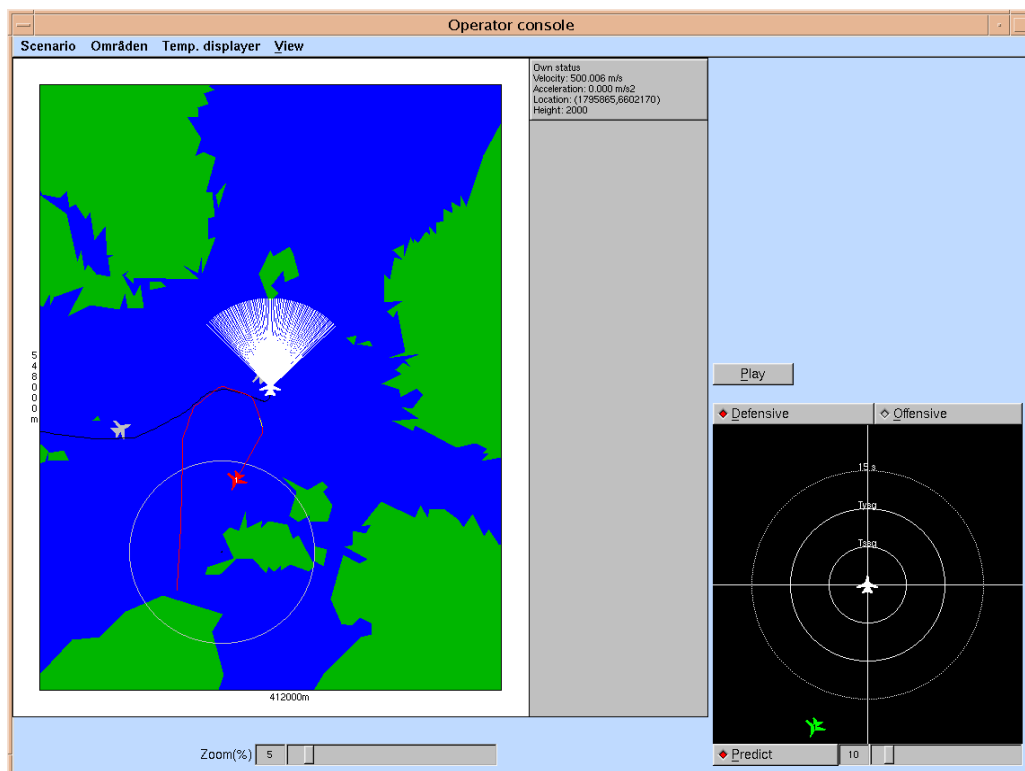


Tomas Berg, Fredrik Lantz, Dan Strömberg

Plattformar i nätverk - Mjukvaruarkitektur och operatörstjänster



Teknisk rapport

Tomas Berg, Fredrik Lantz, Dan Strömberg

Plattformer i nätverk - Mjukvaruarkitektur och operatörstjänster

Utgivare Totalförsvarets Forskningsinstitut, FOI Ledningssystem Box 1165 SE-581 11 LINKÖPING	Rapportnummer, ISRN FOI-R--0696--SE	Klassificering Teknisk rapport
	Forskningsområde 8 Människan i totalförsvaret	
	Månad, år December 2002	Projektnummer E7022
	Verksamhetsgren 5 Uppdragsfinansierad forskning	
	Delområde 81 MSI med fysiologi	
Författare/redaktör Tomas Berg, Fredrik Lantz, Dan Strömberg	Uppdragsgivare/Kundbeteckning Försvarmakten	
	Projektansvarig Dan Strömberg	
	Tekniskt och/eller vetenskapligt ansvarig Författarna	
	Godkänd av Tf IC Andris Laubertz	
Dokumentets titel (I översättning) Plattformer i nätverk - Mjukvaruarkitektur och operatörstjänster		
Sammanfattning (högst 200 ord) Rapporten har två huvudbudskap: (1) Nätverkslösningar kräver speciell mjukvarumetodik, och (2) För framgångsrik informationshantering i plattformsnätverk krävs en öppen systemansats samt enkla basprinciper och flexibla byggstenar. Högkvalitativ mjukvaruarkitektur är ett fundament och förutsättning för samverkan mellan operatörer i mobila farkoster i militära tillämpningar. Krav och former för - samt innehåll i - en sådan arkitektur diskuteras. Avsikten är att identifiera metoder och verktyg med vilka operatörer i plattformer av olika slag kan samverka i gemensamma insatser och uppgifter. Exempel på sådana tjänster ges. Den föreslagna arkitekturen konkretiseras med en egenutvecklad programvara för plattformsbaserad datafusion och sensorstyrning. En datafusionsnod samordnar dessa processer i varje plattform. Konkurrerande sensorberoende uppgifter kan genereras av operatörer och plattformsburna delsystem. Agentteknik används för att modellera dessa uppgifter. Ett ramverk för nätorienterad programvara bildar grund för plattformsextern kommunikation. Möjligheterna att tillgodose krav på kvalitet, interoperabilitet, dynamiska scenarier och utvidgningssbarhet diskuteras. Jämförelse görs med internationella, främst USA-ledda, forskningsprogram med likartade mål. Relaterade industriprojekt jämförs och kommenteras.		
Nyckelord Operatör, tjänst, uppgift, agent, plattform, arkitektur, programvara, mellanvara, nät.		
Övriga bibliografiska uppgifter		Språk Svenska
ISSN 1650-1942	Antal sidor: 44	
Distribution enligt missiv	Pris: Sekretess	

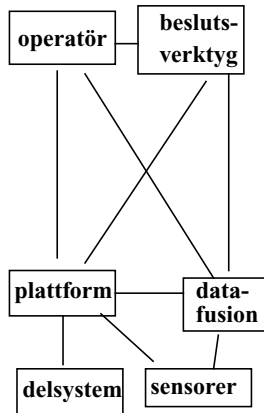
Issuing organization Swedish Defence Research Agency, FOI Command&Control P.O. Box 1165 SE-581 11 LINKÖPING SWEDEN	Report number, ISRN FOI-R--0696--SE	Report type Technical Report
	Research area code 8 Human Systems	
	Month, year December 2002	Project no. E7022
	Customer code 5 Commissioned Research	
	Sub area code 81 Human Factors and Physiology	
Author/s (editor/s) Tomas Berg, Fredrik Lantz, Dan Strömberg	Sponsoring agency Swedish Armed Forces	
	Project manager Dan Strömberg	
	Scientifically and technically responsible The authors	
	Approved by Tf IC Andris Laubertz	
Report title (In translation) Platforms in networks - Software architecture and operator services		
Abstract (not more than 200 words) The report has two messages: (1) Network solutions require special software methodology, and (2) Successful information management in platform networks require open systems solutions plus simple basic principles and flexible building stones. High quality software architecture is a fundamental requirement for cooperation tools to be used by operators in mobile platforms in defence applications. Forms and content for such an architecture are discussed. The goal is to identify methods and services that support operators in platforms of different kinds aiming to cooperate in common tasks. Examples of such services are given. The proposed architecture is concretized with a self-developed tool for platform based data fusion and sensor management. A data fusion node coordinates these processes. Competing sensor tasks that ask for the same resources are generated by operators and platform based sub-systems concurrently. Multi agent technology is used to model these tasks. A framework for network oriented software is fundamental for external communication. Possibilities to satisfy requirements on quality, interoperability, testability and extensibility are discussed. Comparison with international research programs with similar goals is done. Related industrial projects are compared and commented.		
Keywords Operator, service, task, agent, platform, architecture, software, middleware, network.		
Further bibliographic information		Language Swedish
ISSN 1650-1942	Pages: 44	
	Price acc. to pricelist Security classification:	

Innehåll	Kapitel	Sidan
	1. Introduktion	5
	2. Arkitekturmässiga fundamenta	7
	3. Informationskrav	9
	4. Funktionella krav	13
	5. Modeller och metoder	17
	6. Återbesök i arkitekturen	19
	7. Implementationer	23
	8. Internationell utblick	35
	9. Svensk utblick	37
	10. Slutsatser	41
	11. Referenser	43

Projektet Arbetet bakom denna rapport har finansierats inom ramen för FoT-projektet “Datafusion för plattformsoperatörer”. Projektet ingår i FoT-området MSI, Människa/System-Interaktion. Forsknings- och Teknikutvecklings- (FoT) projekten beställs av FM, den svenska försvarsmakten. Rapporten är en leverans från projektet.

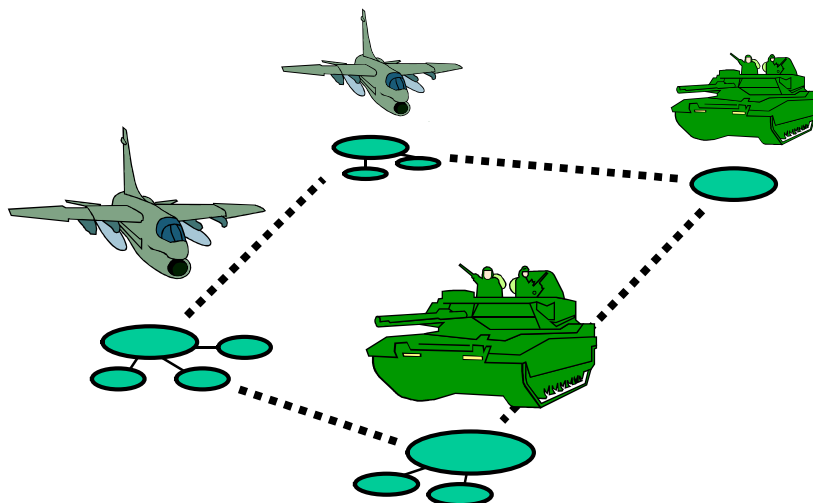
Medhjälpare Flera kund- och forskargrupper inom och utanför FOI har bidragit till arbetet bakom - och resultaten i - denna rapport. Det mesta jobbet har projektdeltagarna bidragit med. Förutom författarna bör nämnas Göran Pettersson, Martin Folkesson, Robert Forsgren, Maria Andersson och Kasper Ezelius. Tomas Kaijser har hjälpt till med korrekturläsning. Vi tackar alla.

1. Introduktion



Denna rapport handlar om plattformar i nätverk. En plattform kan vara mobil och mark-, sjö- eller luftburen. I varje plattform finns en eller flera operatörer, med undantag för obemannade plattformar. För att operatörer i olika plattformar skall kunna samverka krävs väl anpassade beslutsstödsverktyg. Dessa är då beroende av att det finns nätverk mellan plattformarna. För att kunna konstruera sådana beslutsstöd och nätverk måste operatörernas behov vara klarlagda. Rapporten diskuterar dessa behov, och vilka krav de ställer på beslutsstöden och nätverken. Den visar också hur motsvarande konstruktioner bör se ut. Framställningen illustreras förlöpande med ett scenario.

Sett ”uppifrån” kan ett nätverk med samarbetande enheter tecknas på ett tvådimensionellt vis. Varje enhet är inkopplad i nätverket på något sätt, och kommunicerar med en eller flera olika parter. Kommunikationen består av utbyte av kommandon och data. Figur 1.1 visar ett platt nätverk där varje part är en nod.



Figur 1.1. Plattformar i nätverk, plattformarna är utrustade med sensorer

Arkitektur

Vid uppbyggnaden av nätverk är programvaruarkitekturen en konstruktionsmässig huvudfråga. Arkitekturen för ett husbygge kan sägas bestå av materialmässiga fundamenta samt regler och ritningar för den planerade konstruktionen. Motsvarande resonemang gäller för programvaruarkitekturer, om materialet/hårdvaran ersätts med mjukvara. Om konstruktionen gäller ett nätverk, tillkommer regler och planer för hur noder skall kunna kopplas ihop och samverka i nätet. Utan en enhetlig arkitektur kommer nätverket inte att fungera. Innebörden är inte att alla noder måste ha samma interna regelverk, men att vissa regler måste ställas upp beträffande sammankopplingarna av noderna.

uppgifter, tjänster



arkitektur



nätverk

Allmänna fundamenta för programvaruarkitekturer skall behandlas i kapitel 2, men två fakta nämns redan nu: (1) Programvara för nätverk bör byggas enligt flerlagerprincipen. (2) Arkitekturen måste vara öppen. Dessa begrepp diskuteras nedan.

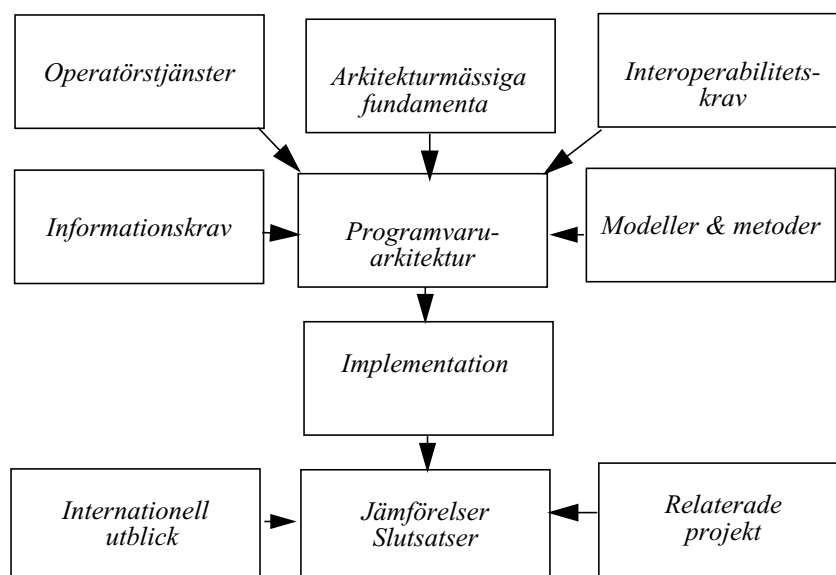
Specifika applikationsberoende arkitekturkrav beror på vilka uppgifter, funktioner och tjänster som kunder och användare vill att nätet ska tillgodose, och vilken strategi som ska tillämpas för användningen av dem. Efter det kan en mera detaljerad arkitektur föreslås och utvärderas, bland annat med avseende på prestanda, funktionalitet, säkerhet, underhållsförmåga och interoperabilitet. Existensen av en bra arkitektur öppnar sedan vägen mot resten av konstruktionen.

Disposition

Ovanstående resonemang utgör röda tråden i denna rapport, se figur 1.2. Först presenteras arkitekturmässiga fundamenta för nätverk. Sedan diskuteras olika användares informationskrav, och särskilt tjänster och uppgifter ur operatörsperspektiv. Även interoperabilitetskraven diskuteras. Ett kapitel ägnas åt modeller och metoder. Efter en förnyad titt på programvaruarkitekturen presenteras några viktiga principer för realisering och design av programvara byggd med sådan arkitektur. En egenutvecklad programvara illustrerar och konkretiserar arkitekturen. Programvaran utvärderas i demonstrationer och scenarier.

Rapportens avslutande del innehåller tre kapitel. Det första ger en internationell utblick på arbeten med koppling till projektet. Det andra beskriver några relevanta bredvidliggande svenska projekt, inklusive en workshop som projektet arrangerat. Därefter följer slutsatser och referenser.

Framställningen illustreras av ett scenario som återkommer i varje kapitel.



Figur 1.2. Disposition.

2. Arkitekturmässiga fundamenta

Flernivå-arkitektur

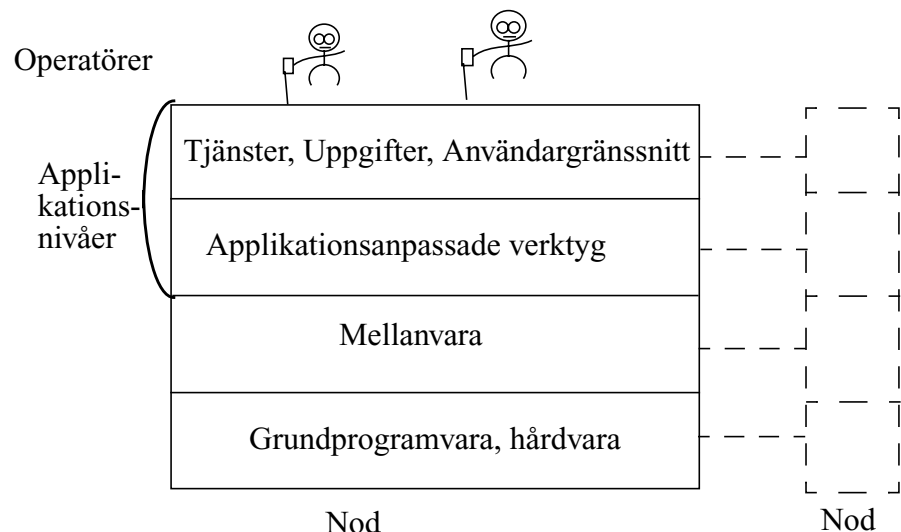
Programvaran för stora nätverk är komplex. Därför är struktur och konstruktion av yttersta betydelse för dess funktion. I programvarutekniken används nivåansatsen som ett sätt att bemästra komplexiteten. I nivåansatsen indelas konstruktionen i nivåer, och enbart en viss typ av frågor hantearas på varje nivå. Flernivåarkitekturer ger möjlighet att 'abstrahera bort' saker för konstruktören och teknikern som inte är omedelbart relevanta för det problem som han/hon för tillfället arbetar med. Materialet kan delas upp och ett problem i taget kan bearbetas.

Abstraktion

Abstraktion har sedan länge betraktats som en viktig programmeringsparadigm, och abstraktionsmekanismer har alltid varit ett forskningsämne i programvaruteknik. Några tekniker som har dominerat i detta avseende är funktions- och objektorienterade programmeringsmetoder samt agent-system.

Mellanvara

Figur 2.1 visar en skiss av några grova nivåer i nätorienterad programvara. Där visas överst två nivåer av applikationsprogramvara, sedan kommer en nivå med mellanvara - från engelskans middleware - och underst en nivå med grundprogramvara som omfattar operativsystem, radionät och hårdvara eller maskinvara. Mellanvaran kan också kallas mellanlagret.



Figur 2.1. Flerlagerarkitektur för nätverksorienterad programvara. Varje nod residerar i en plattform och består av flera lager.

Objektorientering

Mellanvaran är nödvändig i nätverkskonstruktioner. Den innehåller programvara som varken tillhör applikationen eller operativsystemen. Den omfattar bland annat all sådan programvara som används i Internet och webdesign. Ofta används här objekt-orienterade tekniker, bland annat för att underlätta applikationsprogrammerarens kontakter med operativsystem. UNIX och Windows är exempel på operativsystem. Genom att använda objektorienterad mellanvara kan konstruktören använda högnivå-

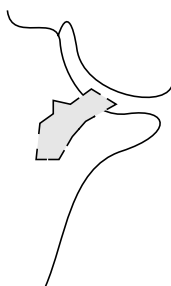
faciliteter, såsom strukturerad programmering, inkapsling, ärvning, modularitet och polymorphism. Separeringen av applikationsprogramvara från operativsystemen har också andra fördelar. Bland annat döljer den skillnaderna mellan olika operativsystem.

Applikationsprogramvaran i figur 2.1 har indelats i två nivåer, en för tjänster och uppgifter och en för applikationsverktyg. I kapitel 6 och 7 skall dessa nivåer förtydligas.

Överst i figuren illustreras ett par användare. Dessa har direkt kommunikation med systemet genom användargränssnitt (user interface), som ingår i arkitekturs toppnivå. Dessutom har en andra nod indikerats i figuren. Denna illustrerar kommunikation till andra noder i nätverket. Varje nivå i en nod kan - logiskt sett - kommunicera med samma nivå i annan nod. En nod representerar programvaran i en plattform, en självständig sensor eller en användarstation.

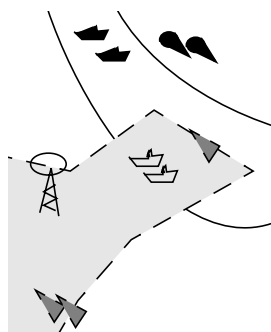
Innehållet i mellanvaran utgörs till stor del av olika programmeringsmönster som kan återanvändas och anpassas i olika nätverk. Dessutom kan finnas applikationsspecifika tillägg, vilket visas i kapitel 6 och 7.

2.1 Illustrativt scenario



En behovssammansatt svensk styrka är insatt i en fredsbevarande operation i Afrika. Styrkan består av fem jaktflygplan, två korvetter och ett tjugotal markfordon. Bland de senare finns två stycken mobila radarstationer av giraff-typ. Styrkan samarbetar med styrkor från Danmark, Norge, Finland och Tyskland och leds av en tysk överste. Den svenska styrkans uppgift är att i en viss region särskilja två stridande parter, och därvid samarbeta med de tyska styrkan. Denna omfattar luft- och markburna enheter. Regionen omfattar både land och hav. Stridshandlingar förekommer både till havs, i luften och på land.

Händelseförlopp



I ett övervakningsuppdrag observerar en svensk jaktpilot en grupp vapenbärande fartyg på väg mot land i det "svenska" området. Fartygens avstånd till land är cirka 200 km. De svenska fartygen ligger strax utanför kusten, med hög beredskap. Övriga svenska flygplan är engagerade i ett uppdrag cirka 1000 km från platsen. Efter överläggning med den markbaserade jaktledaren beslutas att flygföraren i radartyst läge skall göra okulärbesiktning av de observerade fartygen. Radartäckningen skall utföras av en av de svenska korvetterna samt, så snart detta är möjligt, av ytterligare ett av de svenska jaktflygplanen som får till uppgift att stödja bevakningsoperationen. Under tiden skall flygföraren få erforderlig radartäckning av en svensk markbaserad radarstation samt en korvett. Jaktledaren vet från annan källa att en stridande part också har startat ett flygföretag på väg mot regionen.

3. Informationskrav

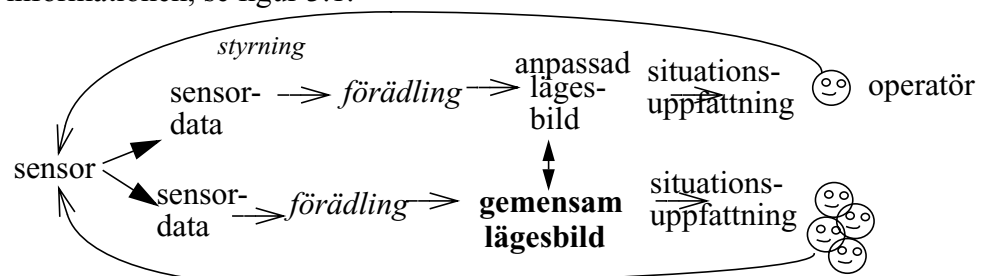
Grundläggande kundkrav ges i [17] och [18]. Dessa kan transformeras till funktioner och uppgifter på operatörsnivå, så att de passar in på arkitekturskissen i figur 2.1. Kundkraven på denna nivå beror på de uppgifter som plattformsoperatörer skall kunna utföra i aktuella typer av uppdrag, bland annat uppdrag i internationella koalitionsinsatser samt samverkan med civila myndigheter. Genom att också ta hänsyn till andra informationskällor, se till exempel [2], är det möjligt att göra en tentativ lista av giltiga användarkrav.

Glb

I försvarsmaktens aktuella NBF-planering fokuseras kundkravet på 'battle space awareness', det vill säga information och datainsamling, situationsanalys, hotbedömning och styrning. Ett dominerande krav är det om gemensam lägesbild, glb. Den engelska termen lyder Common Operational Picture alternativt Consistent Operational Picture, COP. Glb definieras i detta sammanhang såsom dels en mängd data och informationer om omvärlden, dels en uppsättning funktioner för insamling, sammanvägning, framställning och presentation av dessa data.

Plattformsoperatör

I detta dokument fokuseras på plattformsoperatörens behov av situationsuppfattning. En plattformsoperatör har delvis andra krav på informationshanteringen än den som tillhandahålles i en glb. Förmågan till situationsuppfattning är alltid viktig, och kan vara avgörande för hans/hennes förmåga och överlevnad. Vad som ligger i situationsuppfattningen varierar dock från operatör till operatör och från plattform till plattform, även om vissa allmänna faktorer och tillstånd är gemensamma [6]. Situationsuppfattningen är i grunden en mental process hos operatören, men vad som ligger i en plattformsoperatörs situationsuppfattning beror på vilken plattform som ska hanteras, vilken roll operatören har, vilket uppdrag han/hon har och vilken uppdragsmode och fas han/hon befinner sig i. Verktyg för situationsuppfattning bör helst kunna anpassas till alla dessa faktorer. Att operatörsspecifika verktyg inte kan inrymmas i en glb-orienterad verktygsmängd inses lätt genom att titta på en bilförarens normala informationsbehov. Glb kan i detta sammanhang utnyttjas som en av många datakällor, men varje operatör har unika krav på bearbetningen av sensorinformationen, se figur 3.1.



Figur 3.1. Situationsuppfattningen är unik för varje operatör. Den gemensamma lägesbilden är inte tillräckligt underlag för hans/hennes situationsuppfattning.

Situations- uppfattning

Operatörer i plattformar har höga krav på situationsuppfattning. Detta informationsbehov kännetecknas av - respektive är förenad med -

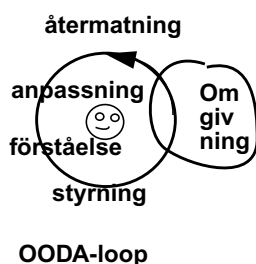
- hög rörlighet hos egen plattform
- snabba och rörliga förlopp i omvärlden
- tidskritisk detektion, uppföljning och analys av inhämtad information
- informationsinhämtning under stress
- hög arbetsbelastning
- reaktiva förlopp, dvs egen manövrering påverkar omgivande förlopp
- krav på insatser för att kontrollera respektive manövrera plattformen
- krav på insatser för att ha "koll på läget"
- krav på insatser för att manövrera delsystem inklusive sensorer

Dessa kännetecken är mer eller mindre framträdande hos plattformsburna operatörer. De gäller i högre grad för operatörer som samtidigt manövrerar sin plattform än för sådana som sitter i ett ledningsrum till exempel på ett fartyg. Vi har här fokuserat på operatörer som har både manövrerande och situationsbedömande uppgifter.

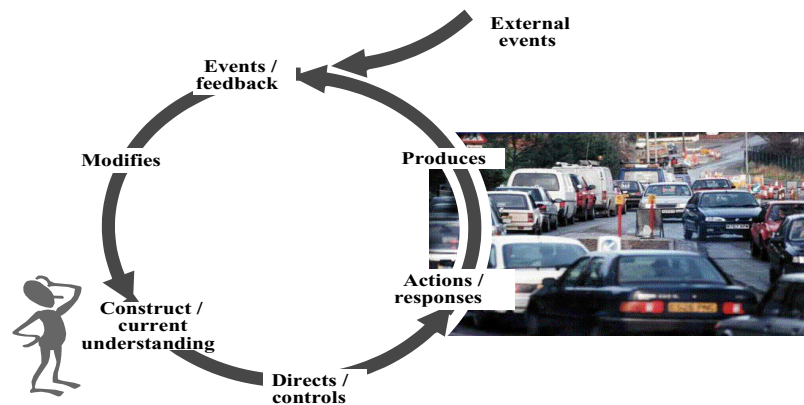
Resursdelning

För bra situationsuppfattning behöver plattformsoperatören ofta data från samma källor som används för att samla in data till glb, se figur 3.1. Olika konflikter kan uppstå i detta sammanhang: Samma sensorer kan behövas för många ändamål och för många operatörer och användare. Processorkraft skall fördelas på olika uppgifter och behov. Samtidigt kan plattformsoperatören vilja använda sin sensorutrustning och processorkraft på ett sätt som inte är optimalt och anpassat för den glb.

Beslutsloopar



Här införs nu begreppet OODA-loop (svenska: beslutscykel eller beslutsloop(!)). OODA står för Observe, Orient, Decide och Act. Beslutsloopen är en modell för de mentala processerna i operatörens interaktion med omgivningen. Den utgör ett cirkulärt flöde med processer för datainsamling, situations- och hotanalys, beslutsfattande samt styrning och anpassning. Den kan användas i alla typer av operatörsarbete, och har av kognitionsvetare kallats för "den grundläggande loopen för mänskligt beslutsfattande" [9], se figur 3.2. Beslutsloopen kan förstärkas genom användande av maskinella hjälpmedel, såsom konstgjorda sinnesorgan eller sensorer. Även om modellen kan kritiseras, se till exempel [15], så är den ett av få tillgängliga verktyg för att modellera interaktionen mellan operatör och maskin, så att de tillsammans bildar ett system. Sensorer och annan processorkraft ingår ofta såsom hjälpmedel i interaktionen mellan operatör och omgivning.



Figur 3.2. Beslutsloopen beskriver de mentala processerna i operatörens interaktion med omgivningen. I figuren är operatören bilförare. (Från [9]).

En beslutsloop kan inriktas mot ett speciellt utsnitt av verkligheten, och en operatör kan arbeta med flera sådana beslutsloopar samtidigt. Olika beslutsloopar kan också nästlas i varandra, så att till exempel en loop gör saker som används i en annan beslutsloop på högre nivå.

I många autonoma plattformsburna delsystem kan en beslutsloops-liknande modell användas för att beskriva deras interaktion med omgivningen. Detta gäller exempelvis transpondersystem, kommunikationssystem, navigationssystem och telekrigssystem. Slutligen kan beslutsloopsmodellen också användas för att hantera beslutsmekanismer och interaktion mellan operatörer och beslutsverktyg, enligt en modell som ska presenteras i kapitel 6 och 7 nedan.

Informationskällor

Plattformsoperatörer kan få sin information på i stort sett fyra olika sett, genom

- förädling av data erhållna från ensam sensor på plattformen
- förädling av data erhållna från nätverk av sensorer på plattformen
- förädling av data erhållna från sensor(er) på en eller flera andra plattformar
- förädling av data erhållna på andra sätt, till exempel HUMINT (human intelligence).

Data och information kan transformeras till en operatör eller användare i olika imperativ

- såsom larm från någon
- såsom information från vänner som vet vilken information han/hon behöver och vill ha
- såsom information från sensorer och andra system som 'vet' vilken information han/hon behöver och vill ha.
- såsom svar på frågor eller beställningar som han/hon ställt.

Informationsförsörjning

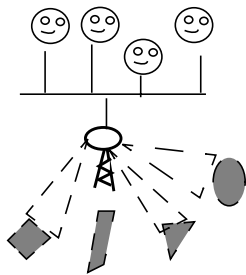
För varje operatör kan det finnas agenter (människor eller system) som känner till dennes existens, mål, planer och syften. De kanske också känner

till hans/hennes belägenhet, och hur hans/hennes uppdrag framskrider. Sålunda ingår information både om det mänskliga läget (såsom vakenhet, motivation, måttnad) och det tekniska interna läget (såsom beredskapsläge, resurser, bränsle, etcetera) i det underlag som informationslämnaren behöver känna till.

Beställning

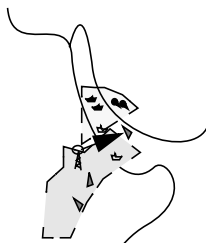
De informationskällor som listats under rubriken 'informationskällor' ovan kan ha mer eller mindre tillförlitlig information om operatörens tillstånd och behov. Ovissheten om operatörens informationsbehov kan minskas genom att använda ett beställningssystem, där informationshanteringen istället baseras på beställningar av olika typer av informationstjänster. Varje operatör och användare får då exakt den information han har begärt. Vissa speciella funktioner såsom larm-funktioner kan hanteras som beställningar som finns i default.

Beslutsfattande



I ett system med många användare och operatörer innebär detta att - i princip - varje användare kan beställa informationstjänster över nätet. Dessa skulle i så fall kunna ställas till enstaka sensorer och plattformar, eller i form av funktioner ("jag vill veta allt om X"). Eftersom alla resurser kan betraktas som bristvara, innebär det att varje informationsproducerande agent själv måste välja vilka beställningar han/hon skall utföra, och hur mycket resurser som skall tilldelas varje beställning. Detta kan hanteras antingen manuellt av en operatör - vilket skulle ta mycket mantid i anspråk - eller av en automat hos honom/henne som styrs av någon slags resursallokeringspolicy som talar om vilka typer av beställningar som kan tillfredsställas i olika situationer hos plattformen. Beställningen kan också innehålla information som underlättar beslutet. I vilket fall bör det fattas enligt regler om vilka slags operationer som plattformen och eller sensorn skall delta i. Exempel på sådana regler finns bland annat i [4]. Distribuerade samverkande agent- och beslutsstödssystem beskrivs bland annat i [28].

3.1 Illustrativt scenario

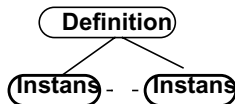


Den lägesbild som är tillgänglig för svenskarna innehåller observationer och målinformation som insamlats och framställts av danska och tyska radarstationer. Anpassningar och justeringar fördröjer bilden med 20 - 40 minuter. Den svenske jaktpiloten är för sin omvärldsinformation därför beroende av radarbilder direkt från markradarstationen i inledningsskedet. Korvetten har radartäckning ovanför - men ej på - havsytan i de aktuella områdena. Stödjande jaktflygplan kan ge radarinformation om cirka 30 minuter.

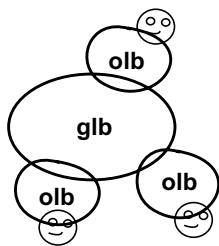
4. Funktionella krav

4.1 Operatörstjänster

En tjänst är en systemkomponent i mjukvara och inte i hårdvara. Här görs inledningsvis en tolkning av begreppet 'tjänst'.



Varje tjänst kan existera i två former. Dels har den en definition, dels kan den instansieras. Definitionen existerar såsom en komponent någonstans i nätverket. Så fort en användare vill bruka den, bildas en instans. Ibland används istället termerna abstrakt tjänst respektive konkret tjänst. Instansen lever så länge som användaren vill. Det kan finnas flera instanser samtidigt av en tjänst, till exempel beroende på att flera användare har anropat den. En tjänst kan realiseras via en mekaniserad beslutsloop/task, se kapitel 6 och 7. Den kan då kräva att andra resurser, såsom sensorer, utnyttjas. Vilka resurser som ska användas kan vara fördefinierat eller bestämmas vid exekveringstillfället. Instansen kan vara fördelad på ett antal noder i nätverket, utan att användaren behöver känna till vilka. Definitionen av en tjänst är en förutsättning för instansieringen.



Föregående kapitel indikerar att informationsbehovet hos en plattformsoperatör normalt inte tillgodoses av en glb. Varje operatör med sensorer på sin plattform kan antas kräva att få utnyttja dem på det sätt han/hon önskar. Den process som operatören deltar i, situationsbedömningen, har utförligt beskrivits av bland annat [6]. Situationsuppfattningen växer fram i en informationsbearbetning som - förenklat uttryckt - kan indelas i tre faser, nämligen perception, situationsanalys och prediktion. Den information som ligger i glb är inte tillräcklig för denna process, vilket inses om man tänker exempelvis på en bilförare. Operatören behöver en operatörs-centrerad lägesbild, olb. Denna kan - men måste inte - överlappa glb. Vissa delar av olb - till exempel flyglägesinformation hos en flygförare eller våghöjdsdata hos en sjöofficer - ingår inte i den gemensamma lägesbilden.

Situationsuppfattning

Ett försök att formulera nätverksorienterade beställningsbara operatörstjänster i detta avseende blir lätt otymplig, eftersom situationsinhämtningen normalt integreras i operatörens totala agerande under ett uppdrags genomförande. Ändå görs här ett försök. Tjänsterna kan klassificeras i tjänster för perception, situationsbedömning, prediktion respektive styrning/anpassning. Dessa nivåer utgör också grund för datafusionsvetenskapen. Lågnivåperception betraktas dock som en mental process som inte styrs av någon identifierbar beställning utan av instinkter, imperativ och funktioner på högre nivåer. Samtliga punkter kan - men måste inte - involvera flera noder, dvs plattformar och/eller sensorer, i nätverket. Snarare än tjänster kan nedanstående lista ses som funktioner vilka en operatör kan framkalla, ofta genom en grafisk interaktion med systemet. Följande indelning är en tentativ och icke-fullständig beskrivning av 'tjänster' och motsvarande funktioner. De låter sig indelas efter viss tvekan.

**Tjänster för
situationsbedöm-
ning**

Operatörfunktioner som underlättar situationsbedömningen.

- ha låg/medel/hög informationsberedskap kring viss plats, väg, område, situation, objekt, företag, företeelse, väderfront
- visa sensorgenererad information om sådan plats etc
- visa databaslagrad information om sådan plats etc
- uppdatera fortlöpande om ny information om sådan plats/tid etc
- visa glb
- sammanväg glb och data i en situationsbild
- informera om tillståndet hos visst mål

**Tjänster för
prediktion**

Operatörfunktioner som underlättar hotanalys och aktionsmöjligheter.

- informera om hotnivån hos visst mål
- prediktera insats vid viss tidpunkt
- visa optimal tid för stridsinsats
- visa tid för största risk

**Tjänster för
styrning/anpass-
ning**

Operatörfunktioner som styr delsystem och sådan informationshantering som påverkar hans eller någon annans situationsuppfattning.

- varna för viss hottyp om så behövs
- informera linjeföring om visst tillstånd eller fakta hos egen plattform
- informera uppdragsledare om ny plan för egen plattform
- informera samarbetande plattform om visst förhållande
- fråga efter tillstånd/tillåtelse att göra viss aktion
- stöd genom att göra viss insats
- gör en ny plan för U, där U är ett uppdrag
- välj fördefinierad sensorstyrpolicy för egen plattform

4.2 Interoperabilitetskrav

Exempel på problem med bristande interoperabilitet brukar inte vara svåra att generera. Problemen kan vara tekniska, semantiska, organisatoriska, metodologiska eller taktiska. De tekniska problemen berör alla nivåer i programvaruarkitekturen enligt figur 2.1. Nedan ges exempel på den lägsta nivån, den som handlar om grundprogramvara, radionät och hårdvara. En motsvarande uppsättning problem finns på övriga nivåer, det vill säga på mellanvarunivån och applikationsnivån. Begreppstolkning och andra semantiska interoperabilitetsoperationer förekommer dels i rent interpersonella informationsutbyten, dels i de tekniska systemen, se till exempel diskussionen om attribut i mellanvaran i avsnitt 6.3.2.

Det finns stora skillnader mellan radiobaserade kommunikationssystem hos olika vapenslag. Samma typ av svårigheter finns hos civila myndigheter som har att samverka i olika räddningsuppdrag eller liknande; exempelvis har polisen, tullverket, sjöräddningen, kommunernas räddningstjänst och kustbevakningen inkomparabla kommunikationsutrustningar. Liknande tekniskt grundade svårigheter uppstår i internationell samverkan.

Följande krav kan ställas på plattformar som ska fungera i internationella militära koalitioner eller räddningsoperationer: (1) Plattformar av olika typ

och av olika nationalitet skall kunna samverka i gemensamma uppdrag. (2) Modifiering eller uppgradering av sensor eller delsystem ombord på en plattform skall icke kräva uppgradering av sensorer eller delsystem på andra plattformar. (3) Mobila plattformar bör kunna sättas in eller gå ut ur uppdrag tillsammans med andra plattformar i "on-the-fly" basis. I kapitel 8 presenteras några aktuella internationella arbeten och ansatser med sikte på ökad interoperabilitet.

Interoperabilitetsproblem och -krav av organisatorisk, metodologisk eller taktisk art berörs ej i detta dokument.

4.3 Illustrativt scenario

Flygföraren får av jaktledaren tillstånd att direkt utnyttja både de markburna och de fartygsburna radarspaningsstationerna. Han/hon beställer då en tjänst som dels startar spaning hos dessa sensorer, dels sammanställer informationen. Han/hon har då nytta av sin hotdisplay, se kapitel 6. Samtidigt inkommer larm från jaktledaren om fientliga flygplan.

5. Modeller och metoder

5.1 Modelleringsansatser

Den ovan beskrivna beslutsloopen har tjänat som modell i ett konstruktionsarbete. Några principer har därvid varit vägledande.

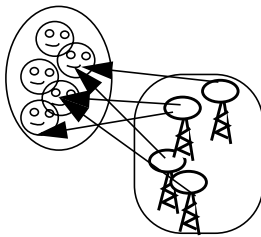
Mekaniserad beslutsloop

Beslutsloopen så som den beskrivs i föregående avsnitt har tjänat som generell princip vid konstruktionen av datafusionsnoden. Det innebär att systemet alltid kan hantera många beslutsloopar, att de genereras/ägs av många olika "agenter" och att varje beslutsloop kan utnyttja flera sensorer.

Multi-agentteknik

Varje beslutsloop kan betraktas som en sensoranvändare, oavsett vem som har skapat den. En sådan sensoranvändare har inte kännedom om vilka sensorer som finns att tillgå, eller om deras prestanda, belastning etcetera. En beslutsloop har inte heller kännedom om andra beslutsloopar. Varje beslutsloop ägs av en plattform. För beslutsloopar som involverar flera plattformar finns nätberoende beslutsloopar.

Varje sensor kan i detta sammanhang sägas vara en utförare, det vill säga en maskin eller agent som utför arbete åt flera sensoranvändare. Sensorn behöver inte veta vilka de aktuella sensoranvändarna är.



Temporära avtal
mellan användare
och sensorer kan
modelleras i ett
multiagentsystem

Vi har alltså två typer av agenter, å ena sidan beställare, dvs tasks eller beslutsloopar och å andra sidan utförare, dvs sensorer. Ett sådant system kan modelleras såsom ett multi-agentsystem. En välkänd applikation för multi-agentsystem är den kommersiella marknaden, där konsumenter/köpare och producenter/säljare ständigt interagerar för att komma överens om uppgörelser/avtal med därtill kopplade förpliktelser. I detta fall utgör beslutslooparna informationskonsumenter och sensorerna informationsproducenter. Upphandling görs och avtal tecknas för varje varv i beslutsloopen. Efter leverans av data & information utförs en förädlingsprocess hos beställaren, där mottaget data bearbetas till exempel med fusion, målföljning och målklassificering. En utförligare beskrivning av denna teknik finns i [28].

Operativsystem

Det finns en välkänd mekanism eller "maskin" som simultant och i små korta tidsintervall och time-sharing-mode kan bearbeta ett stort antal processer, nämligen operativsystem. Denna mekanism finns i alla persondatorer, arbetsstationer, servrar med mera. I detta fall finns samma förhållande. Processerna motsvaras här av arbetande beslutsloopar/tasks. Den mekanism som här avses väljer kontinuerligt vilken uppgift som ska utföras, lämpligen en som har fått "dataleverans" från en sensor, och utför en för denna uppgift lämplig uppsättning behandlingar, bestående bland annat av dataassociation, målföljning, målklassificering, hotanalys, presentation och styrning. En sålunda fungerande maskin utgör därför ett operativsystem för sensorstyrning och datafusion, dvs ett Sensor Management and Data Fusion Operating System, eller SMDFOS.

5.2 Programvarumetodik

De nätverkstjänster som listades i föregående kapitel kan kräva medverkan av flera plattformar. De ska kunna anropas i realtid, under genomförandet av uppdrag. Operatörens tilltro till tjänsterna är i första hand beroende av kvalitet på funktionalitet, interoperabilitet och säkerhet, men andra nödvändiga kvalitetskrav gäller underhållsvänlighet, öppenhet, skalbarhet, utvidgningsbarhet och kompatibilitet.

Programvarustandards

Listorna av tjänster i föregående kapitel har relevans för operatörer och användare som deltar i en behovssammansatt styrka, en internationell koalitionsoperation eller civil räddningsoperation. I alla tekniska stödsystem ställs då höga krav på säkerhet och robusthet. Det är då viktigt att komma ihåg, att en grundläggande förutsättning för att klara dessa krav är goda programvarustandards. Detta kan sägas vara den enstaka viktigaste faktorn för produktionen av säkra och högkvalitativa programvarusystem.

Flernivåarkitektur

God programvarustandard uppnås bäst genom att utnyttja relevant kunskap och förmåga i programvaruteknik [19]. Flernivåarkitekturen är ett uttryck för sådan förmåga och objekt-orienterad programmeringsmetodik är en annan. Utförlig användning av välkända och utvärderade programmeringsmönster är också en första rangens metod för god programvaruteknik. Dessa principer kan väl fungera som vägledande för valet av övriga verktyg och metoder för komplex nätverksprogramvara.

Programmeringsmönster

Några exempel på programmeringsmönster för hantering av objekt i nätverk är tjänsteaccess, konfigurerings, händelsehantering och synkronisering. För dessa och andra programmeringsmönster hänvisas till [19].

6. Återbesök i arkitekturen

Operatörstjänster
Verktyg
Mellanlager
OS, hårdvara

Ovan presenterades först en generell syn på nätverksarkitekturer. Sedan diskuterades uppgifter och tjänster. Här görs ett återbesök i programvaruarkitekturen. Detta och nästa kapitel ägnas åt en konkretisering och exemplifiering av arkitekturen.

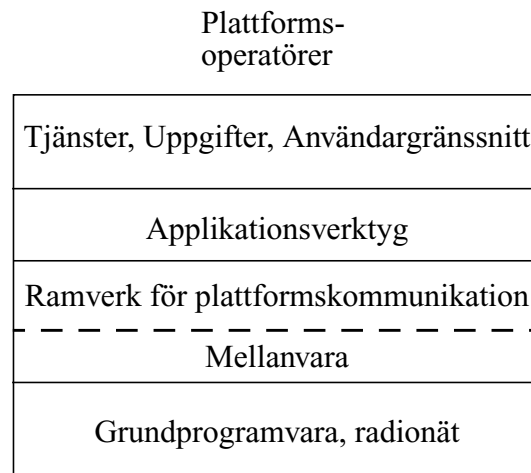
Utgångspunkten är att det finns operatörer i olika mobila plattformar, och att de eftersträvar samverkan. Samverkan betyder i detta fall medverkan i uppdrag där den enskildes mål åtminstone delvis sammanfaller med målet i det gemensamma uppdraget. Vi antar vidare att alla noder, dvs plattformar, både kan utföra och beställa tjänster.

Vi har alltså ett nät där varje nod bör kunna

- Göra informationsbeställningar
- Ta emot och utföra informationsbeställningar dvs produktion av information i plattformsburna sensorer
- Prioritera och välja mellan inkomna konkurrerande informationsbeställningar
- Returnera resultatet till beställande nod
- Sammanställa och presentera resultat av tjänstebeställning
- Hantera avbrott i informationsbeställning

För att kunna studera denna typ av tjänster har ett verktyg, datafusionsnoden, konstruerats. Den kan simulera dels informations- och dataflödet inom en plattform, dvs mellan sensorer, operatörer och delsystem i plattformen, dels informationsflödet mellan plattformar, av godtycklig typ. Den baseras på en objektfiering och mekanisering av beslutsloopen, ibland kallad 'task' (engelskt uttal). Varje operatör eller plattformsburet delsystem kan starta en eller flera beslutsloopar. Nya beslutsloopar kan startas autonomt av existerande beslutsloopar. Datafusionsnoden är ett verktyg för studier av plattformsbaserad datafusion och sensorstyrning. Den passar in på lagret för applikationsanpassade verktyg, se figur 6.1. Datafusionsnoden kan användas av olika beslutsstödsverktyg ombord på en sensorförsedd plattform. Observera dock att datafusionsnoden bara visar på ett sätt att realisera dessa processer.

För att kunna användas i beslutsstöd för samverkan mellan operatörer i olika plattformar, måste datafusionsnoden kompletteras med moduler för nätverkskommunikation. Dessa moduler bildar ett ramverk för plattformskommunikation, vilket på grund av sin sammansättning och funktion är en del av mellanvaran, se figur 6.1. Ramverket presenteras i avsnitt 7.3.



Figur 6.1. Flerlagerarkitektur för nätverkande programvara. Ramverket för plattformskommunikation tillhör mellanvaran men anpassas till applikationen.

6.1 Handlingsfrihet för nätverkssamverkan

Det finns olika modeller för hur nätverkssamarbete skall fungera. Dessa modeller använder sig av plattformar och sensorer på olika vis, de låter plattformarna och sensorerna spela olika roller i ett givet scenario. En modell kan låta enskilda sensorer ha en hög grad av självständighet, medan en annan modell kan lägga all sensorkontroll under en viss plattform och kanske under direkt kontroll av en enskild operatör.

Man kan tänka sig att flera olika modeller för nätverkssamarbete verkar samtidigt i samma nätverksmiljö, fungerande isolerat eller helt integrerat i varandra. En gemensam bas ger större integrationsmöjligheter. Denna gemensamma bas utgörs av mellanvaran och grundprogramvaran, se figur 6.1.

Ramverket måste tillgodose olika behov i olika samarbetsmodeller. På så sätt ger ramverket handlingsfrihet åt den som ska bestämma vilket - eller vilka - slags nätverkssamverkan han/hon vill ha. Ett bra ramverk tillåter att flera slags samverkansformer kan tillämpas, samtidigt eller sekventiellt. Denna möjlighet är mycket värdefull för den som vill kunna samverka i former och nätverk som han/hon inte känner till från början. I nästa stycke diskuteras en nivåstruktur där förutom programvaruarkitekturen också andra systemkomponenter ingår.

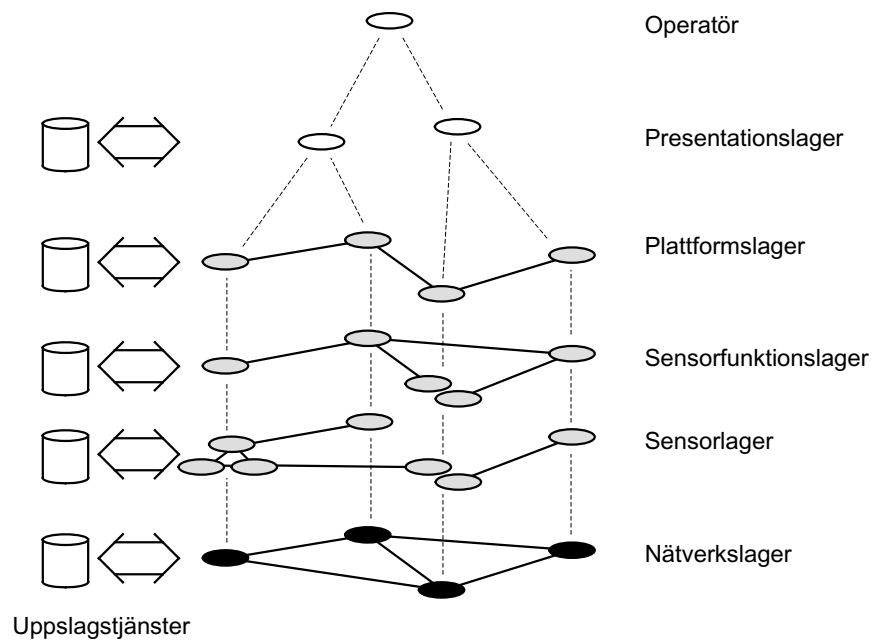
6.1.1 Olika lager av funktionalitet

Olika former av funktionalitet och tjänster, i t.ex. sensorer eller plattformar, grupperas i varsitt lager, se figur 6.2.

Olika samarbetsmodeller, t.ex. DISSY eller WASP (se kapitel 9), kan beskrivas i ett eller flera lager. I de fall då modellerna har liknande funktionalitet ser man ett gemensamt lager, alla modeller utnyttjar t.ex. samma grundläggande nätverkslager. Sensorlagret däremot, är delat i två skikt beroende på de olika sätt som modellerna utnyttjar sig av sensorer.

Komponenter i varje lager kommunicerar med andra komponenter i samma lager, eller utnyttjar sig av underliggande lager.

Följande är ett exempel på en sådan lagerstruktur.



Figur 6.2. Lagerstruktur, olika lager av funktionalitet har identifierats.

Heldragna linjer markerar kommunikation. Bilden visar kommunikation mellan sensorer samt kommunikation mellan plattformar.

Streckade linjer markerar användande. I bilden använder sig plattformarna av sina egna sensorer för informationsinhämtning. Man kan tänka sig scenarion där plattformar direkt använder sig av andras sensorer.

Till varje lager finns en uppslagstjänst. Denna används för registrering och sökning av olika komponenter. Registrering och sökning sker efter olika sökkriterier, t.ex. sensorer kan slås upp efter typ, kvalitet eller geografisk position.

Användning

Varje system som använder komponenter i de olika lagren i figur 6.2 bestäms av någon programvara, vars arkitektur enligt tidigare kapitel bör ha en struktur som visas i figur 6.1. På så vis kompletterar figurerna 6.1 och 6.2 varandra. Varje användning av strukturen i figur 6.2 bör implementeras

av programvara som har en struktur enligt figur 6.1. Resten av detta kapitel behandlar de olika lagren i figur 6.2.

Nätverkslager	I botten finns det faktiska nätverket, det som med mobil teknologi, kablar eller radio bygger upp det fysiska nät över vilket de högre lagren kommunicerar. Detta lager delas av alla ovanliggande lager. Nätverket tillhandahåller den funktionalitet man förväntar sig av ett nätverk, t.ex. bandbreddsstyrning, multicast, mobilitet etc. I en mer detaljerad bild kan detta lager delas upp ytterligare med avseende på just sådan funktionalitet.
Sensorlager	Sensorlagret är i denna bild det grundläggande tjänstelagret. Sensorerna läser av omvärlden och levererar målspar, plotdata, bilder eller andra former av data. Sensorerna i sensorlagret kan också styras.
Sensorfunktionslager	Modeller som kräver mer av sensorer än rent passiv informationsinhämtning har behov av funktioner ovanpå sensorlagret. Exempel på sådana funktioner är inhämtning och anpassning av data från externa sensorer eller förhandling om ägandeskap av målspar, funktioner som används i DISSY-modellen. Varje enskild sensor måste inte nödvändigtvis ha en egen mot svarighet i sensorfunktionslagret.
Plattformslager	Plattformslagret illustrerar samarbete mellan plattformsnoder. Plattformsnoden är en representation av en fysisk plattform, t.ex. en stridsvagn, ett fartyg eller ett jaktplan. Plattformsnoderna koordinerar en plattforms resurser, t.ex. sensorerna ombord plattformen. I detta lager ryms operatörsstyrda mobila plattformar. Plattformsnoderna kan utbyta data, eller begära av varandra att vissa uppgifter utförs.
Presentationslager	Presentationslagret är operatörens gränssnitt mot nätverket. Operatören får här stöd i form av filtrering och uppslagning i informationsdatabaser. Temporaldisplayen som hämtar sin data ur datafusionsprocesser i en plattform hör hemma i detta lager.
Operatör	Ovan alla lager finns operatören. Operatören har kontroll över de underliggande lagren och hämtar den information han/hon behöver.
Uppslagstjänster	Alla lager behöver uppslagstjänster för att organisera sig så att varje komponent kan få kontakt med varje annan komponent. I de olika lagren har uppslagstjänsten olika form, i nätverkslagret organiseras det fysiska nätet, medan sensorlagret kan behöva uppslagning med hänsyn till sensorers geografiska position och förmåga. Ovanliggande lager använder sig även av underliggande lagers uppslagningstjänster.

7. Implementation

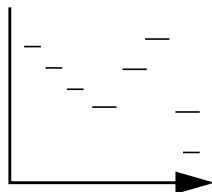
Detta kapitel illustrerar en konstruktion enligt ovanstående arkitektur som har förverkligats. Den består av tre moduler, temporaldisplay, datafusionsnod och ramverk. Implementeringen av datafusionsnod och ramverk beskrivs i detta kapitel. Temporaldisplayen beskrivs kortfattat nedan men har tidigare beskrivits mer detaljerat [26]. Temporaldisplayen ligger i applikationslagret. Datafusionsnoden ligger i verktygslagret och ramverket ligger i mellanvarulagret.

7.1 Datafusionsnod

Designen av datafusionsnoden ([20], [21], [23], [24]) har baserats på följande förhållanden:

- Samverkan mellan en operatör och den maskin som han/hon styr kan beskrivas i termer av beslutsloopar. Varje loop utgör ett cirkulärt flöde där operatör och maskin samverkar. Den kan indelas i faserna data- och informationsinhämtning, data- och informationsanalys, operatörskommunikation samt anpassning/styrning. Enstaka faser kan uteslutas. En beslutsloop kan utföras ett obegränsat antal varv och ha en varaktighet mellan några sekunder och några timmar. Den innehåller normalt en interaktion mellan operatören och omvärlden, men beslutsloopen kan också användas som interaktionsmodell för sambandet mellan sensorsystem och vissa delsystem som utnyttjar sensorantennar, till exempel kommunikationssystem, navigationssystem, telekrigssystem och vapenstyrning. Delsystemet beställer då "beslutsloopar" hos datafusionsnoden. Det finns många typer av beslutsloopar. Inom kognitionsteorin (Cognitive Systems Engineering, CSE) används ibland beslutsloopen som en basal modell för operatörens beslutsfattande, se [9].
- Antal operatörer per plattform kan vara noll, en eller flera. Varje operatör kan ha flera beslutsloopar igång samtidigt.
- Varje beslutsloop utförs i normalfallet med hjälp av en eller flera av de plattformsburna sensorerna. Om flera sensorer kan anlitas kan valet av sensor ske inför varje varv i loopen.
- Operatören/operatörerna bör kunna ha full förfoganderätt över plattformens sensorer. Eftersom han/hon inte kan delta i den korta (millisekundnivå) beslutsprocessen under varje varv i beslutsloopen måste mera långsiktiga processer användas. Dessa operatörsanpassade sensorstyrprocesser utgör sensorstyrpolicies.
- Vissa beslutsloopar kan kräva tillgång till - eller betjänas av - sensorer i flera plattformars sensorsystem. Sådan plattformsoverskridande samverkan behandlas i kapitel 4.
- Enligt föregående punkt kan en plattformas sensorer utnyttjas - till viss del - av operatör på annan plattform. Detta kan dock enbart ske under

Olika sorters beslutsloopar används för spaning, navigering, kommunikation, störning,



sensor som arbetar
tidsdelningsmode.
Ett uppdrag per rad.

förutsättning att operatören på den sensorbärande plattformen i sitt policyval har medgivit att sensorn får utnyttjas på avsett sätt.

- Beslutsloopar i en plattform kan genereras av plattformens operatör(er), andra beslutsloopar på plattformen, plattformsdelsystem samt av källor utanför plattformen.
- Flera sensorer är flerfunktionssensorer. Detta innebär att de kan utföra flera uppgifter simultant, i tidsdelningsmod: varje uppgift utförs i små korta tidsintervall med längre mellanliggande uppehåll. Ett sätt att organisera detta är att för varje uppgift planera nästa steg direkt efter bearbetningen av det föregående. Aktionerna kan ske i tidsintervall som är några millisekunder till några sekunder långa.
- Flerfunktionssensorer karaktäriseras av en hög grad av flexibilitet och anpassning, vilket för ett effektivt utnyttjande nödvändiggör en kontinuerlig styrning och justering av olika sensorparametrar. Denna styrprocess kan med fördel utföras före varje steg i processen.
- För att studera alla dessa frågor behövs ett verktyg med vilket man kan utveckla och utvärdera metoder och algoritmer för datafusion och sensorstyrning.
- I omvärlden finns flera förslag till datafusionsnoder. Ett förslag beskrivs i [7], medan [4] kapitel 15 innehåller ett mera detaljerat förslag, som har påverkat konstruktionen av den egenutvecklade datafusionsnoden.

7.1.1 Implementering

För tidigare versioner av datafusionsnoden se [20] och [21]. Den aktuella versionen har följande delar.

Scenarion

Sensorlösa och sensorförsedda aktörer hanteras i scenarier. Antalet möjliga scenarier är obegränsat. Vid körning väljs ett enda scenario.

Beslutsloopar

Dessa finns bara på plattformsnivå. Antalet beslutsloopar per plattform är obegränsat. Beslutslooparna kan vara av följande typ:

- MTT Multi-Target Målföljning för övervakning i default mode
- MTT Multi-Target Målföljning för övervakning i områdesmode
- STT Single-Target Målföljning
- Målspårsinitiering

För plattformsöverskridande beslutsloopar finns nät-orienterade beslutsloopar (NBL). Varje NBL ägs av en plattform. Den beställer beslutsloopar, sk NBL-barn, hos ett antal andra plattformar och samlar in samt bearbetar målspårsdata från dessa.

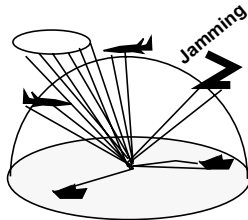
Nya typer av beslutsloopar kan lätt implementeras. Specialanpassade syntaktiska språkkonstruktioner förenklar detta [22]. Bland ännu icke implementerade beslutsloopstyper finns telekrig med offensiv och defensiv

variant, navigation, kommunikation, närområdessökning, övriga övervakning- och spaningsmoder, vapenstyrning och UAV-styrning.

Sensormodeller

Följande sensormodeller finns implementerade:

ESA-radar



ESA-modellen levererar observationer men kan i framtiden tänkas leverera även målspardata från en sensorlokal målföljare. Mätningen indelas i lobar där varje lob har viss varaktighet, riktning, räckvidd, bredd, pulsanal, PRF (pulsrepetitionsfrekvens) och toppeffekt. Alla dessa faktorer påverkar genom radarekvationen möjligheten att upptäcka målet. Vid instansiering ansätts också antennapertur samt lobomfång, det vill säga huruvida antennen är riktad framåt, som hos jaktflygplan, eller åt sidorna, såsom hos 'early warning' system. Varje radaraktion har bestämd utsträckning i tiden och rummet. Antennen förutsätts vara enkanalig. Upptäcktsprocessen indelas i faserna alert och confirm. Varje mätning ackompanjeras av parametrar för noggrannhet, i bäring, elevation, avstånd och radiell hastighet. Avstånds- och hastighetsblinda fläckar liksom upptäcktssannolikhet modelleras, men klottermodellering är ännu ej utförd.

IRST-modell, komplex version.

Detaljerad modell för mekaniskt roterande IRST-sensor. Explicit schemalagda sensoraktioner med dataleverans var 20e millisekund. Modellen innehåller begränsningar beträffande synfält, sökhastighet, upptäcktsavstånd och processorhastighet.

IRST-modell, enkel version.

Ger endast sann azimuth-vinkel till målet. Modellerar ej klotter eller upptäcktssannolikhet.

Antalet möjliga sensorer per plattform är obegränsat.

Varje sensor arbetar i aktioner. Aktionerna beställs av en beslutsloop och schemaläggs av sensorn själv vid beställningstillfället. Efter utförandet av sensoraktionen levereras observationerna till den beställande beslutsloopen som - på plattformsnivå - fortsätter bearbetningen av observationerna med målföljning med mera. Antalet möjliga observationer per sensoraktion är obegränsat.

förhandling
beställning
schemaläggning
väntan
aktion
dataleverans

Plattformar

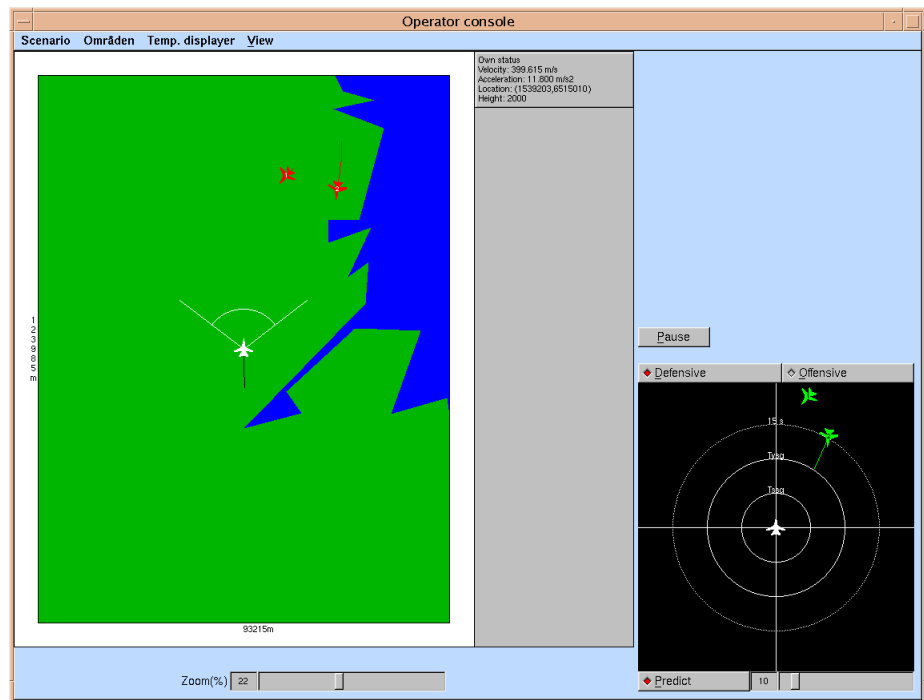
Plattformmodellerna kan betraktas såsom sensorförsedda mål, se Målmodeller.

Varje plattform kan där skicka målspardata till godtycklig annan plattform t ex för målsparfusion.

Varje plattform kan förekomma i flera scenarier.

Antalet möjliga plattformar per scenario är obegränsat.

Dataassociation	En Nearest Neighbour-algoritm har implementerats. Enbart observationer i ett område runt den predikterade tidsuppdateringen för varje målspar tas med i associationsbedömningen. Från dataassociationsprocessen leveras dels en lista med associationer, dvs ihop-parade målspar och observationer och dels en lista med oassocierade observationer. Den senare leder till uppstart av nytt målspar. Detta skall utvecklas till en mera realistisk målfångningsprocedur.
Målföljning	En IMM-baserad (Interactive Multiple Model) målföljialgoritm är implementerad, baserad på tre parallella Kalman-filer, ett Singer-filer, ett generellt målföljefiler och en rakbanemodell. Modellerna sammanvägs för dataleverans.
Målsparfusion	För målsparfusion används metoden Covariance Intersection. Målsparfusion kan göras då målspardata från beslutsloopar på olika plattformar skall utnyttjas.
Målmodeller	Varje mål har en flygbana eller motsvarande med brytpunkter i vilken den manövrerar med transversell och longitudinell acceleration som definieras för varje måltyp. Även maximal flyghöjd och hastighet definieras per måltyp. Antalet möjliga mål per scenario är obegränsat. Varje mål kan förekomma i flera scenarier. Varje mål kan observeras från flera plattformar och till och med från flera beslutsloopar i samma plattform.
Målsparshantering	Varje målspar "ägs" av en beslutsloop. Om beslutsloopen är ett NBL-barn skickas målspardata till NBL-loopen. Om flera beslutsloopar har målsparrepresentation för ett visst mål finns det ett målspar per beslutsloop i denna plattform. Antalet möjliga målspar per beslutsloop är obegränsat.
Temporaldisplay (Hotdisplay)	Temporaldisplayen [26] är en hotvisande display som utvecklats inom projektet. Temporaldisplayen drivs av data från datafusionsnoden. Se figur 7.1. Varje ring representerar en hotzon; objekt inom en sådan kan utföra viss aktion. Egen plattform inritas i mitten. Även egna handlingsmöjligheter kan illustreras, varvid hot och egna möjligheter kan jämföras. I motsats till en spatialt orienterad display visas inte spatialt riktiga avstånd. Det unika med displayen är att alla hot syns samtidigt på skärmen; i spatiella displayer kan hoten ibland visas i skjutzoner, men i så fall enbart för ett mål i taget. Utvärderingar visar att "killrate" ökar i jaktstrid [13]. Temporaldisplayen och en motsvarande spatialt orienterad display visar omvärlden såsom den uppfattas från en viss plattform. Flera plattformar kan delta i scenariot, och varje plattform kan förses med ett eller flera operatörsinterface som innehåller dessa displayer.



Figur 7.1. Temporal- eller hot-displayen visar hot och handlingsmöjligheter för egen plattform. De spatiella avstånden, se vänstra displayen, avspeglar inte de riktiga hoten, se hot-display till höger. Högra hotet är farligare än det vänstra.

Sensorstyrning

Sensorstyrning kan förekomma på flera nivåer. Här avses styrning av sensorerna inom en respektive flera plattformar i syfte att uppnå bästa resurstilldelning såsom det uttrycks i befintliga beslutsloopar.

Sensorstyrningen kan vara sensorbaserad eller beslutsloopsbaserad. I det senare fallet görs den inom varje varv av varje beslutsloop. Några moment i beslutsloopsbaserad sensorstyrning är val av sensor, schemaläggning av mätning/aktion i denna sensor samt sättning av parametrar vid mätningstillfället. Exempel 1: en målföljningsloop väljer efter varje uppdatering att utnyttja ESA-radarn eller att lita enbart till IRST-sensorn. Exempel 2: Ett uppdrag att följa visst mål kan överlätas på endera av två plattformar beroende på deras situation, mätkvalitet och sensorbelastning.

För val av sensor finns en algoritm utvecklad som tar hänsyn till förväntad målföljekvalitet [20] och sensorbelastning. Dessa värden tas in från olika sensorer och används för att räkna fram ett godhetstal för varje sensor. Processen kan liknas vid en förhandling mellan beställare/beslutsloop och sensor. Då sensor valts genereras en beställning om sensoraktion som översänds till sensorn. I denna anges önskemål beträffande nästa sensoraktion för den aktuella beslutsloopen. Beställningen tas emot av sensorn som schemalägger aktionen. Detta går till så att den försöker hitta ett ledigt tidsintervall som ligger så nära den önskade tiden som möjligt med hänsyn till redan schemalagda sensoraktioner. Gamla schemaläggningar kan ändras om den nya aktionen har högre prioritet än en redan schemalagd aktion.

ESA-radarn, som medger elektronisk styrning, kräver en flexibel och kraftfull schemaläggningsmekanism för effektivt utnyttjande.

Operatörsgränssnitt

Operatörsinterface finns för ett simulerat flygplan. Interfacet visar två displayer varav det ena är en spatialt orienterad karta med inritade upptäckta mål och det andra är en hot- eller temporaldisplay, se detta avsnitt. Antalet möjliga operatörer per plattform är obegränsat.

Scenariohantering (GUI)

Alla objektens rörelsebanor specificeras och ritas vid scenariogenereringen. Scenarier kan sparas och återhämtas. Varje mål och plattform har viss predefinierad typ som kan sparas och återhämtas. Varje predefinierad plattform har viss uppsättning sensorer. Nya mål- och plattformstyper kan skapas interaktivt. Under exekvering kan målens och plattformarnas sanna lägen samt observationer och målspar visas, på det sätt som de uppfattas från någon plattform.

Programmeringsmetodik

Programvaran är skriven under UNIX i C++. Strikt objektorienterad programmeringsmetodik har tillämpats. För versionshantering har CVS använts.

Programvaran har till stor del körstatus, dvs skulle i princip kunna användas för drift. Anpassningar måste dock göras till reella plattformar och mål, och kvaliteten på datahanteringen måste ses över. De delar som har simuleringsstatus är i första hand sensormodeller och scenariohanterare.

7.1.2 Samverkan mellan Datafusionsnoder

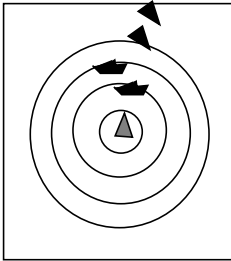
Datafusionsnoden var ursprungligen avsedd att studera sensorverksamheten ombord på enstaka plattform. Tack vare sin generiska struktur kan den efter vissa utvidgningar också användas för att realisera samverkan mellan flera plattformar, inklusive samverkan mellan operatörer i olika plattformar. I avsnitt 6.4 visas hur ramverket för denna kommunikation kan se ut.

7.2 Illustrativt scenario

Datafusionsnod

Den tjänst som flygföraren har beställt innebär att en beslutsloop/task skapas och börjar arbeta. Detta sker i flygplanets datafusionsnod. Eftersom flygplanets egen radarsensor ej kan användas görs en uppslagning i ett sensorregister med avseende på typ, täckningsområde och kapacitet. Registret svarar med referenser till tillgängliga plattformar, dvs den markstående giraffradarn och korvettens radar. Dessa stationer används redan för andra uppdrag, därför undersöks om eventuell tidsdelning är möjlig, och i så fall om det finns oanvända tidsluckor i dem. Detta visar sig vara fallet. Markradarns datafusionsnod tar emot beställningen, och en ny beslutsloop för sökningen genereras. Denna ansöker om en tidslucka för en första sökning, och därefter schemaläggs operationen. Vid operationen genereras ett antal observationer, vilka var och en leder till generering av en målfångningsfunktion och efterföljande generering av målspar. Därefter sker sekventiellt nya intermittenta schemaläggningar, operationer och målsparsuppdateringar.

Temporaldisplay



Efter varje uppdateringsomgång översänds en sekvens av målspar till det patrullerande flygplanets beslutsloop. Denna sammanställer informationen och överför den till två av flygförarens displayer, den taktiska displayen och temporaldisplayen. Den senare visar vilka fientliga aktioner som de ankommande flygplanen kan komma att utföra, samt en prediktion för tidpunkten för när dessa aktioner kan utföras. Den visar också vilka egna aktioner som flygföraren kan vidta, till exempel fly eller använda vapen, samt de tidsintervall under vilka dessa aktioner bör vidtas för att vara optimala.

Även datafusionsnoden på fartyget startar en beslutsloop för sökning i det aktuella området. Denna erhåller kontinuerligt målsparsinformation från den markbaserade giraffen. När den fartygsburna radarn börjar få observationer från målen utförs målföljning även på denna plattform. Inledningsvis har denna information större osäkerhet än den målsparsinformation som giraffen åstadkommer, men efter cirka 30 minuter råder det omvända förhållandet. Då översänds istället fartygets målinformation till giraffen, stridsledningscentralen och flygplanet.

7.3 Applikationsanpassat kommunikationsramverk

Implementation av modeller som kan rymmas inom den beskrivna lagerstrukturen kräver ett ramverk för utveckling av mjukvara. Detta indikeras i figur 6.1. En del i ett sådant ramverk har implementerats, vilket stödjer byggande av kopplingsbara komponenter [25]. Här ges en kort genomgång av funktionaliteten i detta ramverk. Därefter beskrivs en utbyggnad som tillför en indexeringsfunktion av komponenterna i ramverket, som därmed kan användas för att bygga organisationsmodeller som kan fungera i lagerstrukturen beskriven i figur 6.2.

7.3.1 Implementerad funktionalitet

Ramverket är ett klassbibliotek skrivet i C++ ovanpå mellanvaran CORBA, och inriktar sig på tre typer av funktionalitet: datatyper, kopplingskomponenter och register.

Datatyper

Den grundläggande delen är en beskrivning av en generell dataström, samt en metod för att översätta dataobjekt till detta format. Denna dataström innehåller viss metadata, data som beskriver dataströmmens format och innehåll, för att möjliggöra rekonstruktion av dataobjekt på mottagarsidan utan tidigare kunskap om dataformat.

Syftet är att en sändare av data skall kunna översätta sina dataobjekt (radarobservationer, kartor, kommandon) till en dataström som lämpar sig för sändning över ett nätverk, och att en mottagare skall kunna läsa in denna dataström och kunna rekonstruera dataobjekten igen.

Kopplingskomponenter

En kopplingskomponent används för mottagande eller sändande av generella dataströmmar. Det är också möjligt att koppla samman komponenter

parvis, detta sammankopplade par sköter då sändande och mottagande automatiskt. Kopplingskomponenten finns i tre utföranden, sändare, mottagare och en publiceringskomponent.

Sändaren och mottagaren fungerar ihop, en sändare kan sända data till en eller flera mottagare, och en mottagare kan ta emot data från en eller flera sändare. Den som är mottagare tar emot data från sändaren på ett passivt vis.

Publiceringskomponenten sänder också data, men den som vill ha åtkomst till dessa data måste aktivt hämta dem. För detta krävs ingen speciell mottagningskomponent.

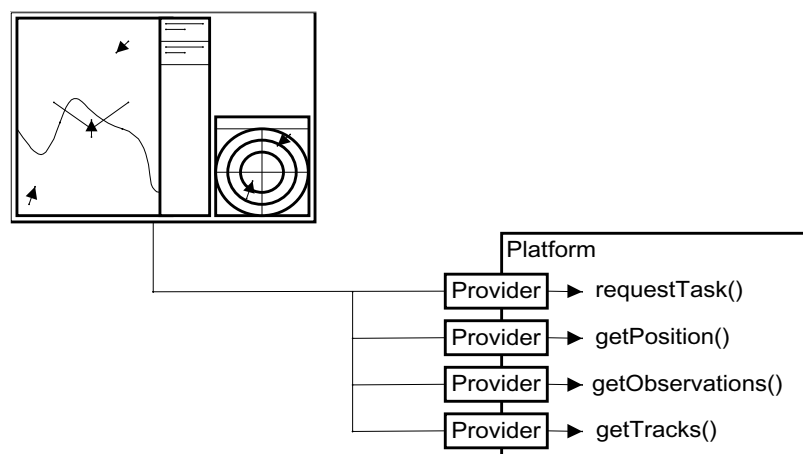
En plattform eller en sensor som är utrustad med kopplingskomponenter kan kopplas samman med andra plattformar och sensorer som är utrustade med kopplingskomponenter.

Register

Den sista delen är ett register där kopplingskomponenter kan registreras. Registreringen består i att ett namn associeras till varje komponent. Den part i en kommunikation som vill sända eller mottaga data gör en uppslagning i registret för att finna den kopplingskomponent som är relevant, och utför sedan kopplingen.

Det existerande ramverket möjliggör sammankoppling mellan olika parter i ett nätverk, med en metod som är oberoende av det underliggande nätverket. Det har tidigare beskrivits hur denna metod är mer flexibel än ett direkt användande av en underliggande nätverksteknik, t.ex. CORBA [25].

Detta ramverk används för att fästa kopplingskomponenter till olika funktioner i datafusionsnoden. Detta har gjort det möjligt att koppla samman flera datafusionsnoder med varandra. Det har också använts till att koppla operatörsdisplayer till de olika datafusionsnoderna, se figur 7.2 nedan.



Figur 7.2. Operatörsdisplay kopplad till funktioner i plattformsnod via kopplingskomponenten Provider, en publiceringskomponent. Operatörsdisplayen består av en spatial display (kartbild till vänster) och en temporaldisplay (cirkelformad till höger).

Ramverket beskriver ingenting i fråga om tjänster eller andra funktioner på högre nivå, dessa måste implementeras i termer av de komponenter som ramverket erbjuder. För att förenkla utvecklingen av tjänster och modeller för nätverkssamarbete, så bör ytterligare funktionalitet läggas till.

7.3.2 Planerad funktionalitet

Attribut

Till den existerande designen tillkommer en attributdefinition. Attributen består av en värdemängd kopplad till ett namn. Värde-mängden skall kunna representera ett enskilt värde, en lista av värden, ett intervall av värden eller en lista av intervall av värden. Ett attribut är alltså en dynamisk variabel som kan innehålla olika värden.

namn
värde-mängd

Attributen kan användas för att beskriva egenskaper hos olika objekt. Om man vill säga att en sensor är av en viss typ, kan man skapa attributet "typ" och tilldela det värdet "radar". Detta blir sedan användbart då man vill katalogisera och söka efter sensorer av en viss typ.

Varje enskilt attribut kan grupperas med andra attribut som då bildar attributmängder. Attributmängden skall kunna kombineras med andra attributmängder med vanliga mängdoperationer som union eller snitt.

Antag att en sensor har attributen "typ = radar" och "räckvidd = 20km". Dessa bildar en attributmängd. En sensoranvändare kan uttrycka sitt sensorbehov i form av en motsvarande attributmängd: "typ = radar" och "räckvidd > 10km". Sensorn är lämplig för användaren endast om ingen konflikt råder mellan sensorattributen och användarens attributmängd. I detta exempel passar sensorn för användarens behov.

Samtliga delar av ramverket byggs ut för att kunna hantera attributmängder. Attributmängderna används som en generell metod för att beskriva och katalogisera olika objekt. Forskning kring ontologier utnyttjas [10].

Datatyper

Den generella dataströmmen förses med en attributmängd som beskriver dess innehåll, attributmängden används som metadata. Metoden för konverterande av dataobjekt till dataströmmar förses med funktioner för att hantera attributmängder.

Kopplingskomponenter

Varje kopplingskomponent utökas till att kunna bindas till en attributmängd, samt till att kunna tillfrågas om sin attributmängd. De kopplingskomponenter som tillåter parvis koppling (sändare och mottagare) ges även en kopplingsfunktion som tar hänsyn till huruvida komponenternas attributmängder är kompatibla, via attributmängdernas mängdoperationer. Samtliga kopplingskomponenter byggs ut med sändningsfunktioner som tar hänsyn till huruvida den sända dataströmmens attributmängd är kompatibel med kopplingskomponenten.

Register

Det existerande registret utökas till att lagra en attributmängd per registrerad komponent. Registret skall tillåta sökningar på godtyckliga attribut och attributmängder. Detta gör att registret fungerar som en riktig databas.

Registret utökas också till att tillåta registrering av referenser till andra register, mängden av samtliga till varandra refererande register bildar då ett

nätverk av register, sammanlagt en distribuerad databas. Detta leder till en skalbarhet av registerfunktionen som kan utnyttjas till att t.ex. modellera en katalogfunktion som täcker en stor geografisk yta.

7.4 Illustrativt scenario

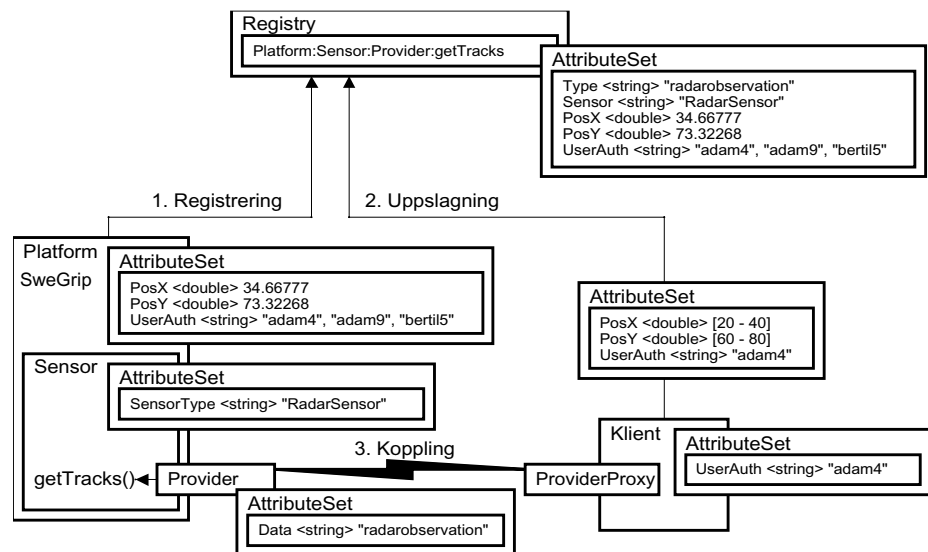
Följande visar hur det utbyggda ramverket kan användas i en applikationsprogramvara.

I scenariot deltar flygplanet *SweGrip*, med sensorn *Sensor* ombord. Flygplanets position beskrivs av dess attribut *PosX* och *PosY*. Flygplanet bär också på ett *UserAuth*-attribut som beskriver vilka användare som är behöriga att hämta data från det. *Sensor* är av en typ som beskrivs av dess attribut *SensorType*. *Sensor* exponerar också en kopplingskomponent vid namn *funktionA*. Komponentens serverar data av en typ som bestäms av dess attribut *Data*. Som motpart i scenariot finns klienten *Klient*. *Klients* användarbehörighet bestäms av attributet *UserAuth*.

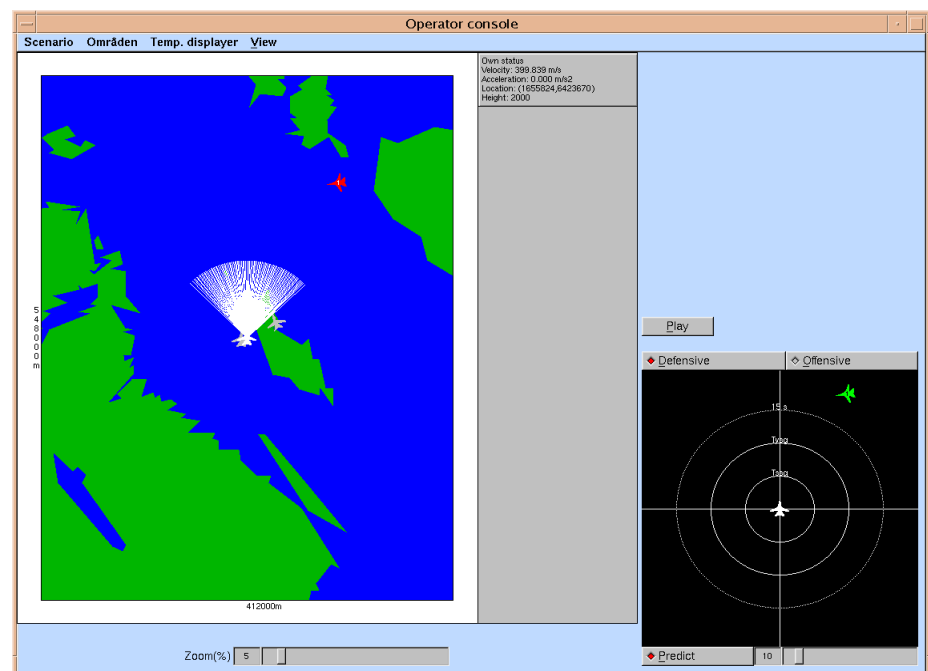
Figur 6.3 beskriver i detalj hur scenariot utspelas. *Sensor* registrerar sig (1), blir upplagen av *Klient* (2), och koppling genomförs (3).

Detta scenario innehåller geografisk sökning och en modell av hur autentisering kan hanteras. Det modifierade ramverket är ett verktyg för att implementera godtyckliga modeller för nätverkssamarbete. Vad som inte tillhandahålls av ramverket är organisation, metod och betydelse av de enskilda attributen och komponenterna.

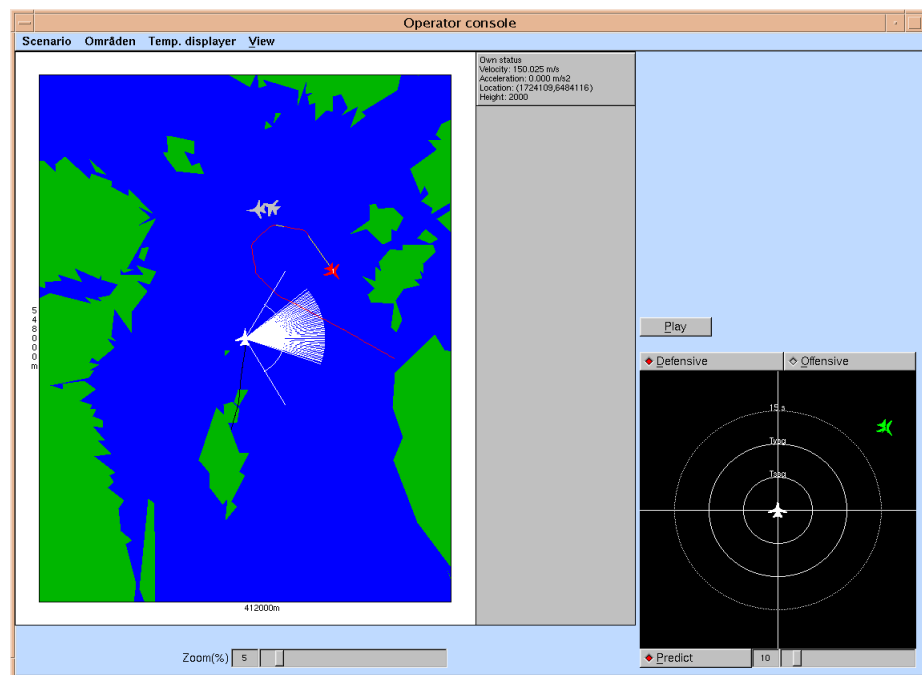
Implementationen illustreras i ett scenario i tre bilder, figur 7.4-7.6. I scenariot finns ett spaningsflygplan med sidtittande spaningsradar, två egna jaktflygplan med nosradar samt ett rödfärgat målflygplan. Spaningsflyget och jakten utbyter radardata. Varje figur visar omvärldsbilden för en operatör. I figur 7.4 och 7.6 visas jaktens omvärld, i 7.5 spaningsflygets omvärld. I 7.4 använder enbart jakten sin radar, men översänder den till spaningsflyget. I figur 7.5-6 har enbart spaningsflyget radartäckning mot målet, men översänder informationen till jakten. Den bild som visas i figurerna 7.5-6 har jaktföraren beställt av spaningsflyget. Jaktföraren ser då målet trots att han ej har egen radartäckning mot det. I nedre högra hörnet visas temporaldisplayen. Varje flygplan har en datafusionsnod och olika flygplan samverkar med varandra via sina datafusionsnoder.



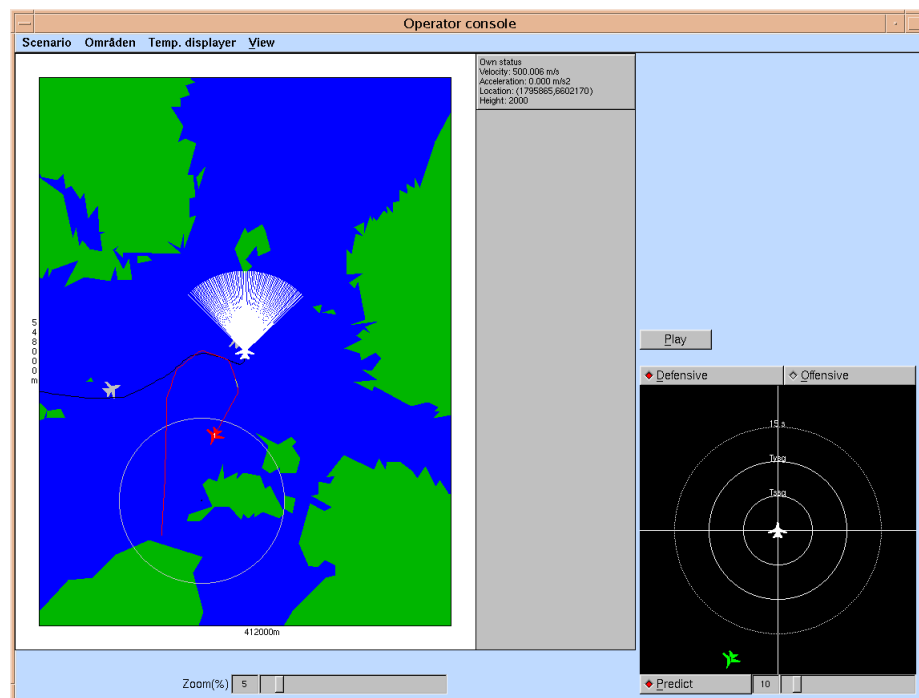
Figur 7.3: (1) Sensor ombord på plattformen SweGrip börjar existera i nätverket. Den kopplingskomponent som Sensor bär på registreras i registret Registry.
(2) Klient (en annan sensor, en plattformsnod, en operatör) gör en uppslagning i registret. I uppslagningsoperationen bifogas UserAuth, samt intervallattribut för PosX och PosY, vilka beskriver en geografisk area. Attributmängden som bifogats uppslagningen matchas mot innehållet i registret som returnerar en referens till en kopplingskomponent.
(3) Klient har nu en referens till en kopplingskomponent och genomför kopplingen. Klient kan nu hämta data från funktionen getTracks i Sensor, ombord på SweGrip.



Figur 7.4. En jaktrote får uppdrag att avvisa anflygande mål (rödmålet). Enbart roteledarens radar är tänd. Displayerna visar jaktens omvärld. Informationen översänds till spaningsflyget.



Figur 7.5. Jakten har avvisast inkräktaren och återvänt mot eget territorium. Enbart spaningsflygets radar är tänd. Displayerna visar spaningsflygets omvärld. Informationen översänds till jakten.



Figur 7.6. Jakten är på väg hemåt. Enbart spaningsflygets radar täcker målet. Informationen översänds till jakten. Displayerna visar jaktens omvärld.

8. Internationell utblick

Agentbaserat stöd för koalitions- operationer

USA:s DoD-ägda forskningsinstitut DARPA finansierar flera program med anknytning till sensornät och datafusion. Inom området 'Intelligent Agents' finns programmet Control of Agent-Based Systems (CoABS). Programmet fokuserar på att möjliggöra snabb integration av nya agent-system och arvssystem i syfte att öka interoperabiliteten och stötta användarnas situationsuppfattning och beslutsfattande. Programmet har utvecklat ett mellanvarulager (se kapitel 2 och 7), en s.k. "CoABS grid", som ska utgöra en infrastruktur för integrering av heterogena agentsystem. Viktiga forskningshypoteser är bland annat att

- agenter utgör en lämplig metafor för att hantera komplexiteten hos verkliga militära operationer
- ett agentbaserat C2ramverk kan stötta flexibla och robusta koalitionsoperationer
- mjukvaruagenter kan möjliggöra interoperabilitet mellan arvssystem och mellan inkompatibla system

Över huvud taget anses agentteknik ge den modularitet och abstraktion som är nödvändig för att hantera stora och komplexa problem och att det är ett bra paradigm för att bygga komplex mjukvara.

Agenterna ska kunna automatisera viktiga C2-funktioner för strategisk planering, såsom informationsinsamling, uppdragsplanering och insatsövervakning. En demonstration ska visa hur internationella styrkor med olika förband och ledningssystem kan kommunicera och samverka i en internationell koalitionsoperation [2].

Amerikanska erfarenheter visar att det är mycket svårt att "hålla koll" på vilka styrkor som deltar i olika operationer, deras förmågor, etc, eftersom detta förändras snabbt och inte finns samlat tillgänglig någonstans i systemet. Därför har det amerikanska flygvapnets forskningsorgan AFRL egen verksamhet inom agentbaserad forskning för koalitionsoperationer. Syftet är att utveckla en generell mall för att lagra information om alla enskilda deltagare i koalitionsoperationer. Denna mall kallas en 'Force Template' (FT) [16] och ska underlätta för olika koalitionsparter att ta del av varandras information. All uppdragsnödvändig information om en entitet ska potentiellt finnas tillgänglig, inklusive dess identitet, vad den vill, vad den kan erbjuda och hur den avser att agera. Ett minimum av information som varje FT måste innehålla är en definition av vilken information den behöver, vilken information den kan tillhandahålla och vilka restriktioner som finns förknippade med informationen (t ex säkerhetsrestriktioner). Alla entiteter som är representerade i operationen ska också representeras av en FT. Detta program fokuserar alltså på innehållet i vad som ska finnas tillgängligt för parterna och inte på tekniken för interaktion.

Självkonfigurerande kopplingar

Samverkan i nätverk mellan rörliga plattformar där deltagare, roller och uppgifter snabbt kan förändras, kräver att deltagarna dynamiskt kan anslutas till och från ett redan driftsatt och operativt nät, dvs att systemen är

självkonfigurerande. Likaså är sådan självkonfigureringsförmåga nödvändig för att skapa interoperabilitet mellan stora systemkomponenter som skrivits i olika programvara. Det kan gälla till exempel fartyg, radarsensorer och vapensystem byggda av olika leverantörer. DARPA har ett program för detta som heter DASADA, Dynamic Assembly for Systems Adaptability, Dependability and Assurance [5].

Uppgifts- och rolldistribution

Inom forskningsområdet 'Software Infrastructure' finns programmet ANT, Autonomous Negotiating Teams [11]. Syftet med detta program är att möjliggöra utvecklandet av informationssystem som utan någon central planering effektivt kan fördela olika uppgifter bland de tillgängliga resurserna, dvs problemet som studeras är distribuerad uppgifts- och resursallokering. Fördelningen ska kunna ske helt autonomt och i realtid. Många uppgifter kräver dessutom att flera resurser samverkar i olika roller kring en lösning och även denna rollfördelning ska lösas. Tekniken som studeras är s.k. förhandlingsmekanismer (se avsnittet om sensorstyrning sidan 27), som anses vara en tids- och kostnadseffektiv lösning. Alla resurser betraktas i detta sammanhang som likar, "peers" [3]. Alltså medges inga speciallösningar för olika typer av resurser, utan alla deltar i förhandlingen på lika villkor. Förhandlingarna genomgår ett antal steg där en medverkande part måste ta reda på vilka andra parter som finns tillgängliga för samverkan kring att lösa en given uppgift, vilka förmågor dessa har, vilka uppgifter de redan utför, hur distribution av information ska ske, etc. Därvidlag ska parterna kunna ta hänsyn till osäkerheter, uppgiftsprioriteter, olika möjliga handlingsalternativ, kunna kompromissa mellan olika mål och avgöra om nya omständigheter står i konflikt med gamla planer. Slutligen måste själva förhandlingsprocessen kunna garanteras att ta en begränsad tid för att möjliggöra realtidsprestanda. Speciellt utpekade tillämpningsområden är telekrigsinsatser och logistikplanering.

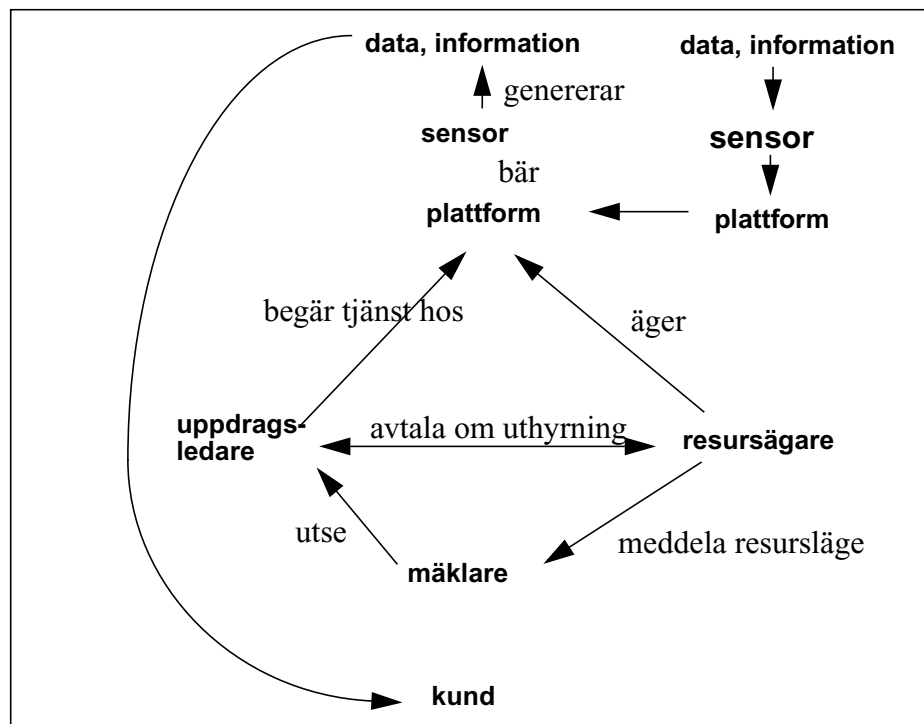
9. Svensk utblick

Här presenteras ett förslag till resursallokering som genererats av projektet och en grupp radarspecialister på FOI i Linköping. Därefter diskuteras två industriella arbeten kring sensornät.

9.1 Resursallokering

Mäkling

I en hierarkisk organisation har varje förband sina egna resurser. Möjligheten för ett förband att få "låna" en spaningsresurs omges då av viss administrativt 'overhead'. Om en spaningsresurs kan utföra flera funktioner simultant, blir detta problem mera fokuserat: varje användare kan där 'låna' bara en viss del av resursen. Detta gäller bland annat elektriskt styrda radarapparater ESA. Behovet av en ordning där "alla" användare kan låna spaningsresurser blir då mera uttalat. Vi har utvecklat en metod för att organisera sådan spridd resursanvändning. Detta medger också en platt organisation. metoden illustreras i figur 9.1.



Figur 9.1. Princip för resursallokering i platt organisation.

Principen är följande:

En godtycklig användare ('kund') begär hos en mäklare tillgång till spaningsresurser för visst ändamål. Mäklaren vet vilka resurser som finns, och känner till deras status, belägenhet med mera. Denna information har han/hon sedan tidigare fått av "resursägarna", det vill säga de förband där dessa resurser hör hemma. Mäklaren utser en uppdagsledare för det spa-

ningsändamål som kunden har begärt. Uppdragsledaren upprättar direktkontakt med resursägaren, för att avtala om villkor för användningen ('pris', tillstånd, tid, status). Uppdragsledaren utnyttjar sitt användningstillstånd för att hos plattformen begära de tjänster han behöver för att uppdraget skall utföras. Eventuellt kopplas en plattformsburen operatör in i detta arbete. Tjänsten tar i anspråk vissa resurser (sensorer) hos plattformen. Insamlad data processas (datafusion, lägesbildframställning) och översänds till kund. Uppdragsledaren kan anlita flera plattformar för späningsjämnaren.

Rollerna i detta schema återfinns i andra organisationer. Uppdragsledare svarar mot projektledare och resursägarna mot linjeorganisationen.

För varje roll finns ett antal beslutshjälpmedel, databaser med mera. Förslaget har genererats inom FOI.

9.2 Sensornät

WASP

WASP [27] står för Wide Area Situation Picture, *gemensam lägesbild* (GLB), och omfattar liksom DISSY ett förslag på metodik för att uppnå densamma. WASP har utvecklats av Saab Technology Systems AB, Järfälla. Kärnan är ett platt nätverk av särskilda korreleringsenheter; WASP Correlation Deamons (WCD). Dessa är kopplade mot datakällor som producerar målspar. Målsparskällorna kan vara autonoma, målföljande sensorer och sensorsystem, men också fusionsnoder och kompletta nätverk.

Varje WCD är också kopplat till ett och samma presentationslager, vilket utgör gränssnitt mot användare. Ur detta lager kan en godtycklig del av GLB extraheras. Själva lägesbilden ligger dock distribuerad i WCD:erna. Presentationslagret är endast till för att möjliggöra extraktion av önskad information.

Uppdatering av GLB sker av varje WCD som har unik eller kvalitetsmässigt överlägsen data för ett visst mål. Uppstår då inkonsistens i GLB korrelerar berörda WCD:er sin del av GLB med data från denna WCD. I annat fall sker ingen korrelering. Korrelering innebär att jämföra "sitt eget" målspar med det från den uppdaterande WCD:n, och avgöra huruvida de här rör sig från samma mål eller ej. Skillnader i noggrannhet är t ex inget skäl för korrelering, endast inkonsistens. Att alla WCD har samma noggrannhet för ett visst målspar är nämligen inte nödvändigt, eftersom den totala lägesbilden ej påverkas av detta.

För att kontrollera kommunikation av data används sk subject based addressing. Det innebär att data endast läses om den är relevant för mottagaren. Varje målsparskälla prenumererar på vissa data, t ex data som rör objekt i ett visst område. Prenumerationen kan existera på olika nivåer av noggrannhet/bandbredd. Genom sk multicasting kan samma information sändas ut till flera prenumeranter.

I WASP används en bayesiansk modell för att korrelera målspar. Korreleringsalgoritmen och användandet av multicasting och subject based addressing gör WASP fullt skalbar med avseende på antal användare och målsparskällor.

DISSY

DISSY, DIstributed SensorSystem [12], är ett förslag till platt arkitektur för sensornät som introducerats vid EMW. Sensorerna i nätet kommunicerar med varandra utan inblandning av någon central fusions- eller styrenhet. Istället används en platt sensorarkitektur, vilket bidrar till robusthet (på grund av decentraliserad funktion) och flexibilitet. Varje sensor "ser" ett antal observerade objekt, och vet vilka andra sensorer som också "ser" varje enskilt objekt. Varje objekt "ägs" av en sensor, och vilken sensor som är ägare kommer sensorerna överens om sinsemellan. Hur detta ska gå till definieras dock ej i [12].

Ägaren till varje målspar upprätthåller och distribuerar målspar-data till övriga sensorer. Dessa kan skicka in observationer men ej målspar-data - de senare distribueras endast av ägaren. Varje sensor kan ta emot data från övriga sensorer och behandla dem på det sätt som är bäst för den (t ex fusionera globala målspar och lokala observationer). Genom att undvika att flera noder distribuerar målspar för varje enskilt mål uppnås en rad fördelar: (1) Fusionsalgoritmerna behöver ej (felaktigt) förutsätta att processbruset i målsparsinformationen från olika avsändare är oberoende. (2) Risken för självsvängning vid återmatning upphör. (3) Målsparsinformationen är mycket bandbreddskrävande.

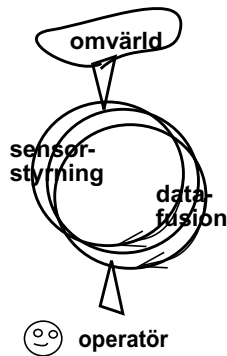
För att realisera detta utförs i varje sensor tre funktioner:

A. Hitta alla sensorer som "ser" samma saker som sensorn själv och 'avtala' om datautbyte med dem. B. Anpassa och tolka erhållna data på ett sätt som passar sensorn bäst. C. Förhandla med övriga sensorer om vem som ska bära ansvaret för målföljning och andra uppgifter för varje mål, dvs 'äga' målet.

Enligt [12] bör sensorsamverkan ej planeras om den bygger på att sensorerna beordras att se ett objekt - det kan hämma flexibiliteten. Varje sensor skall istället kunna användas till sådant som den råkar observera. Denna princip bör dock ställas mot krav på styrning och önskemål om sensortjänster som dels plattformsoperatörer och dels icke plattformsburna användare kan ha.

10. Slutsatser

10.1 Idéer och praktik



Den utvecklade programvaran realiserar en modell som bygger på idéer från olika håll:

- Tanken att datafusion och sensorstyrning hänger ihop återfinns bland annat i [7], kapitel 16.
- Datafusion och sensorstyrning är två moment som ingår i samma iterativa process [4].
- En teori för situationsuppfattningen ges i [6]. Denna teori antyder en iterativ process, som vi har kallat för beslutsloop. Flera av de i denna loop förekommande momenten återfinns också i datafusionsteorin, se till exempel kapitel 1 av [7]. Idén att objektivera och mekanisera beslutsloopen är vår egen, men stöds av kognitionsvetenskaplig teoribildning.
- Idén att processa många beslutsloopar simultant har genererats i diskussioner med radarspecialister på FOI i Linköping och idén har sedan applicerats på operativsystemtekniken.

En viktig iakttagelse är att kombinationen av modellgenerering, konstruktion, implementering och utvärdering ger helt andra lärdomar, insikter och erfarenheter än enbart litteraturstudier.

10.2 Utvärdering

Ett implementeringsarbete som baseras på tidigare oprövade begrepp och konstruktioner har två huvudeffekter. Dels påvisar det den eventuella styrkan hos dessa konstruktioner, dels avslöjar det obönhörligen eventuella luckor och brister som finns i konstruktionerna. På detta sätt utgör det beskrivna implementeringsarbetet en utvärdering av användbarheten hos de använda begreppen och konstruktionerna. De modelleringsansatser som har utvärderats på detta sätt omfattar

beslutsloop
multi-agent-teknik
operativsystem

- objektifiering av beslutsloopen/OODA-loopen/task
- applicering av multi-agent-teknik på tasks och plattformsapparater
- användningen av operativsystemteknik vid task-hanteringen.

Enligt [7], kapitel 26, bör utvärderingskriterier inom *datafusionsområdet* baseras på beslutsfattarens informationsbehov; den viktiga frågan är då hur mycket datafusionen bidrar till detta informationsbehov. Enligt [7], kapitel 16, faller området mjukvaruarkitekturer också inom ramen för *Systems Engineering and Implementation*. Utvärderingskriterier inom detta område har inte kunnat återfinnas i litteraturen. En sammanvägning av ovanstående leder till att följande frågor kan antas vara relevanta vid en utvärdering:

**Utvärderings-
kriterier**

- Hur mycket av plattformsoperatörens informationsbehov kan tillgodoses med den föreslagna konstruktionen?
- Hur väl kan interoperabilitetskrav och robusthetskrav tillgodoses?
- Hur väl används de givna resurserna (sensorer etc) i de aktuella scenarierna?
- Hur pålitlig är den presenterade informationen?
- Hur sårbar (eller robust) är den föreslagna konstruktionen?
- Hur lätt kan en given grupp av enskilda plattformar forma ett nätverk?
- Hur lätt kan individer (sensorer, plattformar) infogas i respektive utgå ur ett befintligt nätverk?
- Hur påverkas nätverket om en enstaka individ uppgraderas?

Dessa frågor kan inte ges ett komplett svar i denna rapport. De handlar i hög grad om operatörsbehov, robusthet, interoperabilitet och portabilitet. Alla dessa faktorer - utom operatörsbehoven - gynnas, enligt [7], kapitel 16, av en öppen systemarkitektur som kännetecknas av distribuerad databehandling, nivåbaserad programvaruarkitektur, realtidsdistribution, programmodularitet, applikationsoberoende, datakällsoberoende funktionalitet, flexibel rekonfigurering och skalbarhet. Den beskrivna konstruktionen har alla dessa kännetecken, och dessutom möjliggörs genom agenttekniken och beslutsloopsmekaniseringen en bred funktionalitet, utvidgningsbarhet, dynamik och underhållsvänlighet.

10.3 Simulering

Den föreslagna arkitekturen visar på några viktiga egenskaper hos ett plattform- och sensornätverk. Det beskrivna systemet är dock utvärderat enbart i laboratoriemiljö, vilket är en brist. I fortsättningen bör proven i denna laboratoriemiljö kompletteras med prov med nätverk av olika typer av reella (ärvda) plattformar. Den funktionalitet som då ingår i datafusionsnoden finns ofta - men enbart till viss del - redan implementerad i andra plattformsburna delsystem. Hur dessa befintliga system skall modifieras för att tillåta en ökad funktionalitet är redan föremål för industriellt utvecklingsarbete. Den i denna rapport beskrivna programvaran och datafusionsnoden kan därvid vara inspirationskälla för fortsatt arbete.

10.4 Kvalitetsvärdering

Anknytningen till internationella projekt utgör en kvalitetsmätare och ger god beredskap för internationellt samarbete. Den i detta system utnyttjade agent-tekniken är en nyckelteknologi i flera aktuella DARPA-program [11], [2], [8], [16].

11. Referenser

- [1] ASTTCDE (2002), *Workshop on Air&Sea Target Tracking & Classification in Distributed Environments*, In a CD (Compact Disc), Strömberg, (Editor), FOI, Linköping, Sweden, August 2002.
- [2] Allsopp, Beatement, Kirton, Bradshaw, Suri, Durfee, Knoblock, Tate, Thompson, *Multiagent Cooperation in International Coalitions*, in IEEE Intelligent Systems, pp 26-35, May/June 2002.
- [3] Barkai, D., *Peer-to-Peer Computing: Technologies for Sharing and Collaborating on the Net*, Intel Press, 2001.
- [4] Blackman, S., Popoli, R., Kapitel 15 *Sensor Management in Design and Analyses of Modern Tracking Systems*, pp 967-1068, Artech House, Boston, 1999.
- [5] DASADA (2002), *The Dynamic Assembly for System Adaptability, Dependability and Assurance*, DARPA program, In www.darpa.mil/ito/research/dasada/index.html.
- [6] Endsley, M., *Towards a Theory of Situation Awareness in Dynamic Systems*, Human Factors, 37(1), page 32-64, 1995.
- [7] Hall, D., Llinas, J., (Editors), *Handbook of Multisensor Data Fusion*, CRC Press, New York, 2001.
- [8] Hedberg, S., *Surveilling Intelligent Systems Research*, in IEEE Intelligent Systems, pp 4-6, May/June 2002.
- [9] Hollnagel, E., Cognition as control: A pragmatic approach to the modelling of joint Cognitive systems, Submitted to a Special issue of IEEE Transactions on Systems, Man, and Cybernetics A: Systems and Humans - "Model-Based Cognitive Engineering in Complex Systems", 2001.
- [10] Horney, T., *Design of an ontological knowledge structure for a query language for multiple data sources*, FOI-R--0498--SE, 2002.
- [11] IEEE Intelligent Systems, May/June 2002.
- [12] Kronhamn, T., *Sensor Inter-Net-Works*, In Proceedings of the Fourth International Conference on Information Fusion, Montreal, Canada, July, 2001.
- [13] Linde, L., Strömberg, D., Pettersson, G., Andersson, R., and Alfredson, J. (2001). *Evaluation of a temporal display for presentation of threats in air combat*. In R. Onken (Ed.), Proceedings of CSAPC'01 8th conference on cognitive science approaches to process control, Universität der Bundeswehr, Neubiberg, Germany. pp. 137-146. Also as FOI-S--0527--SE, September 2001.

- [14] Loborg, P., Andre-Jönsson, H., Strömberg, D., *Arkitekturbeskrivning för prototyp avseende distribuerbara, kopplade beräkningsmoduler*, Arbetspapper, FOI, 2001-05-11.
- [15] Morin, M., *Multimedia Representations of Distributed Tactical Operations*, Dissertation No 771, University of Linköping, September 2002.
- [16] Marmelstein, R., *A Blueprint for Coalitin Interaction within an Infosphere*, in IEEE Intelligent Systems, pp 36-41, May/June 2002.
- [17] Nilsson, P., Preliminär målsättning för formsvarsmaktens ledningssystem 2010, HKV, 2000.
- [18] Nilsson, P., Preliminär målsättning för systemdemonstratorerna 2005 och 2006, HKV, 2000.
- [19] Schmidt, D., Stal, M., Rohnert, H., Buschmann, F., *Pattern-oriented software architecture Volume 2 Patterns for concurrent and networked objects*, Wiley&Son, London, 2001.
- [20] Strömberg, D., Lantz, F., Hörling, P., *En Plattformbaserad Datafusionsnod - Delrapport 1*, (In Swedish), FOA-R--00-0176-706--SE, December 2000.
- [21] Strömberg, D., *Värdering av Systemarkitekturer*, (In Swedish), FOI Memo (3 sidor), Diariennr 01-3252, Oktober 2001.
- [22] Strömberg, D., *A platform-based data fusion and sensor management node*, In Proceedings of radar2002, IEE Conference Publication Number 490, pp 483-487, October 200
- [23] Strömberg, D., *Integration och Styrning av Sensorer i Nätverk*, (In Swedish), FOI-R--0237-SE, November 2001.
- [24] Strömberg, D., Lantz, F., Andersson, M., *On Platform-Based Sensor Management*, In Proceedings of the Fifth International Conference on Information Fusion, July 8-11 2002, Annapolis, Maryland, USA, pp 600-607, July 2002. Also in FOI-S--0529--S, July 2002.
- [25] Strömberg, D., Berg, T., *Statusrapport Juni 2002 - Samverkande Datafusionsnoder - Delprojekt inom LedsystT/FOI*, (In Swedish), FOI Memo (11 sidor), Diariennr 02-1531, June 2002.
- [26] Strömberg, D., Andersson, M., Pettersson, G., Forsgren, R., *Temporal displays* (In swedish). FOI-R--0576--SE, Sept 2002.
- [27] Wallenius K., *Use of Modern Information Technology: WASP - A Common View of the Situation*, Technet Europe, 21st AFCEA Europe Symposium & Exposition, Prague, the Czech Republic, 2000.
- [28] Weiss, G., *Multiagent Systems*, The MIT Press, London, 2000.