varje etapp. Uppenbarligen måste man räkna med att då och då få göra om en etapp, särskilt sådana i början av processen. Ju längre processen framskrider, desto större blir kostnaden för stora omplaneringar, medan man genom att tidigt upptäcka problem och ändra kurs för att lösa dem, inte sällan kan göra stora vinster på längre sikt.

En av grundprinciperna bakom evolutionär systemutveckling är att utvecklingsarbetet börjar med att skapa *yttre funktionalitet* på bekostnad av kvalitet och “skalbarhet”, dvs förmågan hos den inre strukturen att klara av kraftigt ökad systembelastning. Skalbarhet kan vara mycket svår att åstadkomma men visar sig i regel inte vara nödvändig förrän systemet sätts i en verklig, eller realistiskt simulerad, fullskalig driftsmiljö. Detta faktum är samtidigt en av den evolutionära utvecklingsprocessens svagheter, eftersom det innebär att man förr eller senare (och man får inte vänta för länge, för då kan kostnaden bli oöverkomlig) måste hämta in denna kvalitetsmässiga eftersläpning, ofta till stora kostnader men utan motsvarande höjning av systemets funktionella förmåga. Detta kan av beställare och användare upplevas som ett misslyckande, och fenomenet måste därför tydligt anges i utvecklingsplanen som en förutsättning för tillämpning av denna metodik.

Medan erfarna användare antagligen bäst bedömer kvalitet och användbarhet hos systemets yttre funktioner, krävs det systemteknisk kompetens för att bedöma systemets inre struktur och kvalitet. Detta kommer i senare steg av en evolutionär utvecklingsmodell inte sällan att vara viktigare än att bedöma dess yttre förmåga, eftersom kvaliteten hos den inre strukturen i längden är avgörande för möjligheterna att vidareutveckla ett programsystem.

Med en rätt genomförd sådan *evolutionär utvecklingsmetodik* skulle man inte behöva riskera att drabbas av stora, till synes plötsliga, misslyckanden, utan man skulle få bättre förvarning om de delar av ett stort projekt som riskerar att ge problem. Med andra ord vore det möjligt att väsentligt reducera risken för ett totalt misslyckande.

Ett arbetssätt för evolutionär utveckling och vidmakthållande av lednings- och informationssystem ställer bl.a. följande krav:

- *modularisering*: delsystem definieras med avseende på funktion, prestanda, funktionssäkerhet, livslängd och gränssnitt så att de kan modifieras eller bytas ut, utan att systemets egenskaper förändras på ett okontrollerat sätt. Leverantörerna kan därmed ges maximal frihet att på billigaste sätt tekniskt realisera den önskade funktionen
- *flexibel konstruktion*: relationen mellan delsystemens och det totala systemets funktion är så väl känd och beskriven, att systemet kan omstruktureras om den tekniska utvecklingen möjliggör radikal förbättring av vissa delsystem, eller för att systemet skall inriktas mot ändrade uppgifter och hot. Detta kan vara aktuellt såväl under definition och konstruktion, som under vidmakthållande.
- *funktion och beteende*: modulernas funktion under konstruktion, efter reparation eller modifiering, kan verifieras av tillverkaren i tidiga skeden (under konstruktionsfasen helst innan konstruktionen är utförd i detalj) med hjälp av känt beteende hos ingående byggblock samt simuleringsmodeller.
- *verifiering av tillförlitlighet/livslängd*: modulernas tillförlitlighetsegenskaper måste under konstruktion, efter reparation eller efter modifiering kunna bedömas så väl att de risktagningar som görs är väl avvägda.

Genom utvecklingen mot funktions- och objektorientering har programvaruområdet redan kommit långt i utvecklingen mot sådana öppna arkitekturer. Inom mikroelektronikbranschen spelar utvecklingen av olika teknologier för kundanpassade kretsar en liknande roll. I bägge fallen utgör dock verifiering av tillförlitlighet och livslängd ett problem i relation till de krav försvaret, och andra högteknologiområden, måste ställa.

Standardiseringsfrågor är en viktig del av en strategi för evolutionär systemutveckling. Om möjligt bör leverantörsoberoende öppna standarder väljas. En omdömesgill användning av väletablerade kommersiella de facto-standarder kan dock många gånger innebära låga kostnader och obetydligt risktagande, därför att kommersiella krafter kommer att verka för att fungerande gränssnitt finns också till framtida produktgenerationer, liksom till de nya produkter som så småningom övertar marknaden.

Ett exempel på framväxten av industrikonsortier för etablering av industristandarder är *Object Management Group*, OMG, som utvecklat standarderna *CORBA* (se avsnitt 4.5.1) och *UML* (se avsnitt 3.2.1). Ett annat är *Open Software Foundation*, OSF, som i början av 90-talet utan större framgång föresatte sig att skapa ett industrigemensam standard för Unix-baserade operativsystem men som numera mest är känt som organisationen bakom DCE, ett system för interoperabilitet mellan program i nätverk (se avsnitt 4.4.2).

3.1.1 Det organisatoriska perspektivet

Tyvärr är det inte alltid som forskningsrön och andras praktiska erfarenheter tillämpas i den egna praktiken. Inte heller används alltid bästa tillgängliga teknik och organisation i systemutvecklingsprojekten.

Beställaren Försvarsmakten behöver, för att kunna leda en ständigt pågående, distribuerad utvecklingsprocess, själv ha tillgång till kompetenta och stabila beställarfunktioner med kvalificerade kunskaper inom relevanta områden, inklusive ledning, militär taktik och informationsteknologi. För att upprätthålla sin anknytning

till aktuell metodik- och teknikutveckling, behöver beställarfunktionerna nära kontakter och en levande dialog med aktiva och nyfikna forskare. Båda dessa specialistkadrer måste dessutom upprätthålla, och ges möjligheter att upprätthålla, goda samarbetskontakter med de användargrupper för vilka systemen utvecklas.

Till detta kommer krav på adekvat organisation och metodik för drift, vidmakthållande och vidareutveckling av systemen. Den snabba tekniska utvecklingen och de allt starkare kraven på evolutionär systemutveckling innebär att utvecklingsfasen för ett system inte längre kan vara tidsmässigt skild från driftsfasen. Utveckling innebär istället ständig modifiering och utbyggnad av systemets funktioner. För att detta ska kunna ske utan att drift och användning av systemet påverkas negativt, krävs förutom kvalitet och disciplin i utvecklingsarbetet också en rullande och med programsystemutvecklingen koordinerad planering av processerna för dokumentationsutveckling, användarutbildning och teknikinförande. Detta kräver i sin tur framför allt uthållighet hos planering och organisation, ett krav som det visat sig allt svårare att uppfylla under inflytandet av den turbulenta omvärlds- och ekonomiutveckling som karakteriserat de senare åren. I en sådan värld kan det kanske vara frestande att ersätta systematiskt, vetenskapligt förankrat och uthålligt förbättringsarbete med "snabba klipp" på systemutvecklingsmarknaden.

Minst lika viktigt som att i komplexa systemutvecklingsprojekt arbeta evolutionärt, är att utnyttja bästa tillgängliga erfarenhet och kompetens och, inte minst, redan existerande kommersiell eller egenutvecklad metodik och tillhörande programvara. Där sådan metodik inte finns att tillgå, är successivt fördjupad problemanalys, prototyputveckling och demonstration enligt vår mening den bästa utvecklingsprincipen. Utveckling av unika informationssystem är ett slags tillämpad forskning, som kräver sakkunskap och kreativitet på flera plan och inom skilda kompetensområden. Det kan inte vara rimligt att Försvarsmakten bygger upp alla dessa resurser inom den egna organisationen, men däremot borde man, anser vi, skapa sådana nätverk att alla landets och naturligtvis också utländska relevanta kompetenser beaktas när nya system planeras. Detta kan inte ske i form av en enda regelrätt upphandlingsprocess, utan metoden förutsätter effektivt nyttiggörande av kompetens inom den egna organisationen, anlitande av kompetenta forskare från institut och universitet samt förmåga till kommersiell upphandling av kvalificerade konsult- och utvecklingstjänster, alltsammans styrt av över tiden skiftande behov.

Finansiering av sådan verksamhet bör snarare ske som avsättning av en viss andel av den totala verksamhetsbudgeten än som stora, objektbundna, unika projekt. För att garantera att verksamheten bedrivs med rimlig effektivitet trots att inga enkla effektivitetsmått finns, behövs en kvalificerad organisation för initiering och upphandling av utvecklingsarbete samt konkurrens mellan leverantörerna av de - relativt små - enskilda projekten.

En central fråga i samband med evolutionär systemutveckling är organisering och ansvarsförhållanden vid utveckling, tillverkning och vidmakthållande. I industrin finns en trend mot samverkan i nätverk. I vissa fall talar man om virtuella företag. I viss mening är detta en motsatt organisationsform gentemot den, åtminstone under senare decennier, traditionella modellen inom försvaret, nämligen att sköta huvuddelen av systemsammanhållningen genom en huvudleverantör.

Utvecklingen av arbetsorganisation och kommunikationsteknologi innebär ökande möjligheter att hålla samman stora utvecklings- och produktionsprojekt över organisatoriska och geografiska avstånd. I nätverksföretag bildar de olika kompetenser som fordras för att utveckla, producera och driva eller vidmakthålla ett komplext system, en dynamiskt föränderlig struktur som är utspridd geografiskt och organisatoriskt. Utvecklare och tillverkare av delsystem samarbetar med flera systemsammanhållare och dessa upprätthåller normalt också möjligheter att arbeta med alternativa delsystemleverantörer.

Nätverksorganisationernas förmåga att vidmakthålla komplexa system under flera decennier är av lätt insedda skäl ännu svår att bedöma. Mot denna osäkerhet bör ställas att även förutsättningarna för traditionella huvudleverantörer att tillgodose kontrollbehov under lång tid, kan förändras av den strukturomvandling som pågår i näringslivet. Exempelvis kan det bli svårt att rekrytera och behålla kvalificerad personal till en miljö som uppfattas som organisatoriskt och tekniskt föråldrad.

För styrning av evolutionära utvecklingsprocesser inom ramen för en nätverksorganisation krävs en uthållig, stark och sakkunnig ledning, insikt om systemutvecklingsprocessens omfattning och komplexitet, fokusering på att välja ut och behålla kompetens, samtidigt bejakande av konkurrens och öppenhet, en vetenskapligt organiserad forskning och prototyputveckling samt förmåga att leda grupper med stor kompetensvariation. En evolutionär utvecklingsorganisation ställer alltså höga krav på ledarskapsförmåga men ger också möjlighet att delegera svåra tekniska eller personella avgöranden till ”lägre” organisationsnivåer.

I utvecklingen bort från stora sammanhållna enheter med förmåga att inom sig svara för alla viktiga delar av utveckling, tillverkning och vidmakthållande av ett visst komplext system hänger de organisatoriska och affärsmässiga aspekterna samman med de tekniska. Mycket hög teknisk integrationsgrad kräver allmänt sett också en hög grad av organisatorisk integration. För att ett komplext system ska kunna utvecklas, tillverkas och vidmakthållas av ett nätverksföretag krävs en öppen systemarkitektur med väl definierade gränssnitt och kommunikationsstandarder. Öppnare systemarkitekturer och användning av allmänt accepterade standarder kan ge ett alternativt sätt att möjliggöra vidmakthållande över lång tid för komplexa system.

3.1.2 Det processtekniska perspektivet

Ovan har vi anlagt ett organisatoriskt perspektiv på evolutionär systemutveckling. Man kan också anlägga ett mer processtekniskt perspektiv. Detta innebär bl a att man accepterar att ett systemutvecklingsprojekt inte sker i ett vakuum utan i växelverkan, ofta i tävlan, med andra verksamheter i en mycket dynamisk omvärld. Lösningar som är ogenomförbara när projektet ursprungligen planerades kan efterhand te sig alltmer attraktiva[2].

[2] Moore´s lag, som säger att pris/prestandaförhållandet för datorer sjunker till hälften på mellan 18 och 24 månader har gällt sedan början på 70-talet [14][15]. Hur länge vi kan räkna med att behålla en sådan utvecklingstakt är omöjligt att säga, fast en del faktorer talar för att takten kommer att minska successivt under de närmaste femton åren. Ett besläktat exempel utgörs av kapacitetstillväxten för minneskretsar, som medfört att det idag är realistiskt att använda databaser som lagras i primärminnet för många kvalificerade ändamål, något som tidigare inte varit möjligt och som medför att många tidigare sanningar om konstruktion av informationssystem ställs på huvudet.

Sett i det processtekniska perspektivet blir det viktigt att under utvecklingens gång, och inte minst även sedan systemet tagits i drift, kunna ta tillvara nya system, produkter och "halvfabrikat" som kommer ut på marknaden medan den egna systemutvecklingen pågår. En grundläggande fråga blir om och i så fall hur man kan utforma sin utvecklingsprocess och sina etappvisa system så att integration av såväl redan existerande system, egna eller andras, som på marknaden nytillkommande produkter underlättas väsentligt. En utvecklingslinje som är avsedd att underlätta detta är forskningen om s k *medlarsystem* (se avsnitt 3.2.6), en annan framväxten av infrastruktur för dynamiska komponentbibliotek (*mäklarsystem*, se avsnitt 4.5) och en komponentmarknad för programvara (avsnitt 0).

I en artikel med titeln *Reevaluating the Architectural Metaphor: Toward Piecemeal Growth* skriver J O Coplien [16] provokativt att termen *Software Engineering* som myntades av Peter Naur 1968 olyckligtvis öppnade dörren för hårdvarutekniska begrepp i samband med programvaruutveckling. Därigenom ökade programvaru-arkitekternas intresse för formalism och matematisk stringens, vilket bl a ledde till utveckling av språk för arkitekturbeskrivning, formell konstruktionsnotation och formell metodik. I början av 80-talet började objektorienteringsmetodiken dra till sig intresse[3] och den erbjöd nya verktyg för att fånga abstraktioner som inte härrörde från matematik eller logik. Objektorienteringsparadigmens största värde är att den ställde människans vardagliga begreppsbildning i centrum för konstruktionsarbetet. Programmerare relaterar till de resurser och apparater som de arbetar med dagligen, och objektorientering blev en väg att beskriva och utveckla programvara som återspeglar denna förståelse. Coplien hävdar att arkitektur inte så mycket handlar om programvaran själv som om människorna som utvecklar denna programvara. Programvaruarkitekturens grundbegrepp, som *koppling* och *kohesion*, handlar inte om koden utan om dess organisation, en organisation som i mycket återspeglar utvecklingsteamets egen.

Nästa trend inom praktisk programvaruarkitektur är enligt Coplien *mönster* (*patterns*) [17], med vars hjälp den vanliga programmeraren återfår sin värdighet och erkänns som källa till värdefulla arkitektoniska bidrag. Ledningsmetoder och processer drar till sig ett snabbt ökande intresse, vilket visar sig i en explosion av processtandarder och ett växande antal böcker om ledning av programvaruutveckling. En rörelse bort från formalism och stringens, från centraliserad arkitektur mot gradvis tillväxt av programsystem inom ramen för en kollektiv process är vad Coplien tycker sig se. Av mycket att döma kommer programvaruindustrin alltmer till insikt om att dessa fenomen är verkligheten bakom de historiska illusionerna om strikt formalism och mästerlig planering. Programvaruutveckling sker de facto alltid genom gradvis tillväxt och sällan på en grund av genomgripande planering. Detaljplanerad utveckling kan leda till tekniska problem eftersom det är omöjligt att förutsäga hur ett program-avsnitt kommer att användas i framtiden, men också till samverkansproblem i organisationen eftersom en fullständigt planerad utveckling alienerar de utvecklare som inte också är planerare.

[3] i USA; i Europa och särskilt i Skandinavien hade konceptet om än inte själva termen varit i bruk ett helt decennium.

Copliens analys är intressant och provocerande. Man kan utgå ifrån att hans idéer inte får till följd att all forskning om formellt stringent programvarukonstruktion avstannar, och kritikens överdrifter åstadkommer därför kanske ingen stor skada. Men det ligger realism i hans konstaterande att det är människor, ofta organiserade i stora grupper, som faktiskt utvecklar programvaran, och om de inte känner sig motiverade och delaktiga i arbetet blir kvalitén på resultatet antagligen inte högre än den var i produkterna från den klassiska löpandebandsfabriken. Copliens observation att mycket få system, om ens några, utvecklas i en enda väldig ansträngning och sedan går i oavbruten drift tills det är dags att byta till nästa generation, är fundamental och detta faktum måste få till följd att en långt större tyngd än nu läggs vid att skapa strukturer och metoder för gradvis, evolutionär, utveckling av programvara.

3.1.3 Interoperabilitet mellan programsystem

Med interoperabilitet menas förmåga till samarbete mellan programsystem som kan vara skrivna i olika språk, följa skilda gränssnittsstandarder och exekvera på olika plattformar i form av operativsystem och maskinvara [18]. Begreppet brukar beskrivas med utgångspunkt från client-serverprincipen, där en användarprocess körs på ett klientsystem och utnyttjar gemensamma resurser som finns i ett betjäningssystem (server). I client-server-principen ingår mekanismer för hopkoppling mellan klient och betjäning som brukar kallas stickpropp (plug) och vägguttag (socket). Anpassningen, kompatibiliteten, mellan "plug" och "socket" bestäms av gemensamma datatyper och protokoll. Typkompatibilitet eller statisk kompatibilitet innebär att data kan överföras mellan programmen utan att förvanskas, medan gemensamma protokoll, s k dynamisk kompatibilitet, innebär att programmen tolkar innebörden av sekvenser av överförd information på samma sätt.

Språk för gränssnittsdefinition som förutsätter typkompatibilitet ger effektiv och säker kommunikation mellan programsystem. CORBA [19][20], avsnitt 4.5.1, tillåter interoperabilitet mellan olika programmeringsspråk (t.ex. C, C++ och Java) och operativsystem genom att sömlöst översätta typade strukturer (objektstrukturer) mellan delsystem skrivna i olika programmeringsspråk och exekverade i olika omgivningar. Typskillnader kan också hanteras genom typöversättning vid exekveringstillfället. Medlarteknik (eng. *mediators*, avsnitt 3.2.6) kan tillhandahålla sådan dynamisk typöversättning genom att tillåta att vyer av objekt definieras i termer av andra objekt från olika datakällor.

De två huvudsakliga metoderna för att åstadkomma interoperabilitet är *standardisering* respektive *överbryggning av gränssnitt*. Gränssnittsstandardisering innebär att information som skall överföras mellan klient och betjäningssystem översätts till en gemensam standardiserad representation, medan överbryggning innebär direktöversättning i båda riktningarna. Den förra tekniken är mer allmänt användbar, eftersom m olika klientgränssnitt och n betjäningsgränssnitt behöver endast m+n översättningsprogram till och från den gemensamma standarden, medan överbryggning kräver m*n översättningsprogram. CORBA använder t ex en gränssnittsstandard, IIOP, till vilken alla data som skall skickas mellan klient och betjänt översätts.

Överbryggningsteknik ger å andra sidan större flexibilitet, eftersom den kan och måste anpassas till varje givet par av klienter och betjäningssystem, och därmed kan ta hänsyn till dessas speciella funktionella krav.

3.1.4 För- och nackdelar med evolutionär systemutveckling

Framgångsrikt tillämpad evolutionär systemutveckling skulle leda till att utvecklingsresultatet vid varje tillfälle är grundat på erfarenhet av faktisk användning, inklusive bedömningar av vad som fungerar väl och mindre väl samt vad som saknas helt. Denna bedömning görs naturligtvis bäst av systemets användare, under förutsättning att de har adekvat utbildning och erfarenhet för uppgiften, men det är viktigt att inse att användare endast ser systemets yta och yttre funktionalitet (eller förmåga), och därtill oftast bara vissa delar av den, och att användarbedömningar därför bara kan ge en del av det beslutsunderlag som krävs för beslut om hur man ska gå vidare.

Nackdelar med en evolutionär modell är att ett slutdatum för processen inte kan anges från början, att processen sällan leder till användbart resultat förrän flera etapper har genomförts, och att metoden därför kräver en uthållig beställare. Den mest uppenbara fördelen är att realismen i planeringen kan öka dramatiskt och att högre beslutsfattare kan delta i processen på ett sätt som motsvarar projektets betydelse för organisationen, utan att hamna i händerna på den ena eller andra parten i utvecklingsprojektet.

3.1.5 Osäkerheter

Ledningssystemet skall ytterst öka förmågan att fatta rätt beslut i rätt tid. Det går alltid att ifrågasätta om komplexa informationssystem kan göras tillräckligt robusta och tillförlitliga för att verkligen åstadkomma detta. Det är ännu obevisat att evolutionär utvecklingsmetodik, med sin eftersträvade flexibilitet och decentralisering, kan styras och kontrolleras tillräckligt väl för att leda till och upprätthålla väl fungerande och säkra system med adekvat och modern funktion. Det finns också frågetecken kring om det är möjligt att i tillräcklig grad anpassa tekniken till människans behov. Kostnadseffektiviteten för ledningsstödsystemets olika delsystem och komponenter är ytterligare en osäkerhet.

3.2 Metoder

Stora men gradvisa förändringar sker av metodik och teknik för utveckling av komplexa programsystem. Några exempel på aktuella utvecklingsområden är:

- objektorientering
- återanvändbar programvara
- distribuerade grupputvecklingsmiljöer
- robusta distribuerade system
- högre ordningens program, som kan användas för att stödja specificering och utveckling av andra datorprogram.

Några av dessa områden kommer att beröras i fortsättningen. Vi har valt att dela in vår beskrivning i två avsnitt. I det första diskuterar vi metoder och i det andra komponenter, verktyg och infrastruktur. Denna indelning är naturligtvis ganska godtycklig. Metodikansatser behöver stöd i form av programverktyg och utvecklingsomgivningar, och en metodik som t ex objektorientering är ganska ointressant utan stöd i programspråk och utvecklingsverktyg. Med andra ord är det kombinationen av metodik och verktyg som, tillsammans med goda organisatoriska förutsättningar, möjliggör effektiv systemutveckling. Metodiken och verktygen bör vara in i minsta detalj anpassade till varandra.

Så är dock inte alltid fallet i praktiken, delvis därför att samstämda utvecklingsmiljöer kan vara ganska dyra, åtminstone för mindre företag, men också därför att val och utveckling av metodik alltför länge ansetts vara en intern fråga för varje konsult- och utvecklingsföretag. Om en användarorganisation som Försvarsmakten ska kunna ta eget övergripande ansvar för sin systemutveckling, vilket är en förutsättning för den evolutionära modellen, krävs istället att kompatibla och helst samma metoder och verktyg används i alla projekt som ingår i en evolutionär utvecklingsprocess, dvs att det är kundens utvecklingsstrategi och inte leverantörens som avgör vilken metodik som skall tillämpas. Då är det förstås en fördel om man kan välja en metodik som är en global de-factostandard.

3.2.1 Metoder för verksamhets- och systemmodellering

Under de senaste två decennierna har ett stort antal metoder för specificering av programsystem föreslagits. I en artikel i *ACM Computing Surveys* [33] ger R Wieringa en översikt som omfattar en analys av inte mindre än 25 sådana specificerings-metoder, varav19 är baserade på *objektorienteringsparadigmen*. De övriga är s k *strukturerade* metoder, en äldre paradigm. De olika metoderna bygger i sin tur på ett antal tekniker, som i artikeln grupperas i tekniker för specificering av extern interaktion respektive för intern nedbrytning (dekomposition). Teknikerna för extern interaktion indelas i sin tur i tekniker för specificering av funktion, beteende och kommunikation. Författaren ger sedan konkreta exempel på olika typer av tekniker, hämtade från olika specificeringsmetoder. Analysen visar att många av metoderna har stora principiella likheter men också stora syntaktiska och semantiska skillnader.

1996 presenterades ett standardförslag, *UML* (*Unified Modeling Language)*, som är en objektorienterad metod som skapats som en kompromiss mellan några av de dittills mest använda metoderna. Wieringa visar att inte heller UML lyckats uppnå två viktiga designmål, nämligen enkelhet och formell semantik (dvs formell definition av funktionellt beteende, vilket i princip innebär att metoden kan exekveras av en dator, och att det finns ett entydigt sätt att beskriva hur denna exekvering skall ske). Detaljerna i Wieringas översikt är knappast intressant för andra än specialister, men artikeln har ändå ett stort värde genom att den sammanfattar två decenniers ansträngningar att åstadkomma en universellt användbar systemspecificerings-metodik.

I en artikel i ett specialnummer om UML i tidskriften *Communications of the ACM* [34] beskrivs hur UML hittills utvecklats och vilka planer som gäller för de två närmast planerade utgåvorna, UML 1.4 (en mindre revidering som planeras till år 2000) och UML 2.0, en omfattande ny utgåva som planeras till år 2001. Artikel-författaren, C Kobryn, medger att UML råkat ut för ett antal oönskade sidoeffekter i samband med sin ovanligt snabba färd genom standardiseringsprocessen. Förutom de brister som tidigare nämnts, och som Kobryn kallar ”incomplete semantics and notation for activity graphs” respektive ”standard elements bloat”, pekar han också på att UML idag inte är byggt på en strikt metamodell, vilket gör det svårt att integrera UML-utvecklingen med andra standardiseringsprocesser som pågår parallellt. Dessa brister vill man försöka angripa under arbetet med att definiera UML 2.0, som också kommer att innehålla utökade funktioner inom ett antal områden, bl a komponent-baserad utveckling med *Enterprise Java Beans* och *DCOM*. Författaren ser naturligt-

vis en ljus framtid för UML men medger också att metodens omfattning är så stor att den överväldigar en del användare.

Vi vill trots dessa nackdelar ändå hävda att Försvarsmakten bör basera sin framtida systemutveckling på denna standard, som av många tecken att döma är på väg att konkurrera ut de flesta äldre medtävlare. Nedan presenterar vi därför, med generöst medgivande av *Metria GIS-Centrum*, ett utdrag ur beskrivningen av Metrias utvecklingsmodell Symmetria som är grundad på UML och *Rational Rose*-metoden, det sistnämnda en processbeskrivning och ett utvecklingsverktyg för UML-modellering.

3.2.2 Symmetriametoden

En vanlig motivering till att införa modellering som en projektaktivitet i systemutveckling är att ”ingen skulle komma på tanken att bygga ett hus utan ritningar”. Detta gäller särskilt för det fall då huset är stort och bygget inbegriper flera företag och konstruktörer. Om arbetet utförs av en person som gör allt arbete åt sin kund, är behovet av en strukturerad utvecklingsmetod inte lika stort som om många utvecklare gemensamt skall få ett stort projekt i hamn.

Att arbeta på ett gemensamt sätt i hela organisationen underlättar utbyte av både information (t ex programkod och goda exempel från tidigare projekt) som utvecklarresurser. En ytterligare vinst är den kvalitetsmärkning som användning av en välkänd utvecklingsmetod innebär.

Nära kopplat till utvecklingsmetoden är valet av arbetssätt. Modellering, dokumentation och testning är viktiga arbetsmoment men det är också viktigt att återkoppla till tidigare utfört arbete. Att remodellera, bygga om, modifiera kodstruktur etc bör ses som naturliga arbetsmoment och som något som bör ges resurser redan vid projektstart.

Ett system kan inte anses färdigt förrän dokumentationen föreligger. Man måste också se till att dokumentationen kan läsas av alla som behöver den. Detta underlättas naturligtvis av mallar och gemensamma verktyg men också av att man tar del av hur andra projekt har arbetat och använder detta som utgångspunkt.

Det huvudsakliga problemet vid införande av en arbetsmetod är inte metodutvecklingen, dokumentationen eller utbildningsarbetet. Flaskhalsen sitter istället vid att få personalen att börja använda den nya metoden praktiskt istället för att hålla fast vid tidigare invanda arbetssätt.

Det bästa sättet att införa modellering är att helt enkelt lägga in detta som en aktivitet i projektplaner och tilldela den rikligt med tid. Inled med en längre modelleringsfas som avslutas först när hela uppgiften är genomanalyserad. Försök sedan konstruera en första version av systemet så att det går att verifiera vilka delar av modelleringen som var korrekta och vilka som behöver omarbetas. Återgå sedan till modellering och dra nytta av de vunna erfarenheterna, och iterera på detta sätt ett par gånger under projektet.

Steg 1 - Kravanalys enligt användningsfallsprincipen
I varje systemutvecklingsmetod är ett självklart första steg att man tar reda på vad det egentligen är som skall utföras. Man fångar upp de krav som finns på det aktuella systemet och beskriver dessa i en funktionsspecifikation. Specifikt för alla UML-baserade utvecklingsmetoder är att stor vikt lägges på detta första steg. Man säger att en UML-baserad metod skall vara *use-case driven* – användningsfallsbaserad. Hela utvecklingsprocessen skall baseras på de *användningsfall* som identifieras i denna första fas. Ett användningsfall kan sägas vara *ett sätt att använda systemet som passar för en kategori av användare* och kan i praktiken bestå av en funktion eller en sammanhängande grupp av funktioner.

Ett effektivt sätt att identifiera användningsfall är att man i en mindre grupp genomför ett kravanalysmöte. Input till mötet bör vara kundens önskemål, gruppens ingående kompetens samt frågeställningarna:

- vilka är aktörerna?
- vad skall respektive aktör kunna göra i systemet?

Viktigt att tänka på är dels att användningsfallen ges lämpliga namn och dels att de görs lagom stora. Blir indelningen för grov, så att systemet bara består av ett par användningsfall, förloras mycket av poängen med att bryta ner systemet medan en alltför detaljerad uppdelning leder till dålig översiktlighet.

Gränssnittsprototyp
Ett huvudsyfte med att arbeta med användningsfall är att stärka kommunikationen mellan utvecklare och kund. Tanken är att en icke programmeringskunnig person (t ex kunden) lättare skall förstå en sammanställning av användningsfall än en tekniskt orienterad funktionsspecifikation. För att underlätta kommunikationen ytterligare kan man tidigt i projektet ta fram en prototyp på det grafiska gränssnittet. Slutresultatet, i den mån man kan tala om slutresultat i en iterativ process, av denna första fas blir sedan dels sammanställningen av användningsfall och dels gränssnittsprototypen.

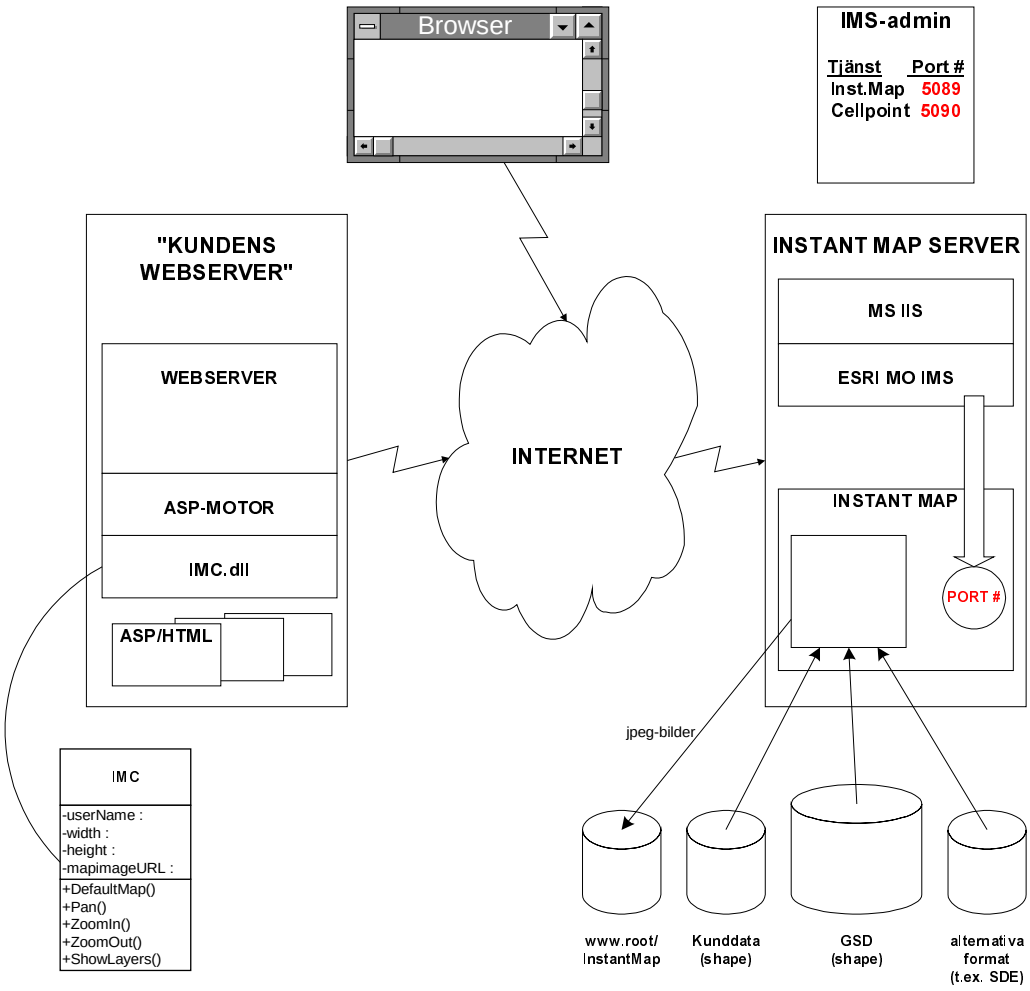
Steg 2 – Användningsfallsanalys
Huvuduppgiften i användningsfallsanalysen är att utgående från användningsfall och en gränssnittsprototyp bygga upp en logisk struktur för systemet. I denna fokuseras inte i första hand detaljfrågor utan mer analys och modifiering av de olika användningsfallen och hur dessa kan realiseras. Inom objektorienterad metodik talar man ofta om s k treskiktsutveckling, vilket innebär att man separerar kod och klasser som behandlar respektive användargränssnitt, logik och lagringsproblematik. I användningsfallsanalysen ligger fokus i första hand på mellanskiktet – de logiska klasserna. Meningen är att man efter detta steg skall ha en god bild över vilka klasser som behövs och hur logiken skall delas upp mellan dessa. Däremot ligger frågor om hur klasserna skall detaljkonstrueras och implementeras utanför.

Steg 3 – Komponent- och systemdesign
När en genomarbetad användningsfallsbeskrivning, en gränssnittsprototyp och ett utkast till klasstruktur har framställts, är det dags att bygga själva systemet. Detta arbete bör utföras med åtminstone två huvudmålsättningar. För det första skall systemet detaljkonstrueras, programmeras och dokumenteras. För det andra är det önskvärt att det som utvecklas går att återanvända, bygga ut och dela upp så att kommande projekt kan dra nytta av resultatet.

Arbetet stöds i båda fallen av en komponentbaserad utvecklingsmetodik. Man talar ibland om kontraktsbaserad utveckling (eng. *design by contract*). Med detta avses att man tilldelar programmeraren av ett delsystem en uppgift (ett kontrakt) som dennes delsystem skall lösa. Indata är givna och utdata skall se ut på ett specificerat sätt men delsystemet blir en ”svart låda” som definieras av sitt kontrakt (uppgift, indata och utdata) och kan alltså när det är färdigt brytas ut som en komponent.

När utvecklingen är färdig bör varje klass tillhöra dels ett specifikt skikt i treskiktsuppdelningen och dels (minst) ett användningsfall. Således behöver man kunna åskådliggöra:

- hela systemet som en mängd av subsystem och varje subsystem som en mängd av klasser
- lagringsstrukturen, d.v.s. någon form av datamodell, och det bör vara enkelt att tolka vilka klasser/objekt som lagras i vilken tabell
- en modell över systemets strukturella uppbyggnad:



Figur 1 Modell över ett systems strukturella uppbyggnad

Steg 4 - Testning

Slutligen skall en testfas läggas in i planeringen. Avsikten är dock *inte* att testningen skall påbörjas efter det att systemet är färdigbyggt och paketerat. Tvärtom bör en

första plan för hur systemet skall testas skapas redan på användningsfallsstadiet. Användningsfallen är ju beskrivningar av vad som skall kunna utföras i systemet och bör sålunda kunna utgöra grund för en systemtest. Varje komponent måste naturligtvis också testas utifrån det kontrakt som definierar den. En god utvecklingsregel är att ingen klass/komponent/delsystem får börja programmeras utan att man först tänkt igenom hur den skall kunna testas.

3.2.3 Ontologier och informationsmodeller

En metodik som kommer att behövas för att åstadkomma interoperabilitet är utveckling av gemensamma begreppsapparater, med en ursprungligen filosofisk-metafysisk term som används allt oftare inom datavetenskapen *ontologier*, för de berörda informationssystemen och deras databasinnehåll. Utan sådana metoder kommer man att hamna i en ändlös rad av beslutsproblem när man successivt vill koppla ihop allt fler system: om två existerande system har olika begrepp för likartade fenomen, hur ska det motsvarande begreppet representeras i det nya systemet ? Och om ytterligare ett system sedermera skall integreras, kommer problemet då att uppstå på nytt ?

Vad är en ontologi?

En ontologi är en gemensam begreppsapparat som används för att dela och återanvända kunskap genom en gemensam uppsättning termer och uttryck som definieras och ordnas i en sambandshierarki under olika ämnesområden.

Till ontologin bör finnas regler för att kombinera termer och relationer i syfte att definiera vokabulärutvidgningar. Ontologin kan även ses som en överenskommelser om gemensamma begrepp och som underlag för en kunskapsbas. De finns olika typer som relateras till innehåll, frågehantering, indexerande ontologier och metaontologier.

I ett militärt sammanhang kan en ontologi ge ett ensat språkbruk, områdesförståelse och möjlighet till ordnad utvidgning, t ex gentemot andra språk och militära organisationer. Bland andra nyttoaspekter på ontologier kan nämnas förbättrad:

- kommunikation mellan människa och organisation
- interoperabilitet mellan system

Erfarenheter visar att användningen av ontologier är trögstartad, främst p g a den stora insats som krävs innan man får resultat.

ATCCIS, Army Tactical Command and Control Information System

Det för Sveriges försvarsmakt sannolikt viktigaste exemplet på genomförd ontologiutveckling inom ledningssystemområdet är ATCCIS. ATCCIS är ett studieprogram finansierat av *NATO Supreme Headquarters, Allied Powers Europe* (SHAPE). Programmet har genomlöpt fyra faser sedan starten 1978; fas IV avslutas under 1999. Till skillnad från de tre huvudsakligen civila ontologiprojekt som beskrivs nedan, har ATCCIS-utvecklingen bedrivits utan att avsikten ursprungligen varit att representera den resulterande ontologin i ett formellt system. Termen ontologi används inte heller av ATCCIS, och en term som oftare förekommer i samband med konventionella informationssystem är *konceptuell modell* [39] eller *informationsmodell*.

ATCCIS övergripande mål är att öka interoperabiliteten hos NATO-ländernas ledningsstödssystem, dvs att öka systemens förmåga att dela och hantera information av gemensamt intresse för att därigenom maximera den operativa effektiviteten hos de kombinerade styrkorna. Interoperabilitet ska uppnås dels genom en gemensam generisk informationsmodell och dels genom en fullständigt specificerad mekanism för automatisk datareplikering mellan databaser.

ATCCIS förslag till gemensam informationsmodell, den s.k. *Land C2 Information Exchange Data Model* [40], är på väg att bli en *NATO Standardization Agreement*. ATCCIS informationsmodell är mest känd som *Generic Hub Data Model* (GH). Enligt [41] är det holländska ledningsstödsystemet ISIS och den portugisiska ledningsträning-anläggningen VIGIRESTE båda baserade på GH.

Det finns goda skäl att undersöka om och hur ATCCIS-standarden skulle kunna tillämpas i den svenska ledningssystemutvecklingen. Det är möjligt att man vid en granskning kommer att finna brister och andra svårigheter om man vill utnyttja denna begreppsapparat i det nya svenska ledningssystemet. Detta är vanliga problem i samband med standardisering, och visar betydelsen av att ha inflytande på eller i alla fall tidig kontakt med standardens utveckling.

Det primära målet för ATCCIS är att öka interoperabiliteten mellan ledningsstöd-system. I [42] föreslås dock att GH också kan användas vid utvecklingen av enskilda ledningsstödssystem. Det är nämligen sannolikt att ett sådant system byggs av programmoduler för t ex optimering, vägplanering, simulering m m. GH kan då vara den gemensamma modell som behövs för informationsutbyte mellan modulerna. Det har också föreslagits [43] att GH används som en s k *Reference Federation Object Model* (FOM) för att underlätta utvecklingen av HLA-federationer (varje HLA-federation måste ha en FOM, som på ett standardiserat sätt beskriver det data som utväxlas mellan federaterna, se avsnitt 4.1.1). Enligt [41] kommer en FOM baserad på GH att användas i NATO-projektet *DiMuNDS 2000* (*Distributed Multi-National Defense Simulation*).

GH syftar till att omfatta all information som behöver kunna utbytas mellan ledningsstödssystem. F n är dock informationen huvudsakligen inriktad på arméns behov. Detta inkluderar naturligtvis diverse information (t ex tillstånd, förmågor, geografiskt läge m m) om militära objekt som trupper, vapensystem, militär materiel och anläggningar etc. Informationen kan avse både generella egenskaper hos en viss *typ* av objekt och egenskaper som ett specifikt objekt har vid en specifik tidpunkt. GH kan även beskriva geografiska och meteorologiska förhållanden. Det är också möjligt att beskriva olika slags aktiviteter och händelser. Genom att länka samman enkla aktiviteter funktionellt eller temporalt kan t ex komplicerade handlingsplaner beskrivas. Till GH finns även utvidgningar som avser information om underrättelsetjänst, eldunderstöd och elektroniska kommunikationssystem.

I [40] beskrivs GH med både logiska och fysiska scheman. Det fysiska schemat (*Data Exchange Schema*, DES) dokumenteras med både *Structured Query Language* (SQL) och *IDEF1X*-diagram [44]. Enligt [41] finns GH också dokumenterad i *Unified Modelling Language* (UML*), Object Model Template* (OMT) och *eXtensible Markup Language* (XML). Den av ATCCIS utvecklade replikeringsmekanismen förutsätter att det data som ska föras mellan databaser följer DES.

GH definierar ett stort antal dataentiteter och deras attribut, samt relationerna mellan dataentiteter. De fem grundläggande dataentiteterna är följande:

Key Entity	Entity Definition	Information Category
OBJECT-ITEM	An individually identified object that has military significance. Examples are a specific person, a specific item of materiel, a specific geographic feature, a specific co-ordination measure, or a specific unit.	Contents (Who and What)
OBJECT-TYPE	An individually identified class of objects that has military significance. Examples are a type of person (e.g., by rank), a type of materiel (e.g., self-propelled howitzer), a type of facility (e.g., airfield), a type of feature (e.g., restricted fire area), or a type of organisation (e.g., armoured division).	
CAPABILITY	The potential ability to do work, perform a function or mission, achieve an objective, or provide a service.	
LOCATION	A specification of position and geometry with respect to a specified frame of reference. Examples are point, sequence of points, polygonal line, circle, rectangle, ellipse, fan area, polygonal area, sphere, block of space, and cone. LOCATION specifies both location and dimensionality.	Positioning and Shapes (Where)
ACTION	An activity, or the occurrence of an activity, that may utilise resources and may be focused against an objective. Examples are operation order, operation plan, movement order, movement plan, fire order, fire plan, fire mission, close air support mission, logistics request, event (e.g., incoming unknown aircraft), or incident (e.g., enemy attack).	Dynamics (How)

De grundläggande dataentiteterna kan i sin tur uppdelas underkategorier, t ex är dataentiteten OBJECT-ITEM uppdelad enligt:

Entity	Entity Definition
FACILITY	An OBJECT-ITEM that is built, installed, or established to serve some particular purpose and is identified by the service it provides rather than by its content (e.g., a refuelling point, a field hospital, a command post).
FEATURE	An OBJECT-ITEM that encompasses meteorological, geographic, and control features of military significance (e.g., a forest, an area of rain, a river, an area of responsibility).
MATERIEL	An OBJECT-ITEM that is equipment, apparatus or supplies without distinction as to its application for administrative or combat purposes (e.g., ships, tanks, self-propelled weapons, aircraft, etc., and related spares, repair parts, and support equipment, but excluding real property, installations, and utilities).
ORGANISATION	An OBJECT-ITEM that is an administrative or functional structure.
PERSON	An OBJECT-ITEM that is a human being to whom military significance is attached.

Alla dataentiteter beskrivs dessutom m h a olika attribut. T ex har dataentiteten PERSON attributen ID-nummer, födelsedatum, blodgrupp, kön och religion.

Andra exempel på ontologier
Exempel på stora internationella ontologier av olika typ och tillämpning är *WordNet*, *WordFact* och *Cyc*. Dessa tre är sinsemellan mycket olika, vilket torde ha att göra med att tekniken för att utveckla och representera ontologier fortfarande är på forsknings-stadiet.

WordNet
WordNet är en lexikal databas av engelska termer med gratis åtkomst via webbadressen http://www.cogsci.princeton.edu/~wn/. Engelska substantiv, verb, adjektiv och adverb är organiserade i synonymuppsättningar som var och en representerar ett underliggande lexikalt begrepp.

Exempel: ordet ”ontology” ger enbart en betydelse:

ontology -- (the metaphysical study of the nature of being and existence)

och några olika synonymer, koordinerade termer och *hypernymer* (this is a kind of...)

World Factbook
World Factbook är snarast ett aktuellt uppslagsverk om olika länder med kartor och vissa fasta rubriker. Det underhålls av CIA och är gratis åtkomligt via webbadressen http://www.odci.gov/cia/publications/factbook/index.html.

Cyc
Cyc (http://www.cyc.com/) tillhandahåller en kunskapsbas över sunt förnuft som bygger på drygt en miljon regler om konsensuskunskap om verkligheten. Cyc-systemets utveckling har innefattat tillämpningsprojekt som bidragit med avancerade fråge- och representationsspråk och slutsatsdragningsstöd som innefattar associationsutfyllnad till mänskligt diffusa frågeformuleringar. Exempel på sunt förnuft hos Cyc: träd står normalt utomhus, en avliden person handlar ej, ett glas med vätska skall bäras upprätt.

Exempel på tillämpningar:
1. Integration av heterogena databaser. Databaser är informationsrika men kunskapsfattiga. Men om man adderar lämplig information gällande tolkning och samband för varje databas och dess olika attribut så kan man utöka frågetolkens omvärldskunskap. På frågan ”Show me people who hold an advanced degree and live in New England” förväntas Cyc veta att vissa yrkestitlar som läkare, advokater och professorer kräver hög utbildning och att New England omfattar 6 USA-delstater. Denna kunskap utnyttjar frågetolken till att omtolka frågan till en uppsättning delfrågor mot ett antal ingående databaser och därefter sammanställa resultaten.
2. *Ontosaurus* är militär Cyc-tillämpning inom projektet *Loom* vid University of Southern California. Loom utgör ett språk och en omgivning för att konstruera intelligenta tillämpningar och dess kärna består av ett kunskapsrepresentations-system. I tillämpningen *High Performance Knowledge Base* (HPKB) ingår Ontosaurus som ett webbaserat gränssnitt mot en kunskapsbas (*HPKB Battlespace Ontologies*) som skall stödja avancerade resonemang om komplexa problem kring stridshantering och krisförståelse. Demonstrationen via nätadressen http://sevak.isi.edu:4676/loom/shuttle.html visar hur man kan navigera ända ned till programkod över hur olika termer och regler beskrivs i systemet.

3.2.4 Programvaruarkitektur
Inom datavetenskapen har begreppen (informations)systemarkitektur och det i stort sett synonyma programvaruarkitektur länge varit i bruk. Dessa begrepp rör den generella strukturen hos ett större programsystem, inklusive de principer och standarder som använts vid struktureringen. Betydelsen av sådan strukturering har länge varit erkänd [45][46] men först på senare år har programvaruarkitektur etablerats som ett eget forskningsområde inom data- och systemvetenskapen [47][48][49]. En rad arbeten inom områden som språk för modulgränssnitt, domänspecifika arkitekturer, språk för arkitekturbeskrivning, böcker om systemkonstruktion, forskning om formella grunder samt om systemomgivningar för arkitekturutformning vittnar om detta.

Ännu råder ingen enighet om vad begreppet programvaruarkitektur exakt bör omfatta. Tre betydelser som är vanliga i litteraturen nämns av Garlan och Perry [47]:

- arkitekturen hos ett visst system ("arkitekturen för detta system består av följande komponenter")
- en arkitektonisk stil ("detta system bygger på en client-server-arkitektur")
- arkitektur som kunskapsområde ("artiklarna i denna tidskrift handlar om arkitektur")

Ett försök till definition gjordes 1994 av en arbetsgrupp inom SEI (Systems Engineering Institute) vid Carnegie-Mellon University i USA. Den lyder:

"*The structure of the components of a program/system, their interrelationships, and principles and guidelines governing their design and evolution over time*."

Den stora kapitaltillgång som redan utvecklade programsystem utgör för många företag måste förvaltas på ett effektivt sätt om företagen ska kunna behålla och öka sin konkurrenskraft. Kunskap om programvaruarkitektur kan bidra till detta genom att erbjuda en begreppsapparat som gör det möjligt att identifiera områdes- (domän-) specifika abstraktioner, isolera återanvändbara moduler och andra arkitekturelement på olika systemnivåer samt åstadkomma interoperabilitet mellan olika programprodukter genom etablering av gemensamma ramverk och gränssnittsstandarder.

Ett annat påtagligt marknadsfenomen är tendensen att anskaffa snarare än själv producera programvara. Denna tendens understryker behovet av att kunna integrera programvara med olika ursprung (egenutvecklad, specialbeställd eller generell) till fungerande helheter. Här har programvaruarkitekturens perspektiv, metodik och begreppsapparat en viktig roll.

Slutligen är dagens krav på korta utvecklingstider och produktcykler ett viktigt argument för standardisering av programvaruarkitekturen och generalisering av programprodukternas struktur och syfte. Genom att etablera metodik för återanvändbarhet, gemensamma arkitektoniska ramverk samt mekanismer för effektiv integration och generering av programvara kan programvaruarkitekturens metoder och synsätt bidra till att dessa krav kan uppfyllas.

Framväxten av systemarkitektur som självständig vetenskapsgren har skett sent i förhållande till de försök som gjorts alltifrån slutet av 60-talet att etablera programvaruteknik som ett studieområde med sina egna teorier och metoder. Men eftersom komplexa programsystem för att alls kunna fungera kräver någon form av principiell strukturering och strikta regler för samverkan mellan olika delsystem och moduler, har bristen på vetenskapliga insikter måst ersättas med praktisk erfarenhet och intuition när ett större system konstruerats. Tillgången på sådan kompetens har många gånger varit mindre än behovet, delvis beroende på att ingen fungerande marknad funnits; officiellt diplomerade systemarkitekter är fortfarande en framtidsvision, fastän *Dataföreningen* sedan någon tid håller välbesökta kurser i ämnet. Om vi nu börjar röra oss från ingenjörskonst till vetenskap på detta område bör det finnas stora vinster att hämta.

En slutsats av ovanstående resonemang om systemarkitektur är att stora organisationer som behöver skydda sin information, sina materiella resurser och sina investeringar måste etablera hållbara principer för sin systemutveckling, men också måste vara medvetna om att sådana principer inte kan eller bör översättas till krav på en under lång tid (mer än 5 - 10 år) stabil och homogen systemstruktur. Arkitekturprinciperna bör ligga på en hög, så långt möjligt teknologioberoende nivå och bör inte innehålla detaljerade instruktioner om vilken hårdvaruleverantör eller programvarutillverkare som skall anlitas. Detta kan tyckas självklart men är idag, kanske mer än någonsin, en utopi.

3.2.5 Återanvändning av programvara

I artikeln *Toward an Engineering Discipline of Software Reuse* [35], som ingår i ett aktuellt specialnummer om arkitektur av tidskriften *IEEE Software*, hävdar författarna att utnyttjande av återanvändbara programvarukomponenter vid programvaruutveckling i industriell skala är en vision som ännu inte förverkligats annat än som isolerade exempel. För att återanvändning skall kunna bli en naturlig bas för all industriell programvaruutveckling krävs enligt författarna att följande delmål uppnås:

- en vetenskaplig bas, inklusive relevanta konstruktionsprinciper, för återanvändning behöver skapas
- brett accepterade tekniska standarder som kan överföra principerna till praktiska lösningar behöver etableras
- ”koherenta” ledningsmetoder och ledningsstandarder behöver införas, som kan medföra att lösningarna realiseras så att en rimlig nivå på produktkvalitet och processmognad vidmakthålls.

Enligt artikeln blir framgångsrika återanvändningsexperiment allt vanligare men framgång är inte typiskt, återanvändning är inte rutin, löftena inte infriade och en rad frågor motiverar fortsatt forskning. Det kan tyckas paradoxalt att återanvändning är ett legitimt forskningsområde: inom alla andra teknikgrenar är återanvändning en självklar del av god design, så självklar att den inte behöver uppmärksammas. Den kanske viktigaste orsaken till att programvaruproduktion visat sig vara så svår att industrialisera är att programvara är mycket informationsrik och därför inte kan karakteriseras, mätas och jämföras lika lätt som andra produkter.

Domänteknologi (*domain engineering*) är en nyckel till återanvändning. I detta begrepp ligger ett skalekonomiskt tänkande, som bl a innebär att en domän, ett tillämpningsområde, måste ha en viss ekonomisk storlek för att motivera en domänspecifik utvecklingsinsats och att omfattningen av utvecklingsinsatsen är beroende av marknadens storlek. Detta är dock inte den enda avgörande faktorn, utan frågor om frihandel vs. protektionism, respektive öppen marknad vs. monopol, spelar fortfarande en stor roll, inte minst på försvarsområdet. Framväxt av en komponentindustri inom en domän underlättas väsentligt, ja förutsätter i många fall, att marknadsmekanismerna fungerar väl inom domänen. Domänteknologiprocesserna blir också helt beroende av hur domänen definieras och avgränsas, vilket på programvaruområdet kan leda till speciella problem, eftersom programvara inte sällan är generisk till sin karaktär, dvs kan tillämpas inom ett stort antal olika domäner.

Domänavgränsningar kan ske ur flera perspektiv:

- gemensam expertkunskap kan definiera en domän, ett ”producentfokuserat domänexpertperspektiv”

- gemensam konstruktion leder till ett ”produktfokuserat programvaru-expertperspektiv”
- gemensam marknad leder till ett ”konsumentfokuserat marknadsförings-perspektiv”

Alla dessa perspektiv finns idag företrädda inom programvaruindustrin, vilket får anses vara en naturlig utveckling med paralleller inom andra, mer mogna industri-områden.

Av delvis samma författare är artikeln [36], som gör en detaljerad analys av återan-vändningsproblemet ur en rad olika aspekter. För den som vill fördjupa sig i metodi-ken kring återanvändning är detta en bra utgångspunkt även om artikeln nu är några år gammal.

I en rapport från Sveriges Verkstadsindustrier [37] presenteras resultat av en enkätundersökning bland svenska företag om användning av programvarukompo-nenter. Av rapporten framgår att inställningen till programvarukomponenter varierar starkt mellan olika företag och även mellan branscher. Rapporten innehåller, förutom en översikt över området och en lista på komponentleverantörer, bland annat konkreta praktiska råd och andra synpunkter som extraherats från företag som tillämpar komponentbaserad utveckling. Vi citerar en observation bland många: ”Det verkar svårast att uppmuntra komponentanvändning inom stora, stela organisationer. Teknikerna är där för långt från affärsverkligheten för att känna hur viktigt det är att snabbt få ut något ganska bra till en måttlig kostnad. Samtidigt är ofta cheferna inte tillräckligt insatta i tekniken för att trycka på.”

Återanvändning av programvara baserad på domänavgränsning är alltså en metodik som fått mycket uppmärksamhet i facklitteraturen under senare år. Att sådana metoder inte kommer att leda till dramatiska produktivitetsökningar hävdas av R L Glass i [38].

”As [the business of making software] matured, our field moved from an era when getting a good product out the door mattered most to an era when meeting a sometimes arbitrary schedule dominated.”

Eftersom utveckling av programvara som kan återanvändas är en tidskrävande verksamhet, minskar intresset för detta när minimering av tiden för leverans till marknaden alltmer kommit att bli ett dominerande mål. Men det finns också andra orsaker till att återanvändning inte är den universallösning på programvaruproblemet som många hoppas på:

”Most reuse scholars agree that if more than 20% of a component must be reworked for its new use, it is more efficient to start from scratch.”

I den mån detta är riktigt, är det naturligtvis ett stort hinder för rationalisering av programvaruproduktion genom återanvändning. Men det finns inget som säger att detta är en naturlag, oberoende av övriga omständigheter kring programvaru-utvecklingen. Nya utvecklingsmetoder, språk och system, skulle dramatiskt kunna ändra denna relation. Detta är i själva verket ett huvudargument för att införa objektorienteringsmetodik. Glass försöker skjuta även detta argument i sank:

”I know that people who use OO approaches tend to build such huge class libraries that the largest reuse problem they encounter is in searching the catalog to find what they need”.

Här tycker vi nog att Glass har hamnat i argumentnöd i sin debattartikel, även om det säkert är motiverat att utfärda en varning för överdrivna förväntningar på återanvändningsteknologin, som i likhet med alla andra kända metodmässiga framsteg leder till inkrementella effektivitetsvinster snarare än stora språng framåt, och detta bara då de tekniska och organisatoriska förutsättningarna är de rätta.

3.2.6 Medlare

Infrastrukturen i ledningsmiljön måste stödja ett distribuerat arbetssätt, som bland annat innebär att data och bearbetning samordnas och görs tillgänglig för flera samtidiga användare. Det måste därför finnas mekanismer som kan bevara konsistensen hos data och samtidigt utföra transport och bearbetning av data på ett effektivt sätt. *Databasteknik* [21][21][22], avsnitt 4.3, tillhandahåller möjligheter att uppfylla dessa krav, där databashanterare (eng. *DataBase Management Systems*, DBMS) erbjuder funktioner som stödjer både informationsmodellering och datarepresentation, såväl som indexering, optimering av åtkomst och konsistenskontroll av data. Dessa mekanismer innefattar bland annat en generell datarepresentationsmetod (kallad *datamodell* i databassammanhang), indexerbara datatyper, frågeoptimering och transaktioner.

I en dynamisk och heterogen omgivning blir det dessutom viktigt att utveckla metoder och system för att *kombinera* relevanta data från många olika informationskällor och sedan *transformera* och *presentera* dem i ett format som är anpassat och förståeligt för olika typer av användare och applikationer. Vidare behövs metoder och standarder för att *dela och utväxla data* mellan system och hjälpmedel.

Ett viktigt behov blir således förmåga att hantera *distribuerade* [23] och *heterogena* [24][25] databaser där det är möjligt att komma åt, söka och kombinera olika typer av data från olika datakällor distribuerade över datornäten. Hantering av distribuerade och heterogena data får en ökad betydelse därför att datornäten visserligen gör det lätt att komma åt nya datakällor, men samtidigt gör det svårare att bearbeta data på ett produktivt sätt. Forskningsinsatser behövs för att åstadkomma effektiv sådan *informationshantering i heterogena databaser*. Detta gäller inte minst för ledningsstödsystem där data från många olika typer av system, verktyg och datarepresentationer behöver kombineras.

För att så långt som möjligt upprätthålla *konsistens* mellan distribuerade databasnoder har olika metoder för s k *databasreplikering* utvecklats [28][29][30][31]. Särskilt svårt blir detta naturligtvis när noderna är mobila och sambandet intermittent [32], något som kommer att karakterisera taktiska system även i framtiden. Noderna i nätverket behöver ha tillräcklig autonom kapacitet för att en meningsfull aktivitet skall kunna upprätthållas även om sambandet med högre staber skärs av, fast ledningsförmågan i så fall naturligtvis reduceras. Avbrott i sambandet kan ha många orsaker. En del av dessa är mer eller mindre oundvikliga när den ena eller båda de kommunicerande noderna är mobila. Metoder för lägesprognosticering kan användas för att reducera

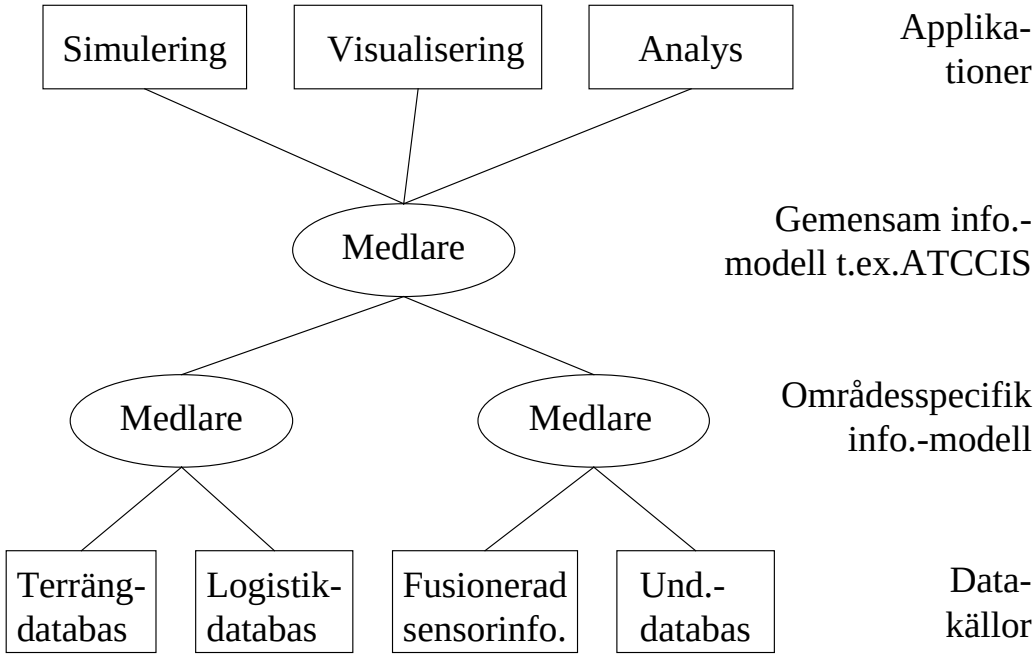
inkonsistens mellan databaser som representerar rörliga objekt eller mål, t ex fordon i rörelse längs en väg.

De olika delsystemen i ett nätverksbaserat system har i regel olika sätt att hantera data. För att underlätta informationsutbytet har ett antal *standarder* utvecklats för att dela och *utväxla data* mellan system och hjälpmedel; dessa innefattar kommunikationsprotokoll på grundläggande systemnivå som t ex CORBA, DCOM och RTI, databasgränssnitt som ODBC och JDBC, standardrepresentation av de data som skall utväxlas som XML, och standardiserade informationsmodeller (ontologier) som ATCCIS. Dessa standarder är viktiga steg på vägen mot enklare informationsintegrering. Emellertid kan man definitivt inte förvänta sig att beskrivning och kommunikation av all information kommer att kunna standardiseras. Dessutom utvecklas många olika standarder för liknande information av olika standardiseringsgrupper. Därför uppstår många olika *informationsöar* som överlappar och tillämpar olika informationsmodeller, vilket genererar behov av system och metoder för hantering av *heterogen information,* som kan kombinera överlappande och ibland motsägelsefulla data från olika datakällor.

En modern ansats för att integrera heterogena data är att införa *medlare* (eng. *mediators*) [26] mellan datakällor och de applikationer och verktyg som använder dessa. Medlarna utgörs av distribuerade *virtuella* databaser, *medlardatabaser*, vars innehåll är helt eller delvis hämtas från andra medlardatabaser och datakällor. Datakällorna omges av inkapslare (eng. *wrappers*) som översätter data till en uniform datarepresentation. I många fall kan en applikation dessutom vara datakälla för ytterligare ett lager av medlare etc. Medlardatabaserna består således av ett antal distribuerade *medlarnoder* som var och en innehåller ett *medlarsystem* bestående av mjukvara för att definiera virtuella *medlardatabaser*. Varje medlardatabas har en informationsmodell som beskriver dess innehåll, en *medlarmodell*. Medlarsystemen tillhandahåller för varje medlarnod gränssnitt mot applikationer och mot andra medlarnoder. Standardiserade gränssnitt (CORBA, ODBC etc) används i möjligaste mån.

Medlarsystem kan vara *aktiva* eller *passiva*. I det förra fallet kan systemet detektera förändringar (uppdateringar) i de anslutna databaserna och aktivt vidta förutbestämda åtgärder, t ex skicka meddelanden till andra i systemet ingående databaser om detta. En passiv medlare kan enbart förmedla information på förfrågan. Emellertid är många datakällor aktiva i den meningen att data inte bara kan hämtas från dem, utan att de på egen hand kan informera medlarsystemet om att data har ändrats eller tillkommit, som t ex sensorer kan. *Övervakare* (eng. *monitors*) tillhandahåller s k *aktiva regler* som *övervakar dataförändringar*, t ex för att kunna varsko när en signifikant trupprörelse har inträffat.

Medlartekniken kan göra det möjligt att utan omfattande omprogrammering koppla ihop tidigare separata, även geografiskt distribuerade och systemtekniskt heterogena, databaser till *ett*, för användaren homogent, välfungerande och säkert system. Även om medlartekniken ännu till stor del är på forskningsstadiet, har den börjat utvärderas i verkliga tillämpningar. Medlartekniken kan bli kärnan i en lösning på interoperabilitetsproblemet för ledningsstöds- och informationssystem.



Figur 2 Exempel på system av distribuerade medlare, datakällor och applikationer

Ett antal forskningsprojekt i USA [50][51][52] men även i Europa [53][54], arbetar med forskning om medlarteknik. Vidare bedriver IBM forskning inom området inom GARLIC projektet [55] och har även baserat produkten *DataJoiner* på en kombination av medlar- och relationsdatabasteknik.

Ett svenskt prototypsystem för aktiv medling är AMOS II (*Active Mediator Object System*) [56][57][58], som utvecklats av Tore Risch och hans medarbetare, numera vid Uppsala universitet. AMOS II är ett *distribuerat* medlarsystem där de virtuella medlardatabaserna kan vara distribuerade över ett nätverk och vara definierade i termer av varandra. De flesta andra medlarsystem är centrala i den meningen att en central medlardatabas integrerar sina datakällor direkt. Sådan central medling försvårar storskalig dataintegrering där information från ett stort antal datakällor integreras. En grundläggande idé med medlararkitekturen [26] som också pekats ut av forskningsprojekten DIOM [60] och PEGASUS [61] är att en medlare skall vara en *modul* med kunskap om hur ett litet antal datakällor integreras. Storskalig dataintegrering kan då åstadkommas genom att definiera medlarmoduler i termer av andra medlarmoduler. I en distribuerad omgivning innebär detta att en virtuell medlardatabas kan definieras i termer av andra virtuella medlardatabaser och/eller några datakällor. Flera forskningsproblem återstår emellertid att lösa för att kunna åstadkomma sådan storskalig dataintegrering. T ex behöver man utveckla nya distribuerade frågeoptimeringstekniker där databasfrågor kan omfatta många distribuerade datakällor, och där kommunikations- och datakvalitet kan variera [59].

3.2.7 Hantering av osäker information

För en militär beslutsfattare är det viktigt att under ett stridsförlopp kunna bedöma kvaliteten på den information som läggs till grund för bedömningar och beslut. När det gäller uppgifter från mänskliga källor är kvalitetsbedömningen ett mycket svårt problem, som underrättelsebedömare ständigt konfronteras med. Problemet blir ännu svårare om man vill använda sådana uppgifter som indata till en formell metod, en beräkningsalgoritm, eftersom det då krävs att man kan översätta en kvalitativ

kvalitetsuppgift (“möjlig ubåt”) till en kvantitativ, t ex ett numeriskt mätetal eller en sannolikhetsfördelning.

Om uppgiften däremot enbart är att beskriva kvaliteten hos innehållet i en databas med grunddata är lösningsprincipen förhållandevis enkel: man förser data med ett kvalitetsmått av specificerat slag. Möjligheterna att ange meningsfulla kvalitetstal varierar naturligtvis starkt med datamaterialets karaktär.

För att tillgodose rimliga kvalitetskrav vid utnyttjande av resultat av en simulering av t ex en stridssituation är problemet återigen mer komplicerat: man behöver dels förse alla indata till den aktuella beräkningsmodellen med tolerans- och andra kvalitetsmått, dels måste modellen utformas på ett mer sofistikerat sätt än som är normalt idag. Den måste från början förses med inbyggda feluppskattningar och bygga på noga genomtänkta giltighetskriterier. Härigenom är det åtminstone i princip möjligt att som resultat från en modell få inte bara en mängd (påstådda) resultatvärden utan också giltighetsområde för resultatet samt spridnings- eller andra kvalitetsmått för resultatvärdena. Kvalitetssäkring av modellresultat är ur teknisk synpunkt mycket nära besläktat med validering av beräkningsmodeller, ett område där det mesta återstår att göra åtminstone vad gäller militära systemsimuleringar.

Ett forskningsområde som blivit allt viktigare på senare år och som handlar om metodik och teknik för hantering och analys av data som är behäftade med många slags osäkerheter är *Management of Uncertainty*. En rad olika angreppssätt har utvecklats för att hantera sådana problem. Korrekt och adekvat användning av sådana vetenskapligt grundade metoder är en förutsättning för att trovärdiga datafusionssystem ska kunna utvecklas.

Det ligger utanför ramen för denna rapport att konkret beskriva de många olika metoder som utvecklats för hantering av osäker information. Däremot vill vi nämna en klassificering av olika slags osäkerhet som beskrivs av A Motro i [62]. De tre typer av osäkerhet som har störst betydelse för informationssystem är *fel*, *onoggrannhet* och *osäkerhet*.

Fel
Lagrad information är enligt Motro felaktig när den skiljer sig från den korrekta informationen och bör inte tolereras i ett informationssystem. Ett specialfall av felaktig information är *inkonsistens*, som föreligger om information representeras på olika, icke samstämda sätt i ett och samma system. Motro tillägger att denna syn på fel är relevant för *diskret* information som anger ett distinkt klassvärde.

Onoggrannhet
Lagrad information som avser en hel mängd av möjliga värden, där det sanna värdet är en medlem i mängden, uppvisar onoggrannhet. Informationen kan t ex vara *disjunktiv* (Johans ålder är antingen 31 eller 32), negativ (Johan är inte 30), eller vara försedd med felmarginaler (Johan är mellan 33 och 35 år). Extremfall av onoggrannhet är *precisa* värden och *odefinierade* värden (*null* values). Ett odefinierat värde anger att ingen information finns tillgänglig, men kan också uppfattas som att värdet är något av de möjliga värdena för den åsyftade egenskapen.

Osäkerhet
När vår information om omvärlden inte kan uttryckas med full säkerhet, behöver vi ett sätt att beskriva existensen och graden av denna osäkerhet. ”Johan är troligen 32” uttrycker osäkerhet men inte onoggrannhet. Vi skulle kunna säga ”Johan är mellan 0 och 200” och därmed överföra all vår osäkerhet till en säker men oprecis (och ointressant) utsaga.

Om vi vet att ett objekt existerar, kan en del eller all information som vi vill använda för att beskriva objektet i vår modell vara okänd. Informationskvaliteten ökar om man vet att informationen måste tillhöra en begränsad mängd alternativ. Den ökar ännu mer om varje alternativ kan förses med en trolighet att det verkligen är det sanna alternativet. Om detta tal är en sannolikhet är informationen *probabilistisk*, och om det betecknar möjlighet är den *possibilistisk*.

Sannolikhetsteori är förmodligen den äldsta formella metoden att beskriva osäkerhet och är lämpligast för att beteckna osäkerhet som är associerad med förekomst av olika värden på fixa variabler (”slumpmässighet”) [63]. Bayesiansk sannolikhetsteori, som utgår från en given *apriorisannolikhet* för en händelse och anger en precis metodik för att uppdatera denna till en *aposteriorisannolikhet* när man vet att andra, associerade, händelser har inträffat. Den har av både teoretiska och algoritmiska skäl kommit att få stark ställning för representation av osäkra samband och hypoteser om verkligheten.

Rough sets (”grova mängder”) är effektivt för att representera onoggrannhet eller brist på specificitet hos granulär information (man anger en undre och en övre gränsmängd för de objekt som har ett visst värde). *Fuzzy sets* och den besläktade ”möjlighetsteorin” (*possibility theory*) har utvecklats för att hantera en annan typ av osäkerhet, som kan kallas brist på klarhet. Fuzzy sets bygger på en utvidgning av den binära mängdläran till graderat medlemskap, och används mest för att representera mänskliga kvalitativa begrepp som ”ganska liten”, ”medelstor” osv). Dempster-Shafer´s *belief theory* (”tilltrosteori”) är en effektiv metodik för att representera information som är både slumpmässig och granulär.

3.2.8 Konstruktionsprinciper för säkerhet
En vanlig liknelse för datasäkerhet är att jämföra informationssystemet med ett hus. I ditt eget hem låser du dörren när du inte är hemma, släpper inte in vem som helst genom dörren och drar för persiennerna när du inte vill att grannarna skall se. En privatperson har inte samma krav på säkerhet som en militär anläggning. Ingen vill ha ständiga legitimationskontroller eller har ens råd med vakter, kameror och vakthundar utanför sitt hem. På samma sätt vill en privatperson inte ha avancerade inloggningsprocedurer och säkerhetsklassning av data i sin hemdator.

De högre militära kraven på säkerhet utesluter därmed de operativsystem som är avsedda för privatpersoner. En lägesrapport över våra egna styrkor får inte vara lika lättillgänglig som trädgårdssoffan hemma. Det handlar ofta om en balans mellan användbarhet och säkerhet. Ett annat vanligt fel är att säkerheten läggs in i efterhand. Säkerhet måste tas med från början som en grundläggande del av arkitekturen och integreras i alla delar av systemet.

Datorvirus är fortfarande ett vanligt problem och kräver ständigt nya uppdaterade versioner av antivirusprogrammen. Allt fler installerar brandväggar för att hålla ute

crackers som försöker bryta sig in. Som vanligt försöker man bota symptomen istället för att ta reda på de verkliga orsakerna till de här problemen. Det är tekniskt möjligt att konstruera datorsystem som sällan eller aldrig har problem med virus och mer eller mindre har inbyggda brandväggar. Detta kan man förhoppningsvis åstadkomma om man följer ett antal grundläggande säkerhetsprinciper.

Dessa säkerhetsprinciper är hämtade ur ett dokument som skrevs redan 1975 av Saltzer och Schroeder [64]. Redan då hade man goda insikter om säkerhet och principerna håller än. Rubrikerna på engelska har behållits för att det i flera fall inte finns någon bra svensk översättning.

- *Economy of mechanism*
  Säkerhetsarkitekturen skall vara så enkel och liten som möjligt. Även om arkitekturen som helhet kan vara komplex så bör varje del vara enkel för att möjliggöra överblick och för att man lättare ska kunna hitta brister och säkerhetshål genom att läsa programkoden rad för rad eller inspektera systemschemat.

- *Fail-safe defaults*
  Utgå från avsaknad av behörighet. Systemet bör alltså ha maximalt skydd från början och öppnas upp efterhand som tillgång till resurser behövs. I ett stort system kan det vara svårt att avgöra om en sällan använd resurs eller fil skall vara skyddad eller ej, varför det är säkrast om normalfallet är ingen behörighet. Dessutom upptäcks nekad behörighet snabbt av användarna vilket gör att fel upptäcks snabbt.

  Många brandväggar bygger på den här principen och installeras just för att de blockerar all trafik. Rent tekniskt borde man bygga in ”brandväggar” i operativ-systemen från början.

- *Complete mediation*
  Alla operationer i systemet måste kontrolleras med avseende på behörighet. Det innebär att den som avser initiera en operation måste autentiseras och kontrolleras i behörighetslistan innan operationen kan påbörjas.

- *Open design*
  Systemdesignen skall inte vara hemlig. Mekanismerna för skydd och säkerhet skall alltså inte baseras på okunskap hos den potentiella anfallaren, utan på hemliga nycklar och lösenord. Krypteringsalgoritmer och andra säkerhetsmekanismer - och dess källkod - bör alltså vara öppna för att kunna analyseras av många bedömare. Dessutom kan den som införskaffar ett system själv försäkra sig om att det verkar säkert genom att studera källkoden.

- *Separation of privilege*
  Åtminstone två villkor måste vara uppfyllda för att man ska kunna öppna ett låst system. Det kan innebära att två olika identifikationsmetoder bör användas för att identifiera en person, till exempel ett lösenord och inloggning från en specifik plats. Det kan även betyda att två olika nycklar ägda av två olika personer måste användas för att låsa upp.

- *Least privilege*
  Inga program och användare bör ha mer behörighet än nödvändigt för att de ska kunna utföra sitt arbete. Detta minskar risken för oavsiktliga skador på systemet. Man bör också ta bort rättigheter efterhand som de inte behövs längre. Denna princip är relaterad till *Fail-safe default*-principen som nämnts tidigare.

- *Least common mechanism*
  Minimera antalet delade variabler, minnesutrymmen och resurser, vilket innebär att alla processer i ett dator bör köras i separata minnesutrymmen. På detta sätt undviker man risken att information eller data läcker över till felaktiga ställen, och undviker att skapa dolda kanaler.

- *Psychological acceptability*
  Säkerhetsfunktionerna skall vara enkla att använda och säkerheten skall inte hindra arbetet i onödan. Detta krävs för att säkerhetsmekanismerna skall kunna accepteras och utnyttjas av användarna. Dessutom bör säkerhetsmekanismerna motsvara användarnas krav för att minska risken för misstag.

Att jämföra säkerheten i ett hus med säkerheten i ett datasystem är egentligen felaktigt eftersom de två konstruktionerna bygger på helt olika paradigmer. Utgående från en uppsättning ritningar och instruktioner bygger man ett hus, men ett datorprogram är instruktionerna i sig. Detta gör att man inte alltid kan ha samma tankemönster vad det gäller datasäkerhet som beträffande fysisk säkerhet. Av samma anledning kan man inte bygga mjukvarusystem med gamla ingenjörsmetoder utan måste använda en mer skapande metod som evolutionär systemutveckling.

4. Byggstenar, verktyg och infrastruktur

I detta kapitel kommer vi att presentera ett urval ur den stora mängd tekniska systemlösningar och komponenter som kan vara av värde i samband med utveckling av ledningsstödsystem. Vi vill peka på generiska metodområden som vi anser värda att uppmärksamma och analysera i samband med att standarder och ramverk för ledningsstödsystem specificeras.

4.1 Simuleringsstandarder

Distribuerad interaktiv simulering är, som nämndes i avsnitt 2.4, en teknologi som utvecklats med syftet att få ett antal självständiga och möjligen geografiskt utspridda simuleringsprogram att samverka i en gemensam simulering. Inledningsvis handlade det främst om att kunna samträna geografiskt utspridda grupper genom att koppla ihop att ett antal likartade realtidssimulatorer. Efterhand ökade dock ambitionerna och visionen är idag att få olika typer av modeller och simulatorer att samverka. För att uppnå detta utvecklades bl a *High Level Architecture* (HLA), som är en ursprungligen amerikansk försvarsstandard för hur olika simuleringsmodeller ska kunna samverka i en gemensam simulering. I detta avsnitt ger vi dels en beskrivning av HLA, dels en rapport om aktuella ansatser för att standardisera kopplingen mellan skarpa ledningsstödsystem och HLA-baserade simuleringar.

4.1.1 HLA-standarden

HLA-standarden är avsedd att vara en flexibel, återanvändbar programarkitektur för skapande av komponentbaserade distribuerade simuleringar. I princip ska således skarpa, virtuella och konstruerade simuleringar kunna kopplas ihop och samverka i en s k HLA-federation. En sådan består av tre komponenter:

- två eller fler simuleringsprogram, s k *federater,* som tillsammans bildar en *federation*;
- ett distribuerat operativsystem, kallat *Run Time Infrastructure* (RTI), som sköter all kommunikation mellan federaterna;
- ett standardiserat gränssnitt mellan federater och RTIn.

Utöver definitionen av gränssnittet federat-RTI består HLA-standarden av tio regler som måste uppfyllas av federater och federationer, samt standardiserade mallar (s k *Object Model Templates* (OMT)) för att beskriva egenskaper hos federater och federationer. Mer exakt måste varje federat ha en *Simulation Object Model* (SOM), som bl a anger vilka objekt och interaktioner (händelser) federaten kan modellera, och vilka av dessa som federaten kan ”publicera” eller ”prenumerera på”. Objekt och interaktioner av den förstnämnda kategorin är sådana som aktivt simuleras av federaten, medan objekt och interaktioner av den senare kategorin kan vara sådana som federaten kan ta emot information om. Analogt beskriver *Federation Object Model* (FOM) vilka objekt och interaktioner som en federation modellerar. Standardiserade federat-RTI-gränssnitt finns för CORBA IDL, C++, Ada 95 samt Java.

Ett program sägs vara HLA-anpassat om det har en SOM, och dessutom kommunicerar med RTIn i enlighet med vad som utlovas i SOMen. Det sistnämnda innebär bl a

att programmet måste ha anrop till RTIn för att sända respektive ta emot data om de objekt och händelser som programmet publicerar respektive prenumererar på.

HLA antogs 1996 som en teknisk standard för alla simuleringar inom US DoD. Detta innebär bl a att efter år 2001 får endast HLA-anpassade simuleringar användas. Sedan 1998 är dessutom HLA antagen som *Distributed Simulation Facility* av *Object Management Group* (OMG), den organisation som ansvarar för CORBA. HLA är dessutom på väg att bli en IEEE-standard.

Utvecklingen av HLA har främst drivits på av *US Defense Modeling and Simulation Office* (DMSO). För att uppmuntra användningen av HLA har DMSO dessutom stött framtagande av metoder och programverktyg för utveckling och exekvering av HLA-baserade simuleringar.

Dessa programverktyg, däribland en RTI-implementation, kan erhållas gratis från DMSO. Bortsett från DMSOs RTI är dock oberoende utvärderingar av existerande verktyg än så länge sällsynta.

HLA-standarden anses allmänt vara ett framsteg jämfört med tidigare ansatser, DIS och ALSP. F n har knappt hundra programvaror HLA-anpassats. Genomsnittstiden för HLA-anpassning av en programvara uppges vara mindre än sex personmånader. Det internationella intresset för HLA ökar också.

En hel del kritik har framförts mot DMSOs RTI. De som sysslar med realtidssimulering verkar inte nöjda med prestanda, och bättre dokumentation eller tillgång till källkoden är ett starkt önskemål. Kritiken mot HLA-standarden i sig är främst inriktad mot OMT, som anses brista när det gäller möjligheter att beskriva metadata, dynamiska egenskaper, aggregeringar m m. Vissa anser också att HLA inte i tillräckligt hög grad anammat objektorienterade koncept.

Kritiken mot OMT är intressant, eftersom den avspeglar en insikt om att interoperabilitet handlar om betydligt mer än kommunikation. I *HLA and Beyond: Interoperability Challenges* [69] gör man en distinktion mellan *teknisk* och *substantiell* interoperabilitet. Det förstnämnda anses uppnått ”when the federation is integrated, and individual federates are passing data in accordance with the federation object model (FOM)”. Som medel för att åstadkomma detta nämns tekniska standarder (t ex HLA), kompatibel hårdvara, koordinerad tidshantering m m. Substantiell interoperabilitet kräver att federaterna samverkar på ett meningsfullt sätt med hänsyn till syftet med federationen. Detta ställer krav på federaternas interna representation av objekt (attribut, beteende, detaljnivå), temporal och spatiell upplösning m m. Om t ex federater har olika temporal upplösning, så kan detta ge en orättvis bild av olika objekts stridsförmåga.

Om distribuerad simulering ska bli en framgång, måste substantiell interoperabilitet kunna uppnås på ett kostnadseffektivt sätt. Härvidlag tycks HLA på sitt nuvarande utvecklingsstadium utgöra en förutsättning snarare än en lösning. I nästa avsnitt beskrivs aktuella ansträngningar för att öka interoperabiliteten mellan skarpa ledningsstödssystem och HLA-baserade simuleringar.

Ett annat potentiellt interoperabilitetsproblem rör RTIn. HLA fastlägger endast gränssnittet mellan federater och RTIn, och säger t ex inget om enligt vilket protokoll RTIn ska vidarebefordra data mellan federater. Detta kan medföra bristande interoperabilitet mellan olika leverantörers RTI implementationer. Olika strategier för att hantera detta diskuteras i [70].

4.1.2 Internationella ansatser att koppla ledningsstödssystem och simuleringar

I USA har en mindre industri byggts upp kring försvarets behov av mjukvarugränssnitt mellan skarpa ledningsstödssystem och simuleringar [71]. Dessa gränssnitt tillkommer nästan alltid i efterhand och är sällan återanvändbara, dvs de duger endast för att länka ihop specifika ledningsstödssystem med specifika simuleringar. Orsaken till detta är bl a bristen på gränssnittsstandarder. Dessutom finns det ofta fundamentala skillnader mellan ledningsstödssystem och simuleringar beträffande vilken information de har och hur denna information representeras. Informationsutbyte kan då kräva komplicerade översättningsmekanismer.

De pågående ansträngningarna att standardisera både ledningsstöds- och simuleringssystem bör förhoppningsvis underlätta kopplingen mellan dessa. Framtida amerikanska ledningsstödssystem måste följa *Joint Technical Architecture* (JTA). JTA påbjuder att ledningsstödssystem måste vara baserade på *Command and Control Core Data Model* (C2CDM) och *Defense Information Infrastructure* (DII) *Common Operating Environment* (COE). DII COE består bl a av en samling av godkända programvaror från operativsystemnivå till applikationsnivå. Som nämnts i föregående avsnitt, måste framtida simuleringar följa HLA-standarden.

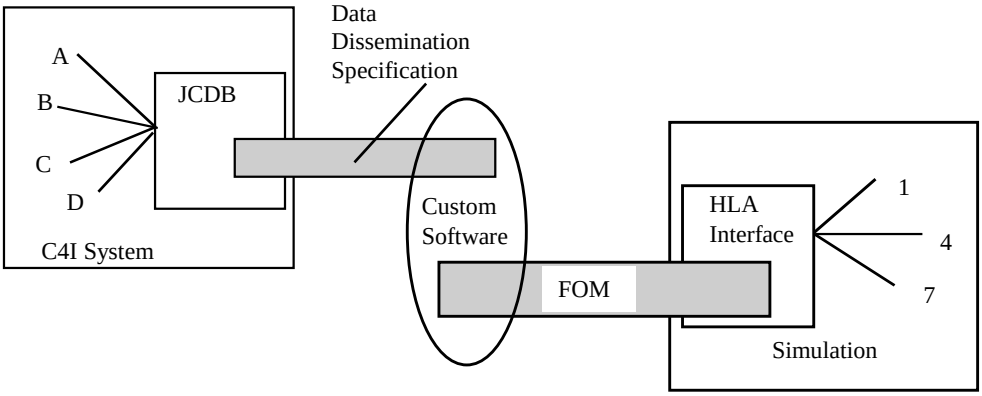
Någon allmänt accepterad standard för koppling mellan ledningsstödssystem och HLA-baserade simuleringar finns dock ännu inte. I det följande redogörs för olika ansatser att förenkla denna kopplingen som presenterats vid *Simulation Interoperability Workshop* (SIW). SIW arrangeras *av Simulation Interoperability Standards Organization* (SISO). SISO stöds ekonomiskt av *US Army Simulation Training, and Instrumentation Command* (STRICOM), DMSO samt ett antal privata företag.

En förutsättning för en gränssnittsstandard är att man på en konceptuell nivå kan beskriva hur informationsutbytet ska gå till, och vilka typer av information som ska utbytas. I [72] presenterar Hieb och Staver en *C4I Interface Technical Reference Model* med syfte att på en konceptuell nivå beskriva vilken typ av information som måste utväxlas mellan ledningsstödssystem och simuleringar. De finner att informationen kan uppdelas i tre klasser: persistenta och icke-persistenta data, samt styrdata för simuleringen. Exempel på persistenta data är terräng- och väderdata, och andra data som sällan eller aldrig ändras under ett simuleringsförlopp. Icke-persistenta (eller transienta) data är order, rapporter, uppdateringar av objekttillstånd m m. Styrdata är exempelvis kommandon för att starta och stoppa en simulering, starta och stoppa loggning av data, synkronisering m m.

En annan referensmodell presenteras av Ressler, Hieb och Sudnikovich [73]. I den ingår en aktivitetsmodell för informationsutbytet. Den innehåller aktiviteter som översättning mellan informationsmodeller, formattering, sändning, mottagning, händelsestyrning, nätverkshantering och datainsamling. Dessutom föreslås att den information som utbyts kan uppdelas i fyra klasser: ledningsinformation (organisationer, utrustning, planerade eller pågående operationer), kommunikations-

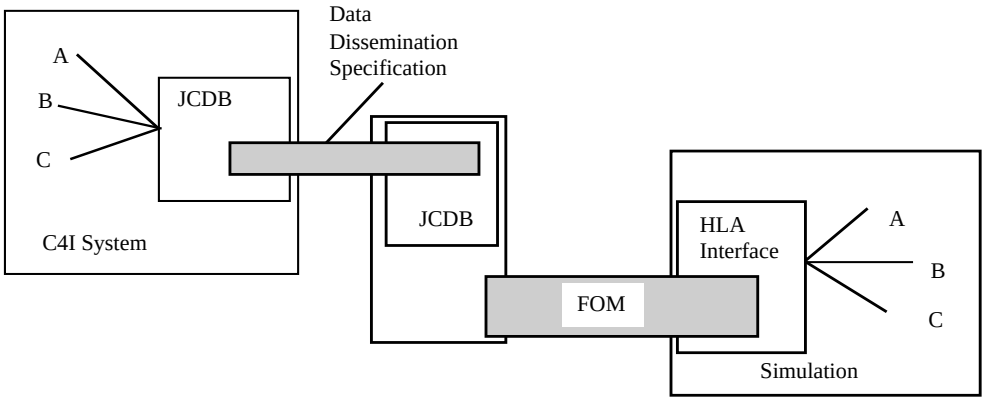
information (det logiska och fysiska nätverket för ledningsstöds- och simuleringssystemet), omgivningsinformation (terräng, luft, hav, anläggningar, väder, etc.) och styrdata för simuleringen (se ovan). Man konstaterar också att det inte för någon av dessa informationstyper finns standarder som används inom både ledningsstödssystem och simuleringar.

Hieb och Blalock [74] diskuterar två olika gränssnitt som illustreras av nedanstående figurer. Enligt dem kommer samtliga *US Army Battle Command Systems* (ABCS) i framtiden att använda informationsmodellen *Joint Common Data Base Transformation Data Model* (JTDM), och kommunikation mellan dessa system kommer i ökande utsträckning att ske genom databasreplikering. I första bilden sker kommunikationen mellan ledningsstödssystemet och simuleringen i två steg genom införande av en federat som sköter översättningen mellan JTDMs informationsmodell och den aktuella FOMens objektmodeller. Enligt Hieb och Blalock har dock stora skillnader mellan JTDMs informationsmodell och existerande simuleringsmodeller påträffats. Enligt W Sudnikovich, *US Army Communication-Electronics Command* (CECOM), har man också haft stora svårigheter att i HLAs OMT beskriva de associationer, relationer och aggregeringar som används i informationsmodellerna för deras ledningsstödsystem. Översättningen kan därför bli komplicerad.



Figur 3 Interoperabilitet mellan ledningsstödsystem och simuleringssystem genom översättning av ledningssystemets informationsmodell

För att undvika potentiellt komplicerade översättningsmekanismer förordar Hieb och Blalock istället att JTDM används även av simuleringsprogrammen. Detta illustreras i nästa bild. Här representeras t.ex. personer på samma sätt i både ledningsstödssystem och simuleringsprogram. Det behövs således inga komplicerade översättningsprogram. Även här sker kommunikationen mellan ledningsstödssystemet och simuleringen sker i två steg genom införande av en federat som i detta fall innehåller en JTDM-baserad databas.



Figur 4 Interoperabilitet mellan ledningsstödsystem och simuleringssystem genom utnyttjande av gemensam informationsmodell (JTDM)

En annan möjlig lösning är att ett RTI-gränssnitt installeras på samma plattform ledningsstödssystemet. Detta alternativ anses dock vara oacceptabelt för dem som utvecklar ledningsstödssystemen.

Krusche och Tolk [41] föreslår att *ATCCIS Generic Hub Data Model* (GH) (se avsnitt 3.2.3) används som en gemensam informationsmodell, och att kommunikation mellan olika system (simuleringar eller ledningsstödssystem) görs genom ”*data mediation functions*” som översätter mellan systemspecifika representationer och GHs representationer. Översättningarna kan då vara komplicerade, men med tanke på redan gjorda investeringar ter sig Krusches och Tolks förslag mer realistiskt än den av Hieb och Blalock förordade lösningen.

SISO har nyligen inrättat en s k *Implementation Study Group* för att studera kopplingen mellan ledningsstödssystem och simulering. Gruppen planerar att ge ut en slutrapport under hösten 2000. Rapporten kommer att bl a att innehålla rekommendationer för utveckling av ett standardiserat gränssnitt mellan ledningsstödssystem och simuleringar, samt förslag på relaterade s k *Reference FOMs*. En Reference FOM (RFOM) är avsedd att underlätta utvecklingen av FOMs för specifika federationer inom en viss domän. Nyligen antogs t ex en RFOM för realtidssimulering som SISO standard. Sudnikovich et al. [75][76] har utvecklat en C4I FOM, som har goda utsikter att bli en RFOM för HLA-federationer där ledningsstödsystem ingår.

Sammanfattningsvis kan sägas att olika ansträngningar görs för att öka interoperabiliteten mellan ledningsstödssystem och HLA-baserade simuleringar. En indikation åt vilket håll utvecklingen går kommer förhoppningsvis mot slutet av år 2000.

4.2 Agentteknik

Agent är en relativt ny datalogisk term som förekommer i uttryck som multiagentsystem, mobila agenter och intelligenta agenter. Termen agent saknar vedertagen definition och de olika förslagen beror på vilket perspektiv användaren har. I denna rapport koncentrerar vi oss på uttrycket intelligenta agenter med beskrivningen ”en intelligent mjukvaruagent är ett autonomt datorprogram med uppgift att underlätta arbete över ett distribuerat, heterogent datorbaserat system”. För att uppnå detta

karakteriseras intelligenta agenter av självständighet, kommunikations- och inlärningsförmåga och i vissa fall även av förmåga att flytta sig mellan olika datorer.

4.2.1 Tillämpningsexempel

De för närvarande mest synliga tillämpningarna av intelligenta agenter spänner över områden som informationssökning, personliga tjänster och självständig bevakning till hantering av trafikstyrning.

Exempelvis har man utvecklat prototyper som skall kunna sköta prioritering, presentation och arkivering av en användares elektroniska post, dels enligt givna instruktioner, dels på basis av maskinell inlärning (programmet följer och drar slutsatser av användarens eget sätt att hantera sin post).

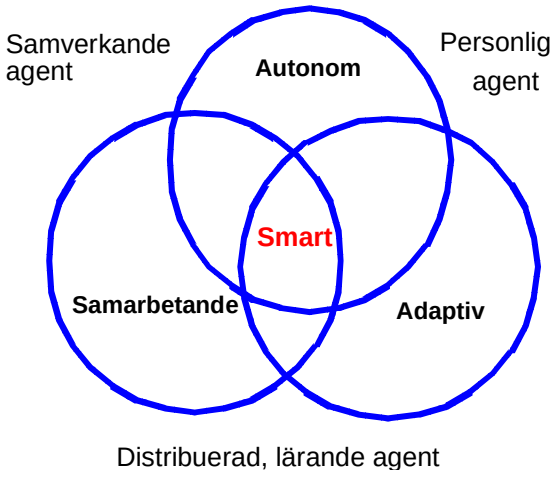
Informationssökning kan underlättas av att agenten utgående från vissa regler lär av användarens söksätt och anpassar den personliga sökprofilen. Överhuvudtaget är intelligenta agenter en metafor för sådana program som kan utföra potentiellt betungande och tidskrävande informationshanteringssysslor, uppgifter som vi människor inte passar att utföra men ändå i ökande grad kommit att behöva hantera i takt med informationsteknikens spridning. För att söka efter information på Internet finns ju s k sökmotorer, men dessa kräver aktiva åtgärder av användaren. Ett program som söker automatiskt på Nätet efter vissa kriterier (*proaktiv* sökning), och ger lämpligt utformade sammanfattningar av vad som hittats, är ett värdefullt komplement till aktiva sökningar.

För grupparbete kan agenter användas för att t ex kontinuerligt söka nya eller gemensamma intressen via inspektion av olika personers sökprofiler. Inom telekommunikation och andra nätverksbaserade tekniker har agenttekniken utnyttjats för underhåll av samband och tjänster genom kontroller, felanalys och korrigeringar med hjälp av permanenta eller rörliga agenter.

En intressant aspekt för hantering av system med lång livslängd är hur evolutionär systemutveckling kan stödjas av agentteknik som en ny paradigm för utveckling av program och programsystem. Agenttekniken kompletterar tidigare paradigmer som strukturerad programmering (från 1974) och objektorienterad programutveckling (från 1982), och dess tillskott är möjligheten att skapa interoperabilitet i ett heterogent, distribuerat system genom standardiserad agentkommunikation mellan interoperabla delsystem.

4.2.2 Vad är en agent respektive en intelligent agent?

En agent är ett program som agerar på uppdrag av en användare, som en slags representant eller ombud. Den innehåller både kod och data, och ibland även metadata såsom nuvarande programtillstånd. En agent sägs ibland ha ett beteende, och kallas därmed intelligent, och kan även ha bundna resurser, såsom pekare till andra objekt, variabelvärden och liknande. Enligt [77] baseras en agentdefinition på nyckelord som självständighet, situationsmedvetande (förmåga att reagera på signaler om sin omgivning och påverka den) och flexibilitet (ta initiativ, vara social).



Figur 5 Egenskaper hos en intelligent agent

En *intelligent (smart) agent* karakteriseras av kombinationen av:

- autonomi - självständighet och förmågan att kunna ta egna initiativ
- samarbete - via kommunikation med andra agenter eller mänskliga användare över ett nätverk, här graderar man samarbete från överenskommelser, koordination, samverkan upp till aktivt samarbete mot gemensamt mål
- adaption - förmågan att uppfånga och lära av erfarenheter

Dessa olika egenskaper lämpar sig för olika tillämpningsområden som ger agenter speciella namn som personliga agenter, samverkande (*collaborative*) och lärande agenter i en distribuerad miljö. Mobilitet är ytterligare en möjlig men ej nödvändig egenskap hos agenter.

4.2.3 Multiagentsystem

Motsatsen till den ensamma expertassistent-agenten är *multi-agent systems* (MAS) som utgår från ovanstående agentbeskrivning och vars miljö karakteriseras av att

- alla agenter har otillräcklig information eller kapacitet (t ex begränsad livslängd) för att ensamma lösa uppgiften
- det finns ingen global systemstyrning
- data är decentraliserat
- beräkningar är asynkrona

MAS intressenter utgår från behovet av robusthet, förmågan att interagera med del-system från olika perioder, t ex införliva existerande system i nya och samtidigt tillåta distribuerade resurser. Det lekfulla exemplet *RoboCup* (robotfotbollsspel) visar hur dagens teknik söker bemästrar enskilda robotars beteende medan samarbete och kommunikation väntar - de mänskliga fotbollslagens taktik och spelfördelning är inte implementerad. Exemplet illustrerar intresset, potentialen och i viss mån utvecklings-stadiet.

4.2.4 Mobila agenter

Begreppet består egentligen av två datavetenskapliga termer: mobilitet och agent. Mobilitet innebär att programkod flyttar sig mellan datorer (eller processorer), antingen autonomt eller med hjälp av ett annat program. Begreppet mobilitet

harfunnits ganska länge inom området processmigrering, som i sin tur är ett område inom distribuerade operativsystem.

En mobil agent är alltså ett mjukvaruabstraktion som kan vandra runt i ett nätverk och representera en användare för olika uppgifter. Begreppet mobil kod förekommer också, och det är ett vidare begrepp. Kodmobilitet innebär förmågan att skicka styrkommandon till andra, godtyckliga objekt i systemet. Idag krävs kontroll av digital signatur samt manuell virusavsökning. Det är långt kvar till dess koden automatiskt autentiseras och kontrolleras. Med kod menas i det här fallet vilken programkod som helst, i form av ett kommando, en funktion, eller en hel agent.

Agenter kan skapas eller raderas av användaren, operativsystemet eller en annan agent. Begreppet *kloning* används för agenter som skapar kopior av en agent och tilldelar dem begränsad livslängd och en anpassad uppgift. Begreppet används ofta för mobila agenter och kan i förlängningen skapa virus eller epidemier - med avsikt eller av misstag.

4.2.5 Standarder för agentkommunikation

En nyckelfråga inom agenttekniken är kommunikation mellan agenter eller med användare och hur standardiserad den tekniken behöver vara för att kunna överbrygga interoperabilitet mellan agentplattformar. För närvarande pågår ett standardiseringsarbete *i The Foundation for Intelligent Physical Agents* (FIPA), en icke-kommersiell organisation, skapad 1996 och registrerad i Genève (se http://www.fipa.org/). Adjektivet ”fysisk” vill markera att kontakt med mänskliga agenter ingår i någon bemärkelse. Organisationen stöds av medlemmar från telekomföretag, DARPA, SUN och forskningsinstitutioner. FIPAs syfte är att verka för utveckling av specifikationer av generisk agentteknologi som maximerar interoperabilitet inom och mellan agentbaserade tillämpningar.

4.2.6 Mäklartillämpningar

Intelligenta agenter i stor mängd kan utnyttjas för att simulera eller direkt lösa problem med ett decentraliserat angreppssätt. Exempel: agenter kan representera 1000-talet enskilda konsumenter med konversationsförmåga och egna mål, handlingslista och uppfattningar, som agerar på en marknad när en produkt annonseras. Exemplet vill visa att stora, dynamiska system kan förstås bättre via en bottom-up-ansats där det makroskopiska beteendet är ett kombinerad effekt av mikroskopiska attribut och den komplexa interaktionen mellan de mikroskopiska delarna - aktörerna. Intressanta tillämpningsområden är datanät, ekologiska nätverk, marknadssystem och ekonomiska system, underhållsfunktioner, beslutsfattande system.

4.2.7 Ledningssystem och agentteknikens potential

Agentteknik har en stor potential men många problem och olösta frågor om bl a säkerhet. Särskilt användning av mobila agenter ter sig problematiskt ur säkerhetssynpunkt. Agenttekniken är samtidigt utsatt för konkurrens från webbtekniken och dess standarder som Java och XML.

Det har hävdats att agenter krävs för att hantera storskaliga, distribuerade och heterogena tillämpningar med fokus på interaktion mellan processer och omvärlden.

Här är intelligens och mobilitet önskvärda egenskaper med förmågan till kommunikation och samarbete.

I ett ledningsstödsystem skulle agenttekniken kunna få många tillämpningar:

- underhåll och stöd för robust samband i ett dynamiskt nät
- agenttjänst med uppgifter om aktuella tjänster på nätet (”gula sidor”)
- stöd för evolutionär utveckling av nätets delsystem
- personliga tjänster för informationshantering och bevakning
- informationssökning - anpassad och proaktiv
- mäklarfunktion om resursåtkomst
- ontologi och språk/meddelandehantering - ensning, uppdatering

4.3 Databasteknik

Databastekniken har kraftigt utvecklats under 1980-talet. Dagens state-of-the art inom området utgörs av relationsdatabaserna som helt dominerar inom den administrativa databehandlingen. Objektorienterade databashanterare [78] skaffade sig en nisch under 90-talet, främst inom CAD/CAM och CASE-områdena, där de traditionella relationsdatabaserna inte alltid haft den funktionalitet och kapacitet som krävts. Relationsdatabaserna har dock fortsatt att ha en utomordentligt stark kommersiell ställning. En svaghet hos tidiga objektdatabaser var avsaknad av frågespråk.

Den idag mest framträdande tendensen inom databastekniken är att man successivt och evolutionärt övergår från den ca tjugo år gamla relationsdatabastekniken till moderna s k objektrelationella databassystem [80], som kan både lagra, utsöka och presentera data av i princip godtycklig struktur, alltså även t ex multimediadata och digitala kartdata. Till skillnad mot tidiga objektdatabaser tillhandahåller objekt-relationella databaser frågespråk lika kraftfulla som relationsdatabasernas SQL. Vidare kan man i objektrelationella databaser lagra och söka inte bara enkla tabeller (som i relationsdatabaserna) utan också användardefinierade datastrukturer. Nya generationer av objektrelationella databaser är under utveckling, som är utbyggbara (*extensible*) med nya datatyper och bearbetningsfunktioner på ett sätt som tidigare inte varit möjligt. Utbyggbarheten åstadkommes genom att användardefinierade moduler kan pluggas in i databasservern. Dessa *plug-ins* har olika namn för olika databas-tillverkare, t ex *DataBlades* (*Informix*), *Data Cartridges* (*Oracle*) och *Data Extenders* (IBM). För *SQL Server* har Microsoft utvecklat ett gränssnitt, *OLE DB*, som tillåter att man ansluter indexerade externa datakällor som virtuella relationsdatabastabeller. OLE DB-gränssnitt finns också för andra ledande databashanterare. Sådana utbyggnader kräver specialistkunnande, men behöver inte alltid vara förbehållet leverantören, utan kan också vara tillgänglig för större användarorganisationer och tillverkare av påbyggnadsprogram.

En annan utvecklingstendens är att stora ansträngningar görs för att kommersiellt introducera nya typer av databastillämpningar, som *datalager* (eng. *data ware-housing*) och *data mining*.

Datalager innebär att verksamhetskritiska data samlas i särskilda databaser, där de kan utsättas för mycket mer ingående analyser än i vanliga, operativa databaser. I ett datalager är maskin- och programvara anpassad för att stödja komplexa, analytiska

bearbetningar, exempelvis för effektiv och flexibel produktion av försäljningsstatistik uppdelad på en mängd olika kategorier [81]. Ett datalager är en stor central databas i vilken data från ett antal datakällor är materialiserade. Syftet med datalager är att understödja beslutsstöd genom att göra det möjligt för användarna att ställa avancerade frågor som analyserar data om försäljning, lagerinnehåll, trender, prognoser etc. Datakällorna är ofta andra databaser i kontinuerlig drift. Om analysfrågorna ställs direkt till dessa databaser försämras deras transaktionsprestanda. I stället materialiseras de data från datakällorna som behövs för beslutsstödet i datalagret med jämna mellanrum, dvs. asynkront[4]. Man bör således vara medveten om att data i datalager inte alltid är helt aktuella, vilket kan ge felaktiga beslut, speciellt i en krissituation.

Data mining [82], eller databrytning som ett försök till svensk översättning lyder, betecknar en klass analysmetoder med vars hjälp man kan upptäcka samband i och dra slutsatser ur t ex driftdata för att få insikter som kan leda till bättre planering. En detaljhandelskedja kan t ex tillämpa databrytningsmetoder på data som samlats in från dess kassadatorer för att finna mönster i kunders köpbeteende, mönster som kan utnyttjas vid planering av varuexponering, annonsering, butikslayout etc. Med sådana metoder förväntar man sig naturligtvis att kunna öka försäljning och lönsamhet. Databrytning utförs ofta mot datalager.

Ytterligare en tendens är att databastekniken snabbt vinner spridning till områden där den från början inte haft någon plats, exempelvis för personligt bruk och för att byggas in som komponenter i stora komplexa system. Denna utveckling förutsätter enklare och mer lättunderhållna system än som är typiskt för dagens stora databasprodukter. Exempelvis måste systemet automatiskt anpassa sig efter varierande behov och inte kräva den ständiga manuella inställning (eng. *tuning*) som stora databashanterare kräver. Spridningen av persondatorer har medfört att sådana system har etablerats på marknaden, och de bästa av dem har idag i många avseenden en funktionalitet som är jämförbar med de stora fleranvändarsystemens. Exempel på kommersiella databassystem avsedda för inbyggnad är *Sybase UltraLight*, *Interbase* (*InPrise*) och svenska *Mimer*. Ett mycket spritt databassystem för personligt bruk är *Microsoft Access*, vilket dock inte anses kunna mäta sig med fullskaliga databashanterare vad beträffar prestanda och skalbarhet.

4.3.1 Federerade databaser och medlarteknik

Federerade databaser eller *multidatabaser* är en viktig och relativt ny teknologi. Med detta menas system som kan fungera som automatiska översättare mellan en mängd geografiskt distribuerade databassystem av olika typ och med delvis olika data. På detta sätt är tanken att man skall kunna uppnå full distribuerad funktionalitet och homogenitet ur användarsynpunkt hos ett system som i själva verket består en rad olika, sinsemellan egentligen tekniskt inkompatibla databaser. Sådan teknik kommer att behöva utnyttjas för att ge ledningssystemet åtkomst till önskade aspekter (vyer) av alla de militärt väsentliga databaser som underhålls av civila myndigheter (exempelvis Vägverkets vägdatabaser).

Ledningssystem behöver kunna kombinera data från olika källor, både olika typer av databaser, simuleringssystem, och andra system. Medlartekniken är en viktig ansats

[4] Ibland ser man i pressen att också konventionella centrala databaser benämns datalager; normalt menar man dock en separat central databas med asynkront materialiserade data.

för byggande av sådana federerade system som kombinerar data från olika databaser och andra system. Medlarsystemet tillhandahåller *dataintegreringsprimitiver* som används i medlarmodellen för att *kombinera* data från inkapslare och andra medlare och presenterar de integrerade data för andra medlare och applikationer som vyer på ett homogent sätt. Dataintegreringsprimitiverna används i medlarmodellerna för att definiera s.k. *integrerade vyer* som kombinerar och transformerar data från andra medlare. I den mån universella *informationsmodeller* (ibland kallade metadatabeskrivningar eller ontologier, t ex ATCCIS, se kap 3.2.3) används för att beskriva verkligheten kan medlare användas för att beskriva hur data konverteras från lokala datarepresentationer till den universella modellen.

När samma information kan erhållas från mer än en källa händer det ofta att data från olika källor står i konflikt med varandra. Bland dataintegreringsprimitiverna måste det därför ingå *konsolideringsprimitiver* (eng. *reconciliation primitives*) för att hantera konflikter mellan data. Här är t ex kraftfulla objektorienterade frågespråk användbara för att möjliggöra formulering av generella integrerade vyer samt frågor mot dessa. Konsolideringsprimitiverna är ofta applikationsberoende, till exempel kan specialiserade datafusionsalgoritmer i vissa fall användas för konsolidering. Av det skälet behöver man kunna plugga in olika konsolideringsalgoritmer i medlarnoderna. Detta kräver *utbyggbarhet* av det system som används för att representera medlarnoderna. Frågeprocessorn i medlarkärnan måste således vara öppen så att de användardefinierade operatorer som implementerar konsolideringsalgoritmerna kan anropas. Vidare krävs ofta användardefinierade algoritmer för att optimera och transformera de vydefinitioner och frågor som är definierade i termer av de nya operatorerna.

När antalet datakällor ökar behövs skalbara medlarsystem som tillåter att nya datakällor kan läggas till utan att implementeringstiden ökar signifikant. En ansats för att åstadkomma detta är att göra medlarna *modulära* så att nya medlare kan definieras i termer av existerande medlarmoduler. Prestanda kan bli lidande av modulariteten, och det är därför viktigt att systemet kan minimera de prestandaförluster som uppstår.

För varje typ av datakälla måste en *inkapslare* (eng. *wrapper*) utvecklas, vilket är programvara för ett gränssnitt mot data från en typ av datakälla. Inkapslaren överför dessa data till ett uniformt representationsformat (datamodell) som används av andra medlare och applikationer. Inkapslarna gömmer detaljer och fysisk datarepresentation hos datakällorna och *höjer abstraktionsnivån*, samt tillåter förändringar i datakällorna utan att gränssnitten mot applikationsprogram och andra medlare behöver ändras.

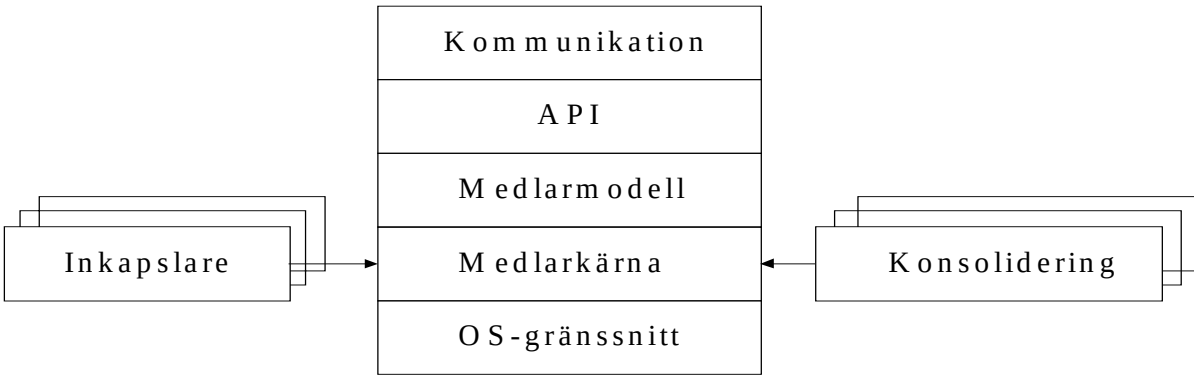
Eftersom datakällorna kan ha varierande abstraktionsnivå måste medlarna ha ett datarepresentationsformat som är tillräckligt generellt för att kunna beskriva data från de flesta typer av datakällor. Relationsdatabasrepresentation kan i detta sammanhang vara allför primitiv och en objektorienterad datarepresentation kan krävas.

Vidare krävs metoder att översätta informationsmodellerna från datakällorna till den informationsmodell som önskas i medlardatabasen. En metod att åstadkomma detta är att definiera *translatorer* och andra medlare med hjälp av ett s k *objektorienterat modellerings- och frågespråk*, som tillåter att objektorienterade *vyer* definieras i medlardatabasen som översätter datakällornas informationsmodeller.

En möjlig kandidat för objektorienterat medlarfrågespråk är nästa generation av SQL, *SQL-99*. SQL-99 är en ISO-standard som stöds av de ledande databastillverkarna och som är bakåt kompatibel med den nuvarande SQL-standarden, SQL-92. Framtida databashanterare kommer således att understödja SQL-99 som standardiserat frågespråk. SQL-99 tillåter definition av frågor och vyer över objektorienterade informationsmodeller. Eftersom SQL-99 förväntas dominera som databasfrågespråk, är det lämpligt att i möjligaste mån basera medlarsystemens frågespråk på utvidgningar av SQL-99 med primitiver för bl a inkapsling och medling. Samma frågespråk kan då användas för att ställa frågor mot medlare som mot traditionella databaser.

Många olika medlare kan finnas tillgängliga över datornäten. När antalet växer kan det bli svårt att veta vilka medlare som tillhandahåller en given tjänst. *Lokalisatorer* är medlardatabaser som innehåller *metainformation* om andra medlare (*metamedlare*), t ex vad de heter, var de finns och vilka tjänster de tillhandahåller. Generella metafrågor om medlardatabaser kan ställas till lokalisatorerna. Vidare utnyttjas lokalisatorerna av medlarsystemet självt för att optimera frågor och vyer som bearbetar global medlarinformation.

Figur 7 illustrerar uppbyggnad av en medlarnod. På den lägsta nivån finns gränssnitt mot operativsystemet. Medlarkärnan innehåller mjukvara för definition av medlarmodeller, dvs. ett modelleringsspråk (frågespråk), minneslagring, transaktionssystem etc. Medlarkärnan kan komma åt data från datakällor och andra medlarnoder via ett antal inkapslarmoduler. Inkapslare är definierade för varje typ av datakälla. För att kombinera data från flera datakällor är medlarmodellen definierad i termer av integreringsprimitiver i datamodelleringsspråket. Specialiserade konsolideringsprogram kan länkas till medlarkärnan och användas i medlarmodellen för hantering av konflikter i integrerade data. Via ett applikationsgränssnitt (API) kan integrerade data hämtas från den virtuella medlardatabasen. Kommunikationsmjukvara möjliggör att andra system och medlare kan komma åt den virtuella databasen över kommunikationsnätet.



Figur 6 Uppbyggnad av medlarnod

4.3.2 Distribuerade databassystem och replikering

Distribuerade databassystem [23] är databassystem där databasen är genomskinligt distribuerad över flera noder, och där databasbetjänten tillhandahåller data för klienterna utan att dessa behöver kunskap om detaljerna av datadistributionen. Behov av sådan distribution uppstår både av prestanda- och tillförlitlighetsskäl. Prestanda förbättras genom att dela databearbetningen över många noder och distribuerade databassystem kan maskera nodkrascher genom att internt upprätthålla redundanta noder. Sådan teknik är av stor betydelse inte minst för militära ledningssystem.

Skillnaden mellan distribuerade och federerade (heterogena) databaser är att en distribuerad databas är en logiskt central databas, som internt är implementerad över flera distribuerade datanoder. En federerad databas är å andra sidan en samling oberoende databaser. I vår alltmer heterogena Internetvärld behövs federerade databaser och medlare för att kombinera heterogena data medan distribuerade databaser behövs för att tillhandahålla snabba och tillförlitliga databasbetjänter.

Redundans och konsistens i distribuerade databaser är viktiga krav som kommer att kunna uppfyllas i hela ledningssystemet förutsatt att frågan ägnas tillräcklig uppmärksamhet vid systemkonstruktion och systemutveckling. Två grundläggande faktorer som måste beaktas vid design av distribuerade databaser är *fragmentering* och *replikering*. Fragmentering innebär att en logisk databasstruktur (t ex relationsdatabastabell) är fysiskt distribuerad över flera datanoder, och replikering innebär att samma data återfinns i flera datanoder. Ett problem för databashanteraren i det sammanhanget är att bevara konsistensen av fragmenterade och replikerade data då data uppdateras utan att klienterna behöver vara medvetna om distributionen. Andra problem innefattar att minimera exekveringen av frågor genom att parallellt hämta data från datanoderna, och att genomskinligt utnyttja redundanta noder för att maskera nodkrascher och kommunikationsavbrott.

Ett problem vid replikering är att kostnaden för att upprätthålla konsistens mellan replikerade data kan vara hög vid hög uppdateringsfrekvens. Ett antal replikeringsalgoritmer har utvecklats för detta. En enkel algoritm är att alltid utföra databasuppdateringarna i en primär datakopia (eng. *primary copy*) och sedan propagera uppdateringarna till den övriga datanoderna (eng. *backup copies*) [28]. För full datakonsistens måste man använda det s k *two-phase commit*-protokollet [22] och vänta tills alla kopior blivit konsistenta, vilket kraftigt försämrar skalbarheten genom att förorsaka stor nätverkstrafik vid uppdateringar där data är replikerade på många noder. Förfinade algoritmer har utvecklats där det räcker med att en majoritet av alla replikat uppdateras vid varje databasuppdatering [84][30] för att systemet skall kunna garantera konsistens; dock är även här skalbarheten dålig. Replikering fungerar dock tillfredställande om replikerade data inte uppdateras ofta eller om antalet replikat är litet.

Ett vanligt sätt att öka prestanda hos replikerade databaser är att kompromissa med konsistensen i replikaten och använda s k ASAP (*As Soon As Possible*) replikering, vilket innebär att systemet propagerar uppdateringar till replikatnoderna i bakgrunden efter varje uppdatering [29]. Man kan då inte garantera att replikaten alltid är konsistenta. Speciellt kan replikerade data vara föråldrade vid hög belastning, t.ex. i samband med en krissituation.

I moderna databashanteringssystem (t ex *SQL Server*, *Oracle* och *Sybase*) har man på liknande sätt möjlighet att uppdatera en huvudkopia av replikerade data, varefter systemet asynkront vidarebefordrar uppdateringarna till ”slav”-replikaten utan den uppdaterande databasoperationen väntar på att replikaten skall bli konsistenta. Replikaten uppdateras således inte direkt. Speciella problem uppstår om man vill uppdatera godtyckligt replikat eftersom det inte kan garanteras vara konsistent med resten av databasen vid uppdateringstillfället. Därför ger man bara möjlighet att uppdatera huvudkopian.

4.3.3 Materialiserade vyer och datalager

En speciell form av replikering är s k *materialiserade vyer* där databashanteraren upprätthåller en fysisk version av (materialiserar) databasvyer. Normalt är en databasvy en namngiven definition av en databasfråga, t ex VEHICLE0 som innehåller alla fordon inom ett givet område. En vy kan ses som en virtuell relationsdatabastabell definierad genom en fråga som är sökbar som andra tabeller, men som inte har något eget fysiskt innehåll lagrat i databasen. Materialiserade vyer innebär att databashanteraren kontinuerligt upprätthåller det fysiska innehållet i tabeller uttryckta som vyer, och t ex uppdaterar VEHICLE0 varje gång ett fordon lämnar eller kommer in i det givna området. Från användarens synpunkt ser en materialiserad vy ut som vilken tabell eller vy som helst, med den skillnaden att dess dataåtkomst kan vara betydligt snabbare. Liksom replikerade data kan vyers innehåll uppdateras synkront eller asynkront. I det senare fallet kan den materialiserade vyns innehåll vara mer eller mindre aktuellt.

I [32] utreds vilka krav som ställs på materialisering av vyer i mobila system där uppkopplingar kan vara osäkra och därför klienter tidvis kan vara utan kontakt med betjänterna. Ju längre tid klienten är bortkopplad desto mer inaktuella blir naturligtvis vyer materialiserade i klienten. När sedan klienten åter kopplas in kan vyernas data börja synkroniseras igen.

Materialiserade vyer är också besläktade med datalager. En viktig skillnad mellan datalager och medlarsystem är att datalager alltid materialiserar data medan medlarsystem tillhandahåller virtuella högnivådatabaser i form av ett antal mer eller mindre komplexa vyer. Medlarsystem kan således oftast tillhandahålla mer aktuella data än datalager, men ställer högre krav på effektiv frågehantering. Internt kan medlarsystem materialisera data för effektiv frågehantering och för att spåra förändringar i data, men till skillnad mot datalager är sådan materialisering genomskinlig och under kontroll av medlarsystemet. Medlarsystem kan således ge bättre underlag för beslut som berör föränderliga data.

Asynkront materialiserade vyer har föreslagits som en teknik att koppla samman heterogena databaser [85] genom att spåra förändringar i de hopkopplade databaserna [86].

4.3.4 Geografiska och spatio-temporala databaser

Försvarsmakten har sedan början av 90-talet haft en organisation för hantering av geografisk information. I en artikelserie i Kartografiska sällskapets medlemstidskrift *Kartbladet* [87] beskriver förre chefen för Operationsledningens topografiska sektion, Mats Söderberg, hur detta arbete bedrivits och vilka resultat som uppnåtts. Sammanfattningsvis kan man säga att det idag inom Försvarsmakten och närstående organi-

sationer, främst Lantmäteriet men även t ex SGU, finns kunnande, system och data inom geoinformatikområdet på hög nivå, även i en internationell jämförelse.

Detta arbete har tidigare saknat direkt organisatorisk knytning till Försvarsmaktens ledningssystemutveckling, vilket har inneburit att utvecklingsarbetet på geoinformatikområdet inte alltid varit samordnat med ledningssystemutvecklingen, och det har t ex utvecklats geografiska analys- och beslutsstödsmetoder (t ex för sökning av möjliga grupperingsplatser för artilleri) som en från ledningssystemutvecklingen fristående aktivitet. Vi ser detta som ett exempel på decentraliserad utveckling, ett antagligen nödvändigt fenomen i en stor organisation, som skapar problem endast om man saknar förmåga att integrera dess resultat evolutionärt.

Spatio-temporala databaser
Spatio-temporala databaser [88] hanterar geometrier som förändras över tiden. För att hantera sådana tidsberoende geometrier behövs en databasmodell och ett frågespråk som kan representera och operera på abstraktioner av rörliga objekt, speciellt rörliga punkter och rörliga regioner. Förutom dessa grundläggande objekttyper visar det sig att ett ganska stort antal ytterligare objekttyper behövs. Man behöver t ex datatypen linje för att representera projektionen av en rörlig punkt på ett plan. Man måste då åstadkomma:

1) ortogonalitet vid konstruktionen av typsystemet
2) genericitet och konsistens hos operationer, dvs operationer ska kunna utföras på så många typer som möjligt och uppträda konsistent
3) slutenhet och konsistens mellan struktur och operationer på icke-temporala och relaterade temporala datatyper.

Det akademiska EU-nätverket *Chorochronos* (http://www.dbnet.ece.ntua.gr/~choros/) har skapat en samarbetsstruktur för forskningen inom detta område, som tidigare saknats i Europa, och som fört europeisk akademisk forskning om spatio-temporala databaser till första ledet.

Interoperabilitet för geoinformation och OpenGIS Consortium
OpenGIS Consortium (OGC) (http://www.opengis.org) är en fristående, icke vinstdrivande organisation som skapats med syfte att specificera geografisk informationsteknik som kan möjliggöra interoperabilitet över Internet mellan olika organisationers system och databaser för geodata.

Organisationen har medlemmar av flera olika kategorier: styrande, strategisk, teknisk och associerad, med olika nivåer av inflytande. De flesta större GIS-tillverkare och databasleverantörer ingår, antingen som styrande eller tekniska medlemmar. Av särskilt intresse för Försvarsmakten kan vara att såväl NATOs arbetsgrupp för geografisk information, *Digital Geographic Information Working Group* (DGIWG) som US DoDs *Geospatial Integrated Product Team* (GIIPT) är medlemmar i organisationen (GIIPT är strategisk medlem och kan därmed bl a utnyttja OpenGIS kompetens för att finna lösningar till sina egna interoperabilitetsproblem).

Organisationen tillkom i USA år 1994, och har idag ca 200 medlemmar från de flesta industriländer. OpenGIS har också etablerat nära samarbete med ISO TC 211, den internationella standardiseringsorganisationens utskott för geoinformation. En målsättning är att småningom etablera OpenGIS som en formell internationell

teknologistandard. Konsortiet hävdar att det starka deltagandet av GIS-leverantörer i konsortiet innebär att OpenGIS-produkter snart kommer att lanseras på marknaden och då etablera OpenGIS som en de factostandard.

Att åstadkomma interoperabilitet är ett mycket svårt problem för användare av geografisk informationsteknik. Under de år som GIS-tekniken vunnit spridning har detta problem alltmer kommit i förgrunden. Olika ansträngningar har gjorts för att minska svårigheterna, inte minst genom standardisering av geodataformat. Idag finns ett stort antal olika standarder men interoperabilitet mellan olika standarder och system är fortfarande ett avlägset mål. OpenGIS har tagit ett nytt grepp på detta problem genom att konsortiet skapades av en intressentgrupp med starkt inslag av GIS- och databassystemleverantörer, alltså av företag vars affärsidé är att producera system för registrering, lagring, hantering och presentation av data, och speciellt geodata.

OpenGIS har ambitionen att möjliggöra interoperabilitet mellan databaser för geodata av alla slag: geografiska kartdata, rasterdata av olika slag inklusive t ex satellitbilder och fjärranalysdata, punkt- och vektordata, tredimensionell information som 3D-terrängdata, utbredningsfält för t ex väderdata eller föroreningsspridning, och spatio-temporala ("fyrdimensionella") data av olika karaktär och ursprung.

Man har föresatt sig att angripa detta problem genom att utforma en specifikation för ett ramverk för distribuerad åtkomst till geodata och bearbetningsresurser för geografisk information. Avsikten är att skapa en leverantörsoberoende mall för interoperabel programvara. Detta avser man att göra genom att skapa specifikationer på tre nivåer:

1. en gemensam metodik för att representera geografiska fenomen, matematiskt och begreppsmässigt (*Open Geodata Model*)

2. en gemensam modell för att implementera tjänster för åtkomst, hantering, bearbetning, representation och delgivning av geodata mellan avnämargrupper (*OpenGIS Services Model*)

3. ett ramverk av regler och programvara som bygger på användande av Open Geodata Model och OpenGIS Services Model och som är avsett att lösa såväl det tekniska som de organisatoriska interoperabilitetsproblemen

OpenGIS-specifikationen utvecklas och offentliggörs i etapper under en följd av år. Det består av en överordnad, "abstrakt" specifikation samt ett antal implementeringsspecifikationer för olika, konkurrerande distribuerade datorplattformar (*Distributed Computing Platforms*, DCP), bland dem CORBA, OLE/COM, DCE och Java.

Det är tydligt att om OpenGIS lyckas i sina föresatser kommer denna standard att få en starkt positiv effekt för lösningen av interoperabilitetsproblemet, som kan anses vara geoinformatikområdets svåraste problem.

4.3.5 Bild- och multimediadatabaser

Ett teknikområde som blir allt viktigare när datorer nu fått sådan kapacitet att de effektivt kan hantera bild- och videoinformation är databasteknik för bilder och multimedia. Sådan teknik kommer att få stor betydelse inte minst för framtida underrättelsesystem, eftersom allt fler sensortyper får förmåga till bild- och video-

registrering. Tekniken på detta område är än så länge ganska lite utvecklad, särskilt om man vill kunna utföra effektiva sökningar på stora mängder lagrade data.

Ett system som utvecklats med stöd av EUs forskningsprogram inom informationssystemområdet är *RasDaMan* [89], som är en databashanterare för rasterdata i två eller fler dimensioner, byggd som ett lager ovanpå en relationsdatabashanterare som t ex Oracle. Systemet innehåller ett frågespråk (*RasQL*) för rasterdata, baserat på AFATL, en algebra för bilddata [90].

RasDaMan har bl a använts i ett EU-projekt kallat ECHBD, *European Computerized Human Brain Database* [91], för att lagra tredimensionella s k funktionella bilder av hjärnan hos försökspersoner som utför olika förelagda uppgifter. Erfarenheter från användning av detta system visar att RasDaMan har bra förmåga att lagra och återvinna stora mängder bildinformation och att man i RasQL kan uttrycka komplexa sökvillkor över bilddata (baserade på pixelvärden m m). Däremot är kopplingen mellan RasDaMans frågespråk och värdsystemets mycket svag med denna tvålagersarkitektur. Det är därför omöjligt att i en och samma sökfråga kombinera sökning med avseende på pixelvärden och sökning på beskrivande information, metadata, som lagrats i värddatabasen. För att lösa detta problem försöker man nu med hjälp av medlarteknik i ett nytt EU-projekt (*Neurogenerator*) integrera rasterdatahanteringen och relationsdatabashanteringen i en homogen, objektorienterad datamodell. Om detta lyckas, kommer det att visa hur man med medlarteknik kan ”skarvlöst” integrera traditionella databasfunktioner och effektiv hantering av rasterdata. Detta är vad som krävs för att skapa en effektiv arkitektur för integrerad behandling av bilddata och annan information i t ex underrättelsedatabaser.

4.4 Informationssäkerhet

Datorerna ger oss egentligen enorma möjligheter vad det gäller kryptering, digitala signaturer och loggning. Det är teoretiskt fullt möjligt att spåra ett meddelandes väg genom nätverket under förutsättning att bra autentiserings- och loggningsmöjligheter finns. Jämför detta med ett vanligt pappersbrev som är synnerligen lätt att förfalska. Tekniken finns och är i huvudsak säker. För att ta till en metafor så går det att pussla ihop ett säkert system, men problemet är alla de glipor mellan pusselbitarna där informationen kan läcka ut och där angriparna tar sig in. Dessvärre måste man inse att dessa pusselbitar aldrig kommer att få perfekt passform och istället måste man ta risken och räkna med en del spill.

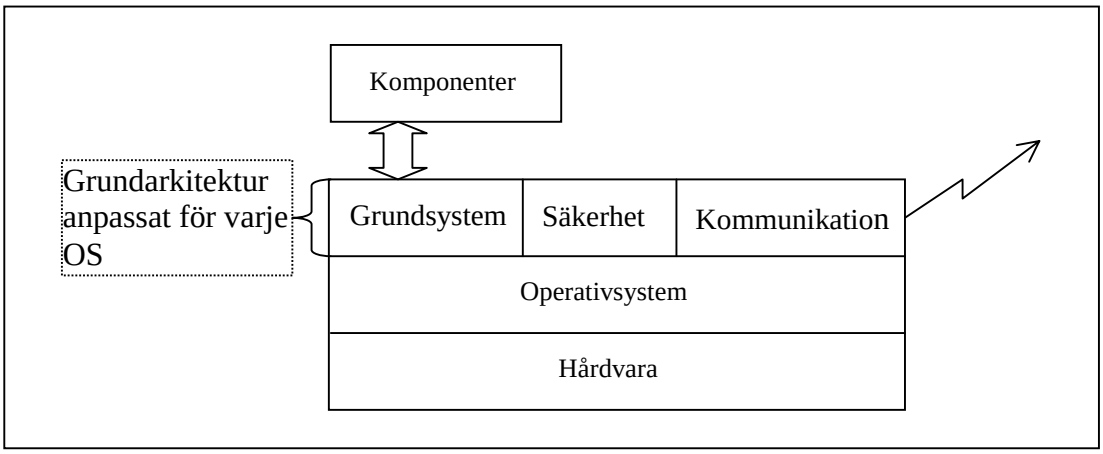
Vad är då egentligen problemet? Det består ofta i att säkerhetssystem kan bli ganska komplexa och därmed ha en tendens att fallera i tid och otid. Inget datorprogram är perfekt. Även ett mycket litet fel kan få ödesdigra konsekvenser. Enklare och mindre program ger lägre risk för fel men gör å andra sidan att vi får fler pusselbitar och därmed fler glipor. Ställ sedan detta problem mot den motsättning som finns mellan användarvänlighet och säkerhet, och det ter sig ännu svårare att lösa.

Den slutliga säkerheten i ett nätverksbaserat system är inte bara beroende av den tekniska implementationen, utan även av säkerhetspolicy, utbildning och övning samt av satsade resurser. Säkerhet är inte något man hämtar ur en handbok eller lägger till som en modul. En del s k COTS-programvaror (COTS står för *Commercial-Off-The-Shelf*, kommersiell hyllvara) för säkerhet är byggda som färdiga produkter och därför

måste man ställa upp ett antal krav på sådana programprodukter. Å andra sidan kan man hävda att COTS-produkter är mer granskade och inträngade vilket kan ge en högre säkerhet.

4.4.1 Krav och egenskaper

Basen för datasäkerhet utgörs av sekretess, integritet och tillgänglighet. För att kunna bygga denna bas behövs kryptering, autentisering, digitala signaturer, nycklar, behörighetssystem och även rollhantering. För att kontrollera alla händelser i säkerhetssystemet behövs dessutom någon form av loggning och övervakning.



Figur 7 Arkitekturskikt i ett informationssystem

Kryptering, digitala sigill och autentisering

Två typer av krypteringsmetoder finns. Den ena metoden, symmetrisk kryptering, bygger på användning av samma hemliga nyckel vid kryptering och dekryptering. Asymmetrisk kryptering möjliggör användande av två olika nycklar, där en används för kryptering och den andra för dekryptering. Asymmetriska system tillåter, såvida det inte går att härleda den ena nyckeln ur den andra, att den ena nyckeln i ett par är öppen medan den andra måste hållas hemlig.

Integriteten hos data skyddas med digitala sigill (eng. *signatures*) och felaktigheter bör upptäckas tidigt. Ett digitalt sigill bygger på användning av öppen-nyckelsystem där upphovsinstansen krypterar en unik representation av data med sin hemliga nyckel vilket gör det möjligt att med hjälp av motsvarande öppna nyckel verifiera att data är oförändrat. Om så inte är fallet måste meddelandet sändas om. Om det är databasinformation som förstörts, får man istället göra s k tillbakarullning till en tidigare, korrekt reservkopia. Orsaken till felet måste också utredas.

För att kunna styrka behörighet krävs ett *autentiseringssystem*. Ett sådant system kräver dels en metod för hantering av ett autenticitetsbevis, dels en organisation som kan utfärda sådana bevis. Både hemliga och öppna nyckelsystem kan användas. Hanteringen av nycklar, autenticitetsbevis och certifikat kan ske på olika sätt. För vissa ändamål, t ex personlig identifiering, kan smartkort vara lämpliga.

Autentisering kan man likställa med igenkänning internt i systemet och är då inte begränsat till att känna igen fysiska objekt utan också abstrakta objekt – data, program

etc. Tekniken med digitala signaturer baserat på öppna nycklar är klippt och skuren för autentisering, men liksom vad gäller rollhantering har administration av certifikaten med de digitala nycklarna visat sig vara svår. Ju mindre statisk struktur desto fler oförutsedda tillstånd kan man hamna i med åtföljande problem med autentisering.

Behörighetssystem
Ett behörighetssystems uppgift är att med hjälp av en identitetsuppgift från en användare kunna avgränsa vilka data och vilka operationer på dessa data denna användare kan utföra. Ett system som realiserar dessa avgränsningar kallas också med ett engelskt låneord för accesskontrollsystem. Ett sådant system förutsätter att de resurser som ska skyddas också kan klassificeras på ett enhetligt sätt. Utgående från klassificeringen kan sedan begränsningar i tillgänglighet göras för varje användare. I praktiken är komplikationerna med klassificering av data på ett enhetligt sätt så stora att formella system ännu inte kunnat realiseras. Det pågår ett europeiskt arbete med att ta fram *Common Criteria* för klassificering. Behörighetssystem baserade på den roll en individ har i en organisation förekommer också. Här definierar rollen behörighetsnivån och varje behörig individ tilldelas en förutbestämd roll.

Behörighetskontrollen kan behöva ligga både på operativsystemnivå och applikationsprogramnivå. Ett exempel är webläsare med inbyggda interpretatorer för Javakod som importeras från nätet. Denna importerade kod måste av säkerhetsskäl kunna hindras från att utföra operationer som ett lokalt program tillåts utföra. Inbyggd i dessa webläsare finns därför ett kontrollsystem som mer eller mindre flexibelt kan realisera en säkerhetspolicy.

Rollhantering
Syftet med rollhantering är att möjliggöra att objekt (individer, förband etc) dynamiskt kan växla mellan olika roller under iakttagande av en säkerhetspolicy. Idag finns prototyper till sådana system (jfr t ex SESAME), men det återstår bland annat automatisk klassning av data som möjliggör dynamisk rollväxling.

Säkerhetspolicy, vem som är behörig till vad, är oftast en kombination av individbehörighet och rollbehörighet. Hanteringen av roller – administration av vilka som kan inneha en viss roll, hur man växlar roll – har visat sig vara ett svårt problem. Kanske underlättas t ex rollväxling av aktiva kort, där kortet tjänar som ”token” för en viss roll. Rollhantering kommer nog ändå att vara svårhanterlig [94]. Ju mer autonom strukturen är, desto mindre beroende av rollhantering bör man bli.

Nyckelhanterings- och auktoriseringstjänster
Dessa skall tillhandahålla kryptonycklar och digitalt signerade certifikat att användas för att upprätthålla såväl sekretess som integritet. Tjänsten är grunden för att i ett distribuerat system kunna implementera en säkerhetspolicy, vem är behörig till vad. För att åstadkomma skalbarhet måste tjänsten baseras på öppen-nyckel-teknik. Det måste, av t ex sårbarhetsskäl, gå att klara sig utan tjänsten. Delar av denna tjänst finns i vissa system redan idag (t ex i DCE/Kerberos), men är då inte baserad på öppna nycklar. Tjänsten förutsätter att andra komponenter verkligen anropar den.

Loggning och spårbarhet
Alla händelser och inkomna meddelanden bör loggas för att kunna spåra intrång i systemet. Dessutom skall alla logghändelser innehålla identitet på personen eller

programmet som orsakade händelsen. Dessa loggar kan även användas för intrångsdetektering.

Informationsflöde
Ju mer information som flödar i systemet desto större risk för obehörig avlyssning och manipulation. Hög autonomitet bör kräva lägre informationsflöde. Samma torde gälla för mer sofistikerade sensorer (om man inte i stället väljer att låta sensorerna generera fler och mer detaljerade observationer).

Logiska nät
Begreppet innebär att kunna konfigurera en infrastruktur i olika logiska nät, säkerhetsmässigt skilda från varandra. Till exempel koppla samman två lokala nät över ett större osäkert nät. Denna funktion behövs t ex för att på samma infrastruktur bygga dels ett stridsledningsnät, dels ett nät för logistik och beslutsstöd. Det finns idag VPN (Virtual Private Network) som innebär kryptografisk separation av nät.

4.4.2 Standarder och produkter

Kryptering
Det finns ett stort antal krypteringalgoritmer. Många av dessa har visats sig vara mindre säkra efter noggrann analys av ett stort antal personer runt om i världen. Det är alltså viktigt att specifikationen av algoritmen är öppen för att möjliggöra analys av den av krypteringsexperter. Även om den bara är 56 bits så har DES-algoritmen visats sig vara tålig, och en ny ersättare har efterlysts. Till exempel Blowfish kan komma att ersätta DES. Den här rapporten kommer inte att gå in närmare på krypterings-algoritmer.

Krypteringsprotokoll
Det är ofta lämpligt att slå ihop autentisering och kryptering till ett protokoll, eftersom nyckelutbyten blir säkrare. Många moderna protokoll baserar sig på den principen. Exempel på användbara krypteringsprotokoll är IPsec, Secure Socket Layer (SSL), Kerberos (som egentligen mest är ett autentiseringssystem), secure shell (ssh) och PGP [95].

- *IPsec*
  Internet Protocol Security är ett krypteringsprotokoll på paketnivå som utvecklats av Internet Engineering Task Force (IETF). Samma typ av säkerhet kommer så småningom att ingå i nästa version av IP-protokollet, IPv6, men IPsec är en möjlighet för de som inte har inför IPv6 än. Med IPsec kan man autentisera de data som finns i IP-paketen och även kryptera innehållet. Protokollet används ofta för VPN.

- *SSL*
  Secure Socket Layer är ett protokoll på transportnivå som utvecklats av Netscape, men är numera standardiserat av IETF och kallas TLS. Det används främst av webläsare och är egentligen mest avsedd för dessa eftersom dess uppkopplings- och sessionshantering lämpar sig speciellt för HTTP-protokollet. Det kan dock även användas för andra TCP/IP-tjänster. SSL använder sig av digitala certifikat för att autentisera användaren eller webservern, och krypterar datatrafiken med valbar krypteringsalgoritm. Detta är ett bra protokoll som skyddar väl mot ”man in the middle”-attacker.

- *ssh*
  Secure shell är en ersättare för Unix-verktygen rlogin, rsh, rcp. Ssh möjliggör säkra X11 sessioner och kan upprätta säkra tunnlar mellan datorer. Det använder sig av öppna och privata nycklar för att autentisera både användaren och servern, har en stark kryptering och komprimering av data. Det är nödvändigt med bra och säkra fjärrstyrningsverktyg.

- *PGP*
  Pretty Good Privacy är ett program för kryptering av elektronisk post och filer. PGP använder öppen-nyckel-kryptering och har en mer informell icke-hierarkisk struktur för signering av nycklar.

Några andra protokoll som inte tas upp här är S/MIME, PPTP, SET, PEM, SKIP. De är antingen knutna till ett företag eller ej särskilt spridda.

Säkerhetsarkitekturer
Det finns både forskningsprojekt och färdiga produkter inom området säkerhetsarkitekturer. De mest intressanta ur evolutionär och distribuerad synvinkel är CRISIS och Legion. Båda innehåller mycket av tankarna i konstruktionsprinciperna i tidigare kapitel (se avsnitt 3.2.8). Den förmodligen säkraste produkten är SESAME och i viss mån Kerberos.

- *CRISIS – Wide Area Security Architecture*
  CRISIS är ett autentiserings och behörighetssystem som har ett antal mål: redundans för att undvika enpunktsattacker, cachning för att öka prestandan i ett långsamt distribuerat nätverk, rollhantering och ett finkornigt behörighetssystem, och fullständig loggning av alla beslut i behörighetssystemet. Projektet som huvudsakligen är akademiskt är sponsrat av bland annat DARPA och vissa datortillverkare [96].

- *Legion – a new model of security for distributed systems*
  Legion är en ny säkerhetsmodell som är tänkt att hantera mycket stora distribuerade system. Det skall köras ovanpå existerande system och ny funktionalitet skall kunna läggas till dynamiskt. Legion är ett objektorienterat system och är tänkt att vara en evolutionär modell som så småningom skall komma fram till en balans mellan höga krav på säkerhet och användbarhet [97].

- *SDSI – simple distributed security infrastructure*
  SDSI är en enkel öppen-nyckel-infrastruktur för att hantera behörigheter och säkerhetspolicies. Den använder sig länkade lokala namnrymder istället för en hierarkisk global namnrymd. SDSI borde vara lämpligt att använda i större system eftersom man inte blir beroende av en global namnserver [98].

- *CDSA - Common Data Security Architecture*
  CDSA är en säkerhetsarkitektur som är tänkt som en gemensam standard för hela industrin. Den är speciellt avsedd för distribuerade applikationer i PC-miljö. CDSA har ett antal lager där varje komponent i lagret kan bytas ut eller nya läggas till. Det är en öppen modell och nya teknologier inkluderas efterhand [99].

- *Kerberos*
  Kerberos är ett säkerhetssystem där användaren endast behöver logga in en gång på nätverket och får därmed tillgång till de tjänster som finns. Användarna identifierar sig genom att ge ett lösenord och får då en ”biljett” som de kan använda för att kontakta andra tjänster i nätverket. Kerberos krypterar all data-trafik men skickar aldrig lösenord över nätet. Den senaste Kerberos version 5 kommer att ingå i Windows 2000. Nackdelen med Kerberos är att det kräver ständig tillgång till en central nyckelserver och att det är starkt beroende av korrekt tidsynkronisering. Det är därför tveksamt om Kerberos är tillräckligt robust och feltolerant för militära tillämpningar.

- *SESAME – a network authentication service*
  SESAME är en förkortning för ”Secure European System for Applications in a Multi-vendor Environment”. Det är ett autenticeringssystem baserat på samma ideer som Kerberos med dess single login och kryptering av kommunikationen. SESAME använder sig av öppen-nyckel kryptografi, har rollbaserat behörighets-system, och har separata autentiserings- och krypteringsnycklar. SESAME liknar till stor del Kerberos och har därmed samma nackdelar [95].

- *DCE*
  DCE står för Distributed Computing Environment och är framtaget av Open System Foundation som är en grupp av dataföretagen IBM, HP och DEC. Det är ett distribuerat system som är tänkt som ett alternativ till Unix men som skall köras ovanpå andra operativsystem som VMS, Windows och även Unix. DCE innehåller ett antal tjänster, såsom trådar, Remote Procedure Call, tid, namn-kataloger, säkerhet och ett distribuerat filsystem. Remote Procedure Call (RPC) innebär att en klient kan anropa en procedur i en server. Det är delvis baserat på Kerberos.

Krypterande filsystem
Något som bör ingå i en säkerhetsarkitektur är att alla filer bör vara krypterade. En nackdel med detta är att data kan förloras när nyckeln försvinner eller att trasiga filer blir svåra att reparera. Dessutom kan krypterande filsystem blir långsamma vid data-bashantering. Det finns även krypterande distribuerade filsystem som *Secure NFS*. Detta system autentiserar med nycklar och delar ut filerna krypterat.

4.4.3 Databassäkerhet
Det är viktigt att databasens innehåll skyddas från otillåtna uppdateringar. Exempel på sådana uppdateringar kan vara fel i indata, programfel, inkonsistenta uppdateringar, samt illasinnade användare. För detta ändamål tillhandahåller databashanterarna säkerhets- och dataintegritetsfaciliteter. Frågor kring *databassäkerhet* har fått förnyad uppmärksamhet och det har bl a getts ut en ny bok som beskriver området [83].

Moderna relationsdatabashanterare tillhandahåller egna system för autentisering, kryptering, privilegietilldelning, etc, vilka är oberoende av liknande operativsystem-funktioner. Man vill knyta säkerhetsinformationen till databasens innehåll snarare än till filsystem etc. För detta tillhandahåller SQL-92 ett antal primitiver för att tilldela *rättigheter* till (grupper av) användare och program. För varje kombination av *subjekt* (användare etc) och *objekt* (tabeller, vyer, etc) i databasen specificeras de tillåtna *åtkomsträttigheterna* (läsning, skrivning, ändring). Databasadministratören har

fullständiga rättigheter. Med hjälp av speciella SQL-92-kommandon (GRANT och REVOKE) kan man delegera (frånta) rättigheter till (från) andra användare. Man kan givetvis inte delegera mer rättigheter än man själv har.

Eftersom åtkomsträttigheter kan knytas till generella databasvyer kan ett subjekts rättigheter bero av databasens dynamiska innehåll. Man kan t ex föreskriva att ett subjekt bara får se vyn VEHICLE0, som innehåller de fordon som befinner sig inom ett givet område.

För t ex militära tillämpningar har man föreslagit s k flernivåsäkerhetssystem (*multilevel security*) där varje subjekt och objekt också har en säkerhetsnivå (*top secret, secret, confidential, unclassified*). Ett subjekt kan bara komma åt de objekt som inte har högre säkerhetsnivå än subjektet själv.

Moderna databashanterare har också ett flertal primitiver för att bevara integriteten hos data. Integritetsbegränsningarna kontrolleras närhelst databasen uppdateras. Bl a understöds följande av SQL-92:

- *domänbegränsningar* är användardefinerade restriktioner på enskilda datavärden (t.ex. positiva, icke-tomma, < 5, etc.)
- *nyckelbegränsningar* garanterar att tabellnycklar är unika
- *referensbegränsningar* upprätthåller samband mellan dataelement från olika tabeller. Nyckeln i en tabell kan vara refererad som en s k främmande nyckel i en annan tabell, t ex kan en tabell FORDON ha en främmande nyckel REGID som identifierar det område där fordonet befinner sig. Ett krav är då att bara sådana REGID som finns som nycklar i tabellen REGIONS får förekomma
- *allmänna begränsningar* förhindrar uppdateringar som bryter mot allmänna policybestämmelser (t ex att inget område får innehålla mer än 20 fordon). Det bör observeras att upprätthållande av allmänna begränsningar kan göra databasuppdateringar mycket långsamma.

4.5 Kommunikationsstandarder och mellanskiktsprogram

För kommunikation mellan delsystem används alltmer grundläggande *infrastrukturer för distribuerade system*. I ledningsomgivningen utnyttjas dessa grundläggande protokoll för kommunikation mellan medlarsystem och med applikationer och datakällor. Exempel på standardprotokoll för kommunikation mellan olika system är TCP/IP, CORBA och DCOM (Microsoft) och RTI (HLA). För WWW och Intranet har vidare det textbaserade HTTP-protokollet som körs ovanpå TCP/IP blivit utomordentligt viktigt. Protokollet RPC (*Remote Procedure Calls*) är ett standardtillägg till TCP/IP som lanserades för Unix i början av 80-talet. Med RPC kan man anropa procedurer i främmande processer/program över ett lokalt nätverk, bestående av Unixdatorer och numera även Microsoft Windowssystem.

Dessa ”tillståndslösa” protokoll har gjort det betydligt enklare att bygga distribuerade system. För access till relationsdatabaser har *ODBC* (ett SQL-gränssnitt) och *JDBC* (Java-variant av ODBC) blivit populära de facto-standarder. Vissa av dessa protokoll bör användas för att kommunicera mellan delsystem. För att definiera gränssnitt mot olika typer av system behöver gränssnitt mot applikationsprogram (APIn) definieras för olika programmeringsspråk. APIn behövs således gentemot traditionella programmeringsspråk som C++, C och Java.

Det nya programmeringsspråket Java har blivit populärt genom att göra det enklare att utveckla plattformsoberoende applikationsprogram i inter- och intranätmiljöer. Eftersom Java kan bli något av ett standardprogrammeringsspråk för nätapplikationer är det viktigt att Java-baserade applikationer kan integreras i medlarmiljön. I Java-världen har man infört egna standarder och terminologier för t ex kommunikation mellan distribuerade system (RMI), och med relationsdatabaser (JDBC) [102].

En viktig men plattformsspecifik interoperabilitetsstandard är Microsofts COM (*Component Object Model*), som är grunden för OLE och ActiveX. Det pågår arbete med att definiera överbryggningar mellan COM och CORBA, tillsvidare kallad *COM/CORBA interoperability*. En beskrivning av dessa standarder finns i [101].

4.5.1 CORBA

Ett viktigt exempel på gränssnittsstandardisering är *CORBA (Common Object Request Broker Architecture*) [19][20]. CORBA-standarden är en typisk kollektiv industristandard som, liksom UML (se avsnitt 3.2.1), skapats och vidareutvecklas av *Object Management Group*, OMG.

I CORBA-konceptet ingår ett "mäklarsystem", en s k *object request broker*, ORB, som sköter kommunikationen mellan tillämpningsprogram eller snarare tillämpnings-objekt, dvs programdelar som inkapslar såväl data som funktioner. I CORBA talar man om *objekttjänster*, dvs tjänster i form av objekt som tillhandahålls av system-miljön, och *gemensamma resurser*, dvs objekt från gemensamma bibliotek. Tjänster som tillhandahålls av ett objekt beskrivs i ett gränssnittsdefinitionsspråk (IDL) och lagras i ett gränssnittsförråd. ORB-funktionen skapar med utgångspunkt från IDL-specifikationerna bryggor mellan klienterna och ORBen, respektive mellan ORBen och betjäningsystemen.

CORBA specificerar en standard som för att kunna tillämpas kräver tillgång till en omfattande programvara, en ORB. Sådana programvaror har utvecklats av flera leverantörer, och i början av CORBAs utveckling var dessa sinsemellan inkompatibla. I CORBA 2.0 ingår ett *CORBA Internet Inter-ORB Protocol* (IIOP), med vars hjälp godtyckliga ORBar kan kopplas samman över Internet.

CORBA tillhandahåller en rad olika slags tjänster:

- *statiska och dynamiska metodanrop*: tjänster över nätet kan anropas antingen enligt en helt fördefinierad anropssignatur eller med sen bindning (late binding), vilket innebär att exakt vilken tjänst som används inte bestäms förrän den anropas
- *språkneutrala anropssignaturer*: anropssignaturer som av objektprogrammeraren definieras i IDL och av CORBA-systemet översätts till en språkberoende bindning i en prekompileringsprocess
- *självbeskrivande servergränssnitt*: varje servergränssnitt som är känt av systemet har en metadatabeskrivning i ett s k Interface Repository. Klienter som tillämpar sen bindning använder dessa metadata vid för att få information om hur en tjänst skall anropas
- *lokal/global transparens*: anrop lokalt inom en dator har samma syntax som ett anrop över Internet, så att programmen inte behöver ändras när man vill flytta en tjänst till en lokal till en global server

- *inbyggd säkerhet och transaktionshantering*: ORBen lägger in information i sina meddelanden som sköter säkerhetsskydd och transaktionshantering mellan datorer och ORB-system
- *polymorfa meddelanden*: en ORB förmedlar ett anrop till en metod i ett värdobjekt, vilket betyder att det som verkligen utförs bestäms av såväl metodanropet som värdobjektet, ett avancerat exempel på objektorienteringsparadigmens polymorfism
- *samexistens med arvssystem*: CORBAs separering av specifikation och implementering kan utnyttjas för att kapsla in redan existerande program, vilket gör att CORBA kan användas för att underlätta evolutionär systemutveckling.

CORBA 2.0, som är den senaste versionen av standarden, innehåller bl a mekanismer för samverkan mellan DCE (se kap 4.4.2) och CORBA (*DCE ESIOP*), som innefattar stöd för Kerberos säkerhetsfunktioner, distribuerad tid, autentiserade RPC-anrop m m.

IBM har en CORBA-baserad objektmäklare [92], kallad *Component Broker*, som ingår i produkten *Websphere Enterprise Edition*. Denna produkt innehåller också ett komponentbibliotek baserat på Java, *Enterprise Java Beans* (EJB). Miljön stödjer distribuerade tillämpningar som innehåller CORBA- och Javakomponenter, och har också fullt stöd för Microsoft COM-objekt under Windows NT. För integration av Javakomponenter används RMI, *Remote Method Invocation*, som är en metod för att länka Javaprogram till CORBA, som baseras på CORBA IIOP.

4.6 Operativsystem

Ledningsstödsystemet kommer att använda sig av ett underliggande operativsystem. Grundkravet på detta är att det kan hantera TCP/IP, har ett filsystem och har en hög säkerhetsnivå i enlighet med konstruktionsprinciperna i 3.2.8. Operativsystemet måste alltså tillhandahålla både tillräcklig säkerhet och kommunikationsmöjligheter. Ledningssystemet bör vara så plattformsoberoende som möjligt.

Ur säkerhetssynvinkel kan det vara intressant att notera att det är en viss typ av operativsystem som huvudsakligen drabbas av virus, nämligen Windowssystem och då främst Windows 95/98. I till exempel Linux, som är en modern Unix-variant, existerar knappast virus. Mycket resurser läggs ner på att utveckla antivirusprogram, men ingen verkar reflektera över att andra operativsystem inte alls har samma problem. Det är rimligt att anta att det måste finnas en fundamental skillnad i säkerhetsarkitekturens konstruktion mellan dessa system. Även i ordbehandlingsprogrammet Word har man problem med *makrovirus*.

Skillnaden är att man försökt göra Windows okomplicerat och enkelt att använda och därmed fått göra avsteg från många av säkerhetsprinciperna i avsnitt 3.2.8. Man separerar inte användarnas rättigheter och använder sig inte av flera säkerhetsnivåer.

Ur säkerhetssynvinkel är det en fördel att ha flera olika operativsystem i nätverket och till och med olika versioner av dem. Om man har exakt samma operativsystem överallt kan ett virus eller någon annan attack lamslå alla datorer på hela nätverket. Genom att ha flera olika system minskar risken att alla drabbas av samma programfel eller attack. Heterogenitet är alltså inte bara en nackdel.

Windows NT
Försvarets handböcker rekommenderar detta operativsystem, troligen på grund av dess popularitet och användarvänlighet. Det finns också många programprodukter (COTS) till detta operativsystem och det är internt homogent i den meningen att alla produkter från Microsoft är väl anpassade till det, däremot sällan några andra. Fördelarna med Windows NT är att det har ett bra filsystem som återhämtar sig väl efter en krasch och det har ett bättre behörighetssystem än Windows 95/98. En nackdel med Windows NT är att källkoden och protokollen är hemliga och därmed svåra att inspektera. Kommunikationsprotokollen i Windows NT har visat sig ha allvarliga säkerhetsbrister.

Windows 2000
År 2000 kommer NT att ersättas av en ny version av Windows. Det är svårt att säga något om dess kvalitéer då stora delar av operativsystemet är nyskrivna och dessutom är systemet inte släppt och i drift än. Det ser dock ut som om Microsoft arbetat ganska mycket på att lägga in bra säkerhetsfunktionalitet i operativsystemet. I Windows 2000 kommer det att finnas ett integrerat Kerberossystem, ett Ipsec-protokoll för VPN, ett krypteringssystem baserat på öppen-nyckel-infrastruktur samt SSL/TLS för säkra klient-serveruppkopplingar.

Linux
Linux är en Unix-variant som från början var ett amatörprojekt och som nu vuxit till ett fullvärdigt operativsystem. Det baserar sig på öppen källkod (*Open Source*) och använder sig mycket av en evolutionär utvecklingsmodell med många snabba prototyper som sätts i drift direkt av orädda användare. Trots, eller kanske på grund av, detta är det ett mycket stabilt system med utmärkta prestanda. Linux är till stor del modul- eller paketbaserat och man installerar de delar som behövs, till exempel Kerberos för säkerhet. Linux nackdel är den amatörism som fortfarande till viss del hänger kvar. Även om Linux sällan kraschar, oftast bara vid strömavbrott, så kan det få problem med filsystemet vid återstart [99]. Tyvärr är Linux i grundinstallation inte särskilt säkert från nätattacker. Det krävs att någon gör nödvändiga fixar, och då det i Linux inte är några problem att göra djupa ändringar i operativsystemet, kan man förvandla Linux till en säker bastion. För den vanlige okunnige användaren är dock Linux inte det ur säkerhetssynpunkt bästa valet.

Övriga Unix och Windows varianter
Andra ur säkerhetssynpunkt bra operativsystem är Unix-varianterna Solaris, HPUX, AIX m fl. MacOS har visat sig vara tåligt ur defensiv synvinkel eftersom det har en annorlunda intern uppbyggnad än det marknadsdominerande Windows. Windows 95/98 rekommenderas ej då det är avsett för privatpersoner.

4.7 Utvecklingsmiljöer

För att kunna utveckla system behöver man en *utvecklingsmiljö*. En sådan omfattar programspråk med editor, kompilator, versionshanterare och klassbibliotek för en rad olika ändamål: generella datalogiska och numeriska metoder och algoritmer, användargränssnitt, grafik, nätverkskommunikation, databaskopplingar (vi utgår ifrån att vi bestämt oss för att arbeta inom objektorienteringsparadigmen). Till detta kommer mycket snart behov av ett modelleringsverktyg (se 3.2.1 och 3.2.2), som helst ska vara anpassat till den övriga miljön, så att man kan röra sig mellan specifikation och programkod i båda riktningar utan att information går förlorad.

En av huvudidéerna bakom evolutionär utveckling är att utnyttja existerande investeringar i programsystem så länge som möjligt, även efter det att systemet som helhet inte längre uppfyller ställda krav. Vid systemmodifiering vill man kunna utgå från existerande programkod och även återanvända delar av den vid nyutveckling när så är möjligt. Detta innebär att inte bara programspråk och komponentbibliotek, utan i så stor utsträckning som möjligt också utvecklingsmiljön, bör vara så stabila över tiden som marknads- och teknikutvecklingen tillåter. Av detta följer också att beställaren måste kräva nyttjanderätt till källkoden när man köper utveckling av ett programsystem från ett utvecklingsföretag. Nyttjanderätt innebär att beställaren har tillgång till systemets källkod och rätt att i egen regi vidareutveckla den för sitt eget behov, däremot inte nödvändigtvis rätt att sälja den vidare, något som kräver äganderätt.

Rationella produktionssystem kan därför bara byggas på standarder: industristandarder som CORBA, C++, Java, OpenGIS, halvofficiella från t ex den professionella organisationen IEEE, och officiella standarder från ISO, ANSI, CEN, SIS. Den officiella standardiseringen kommer vanligen åtskilliga år efter etableringen av en de facto-standard, och det händer ofta att den standardiserade produktklassen då redan är på utgående.

Så länge man bara vill få fram en första prototyp för att pröva idéer och koncept kan man använda vilka verktyg som helst, bara man (och inte minst ens chefer!) är medveten om att ens val ofta kan innebära total omskrivning av programmet om det visar sig så användbart att man vill ha med den i en driftsversion. Men evolutionär utveckling bygger alltså på att man gör ett strategiskt val av utvecklingsmiljö och sedan håller fast vid denna under mycket lång tid, och man kan fråga sig om den snabbrörliga informationstekniken alls tillåter en att göra sådana långsiktiga val.

Hur är det då med produkter som Windows eller Visual Basic, två mycket populära system som utvecklats av marknadsledande Microsoft? Eftersom Windows idag är det i särklass mest populära operativsystemet för personligt bruk, vore det quixotiskt att hävda att det inte bör användas (se dock avsnitt 4.6). Visual Basic och Object Pascal är däremot exempel på populära språk med anknutna utvecklingsmiljöer, som av ovannämnda skäl inte lämpar sig för evolutionär systemutveckling.

En av de bästa böcker som skrivits om objektorientering (*OO*) och programsystemutveckling är Bertrand Meyers Object-Oriented Software Construction, 2nd edition [103]. Meyer gör i denna monumentala bok på mer än 1200 sidor en exposé över objektorienterad programsystemutveckling och diskuterar programvarukvalitet, programspråk, utvecklingsmiljöer och framför allt, konstruktion av program med hjälp av objektorientering. Meyer är inte bara författare till denna och andra böcker om objektorientering, utan också upphovsman till ett objektorienterat språk, *Eiffel*. Han är också en entreprenör som skapat företaget *ISE* för utveckling och marknadsföring av en komplett, plattformsoberoende utvecklingsmiljö för detta språk. För den som i likhet med ett par av författarna tidigt blev bekanta med och förtjusta i denna miljö, är det tråkigt om än inte helt oväntat att tvingas konstatera att den idag för en kommersiellt tynande tillvaro.

Samma fenomen fast i större skala har drabbat de flesta andra stora språkprojekt: IBMs PL/I redan på 70-talet, US DoDs Ada, Smalltalk osv. Ada utsågs till och med av

såväl US DoD som det svenska försvaret till enda godkända programspråk i början av 80-talet, en bestämmelse som aldrig blev fullt genomförd och idag enbart finns kvar på papperet. Ada konstruerades ursprungligen som ett realtidsspråk, och som sådant har det fortfarande få konkurrenter. Ada blev färdigt alldeles före genombrottet för objektorienterade metoder och språk, och när man i mitten på 90-talet lanserade det objektorienterade Ada 95 för att om möjligt ta igen den förlorade terrängen var det för sent.

Idag är det tre generella programspråk som kan anses så väl etablerade och stabila att de kan ligga till grund för evolutionär utveckling, nämligen C++, Java och C, det sistnämnda dock ett lågnivåspråk som inte bör användas annat än för speciella ändamål, som programmering av kommunikationsprotokoll, drivare för i/u-enheter och andra operativsystemnära funktioner. Java är ett modernare och betydligt bättre språk än C++, som dessutom har mycket bra egenskaper för utveckling av nätverks-baserade system, och det stöds av inte bara Sun Microsystems, som utvecklade det, utan också av allt fler företag som utvecklar nät-verksbaserade system. Ändå är det förstås omöjligt att veta hur länge t ex Java kommer att attrahera programmerare och programvaruutvecklare, det avgörande kriteriet för ett programspråks användbarhet.

4.7.1 Java

Java är ett objektorienterat programspråk som liknar C och C++ men är enklare och ger robustare program. Språket har utvecklats av Sun Microsystems, till en början (1990) som ett inslag i ett forskningsprojekt som studerade nya metoder för nätverksbaserade system. Robusthet och säkerhet har varit huvudkrav på detta språket, eftersom det har till syfte att möjliggöra "nerladdning" av främmande kod till användarens egen dator just då koden behövs för en viss tillämpning. Sådan främmande kod är ett potentiellt dödligt hot mot ett informationssystem, och det är inte utan skäl som de flesta säkerhetskänsliga organisationer valt att vara försiktiga med introduktion av Internettjänster och hemsidor. Emellertid är det en avgörande skillnad mellan filer som används som ren data för t ex presentation, och filer som innehåller data som är avsedda att exekveras, dvs program. Så länge det egna systemet inte innehåller några "trojanska hästar" som har lagts dit just för att senare kunna släppa in fientlig kod förklädd till harmlösa datafiler, så kan främmande data inte skada ett system.

Vad Java vill erbjuda är en mekanism som gör det möjligt att utan risk ladda in och exekvera främmande kod i våra datorer. Java har också andra intressanta och potentiellt värdefulla egenskaper. Den viktigaste är att Javaprogram är tänkta att vara helt portabla ("*write once, run anywhere*"), ett ideal som tyvärr visat sig svårt att uppnå. Ett program behöver inte framställas i olika versioner för olika datorer eller operativsystem utan garanteras körbart i varje dator som har tillgång till en virtuell Javamaskin. Javamaskinen är en exekveringsomgivning som översätter det nerladdade Javaprogrammet till den lokala maskinens språk och därefter utför dessa instruktioner.

Ett Javaprogram representeras av en speciell teckenkod, som bildar den datamängd som skickas över nätverket vid nerladdningen. Den virtuella Javamaskinen låter programmet genomgå en analysfas, där det kontrolleras att teckenkoden följer Javaspråkets säkerhetsregler. Om så inte är fallet kommer koden inte att utföras. Men för formellt korrekta Javaprogram garanterar språkets exekveringsmodell och

Javamaskinens logik att programmet är säkert och inte kan orsaka vare sig sammanbrott eller intrång i användarens datorsystem.

Den enda slags fel- eller sabotagemöjlighet som man upptäckt och inte kunnat rätta till är att ett Javaprogram kan låsa resurser och därefter gå i en oändlig loop [104]. Detta innebär att nätverksbläddraren, *browsern*, där Javamaskinen ingår, måste startas om, men någon annan skada uppstår inte. Denna typ av fel eller sabotage är inte begränsade till Java eller nerladdade program, utan kan åstadkommas genom bombardering av en nod med epostmeddelanden och liknande. För att skydda sig mot sådana angrepp kan man använda en effektiv brandvägg, som stoppar eller prioriterar ner obehöriga eller utomstående avsändares meddelanden, så snabbt att behörig trafik inte allvarligt bromsas upp. Man kan också använda speciell teknik för resursbegränsning där man explicit tilldelar cpu-cykler och minne åt programmet. Vi vet dock inte om det är möjligt att med sådana metoder skydda sig mot mycket intensiva meddelandebombardemang, med bibehållande av tillräcklig kommunikationskapacitet för behörig trafik.

5. Enkelt exempel

Exemplet utgår från rörliga arméstaber i fält som nyttjar ett gemensamt informationssystem Fenix för att hålla den centralt fastställda, gemensamma lägesbilden ständigt aktuell hos alla staber. I exemplet förenklar vi frågan till att olika fordon håller reda på sin position och gör den synlig för alla övriga.

Texten visar ett schematiskt exempel på följande:

- en arkitektur för ett utvidgningsbart potentiellt heterogent nätverksbaserat aktivt informationssystem
- hur ny teknik adderas till befintliga system, dvs hur olika byggstenar fogas samman i en evolutionär utvecklingsanda
- teknik med aktiva medlare anpassad för arméns informationssystem Fenix
- möjligheten att enkelt addera och radera övervakare (ett aktivt system påpekar själv förändringar för intressenten, väntar ej på dennes förfrågan)
- illustration av teknik för databasreplikering
- tekniskt underlag för semiautomattolkade meddelanden

.

Vi tänker oss att utgå från arméns informationssystem Fenix (i en version från sommaren 1999) installerat i portabla arbetsstationer med e-postanslutningar och placerade i fordon med GPS-sändare. Varje fordon bestämmer regelbundet sin position m h a GPS, lagrar den i sin lokala Fenixdatabas och skickar informationen vid behov, t ex vid tydlig förflyttning sedan senaste position, vidare per e-brev till ett centralt stabsfordon. Detta tar emot meddelanden och skapar en aktuell lägesbild i realtid som sedan kan vidarebefordras till övriga fordon. Texten i e-breven skickas i ett strukturerat textformat sådant att texten maskinellt kan tolkas och lagras i andra Fenixdatabaser (databasreplikering). Om aktiva medlare används i systemet kommer även lägesbilden att påverkas omgående där så önskas, t ex i andra fordon, om inte så sker detta först då operatören frågar databasen efter ändringar.

Syftet är att koppla samman befintlig programvara från de tre systemen Amos, Fenix och GeoPres, där

- Amos-systemet hanterar aktiv medling av databaser
- Fenix är en tidig prototyp till lednings- och informationssystem för armén
- GeoPres är en militär standard för geografisk datahantering och presentation.

Målet är i första hand att förbättra databasreplikeringen så att lägesbilden distribueras snabbt och är lika för alla i realtid och i andra hand att underlätta möjligheter till automatisk avtolkning av textmeddelanden, som sedan skickas vidare till andra Fenix-installationerna i ett strukturerat format.

Systemkomponenter och anpassningsbehov

Fenix är ett NT-baserat informationssystem där rapporter kommer in via TODAKOMs möjligheter som e-post, telefon eller fax. Rapporterna omtolkas och klassificeras manuellt och lagras i Fenix databaser. Detta underlag, kompletterat med kartadata (GeoPres-data), egna styrkors kända positioner och fiendens troliga position och förbandstyp, presenteras på grafiska skärmar som en lägesbild som sedan godkänns och distribueras ut till förbanden och dess lokala Fenix-databaser.

Förändringar i Fenix:

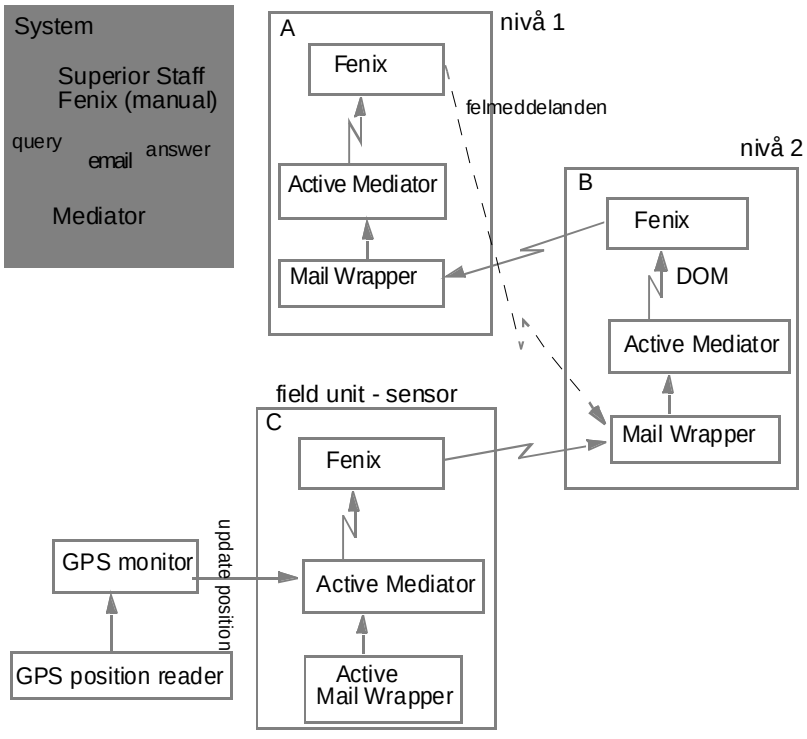
- addera en aktiv medlare och ett e-posthölje i kommunikationen mellan olika Fenix- stationer och nivåer (1, 2 ,3)
- addera en automatisk texttolk som omvandlar strukturerad text och lagrar informationen anpassad i databasen

En medlardatabas presenterar högnivåvyer av kombinationer av datakällorna. Den ger dataåtkomst på en hög abstraktionsnivå lämpad för användare och hanterar även egna metadata som ej lagras i underliggande databaser. Amos databashanterare lämpar sig dels för inbäddad implementation i andra system, dels för integration och koordination av data över ett nätverk. Följande kompletteringsbehov för Amos ställs av vår tillämpning:

- addera e-postsändare och e-posthölje
- addera övervakare
- Amos II utvecklas från en passiv till en aktiv medlare

E-posthanteringen utvecklas så att meddelanden passerar ett e-posthölje (wrapper) som låter all informationen passera medlaren innan informationen skickas till Fenix.

Följande bild visar hur komponenterna samverkar i ett struktur med tre nivåer där en GPS-läsare skickar sin position till en lokal Fenixinstallation.



Figur 8 Komponentdiagram

Den innehåller även ett aktivt e-posthölje för att ta emot annan information. Den av lokala Fenix strukturerade informationen sänds vidare till nivå 2, t ex fordonsdatabasen, vars e-posthölje skickar informationen till sin Fenix-databas via sin aktiva medlare som därefter omedelbart kan visa informationen på fordonsskärmen.

Proceduren upprepas från nivå 2 till nivå 1, t ex ett stabsfordon. Härifrån kan felmeddelanden skickas åter till avsändaren på nivå 2. Funktionen DOM i nivå 2 i figuren avser en tänkt tolkare av strukturerad information. DOM är ett av verktygen att tillämpningsanpassa hanteringen av XML-baserade meddelanden (Document Object Model).

6. Referenser

[1] I Alm, E A Eriksson, T Lindgren, S Odar, S Sköld & P Svensson: *Ny systemarkitektur för evolutionär utveckling av lednings- och informationssystem för försvaret. En förstudierapport från projektet Systemarkitektur*. FOA-R--98-00673-505--SE, Avdelningen för Försvarsanalys, Försvarets forskningsanstalt, Stockholm, 1998.

[2] J Blaker: The Owens Legacy. The Former Vice Chairman of the Joint Chiefs Laid the Groundwork for a Revolution. *Armed Forces Journal International*, July 1996, pp. 20-22.

[3] K Pallin & J Lagerlöf: Doktrinen är ledningens själ. *FOA-tidningen*, vol. 37, nr 4, oktober 1999, 8-13 (specialnummer om ledning).

[4] K Mellberg & G Schyborger: Behovet av och möjligheter med teknik för framtida underrättelsetjänst, sid 17-42. Årsberättelse 1995 i avd IV. *KKrVA Handlingar och Tidskrift* 5/95.

[5] L Jönsson, G Neider, J Schubert & P Svensson*: Informationsfusion i den taktiska underrättelseprocessen. En informell introduktion till nyckelteknologin för informationsöverlägsenhet.* FOA-R-98-00902-505--SE, Avdelningen för Ledningssystemteknik, Försvarets forskningsanstalt, Linköping, December 1998.

[6] H S Marsh: From the Fog of War to Information Overload: A New Challenge for Command and Control, MITRE Corporation, Boston, MA.

[7] E Englund, J Grönkvist, A Hansson, J Nilsson, I Söderquist & J Zander*: Tactical Radio Access Networks – A Concept Study; Report 1*. FOA-R-98-00940-504--SE, Avdelningen för Ledningssystemteknik, Försvarets forskningsanstalt, Linköping, December 1998.

[8] *Försvarsmaktens systemutveckling. En granskning av projekten ORION, ATLE-IS, LI FV och LIM*. Riksrevisionsverket rapport RRV 1997:49.

[9] H Enquist: Informationssystemarkitektur och militär ledning. Föredrag vid MILINF 96.

[10] S C Chamberlain: Model-Based Battle Command: A Paradigm Whose Time Has Come. *Proc. 1995 Symposium on C2 Research and Technology*, National Defense University, Washington DC, 1995.

[11] I Alm U Backlund, M Bengtsson, C Carlsson, (red), L Carlsson, A-L Christiansen, P Grahn, E Jungert, A Lauberts, D Letalick & C-L Westerlund*: Multisensorteknik och datafusion - ett FOA-perspektiv*, FOA-R--98-00710-408--SE, Avdelningen för Sensorteknik, Försvarets forskningsanstalt, Linköping, 1998.

[12] S E Johnson & M C Libicki (eds.): *Dominant Battlespace Knowledge*, Center for Advanced Concepts and Technology, National Defense University, Washington, DC, 1996.

[13] Department of Defense Modeling and Simulation (M&S) Master Plan, October 1995, http://www.dmso.mil/documents/.

[14] G E Moore: Cramming More Components Onto Integrated Circuits, *Electronics*, Vol. 38, No. 8, April 1965, pp. 114-117.

[15] M Bohr: Silicon Trends and Limits for Advanced Microprocessors, *Commun. ACM* 41(3), 80-87, 1998.

[16] J O Coplien: Reevaluating the Architectural Metaphor: Toward Piecemeal Growth, *IEEE Software* Sept/Oct 1999, pp. 40-44.

[17] J O Coplien: *Software Patterns*. SIGS Books, New York 1996.

[18] P Wegner: Interoperability. *ACM Computing Surveys*, 28 (1), March 1996, pp. 285-287.

[19] R Orfali, D Harkey & J Edwards: *Instant CORBA*. Wiley 1997.

[20] R Orfali & D Harkey: *Client/Server Programming with JAVA and CORBA*, John Wiley & Sons, 1997.

[21] R Elmasri & S B Navathe: *Fundamentals of Database Systems*, Benjamin/Cummings, 1994.

[22] A Silberschatz, H Korth & S Sudarshan: *Database System Concepts*, McGraw-Hill, 1996.
[23] M T Özsu & P Valduriez: *Principles of Distributed Database Systems*, Prentice-Hall, 1991.
[24] O A Bukhres & A K Elmagarmid*: Object-Oriented Multidatabase Systems - A Solution for Advanced Applications*, Prentice-Hall, 1996.
[25] W Kim: *Modern Database Systems*, Addison-Wesley, 1995.
[26] G Wiederhold: Mediators in the Architecture of Future Information Systems, *IEEE Computer*, March 1992.
[27] G Wiederhold: Mediation in information systems. *ACM Computing Surveys*, 27 (2), June 1995.
[28] P A Alsberg & J D Gray: A principle for resilient sharing of distributed resources, *Proc. Int'l Conf. On Software Engineering*, pp 627-644, 1976.
[29] C Pu & A Leff: Replica control in distributed systems: An asynchronous approach, *Proc. 1991 SIGMOD Conf.*, 1991.
[30] R H Thomas: A majority consensus approach to concurrency control for multiple copy databases, *ACM Transactions on Database Systems*, Vol. 4, No. 2, June 1979.
[31] A A Helal, A A Heddaya & B B Bhargava: *Replication Techniques in Distributed Systems*, Kluwer, 1996.
[32] O Wolfson, P Sistla, S Dao, K Narayanan & R Raj: View maintenance in mobile computing, *ACM SIGMOD Record*, Vol. 24, No. 4, Dec. 1995.
[33] R Wieringa: A Survey of Structured and Object-Oriented Software Specification Methods and Techniques. *ACM Computing Surveys*, Vol. 30, No. 4, December 1998, pp. 459-527.
[34] C Kobryn: UML 2001: A Standardization Odyssey. *CACM*, Vol. 42, No. 10, October 1999, pp. 29-37.
[35] A Mili, S Yacoub, E Addy & H Mili: Toward an Engineering Discipline of Software Reuse, *IEEE Software* Sept/Oct 1999, pp. 22-31.
[36] H Mili, F Mili & A Mili: Reusing Software: Issues and Research Directions. *IEEE Trans. Software Eng*., Vol. 21, No. 6, June 1995, pp. 528-562.
[37] M Josefsson & Ö Oskarsson: *Programvarukomponenter i praktiken – att köpa tid och prestera mer*. Sveriges Verkstadsindustrier 1999.
[38] R L Glass: Reuse: What's Wrong with this Picture? *IEEE Software*, March/April 1998, pp. 57-59.
[39] P P Chen, J Akoka, H Kangassalu & B Thalheim*: Conceptual Modelling – Current Issues and Future Directions*. Springer LNCS No. 1565, 1999.
[40] The Land C2 Information Exchange Data Model (ADATP-5), Draft 1.0, 1 October 1999.
[41] S Krusche & A Tolk: A SHADE Approach for Coupling C4I-Systems and Simulation Systems. *Fall 1999 Simulation Interoperability Workshop* http://www.sisostds.org/.
[42] A Tolk: Requirements for Simulation Systems when being used as Decision Support Systems. *Fall 1999 Simulation Interoperability Workshop* http://www.sisostds.org/.
[43] A Tolk: Using ATCCIS as an Information Layer to Couple CGF Federates and Closed Combat Simulations. *Fall 1999 Simulation Interoperability Workshop* http://www.sisostds.org/.
[44] *FIPS* (*Federal Information Processing Standards*) Publication 184, December 1993.
[45] E W Dijkstra: The structure of the THE multiprogramming system. *Comm. of the ACM*, 11 (5), pp. 341-346, 1968.
[46] D L Parnas, P C Clements & D M Weiss: The modular structure of complex systems, *IEEE Trans. on Software Eng*., SE-11, pp. 259-266, March 1985.
[47] D Garlan & D E Perry: Introduction to the Special Issue on Software Architecture. *IEEE Trans. on Software Eng*., 21 (4), pp. 259-266, April 1995.
[48] M Shaw & D Garlan: *Software Architecture: Perspectives on an Emerging Discipline.* Prentice-Hall 1996.
[49] P Donohoe: *Software Architecture*. Kluwer Academic, Boston 1999.

[50] G Wiederhold (guest ed.): Special Issue on Intelligent Integration of Information, *Journal of Intelligent Information Systems*, Kluwer, June 1996.
[51] A Tomasic, L Raschid & P Valduriez: Scaling Access to Heterogeneous Data Sources with DISCO, *IEEE Transactions on Data and Knowledge Engineering*, Vol. 10, No. 5, pp 808-823, 1998.
[52] H Garcia-Molina & al: The TSIMMIS Approach to Mediation: Data Models and Languages, *Journal of Intelligent Information Systems (JIIS)*, Vol. 8, No. 2, Kluwer, 1997.
[53] P M D Gray & al: KRAFT: Knowledge Fusion from Distributed Databases and Knowledge Bases, Database and Expert System Applications, *Proc. Intl. Workshop on Database and Expert Systems Applications, DEXA'97 (IEEE),* Toulouse, France, 1997.
[54] P C Lockemann & al: The Network as a Global Database: Challenges of Interoperability, Proactivity, Interactiveness, Legacy, *23rd Intl. Conf. on Very Large Databases, VLDB'97*, Athens, Greece, Aug. 1997.
[55] L Haas, D Kossmann, E Wimmers & J Yang: Optimizing Queries across Diverse Data Sources. *23rd VLDB Conf.*, pp. 276-285, Athens, Greece, 1997.
[56] G Fahl & T Risch: Query processing over object views of relational data, *The VLDB Journal*, Springer, No. 6, Nov. 1996.
[57] V Josifovski & T Risch: Functional Query Optimization over Object-Oriented Views for Data Integration, *Journal of Intelligent Information Systems (JIIS)*, Vol. 12, No. 2-3, 1999.
[58] V Josifovski & T Risch: Integrating Heterogeneous Overlapping Databases through Object-Oriented Transformations, *25th Intl. Conf. On Very Large Databases, VLDB'99*, Edinburgh, Scotland, September 1999.
[59] V Josifovski, T Katchaounov & T Risch: Optimizing queries in distributed and composable mediators, *4th Conference on Cooperative Information Systems, CoopIS'99*, Edinburgh, Scotland, September 1999.
[60] L Liu & C Pu: An Adaptive Object-Oriented Approach to Integration and Access of Heterogeneous Information Sources, *Journal of Distributed and Parallel Databases*, Vol. 5, No. 2, Kluwer, April 1997.
[61] W Du & M Shan: *Query Processing in Pegasus*, in [24].
[62] A Motro & P Smets (red.): *Uncertainty Management in Information Systems. From Needs to Solutions.* Kluwer Academic 1997.
[63] B Bouchon-Meunier, R R Yager & L A Zadeh (red.): *Information, Uncertainty and Fusion.* Kluwer Academic 1999.
[64] J Saltzer & M Schroeder: The Protection of Information in Computer Systems, *Proceedings of the IEEE*, Vol.63, No.9, November 1975.
[65] B Riggs: Streaming Media Goes Mainstream. *InformationWeek* No. 764, Dec. 6, 1999, p. 69.
[66] N I Durlach & A S Mavor: Virtual Reality: Scientific and Technological Challenges. 1995.
[67] Databilder direkt på ögat. *Ny Teknik-Teknisk Tidskrift* 1997:19, p. 35.
[68] S Mann: Wearable Computing: A First Step Toward Personal Imaging. *IEEE Computer*, February 1997, pp. 25-32.
[69] J Dahmann, M Salisbury, P Barry & C Turrell: HLA and Beyond: Interoperability Challenges. *Fall 1999 Simulation Interoperability Workshop* http://www.sisostds.org/
[70] M D Myjak, D Clark & T Lake: RTI Interoperability Study Group Final Report. *Fall 1999 Simulation Interoperability Workshop* http://www.sisostds.org/.
[71] D H Timian, M R Hieb, J Glass & M J Staver: Using Standard Command, Control, Communications, Computers, and Intelligence (C4I) Components to Interface to Simulations. *Spring 1999 Simulation Interoperability Workshop* http://www.sisostds.org/.

[72] M R Hieb & M J Staver: The Army's Approach to Modeling and Simulation Standards for C4I Interfaces. *Fall 1998 Simulation Interoperability Workshop* http://www.sisostds.org/.
[73] R L Ressler, M R Hieb & W Sudnikovich: M&S/C4ISR Conceptual Reference Model. *Fall 1999 Simulation Interoperability Workshop* http://www.sisostds.org/.
[74] M R Hieb & J Blalock: Data Alignment between Army C4I Databases and Army Simulations. *Spring 1999 Simulation Interoperability Workshop* http://www.sisostds.org/.
[75] W Sudnikovich, J D Roberts & J Lacetera: Implementation of a Prototype C4I FOM. *Spring 1999 Simulation Interoperability Workshop* http://www.sisostds.org/.
[76] W Sudnikovich & J D Roberts: Implementation of a Prototype C4I FOM: Continued Progress. *Fall 1999 Simulation Interoperability Workshop* http://www.sisostds.org/.
[77] N Jennings, K Sycara & M Wooldridge: A Roadmap on Agent Research and Development, *Autonomous Agents and Multi-Agent Systems*, Vol. 1, pp. 7-38 (1998), Kluwer Academic Publishers.
[78] R G G Cattell: *Object Data Management - Object-Oriented and Extended Relational Database Systems*, Addison-Wesley, 1994.
[79] D W Embley: *Object Database Development - Concepts and Principles*, Addison-Wesley, 1998.
[80] M Stonebraker: *Object-Relational Databases*, Morgan Kaufmann, 1996.
[81] H S Singh: *Data Warehousing, Concepts, Technologies, Implementations, and Management*, Prentice Hall, 1997.
[82] U M Fayyad, G Piatetsky-Shapiro, P Smyth & R Uthurusamy (eds.): *Advances in Knowledge Discovery and Data Mining*, AAAI Press, 1996.
[83] S Castano, M Fugini, G Martella & P Samarati: *Database Security*, ACM Press/Addison-Wesley, 1995.
[84] H Garcia-Molina & D Barbara: How to assign votes in a distributed system, *J. ACM*, Vol. 32, No. 4, Oct. 1985.
[85] M Rusinkiewicz, A Sheth & K Karabatis: Specifying interdatabase dependencies in a multidatabase environment, *IEEE Computer*, Vol. 24, No. 12, Dec. 1991.
[86] T Risch: Monitoring database objects, *Proc. 15th Int'l Conf. On Very Large Databases, VLDB'89*, 1989.
[87] M Söderberg: Geografisk information och dess användning inom Försvarsmakten. Del 1 – 4. *Kartbladet* 1998:3, 1998:4, 1999:1, 1999:2.
[88] R H Gueting, M H Böhlen, M Erwig, C S Jensen, N A Lorentzos, M Schneider, M Vazirgiannis: A Foundation for Representing and Querying Moving Objects. Technical report 238, Fachbereich Informatik, FernUniversität Hagen, September 1998.
[89] P Baumann & al.: The RasDaMan approach to Multidimensional Database Management. *Proc. 12th Annual Symposium on Applied Computing*, San José 1997.
[90] G Ritter, J Wilson & J Davidson: Image Algebra: An Overview. Computer Vision, Graphics, and Image Processing, Vol. 49, No. 1, pp. 297-331, 1990.
[91] J Fredriksson, P Roland & P Svensson: Rationale and Design of the European Computerized Human Brain Database System, *Proc. 11th Intl. Conf. on Scientific and Statistical Database Management*, IEEE Computer Society Press, 1999.
[92] White Paper: Middleware. *Datateknik 3.0*, nr. 7, november 1999.
[93] A Caglayan & C Harrison: *Agent Sourcebook*, Wiley 1997.
[94] R S Sandhu: Role Based Access Control Models, *IEEE Computer*, Feb. 1996.
[95] P Ashley & M Vandenwauver: *Practical Intranet Security (Kerberos/SESAME)*, Kluwer Academic Publishers 1999.
[96] E Belani, A Vahdat, T Anderson & M Dahlin: The CRISIS Wide Area Security Architecture, *7th USENIX Security Workshop*, 1998.
[97] W Wulf, C Wang & D Kienzle: Legion - A New Model of Security for Distributed Systems, *IEEE Computer* May 1999.

[98] R Rivest & B Lampson: SDSI - A simple distributed security infrastructure, *Proceedings of the USENIX Symposium 1996*, April 1996. http://theory.lcs.mit.edu/~cis/sdsi.html
[99] *Common Data Security Architecture*, http://www.intel.com/ial/security/
[100] O Sigurdsson: Dålig säkerhet i Linuxpaketet. *Datateknik*, nr. 14, 17 september 1998, p 33.
[101] R Orfali, D Harkey & J Edwards: *The Essential Distributed Objects Survival Guide*. John Wiley 1996.
[102] G Reese: Database Programming with JDBC and Java, O'Reilly, 1997.
[103] B Meyer: *Object-Oriented Software Construction*, 2nd edition, Prentice Hall 1997.
[104] D Lindahl*: Mobil kod och datasäkerhet: Säkerhetsmekanismer i Javas virtuella maskin.* FOA-R-99-01166-503--SE, Avdelningen för Ledningssystemteknik, Försvarets forskningsanstalt, Linköping, juni 1999.

7. Ordlista

Ordlistan innehåller termer, som förekommit i militära eller akademiska sammanhang under 1990-talet och som nyttjas i eller har en koppling till områden kring system-arkitektur. Engelska termer beskrivs i ett separat avsnitt.

7.1 Indelning

Ordlistan är uppdelad i följande separata avsnitt:

- Termer om systemarkitektur (inom parentes anges källa t ex NE)
- Engelska termer med översättning
- Akronymer - engelska och svenska
- Källor

Inom parentes anges bl a källa (se sista kapitlet i bilagan)

7.2 Termer om systemarkitektur

Term	**Förklaring**
adaption	datafusion nivå 4, se resursadaption
aggregera	gruppera, sammanfatta (motsats: disaggregera)
agent	datorprogram som agerar på uppdrag av en användare, som en slags representant eller ombud. Egenskaper: självständighet (innehåller program, data, och programtillstånd), situationsmedvetande (förmåga att reagera på och påverka signaler i sin omgivning), självstyrande (kan ta initiativ)
autentisering	kontroll av uppgiven identitet, t ex vid inloggning, vid kommunikation mellan två system eller vid utväxling av meddelande mellan användare
applikation	tillämpning
brandvägg	hinder mot oönskad kommunikation mellan olika datornät, främst mot intrång
datafusion	en kategori informationsbehandlingsprocesser där osäker, ofullständig och motstridande information från olika källor slås samman för att ge en mer komplett och mindre osäker översikt över ett aktuellt problem baserat på hela informationsmaterialet. Datafusion spänner över ett stort område, från multisensordatafusion (nivå 1), över situationsanalys (nivå 2), hotanalys (nivå 3) och adaption (nivå 4) med koppling till beslutsstöd och AI-tilllämpningar.
datalager	databas där data lagras på ett sätt som underlättar de avancerade sökningar och sammanställningar som behövs för beslutsstöd och analys inom en organisation
dataoberoende	möjlighet att ändra data utan att ändra tillämpningar
demonstrator	ett program eller system som syftar till att demonstrera en metod eller teknik
disaggregera	sönderdela, återupplösa - återskapa delar till en sammanfattning, motsats till aggregera
distribuerad databas	en databas vars innehåll är genomskinligt distribuerad över flera datorer i ett nätverk. Exempel på andra distribuerade tillämpningar: ledning, grupp-utvecklingsmiljöer, system, utvecklingsprocess, funktionalitet, simulatorer, sensorer
doktrin	lära eller lärosats avsedd att styra det praktiska handlandet (NE)
domänteknologi	ordet domän används för att benämna eller gruppera en uppsättning system eller funktionsområden inom ett system som har likartad funktionalitet. Domän-teknologi är grunden för "produktlinjeansatser" för programvaruutveckling och handlar om möjligheten att utföra underhåll på samt att förstå, använda och återanvända system
evolutionär sys-	(här) visionen av en metodik och teknik som skulle göra det möjligt att bygga ut

temutveckling och förvaltning	och modifiera ledningssystemets funktioner successivt och i takt med förändrade behov och tillgång till nya tekniska lösningar.
federat	samverkande simuleringsprogram i ett system (HLA)
GIS-Centrum	Lantmäteriverkets centrum för GIS-frågor i samarbete med försvaret
heterogen	oenhetlighet - innehållande olika beståndsdelar (NE)
hotanalys	datafusion nivå 3: analyserar risker och möjligheter för de egna styrkorna att möta motståndaren på ett effektivt sätt
informationsfusion	motsvarar datafusionens nivå 2-4: situationsanalys, hotanalys och adaption
informationskrig-föring	användning av information i krigföring och användning av informationsbaserade vapen
informationskvalitet	kvalitetsnivåer hos: grunddata i en databas, beräknade data (modellkvalitet), genuint osäker information
informationsmodell	beskrivning av data (metadata)
informationssystem	system som omfattar programvara, datalager, dataflöden, maskinell utrustning, verksamhetsprocesser
informationssäkerhet	Problem: sekretess, integritet, behörighetskontroll (datasäkerhet), tillförlitlighet, nyckelhantering; (inom försvaret:) mobilitet, heterogena system, fysiska skydd, degraderande system
intelligent agent	ett autonomt datorprogram med uppgift till underlätta arbete över ett distribuerat, heterogent datorbaserat system
interoperabilitet	förmågan till samarbete mellan programsystem som kan vara skrivna i olika språk, följa skilda gränssnittsstandarder och exekvera på olika plattformar i form av operativsystem och maskinvara
konnektivitet	fullständig sammanfogning (NE)
konsistens	motsägelsefri
ledning	ledning i svensk militär bemärkelse innefattar processer som målidentifiering, uppgiftstilldelning och resursallokering, och utgör det område där förståelse av stridsrummet förvandlas till uppgifter och uppdrag som är avsedda att påverka, kontrollera och dominera detta stridsrum.
ledningsdoktrin	ett i förväg fastställt regelverk, i regel nationellt
ledningskrigföring	behandlar hur man kan angripa och slå ut motståndarens ledningsfunktioner på alla plan, respektive skydda sina egna
ledningssystem	militärt syfte: leda försvarets förband i krig, kris och fred enligt ledningsdoktrinen - teknisk aspekt: ge stöd främst till de operativa och taktiska ledningsnivåerna, dvs chefer och staber på divisions- eller åtminstone brigadnivå och däröver (lägre nivå: stridsledningssystem)
lokalisator	medlardatabas som innehåller metainformation om andra medlare (metamedlare)
lägesbild	den gemensamma lägesbilden är ett centralt mål i en militär organisation
meddelandebaserad och kommuni-kations-intensiv	rådande paradigm för ledningsstödsystem enligt Sam Chamberlain
medlare	skapar gemensamma vyer av heterogena databaser och åstadkommer dataoberoende, så att strukturen hos databaser kan ändras utan att högre systemnivåer ("applikationer") behöver ändras (medlarteknik)
medlarsystem	möjliggöra automatisk översättning mellan olika databasers begreppsvärldar eller scheman
metadata	metainformation - data om data
Metria	produktionsbolag inom Lantmäteriverket, Sverige
modellbaserad ledning	en önskad paradigm för konstruktion av ledningsstödsystem

Moore's lag	IT-utvecklingens fördubblingstid om c:a 18 månader (efter 1970)
multisensordatafu-sion	datafusion nivå 1: att uppnå större robusthet, precision och överblick från sensorsystemen genom att i realtid kombinera information från flera sensorer, ofta baserade på olika slags sensorteknik, i ett integrerat system.
ontologi	gemensam begreppsapparat som nyttjas att förstå och beskriva ett ämnesområde. (NE:) läran om begrepp eller kategorier som man behöver anta för att kunna ge en sammanhängande beskrivning och förklaring av (någon del av) verkligheten
operation (militär)	en serie samordnade förflyttningar och strider med utnyttjande av samverkande mark-, sjö- och luftstridskrafter, understödda av civila resurser för att uppnå ett operationsmål eller operativt syfte (benämndes förr fälttåg) (källa: NE)
operationell arki-tektur	verksamhetsarkitektur - enligt US DoD: 'Operational Architecture is the total aggregation of missions, functions, tasks, information requirements, and business rules'
operativ doktrin	uttalar avsikter hur operationer avses föras i krig (i termer av operationsfrihet, luft- och sjöherravälde, gräns-, kust- och luftinvasion, perifer-, skal-, djup- och ytförsvar)
paketförmedling	teknik att överföra information i digitala datapaket över nätet
öppen nyckel	för kryptering - en nyckel i ett asymmetriskt krypteringssystem motsvaras av en privat nyckel. En nyckel är helt enkelt en konstant sträng av data
reaktiv planering	fortlöpande omplanering i realtid av mer eller mindre komplexa uppgifter
redundans	(informationstekniskt) meddelanden som kan tas bort utan att informationen går förlorad
replikering	(här:) samma data återfinns i flera databaser dvs filosofi om och teknik för uppdatering av information i ett distribuerat databassystem
resursadaption	datafusion nivå 4, (om)styrning av resurser för att nå ett givet mål
schema för databaser	namn och andra karakteristika hos deras datatermer och de semantiska sambanden datatermerna emellan
signaturbibliotek	databaser över målsignaturer, beskriver hur olika måltyper i detalj ter sig eller "ser ut" för ett visst slags sensor
situationsanalys	datafusion nivå 2: identifiera den situation som har orsakat observerade data och händelser
skalbarhet	förmågan hos den inre strukturen att klara av kraftigt ökad systembelastning
spatio-temporal	rumslig (geografisk) och tidsmässig t ex modell eller databas
strategi	läran om användningen av militära eller andra maktmedel för att i kamp med en motståndare nå politiska mål, såväl krigsmål som andra mål som fred, neutralitet, reglerade maktförhållanden. Termen motsvarar sammansättningar som total, stor eller högre strategi till skillnad från militär strategi ~ operativ strategi. **Medel**: militära resurser + produktionsapparat för dessa (källa: NE)
Symmetria	utvecklingsmetodik vid Metria GIS-Centrum
systemarkitektur	här: formulera en övergripande arkitektur för framtida datorstödda ledningssystem . Enligt USDoD: fysisk implementation av den operationella arkitekturen baserad på den tekniska, samt med layout för och relationer mellan system och dess kommunikation
taktik	läran om användning av militära förband för att i strid nå en lokal framgång. Grundas på vapenverkan, rörelse och skydd som kombineras i termen stridsteknik. Principer: överraskning, kraftsamling, handlingsfrihet. Förutsättningar: underrättelsetjänst och bra ledning. Uppdragstaktik: chefen detaljstyr ej medel och metoder (i motsats till kommandostyrd taktik) (NE)
teknisk arkitektur	byggstenar på vilka system baseras
topologisk struktur	i nätverkssammanhang: relationer mellan noder i terrmerna en-till-en, en-till-många, många-till-många

ÖB Grundsyn Ledning	styrdokument inom försvarsmakten, fastställt 1994

7.3 Engelska termer

Term	Förklaring
browser	bläddrarprogram, ofta i webbsammanhang
client –server	klient - betjäning
constraint satisfaction	problem: givet ett antal logiska villkor som involverar ett antal variabler, bestäm värden till variablerna så att samtliga villkor är samtidigt satisfierade
data mediation functions	medlingsfunktionalitet - funktioner för integrering och transformering mellan datarepresentationer
data mining	databrytning, en klass av analysmetoder med vars hjälp man kan upptäcka samband i och dra slutsatser ur t ex driftdata för att få insikter som kan leda till bättre planering
data warehousing	datalager (se datalager)
design by contract	kontraktsbaserad utveckling
domain engineering	domänteknologi
enhanced reality	förstärkt/utökad verklighet inom VR-teknik
fail-safe defaults	utgå från avsaknad av behörighet
feature	skapelse, *featurism* ~ jakten efter nya finesser i system och applikationer
fuzzy logic	oskarp logik
graceful degradation	gradvis degradering
intelligence	underrättelse(tjänst)
management of uncertainty	akademiskt forskningsområde som utvecklar teori och metodik för att hantera osäker information
mediator technology	medlarteknik
middleware	mellanskiktsprogram
monitor	övervakarprogram
pattern	mönster, bl a inom praktisk programvaruarkitektur
plug in	insticks-, mekanism för hopkoppling mellan klient och betjäning
reconciliation primitives	konsolideringsprimitiver för dataintegrering vid datakonflikter
Revolution in Military Affairs	uttryck myntat av William Owens som syftar till paradigmbyte inom USAs försvarsmakt (RMA)
signatures	digitala sigill
socket	vägguttag, mekanism för hopkoppling mellan klient och betjäning
system of systems	supersystem innehållande flera olika, fristående system
telegeoprocessing	sammanlänkning av telekomsystem och geografiska informationssystem
translator	översättarprogram
use-case driven	användningsfallsbaserad utveckling (se UML)
wearable computers	personlig bärbar datorutrustning med låg effektförbrukning
wrapper	inpackare t ex runt datakällor

7.4 Akronymer

Akronymer är initialförkortningar bestående av initialer från flera ord, t ex US för United States.

Akronymer	Beteckning och sammanhang
ABCS	US Army Battle Command System
ACM	Association for Computing Machinery - bl a tidskriftsutgivare
ActiveX	Microsoft's teknik att förenkla informationsdelning mellan distribuerade tillämpningar, baserad på OLE
AMOS	Active Mediator Object System
API	Application Interface - applikationsgränssnitt
ASAP	As Soon As Possible – metod för databasreplikering
ATCCIS	NATO Allied Tactical Command and Control Infrastructure System
ATCCS	Army Tactical Command and Control System (USA)
ATLE	Arméstridskrafternas Taktiska Ledningssystem (försvarsmaktsprojekt)
C2CDM	Command and Control Core Data Model
C2W	Command and Control Warfare
C4I	Command, Control, Communications, Computer applications, and Intelligence
CAD/CAM	Computer-Aided Design/Computer-Aided Manufacturing
CAMA	Computer-Aided Messaging Architecture, svensk försvarsmaktsstandard för meddelandebaserad kommunikation
CASE	Computer Aided Software Engineering
CGF	Computer Generated Forces
CMMS	Conceptual Models of Mission Space, inom DIS
COA(A)	Course of Action (Analysis)
COM	Component Object Model, Microsoft-standard för API (jfr DCOM)
CORBA	Common Object Request Broker Architecture
COTS	Commercial Off The Shelf - ofta om civilt utvecklad programvara
CSCW	Computer Supported Cooperative Work
CTS	Celsius Tech Systems, företag inom svensk försvarsindustri
DARPA	Defense Advanced Research Projects Agency - driver US DoD forskningsprogram
Datal	Arméns taktiska ledningssystem, svenskt projekt under 1980-talet
DBMS	DataBase Management System - databashanterare
DCOM	Distributed Component Object Model från Microsoft
DCE	Distributed Computing Environment från OSF
DDB	Dynamic DataBase - ett DARPA-projekt
DES	Data Exchange Schema
DII COE	Defense Information Infrastructure Common Operating Environment
DiMuNDS 2000	Distributed Multi-National Defense Simulation, NATO-projekt
DIS	Distribuerad interaktiv simulering - teknik (tidigare även standard som nu ersätts av HLA)
DMSO	US Defense Modeling and Simulation Office (HLA-utveckling)
DoD	United States Department of Defense
DOM	Document Object Model

DGIWG	Digital Geographic Information Working Group
EJB	Enterprise Java Beans
ER	Enhanced Reality - förstärkt/utökad verklighet inom VR-teknik
Fenix	tidig prototyp för svenska arméns informationssystem
FIPA	The Foundation for Intelligent Physical Agent - standardiseringsorgan
FMS	Fleet Management Systems - kombinerade informations- och kommunikationssystem med vars hjälp en mängd rörliga enheter kan kontrolleras och övervakas
FOA	Försvarets Forskningsanstalt, Sverige
FOM	Federation Object Model (HLA)
FoU	Forskning och utveckling
GH	Generic Hub Data Model inom ATCCIS
GIIPT	US DoDs Geospatial Integrated Product Team
GIS	geografiska informationssystem
GPS	Global Positioning Systems
HLA	High Level Architecture - standardiserat kommunikationsgränssnitt för informationsutbyte mellan simuleringsmodeller
HTTP	Hypertext Transfer Protocol
HKV	Försvarsmaktens högkvarter
IDL	Interface Definition Language, gränssnittsdefinitionsspråk för CORBA
IEEE	Institute of Electrical and Electronics Engineers, Inc.
IIOP	Internet Inter-ORB Protocol, ingår i CORBA
IP	Internet Protocol
ISIS	holländskt arméledningssystem
IT	informationsteknik
IW	Information Warfare - informationskrigföring
JCDB	Joint Common Data Base
JDBC	Java DataBase Connectivity - standard för kommunikation mellan Java och relationsdatabaser
JTA	Joint Technical Architecture, mall för US ledningsstödssystem
JTDM	Joint Common Data Base Transformation Data Model, logisk datamodell
KTH	Kungliga Tekniska Högskolan, Stockholm
LEO	första svenska systemet för operativ ledning från 1970-talet
LUST	LedningsUtvecklingsSTudie, startad inom HKV 1999 som konsekvens av SAIC-studien
MAS	Multi-Agent System
MASIF	Mobile Agent System Interoperability Facility
MBBC	Model-Based Battle Command, koncept myntat av Sam Chamberlain vid US Army Research Laboratory
MoS	Modellering och Simulering - begrepp
NATO	North Atlantic Treaty Organization
NFS	Network File System - distribuerat filsystem
NP	Nondeterministic Polynomial time - en klass inom komplexitetsteorin för beräkningsmässigt svåra matematiska problem, ofta av kombinatorisk natur (NP-fullständighet)
ODBC	Open DataBase Connectivity - generellt SQL-gränssnitt
OGC	OpenGIS Consortium

OKBC	Open Knowledge Base Connectivity
OLE	Object Linking and Embedding technolog från MicroSoft
OMG	Object Management Group, standardiseringsorgan
OMT	Object Model Template
OPIL	Operativa insatsledningen - ny stab i toppen på svenska försvarets organisation från år 2000-07 (enligt plan)
ORB	Object Request Broker, mäklarsystem inom CORBA
OSF	Open System Foundation
PGP	Pretty Good Privacy - ett program för kryptering av elektronisk post och filer, använder öppen-nyckel-kryptering
PIN-kod	en typ av säkerhetskod
RMA	A Revolution in Military Affairs
RMI	Remote Method Invocation, metod att länka Javaprogram till CORBA
ROLF	Rörlig Operativ LedningsFunktion - Försvarshögskoleprojekt som utvecklar en demonstrator för en framtida stabsmiljö, baserad bl a på utnyttjande av VR-teknik (Akvarieprojektet)
RPC	Remote Procedure Calls, standardtillägg till TCP/IP för att anropa procedurer i främmande processer/program över ett lokalt nätverk
RRV	Riksrevisionsverket, som 1996 granskade försvarets utveckling av nya ledningsstödsystem
RTI	Run Time Infrastructure, distribuerat operativsystem som sköter all kommunikation mellan federater (HLA)
SAIC	Science Applications International Corporation, amerikanskt konsultföretag som specialiserat sig på att sälja kunskaper och studier till US Department of Defense. 1998 genomförde det på svenskt uppdrag studien “Dominant Battlespace Awareness for the Swedish Armed Forces 2020”. Här omnämns SAIC-modellen.
SE	Synthetic Environments - syntetiska omgivningar inom VR-teknik
SESAME	Secure European System for Applications in a Multi-vendor Environment - en nätverksbaserad behörighetstjänst
SHAPE	NATO Supreme Headquarters, Allied Powers Europe
SISO	Simulation Interoperability Standards Organization
SOM	Simulation Object Model, inom HLA
SQL	Structured Query Language, frågespråk för relationsdatabaser
SSL	Secure Socket Layer
STRICOM	US Army Simulation Training, and Instrumentation Command
STRIC	flygvapnets stridsledningssystem
STRIMA	marinens stridsledningssystem
TCP/IP	Transmission Control Protocol / Internet Protocol
UML	Unified Modeling Language, metod för objektorienterad modellering av Booch, Rumbaugh och Jacobson
VMF	Variable Message Format, ett nytt kommunikationsformat som utvecklas av US Army för användning i de s k Force XXI-övningarna
VPN	Virtual Private Network
VR	Virtual Reality - virtuell verklighet
VRT	Virtual Retinal Display
WAP	Wireless Application Protocol - för mobiltelefoni över Internet
WWW	World Wide Web

XML	eXtensible Markup Language (ny standard för webbinformation, jfr HTML)

7.5 Källor

Beteckning	**Förklaring**
Svenska datatermgruppen	http://www.nada.kth.se/dataterm/rek.html
DoD Dictionary of Military Terms	På Internet adress http://www.dtic.mil/doctrine/jel/doddict/ fanns 1998-02 utdrag ur Joint Publication 1-02, "DOD Dictionary of Military and Associated Terms"
NE - Nationalencyklopedin	den svenska nationalencyklopedin om 20 band, utgiven under perioden 1989 - 1996