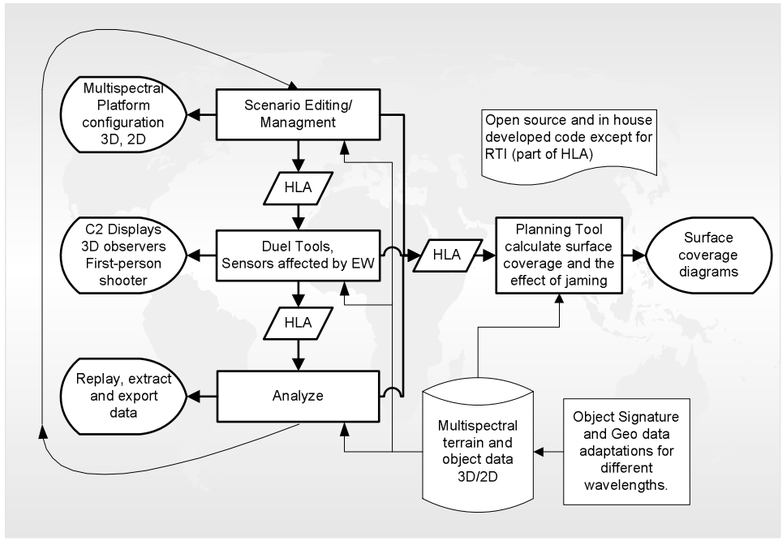
Systembeskrivning
LKS V2 teknikplattform

LARS TYDÉN, HANNA ANDERSSON, FREDRIK FORSLUND, LEIF FESTIN,
FREDRIK MÖRNESTEDT, SEBASTIAN OLSSON, MIKAEL PETERSSON, CHRISTER WIGREN



FOI är en huvudsakligen uppdragsfinansierad myndighet under Försvarsdepartementet. Kärnverksamheten är forskning, metod- och teknikutveckling till nytta för försvar och säkerhet. Organisationen har cirka 1250 anställda varav ungefär 900 är forskare. Detta gör organisationen till Sveriges största forskningsinstitut. FOI ger kunderna tillgång till ledande expertis inom ett stort antal tillämpningsområden såsom säkerhetspolitiska studier och analyser inom försvar och säkerhet, bedömning av olika typer av hot, system för ledning och hantering av kriser, skydd mot och hantering av farliga ämnen, IT-säkerhet och nya sensorers möjligheter.

FOI
Totalförsvarets forskningsinstitut
Ledningssystem
Box 1165
581 11 Linköping

Tel: 013-37 80 00
Fax: 013-37 81 00

www.foi.se

FOI-R--2275--SE Användarrapport **Ledningssystem**
ISSN 1650-1942 Maj 2007

Lars Tydén, Hanna Andersson, Fredrik Forslund, Leif Festin, Fredrik Mörnestedt,
Sebastian Olsson, Mikael Petersson, Christer Wigren

Systembeskrivning

LKS V2 teknikplattform

Utgivare FOI - Totalförsvarets forskningsinstitut Ledningssystem Box 1165 581 11 Linköping		**Rapportnummer, ISRN** FOI-R--2275--SE	**Klassificering** Användarrapport
		Forskningsområde 6. Telekrig och vilseledning	
		Månad, år Maj 2007	**Projektnummer** E7546
		Delområde 69 Breda projekt inom telekrig och vilseledning	
		Delområde 2	
Författare/redaktör Lars Tydén Hanna Andersson Fredrik Forslund Leif Festin Fredrik Mörnestedt Sebastian Olsson Mikael Petersson Christer Wigren		**Projektledare** Martin Castor **Godkänd av** Mikael Sjöman **Uppdragsgivare/kundbeteckning** FMV **Tekniskt och/eller vetenskapligt ansvarig** Lars Tydén	

Rapportens titel

Systembeskrivning LKS V2 teknikplattform

Sammanfattning (högst 200 ord)

Denna rapport beskriver den tekniska plattformen som Ledningskrigföringssimulatorn LKS använder. Plattformen använder HLA för distribuerade simuleringar och en lång rad med modeller ingår genom att merutnyttja modellresultat från arbeten gjorda inom andra FoT projekt. Utvecklingen av teknikplattformen för LKS har inneburit att en nätverksmodell med CNO, stabsverktyg för presentation av lägesinfo speciellt med avseende på telekrig och CNO samt C2 funktionalitet nyutvecklats.

Plattformen består av tre delar, en del för scenarioplanering och konfiguration, en del för den dynamiska duellen där duellen utspelas med simulerad tid samt en utvärderingsdel där dueller kan återuppspelas och loggade data analyseras. Designen är gjord så att olika plattformar (t ex stridsvagnar och fartyg) konfigureras med olika komponenter (t ex sensorer och vapen) och simuleras distribuerat över ett nätverk.

Med LKS finns möjligheten att sätta upp scenarier för att värdera några mot några i enskilt användande eller som värderingsduell där man kan bilda lag som kan duellera mot varandra.

Nyckelord

Simulering, Telekrig, CNO, HLA, Ledningskrigföring, Informationskrigföring, EWSim.

Övriga bibliografiska uppgifter	**Språk** Svenska
ISSN 1650-1942	**Antal sidor:** 57 s.
Distribution enligt missiv	**Pris: Enligt prislista**

Issuing organization
FOI – Swedish Defence Research Agency
Ledningssystem
Box 1165
581 11 Linköping

Report number, ISRN
FOI-R--2275--SE

Report type
User report

Programme Areas
6. Electronic Warfare

Month year
Maj 2007

Project no.
E7546

Subcategories
69 Interdisciplinary Projects regarding Electronic Warfare

Subcategories 2

Author/s (editor/s)
Lars Tydén
Mikael Petersson
Hanna Andersson
Christer Wigren
Fredrik Forslund
Leif Festin
Fredrik Mörnestedt
Sebastian Olsson

Project manager
Martin Castor

Approved by
Mikael Sjöman

Sponsoring agency
FMV

Scientifically and technically responsible
Lars Tydén

Report title (In translation)
Systemdescription C2WS

Abstract (not more than 200 words)
This report describes the technical platform used by the command and control warfare simulator (C2WS). The C2WS uses HLA for distributed simulations and a lot of models are included by reuse of results given from other R&D projects.

The C2WS consists of three parts, one for scenario planning and configuration, where models are configured, one for dynamic duels, and finally one assessment module, where the duels can be replayed and logged data analyzed

Keywords
Simulation, Electronic Warfare (EW), CNO, HLA, Federation, Communication, EWSim

Further bibliographic information

Language Swedish

ISSN 1650-1942

Pages 57 p.

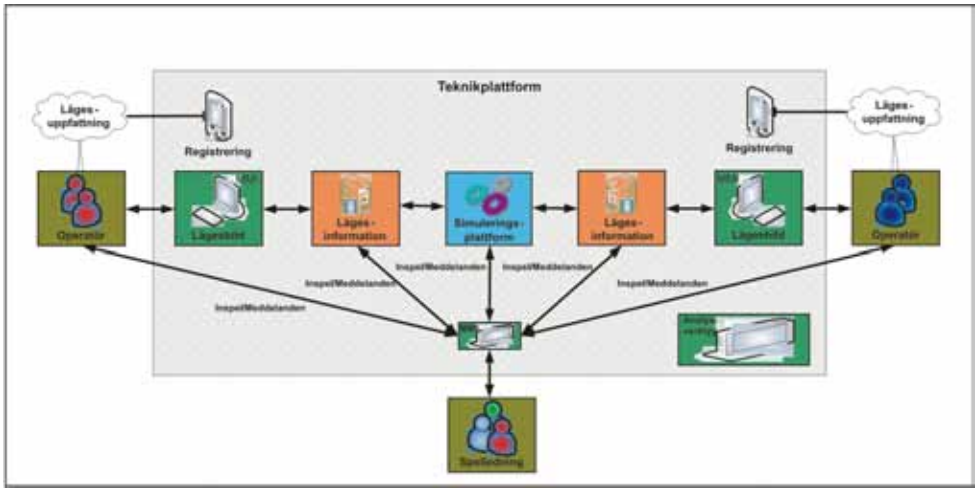
Price acc. to pricelist

Innehåll

1 Inledning

Denna rapport beskriver den tekniska plattformen som det FMV ledda demonstratorprojektet Ledningskrigföringssimulatorn (LKS) tagit fram i etapp ett av två. I LKS är ett av de mer övergripande syftena att påvisa effekter av ledningskrigföring som i projektet består av telekrig (Tk) och CNO (Computer Network Operations). Teknikplattformen innehåller redan nu mycket funktionalitet då den merutnyttjar resultat och programvara som utvecklats inom FoT projekt. Främst så har ett programpaket som kallas för EWSim (Electronic Warfare Simulation interface model) [EW] för distribuerade telekrigssimuleringar använts. Den tekniska plattformen finns idag installerad på FMV Smartlab samt på FOI Ledningssystem.

Utvecklingen av teknikplattformen för LKS har inneburit nyutveckling av en nätverksmodell med CNO, stabsverktyg för presentation av lägesinformation speciellt med avseende på telekrig och CNO samt att C2 funktionalitet utvecklats och integrerats med existerande modeller. Nätverksmodellen tydliggör effekterna av telekrig och CNO och hur nya möjligheter uppstår för samverkan mellan dessa, men även hur en nätverkslösning kan angripas, framför allt att det är lättare att angripa nätverk då hänsyn inte tagits till telekrig och CNO vid upprättandet av nätverk. LKS kan användas som i Figur 1 [FEDEP2v2] med två spelade sidor eller med en eller båda sidorna simulerade.



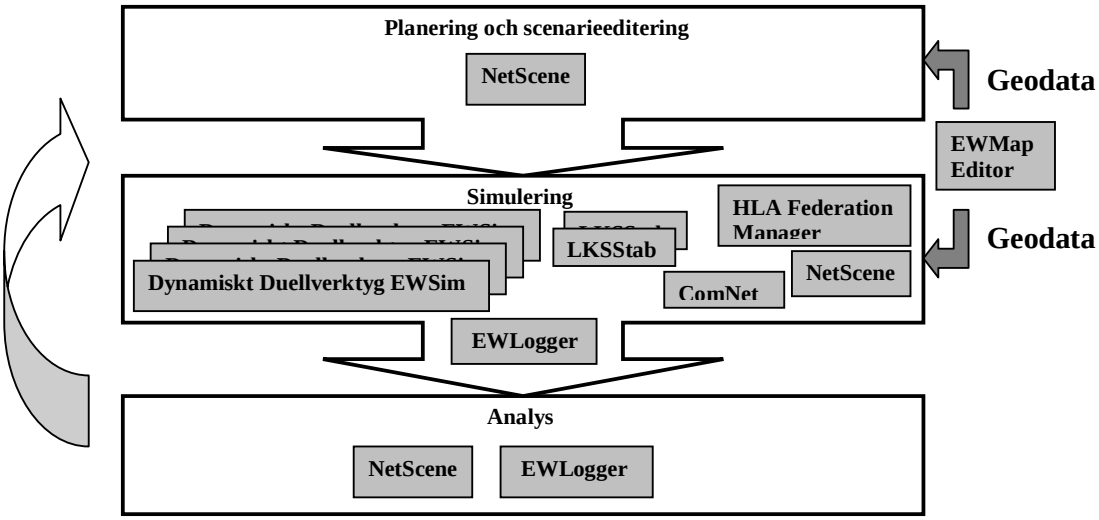
Figur 1 Principskiss för demonstrator LKS v2.

LKS-projektet har genomförts enligt metoden FEDEP (Federation Development and Execution Process IEEE 1516.3) vilket är en generell process för att ta fram en HLA-federation (High Level Architecture). FEDEP består av sju steg och i de först två stegen definieras mål samt görs en konceptuell analys. Dessa har ett IPT genomfört med personal från FM, FMV, FHS, FOI, och FrontEnd. De därefter följande stegen som är design utveckling och test samt integration, har genomförts av FOI avdelningen Ledningssystem på uppdrag av FMV, där FrontEnd har kontrollerat sluttesterna. Inom FOI har två grupper arbetat med LKS, en med kompetens inom Tk och CNO som tagit fram teknikplattformen och en grupp med kompetens inom MSI och VV&A som genomfört experiment och utvärdering. Förutom denna rapport finns en som beskriver designen av LKS teknikplattform [DES]. Den är skriven för tekniskt kunniga inom HLA, telekrig och CNO. Det finns dessutom en rapport som i detalj beskriver det verktyg som staberna jobbar mot, den är skriven för den som ska vara operatör i en spelad stab.

2 LKS teknikplattform

I LKS kan flera olika plattformar (t ex stridsvagnar och helikoptrar) konfigureras med olika komponenter (t ex sensorer och vapen) och simuleras distribuerat över ett nätverk.
En simulering kan sägas bestå av tre skeden, planering och scenarieeditering, simulering samt analys.

LKS teknikplattform är en HLA-baserad simuleringsmiljö som bygger på standarden RPR FOM (Real-time Platform-level Reference Federate Object Model) med några tillägg och består av en samling verktyg, Dynamiskt Duellverktyg EWSim, NetScene, MOSART HLA Federation Manager, EWLogger, ComNet och LKSStab, se Figur 2.



Figur 2 De delar som ingår i LKS.

NetScene används främst för planering och scenarieeditering. Där konfigureras i simuleringen ingående plattformar med önskade komponenter, samt att man bestämmer hur simuleringen ska distribueras över nätverket. Plattformarna placeras ut i den syntetiska terrängen och kan konfigureras med olika beteenden, t ex att följa en given bana. NetScene kan även utföra räckviddsberäkningar och siktanalyser för olika plattformar och komponenter för att underlätta planeringsarbetet.

Dynamiskt Duellverktyg EWSim är ett dynamiskt simuleringsverktyg som konfigureras av NetScene via HLA till en anpassad federat, se avsnitt 2.4, för en eller flera plattformar/komponenter med vapen och telekrigutrustning.

MOSART HLA Federation Manager är en simuleringskontrollfederat, som bl a hanterar start, stopp och paus av en simulering till deltagande federater.

EWLogger är en loggningsfederat som spelar in simuleringen. En inspelad simulering kan sedan återmatas in i alla federater och återuppspelas för utvärdering.

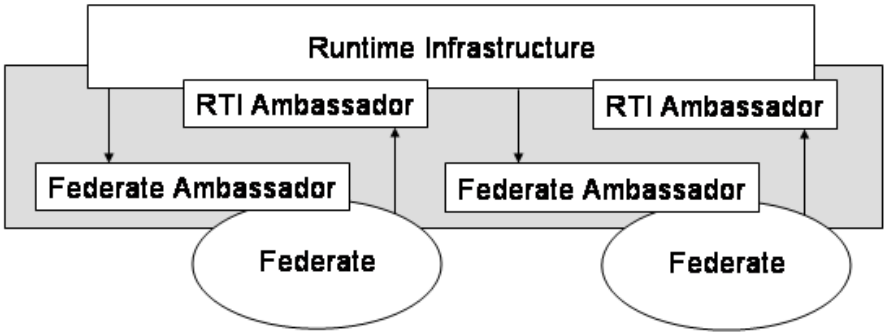
ComNet simulerar kommunikationen mellan noder i federationen både avseende nätverksbildning och själva signalöverföringen. Nätverkssimuleringen innefattar det digitala datanätet där meddelanden kan routas via andra noder i nätverket för att kunna nå den slutliga

destinationen. Den direkta signalöverföringen mellan två noder kan ske antingen via kabel eller radio. Även system för spaning och störning av kommunikationen simuleras i ComNet.

LKSStab är ett federat (program) vars främsta uppgift är att presentera lägesinformation för en användare och låta denne skicka order till övriga enheter i spelet. Lägesvyerna baseras på rapporter som kommit in via medelanden från övriga spelare (spelade eller simulerade) och presenteras främst genom uppritning av lägesinformation i en kartbild, men användaren har också ett antal andra vyer till sitt förfogande såsom en kartvy som visar uppspanat nätverksläge och en logisk nätverksvy.

2.1.1 HLA och MOSART

HLA är en standard för distribuerad simulering. Principen är att det finns en RTI (Run-Time Infrastructure) som är en central applikation som körs på en dator i ett nätverk. På samma dator eller andra datorer i nätverket kan sedan andra moduler som ska delta i en simulering exekveras. Dessa kallas federater. Federaterna kopplar upp sig mot RTI:n och bildar då vad som kallas en federation. Kommunikationen mellan en federat och RTI:n sker via API:er[1] som kallas ambassadörer. För att alla i en federation ska veta vilka data som kan utbytas finns en FOM (Federation Object Model) som specificerar detta. Data som kan utbytas är av objekttyp (långlivade) eller interaktionstyp (kortlivade). Båda typerna kan innehålla attribut. För att utbyta objektdata används principen publicera/prenumerera.



Figur 3 Federation med anslutna federater.

I LKS används ett antal programvarubibliotek som går under samlingsnamnet MOSART [MOSART] för integration mot HLA. Programbiblioteken är framtagna inom MOSART projektet som primärt syftar till att underlätta integration av komponenter i större simuleringar och demonstratorer. I projektet byggdes en modulär mjukvarumiljö upp, som tillhandahåller basfunktionalitet och möjliggör integration av egna och kommersiella mjukvaror. Mjukvaran för integration av moduler skall sänka tröskeln för att ta fram större simuleringar och demonstratorer. Detta uppnås genom att specifika krav på kunskaper om distribuerad simulering, HLA, har kunnat döljas för användaren.

MOSART tillhandahåller även funktionalitet utöver HLA kopplingen, såsom synkronisering och hantering av objekt och av attribut i RPR FOM:en, hantering av koordinattransforma-

[1] Application Program Interface - regler för hur fristående enheter kan kommunicera med en central del i programmeringssammanhang.

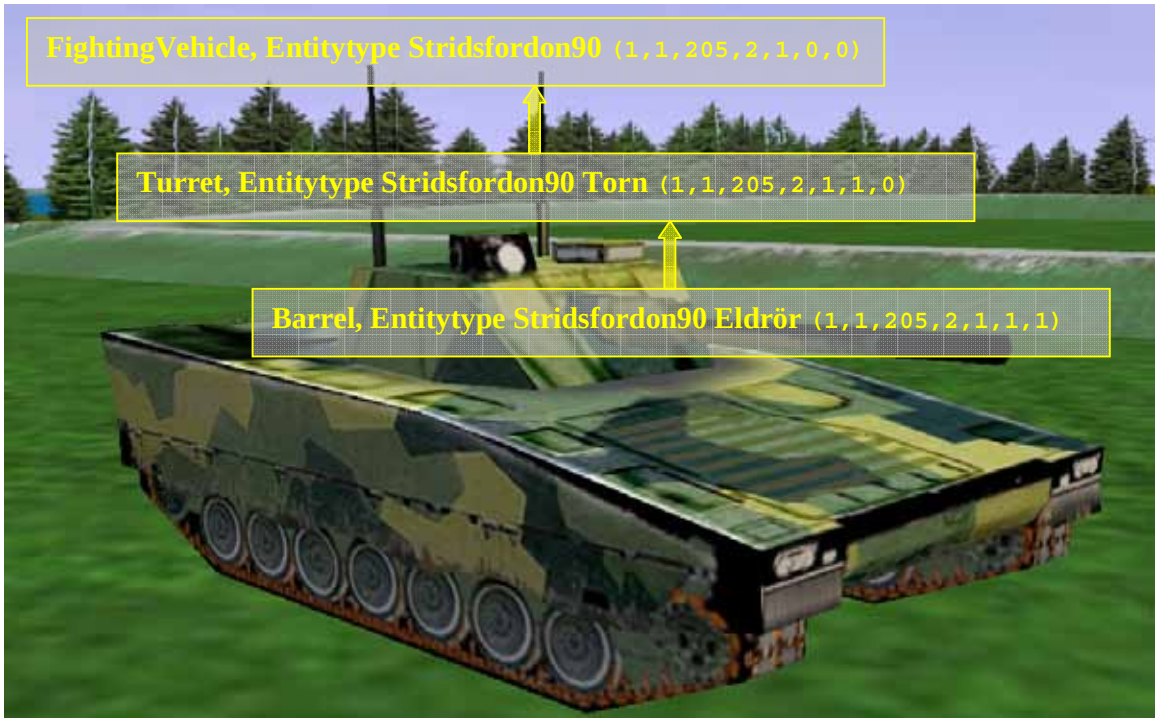
tioner, scenariostyrda händelser som paus, play, återstart och avslut samt dödräkning på objekts positioner.

2.1.2 RPR FOM

RPR FOM är en referens-FOM som definierar HLA-objekt och deras attribut samt interaktioner som är lämpliga för realtids- och plattformsbaserade simulatorer. En plattform är i detta avseende en fysisk enhet t ex flygplan, fartyg, soldat och ammunition. Varje plattform består av ett eller flera HLA-objekt, som är definierade i RPR FOM:en. LKS använder sig av en utökad version av RPR FOM, för att tillgodose behovet av HLA-objekt och interaktioner för specialiserade federater.

2.1.3 Plattformen och HLA-objekt

En plattform är samling av ett eller flera HLA-objekt som tillsammans representerar en specifik konfiguration av en enhetstyp. Varje HLA-objekt representeras av en HLA-klass i FOM:en t ex *GroundVehicle*, där attributet *EntityType* [ENTITY] som bygger på DIS-standarden[2] anger typen på objektet t ex Stridsfordon 9040. Objekten knyts samman med *IsPartOf*-attributet som anger objektets föräldrarelation (t ex tornet på stridsfordonet har stridsfordonet som förälder). Detta möjliggör att olika konfigurationer av t ex Stridsfordon 90 kan skapas samt att en plattforms olika delar kan simuleras distribuerat (t ex tornet på en federat och stridsfordonet på en annan).

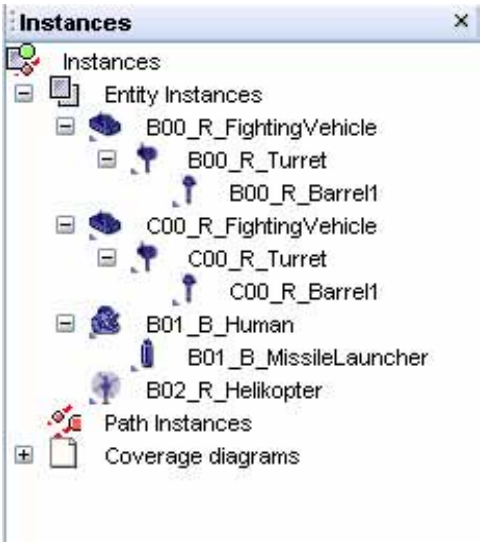


Figur 4 Exempel på en plattform. De gula rutorna visar HLA-objekten med *EntityType*, pilarna visar *IsPartOf*-förhållandet.

[2] Distributed Interactive Simulation

2.1.4 LKS-federation

I NetScene konfigureras plattformar och objekt.



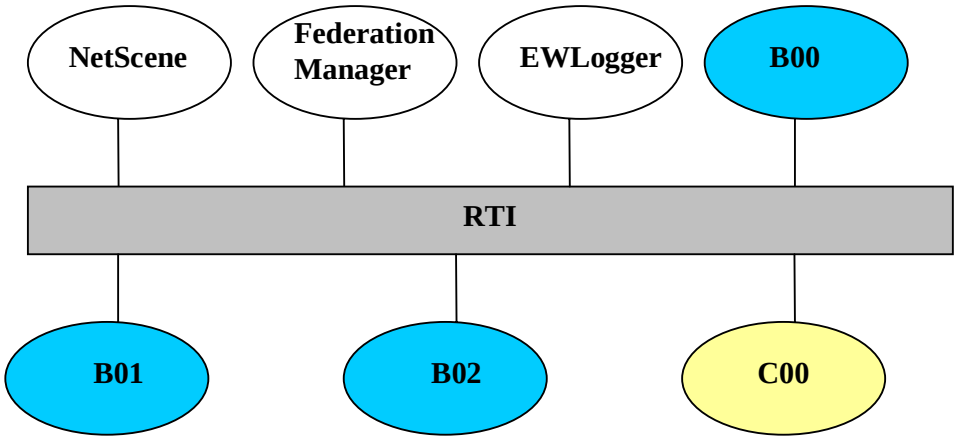
Figur 5 Instansvyn i NetScene, visar plattformarna och deras HLA-objekt.

Objektnamnet anger vilken federat som skall simulera objektet, vilken sida det tillhör samt dess namn: ex. *B00_R_ FightingVehicle*, *B00* anger federaten objektet ska skapas på, *_R_* anger objektets sida, R = röd, B = blå, G = Grön Y=Gul, samt slutligen namnet på objektet *FightingVehicle.* I exemplet i Figur 5 ingår fyra plattformar, två stridsfordon, en MANPADS-skytt samt en helikopter, vilket i detta exempel ger fyra federater som simulerar nio objekt, enligt Figur 6.

Objekt \ Federat	B00	B01	B02	C00
B00_R_FightingVehicle	X			
B00_R_Turret	X			
B00_R_Barrel	X			
C00_R_FightingVehicle				X
C00_R_Turret				X
C00_R_Barrel				X
B02_R_Helikopter			X	
B01_B_Human		X		
B01_B_MissileLauncher		X		

Figur 6 Fördelningen mellan objekt och federater i exemplet .

De fyra federaterna i exemplet som visas i Figur 5 kan simuleras i instanser av Dynamiskt Duellverktyg EWSim eller av externa applikationer. NetScene konfigurerar scenariot och distribuerar objekt till respektive federat via HLA, vilken kan ge en federationssammansättning enligt Figur 7.



Figur 7 Exempel på en EWSim-federation. De blåa federaterna är instanser av Dynamiskt Duellverktyg EWSim, medan den gula är en extern federat.

2.2 Planerings- och scenarieeditor NetScene

NetScene är ett program som används till planering och scenarieskapande. Tanken är att det ska vara lätt och smidigt att ändra och bygga ut scenarier; det är bara att klicka, dra och släppa olika stridsobjekt på en karta över ett område. Den har även ett 3D-fönster som gör det möjligt att se hur terrängen ser ut i höjdled och att skapa banor i luften. Banorna simulerar de olika stridsobjektens tänkta manöverrörelser. Med flera olika objekt på en och samma karta har man alltså skapat en fiktiv värld av olika stridsobjekt som kan röra sig individuellt. Dessa objekt kan sedan interagera med varandra.

2.2.1 Scenarieeditering

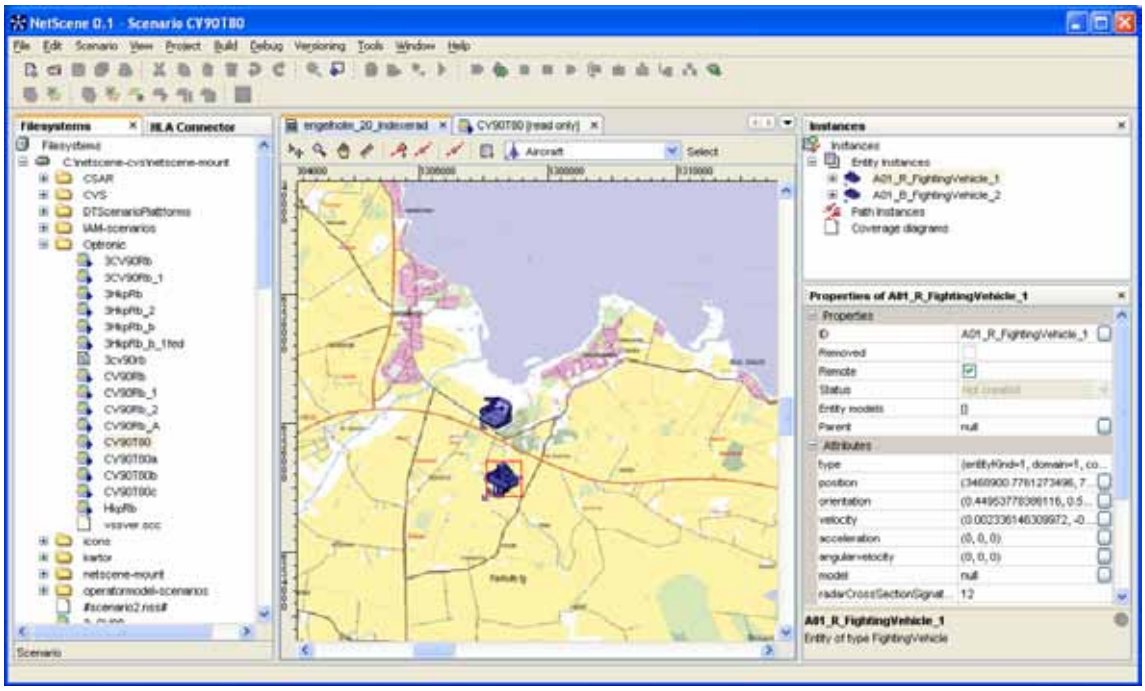
Ett scenario beskrivs av en scenariefil med tillhörande modellfil. De båda filerna är i XML[3]-formatet. Modellfilen definierar de olika objekt som ett scenario kan innehålla, på ett hierarkiskt sätt. Scenariefilen bestämmer de objekt som skall ingå i ett visst scenario. Modellfilen anges i scenariefilen så i praktiken behövs bara en modellfil, bara den innehåller alla objekt som ingår i de olika scenarier som man vill göra.

I modellfilen definieras alla objekt och vilka attribut och parametrar de har, t ex typ, position och orientering. Eftersom det är en hierarkisk struktur går det bra att definiera basobjekt och sedan låta mera speciella objekt ärva de grundläggande parametrarna och endast addera det som är utmärkande för just det objektet. Det finns en speciell parameter som definierar ett lastutrymme och vad som kan placeras där. Det medför att man får kontroll på hur objekt kan placeras på andra objekt. T ex har ”FightingVehicle” en ”TurretContainer” som en ”Turret” passar i, vidare har ”Turret” en ”BarrelContainer” som ”Barrel” passar i o s v. Denna hierarki ska passa in i den HLA-struktur som beskrivs i avsnitt 2.1.3.

Scenariefilen definierar i sin tur de olika objekten som skall ingå i scenariot. Alla fysiska och icke-fysiska objekt är med och får sin uppsättning parametrar. Det innebär att alla objekt i scenariot är individuella och kan anpassas till den uppgift de har i scenariot. För att det inte skall bli alltför tungrott så går det utmärkt att skapa plattformsfiler, scenariefiler, som består av en eller flera objekt (plattformar) t ex en stridsvagn med tillhörande utrustning. Dessa plattformsfiler kan man sedan dra in i kartan och släppa där de ska vara.

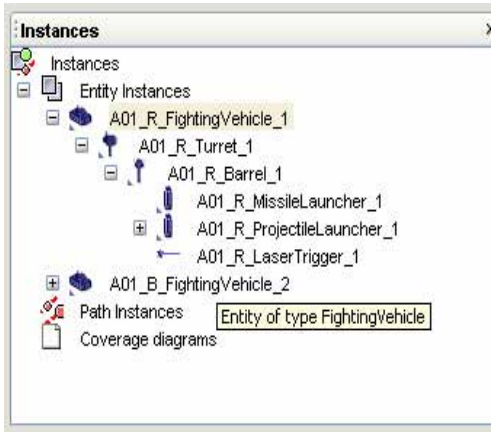
Figur 8 nedan visar hur NetScene kan se ut när ett scenario och en karta är laddad. NetScene är uppdelat i olika fönster, som kan flyttas och dockas så att användaren bestämmer vilka som skall synas och hur de ska vara konfigurerade. I detta exempel finns *Filesystems*, *HLA Connector* (gömd bakom Filesystems), *Instances*, *Properties of…* och det allmänna fönstret där kartvyn och de filer som öppnas för editering dyker upp.

[3] XML är ett textfilformat med ändelsen xml, som har etiketter, ”taggar”, vilka gör det möjligt att strukturera upp data så det lätt kan läsas in av programmet, samtidigt som det är möjligt att läsa och editera filen för hand. Om mer information önskas besök http://www.w3.org/XML/



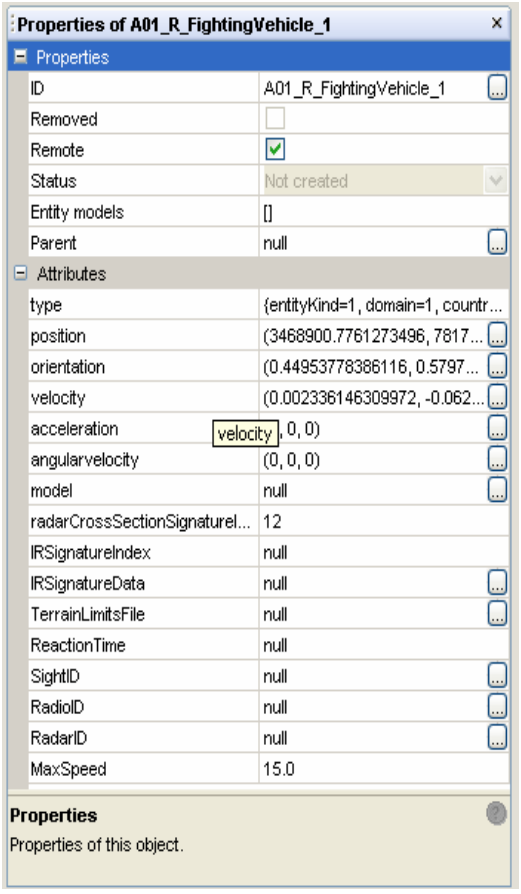
Figur 8 NetScene huvudfönster.

I *Instances*-fönstret (Figur 10) visas de objekt som scenariot innehåller. De egenskaper som objekten har kommer upp i *Properties of…*-fönstret (Figur 9) så fort ett objekt väljs, i detta fall A01_R_FightingVehicle1.



Figur 10 *Instances*-fönster.

I *Properties of…*-fönstret är det möjligt att ändra värdena för de olika egenskaperna som objekten har. Parametrarna beskrivs i Tabell 1.

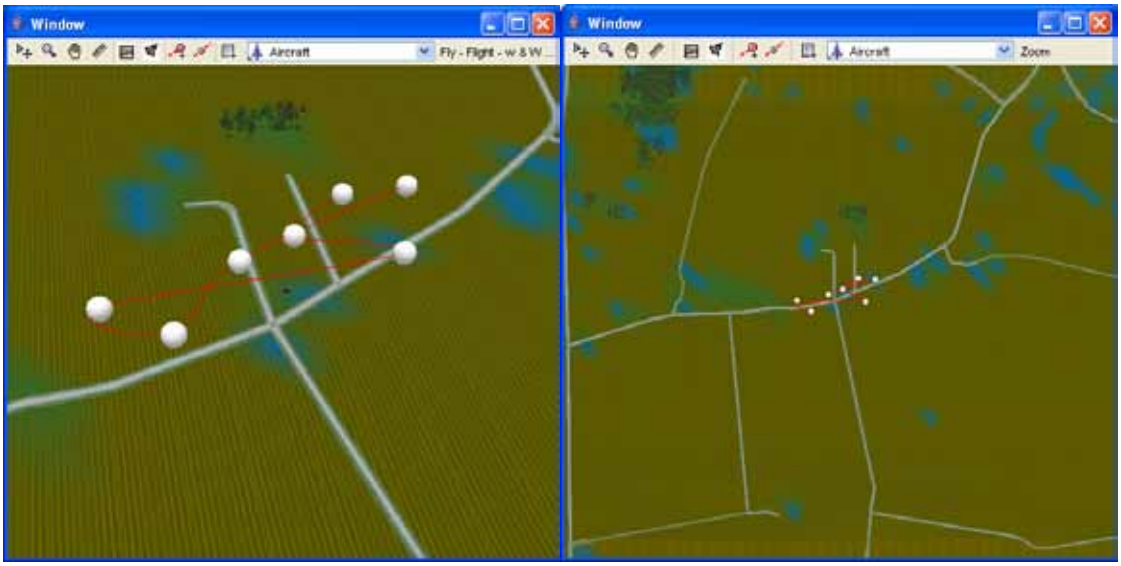


Figur 9 *Properties of…* fönster

Tabell 1 Beskrivning av de allmänna objektparametrarna.

Parameternamn	Beskrivning
ID	Unikt namn som identifierar objektet.
Remote	Bestämmer om objektet styrs lokalt från NetScene eller från en extern applikation t ex EWSim.
Entitymodels	Rörelsemodell för att möjliggöra rörelse efter en given bana.
Parent	Förälder till objektet.
Attributes…	De parametrar som är specifika för detta objekt.

Det går även att se ett scenario i ett 3D-fönster (Figur 11) i de fall som en 3D-terrängfil finns specifikt angiven i scenariefilen. Då skapas ett externt fönster där det är möjligt att flytta vyn som i en karta eller att flyga omkring som en observatör. Att se objekten röra sig i 3D-fönstret ger en känsla av direkt närvaro i scenariot, man kan där se objekten röra sig över den tänkta marken, korsa vägar etc. I 3D-fönstret väljer man antingen ortovyn, en kartvy där användaren befinner sig i en kartbild, eller flygvyn där användaren får upplevelsen av att se scenariot som från ett flygplan.



Figur 11 Bana i 3D-fönster. Till vänster visas den i en flygvy och till höger i en ortovy.

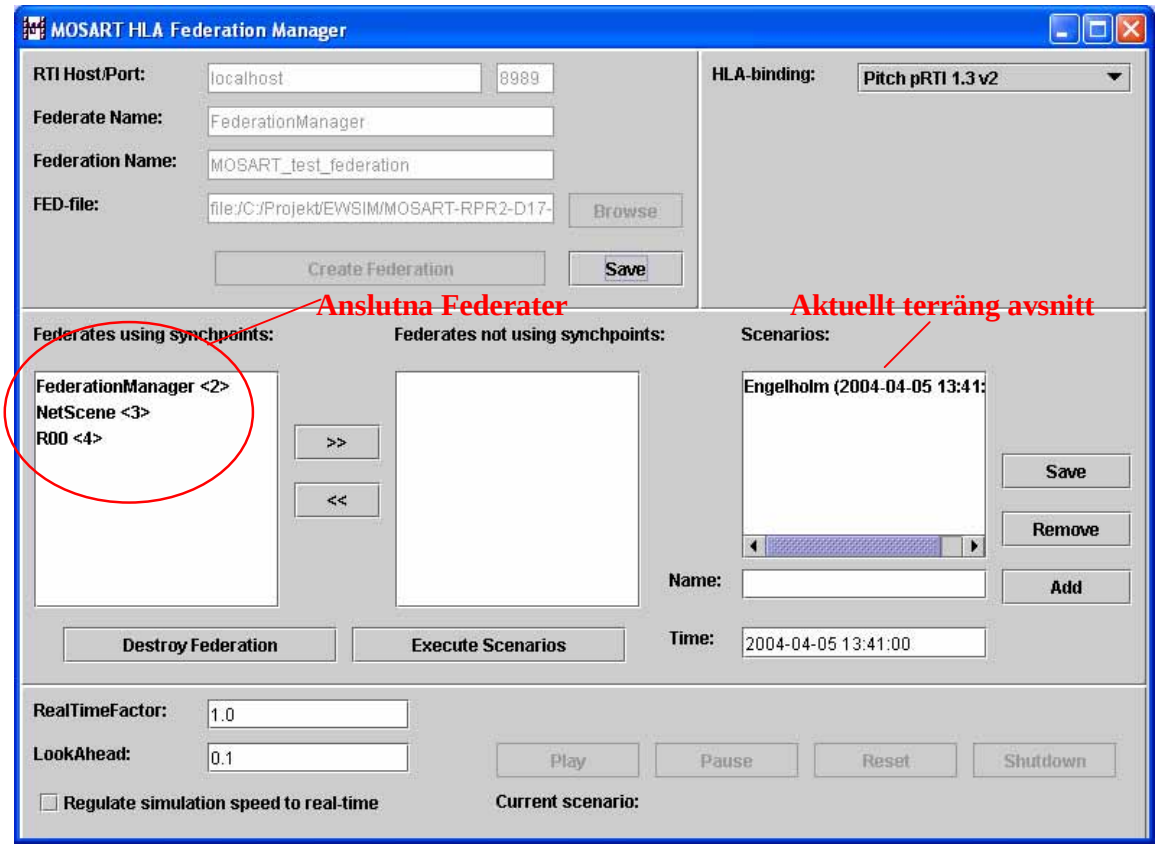
De vita bollarna hopbundna med ett rött streck i Figur 11 är en bana. Dessa banor skapas genom att klicka där brytpunkterna skall vara. När banan är skapad går det bra att lägga till, ta bort och flytta brytpunkter. Banorna skapas antingen i 3D-fönstret eller i kartvyn i NetScene. Vill man ha en speciell höjd på banan måste man däremot vara i 3D-vyn.

När banorna är skapade går det att koppla ihop objekt med banorna och låta objekten åka längs banorna m h a enkla rörelsemodeller. Det går att köra en simulering direkt i NetScene

för att se hur objekten rör sig i scenariot. Således kan användaren själv skapa de strids-scenarier som önskas, och själv bestämma vad som ska hända i dessa.

2.3 MOSART HLA Federation Manager

MOSART HLA Federation Manager är en federationskontrollfederat som styr scenariots exekvering genom att hantera tiden och händelser som start, stopp och paus, samt att specificera i vilket terrängavsnitt aktuell simulering skall köras. Figur 12 visar utseendet av MOSART HLA Federation Manager.

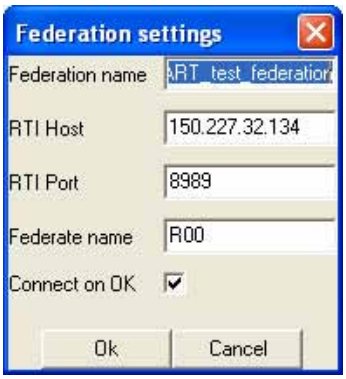


Figur 12 MOSART HLA Federation Manager.

2.4 Dynamiskt Duellverktyg EWSim

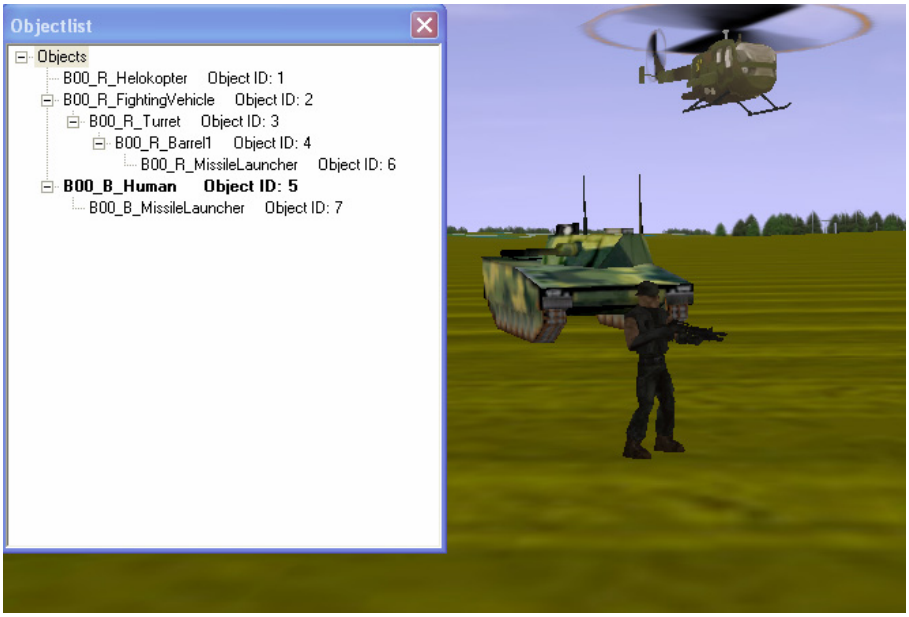
Dynamiskt Duellverktyg EWSim är ett gemensamt gränssnitt för NetScene-konfigurerade federater. Vid uppstart anges federationsinställningar:

- Federation name, namnet på federationen.
- RTI host. IP-numret till den dator som kör RTI:et.
- Federate name, namnet på den egna federaten.



Figur 13 *Federation Settings*-fönstret.

När scenariot exekveras i Federation Manager laddas terrängen, och vid start skapas plattformarna och deras HLA-objekt. För varje plattform som federaten ska simulera laddas det specialiserade s k *EventHandlers*, som hanterar användarens interaktioner. Om federaten äger flera plattformar kan användaren växla mellan vilken plattform han för tillfället styr (via menyn, Control→Toggle vehicle eller kortkommandot alt+t). Vilka interaktioner användaren kan utföra på plattformen kan man läsa i fönstret *Handler Commands* (öppnas via menyn, View→Handler Commands). Ingående delar i ett scenario kan visas under simulering i fönstret *Objectlist*, se Figur 14.

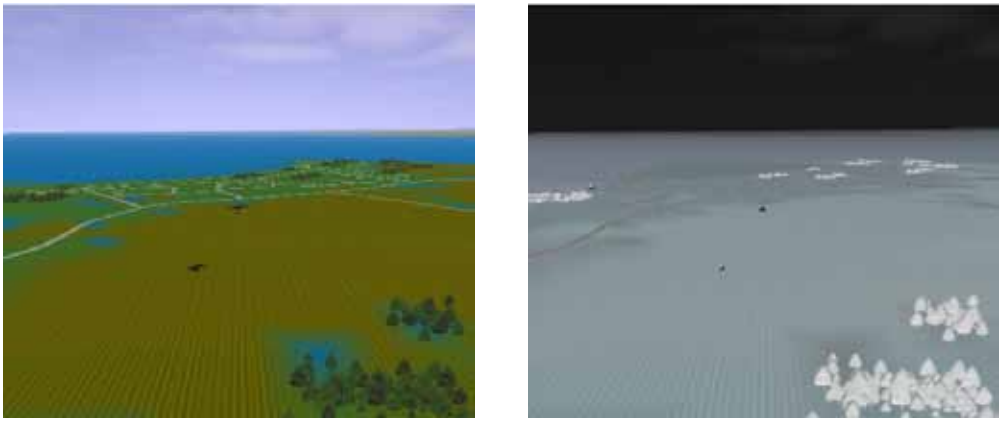


Figur 14 Dynamiskt Duellverktyg EWSim med objektlistan öppen. Listan visar i simuleringen deltagande plattformar och objekt.

2.5 EO Modeller

För att simulera dueller/konfliktsituationer där EO-området finns representerat behövs terräng och objekt inom det visuella, IR och/eller UV-området. Dessutom behövs sensorer och system som kan verka inom dessa områden.

I EWSim beskrivs terräng och objekt inom EO-området i huvudsak genom deras 3D-geometri och deras texturer (bilder som ”klistras” på 3D-objektets yta). Texturer kan väljas beroende på om de studeras via en visuell eller en IR-kamera, se Figur 15.



Figur 15 Exempel på hur samma vy kan visas med visuell textur (till vänster) och IR textur (till höger).

I vissa fall används inte den ovan beskrivna metoden utan tabeller för att slå upp en total signatur vilka kan ges för varje enskilt objekt. IRST:n (InfraRed Search and Track) i EWSim använder t ex tabeller för att beräkna om ett objekt kan upptäckas (däremot används IR-vyn för visualisering). I andra fall används bara vetskapen om att en viss typ av objekt har dykt upp och förväntade egenskaper hos det objektet. UV-varnare i EWSim tittar bara efter missiler och antar att de ger UV-signatur under en viss begränsad tid efter avfyring. Optiksignatur (den retursignal som optikspanare får) bestäms av egenskaper för optiken som kan finnas på ett system.

De EO-sensorer som kan användas i EWSim kan vara enkla kameror inom det visuella eller IR-området men de kan också vara mer avancerade system som automatiskt hittar tänkbara mål/hot. Exempel på avancerade spaningssystem inom det elektrooptiska våglängdsområdet kan vara IRST eller optikspanare. Den förra letar efter objekt med förhöjd signatur över bakgrunden och den senare letar efter optiska system riktade så att den egna plattformen finns inom dess synfält. Dessa system kan innehålla logik för att bara varna för mål som närmar sig den egna plattformen. Sensorer i form av rena varnare, t ex laservarnare eller UV-varnare kan också förekomma.

Varning kan ges från laservarnare, UV-varnare, IRST, optikspanare m fl. Dessa varningar kan konfigureras i ett varnare motmedels -system (VMS) och resultera i en motåtgärd. De EO-motåtgärder som finns är att lägga ut rök, fälla facklor eller använda laserstörning (DIRCM). Det går givetvis att blanda EO-motåtgärder med andra typer av motåtgärder, t ex en motåtgärd i form av moteld men även radar- eller kommunikationsmotåtgärder kan användas.

Modellerna i EWSim är generiska, d v s för att skapa ett specifikt system med dessa modeller sätts parametrar till olika värden. För fler detaljer om de enskilda modellkomponenterna se referens [ESOPTR].

Detta kapitel beskriver IRST speciellt då det endast är denna som använts i LKS.

2.5.1 IRST

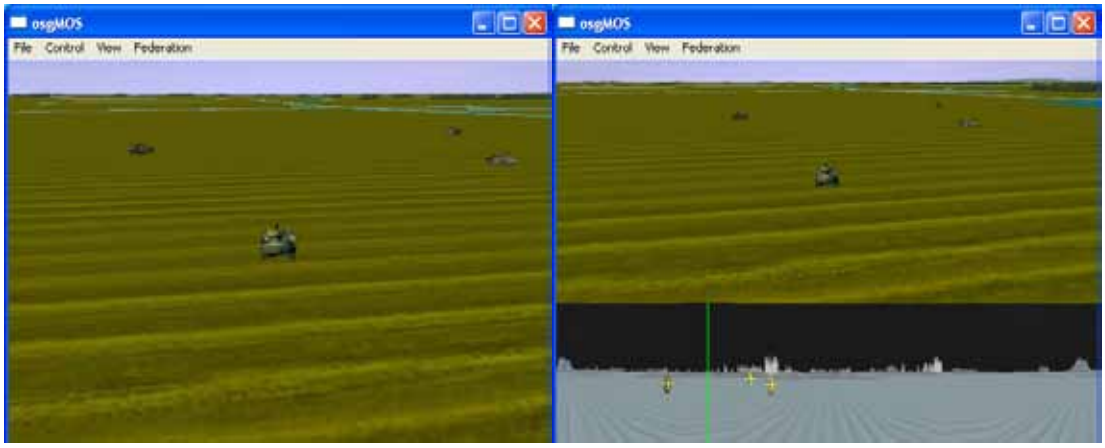
I Allmänt om IRST

En IRST är en sensor som ofta klarar att svepa över ett stort vinkelområde av scenen och kan användas för att upptäcka och följa objekt med ett, inom IR-området, lågt signal-till-brus-förhållande (inkluderat klotter), t ex inkommande missiler. IRST förekommer både i mark-, fartygs- och flygmonterade applikationer.

II IRST i EWSim

En IRST kan konfigureras i NetScene och kopplas till ett markfordon, en flygande plattform eller till ett fartyg, vilka kan styras i EWSim. IRST:n kan i modellen användas både som varnare och för att upptäcka objekt. För upptäckt krävs att objektet ligger inom IRST:ns synfält, inte är skymt och har en signatur (som ger en IR-signal) som överstiger bakgrundens (eller en given brusnivå) med en given faktor (SNR). SNR-beräkningarna tar hänsyn till målets IR-signatur som funktion av aspektvinkel, sensorprestanda och atmosfärens dämpning. I varnarsammanhang måste dessutom mottagen strålningseffekt öka och objektet får inte röra sig i vinkelled, vilket är karakteristiskt för ett närmande hot.

När ett objekt har ett IRST-system kan detta aktiveras i EWSim. Användaren kan sedan välja om en IRST-vy skall visas i fönstret eller verka i bakgrunden (det senare kan vara ett alternativ speciellt om IRST:n endast används som varnare kopplat till ett automatiskt VMS), se Figur 16.

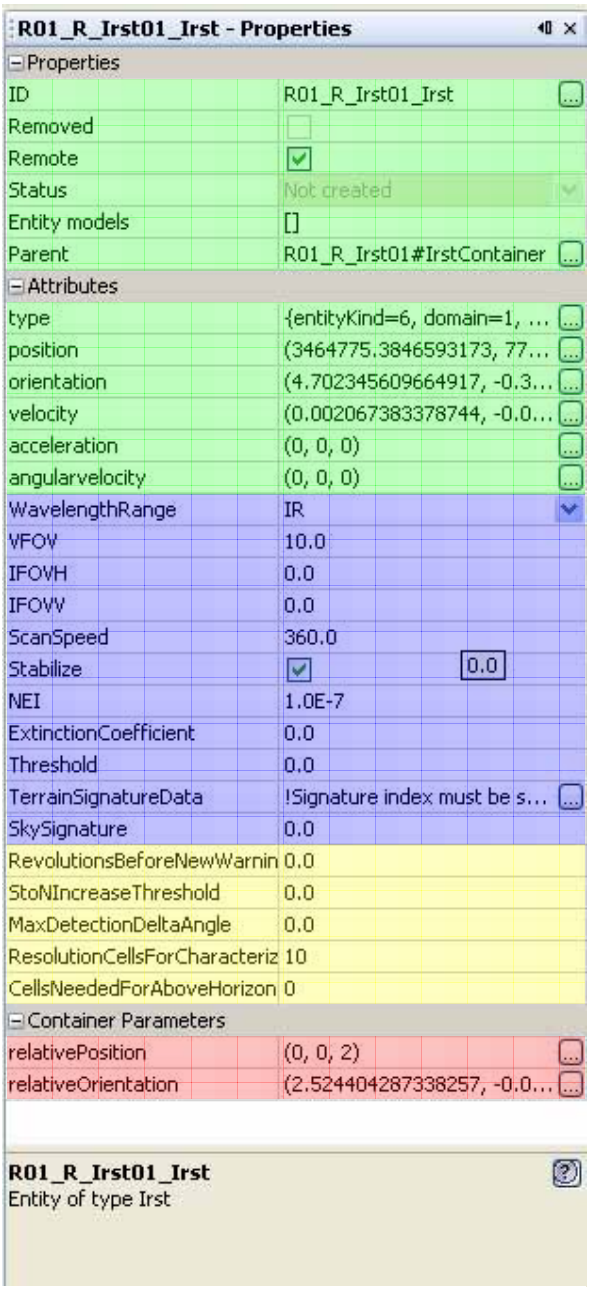


Figur 16 Användargränssnitt som visar en översiktsbild med kameran positionerad bakom och ovanför det styrda objektet (kamerans synfält är centrerat kring objektet). Det styrda objektet är utrustat med en IRST. Till höger har IRST:n aktiverats och en IRST-bild dyker upp i nederkanten av användargränssnittets fönster.

En IRST-vy visar upptäckta målkandidater som gula ’+’. Målkandidater, som resulterar i en varning genererar ett rött ’+’ tills denna varningsposition är överlämnad till plattformens VMS. Orienteringen av plattformen på vilken IRST:n är monterad (och ett eventuellt kanontorns orientering) visas också i IRST-vyn.

Objekt som skall vara möjliga att upptäcka måste initieras med IR-signaturdata. Detta kan ske genom att ge data direkt i NetScene eller om inga data matas in genom att utnyttja entitetstypen. I det senare fallet läses data från en fil med namnet #Kind_#Domain_#Country_#Category_#Subcategory_#Specific_#Extra.txt, där #XXX svarar mot en siffra i entitetsnumret, t.ex. 1_1_205_2_1_0_0.txt för stridsfordon 90. Filerna skall ligga under datakatalogen i en map som heter IrSignatureData. I fall denna fil saknas läses data från en fil med namnet default.txt i samma katalog.

2.5.2 Konfigurera IRST i NetScene



Figur 17 Parametrar för IRST

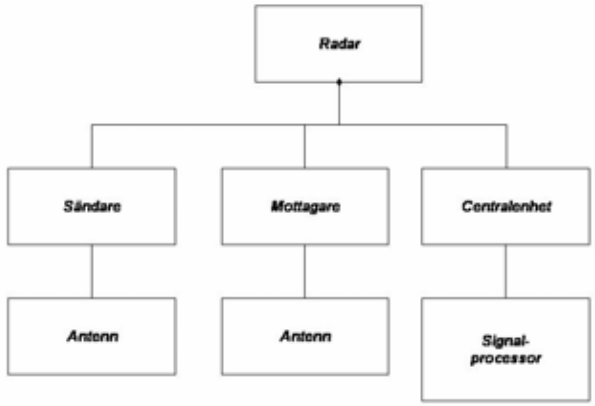
En IRST bygger på parametrar från RPR FOM.ens *BaseEntity*, dessa är grönmarkerade i Figur 17. En del av dessa parametrar handlar t ex om position, orientering och hastighet men eftersom en IRST inte förekommer som enskild komponent utan alltid sitter på en plattform eller på annat objekt så kommer deras värden under konfigurering inte att påverka slutresultatet. Däremot kommer de parametrar som är rödmarkerade att påverka positionen relativt plattformens position. Den relativa orienteringen kommer att ändras under simuleringens gång eftersom IRST:n roterar när den är påslagen. Förutom dessa parametrar finns parametrar som bestämmer egenskaper för IRST:n och därmed möjligheten för IRST:n att upptäcka och lösa upp mål (blåmarkerade parametrar). *WavelengthRange* kan vara IR eller visuell och bestämmer hur sensorbilden från IRST:n visualiseras. Dessutom påverkar denna parameter vilka signaturdata som skall användas från alla objekt. Det vertikala synfältet i grader bestäms av parametern *VFOV* (det horisontella är alltid 360º). *IFOVH* och *IFOVV* bestämmer upplösningen i radianer i horisontal- och vertikalled. Dessa parametrar påverkar intensitetsnivån från ett punktmål i förhållande till bakgrunden, de påverkar hur noggrant en IRST kan mäta in ett mål i vinkelled samt vilka möjligheter en IRST har att karakterisera ett mål. Parametern *ScanSpeed* (º/s) talar om hur ofta IRST:n kan uppdatera sina måldata, 360 º/s innebär att IRST:n mäter ett varv per sekund. *Stabilize* bestämmer om IRST:ns vertikala axel alltid skall peka rakt uppåt eller om

den skall ha samma vertikala orientering som den plattform till vilken den är kopplad. *NEI* (noise equivalent irradiance) är ett mått på hur känslig sensorn är (W/m^2), ett lägre värde på *NEI* innebär att objekt som visar lägre signatur lättare kan upptäckas. *ExtinctionCoefficient* talar om hur snabbt atmosfären dämpar kontrasten från objekt, 0.0 innebär att ingen attenuering sker, 0.7 innebär att ungefär halva kontrasten försvinner på en kilometers avstånd. *Threshold* är ett tröskelvärde som signal-till-brus förhållandet (eller signal-till-bakgrundsförhållandet) måste överstiga för att ett objekt skall upptäckas. *TerrainSignatureData* beskriver signaturer i tabellform för olika typer av terränger. *SkySignatur* är ett värde på bakgrundsstrålningen i W/sr om det inte finns någon terrängen på mindre avstånd än 10km.

De parametrar som är gulmarkerade i Figur 17 är parametrar som styr varnarfunktioner. *RevolutionsBeforeNewWarning* talar om hur många varv ett redan upptäckt objekt måste vara osynligt för att en ny varning skall ges. *StoNIncreaseThreshold* talar om hur snabbt signal-till-brusförhållandet måste öka i %/s för att varning skall ges. *MaxDetectionDeltaAngle* talar om hur mycket vinkeln för ett upptäckt objekt får ändra sig i rad/s för att objekt skall betraktas som ett hot och varning ges. De två sista parametrarna *ResolutionCellsForCharacterization* och *CellsNeededForAboveHorizon* används för att bestämma om IRST:n kan dra några slutsatser om ett upptäckt mål. Om ett objekt täcker tillräckligt många bildelement (storleken på ett bildelement bestäms av *IFOVH* och *IFOVV*) kan IRST:n automatiskt karakterisera målet och rapportera detta eller om ett mål befinner sig tillräckligt högt över horisonten kan IRST:n dra slutsatsen att det är en flygande plattform.

2.6 Radarmodeller

Radarmodellen i EWSim är en generisk modell, d v s utgående från samma modell kan man simulera olika radarsystem genom att parametersätta på olika sätt. Radarmodellerna är utvecklade för att underlätta interaktioner modellerna emellan. Det centrala i framtagandet av radarmodellerna har varit telekrigsaspekten, d v s möjligheten att störa och att skydda sig mot störning i radarfallet. Radarmodellen är uppbyggd i modulform där delar med centrala egenskaper har modellerats som egna objekt. De delar som har störst enskild betydelse för modellens uppträdande är sändare, mottagare, antenner och signalprocessor, se Figur 18.



Figur 18 Radarmodellens struktur och centrala delar

Varje radarobjekt har själv ansvaret för att göra erforderliga beräkningar för att ta fram mottagna signaler. En kanaladministratör i EWSim håller reda på radarmottagare och sändare samt inställningarna för dessa. Även passiva objekt, radarmålareor, modelleras med sändare och mottagare vilket gör att kanaladministratören också kan hålla reda på dessa. Då ett radarobjekt ska göra beräkningar för vad den tar emot kommunicerar den med kanaladministratören om vilka sändare som är rätt inställda (frekvens, på eller av etc.) för att radarn ska kunna ta emot signaler från dem. För de sändare som är rätt inställda görs en kontroll om de är skymda av terrängen eller inte. Radarn har beräknat en terrängkarta där avståndet till terrängen runt om radarn är lagrat. Vid kontroll av terrängmask slås rätt värde upp i terrängkartan och jämförs med avståndet till sändaren. Om sändaren ligger framför terrängen görs mer specifika radarberäkningar för att ta fram måldata. I annat fall sorteras sändaren bort då den inte kan påverka radarmottagaren.

Eftersom radarmodellen är uppbyggd av moduler är det lätt att byta ut t ex en antenn mot en annan utan att behöva ändra i koden. Detta är en förutsättning för att det ska vara lätt att modellera olika system på ett smidigt sätt utan att behöva ha en mängd parametrar som måste ställas in på rätt sätt.

Radarantennerna styrs av avsökningsmönster. Dessa mönster kan man som användare definiera utifrån vissa parametrar för att få de mönster som systemet använder. Ett specialfall är en spaningsradar som har en antenn som snurrar med 2 varv per sekund. Ett annat är en eldledningsradar som i pålåsningsfas utför en nickande rörelse för att hitta målet i höjdled.

För att kommunicera radarinformationen finns en radarsignalklass som innehåller information på pulsskurnivå såsom prf (pulsrepetitionsfrekvens), medeleffekt, frekvens etc. Alla objekt

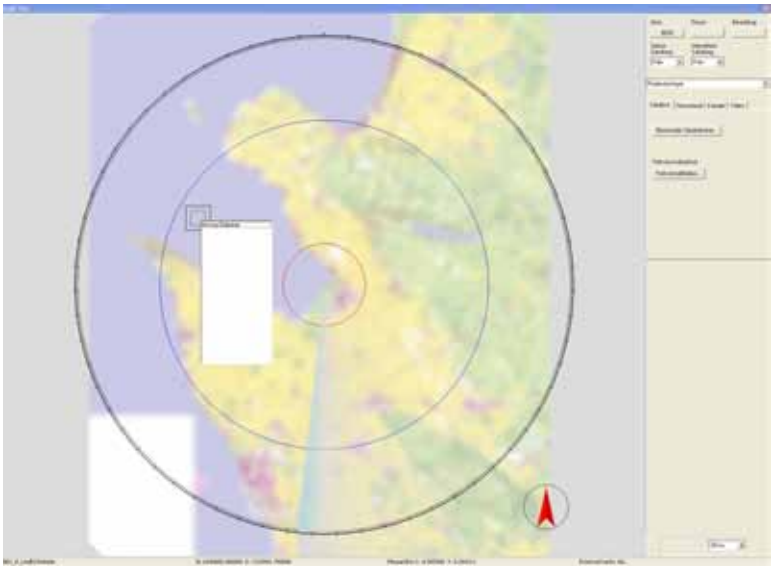
som kan påverka en radarsignal har en radarsignatur som håller reda på alla typer av interaktioner mellan objektet och signalen. Interaktionerna kan vara passiva i form av studs av signalen eller aktiva i form av en radarstörare.

Funktioner som finns i radarn är t ex pulskompression, koherent integration, olika prf-typer, störundertryckning som MTI etc. Vilka funktioner som används beror på vilket system som modelleras.

2.6.1 Spaningsradar

I EWSim finns en parametriserad spaningsradarmodell som efterliknar försvarets underrättelseenhet UndE23. Modellen innehåller ett litet urval av de funktioner som finns i systemet. Inställningar som finns är intermittent sändning som gör att radarn sänder vissa antennvarv. Operatören kan välja att radarn sänder 30%, 50% eller 70% av antennvarven. När valet sker slumpar radarn fram de antennvarv då radarn ska sända av 128 stycken. Denna slumpning gör det svårare att se mönster i radarsändningen. Då antennen snurrat 128 varv upprepas samma mönster. Operatören kan också ställa in blockerade sändsektorer. Dessa sektorer hindrar radarn från att sända just där, men i övriga sektorer sker sändning som vanligt. Radarn kan jobba i aktiv, passiv eller beredskapmod. I aktiv mod sänder radarn själv och kan på så sätt se radarreflektorer. I passiv mod sänder inte radarn själv, men kan se andra radaremittrar och aktiva radarstörare om dessa sänder med inställningar som matchar radarns egna inställningar. I beredskapsläge är radarn i vila.

Ett användargränssnitt har tagits fram för att ge möjlighet att som operatör styra radarn, se Figur 19.



Figur 19 Användargränssnitt för spaningsradar i EWSim.

I användargränssnittet finns möjlighet att ta fram måldata i form av koordinater och bäring för invisning via radio samt att automatiskt skicka målkoordinater till fördefinierade eldenheter.

Spaningsradarn kan störas av radarstörsändaren, se 2.6.3.

2.6.2 Konfigurera Spaningsradar i NetScene

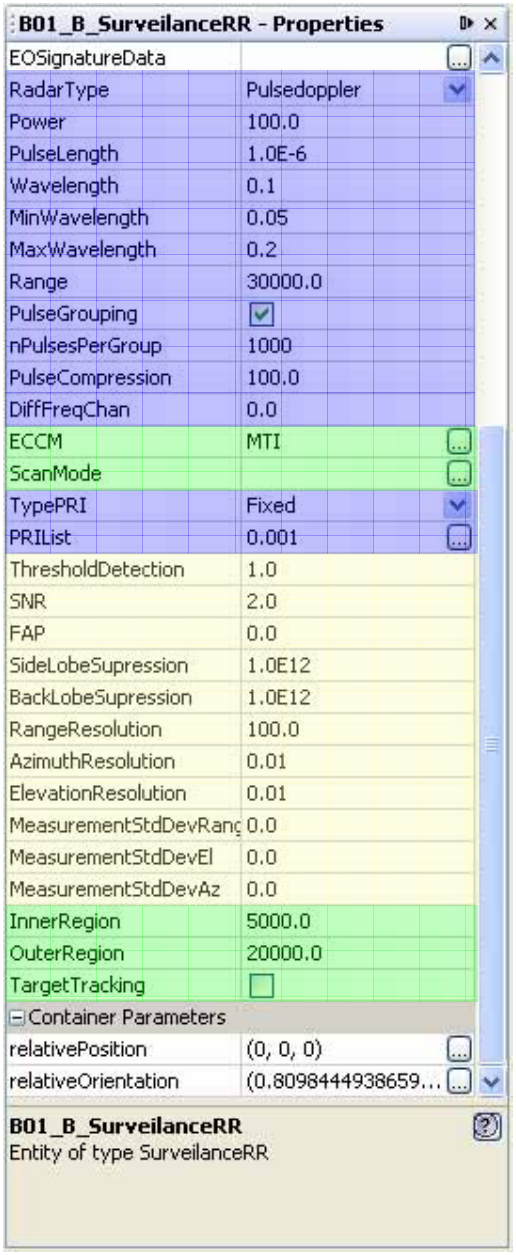
En spaningsradar (*SurveillanceRR*) bygger på parametrar från *PhysicalEntity* (som i sin tur ärver av *BaseEntity*). Dessa parametrar är beskrivna tidigare i dokumentet, avsnitt 2.5.2. Parametrarna som är specifika för spaningsradarn är markerade i blått, grönt och gult i Figur 20. Blåmarkerade parametrar beskriver radarns egenskaper vid sändning, gulmarkerade beskriver egenskaper vid mottagning.

Power anger den utsända strålningens effekt i Watt. Strålningen dämpas med hänsyn till den sändande antennen (antenndiagrammet). *Range* anger radarns räckvidd.

Parametern *RadarType* bestämmer modulationen på den utsända radarstrålningen, och kan sättas till ”Pulsedoppler”, ”Pulse” eller ”CW”. En pulsad radar kan mäta in positionen på ett radarmål. Vid CW (kontinuerlig strålning) kan man endast erhålla bäringen till målet. Med CW och pulsdoppler kan målets hastighet mätas in, något som inte är möjligt med den konventionella pulsningen. *PulseGrouping* är en metod som förbättrar signal-till-brus-förhållandet vid pulsad strålning, ju fler pulser desto bättre SNR. Antalet pulser anges med *nPulsesPerGroup*. En annan metod för förbättring av signalnivån är *PulseCompression*.

Wavelength anger radarns våglängd i meter och är bland annat nödvändig för att beräkna doppler. *Pulselength* anger radarns bandbredd och bestämmer tillsammans med *MinWavelength* och *MaxWavelength* hur många kanaler som radarn kan sända på. Bandbredd och våglängd är även viktiga parametrar för att beräkna bruseffekt vid störning. De ger hur stort överlappet i våglängd är med den störande signalen.

ThresholdDetection anger vilken mottagen effekt som krävs för att ett radarmål ska upptäckas. För att måldata ska kunna extraheras krävs dessutom att signalnivån är tillräckligt stor i förhållande till bruset. Lägsta möjliga SNR-nivå anges av parametern *SNR*. Mål kan detekteras även utanför mottagarens huvudlob, i sid- och backlober. Hur mycket strålningen undertrycks utanför huvudloben anges av



Figur 20 Parametrar för spaningsradar.

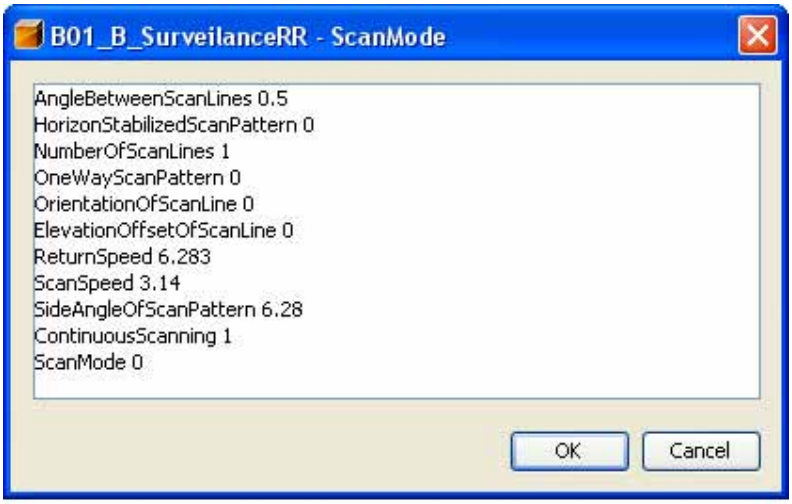
SideLobeSupression och *BackLobeSupression*.

Hur noggrant radarn kan positionsbestämma ett målobjekt bestäms av upplösningen i avstånd och vinkel; *RangeResolution*, *AzimuthResolution* och *ElevationResolution*. Osäkerheten i varje mätning ges av standardavvikelserna *MeasureStdDevRange*, *MeasureStdDevAz* och *MeasureStdDevEl*. *TargetTracking* anger om upptäckta mål ska målspåras vilket då görs m h a kalmanfilter. Observera att det endast är målspår som rapporteras vidare till C2 funktionerna.

I varje tidssteg finns en sannolikhet för falskmål att dyka upp. Denna sannolikhet anges av *FAP* (False Alarm Probability). För att undertrycka falskmål finns en rad motmotmedel (CCM). Dessa anges i parametern *ECCM* som är en textsträng där en rad CCM kan anges med mellanslag. Följande CCM finns tillgängliga: MTI, DickeFix och AccLimit. MTI står för Moving Target Indicator och undertrycker falskmål som rör sig för fort. AccLimit undertrycker istället mål som accelererar för fort. DickeFix reducerar effekten från brus.

DiffFreqChan, *TypePRI*, *PRIList*, *InnerRegion* och *OuterRegion* används för tillfället inte av radarmodellen.

Om inget annat anges söker spaningsradarn av omgivningen med en antenn som snurrar 2 varv per sekund. Användaren kan även välja att definiera ett eget avsökningsmönster. Mönstret beskrivs av parametern *ScanMode*, en textsträng som måste vara skriven på ett visst sätt. Varje rad innehåller en parameter och tillhörande värde, separerade med mellanslag. Exempel på en sådan textsträng ges i Figur 21. Det är viktigt att parameternamnet är rättstavat och att man skiljer på stora och små bokstäver i namnet. Parametrarna motsvarar direkt de som beskrivs i Tabell 2. Parametrar av typen *bool* sätts till 0 för falskt och 1 för sant.



Figur 21 Exempel på initiering av ett avsökningsmönster från NetScene.

Typ	Namn	Funktion
bool	*m_bHorizonStabilizedScanPattern*	Sann om avsökningsmönstret är horisontstabiliserat.
int	*m_nNumberOfScanLines*	Antal söklinjer i vertikalled.
float	*m_fAngleBetweenScanLines*	Vinkeln mellan två vertikala söklinjer.
bool	*m_bOneWayScanPattern*	Sann om radarn endast sänder då antennen svänger i en riktning. I returen är då radarn tyst.
float	*m_fOrientationOfScanLine*	Anger vinkeln mot horisonten om avsökningen inte är horisontstabiliserad.
float	*m_fElevationOffsetOfScanLine*	Anger offset i elevationsled givet i radianer för avsökningsmönstret (0 om ingen offset, positivt uppåt, negativt nedåt).
float	*m_fSideAngleOfScanPattern*	Anger sidovinkeln i sökmönstret.
float	*m_fScanSpeed*	Hastigheten antennen svänger med vid sändning.
float	*m_fReturnSpeed*	Hastigheten antennen svänger med då radarn är tyst.
bool	*m_bContinuousScanning*	Sann om antennen sänder kontinuerligt och aldrig byter riktning (specialfall för spaningsradar).
Vec3	*m_StartForwardVector*	Anger antennens riktning vid starten av avsökningen.

Tabell 2 Parametrar för avsökningsmönster.

2.6.3 Radarstörsändare

En generisk modell av en radarstörsändare, GenPod, har utvecklats till EWSim. Modellen har utvecklats parallellt med radarmodellen för att kunna utnyttja gemensam struktur. GenPod beskrivs i [GENPOD].

De störsändare som har implementerats i EWSim är brusstörare och repeterstörare. Brusstöraren ställs in med frekvens, bandbredd och effekt och kan sedan stängas av och sättas på. Brusstörsändaren utnyttjar inte någon inkommande radarsignal för analys av frekvens mm. utan är själv aktiv oavsett om någon radarsignal detekterats. I radarn tas brussignaler emot och för dessa beräknas hur stor del av störsignalens frekvensområde som överlappar radarns frekvensområde för att ta fram hur mycket störningen påverkar. Om störsignalens frekvens till någon del ligger inom radarns frekvensområde kommer störsignalen att höja brusnivån i radarn. Effekten brusstörningen har på radarn är alltså att brusnivån höjs. Då radarn letar efter mål kräver den ett visst signal-brus-förhållande för att målet ska kunna detekteras. Om brusnivån höjs tillräckligt mycket kommer alltså målekon att drunkna i bruset.

En enkel repeterstörare finns som tar emot en radarsignal och sedan skickar ut kopior av radarsignalen med viss tidsfördröjning. Repeterstöraren är enkel i den mening att tidsfördröjningen endast gör att målen förskjuts radiellt ut från radarsensorn. Mer sofistikerade repeterstörare finns utvecklade som möjliggör även förskjutning i vinkel genom att analysera

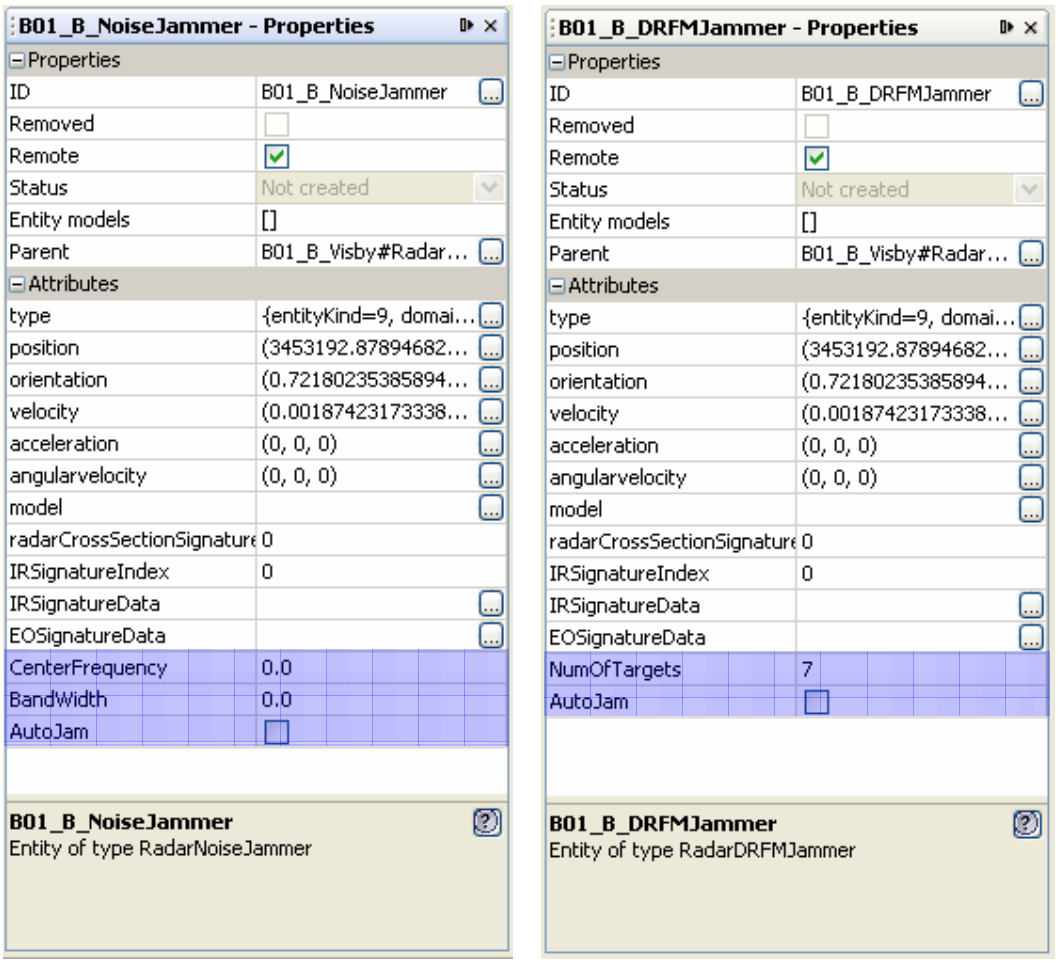
radarsensorns sändmönster. Dessa finns dock inte modellerade i EWSim i dagsläget. I radarn tas störsignalerna emot från repeterstöraren och behandlas på samma sätt som om det vore signaler från skrovekon. Detta betyder att en brusstörsändare även kan dölja signaler från en repeterstörsändare.

Radarstörsändaren kan störa både spaningsradar och eldledningsradar.

2.6.4 Konfigurera Radarstörsändare i NetScene

Två typer av radarstörsändare kan konfigureras i NetScene (*RadarNoiseJammer* och *RadarDRFMJammer*), se Figur 22. Radarstörsändarna bygger, precis som spaningsradarn, på parametrar från *PhysicalEntity*. Parametrarna som är specifika för störsändaren är markerade i blått i figuren.

För brusströraren anges frekvensbandet som ska störas genom parametrarna *CenterFrequency* och *BandWidth*. För repeterstöraren anges hur många falskmål som ska genereras, *NumOfTargets*. Parametern *AutoJam* saknar funktion vid konfigurering (visar om störaren är på eller av).



Figur 22 Parametrar för radarstörsändare. Brusstörare till vänster och repeterströrare till höger.

2.6.5 Radarvarnare

En modell av radarvarnare har tagits fram till EWSim. I radarvarnaren kan man ställa in tröskelnivå, frekvensområde, vinkelområde samt vinkelnoggrannhet. Varnaren interagerar med radarsignaler och om signalen kan detekteras tas data ur signalen fram. I dagsläget används varnaren i ett VMS där radarstörsändare automatiskt aktiveras om radarvarnaren ger varning. Radarvarnaren används alltså som en radarmottagare som tar emot radarsignalen så att den kan analyseras i störsändarens signalprocessor. Rent modellmässigt är radarvarnaren uppbyggd på samma sätt, d v s radarvarnaren är en radarmottagare, fast med ytterligare egenskaper.

Radarvarnaren kan aktiveras av radarsignaler från spaningsradar, eldledningsradar eller från radarstörsändare.

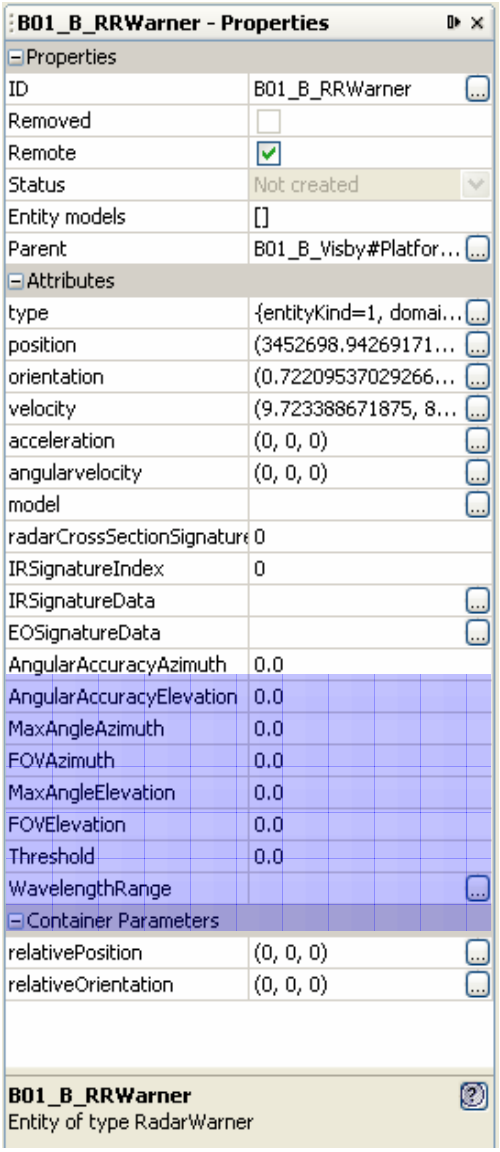
2.6.6 Konfigurera Radarvarnare i NetScene

I likhet med övriga radarmodeller bygger radarvarnaren (*RadarWarner*) på klassen *PhysicalEntity*. Parametrar specifika för radarvarnaren är markerade i blått i Figur 23.

Varnarens vinkelnoggrannhet ges av *AngularAccuracyAzimuth* och *AngularAccuracyElevation*. Den rapporterade riktningen mot hotet skiljer sig slumpmässigt mot den verkliga, inom dessa gränser (felet i en given riktning är maximalt halva den angivna noggrannheten). *MaxAngleAzimuth* och *FOVAzimuth* ger varnarens täckningsområde i bäring. Utanför detta område ges inga varningar. *MaxAngleElevation* och *FOVElevation* ger analogt täckningsområdet i elevation.

För varning krävs att effekten hos den inkommande radarsignalen överstiger varnarens tröskelnivå, *Threshold*. Dessutom måste radarsignalens våglängd stämma överens med varnarens våglängdsband som ges av parametern *WavelengthRange*. Våglängdsbandet anges i en textsträng med två rader för undre och övre gränsfrekvens. Exempel:

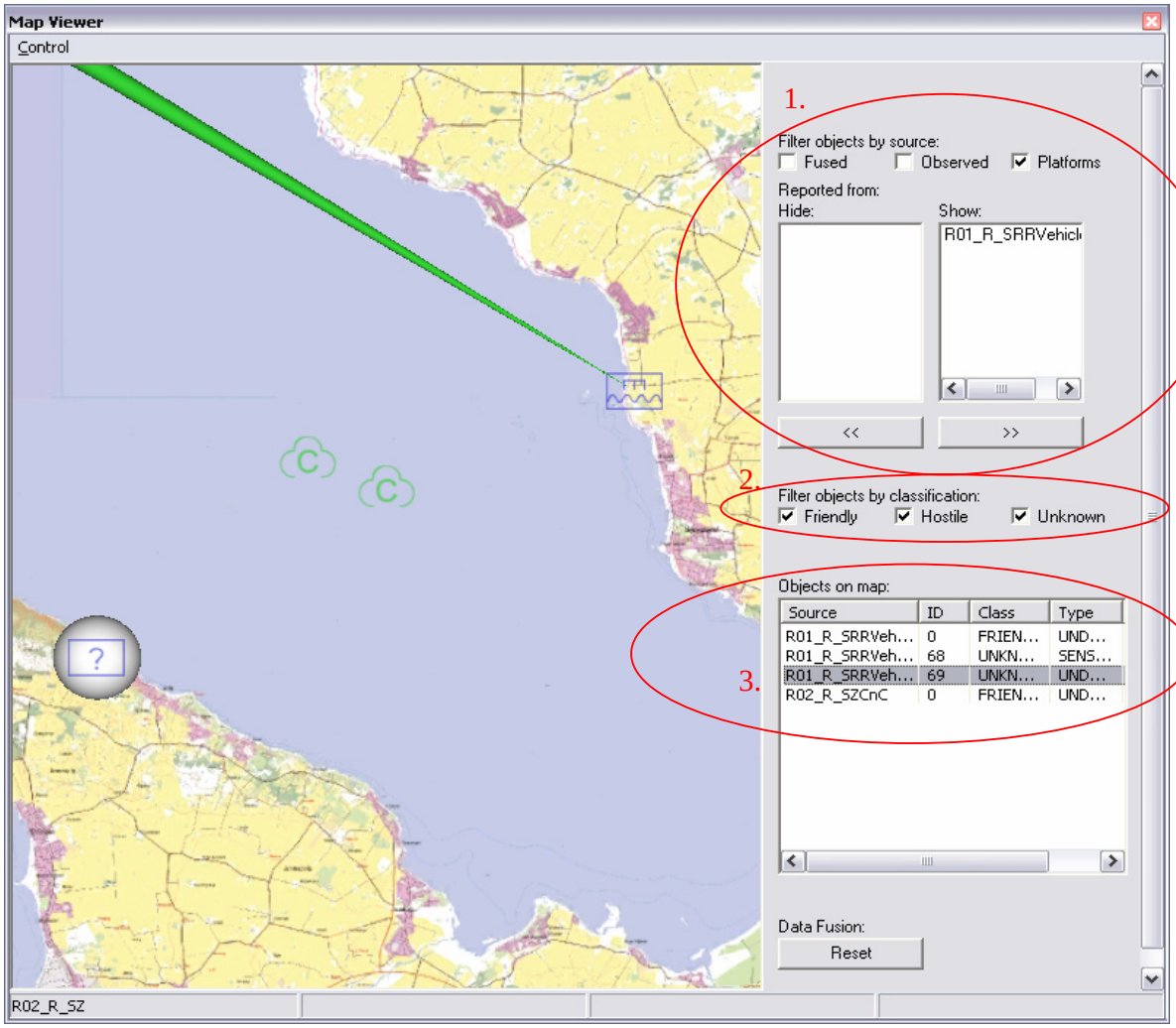
```
Min 0,5
Max 2
```



Figur 23 Parametrar för radarvarnare.

2.7 Ledningsvyn

För att starta ledningsvyn i EWSim välj i menyn *Control* → *Command and Control* eller tryck snabbkommando Alt-C. Om simuleringen inte är startad eller den för tillfället kontrollerade plattformen inte har en C2 enhet kommer en tom vy att visas. Om en giltig C2 enhet finns visas en kartvy centrerad över den kontrollerade plattformen. För att flytta kartvyn, klicka i kartan med vänster musknapp och flytta musen med knappen intryckt. För att ändra förstoring, tryck in mittenknappen alternativt Ctrl+vänster knapp och flytta musen under tiden knapparna hålls inne. Till höger finns en meny med inställningar för vad som ska visas och hur informationen ska filtreras. Vid högerklick i kartan visas menyn för att skicka meddelanden till markerade enheter.



Figur 24 Ledningsvy i EWSim

2.7.1 Filtrering av data

Data kan filtreras antingen på källa eller klassificering eller kombinationer av båda. I Figur 24 återfinns källfiltreringen i markering 1, medan filtrering på klass finns i 2. Källa kan vara egen observerad lägesbild, inrapporterade positioner för egna sidans plattformar eller inrapporterade lägesbilder. Utöver dessa kan även en fusionerad lägesbild visas där data både

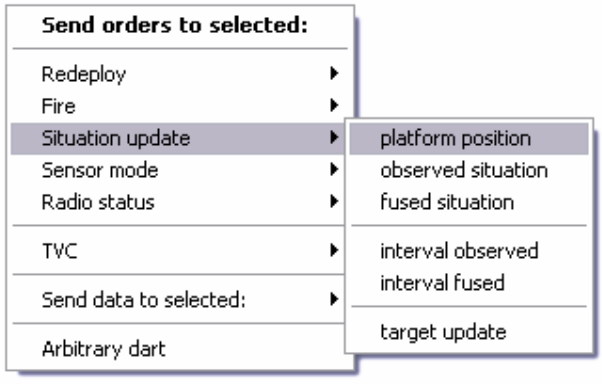
från egna och rapporterade observationer sammanfogas till en gemensam lägesbild. För rapporterade lägesbilder finns två listor, en för visade och en för dolda. För att flytta rapporterade lägesbilder mellan listorna används pilknapparna under listorna. För att visa eller dölja egen observerad bild samt plattformar används kryssrutor. Detsamma gäller den fusionerade lägesbilden.

2.7.2 Objekt på kartan

Data som stämmer in på vald filtrering visas som objekt på kartan. Dessa objekt är antingen positioner eller bäringar. Positioner visas som ikoner enligt NATO-standard, där både färger och symboler är beroende av målidentifiering och -klassificering. Bäringar visas med en bredd som motsvarar riktningens osäkerhet. Alla målobjekt som visas på kartan återfinns även i en lista i menyn. I Figur 24 återfinns denna lista i 3. Genom att klicka på objekt i kartan med vänster musknapp kan motsvarande objekt i listan markeras. Med shift och/eller ctrl kan flera objekt markeras och avmarkeras i listan. Genom att klicka på ett objekt i kartan läggs det till som markerat i listan medan övriga markeringar kvarstår. Utöver inrapporterade objekt visas alltid en ikon för egen position på kartan.

2.7.3 Skicka meddelanden

Genom att högerklicka i kartan öppnas snabbmenyn för att skicka meddelanden.



Figur 25 Snabbmeny för meddelanden

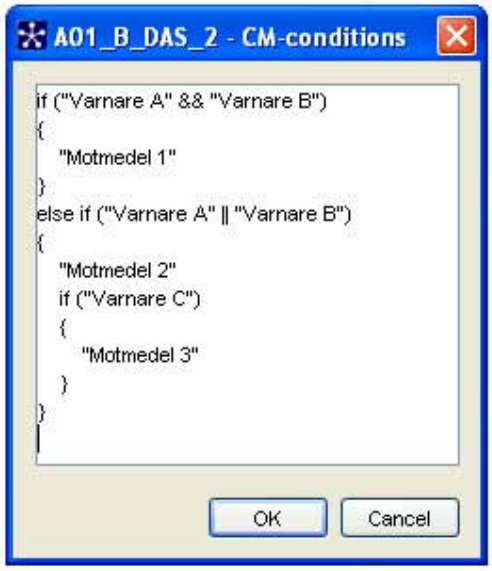
Merparten av de meddelanden som finns i snabbmenyn är order till mottagaren att agera på något sätt. Dessa återfinns som ”*Send orders to selected:”*. I Figur 25 visas ett exempel där en sådan order är markerad. Just detta meddelande betyder att man skickar en order om att mottagaren ska skicka tillbaka information om sin position. Om man istället vill skicka sitt eget läge till någon återfinns detta i undermenyn ”*Send data to selected:*”. Om meddelandet kräver indata från användaren öppnas ett GUI[4] som beskriver vad som ska anges. Detta kan t.ex. vara att sätta intervalltid eller att markera positioner på kartan. Meddelanden som inte kräver indata från användaren skickas direkt.
Längst ner i snabbmenyn finns ”*Arbitrary dart*” som öppnar menyn där samtliga meddelanden finns i klartext. Medelandena kallades för Dart i tidiga versioner av programet men heter numera FFM.
Om meddelanden skickas på det sätt som beskrivs ovan skickas de till samtliga enheter som finns markerade i objektlistan i Figur 24, markering 3. Om ett mål markerats (till skillnad från en plattform) kommer meddelandet att skickas till den plattform som har rapporterat målet. För att skicka till alla i eget lag finns i ledningsvyns huvudmeny *Control→Send FFM to all team* som skickar meddelande till samtliga enheter på egen sida.

[4] Graphical User Interface – grafiskt användargränssnitt som tillåter användaren att interagera med programmet

De meddelanden som finns att skicka finns i appendix 0.

2.7.4 Dynamiskt Duellverktyg EWSim automatiserade Regler

I EWSim sker en logisk sammanvägning av registrerade händelser för att generera en lämplig åtgärd. Denna logik är konfigurerbar i NetScene. Logiken ges i form av en textsträng som kopplar ihop id för varnare med id för motmedel i en logisk satsstruktur (lik den i C++). Ett exempel på detta ges i Figur 26.



Figur 26 Varnare och motmedel kopplas ihop i en logisk struktur.

Satserna avkodas med speciella nyckelord:
`if, else if, else, { }, " ", &&, ||, !, ( )`

`if` och `else if` testar logiska uttryck som ska omges av parenteser och byggs upp av:

- Id för varnare som anges inom `" "`. När det logiska uttrycket evalueras omvandlas dessa till värdena *sant* eller *falskt*. Om varnaren finns på plattformen och har genererat en varning så genereras värdet sant, annars falskt.
- De logiska operatorerna OCH (`&&`), ELLER (`||`) och NOT (`!`), vilka är logiska operationer i enlighet med den boolska algebran.
- Parenteser som visar i vilken ordning uttryck ska evalueras. Om inga parenteser anges har `!` högst prioritet, följt av `&&` och sist `||`.

I exemplet i Figur 26 är den första if-satsen sann om varnarna med id ”Varnare A” och ”Varnare B” tillhör den aktuella plattformen och har gett varning från samma riktning.

Om ett givet logiskt uttryck är sant genomförs operationerna inom klamrarna { } som ska följa direkt efter. I exemplet kommer motmedel med id ”Motmedel 1” att aktiveras (såvida det finns ett sådant motmedel på den aktuella plattformen). På samma sätt aktiveras ”Motmedel 2” om någon av ”Varnare A” eller ”Varnare B” gett varning. ”Motmedel 3” aktiveras om dessutom ”Varnare C” varnat.

Här följer ett exempel på hur ett C2Condition kan se ut.

```
if ("POSITION_REACHED"  "POS"  "1302470"  "6257170" "RETRIGGERTIME" "5")
{
        "SEND_MESSAGE_TO" "OBJ" "B01_R_C2_FightingVehicle"
         "FORMAT" "444"
         {
            "SNABBGRUPPERING*
             FROM:     *U:
             MOT*X: 1 *Y: 1 *
             MOT:
             TEXT:
              ------SLUT------|"
         }

         "SEND_MESSAGE_TO" "OBJ" "B01_R_C2_FightingVehicle"
         "FORMAT" "444"
         {
            "SNABBGRUPPERING*
             FROM:     *U:
             MOT*X: 1 *Y: 1 *
             MOT:
             TEXT:
              ------SLUT------|"
         }
}
```

2.8 Nätverkskommunikation

Kommunikation i LKS simuleras i en nätverksfederat kallad ComNet. Här simuleras både nätverksbildning och själva signalöverföringen. Nätverkssimuleringen innefattar det digitala datanätet där meddelanden kan routas via andra noder i nätverket för att kunna nå den slutliga destinationen. Den direkta signalöverföringen mellan två noder kan ske via antingen kabel eller radio. Även system för spaning och störning av kommunikationen simuleras i ComNet.

2.8.1 Vågutbredning

Ett radiomeddelande överförs genom att modulera en bärvåg. Några exempel på olika modulationstyper är FM (frekvensmodulering) och AM (amplitudmodulering). För att försvåra upptäckt, avlyssning och störning kan sändningen utnyttja någon bandspridningsteknik, t ex frekvenshopp eller direktsekvensspridning.

Det begrepp som oftast används för att fastställa om en kommunikationslänk är av tillräcklig kvalitet kallas SNR (Signal-to-Noise Ratio, *signal till brus förhållande*). Detta är kvoten mellan i mottagaren mottagen signaleffekt (S) och i mottagaren närvarande brus (N). Bruset härrör från många olika källor, men kan delas in i externt brus som leds in via antennen och internt brus som genereras i mottagaren.

De frekvensband som normalt används för taktisk radiokommunikation inom Försvarsmakten är kortvågsbandet (HF, 1.6-30 MHz) och truppradiobandet (VHF, 30-88 MHz). För denna kommunikation finns utrustning som stödjer både röst- och datasändningar, t ex Ra763 (HF) och Ra180 (VHF). Se Figur 27.

Figur 27 Radio 180, för röst- och datakommunikation på truppradiobandet.

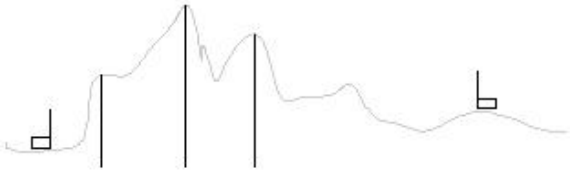
Radiomodellen i LKS är generisk. Vilken verklig radio den motsvarar bestäms av satta parametervärden. Modellen konfigureras i NetScene, och styrs sedan under simuleringens gång från radiosystemets användargränssnitt i LKS. Den funktion som hittills simulerats motsvarar en Ra180 med fixfrekvenssändning.

I det simulerade radiosystemet sker beräkning av SNR i mottagaren. Den starkaste sändaren med parametrar som överensstämmer med mottagaren räknas som nyttosignal. Alla andra sändare som på något sätt påverkar mottagaren ses som störningar som summeras ihop med det allmänna bakgrundsbruset till en bruseffekt.

I dagsläget finns två möjligheter att beräkna vågutbredning för kommunikation. Den ena är en enkel frirymdsberäkning där hänsyn endast tas till frekvens och avstånd mellan sändare och mottagare. Det är denna modell som används för närvarande. I den andra mer avancerade modellen används delar av vågutbredningsbiblioteket Detvag-90®[DETVAG]. Detvag-90 har utvecklats på Institutionen för Informationsöverföring på FOI och används bland annat i WRAP [WRAP] och Freke-Tavast [F-T]. Den Detvag-modell som används i EWSIM är kvadratrotsmodellen. Modellen kombinerar en sfärisk-jord-modell med en kniveggsmodell (3 eggar) och tar hänsyn till både höjd- och terrängdata, se Figur 28. Modellens giltighet är till viss del beroende av antennhöjder och terrängens beskaffenhet, men kan generellt sägas ligga mellan några MHz upp till några GHz.

Terrängdata översätts under beräkningen till konduktivitet och relativ dielektrisk permittivitet. Datat läses in i datorns primärminne då en simulering startas, vilket gör att beräkningarna är mycket snabba, c:a 10 ms för en 10 km förbindelse med 50 m upplösning.

I modellen ingår endast beräkningar längs med en terrängprofil. Flervägsutbredning via t ex studsar i bergssidor tas inte hänsyn till.



Figur 28 Kniveggsmodellen med de tre dominerande eggarna i en terrängprofil.

2.8.2 Nätverk

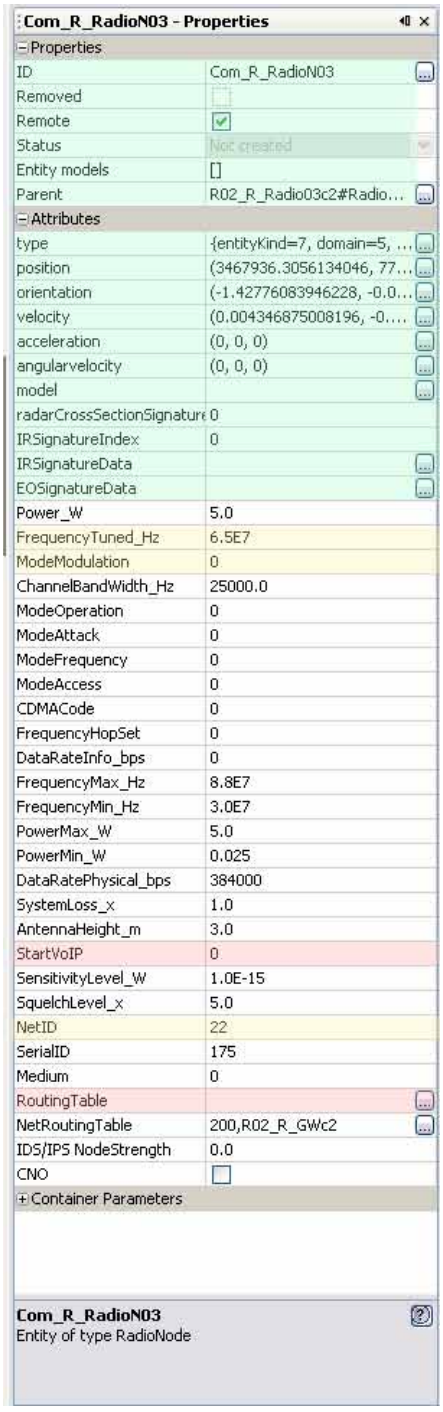
Med flera radioapparater kan man skapa ett nätverk för att förmedla information. Denna information kan bestå av t.ex. lägesrapporter för ledningssystemet. Då information som skickas och tas emot är avsedd för en C2-enhet krävs att radion placeras som ett underobjekt till C2-enheten, se Figur 30. Inga sändningskrockar antas ske inom nätet utan all kommunikation sker utan kollision via accessprotokollet TDMA (Time Division Multiple Access, används av t ex GSM). Dock kan sändningskrockar ske med sändare i andra nät.

Alla noder i nätet fungerar som router vilket betyder att sändare och mottagere inte nödvändigtvis behöver ha direktkontakt med varandra. Om direktkontakt saknas kommer meddelanden att automatiskt dirigeras om i nätet så att de via andra noder når den slutliga mottageren. Inga inställningar behöver alltså göras för att ställa in hur ruttberäkningen ska ske inom ett nät.

Om man vill koppla samman olika nät med varandra så att noder i ett nät kan kommunicera med noder i ett annat kan man använda sig av en gateway. För att skapa en gateway används två (eller flera) radioapparater som läggs under samma C2. Varje apparat ställs in för att ansluta till ett visst nät men tillåter även att meddelanden kan ruttas mellan apparaterna och därmed näten (tas emot i den ena radion och sändas ut i en annan). De blir noder i varsitt nät, men tillsammans skapar de en gateway.

2.8.3 Kommunikationsinställningar i NetScene

Radiosystemen bygger på parametrar från *PhysicalEntity*, dessa är grönmarkerade i figuren nedan. Parametrar som inte sätts från *PhysicalEntity* är bland annat frekvens, uteffekt samt modulation. Dessa kan ändras via användargränssnittet i Netscene som ses i Figur 29. Rödmarkerade parametrar används för närvarande inte.



Figur 29 Parametrar för kommunikationsutrustning.

För att en radio ska kunna kommunicera med en annan radio måste parametrar markerade gula vara identiska. Apparater med samma nätID (*NetID*) tillhör samma nät och kan kommunicera med varandra.

Övriga parametrar som påverkar radions räckvidd är uteffekt (*Power*_W), kanalbandbredd (*ChannelBandWith_Hz*), systemförluster (*SystemLoss*_x) samt antennhöjd (*AntennaHeight*). Observera att alla parametrar anges linjärt och inte i dB.

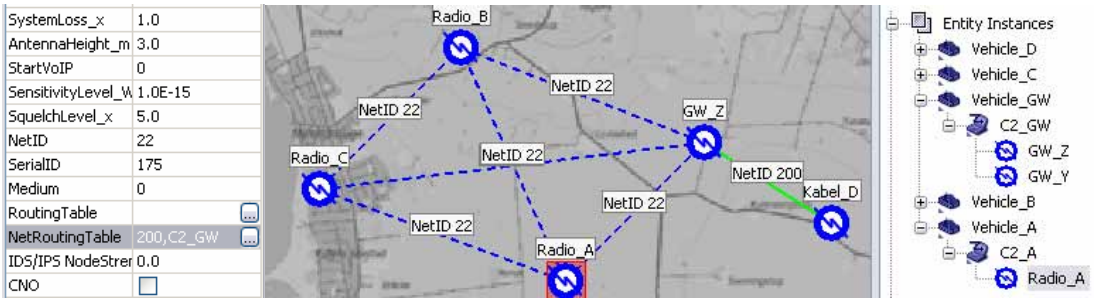
Modulationsparametern (*ModeModulation*) anger om radion ska använda FM(0) eller AM(1).

Parametern *Medium* anger om radion ska kommunicera via eter(0) eller kabel(1). Förutom oändlig räckvidd och att en kabel ej kan störas fungerar kabel på samma sätt som radiotrafiken via eter.

IDS/IPS NodeStreangth är ett mått på hur bra CNO-skyddet mot intrång och attacker är. Värdet kan varieras mellan 0 och 1. För att kunna använda radion för CNO-attacker mot fientligt nät ska *CNO* kryssas i. Mer om CNO beskrivs i 2.8.8 CNO.

Om en radio ska kunna kommunicera med en radio i ett annat nät än sitt eget måste det ske via en gateway, även om de har samma frekvens och modulation. För att ange vilket gateway som ska utnyttjas används paramterern *NetRoutingTable*. Parametern anges på formatet *nät,gateway* där nät anger vilket nät man vill nå och gateway vilken C2-enhet som ska agera gateway för att kunna nå detta nät. I Figur 30 visas hur detta kan se ut.

Nod A som tillhör nät 22 vill prata med nod D som tillhör nät 200, eftersom de har olika nätID kan de inte prata med varandra direkt. Istället måste kommunicationen gå via en gateway. Nod GW har två radioapparater, en i nät 22 och en i nät 200. Nod GW kan därför användas som gateway. För att nod A ska kunna nå nod D ska *NetRoutingTable* parametern i A sättas enligt följande: “200, C2_GW”, 200 för nätnummer som ska nås och C2_GW för den C2 som ska agera gateway för att nå detta nät.



Figur 30 Nod konfigurerad för att kunna nå ett externt nät via gateway.

Observera att detta attribut sätts per nod. Om nod A vill kommunicera med ytterligare ett nät (330) ska *NetRoutingTable* skrivas “200, C2_GW;330,C2_GW3”.

Det finns även möjlighet att sätta en default gateway. När en default gateway är specificerad används den för alla meddelanden som inte är till det egna nätet eller till nät med redan specificerad gateway. För att markera en gateway som default sätts ”-1” istället för nätnummer.”2,B;-1,C” betyder alltså att meddelanden till nät 2 ska använda nod B som gateway medan meddelanden till alla andra externa nät kommer att använda sig av nod C.

2.8.4 Signalspaning

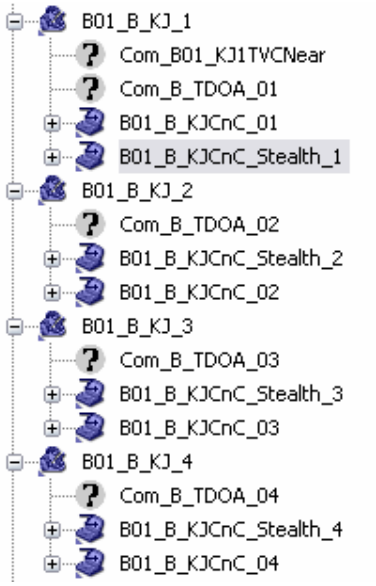
För närvarande finns det två simulerade varianter av pejlsystem i EWSim, det ena är det traditionella pejlsystemet (DF_Classic) som levererar bäringar efter att ha mätt upp den infallande vågfrontens vinkel i förhållande till antennens kalibrerade norrpunkt. Detta system kräver minst två samarbetande pejlar för att kunna fastställa en position för en uppspanad sändare. En position kan fås när en datafusion har gjorts över två systems bäringar. Där bäringarna korsar varandra bildas ett så kallat pejlkryss/position. Kvalitén på levererade bäringar beror till stor del av den mottagna signalens styrka, eller rättare sagt signalens SNR-värde. Ett lågt SNR-värde medför vinkelfel och kan på stora avstånd resultera i ganska stora avvikelser i riktning/position. I EWSim används uteslutande SNR-värdet för att fastställa vinkelfelet.

Det andra pejlsystemet använder tidsmätning för att fastställa signalkällans position, så kallad Time Difference of Arrival teknik (TDOA). Minst tre stycken pejlar/receptorer behövs för att fastställa en position, pejlarna bör vara åtskilda i rummet (1 – några kilometer) beroende på terräng. Signalens ankomsttid i pejlarna räknat från källan kommer att variera och det är den variationen som ligger till grund för en positionsberäkning. EWSim använder inte den tidsupplösning som behövs för att simulera tidsfördröjda signaler, här används istället principen minst tre stycken pejlar (efter beräkning) uppfatta en viss sändare samtidigt, med viss signalstyrka och visst SNR så kan den också lägesbestämma sändaren med viss nogrannhet. Nogrannheten i EWSim beror på storleken av SNR-värdet och signalstyrkan. Fler system än tre kan användas men det är bara de tre mottagare som har de starkaste signalerna som används för lägesbestämningen.

Båda pejlsystemen styrs av en kontrollenhet kallad Televapencentral (TVC). Det klassiska pejlsystemet styrs av en enhet som heter TVC_Far medans TDOA-systemet styrs av en enhet som heter TVC_Near. Det är TVCns uppgift att sammanställa och bearbeta pejldata och sedan skicka detta vidare till ledningssystemet.

2.8.5 Signalspaningsinställningar i NetScene

Ett TDOA-system definieras enligt Figur 31. Här finns fyra stycken TDOA-eneheter (Com_B_TDOA01 – 04), en TVC_Near (Com_B01_KJ1TVCNear) och fyra stycken C2-enheter för datakommunikation mellan TDOA-enheterna och TVCn (B01_B_KJCnC_Stealth_1 – 4).



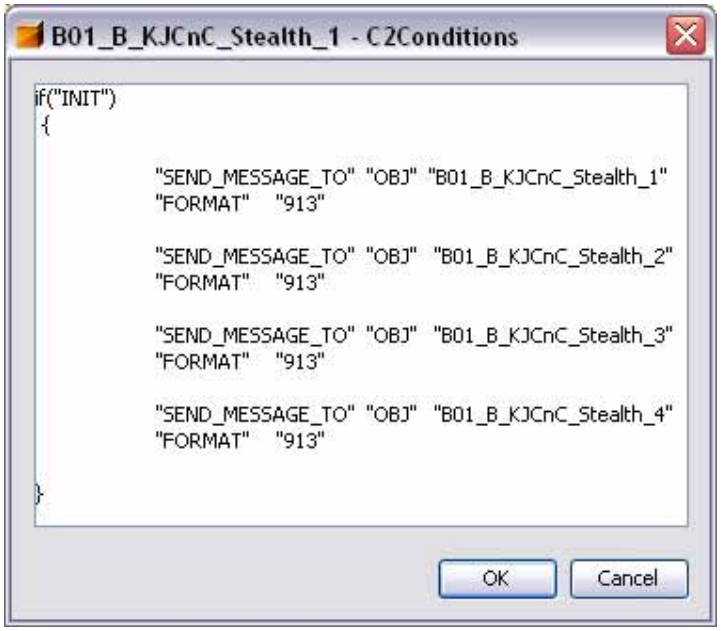
Figur 31 TDOA system.

TVCn måste veta namnet på TDOA-enheternas C2, detta specificeras i TVCns parameterinställning i NetScene enligt Figur 32.

SquelchLevel_x	5.0
NetID	55
SerialID	111
Medium	0
Subunit1	B01_B_KJCnC_Stealth_1
Subunit2	B01_B_KJCnC_Stealth_2
Subunit3	B01_B_KJCnC_Stealth_3
Subunit4	B01_B_KJCnC_Stealth_4

Figur 32 Parametrar för TVC.

TDOA-enheterna måste också veta till vilken C2 de skall rapportera uppspanat data, detta görs genom ett så kallat C2-condition som specificeras i den C2 man vill använda och som sitter på samma plattform som TVCn. I detta fall är det för B01_B_KJCnC_Stealth_1 som vi skall specificera ett C2-condition, se nedan. Meddelandet specificerar också vilken C2 som TDOA-enheterna får använda i sin rapportering.

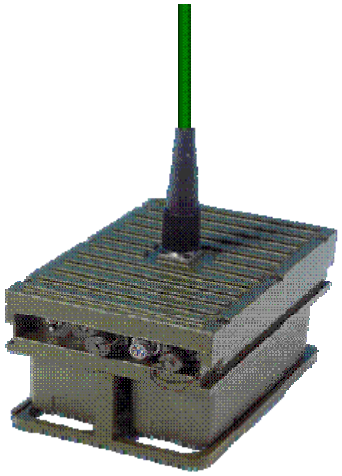


Figur 33 C2-condition för att beordra TDOA att skicka data till TVC.

I detta exempel kommer TDOA-eneheterna att skicka data till TVCn som sammanställer det. Är det tillräckligt med data för en viss sändning kommer TVCn att rapportera detta till sin plattforms ledningsvy. Denna kan visa informationen som positioner eller bäringar. Vill en stab ha denna information får den begära detta genom ett meddelande (*916 Send sigint info to me*) till TVCns C2. Staben kan då förutom bäringar och positioner visa hyperblar som beskriver vilka enheter som uppfattat vad, samt frekvenser och sändarens nättillhörighet.

2.8.6 Radiostörning

Radiostörsystem används för att störa motståndarens radiokommunikation och kan delas in i två huvudtyper, när- och fjärrstörsändare. Närstöraren har begränsad effekt och livslängd. Den står nära sitt mål och stör rundstrålande. Fjärrstöraren står långt borta och använder riktantenner. Den har i princip obegränsad effekt och livslängd. Störningen kan i båda fallen antingen ske bredbandigt med vitt brus eller kanalvis (smalbandigt). Exempel på verkliga system för fjärr- och närstörsändare är MOPS (Modulär Plattform för Störning) respektive TARAX [TARAX]. Se Figur 34.



Figur 34 Närstörsändaren TARAX.

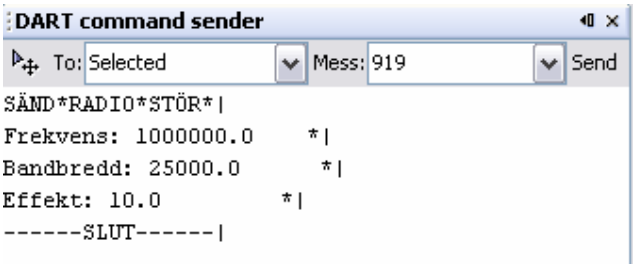
Ett radiostörsystem är egentligen bara en särskild typ av radiosystem. Den generiska modellen för ett störsystem är densamma som för ett vanligt radiosystem. Funktionen styrs av parametervärden, t ex att modulationen är vitt brus istället för FM. Det radiostörsystem som hittills simuleras i EWSim har via parametervärden satts att efterlikna en förenklad TARAX med förmåga att störa bredbandigt med vitt brus inom frekvensområdet 30-88 MHz. Systemet konfigureras i NetScene, och styrs sedan under simuleringens gång via meddelanden i ledningssystemet.

Rent simuleringstekniskt kan en störare betraktas som en vanlig radiosändare. Skillnaden ligger endast i att den transmitterade effekten från en störsändare innehåller brus istället för information. Störsignalen kommer därför endast att bidra till N-delen vid beräknandet av en kommunikationslänks SNR-värde. I mottagaren vägs alltså samtliga sändningar och störsändningar ihop vid beräknandet av SNR-värdet.

2.8.7 Konfigurera Radiostörsystem i NetScene

En plattform som skall ha störförmåga måste ha ett TDOA- eller Jammer-objekt. I princip behöver man inte ställa in dess parametrar utan det görs via en störorder från ledningssystemet. Idag kommer inga radionoder att störas ut som har samma nätId som störobjektet. Vill man simulera den effekten skall radionoder och störobjekt ha olika nätId.

För att starta en radiostörning används meddelandet i Figur 35 Störorder i LKSStab.



Figur 35 Störorder i LKSStab.

I det här fallet kommer den valda störaren att störa på frekvens 1MHz med en bandbredd på 25kHz. Effekten kommer att ligga på halva bandbredden ovanför centerfrekvensen 1MHz och halva under, dvs 975000Hz – 1025000Hz. Störeffekten är 10W.

För att avbryta radiostörningen skickas meddelandet 921, Radiostör av.

2.8.8 CNO

Med CNO(Computer Network Operations) kan attacker/spaning utföras mot det simulerade nätverket. Det behövs en radio som ska kunna anslutas till det fientliga nätet som ska attackeras. Radion är av samma typ som de som används till det egna nätverket men alternativet *CNO* ska kryssas i, se Figur 29 Parametrar för kommunikationsutrustning. CNO kan genomföras i ordningen inkoppling till det fientliga nätet, passiv spaning, aktiv spaning och attack.

Informationen om villka nätverksnoder som finns och kan attackeras fås genom den egna SIS pejlen och visualiseras för användaren med symboler i de kartvyer som finns i Netscene och Stabsverktyget. För en beskrivning av de symboler som finns och hur de tolkas se kapitlet 2.9 om stabsverktyget.

En passiv spaning kan som namnet antyder inte upptäckas av fienden och är ett nödvändigt första steg för att kunna utföra mer avancerade attacker. När passiv spaning aktiveras för en nätverksnod som har den CNO förmågan så sker automatisk en inkoppling till fientligt nät. Inkopplingen innebär att den egna noden tilldelas samma id (motsvara ip nummer) och nätid (motsvara nätmask) som den fiendenod användaren väljer att attackera. Obervera att det förutsätts att den radio som är ansluten till nätverksnoden har alla sina parametrar som t.ex. frekvens förinställda i scenariot då dessa inte sätts i denna version. Det är endast den av användaren valda fiendenodens information som överförs till den attackerande nätverksnoden.

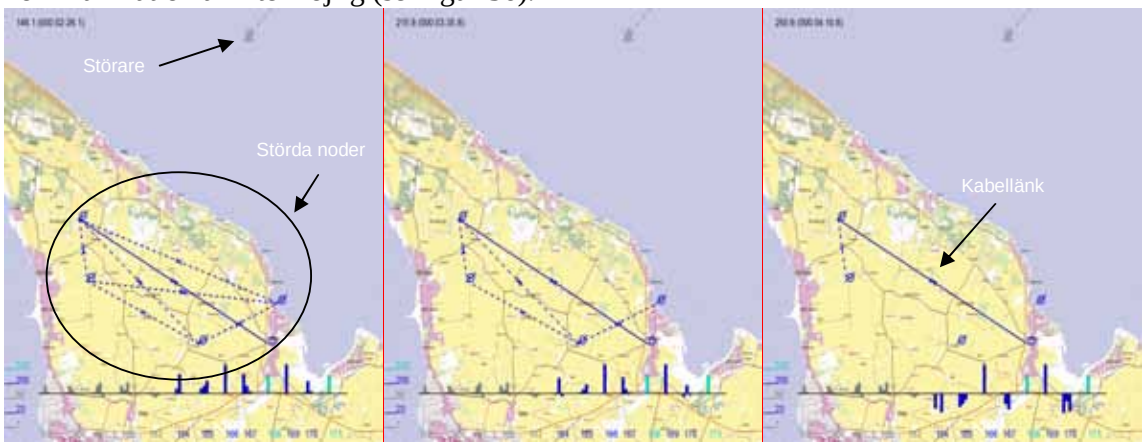
Nästa CNO åtgärd som är möjlig är aktiv spaning. Till skillnad från den passiva så finns det i normala fall en ganska så stor möjlighet att den upptäckas av den som attackeras. En upptäckt visas dock inte i denna version för den som utsätts för attacken, dvs. det gör att även aktiv spaning förbli oupptäckt för den som attackeras. En aktiv spaning utförs för att kunna få information om nätverksnodens svagheter. Den informationen kan sedan användas för att göra ytterligare attacker.

När information om en fiendes nätverksnod har inhämtats så kan användaren välja att utföra en *ping of death* attack, som är en typ av DoS attack, mot den nätverksnoden. Denna typ av attack kan få en mycket kraftig inverkan på fiendens möjlighet att kommunicera då effekten är att all inkommande och utgående nätverkstrafik blockeras under en kortare tid i den attackerade noden. Även all trafik som i normala fall routas via den attackerade noden blockeras. Detta är en simulering av att den tänkta datorn som nätverksnoden består av måste

starts om. Tiden det tar att starta om datorn är förinställd på 2 minuter. Den som utsätts för den här typen av attack få ingen information om det men då kommunikationen mer eller mindre slås ut så får det effekten att ordrar som skickas inte alltid kommer fram. Det innebär också att information från sensorer inte alltid kommer fram till staben. Det är alltså ett sätt att förblinda fienden så att den inte kan se vad som händer men den som är observant förstår oftast att något är på gång. Problemet för den som attackerar är att den inte säkert vet vilken effekt attacken får då det kan vara svårt att avgöra hur viktig och vilken funktion den attackerade nätverksnoden har.

2.8.9 Visualisering

Under en simulering visas i ComNet en visuell bild av hur det simulerade nätet används. Länkar mellan noder visas med streckade linjer för radiolänkar och heldragna linjer för kabellänkar. Mitt på länklinjerna visas i vilken riktning som länken går, >< markerar att kommunikation kan gå åt båda håll och > eller < visar att länken bara går åt ena hållet. Längst ner i ComNet-bilden visas ett diagram med störbrusförhållandet för alla länkar. Detta diagram är grupperat så att alla möjliga länkar inom ett nät till en mottagare ligger intill varandra. Staplar som är riktade uppåt (från den svarta horisontella linjen) motsvarar en länk där signalnivån är högre än brusnivån och kommunikation är möjlig (högre stapel motsvarar bättre länk) och en stapel riktad neråt innebär att brusnivån är starkare än signalnivån och kommunikation är inte möjlig (se Figur 36).



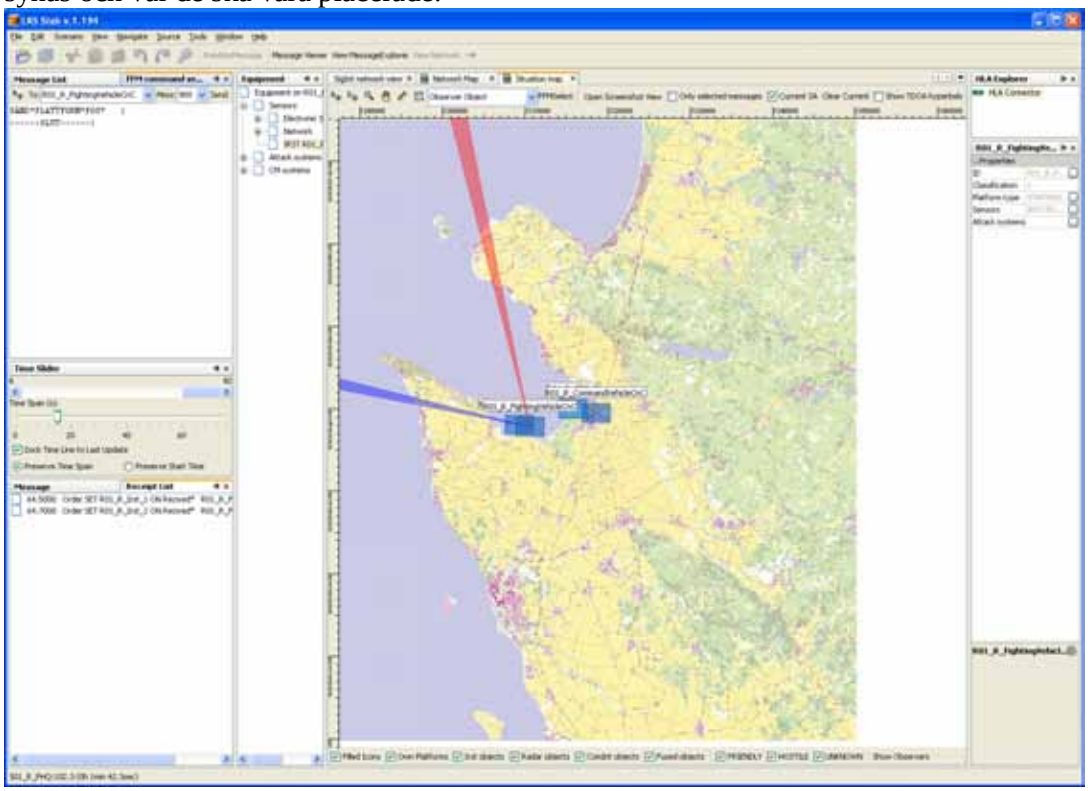
Figur 36 Bilder som visar hur länkar mellan noder i ett nätverk förändras när störningen går från att vara avslagen (vänster) till att öka successivt (höger).

När en kommunikationsstörare är aktiv visualiseras detta genom att symbolen för störaren växlar färg (med en period på två simulerade sekunder). Det går också att se när en nod är påverkad av en CNO attack genom att storleken på den utsatta nodens symbol pulserar (också detta med en period på två simulerade sekunder).
Under en simulering när meddelanden och data skickas över en kommunikationslänk visualiseras detta i ComNet med att en cirkel rör sig längs länken (från sändaren till mottagaren) samt att staplar dyker upp i anslutning till nodsymbolerna (uppåtriktad stapel vid sändande nod och nedåtriktad stapel vid mottagande nod). För att detta skall uppfattas tar det en simulerad sekund för cirkeln att komma fram till sin destination och staplarna visas också under en simulerad sekund.

2.9 LKSStab verktyget

LKSStab är ett stabsverktyg utvecklat som en fristående applikation för att vara den del i en dynamisk simulering som används av en grupp försökspersoner som agerar ledningsgrupp. Under en simulering körs den typiskt i en federation där det även ingår en scenariomotor (Netscene, ref), nätverkssimulator (comnet) och ett eller flera EWSim-federat. Detta avsnitt presenterar kort hur LKSStab används. För en utförligare beskrivning av olika vyer och funktioner, se [STAB].

Figur 37 visar ett typiskt utseende hos LKSStab. I mitten visas en kartvy och runt omkring finns ett antal olika fönster med egenskaper, utrustningslista, mottagna rapporter och andra funktioner. Applikationen bygger på NetBeans [NB], och arvet därifrån gör att vyerna är mycket anpassningsbara och användaren kan själv till stor del välja vilka fönster som ska synas och var de ska vara placerade.



Figur 37 Applikationen LKSStab

Längst ner till vänster finns en klocka som talar om hur länge en simulering varit igång. Denna simulering kan även vara en uppspelning av en loggad simulering. LKSStab kontrollerar själv om en logger styr en uppspelning och skiftar automatiskt mellan att simulera och spela upp ett scenario. De flesta inställningar av hur informationen ska presenteras fungerar likadant i båda fallen.

Man kan välja att när som helst under en simulering spara aktuell situation med historik genom att använda *File – Save simulation.* Alla inkomna meddelanden sparas då som en textfil med ändelsen .ssm. Dessa filer kan sedan öppnas i LKSStab med hjälp av *File – Open Simulation*.

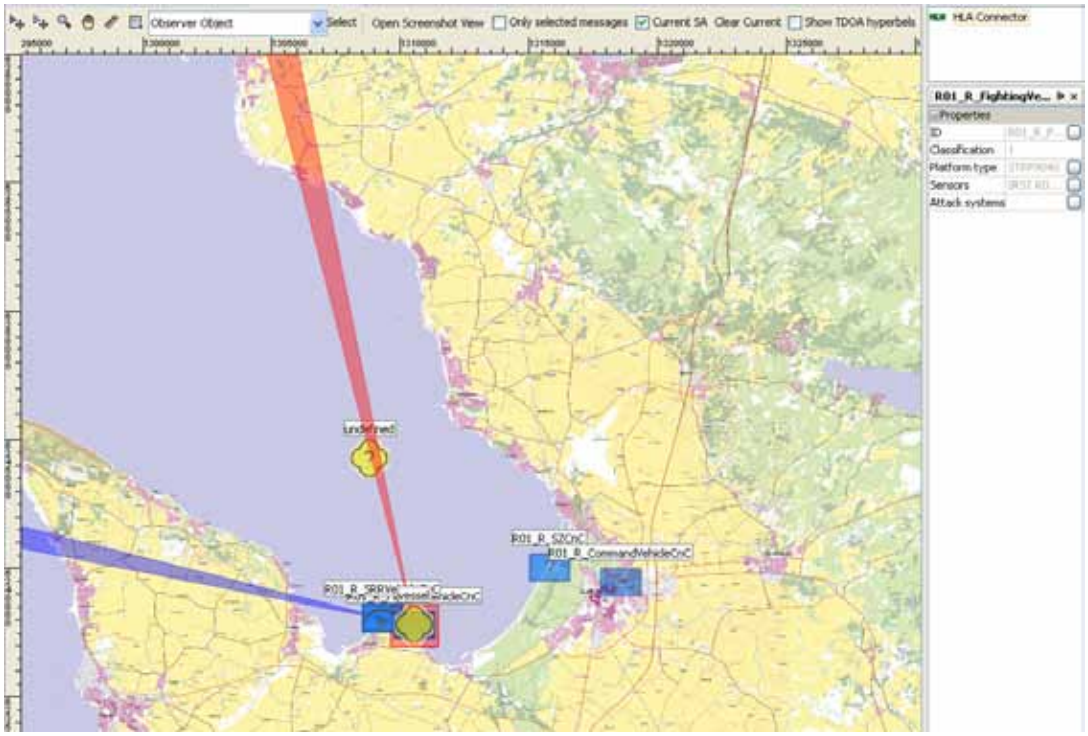
LKSStabs huvudsakliga uppgift är presentera en lägesbild för användaren samt att tillhandahålla ett gränssnitt för att order och rapporter ska kunna skickas och tas emot. Lägesbilden byggs upp av rapporter som tas emot via FFM från sensorer och andra enheter som simuleras i EWSim-federaten. Lägesbilden presenteras främst genom uppritning av lägesinformation i en kartbild, men användaren har också ett antal andra vyer till sitt förfogande.

De båda kartvyerna *Situation Map* och *Network Map* öppnas automatiskt då en karta väljs. Detta kan man göra genom *Open map* som finns under *File*. *Open Map* finns också som en genväg längst till vänster i programmets toolbar. Intressanta fönster utöver kartvyerna är:

- Properties: Fönster som visar egenskaper för ett markerat objekt.
- Logical Network View: visar logisk nätverksvy, uppspanat läge med information från signalspaning och CNO.
- Network Property Filter: bestämmer vilka egenskaper som ska visas i nätverksvyerna. Vyn är inte så användbar för tillfället, då nätverksnoder inte har så många olika egenskaper, men finns ändå med för framtida behov.
- HLAExplorer: Visar HLA connector som låter användaren ansluta till federation manager.
- Equipment: Visar den utrustning som en markerad enhet har rapporterat att den har.
- FFM Order: (DART Order i versioner tidigare än 1.195) visar ett fönster varifrån användaren kan skicka FFM-meddelanden till andra spelare i samma lag.
- Time Slider: Fönster där användaren kan styra över vilket tidsintervall som ska visas i lägesvy och inkomna meddelanden.
- View MessageExplorer (finns även som genväg i toolbar): Visar en lista på inkomna FFM-meddelanden.
- View Receipts: Visar inkomna kvittenser
- Message Viewer (finns även som genväg i toolbar): Fönster där texten hos ett valt FFM-meddelande visas.

2.9.1 Situation Map

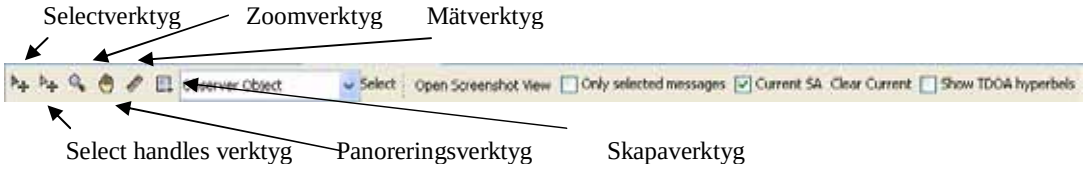
Situation Map visar aktuell lägesbild i en kartvy. Här visas en grafisk representation av information som kommit in via FFM, exempelvis positioner inrapporterade av egna plattformar, målbäringar och positioner inrapporterade av olika sensorer.



Figur 38 Lägesvy i LKSStab med inrapporterade positioner och bäringar.

Hur en lägesbild kan se ut framgår av Figur 38. Positioner är utmärkta med ikoner som följer NATO-standard APP-6a efter typ av objekt i den grad de går att identifiera och klassificera. Ikonuppsättningen finns i en fylld och en ofylld variant, och användaren kan byta mellan de båda för att i olika fall få högsta läsbarhet av lägesbilden. Bäringar är utmärkta med en riktningsanvisning utgående från rapporterande sensor. Färgen hos ikoner och bäringar bestäms av om de klassificeras som *friendly* (blå), *hostile* (röd), *neutral* (grön) eller *unknown* (gul). Hur mycket information som ska visas styrs via *Time Slider* och checkboxar med filterval.

I vyns överkant finns en toolbar enligt Figur 39. Här finns olika verktyg och checkboxar som alla påverkar lägesvyns utseende.



Figur 39 Övre toolbar för *Situation Map*.

Verktygen till vänster används för att markera objekt, panorera, zooma och mäta i kartan. Med hjälp av skapaverktyget kan man skapa en Observer Node, som används för att markera intressanta objekt och föra anteckningar om dem. Knappen *Open Screenshot View* öppnar en ny vy med en ögonblicksbild av hur läget ser ut och vissa ritmöjligheter. *Screenshot*vyns toolbar innehåller verktyg för att kunna rita linjer, ellipser och rektanglar.

Checkboxarna *Only Selected Messages* och *Current SA*, knappen *Clear Current* samt checkboxen *Show TDOA hyperbels* bestämmer alla hur mycket information som ska visas i

lägesvyn. När *Only Selected Messages* är ikryssad visas bara den information som rapporterats in från de meddelanden som för tillfället är markerade i *Message List* fönstret. När *Current SA* är ikryssad visas aktuell lägesbild oavsett hur *Time slidern* är inställd. Aktuell lägesbild är det senast inrapporterade läget för varje plattform samt senast inrapporterde lägesbilder från varje annan enhet. Blir det för mycket information i vyn så kan aktuell lägesbild rensas med hjälp av knappen *Clear Current* för att sedan byggas upp på nytt. När *Show TDOA hyperbels* är ikryssad visas alla hyperbler som fås ur TDOA data. Är den inte ikryssad visas bara den resulterade positionen.

Även i vyns nederkant finns en toolbar. Denna styr i första hand vilken information som ska visas i vyn baserat på vilken typ av information det är, se Figur 40.

Figur 40 Nedre toolbar för *Situation Map*.

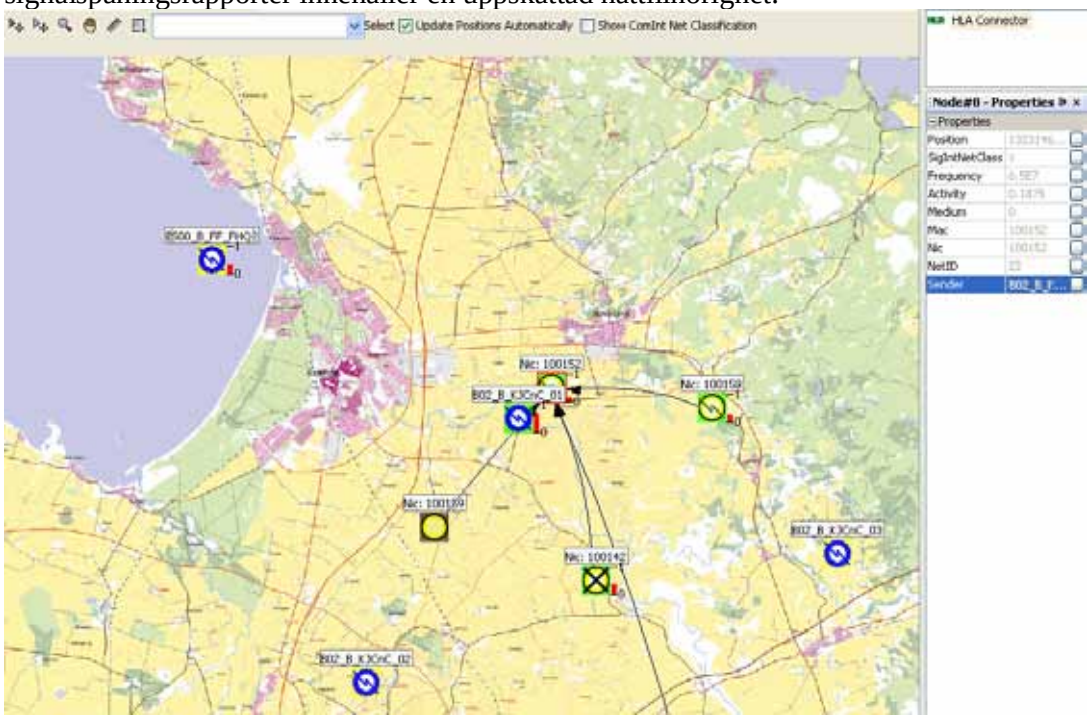
När checkboxen *Filled Icons* är ikryssad visas alla ikoner i sin fyllda variant och annars i den ofyllda varianten. De fem typcheckboxarna ger användaren en möjlighet att sortera bort information ut lägesbilden och göra den mer lättöverskådlig. Det är möjligt att välja synlighet hos egna plattformar, objekt rapporterade av en IRST, radar, och kommunikationssignalspaning samt objekt som är skapade av en fusionsfunktion. På samma sätt kan de tre klasscheckboxarna användas för att bestämma synlighet hos objekt som är klassificerade som *friendly*, *hostile* eller *unknown*.

När användaren högerklickar i *Situation Map* öppnas en högerklicksmeny. Den innehåller genvägar för att snabbt kunna sända de vanligaste typerna av FFM. Mottagare av FFM sätts till de egna plattformar som för tillfället är markerade. Finns det inga markerade objekt, sätts alla plattformar som mottagare. Precis som när man inte använt sig av högerklicksmenyn kan dock både mottagare, FFMtyp och innehåll i meddelandet ändras i vyn *FFM Command Sender* innan det skickas.

2.9.2 Network Map

Network Map är en kartvy som visar uppspanat nätverksläge. Informationen bygger på rapporter från signalspaning och CNO. Noder i egna nätverket antas kända och placeras vid uppstart i sin respektive startposition. Uppdatering av egna nätverksnoders position sker via egna plattformars positionsuppdateringar. Egenskaper hos en markerad nod visas i *Properties*-fönstret. Ett exempel på hur nätverksläget kan se ut under en simulering visas i Figur 41. Ikonerna som markerar nätverksnoder är blå om de tillhör det egna laget och gula om de är uppspanade. Mitten på ikonerna är vit om de har en känd, inrapporterad position, annars skuggad i ramens färg. Finns en blixtformad antennsymbol i ikonen representerar den en radionod, annars är det en kabelnod. Ett kryss över ikonen markerar att aktiv scanning rapporterat att noden är känslig för DOS-attack och ett utropstecken markerar att en nod attackerar eller blir attackerad av en DOS-attack.

Till höger om själva ikonen finns en röd stapel som visar nodens aktivitet som ett tal mellan 0 och 1. Aktivitetsstapeln visas dock enbart i de fall som rapporter om aktivitetsnivån finns tillgängliga. Bakgrundsfärgen för en ikon är relevant på så vis att noder som till synes tillhör samma nät får samma bakgrundsfärg. Nättillhörighet fås genom passiv scanning, men även signalspaningsrapporter innehåller en uppskattad nättillhörighet.



Figur 41 Uppspanat nätverksläge visat i *Network Map* med information byggd på rapporter från signalspaning och CNO.

Genom passiv scanning fås också information om vilka noder som kan kommunicerar med varandra. Det är dessa länkar som syns i kartvyn. Däremot har man ingen kunskap om vägen som kommunikationen tagit mellan dessa noder, vilket gör att nätverksbilden som helhet inte behöver stämma överens med verkligheten. Det man kan säga är *att* de noder som har en länk sinsemellan kan kommunicera med varandra, inte *hur* de gör det.

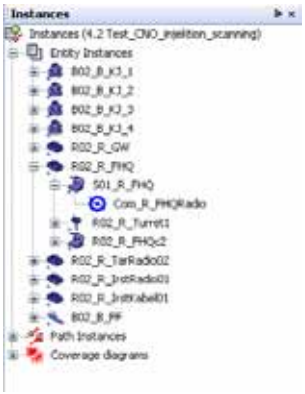
I vyns överkant finns en toolbar. Verktygen fungerar på samma sätt som är beskrivet för *Siuation Map* i avsnitt 2.9.1, förutom att skapaverktyget inte har någon funktion i *Network Map*. Själva kartans läge, dvs zoomnivå och panoreringsläge, är ihopkopplat i de båda vyerna, så att en ändring i *Situation Map* påverkar även kartan i *Network Map* och tvärtom. Checkboxen *Update Positions Automatically* är i defaultläget ikryssad, vilket innebär att alla ikoner som representerar nätverksnoder flyttas till sin nya position då nya rapporter kommit in. I urkryssat läge så kan användaren själv flytta runt alla ikoner till valfri plats i kartvyn. De noder som inte har någon inrapporterad position kan alltid flyttas till valfritt läge.

I defaultläge visas nättillhörighet hos noderna i första hand baserat på rapporter från passiv scanning, eftersom den är mer tillförlitlig än klassificering från signalspanig. Kryssas checkboxen *Show ComInt Net Classification* i visas istället signalspaningens klassificering i första hand. Detta kan vara användbart som jämförelse då noder som tillhör samma nät kan få olika bakgrundsfärg beroende på att informationen kommer från olika källor. Checkboxen påverkar endast själva kartvyn. All information från olika källor finns alltid tillgänglig i *Properties*vyn.

När användaren högerklickar i *Network Map* öppnas en högerklicksmeny. Den innehåller genvägar för att snabbt kunna sända de vanligaste typerna av FFM som gäller nätverksoperationer. Mottagare av FFM sätts till de egna plattformar som för tillfället är markerade och injektionsnod sätts till för tillfället markerad uppspanad nod. Max ett markerat objekt per sida tillåts, i annat fall får användaren själv sätta värdena i vyn *FFM Command Sender*. Precis som när man inte använt sig av högerklicksmenyn kan också både mottagare, FFMtyp och innehåll i meddelandet ändras i den vyn innan det skickas.

2.9.3 Konfigurera LKSStab i NetScene

När man vill ha med en stab i ett simulerat scenario måste den i Netscene ges ett eget federatnamn, eftersom LKSStab är en fristående applikation. I entitetsträdet placeras staben under den plattform den ska befinna sig på. Under staben placeras det radiosystem, ägt av ComNet, som staben ska använda sig av. Ett exempel på hur det kan se ut visas i Figur 42, där en instans av LKSStab har fått federatnamnet S01.



Figur 42 En LKSStabs placering i entitetsträdet under scenarioeditering. S01_R_FHQ är en instans av LKSStab.

2.10 Loggningsverktyg, EWLogger

EWLogger är ett verktyg för att lagra, analysera och återuppspela en HLA-simulering. Det går att köra EWLogger som konsolapplikation, windows-applikation eller integrerat i en HLA-federat. Användargränssnittet för EWLogger visas i Figur 43.

Figur 43 Användargränssnitt för EWLogger.

EWLogger kör hela tiden i bakgrunden och loggar kontinuerligt de data som utbyts över HLA. Den fungerar snarlikt en ”bandspelare” med funktioner som play, rec och pause. Vid återuppspelning, kan man med en timeslider snabbt hoppa till en önskad tidpunkt i simuleringen och fortsätta uppspelningen ifrån den tidpunkten. Ett flikfönster med en instansvy visar aktiva objekts status.

2.10.1 Filformat, objektlagring

En logg består av tre filer, projektfil (ewl) med översiktlig information om körningen, deltafil med HLA-förändringar i tidsordning samt en keyframefil som gör att man enkelt kan läsa upp alla objekts tillstånd i ett givet ögonblick. Allt lagras med hjälp av HLA-encoders för att göra det möjligt att enkelt läsa filerna under alla operativsystem som MOSART stödjer (Windows XP, Linux, Sun Solaris).

2.10.2 Användning under och efter ett spel

Att logga data med loggern under ett spel gör man genom att starta logger samt därefter trycka på knappen nytt och ange vilken fil man vill spara loggen i. Därefter trycker man på Rec knappen och ansluter loggern till federationen genom att trycka på HLA Connect. Därefter loggas all HLA trafik och när simuleringen avslutas sparas loggfilen automatiskt. Vill man spara loggen till XML filer gör man detta genom att först öppna loggfilen därefter gå in under file och Save as XML varvid loggfilen sparas även som XML filer. Då XML filen kan bli för stor för att kunna öppnas i t ex Excel så kommer denna att vid behov delas upp i flera filer automatiskt.

Återuppspela en simulering gör man genom att starta EWLogger samt öppna en logg fil och därefter ansluta loggern till federationen som man vill återuppspela loggade data i, vilket t. ex. kan vara ComNet, EWSim, och NetScene, genom att trycka på play och därefter HLA Connect.

När loggen startas och avslutas m.h.a. det program som kan starta hela federationen, från en annan dator “Remote process manager”, så skapas och sparas en loggfil automatiskt.

3 Starta och köra ett scenario

Starta grundkomponenterna för HLA federationen.
När EWSim har installerats kan man starta alla ingående komponeneter från Windows start-meny. Grundkomponenterna för att federationen och editering av scenarier startas:

Start>pRTI 1516>Start pRTI1516	Startar RTI:et.
Start>EWSim>FederationManager	Startar federationsstyrningen.
Start>EWSim>NetScene	Startar scenarioeditorn.

Öppna ett scenario
Första gången efter en installation måste foldern med scenarierna öppnas genom att:
I NetScene gå till *File > Open Project*
Gå till den folder där scenarierna ligger klicka därefter tryck på *Open Project Folder*-knappen
Efter detta ska en mapp finnas under *Projects*-fliken i NetScene.

Ladda ett scenario för LKS genom att dubbelklicka på ett scenariofilnamn som ligger under LKS-mappen.
Öppna en karta genom att dubbelklicka på filen *engelholm_20_indexerad.png* i mappen *kartor*.

Antal federater i ett scenario kan man se i NetScene i vyn *Instances*.

Starta och anslut federater till RTI.
Kontrollera att RTI:et är startat

Starta federationen genom att trycka på knappen *Create Federation* i FederationManager. När federationen är startad kommer FederationManager in i listan *Federates using synchpoints*.

I NetScene, se till att *pRTI1516* är vald som *HLA binding* och anslut NetScene genom att högerklicka på *HLA Connector* och välja *connect*. NetScene ska dyka upp i listan *Federates using synchpoints* i FederationManager.

EWSim startas genom att köra *Start>EWSim>EWSim*. I EWSim-fönstret under *Federation>Federation Settings* anges federatnamnet på respektive federat i rutan *Federat name* (samma tre bokstäver som de tre första i namnen som angetts i NetScene på de objekt som federaten ska skapa). I rutan RTI host anges ip-numret på den dator som kör RTI:et. Tryck på OK. Federaten kommer att dyka upp i listan över *Federates using synchpoints* i FederationManager. Detta förfarande görs för alla EWSim-federater som ska anslutas till RTI:et.

ComNet startas och ansluts genom att köra *Start>EWSim>ComNet.*
Kontrollera att ComNet dyker upp i listan *Federates using synchpoints* i FederationManager.

(LKS Stab) startas genom att gå till *Start>EWSim>LKS Stab*.

Avsluta Scenariot
Avsluta alla EWSim-applikationer genom att stänga dess fönster. NetScene avslutas genom att högerklicka på *HLA Connector* och välja *Disconnect*. FederationManager avslutas genom att trycka på knappen *Shutdown*.

Om det är några problem med att avsluta federationen kan man i *pRTI Explorer* manuellt avsluta federater. Detta görs genom att välja önskad federat i listan (*Federations - ”ditt federationsnamn” - ”federatnamnet”*) och trycka på knappen *Resign Federate.* När alla federater är avslutade kan man välja federationen (*Federations - ”ditt federationsnamn”*) och trycka på knappen *Destroy Federation Execution*.

Använda remote start och stop
Starta *Start>EWSim>RemProcMgrSrvr* på alla datorer som ska användas till fjärrstyrning. Alltså oavsett om man ska starta processer från datorn eller på datorn. Läs mer om funktionaliteten i ReadMe.txt som ligger i samma katalog som programmet.

Starta *Start>EWSim>RemProcMgrClntGUI* på den maskin som ska starta alla program, alltså den dator som bestämmer vilka datorer som ska starta vilken process. Alternativt kan man starta *Start>EWSim>RemProcMgrClnt*, vilket har samma funktion, men inget grafiskt gränssnitt.

För att starta applikationer läses en textfil in (i *RemProcMgrClntGUI*) som definierar vilka applikationer som ska startas av vilka datorer. Alla datorer som ska starta applikationer måste ha *RemProcMgrClnt* startat då filen läses in i *RemProcMgrClntGUI*. Ett exempel på textfil finns i filen *RemoteStartEWSim.txt*.
Varje rad i textfilen definierar en applikation som ska startas och innehåller i följande ordning:

1. Datorn som ska starta applikation (IP-nummer)
2. Process-ID
3. Kommandorad

Kommandoraden är beroende av vilken applikation som ska startas. Läs mer i *../EWSim/EWSimSrc/src/RemProcMgr/Remote start ReadMe.txt*.

För EWSim ser kommandoraden ut enligt:
”Datornamn på den dator som ska köra EWSim” ”Sökväg till EWSim.exe” --connect --federatename ”federatnamn” --rtihost ”RTI-adress”
Texten inom citationstecken ska ändras. *”RTI-adress”* är den datorn som kör RTI:et (IP-nummer).

Bilaga: Installation

Då LKS är baserat på EWSim som bygger på nätverkskommunikation via HLA krävs att ett RTI är installerat på datorn. Både det RTI som används samt vissa delar av programvaran för LKS bygger på Java. Då Java inte är fullständigt bakåtkompatibelt krävs även att rätt version av Java installeras. På grund av detta kan även uppdateringar via Java-update göra att vissa program inte fungerar korrekt. Ignorera därför uppdateringar via Java-update.

Alla programvaror som krävs för att köra LKS medföljer installationsfilerna i rätt versioner. För att programvarorna ska fungera ihop krävs att de installeras på följande sätt och ordning:

1 Installera Java
Kör *jdk-1_5_0_06-windows-i586-p.exe*.
Använd de föreslagna inställningarna och se till att installationskatalogerna är *C:\Program Files\Java\jdk1.5.0_06* samt *C:\Program Files\Java\jre1.5.0_06*.

2 Installera CVI
Kör *setup_CVI_RT.exe* med föreslagna inställningar.

3 Installera pRTI 1516
Kör *install_prti1516_v3.1.9.2.exe* med föreslagna inställningar.

4 Installera EWSim
Kör *EWSim_setup.exe* med föreslagna inställingar.

5 Installera NetScene
Kör *NetScene2.0_setup.exe* med föreslagna inställningar.

6 Installera Data
Kör *EWSimData_setup.exe* med föreslagna inställningar. Alternativt kan data kopieras manuellt och ska då läggas i katalogen *C:\Program Files\EWSim\Data.*

7 Inställningar pRTI
För att pRTI ska fungera korrekt krävs att inställningar görs i LRC settings. Starta därför pRTI 1516/LRC settings från startmenyn.
Ändra inställningar enligt följande:

Time Factory:
se.foi.hla1516.time.LogicalTimeDoubleFactory

Time Interval Factory class:
se.foi.hla1516.time.LogicalTimeIntervalDoubleFactory
Välj *Save as default*.

8 Starta om datorn
För att kunna börja använda programvarorna för LKS krävs en omstart av datorn.

Bilaga: C2 Conditions

IF satsen
If satsen inleds med ett av följande typer av event:
POSITION_REACHED
MESSAGE_RECEIVED,
DISCOVERED_NEW_TARGET

Efter event strängen kan ett eller flera nyckelord finnas. Efter nykelorden anges en eller flera datasträngar

Event nyckelord
RETRIGGERTIME ”antal sekunder tills eventen triggas igen” -1 innebär att eventen endast triggas en gång
POS ”X-koordinat” ”Y-koordinat”

Action blocket

Action blocket inom {} bestå av en eller flera action se nedan
SNABBGRUPPERING
GRUPPERINGSORDER
AVBRYT_GRUPPERING
WAIT
SET_SENSOR_ON
SET_SENSOR_OFF
SET_SENSOR_PASSIVE
SENSOR_AREA_ANGLE
SENSOR_INTERMITTENT
SEND_TARGET_POS
SEND_TARGET_BEARING
SEND_TARGET_CLASS
SEND_PLATFORM_POS
SEND_SITUATION
SEND_SITUATION_INTERV
SEND_TARGET_UPDATE
SEND_TDOA_DATA
SEND_SIGINT_INFO
RECEIVE_TARGET_POS
RECEIVE_TARGET_BEARING
RECEIVE_TARGET_CLASS
RECEIVE_PLATFORM_POS
RECEIVE_EMPTY_SITUATION
RECEIVE_NEW_INFO
RECEIVE_TDOA_DATA
RECEIVE_SIGINT_INFO
RADIO_SILENCE
RADIO_SILENCE_OFF
SEND_MESSAGE_TO

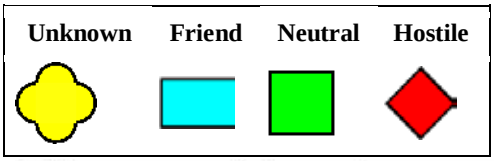
Bilaga: Meddelanden

Kod	*Namn*	*Funktion*
442	FOLLOWPLATFORM	Följ annan plattform
443	SNABBGRUPPERING	Order om gruppering
444	GRUPPERINGSORDER	Order om snabbgruppering
601	ORDER_SET_CND	
602	ORDER_SET_CNA	
901	ORDER_SET_SENSOR_ON	Order att slå på sensor
902	ORDER_SET_SENSOR_OFF	Order att slå av sensor
903	ORDER_SET_SENSOR_PASSIVE	Order att sensor ska vara passiv
904	ORDER_SENSOR_AREA_ANGLE	Order som definierar sökområde för sensor
905	ORDER_SENSOR_INTERMITTENT	Order att sensor ska sända intermittent
906	ORDER_SEND_TARGET_POS	Order att skicka registrerade målpositioner
907	ORDER_SEND_TARGET_BEARING	Order att skicka registrerade målbäringar
908	ORDER_SEND_TARGET_CLASS	Order att skicka klassificering av registrerade mål
909	ORDER_SEND_PLATFORM_POS	Order att skicka sin egen position
910	ORDER_SEND_SITUATION	Order att skicka lägesbild (efter datafusion)
911	ORDER_SEND_SITUATION_INTERV	Order att skicka lägesbild (efter datafusion) med visst intervall
912	ORDER_SEND_TARGET_UPDATE	Order att skicka uppdaterad information om viss mål
913	ORDER_SEND_TDOA_DATA	Order att TDOA-enheter ska skicka TDOA-data
914	ORDER_SEND_SISBEARING_DATA	Order att SiS-enheter ska skicka SiS-data
915	ORDER_SEND_SIGINT_INFO	Order att TVC ska skicka signalspaningsrapport
916	ORDER_SEND_SIGINT_INFO_TO_ME	Order att TVC ska skicka signalspaningsrapport till annan ledning
917	ORDER_SEND_OBSERVEDSITUATION	Order att skicka observerad lägesbild (från egna sensorer innan datafusion)
918	ORDER_SEND_OBSERVEDSITUATION_INTERV	Order att skicka observerad lägesbild (från egna sensorer innan datafusion) med visst intervall
919	ORDER_SEND_RADIO_JAM_SIGNAL	Order att radiostörare ska starta störsändning
920	ORDER_RESET_TDOA_STATE	Order att nollställa TDOA-systemet
921	ORDER_STOP_RADIO_JAM_SIGNAL	Order att stoppa radiostörsändning
922	ORDER_SET_TVCNEAR_REPORT_STATE	Order för att sätta rapporteringsstatus för närstörenhet
923	ORDER_SET_TVCFAR_REPORT_STATE	Order för att sätta rapporteringsstatus för fjärrstörenhet
924	ORDER_RADIO_SILENCE	Order att iakttaga radiotystnad
925	ORDER_RADIO_SILENCE_OFF	Order att avbryta radiotystnad
926	SENDING_TARGET_POS	Skickar målposition
927	SENDING_TARGET_BEARING	Skickar bäringar till mål

928	SENDING_TARGET_CLASS	Skickar målklassificering
929	SENDING_PLATFORM_POS	Skickar plattformsposition
930	SENDING_EMPTY_SITUATION	Skickar en tom lägesbild (inga mål att rapportera)
931	SENDING_NEW_INFO	En order som skickas automatiskt av systemet för att mottagaren ska veta vad för information som den ska ta emot
932	SENDING_TDOA_DATA	Skickar TDOA-data
933	SENDING_SISBEARINGDATA	Skickar SiS-data
934	SENDING_SIGINT_INFO	Skickar signalspaningsrapport
935	AVBRYT_GRUPPERING	Avbryter grupperings- eller snabbgrupperingsorder
936	WAIT	Används för att vänta innan nästa order kan hanteras
937	LINK_FORCESTATUS	
938	LINK_FORCETRAFFIC	
939	ORDER_FIRE	Eldgivningsorder
940	ORDER_FIRE_AT_WILL	Order för fri eldgivning
941	ORDER_SEND_SIGINT_INFO_TO_STAFF	Order att skicka signalspaningsrapporter till stab
942	ORDER_SET_RADAR_NOISEJAMMER_ON	Order att aktivera radarbrusstörare
943	ORDER_SET_RADAR_NOISEJAMMER_OFF	Order att inaktivera radarbrusstörare
944	ORDER_SET_RADAR_DRFMJAMMER_ON	Order att aktivera repeterstörsändare för radar
945	ORDER_SET_RADAR_DRFMJAMMER_OFF	Order att inaktivera repeterstörsändare för radar
946	ORDER_MULTIMESSAGE	Innehåller flera meddelanden

Bilaga: Symboler

De symboler som används I LKS är av APP- 6a typ enligt NATO standardöverenskommelsen STANAG 2019 dessa kan antingen visas som fyllda eller ofylda där färgen anger styrketillhörighet. Utöver dessa symboler finns ettt antal som inte finns representerade I APP-6a standarden för CNO, Störsändare och nätverk.



Entity name	font	char	enemy, friendly, unkown, neutral
UNDEFINED	Land	0	
AIRCRAFT	Air	0	
VESSEL	Sea	0	
VEHICLE	Land	0	
SENSOR	NA-Equp	alt+0204	
STRF	Land	M	
Strv	Land	A	
Spaningsradar	Land	alt+0226	
ZSU23	Land1	alt+0150	
Hkp14	Air	%	
Hkp9	Air	a	
J35	Air	P	
SK37E	Air	J	
JAS39	Air	P	
TP84	Air	C	
UAV	Air	V	
Fregat	Sea	F	
RADIO_SYSTEM	Land	m	

Referenser

[DES] Lars Tydén, Leif Festin, Sebastian Olsson, Christer Wigren, ”Systembeskrivning LKS V2 Federationsdesign”, FOI-rapport FOI-R--2274--SE, (2007).

[MOSART] A. Törne, T. Horney, M. Brännström, P. Hörling, L. Tydén, ”Projektutnyttjande MOSART”, FOI-Memo 1128 (2004).

[ENTITY] **http://www.sisostds.org/dis%2Ddd/entity/**

[ESOPTR] H. Andersson, C. Hedberg, M. Petersson, L. Tydén, C. Wigren, ”Integrerad EO duellmodell”, FOI-rapport FOI-R--1724--SE, (2005).

[GENPOD] P. Klum, Å. Andersson, ”GENPOD”, FOI-rapport FOI-R--1414--SE (2004).

[DETVAG] B. Asp, G. Eriksson, and P. Holm, “Detvag-90® -- Final Report”, FOA-rapport FOA-R--97-00566-504--SE (1997).

[WRAP] AerotechTelub, **http://www.wrap.se**

[F-T] N.Appleby, “Freke-Tavast 1.3”, FOI-Memo 04-827 (2004).

[TARAX] **www.trltech.co.uk**

[STAB] H. Andersson, L. Tydén, ”Systembeskrivning LKS Stabsverktyg”, FOI-rapport FOI-R--2276--SE, (2007).

[NB] www.netbeans.org