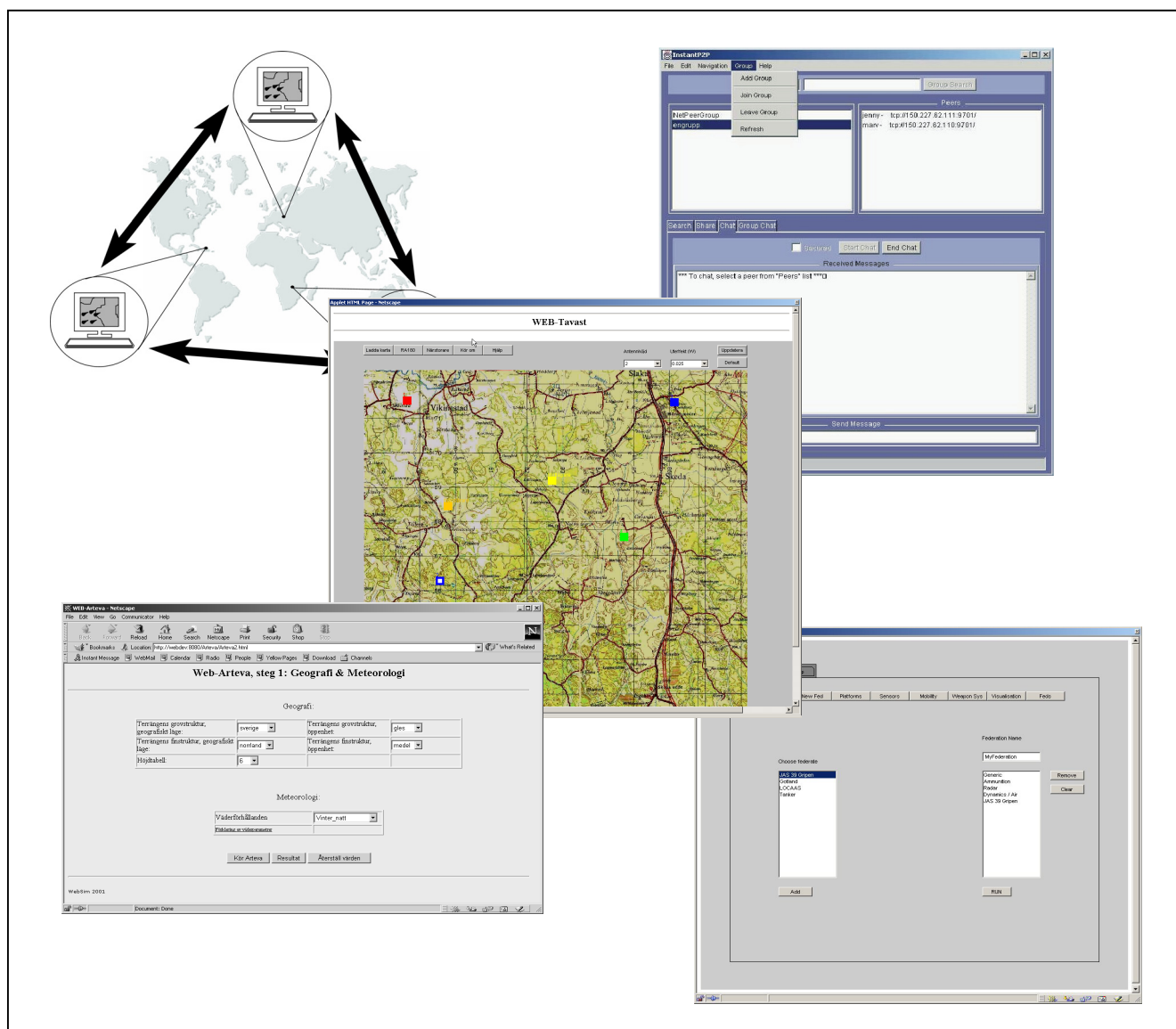


Webbaserade HLA-Federationer och Simuleringar

– Metoder och Möjligheter

Farshad Moradi, Martin Eklöf, Jenny Ulriksson



Webbaserade HLA-Federationer och Simuleringar

- Metoder och Möjligheter

Farshad Moradi, Martin Eklöf, Jenny Ulriksson

Utgivare Totalförsvarets Forskningsinstitut - FOI Systemteknik SE-172 90 Stockholm	Rapportnummer, ISRN FOI-R--0739--SE	Klassificering Användarrapport
	Forskningsområde 2. Operationsanalys, modellering och simulering	
	Månad, år December 2002	Projektnummer E6030
	Verksamhetsgren 5. Uppdragsfinansierad verksamhet	
	Delområde 21. Modellering och simulering	
Författare/redaktör Farshad Moradi Martin Eklöf Jenny Ulriksson	Projektledare Farshad Moradi	
	Godkänd av Monica Dahlén	
	Uppdragsgivare/kundbeteckning FM	
	Tekniskt och/eller vetenskapligt ansvarig Rassul Ayani	
Rapportens titel Webbaserade HLA-Federationer och Simuleringar – Metoder och Möjligheter		
Sammanfattning (högst 200 ord) Den explosionsartade utvecklingen inom webbteknologin och Internet å ena sidan och utvecklingen inom närliggande teknologier å andra sidan har skapat nya möjligheter för många discipliner. Modellering och simulering (M&S) är inget undantag. Webbteknologin kan bl. a. erbjuda nya möjligheter för att utveckla, distribuera och exekvera simuleringsmodeller. Den här rapporten redovisar resultatet av ett arbete inom området webbaserad modellering och simulering. Projektet, ”Webbaserade HLA-federationer och simuleringar”, bedrevs under åren 2001 och 2002 på uppdrag av försvarsmakten med målet att undersöka och studera området och identifiera möjligheter som detta kan erbjuda FM. Projektet var under första året framförallt inriktat på kopplingen mellan webbteknologin och HLA-federationer samt webbteknologin och befintliga simuleringar. Arbetet under 2002 var fokuserat på komponentbaserad modellutveckling, effektivt utnyttjande av nätverksresurser via Peer-to-Peer- (P2P-) teknologi och kollaborativ modellutveckling och -exekvering. Våra slutsatser från arbetet bekräftar våra preliminära antaganden om fördelarna med kopplingen mellan webb och HLA-federationer och simuleringar. Vi tror att området har stor potential och att webbaserade simuleringar skulle kunna ha en central roll i framtida nätverksbaserade försvarssystem.		
Nyckelord Webbaserad Simulering, Distribuerad Simulering (DS), High Level Arkitecture (HLA), Peer-to-Peer (P2P), JXTA		
Övriga bibliografiska uppgifter	Språk Svenska	
ISSN 1650-1942	Antal sidor: 67 s.	
Distribution enligt missiv	Pris: Enligt prislista	

Issuing organization FOI – Swedish Defence Research Agency Systems Technology SE-172 90 Stockholm	Report number, ISRN FOI-R—0739--SE	Report type User report
	Programme Areas 2. Operational Research, Modelling and Simulation	
	Month year December 2002	Project no. E6030
	General Research Areas 5. Commissioned Research	
	Subcategories 21 Modelling and Simulation	
Author/s (editor/s) Farshad Moradi Martin Eklöf Jenny Ulriksson	Project manager Farshad Moradi	
	Approved by Monica Dahlén	
	Sponsoring agency Swedish Armed Forces	
	Scientifically and technically responsible Rassul Ayani	
Report title (In translation) Web-based HLA federations and Simulations – Methods and Possibilities		
Abstract (not more than 200 words) Modelling and Simulation (M&S) is a vital part of the Network-Centric Warfare (NCW) concept, adapted by the Swedish Armed Forces (SwAF). However, simulation model development, implementation, testing, and execution are time consuming and expensive processes. Hence, it is of great interest to combine the web and Internet technologies, and M&S in order to e.g. utilize the simulation models and computing resources more efficiently. This report presents the results from research conducted within the area of Web-based M&S. The project “Web-based HLA Federations and Simulations” was carried out during 2001 and 2002 on behalf of the SwAF. The main objective of the project was to investigate and study the benefits and advantages of the impact of networking technologies and distributed computing techniques on M&S for the SwAF. During the first year the potential of the combination of Web-technology and HLA federations, and Web-technology and legacy simulations was investigated. The focus of the project during 2002 was on component-based model development, distributed computing techniques, particularly Peer-to-peer computing, and collaborative model development and execution. The results from the project confirm our preliminary assumptions regarding the advantages of combining networking technologies, such as Internet and the World Wide Web, and M&S. We believe that the area has a great potential and will play a central role in the future NCW systems.		
Keywords Web-based Simulation, Distributed Simulation (DS), High Level Arkitecture (HLA), Peer-to-Peer (P2P), JXTA		
Further bibliographic information	Language Swedish	
ISSN 1650-1942	Pages 67 p.	
	Price acc. to pricelist	

Innehållsförteckning

Sammanfattning	7
Akronymer.....	9
1 Projektet.....	11
1.1 Bakgrund.....	11
1.2 Mål.....	11
1.3 Nyttan för FM.....	12
1.4 Genomförande.....	13
1.5 Rapportens omfattning och struktur.....	15
2 Bakgrund	17
2.1 Distribuerad simuleringsmiljö.....	18
2.1.1 HLA.....	18
2.1.1.1 HLA gränssnittsspecifikation.....	19
2.1.1.2 HLA OMT.....	19
2.1.1.3 HLA-regler.....	19
2.1.1.4 Hierarkiska federationer.....	20
2.1.2 JXTA & P2P.....	20
2.1.2.1 Bakgrund.....	20
2.1.2.2 JXTA.....	21
2.1.2.2.1 JXTA-definitioner.....	22
2.1.2.2.2 JXTA-protokollen.....	23
2.1.2.2.3 JXTA och möjligheter inom webbaserade simuleringar.....	23
2.1.3 Kollaborativ modellutveckling och –exekvering.....	25
2.1.3.1 Kollaborativ modellutveckling.....	25
2.1.3.2 Kollaborativ modellexekvering.....	26
2.1.3.3 Implementering.....	26
2.2 Webbaserade HLA-federationer.....	28
2.2.1 Introduktion.....	28
2.2.2 Nyttan med webbaserade HLA-federationer.....	29
2.2.3 Komponentbaserade simuleringar.....	30
2.3 Webbanpassade Simuleringar.....	31
2.3.1 Nyttan med webbanpassade simuleringar.....	31
2.3.2 Lämpliga webbt tekniker.....	32
2.4 Relaterade projekt och miljöer.....	33
2.4.1 Simulation Based Acquisition (SBA) Demonstrator.....	33
2.4.2 ALICE (Advanced, Scalable and Integrated Computing Engine).....	34
2.4.3 SIMULTAAN (holländska för simultan).....	34
2.4.4 NanoComp.....	35
2.4.5 SimKit.....	35
2.4.6 Silk®.....	35
2.4.7 Nätverksbaserad Modellering och Simulering (NBMS).....	36
3 Utveckling av en webbaserad simuleringsmiljö.....	37
3.1 Webbanpassad utveckling av webbaserade HLA-federationer.....	37
3.1.1 Metodik.....	37
3.1.2 Utveckling av prototyp.....	37
3.1.2.1 Testfederater.....	38
3.1.2.2 Lämpliga webbt tekniker.....	38
3.1.2.3 Webbgränssnitt.....	40
3.1.2.4 Implementering.....	41

3.1.3	Hierarkisk federationsutveckling	42
3.1.3.1	Metoder och tekniker	42
3.1.3.2	Implementering	43
3.2	Webbanpassning av simuleringsmodeller	43
3.2.1	Val av simuleringsmodeller	43
3.2.1.1	Tavast2	43
3.2.1.2	Arteva	44
3.2.2	Implementering av Webb-Tavast	44
3.2.3	Implementering av Webb-Arteva	45
3.3	Distribuerad exekveringsmiljö	47
3.3.1	Design av miljö	47
3.3.2	Implementering av exekveringsmiljön	52
3.3.2.1	JXTA	52
3.3.2.2	HLA	53
3.3.2.2.1	HLA-Manager	53
3.3.2.2.2	Federationsinitiering	53
3.3.2.2.3	Tidshantering	54
3.3.2.2.4	Avbrottshantering	55
3.3.2.2.5	Krav på deltagande federater	56
3.3.2.2.6	GenericHLAFederate	56
3.3.3	Resultat	57
4	Diskussion	59
4.1	Webbanpassade HLA federationer	59
4.2	Webbanpassade simuleringsmodeller	59
4.3	Distribuerad exekveringsmiljö	60
4.4	Framtidsvision	61
4.4.1	Konstruktionsmiljö	61
4.4.2	Exekveringsmiljö	62
5	Slutsatser	63
6	Referenser	65

Sammanfattning

Inom försvaret har utvecklingen gått mot webbaserade och nätverksbaserade system. Den senaste inriktningen kallas för "Network Centric Warfare", och visar på betydelsen av nätverk och nätverksbaserad kommunikation inom framtida försvarssystem. Den explosionsartade utvecklingen inom webbrelaterade teknologier och Internets snabba spridning har skapat nya möjligheter för många discipliner. Modellering och simulering (M&S) är inget undantag från detta. Webbteknologin erbjuder en infrastruktur som har potential att kunna revolutionera modellering och simulering. En annan känd infrastruktur med liknande effekter är järnvägen. Introduktionen av järnvägen förändrade, förutom vårt sätt att fysiskt förflytta oss, våra samhällsstrukturer. Det går inte att på förhand förutse alla möjligheter som en ny infrastruktur kan erbjuda, men när den väl är på plats kommer människan att hitta på de mest innovativa sätt att använda den.

Webbteknologin har potentialen att markant ändra vårt sätt att utveckla (kollaborativt och komponentbaserat), dokumentera (dynamiskt via multimedia), analysera (genom öppen och utbredd utvärdering) och exekvera modeller (genom distribution). Tänkbara användningsområden inkluderar:

- Modeller kan skapas och presenteras på webbsidor och därmed bli mer tillgängliga.
- Den potentiella användargruppen av datormodeller blir betydligt större genom utbredd distribution av dessa.
- Modellutvecklare och -användare som är lokaliserade på olika platser, kan samarbeta via webben och tillsammans konstruera modeller, genom t. ex. komponentbaserad utvecklingsteknik.
- Olika användare (t. ex. beslutsfattare), som befinner sig på geografiskt skilda platser, med tillgång till nätverket, kan koppla upp sig mot varandra och i realtid följa samma simuleringar som alla dessutom kan interagera med, t. ex. lägga till nya enheter, ändra förutsättningar, osv.
- Exekvering av simuleringsmodeller kan ske på ett nätverk av datorer. Genom att distribuera exekveringen kan lediga resurser inom nätverket (såsom minne och bearbetningskapacitet) utnyttjas effektivt.

Den här rapporten redovisar resultatet av ett arbete inom området webbaserad modellering och simulering. Projektet, "Webbaserade HLA-federationer och simuleringar" bedrevs under åren 2001 och 2002 på uppdrag av försvarsmakten (FM), med målet att undersöka och studera området och identifiera möjligheter som detta kan erbjuda FM. Projektet var under första året framförallt inriktat på kopplingen mellan webbteknologin och HLA-federationer, samt webbteknologin och befintliga simuleringar. Under 2002 var arbetet fokuserat på komponentbaserad modellutveckling, effektivt utnyttjande av nätverksresurser via Peer-to-Peer¹- (P2P-) teknologi och kollaborativ modellutveckling och -exekvering.

Våra slutsatser från arbetet bekräftar våra preliminära antaganden om fördelarna med kopplingen mellan webb och HLA-federationer och simuleringar. Vi tror att området har stor potential och att webbaserade simuleringar skulle kunna ha en central roll i framtida nätverksbaserade försvarssystem. För att kunna dra nytta av området bör dock framförallt metodiken och tekniken utvecklas vidare. Ett krav för att arbeta med webbaserad M&S är att de plattformsoberoende miljöer och tekniker som används har hög säkerhet och är anpassade för webben. Javateknologin är den mest lämpliga för

¹ Decentraliserad kommunikationsteknologi

tillfället, eftersom den erbjuder ett standardiserat, konsistent, säkert och plattformsoberoende ramverk för utveckling av skalbara, webbaserade och distribuerade applikationer.

Komponentbaserad modellutveckling, som är den givna utvecklingstrenden efter objektorienterad modellutveckling, erbjuder ett kostnads- och tidseffektivt sätt att bygga modeller och simuleringar. Det är dock viktigt att välja ändamålsenlig arkitektur. Samtidigt krävs det mycket arbete för att effektivt kunna utnyttja tekniken. Här skulle vi dessutom vara mycket betjänta av att undersöka och följa utvecklingen av det senaste inom programvaruutvecklingsmetodik, -teknik och -verktyg. Många av de problem som vi möter idag inom komponentbaserad M&S, har hanterats med framgång inom programvaruutvecklingsdomänen.

Att webbanpassa befintliga simuleringar kan leda till att de blir både användarvänligare och mer lättillgängliga, men det kan i vissa fall innebära stora ändringar i koden för att lyckas med uppgiften. Om det dessutom inte används lämplig teknik eller om simuleringen innehåller tung grafik, kan det uppstå för stora fördröjningar och tidskrävande körningar.

Den största fördelen med exekvering av simuleringar inom en P2P-miljö är att arbetet fördelas jämnt mellan noder, vilket eliminerar möjliga flaskhalsar i systemet. Resultaten från detta arbete visar att P2P-system har många fördelar jämfört med centraliserade system, men att de är mer komplexa att implementera.

Vår framtidsvision är ett system med bl. a. följande egenskaper:

- Ett distribuerat nätverk med noder av olika slag, t. ex. sensorer, databaser, ledningscentraler, soldater utrustade med mobila informationsterminaler (handdatorer, mobiltelefoner, etc.), osv.
- Noderna kan anpassas för individuella användare så att nätverket kan ha olika funktioner.
- Noderna skall kunna erbjuda tjänster av olika slag t. ex. simuleringsmodeller, sensordata, geografiska informationssystem, osv.
- Den verkliga världen skall vara integrerad med en virtuell värld.
- Den virtuella världen skall kunna befolkas med datorgenererade styrkor på olika aggregationsnivåer för att simulera fiender och olika taktiska förhållanden.

En fortsättning på det arbete som har utförts inom projektet är att studera närmare de frågor och aspekter på webb och M&S som har identifierats. Områden av särskilt intresse är kollaborativ M&S, P2P (resursallokering och -utnyttjande, etc.), komponentbaserad M&S och alternativa arkitekturer till High Level Architecture (HLA), distribuerade modellbibliotek och generiska webbaserade visualiseringsmoduler. Samtidigt är det av stor vikt att undersöka påverkan av webb på andra M&S-områden, som t. ex. VV&A (Verifiering, Validering och Ackreditering), CMMS (Conceptual Models of the Mission Space) och CGF (Computer Generated Forces).

Det skulle dessutom vara mycket fördelaktigt att bygga ett försvarsgemensamt simuleringsnätverk baserad på ovanstående tekniker med en distribuerad databas, som alla inom försvarsmakten kan ha tillgång till och själva lägga in modeller i. Databasen håller reda på resurser och komponenter inom nätverket, såsom t.ex. HLA-federater. Det skulle sedan kunna gå att leta i databasen via webbläsare efter alla tillgängliga komponenter vilket skulle medföra återanvändning av modeller och stor besparing av arbetstid och -kostnad.

Akronymer

ALICE	“Advanced Scalable and Integrated Computing Engine”
API	“Application Programmer’s Interface”
ASP	“Active Server Pages”
AWT	“Advanced Window Toolkit”
CGF	“Computer Generated Forces”
CGI	“Common Gateway Interface”
CMMS	“Conceptual Models of the Mission Space”
CMS	“Content Management Systems”
COM	“Component Object Model”
CORBA	“Common Object Request Broker Architecture”
COTS	“Commercial-Off-The-Shelf”
CPU	“Central Processing Unit”
DIS	“Distributed Interactive Simulation”
DLL	“Dynamic Link Library”
DMSO	“Defence Modelling and Simulation Office”
FEDEP	“Federation Development and Execution Process”
FOM	“Federation Object Model”
HLA	“High Level Architecture”
HTML	“Hypertext Markup Language”
HTTP	“Hypertext Transfer Protocol”
IP2P	“Instant Peer To Peer”
JNI	”Java Native Interface”
JSP	”Java Server Pages”
JVM	“Java Virtual Machine”
LAN	“Local Area Network”
M&S	“Modelleing and Simulation”
NBMS	“Net-based Modelling and Simulation”
NOW	“Network Of Workstations”
OMT	“Object Model Template”
P2P	“Peer To Peer”
PBP	“Peer Binding Protocol”
PDP	“Peer Discovery Protocol”
PEP	“Peer Endpoint Protocol”
PIP	“Peer Information Protocol”
PMP	“Peer Membership Protocol”
PRP	“Peer Resolver Protocol”
pRTI	“Portable Runtime Infrastructure”

RCI	“Runtime Communication Infrastructure”
RMI	“Remote Method Invocation”
RSS	“Resource Sharing System”
RTI	“Runtime Infrastructure”
SBA	“Simulation-Based Acquisition”
SBM	“Simulation-Based Methodology”
SCM	“Simulation Component Model”
SSA	“Simultaan Simulator Architecture”
UPnP	“Universal Plug and Play”
VV&A	“Validation, Verification and Accreditation”
WWW	“World Wide Web”
XML	“eXtensible Markup Language”

1 Projektet

1.1 Bakgrund

Inom försvaret har utvecklingen gått mot webbaserade och nätverksbaserade system. Den senaste inriktningen kallas för "Network Centric Warfare", och visar på betydelsen av nätverk och nätverksbaserad kommunikation inom framtida försvarssystem. Den explosionsartade utvecklingen inom webbt teknologin och Internet å ena sidan och utvecklingen inom närliggande teknologier som Java, XML (eXtensible Markup Language), CORBA (Common Object Request Broker Architecture), COM (Component Object Model), etc. å andra sidan har skapat nya möjligheter för många discipliner. Modellering och simulering (M&S) är inget undantag i det här fallet. Webbt teknologin kan bl. a. erbjuda nya möjligheter för att utveckla, distribuera och exekvera simuleringsmodeller.

Den här rapporten redovisar resultatet av ett arbete inom området webbaserad modellering och simulering. Projektet, "Webbaserade HLA-federationer och simuleringar" bedrevs under åren 2001 och 2002 på uppdrag av försvarsmakten (FM) med målet att undersöka och studera området och identifiera möjligheter som detta kan erbjuda FM. Projektet var under första året framförallt inriktat på kopplingen mellan webbt teknologin och HLA-federationer, samt webbt teknologin och befintliga simuleringar. Under 2002 var arbetet fokuserat på komponentbaserad modellutveckling, effektivt utnyttjande av nätverksresurser via Peer-to-Peer- (P2P-) teknologi och kollaborativ modellutveckling och -exekvering. Grundtanken i projektet var att framförallt studera, undersöka och identifiera möjligheterna, men också utforska på vilket sätt utvecklingen skulle kunna påverka FMs modeller och FMs sätt att utnyttja teknik- och metodutvecklingen inom webbaserad M&S.

1.2 Mål

Som framgått av ovanstående beskrivning, var det primära syftet med projektet under 2001 att undersöka möjligheter och förutsättningar, och utveckla metoder för att bygga å ena sidan webbaserade HLA-federationer och å andra sidan webbaserade användargränssnitt för befintliga simuleringsprogram.

Webbaserade HLA-federationer

I en distribuerad simuleringsövning (t. ex. en HLA-federation) körs vanligtvis varje simulering (federat²) på en egen dator, vilket innebär att simuleringsprogrammen måste installeras på varje maskin och dessa program har vanligtvis krav på vad och hur maskinerna skall vara. Däremot i en webbaserad federation³ körs simuleringar på centrala och kraftfulla datorer och det enda som behöver finnas på klientdatorn är en standardwebbläsare. På så sätt kan man t. ex. garantera en stabilare simuleringsmiljö som dessutom är lämplig för tunga beräkningsmodeller.

Webbaserade användargränssnitt

För närvarande har FM ett stort antal gamla simuleringsmodeller som antingen inte används alls eller inte i den utsträckning som de borde. Detta beror delvis på att dessa program körs på gamla maskiner som är krångliga att underhålla eller använda och delvis på att de inte har moderna och användarvänliga gränssnitt, vilket gör dem svåra

² "Federat" är benämningen på alla applikationer som ingår i en HLA simulering.

³ En sammansatt simulering i HLA:s regi, där flera federater interagerar med varandra kallas för "federation".

att arbeta med. Genom att anpassa befintliga simuleringsprogram, som av ovannämnda skäl inte används i den utsträckning som de borde, till webbtekniken ökar man deras användbarhet och tillgänglighet.

Vi har förutom de konkreta mål som sattes upp under 2001 haft det långsiktiga målet att undersöka och identifiera möjligheter, samt för- och nackdelar med kombinationen av webbteknologin med M&S. Under 2002 har vi arbetat vidare med detta mål och fokuserat insatserna kring komponentbaserad modellutveckling inom ramen för distribuerade simuleringar, decentraliserade lösningar för utdelning och effektivt utnyttjande av datorresurser, samt kollaborativa metoder för utveckling och exekvering av modeller.

1.3 Nyttan för FM

I det här avsnittet beskrivs fördelarna med webbaserade HLA-federationer och simuleringsmodeller, komponentbaserad modellutveckling, distribuerad exekvering av modeller inom en P2P-miljö och kollaborativ utveckling och exekvering av modeller. Rapporten berör förutom dessa punkter även andra potentiella fördelar med området.

Genom att integrera HLA-federationer och simuleringar med webbtekniken kan man minimera både kostnaden och arbetet och samtidigt öka flexibiliteten för organisation och utövning av distribuerade simuleringar. I en webbaserad federation körs simuleringar på centrala servrar och det enda som krävs av klientdatorn är en standardwebbläsare. Detta innebär att man först och främst kan få mycket bättre prestanda, genom att använda centrala och kraftfulla datorer. Dessutom blir det mycket snabbare, enklare och ur underhållssynpunkt mer robust, att sätta ihop träningsmiljöer (eftersom det inte finns några större krav på hur klientdatorerna skall vara). En sådan träningsmiljö kan för övrigt vara ett nätverk av heterogena datorer, d.v.s. datorerna av skilda typer.

Många befintliga simuleringsprogram körs på gamla maskiner och/eller har icke-moderna eller -användarvänliga gränssnitt, vilket försvårar utnyttjandet. Ett webbgränssnitt skulle kunna ge nytt liv åt dessa gamla, men dock kraftfulla simuleringsmodeller. Genom att anpassa dessa simuleringsprogram, som av ovannämnda skäl inte används i den utsträckning som de borde, till webbtekniken ökar man deras användbarhet och tillgänglighet. Samtidigt kan användarna få tillgång till ett stort antal simuleringsmodeller som de kan köra via vilka datorer som helst, om de bara är kopplade till nätverket. Med större tillgänglighet menas här att användare har lättare att annonsera ut sina modeller och att hitta och även prova andras modeller. Ur ekonomisk synvinkel är det mycket värdefullt att återigen och mer effektivt kunna utnyttja redan befintliga modeller.

Komponentbaserad utvecklingsmetodik har framgångsrikt använts inom olika industrier och discipliner under en längre period. Objektorienterad programvaruutveckling har t. ex. varit den metodik som med fördel utnyttjats inom mjukvaruindustrin. Den största fördelen med komponentbaserad utvecklingsmetodik är återanvändbarhet, där man istället för att bygga komplexa sammansatta system utvecklar mindre specialiserade delsystem med bestämda gränssnitt som kan kopplas samman i olika konstellationer och utgöra större system. En annan fördel är att eftersom samma ackrediterade delsystem använts för att bygga mer komplexa system får man bättre kvalitet på systemen.

Genom att utnyttja kollaborativa utvecklings- och exekveringsmetoder kan modellutvecklare, -programmerare och -användare som arbetar med samma projekt och är lokaliserade på olika orter samarbeta via nätverket och tillsammans konstruera modeller, genom t. ex. komponentbaserad utvecklingsteknik. På så sätt ökar man

modellernas kvalité och användbarhet. Dessutom blir det möjligt för olika användare (t. ex. beslutsfattare), som befinner sig på geografiskt skilda platser med tillgång till nätverket att koppla upp sig mot varandra och i realtid följa samma simuleringar som alla dessutom kan interagera med, t ex lägga till nya enheter, ändra förutsättningar, osv.

En majoritet av dagens webbaserade applikationer bygger på det traditionella klient-server-tänkandet, vilket trots kontinuerlig utbyggnad av Internets kapacitet skapar obalans i resursutnyttjandet. Välutnyttjade delar av nätverket blir allt mer hårt trafikerade, medan andra förblir outnyttjade. Genom att utnyttja P2P-teknologi och distribuera exekveringen av simuleringsmodeller kan man på ett effektivt sätt utnyttja lediga resurser (såsom minne och bearbetningskapacitet) på nätet. Exekvering kan initieras från klienter, medan modellerna själva körs på maskiner (med lediga resurser) som kan vara distribuerade. Ett bra exempel är alla datorer som inte för tillfället utnyttjas av sina användare.

1.4 Genomförande

Arbetet under 2001 var fokuserat på att finna svaren på följande frågor kring webbsimuleringar och återanvändning av simuleringsprogram:

- I vilket mån är det möjligt och praktiskt att skapa webbaserade HLA-federationer?
- Hur kan man anpassa befintliga simuleringsmodeller och program till webbtekniken på bästa sätt?
- Vilka tekniker är bäst lämpade?
- Vilka typer av simuleringsmodeller och program kan man anpassa till den nya tekniken?

För att arbeta med ovanstående frågor satsade projektgruppen på att utveckla en webbaserad miljö för utveckling av distribuerade simuleringar, som skulle hjälpa oss att bättre förstå området, samt agera som en plattform för fortsatt arbete.

Arbete under året bedrevs i följande spår:

Webbaserade HLA-federationer

Arbetet inom detta spår gick ut på att utveckla webbaserade HLA-federater och federationer, samt identifiera lämpliga metoder och tekniker för detta (och ev. utveckla nya). Gruppen skulle dessutom bygga grunden och identifiera lämpliga metoder och tekniker för att bygga en miljö för utveckling och exekvering av dessa typer av federationer. Arbetet utfördes i följande steg:

- En del av arbetet handlade om att identifiera ett antal lämpliga federater och federationer. Dessa federater skulle vara enkla, men dock tillräckligt komplexa för att implementeringen skulle leda till intressanta resultat. Federationen skulle dessutom innehålla en styrande federat (*"federation manager"*) med grafisk presentation.
- Ett krav på miljön var att användare ska kunna komponera federaterna genom att sätta ihop delmodeller, vilket innebar att gruppen skulle identifiera ett antal lämpliga delmodeller/komponenter, som utgjorde

byggstenarna till federaterna (FOM⁴, "Federation Object Model", och Fed-filerna, "Federation execution details", skulle utvecklas automatiskt).

- Vi arbetade samtidigt med att studera och identifiera lämpliga tekniker för utveckling av ovanstående federater. Federaterna skulle kunna köras via webben. Bland tekniker som identifierades kan vi nämna kombinationen med Java-servlet och -applet, se avsnitt 2.3.2.

Webbaserade simuleringsmodeller

Arbetet inom detta spår gick ut på att utveckla och identifiera metoder och tekniker för anpassning av befintliga simuleringsmodeller till webben. Vi satsade också på att bygga grunden och identifiera lämpliga metoder och tekniker för utveckling av en miljö för exekvering av dessa typer av simuleringar. Arbetet delades in enligt följande:

- Första steget var att identifiera ett antal intressanta befintliga simuleringsmodeller som kunde kopplas till webben. Någon av simuleringarna skulle ha ett grafiskt gränssnitt. För den här delen av projektet beslutade vi att webbanpassa TAVAST2 och ARTEVA. En mer detaljerad beskrivning av arbetet följer i avsnittet 3.2.
- Senare identifierade vi lämpliga tekniker för att bygga "grafiska" webbgränssnitt, samt studerade möjligheterna med att presentera befintligt gränssnitt i en webbläsare.
- En gemensamt punkt för båda spåren var att utforska hur en distribuerad exekveringsmiljö skulle kunna byggas.

Under 2002 arbetade vi vidare med utvecklingen av vår webbaserade miljö, dock med en fokusering kring följande frågor:

- Vilka metoder skall användas för komponentbaserad modellutveckling?
- Vilka är de effektivaste metoderna för att identifiera och utnyttja (outnyttjade) nätverksresurser på bästa sätt?
- Vilka metoder för effektivt samarbete (för t ex utveckling av modeller) via nätet skall användas?

Arbete under året bedrevs i följande spår:

Komponentbaserad modellutveckling

Detta spår var en fortsättning på det arbete som startades under 2001, där vi tittade på HLA som en standard för utveckling av komponentbaserade simuleringsmodeller. Här studerades möjligheterna, samt identifierades lämpliga tekniker för utveckling av "hierarkiska federationer", dvs. federationer vars ingående federater är federationer. Dessutom utförde vi, tillsammans med projektet "Nätverksbaserad Modellering och Simulering (NBMS)", en undersökning för identifiering av metoder för automatisk matchning av federater via deras SOM:ar⁵ och automatisk utveckling av FOM:ar. Det senare resulterade i en utvärdering av SOM:arnas lämplighet som enda informationskällan för identifiering av lämpliga delmodeller (federater) och utveckling av sammansatta simuleringsmodeller (federationer) [NätSim, 2002].

⁴ Federation Object Model, standardiserade beskrivningar av sammansatta simuleringsmodeller inom HLA

⁵ Simulation Object Model, standardiserade beskrivningar av simuleringsmodeller inom HLA

Distribuerad exekvering av modeller i en P2P-miljö

Under 2001 utförde gruppen en undersökning av olika plattformar för utveckling av en P2P-miljö. P2P är en alternativ nätverksteknologi som inte utesluter, men som naturligt inte bygger på centraliserad kontroll. I ett P2P-nätverk kommunicerar nätverksenheter direkt med varandra, utan stöd från centrala servrar och resurser. På det sättet kan beräkningsresurser och data utnyttjas i varje enskild del av nätverket. Resultatet från undersökningen var att använda JXTA, som är en P2P-plattform utvecklad av Sun Microsystems. Arbetet under 2002 gick ut på att utveckla en distribuerad feltolerant exekveringsmiljö med målet att kunna:

- Utnyttja beräkningsresurser i ett nätverk av arbetsstationer ("Network of Workstations", NOW)
- Använda ett distribuerat fildelningssystem (modellbibliotek)
- Bygga på ett decentraliserat koncept (P2P-system)
- Exekvera HLA-baserade simuleringar
- Och slutligen skall simuleringar kunna migreras mellan olika noder för att t. ex. få bättre fördelning av resurser eller när resurser på en viss nod inte är tillgängliga längre

Första steget i utvecklingen av en sådan miljö var att identifiera och ev. utveckla metoder för upptäckt och allokering av lediga resurser. Den utvecklade miljön har kopplats till vår webbaserade plattform.

Övrigt

Förutom de ovannämnda spåren utfördes i samarbete med projektet "Nätverksbaserad Modellering och Simulering" dels en studie av kollaborativa metoder och dels en undersökning av lämpliga tekniker, metoder och verktyg för utveckling av en kollaborativ miljö för utveckling och exekvering av modeller. Resultatet från den här delen av arbetet redovisas i kapitel 2.

Dessutom utfördes en undersökning av det senaste inom området genom att följa utvecklingen inom- och utomlands. Ett antal intressanta nationella och internationella projekt identifierades, och ett urval av dem kommer att redovisas i rapporten.

Projektgruppen arbetade också fram ett konferensbidrag som presenterades vid European Simulation Interoperability Workshop 2002 (ESIW '02) i London, England [Ulriksson et al., 2002]. Presentationen mottogs väl av åhörarna och resulterade i ett antal värdefulla kontakter. Dessutom deltog vi under hösten i en internationell konferens om P2P, som hölls i Linköping. Resultatet av besöket redovisades i en reserapport [Sparf, 2002] som visar på aktuella och pågående verksamheter inom området.

För övrigt har projektet haft nära samarbete med National University of Singapore (NUS) inom området komponentbaserad M&S, samt KTH via bl. a. två examensarbetare.

1.5 Rapportens omfattning och struktur

Kapitel 2, Bakgrund, ger en generell beskrivning av den distribuerade simuleringsmiljön med en kort introduktion till HLA, decentraliserade nätverk och kollaborativa miljöer för att skapa förståelse för koncepten och beskriva nyttan för försvaret. Samtidigt ger den en bakgrund till området webbaserade simuleringar. Mer specifikt belyser det nyttan av webbaserade HLA-federationer och webbanpassade

simuleringar. Slutligen diskuteras ett antal relevanta projekt och miljöer. En läsare som redan är bekant med dessa begrepp kan hoppa direkt till kapitel 3.

Kapitel 3, *Utveckling av en webbaserad simuleringsmiljö*, beskriver metodik och implementeringen av en webbaserad miljö för utveckling av HLA-federationer samt för plattformsoberoende användning av befintliga simuleringsmodeller. I detta kapitel beskrivs även utveckling av en distribuerad exekveringsmiljö genom utnyttjande av P2P-teknologi. Som tidigare har beskrivits är den distribuerade exekveringsmiljön en del av den totala webbaserade miljön som har utvecklats inom projektet. Slutligen presenteras erfarenheter och resultaten från utnyttjandet av miljön.

Kapitel 4, *Diskussion*, diskuterar erfarenheter och resultaten från utvecklingen och utnyttjandet av miljön, samt projektets framtida inriktning.

Kapitel 5, *Slutsatser*, presenterar våra slutsatser och områdets potential inom framtidens försvar.

2 Bakgrund

Webbaserade simuleringar är ett nytt växande område inom M&S. Den största drivande faktorn är den fenomenala tillväxten inom "World Wide Web" och åtföljande teknologier. Området har i viss utsträckning uppfattats och utpekats som endast en teknisk framgång och förbättring men inget mer. Detta kan delvis stämma på grund av de tekniska möjligheter som webbt teknologin erbjuder, men det finns också andra uppfattningar. Det finns forskare som anser att webbt teknologin radikalt kan ändra vårt sätt att modellera och simulera. Webbaserade simuleringar som forskningsområde uppkom någon gång under mitten av 90-talet och fick mycket uppmärksamhet under "Winter Simulation Conference", 1996. Sedan dess har området fått en egen konferens "INTERNATIONAL CONFERENCE ON WEB-BASED MODELING & SIMULATION" som går en gång per år i USA. Under vintersimuleringskonferensen 1996 presenterades tre artiklar som behandlade området. Två artiklar beskrev Java-baserade simuleringsprogram och i en artikel [Fishwick, 1996] presenterade Fishwick sina synpunkter om webbaserad simulering som ett nytt delområde inom modellering och simulering. Han beskrev några potentiella effekter av webbt teknologin på simulering, och belyste speciellt tre områden: utbildning och träning, publikationer och simuleringsprogram. Han förklarade t.o.m. spridningen och utvecklingen av webben som 2000-talets guldrush. Andra forskare som Page [Page, 1998] har utgått ifrån Fishwicks observationer och identifierat fler användningsområden. I [Page, 1998] klassificerar Page området webbaserade simuleringar i fem basgrupper; simulering som hypermedia, metodforskning, webbaserade simuleringsprogram, distribuerad simulering och simulering av WWW.

Oavsett hur man beskriver området är kärnfrågan densamma, nämligen vad effekterna av att kombinera webbt teknologin med modellering och simulering är. Eller snarare hur webbt teknologin kommer att påverka vårt sätt att modellera och simulera. Det är nämligen svårt, eller till och med omöjligt, att förutse (eller se) hur mycket webben kan påverka M&S och alla möjligheter som en sådan kombination kan erbjuda.

Webbt teknologin erbjuder en infrastruktur som har potential att kunna revolutionera modellering och simulering. En annan känd infrastruktur med liknande effekter är järnvägen. Introduktionen av järnvägen förändrade, förutom vårt sätt att fysiskt förflytta oss, våra samhällsstrukturer. Det går inte att på förhand förutse alla möjligheter som en ny infrastruktur kan erbjuda, men när den väl är på plats kommer människan att hitta på de mest innovativa sätt att använda den. Ett annat exempel är uppfinningen av monteringsband som revolutionerade bilindustrin och människans sätt att tillverka bilar. Innan monteringsbandet kom var teknikerna tvungna att förflytta sig i fabriken mellan olika stationära komponenter, men efter det blev de stationära medan komponenterna istället blev rörliga.

Vi tror att WWW eller Internet, som infrastruktur kan erbjuda många nya möjligheter för att skapa, distribuera och exekvera simuleringsmodeller.

Tänkbara användningsområden:

Webbt teknologin har potentialen att markant ändra vårt sätt att utveckla (kollaborativt och komponentbaserat), dokumentera (dynamiskt via multimedia), analysera (genom öppen och utbredd utvärdering) och exekvera modeller (genom distribution). Tänkbara användningsområden inkluderar:

- Modeller kan skapas och presenteras på webbsidor. På så sätt kan användare med hjälp av webbläsare (t. ex. Netscape eller Internet Explorer)

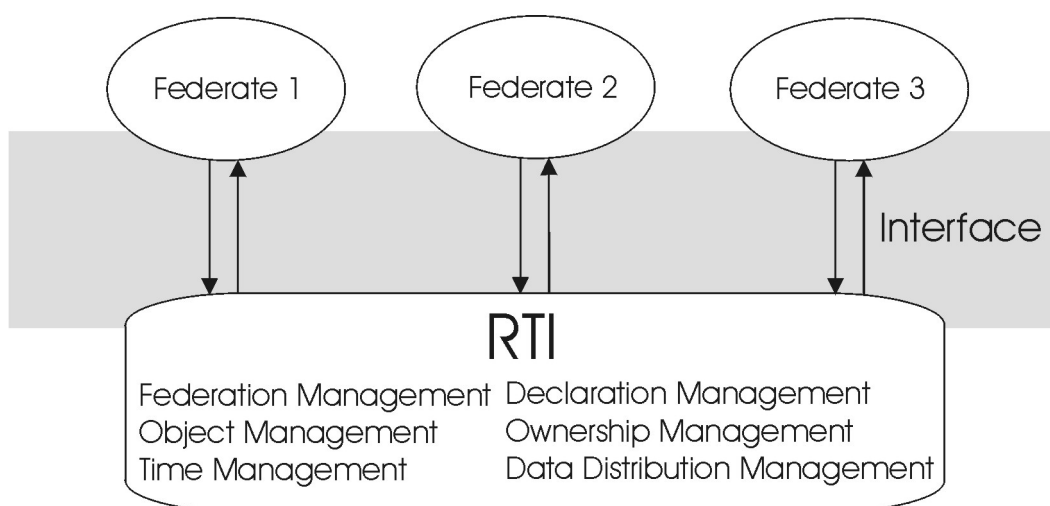
komma åt och exekvera dessa modeller. Genom att utnyttja webbläsare blir simuleringsmodeller mer tillgängliga. Eftersom webbläsare exekverar kompilerade klasser behöver källkoden inte avslöjas.

- Den potentiella användargruppen av datormodeller blir betydligt större genom utbredd distribution av dessa. Den värdefulla återkopplingen från denna användargrupp kan bli avsevärt större än vad som är möjligt idag med de traditionella distributionsmetoderna.
- Modellutvecklare som arbetar med samma projekt och är lokaliserade på olika platser, kan samarbeta via webben och tillsammans konstruera modeller, genom t. ex. komponentbaserad utvecklingsteknik. Senare kan var och en utveckla sina delar av modellen och via webben göra dem tillgängliga för andra medlemmar att se och komma åt.
- Olika användare (t. ex. beslutsfattare), som befinner sig på geografiskt skilda platser, med tillgång till nätverket, kan koppla upp sig mot varandra och i realtid följa samma simuleringar som alla dessutom kan interagera med, t. ex. lägga till nya enheter, ändra förutsättningar, osv.
- Exekvering av simuleringsmodeller kan initieras från klienter, medan modellerna körs på andra maskiner inom nätverket. Genom att distribuera exekveringen kan lediga resurser inom nätverket (såsom minne och bearbetningskapacitet) utnyttjas effektivt.

2.1 Distribuerad simuleringsmiljö

2.1.1 HLA

”High Level Architecture”, HLA, är en standardiserad arkitektur som möjliggör sammankoppling av oberoende utvecklade komponenter till ett simuleringsystem. Med komponenter avses i detta sammanhang simuleringsmodeller eller verktyg som kan återanvändas i simuleringar andra än den som komponenten ursprungligen utvecklades för. HLA stödjer sammankoppling av komponenter utvecklade i skilda programmeringsspråk och/eller operativsystem [Dahman et al., 1997].



Figur 1. Principen för hur tre federater bildar en federation genom att ansluta till RTI och utnyttja de tjänster som HLAs gränssnittsspecifikation definierar.

I HLA benämns en komponent federat och en mängd federater, som utgör en simulering, benämns federation. En federat kan vara en simuleringsmodell, en bemannad simulator, ett supportsystem (system för visualisering eller management av en federation) etc. Federater skapar en federation genom att koppla upp sig mot ett RTI, "Run-Time Infrastructure". Ett RTI kan liknas vid ett distribuerat operativsystem för en federation, se figur 1. HLA-standarden utgörs formellt av tre delar; gränssnittsspecifikation, "Object Model Template", (OMT) och HLA-regler [Kuhl et al., 1999]. Nedan beskrivs dessa tre komponenter översiktligt.

2.1.1.1 HLA gränssnittsspecifikation

RTI erbjuder ett antal tjänster som möjliggör interaktion mellan federater i en federation, samt lägger grunden till management av en federation. De tjänster som finns tillgängliga från RTI under exekveringen av en federation är specificerade i [HLA Interface Specification]. Tjänsterna delas in i sex huvudkategorier och kan översiktligt beskrivas på följande sätt:

1. "Federation Management" hanterar:
 - Skapande av federationer samt federaters anslutning till federationer
 - Sparande av federationer samt återskapande av federationer
2. "Declaration Management" hanterar:
 - Federaters deklaration att erhålla data av viss typ
 - Federaters deklaration att producera data av viss typ
3. "Object Management" hanterar
 - Utbyte av data mellan federater
 - Specificering av hur data transporteras
4. "Ownership Management" hanterar:
 - Hur ansvar för uppdatering av data kan delas mellan federater
 - Hur ansvar för uppdatering av data kan överföras mellan federater
5. "Time Management" hanterar:
 - Federationens tid
 - Kontroll över hur tidsstämplade meddelanden levereras till federater
6. "Data Distribution Management" hanterar:
 - Konsument – producent förhållanden mellan federater
 - Filtrering av information baserat på intresse

2.1.1.2 HLA OMT

HLA definierar två typer av objektmodeller; "Federation Object Model", FOM och "Simulation Object Model", SOM. I en FOM deklarerar de objekt, attribut och interaktioner som delas av federater i en federation. En SOM definierar de objekt, attribut och interaktioner som en federat kan bidra med i de federationer som den kan komma att delta i. HLA specificerar inte vad som definieras i en FOM eller SOM, men HLA kräver att dokumentationen följer en specifik mall, nämligen HLA OMT. En delmängd av federationens FOM krävs av RTI som indata vid simuleringsstarten. Denna delmängd benämns FED-fil [Dahman et al., 1997].

2.1.1.3 HLA-regler

HLA-reglerna specificerar principer och konventioner som federater måste följa. Ett exempel är att under exekveringen av en federation måste samtliga data utbytas via RTI.

Reglerna är nödvändiga för att interaktionen mellan federater skall utföras korrekt under exekveringen av en federation. För en fullständig specifikation av HLA-regler, se [HLA Rules].

2.1.1.4 Hierarkiska federationer

HLA är utvecklat för att underlätta interoperabel och komponentbaserad simulering och används för att skapa standardiserade komponenter som sammankopplas till simuleringar. Det finns dock fördelar med att återanvända en hel federation och använda denna i en ny federation. Detta kallas hierarkiska federationer, eller hierarkisk federationsutveckling och tillåter en mer dynamisk form av återanvändning av simuleringar. HLA har idag inget inbyggt stöd för denna typ av utveckling.

Under de senaste åren har en rad tekniker utvecklats för att hantera hierarkiska federationer. I [Liu et al., 2002] ges en översiktlig beskrivning av de koncept som ofta utnyttjas, nämligen; "federate gateway" FG, "federate proxy" FP, "RTI broker" och "RTI interoperability protocol". En "Federate gateway" skapar kopplingar mellan federater i två eller fler federationer utan att ansluta till RTI. "Federate proxy" är en federat som ansluter till de federationer som den kopplar samman. En "RTI broker" liknar en "federate proxy" med tillägget att den även kommunicerar RTI specifik information mellan två eller fler RTI. "RTI interoperability protocol" är en lågnivå-lösning som utnyttjas inom ett specifikt RTI i syfte att koppla samman lokala RTI-komponenter.

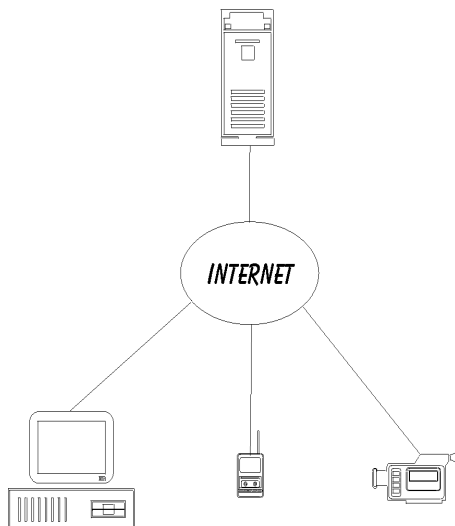
2.1.2 JXTA & P2P

2.1.2.1 Bakgrund

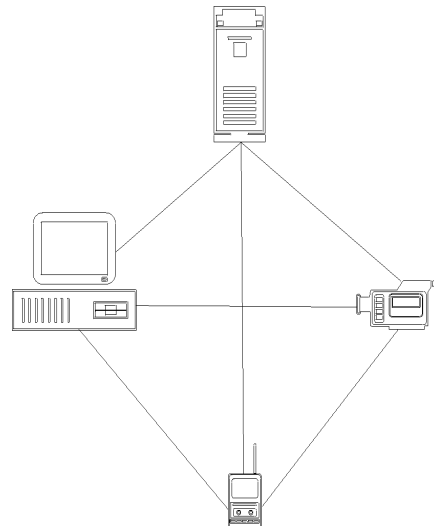
Bandbredd, information och beräkningsresurser inom nätverk skulle potentiellt kunna utnyttjas mer effektivt om befintlig webbt teknik inte baseras på centraliserad kontroll. En majoritet av dagens webbaserade applikationer bygger på det traditionella klient-server-tänkandet, vilket trots kontinuerlig utbyggnad av Internets kapacitet, skapar obalans i resursutnyttjandet. Välutnyttjade delar av nätet blir allt mer hårt trafikerade, medan andra förblir outnyttjade. Samtidigt som processor- och lagringskapacitet ständigt ökar, tenderar utnyttjande av beräkningsresurser att ackumuleras kring ett fåtal centra [Gong, 2001].

Peer To Peer (P2P) är en alternativ nätverksteknologi som inte utesluter, men som naturligt inte bygger på centraliserad kontroll. I ett P2P-nätverk kommunicerar nätverksenheter direkt med varandra, utan stöd från centrala servrar och resurser. På det sättet kan beräkningsresurser och data utnyttjas i varje enskild del av nätverket. De kollektiva resurserna i ett P2P-nätverk är därmed större än i ett centraliserat system [Sundsted, 2001].

Vidare kan en server i ett centraliserat system inte initiera kontakt med en klient, vilket kan ske godtyckligt mellan P2P-noder. Eftersom funktionaliteten inte är beroende av ett fåtal enheter, som i det centraliserade fallet, ökar prestanda i informationssökning och informationsbearbetning. Utöver detta medför P2P-strukturen att den övergripande stabiliteten kan förbättras jämfört med centraliserade system [Gong, 2001].



Figur 2A. Centraliserat "klient-server"-system



Figur 2B. Decentraliserat P2P-system

Figur 2A visar ett centraliserat system likt dagens nätverk, där en rad olika typer av klienter kan utnyttja tjänster som erbjuds av en central webbserver. Figur 2B visar strukturen hos ett P2P-system, där de olika nätverksenheterna kan etablera direktkontakt med varandra utan kontakt med en central enhet.

Fram till idag har P2P-teknologier främst utnyttjats i applikationer för fildelning. Den mest välkända av dessa är troligtvis Napster, som främst användes för distribuering av musikfiler. Användare av Napster installerade en klient och kunde därefter söka och ladda ner musikfiler från andra anslutna användare. Andra liknande programvaror existerar idag i form av Gnutella, MojoNation, Freenet och FastTrack.

Ett centralt koncept inom P2P är förmågan att utföra beräkningar (utnyttja resurser) där det bäst lämpar sig, så kallad resursallokering. Ett projekt värt att nämna i detta sammanhang är SETI@Home. SETI@Home är inte en renodlad P2P produkt, men det belyser en mycket framgångsrik form av resursallokering. Idén bygger på att låta lediga datorresurser utnyttjas för analys av radioastronomiska data, i syfte att upptäcka främmande liv i universum. De stunder som en dator inte utnyttjas startas ett program som laddar ner en mindre mängd radioastronomiska data för analys. När resultatet av analysen är klar skickas svaret tillbaka till SETI@Home. Enligt den senaste statistiken har 1 244 672 år CPU-tid (2002-12-09) utnyttjats för denna typ av analys. Även om framtida P2P-applikationer inte kommer att ha lika stor genomslagskraft som SETI@Home, påvisar detta den resurspotential som finns tillgänglig, men som är outnyttjad, i dagens Internet [SETI@Home].

P2P-plattformar för utveckling av distribuerade applikationer existerar i varierande former, från Java-baserade lösningar, t. ex. JXTA och JINI, till Microsoft-lösningar som UPnP (Universal Plug and Play).

2.1.2.2 JXTA

De flesta av de P2P-produkter som finns tillgängliga på marknaden idag är visserligen kraftfulla men medför oftast begränsningar. Såsom f.d. Napster finns det många P2P-nätverk som endast tillåter delning av vissa filer, exempelvis musikfiler. Många av dessa produkter är även plattformsb beroende. Den typ av produkter som är kommersiellt tillgängliga är dessutom bara tänkta att användas, inte utvecklas individuellt.

Tekniken JXTA är enligt Sun [JXTA Home Page] varken plattformsb beroende eller begränsad vad gäller fildelning. Projektet som tog fram denna teknik skapades som ett öppet projekt för att lägga grunden till distribuerad vidareutveckling av tekniken. Därför är all källkod och demonstrationer m.m. tillgänglig på Suns hemsida. När JXTA-plattformen laddas ner medföljer även exempel, dokumentation och ett antal prototypimplementeringar. Dessa möjligheter gjorde att JXTA valdes som testprodukt i projektet där vi skulle identifiera och utvärdera en teknik som skulle kunna möjliggöra vissa idéer bakom webbaserad simulering såsom t.ex. resursallokering. Med resursallokering menas i det här sammanhanget att resurser i ett nätverk på något sätt kan sökas upp och sedan användas från sitt ursprungliga ställe. En klient skulle t.ex. kunna leta upp en ledig resurs i ett nätverk och skicka dit en process för exekvering i stället för att exekvera den hos sig själv. Med resursallokering avses fortsättningsvis detta.

2.1.2.2.1 JXTA-definitioner

Nedan följer en definition av termer som används i samband med P2P-teknologier. Termerna är hämtade från JXTA men förekommer i samband med ett flertal P2P-plattformar, med viss modifikation. En nod i ett P2P-nätverk benämns "*peer*". En mängd olika typer av objekt, exempelvis en sensor, telefon eller PC, kan utgöra en "*peer*". Det grundläggande villkoret för att enheten kan benämnas JXTA-peer, är att den implementerar ett eller flera av JXTA-protokollen. En samling av dessa enheter utgör en "*peer group*". En sådan grupp omfattar ett antal "*peers*" med ett gemensamt intresse. Gruppen delar resurser och tjänster inom nätverket. I detta sammanhang är det viktigt att belysa att JXTA-protokollen inte dikterar när, var eller varför en grupp skapas [Gong, 2001]. En "*peer*" kan tillhöra ett obegränsat antal "*peer groups*" [Sundsted, 2001].

Alla resurser inom ett JXTA-nätverk representeras av "*advertisements*". Om en användare skapar en ny "*peer group*" publiceras metadata kring denna via ett "*peer group advertisement*" i "*net peer group*". Då varje enskild "*peer*" alltid är medlem av "*net peer group*", kan andra noder hitta och ansluta sig till den nya gruppen. Av alla meddelanden är "*advertisements*" det vanligast förekommande dokumentet som utbyts mellan noder i ett JXTA-nätverk [Gong, 2001].

För att skicka olika typer av data mellan individuella P2P-noder utnyttjas "*pipes*". Dessa kan liknas vid virtuella kommunikationskanaler. "*JXTA pipes*" är envägsriktade och asynkrona kommunikationskanaler. Enligt JXTA-specifikationen finns två generella typer av "*pipes*": "*point-to-point*" och "*propagate*". "*Point-to-point*" erbjuder sammankoppling av två noder där meddelanden kan skickas i en riktning. "*Propagate*" innebär att en enskild nod kan sammankopplas till en eller flera noder och distribuera information till alla [Sundsted, 2001].

I ett P2P-nätverk samarbetar och kommunicerar noder (*peers*) för att publicera, upptäcka och initiera tjänster ("*services*") inom nätverket. Två typer av tjänster kan särskiljas:

- "Peer service"
- "Peer group service"

En "*peer service*" är enbart tillgänglig från den "*peer*" som publicerar tjänsten. Ett flertal instanser av tjänsten kan existera, men varje enskild instans ("*peer*") publicerar själv innehållet. En "*peer group service*" består av en samling instanser av tjänsten (som eventuellt samarbetar) och finns hos ett flertal av gruppens medlemmar. Om en enskild medlem faller bort påverkas inte tjänsten eftersom sannolikheten är stor att den finns

replikerad hos en annan ”peer”. Tjänster av detta slag publiceras kollektivt [Gong, 2001].

2.1.2.2.2 JXTA-protokollen

JXTA plattformen baseras på 6 protokoll som alla är oberoende av varandra. En peer behöver bara implementera det eller de protokoll som är nödvändiga för just den noden. Vid behov kan även endast valda delar av protokollen implementeras.

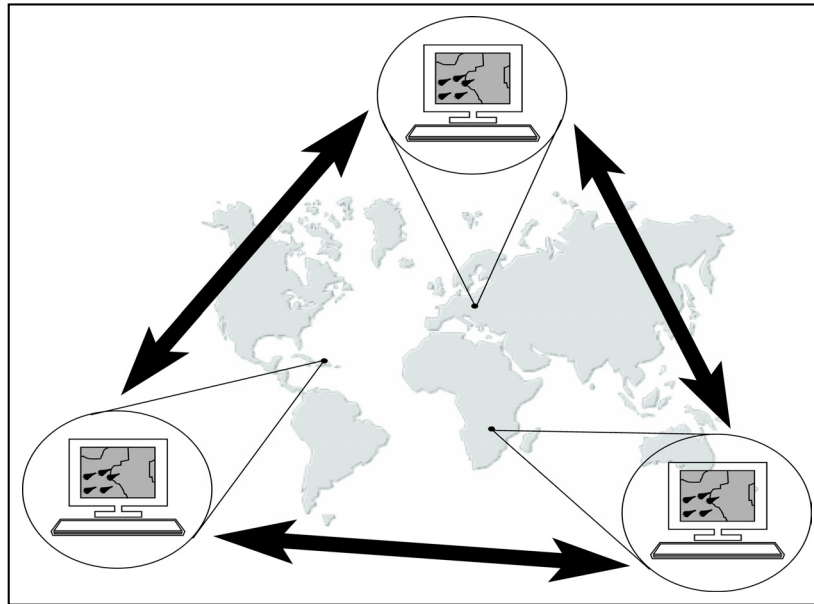
Protokollen tillhandahåller tjänster för att skapa och hantera peers och peer groups mm. Med hjälp av protokollen kan en peer publicera en resurs (exempelvis ett program, eller en fil etc.) och den kan även söka i nätverket efter resurser som en annan peer har publicerat. Samtliga protokoll nämns kortfattat nedan (se [JXTA Home Page] för mer utförligt information):

- PDP – Peer Discovery Protocol
 - Används för att publicera och leta efter resurser i nätverket mm.
- PRP – Peer Resolver Protocol
 - Används för att skicka och ta emot förfrågningar mellan peers. Förfrågningarna rör här exempelvis endpoints och deras status mm.
- PIP – Peer Information Protocol
 - Med hjälp av detta protokoll kan en peer få information om andra peers, exempelvis status och uppdateringar mm.
- PMP – Peer Membership Protocol
 - Detta protokoll tillhandahåller tjänster för att peers ska kunna organisera och gruppera sig själva mm.
- PBP – Peer Binding Protocol
 - Innehåller egenskaper för att t. ex. etablera en pipe, binda pipes till endpoints etc.
- PEP – Peer Endpoint Protocol
 - Används för att söka och optimera användningen av endpoints.

2.1.2.2.3 JXTA och möjligheter inom webbaserade simuleringar

I ett långt perspektiv kan JXTA-teknologin erbjuda en rad möjligheter om den integreras i en webbaserad simuleringsmiljö. En av fördelarna med JXTA är att teknologin är plattformsoberoende. Detta beror på att plattformen bygger på Java-teknologi och informationsöverföringen sker i form av XML-dokument. Oberoendet medför att plattformar av olika typer, med sina respektive simuleringsmodeller, kan ingå i ett webbaserat, distribuerat simuleringsnätverk. Vidare medför utnyttjandet av JXTA (P2P-teknologi) att en förhållandevis stabil simuleringsmiljö kan byggas. I en traditionell ”klient-server”-hierarki är miljön uteslutande beroende av funktionaliteten hos en central enhet. Ett eventuellt bortfall av denna enhet skulle kunna resultera i en total kollaps av hela nätverket. I ett P2P-nätverk har ingen av de enskilda nätverksenheterna den stora betydelsen. En förlust av en enskild nod innebär inte att hela nätverket kollapsar, trots att just den noden erbjuder en unik tjänst. P2P-nätverket fortsätter att fungera om än med begränsad funktionalitet. I det ideala fallet finns tjänsten duplicerad hos en annan nod som gör att funktionaliteten kan upprätthållas trots bortfallet.

Beräkningsresurser skulle även kunna utnyttjas mer effektivt i ett P2P-nätverk jämfört med i ett centraliserat system. Vid simulering i den distribuerade miljön skulle funktioner för resursallokering finnas implementerade. Detta innebär att beräkningar (processer) exekveras där, för stunden, lediga resurser finns tillgängliga inom nätverket. Utnyttjandet av P2P-teknologi i en webbaserad simuleringsmiljö skulle även underlätta utveckling av nya tjänster inom systemet. Enskilda användare skulle kunna lägga till nya modeller/simuleringar genom förändring av den nod som han/hon förfogar över. Genom att undvika vägen via en central enhet skulle arbetet kunna effektiviseras och en kollaborativ miljö för utveckling skulle skapas. (Avsnitt 3.3 ger en utförlig beskrivning av hur JXTA-plattformen har utnyttjats i projektet.)



Figur 3. En kollaborativ decentraliserad miljö

JXTA är en teknik under utveckling och som ständigt får nya användningsområden. Några av de öppna projekten under JXTA skulle kunna vara intressanta för integrering med webbaserade simuleringar. Av speciellt intresse pågår för närvarande ett projekt benämnt JxtaSpaces, inriktat mot distribuering av ett delat minne inom ett JXTA-nätverk. Syftet är att skapa ett system som ger illusionen av ett delat minne, byggt på den meddelandemodell som JXTA utgör. Tanken är att en huvudprocess delar upp ett jobb och placerar delarna i ett virtuellt, delat minne. Därefter hämtar ett antal subprocesser (enskilda noder) de olika delarna och utför bearbetningen. Resultatet deponeras i det virtuella minnet där huvudprocessen samlar ihop bitarna och färdigställer resultatet [JxtaSpaces]. Ett annat projekt av intresse är Parlor som syftar till att skapa en kollaborativ miljö för utveckling av applikationer [Parlor]. Dessa projekt är endast påbörjade och därför bör uppdateringar av projekten kontrolleras kontinuerligt.

Nackdelar med JXTA, framför andra P2P-tekniker, kan vara att tekniken är ny och ännu inte särskilt väl dokumenterad. Den är dessutom inte komplett, utan utveckling sker kontinuerligt i ett "open source" projekt. Vidare kan tilläggas att säkerhetsaspekter i P2P-nätverk i allmänhet inte är särskilt utvecklade.

2.1.3 Kollaborativ modellutveckling och –exekvering

2.1.3.1 Kollaborativ modellutveckling

Många områden påverkas av utvecklingen inom webbt teknologin och några har redan diskuterats i detta dokument. Ett annat viktigt område är kollaborativa miljöer, eller i detta sammanhang kollaborativ modellering och simulering, det vill säga M&S där olika användare kan samarbeta om att modellera eller simulera något. Kollaborativ M&S kan ske i olika former med avseende på tid och plats. Beroende på var projektdeltagarna geografiskt befinner sig, och när de arbetar kan fyra olika situationer uppstå.

- Modellutvecklare befinner sig på samma plats och arbetar samtidigt.
- Modellutvecklare befinner sig på samma plats, men arbetar under olika tider.
- Modellutvecklare befinner sig på olika platser, men arbetar samtidigt.
- Modellutvecklare befinner sig på olika platser och arbetar under olika tider.

Samma plats, samma tid

Detta är ett klassiskt fall, där medlemmar i en projektgrupp tillsammans under möten och diskussioner utvecklar modeller. I den här typen av samarbete sker kommunikation och överföring av kunskap huvudsakligen i form av muntliga samtal och med hjälp av presentations- och dokumentationsmaterial. Alla idéer presenteras, diskuteras och dokumenteras under möten. Utvecklingsarbetet kan antingen ske under samma former eller så kan arbetet delas, där var och en utvecklar sina delar separat. Gruppen sätter ihop delarna och testar modellen under liknande möten. Utnyttjandet av nätverk och webb är begränsat i det här fallet.

Samma plats, olika tider

Det här fallet lämpar sig kanske mindre väl för modellutveckling och påminner mest om skiftarbete inom verkstadsindustrin. Arbetet delas in i ett antal delmoment och var och en utför sin del. Synkronisering av arbetet sker via meddelanden (anteckningar, filer, etc.) som lämnas till andra medlemmar i gruppen. I detta fall är också användningen av webb och nätverk begränsad.

Olika platser, olika tider - olika platser, samma tid

Dessa fall är de som påverkas mest av webbt teknologin och här skulle webben kunna erbjuda stora möjligheter. Modellutvecklare som arbetar med samma projekt och är lokaliserade på olika platser kan samarbeta via webben och tillsammans konstruera modeller, genom t. ex. komponentbaserad utvecklingsmetodik. I princip skulle medlemmarna kunna komma åt samma miljö och interagera med modellen och varandra samtidigt eller under olika tider. När konstruktionsfasen är klar kan arbetet delas bland medlemmarna, där var och en utvecklar sina delar av projektet och via webben gör dem tillgängliga för andra medlemmar att se och komma åt.

Med t.ex. Javas möjligheter att importera externa klasser, kan nya delprojektklasser snabbt skapas och kopplas till existerande projektklasser. Eftersom klasserna själva (och inte deras kopior) används och testas av andra projektmedlemmar blir sannolikheten för fel, som kan uppstå på grund av felaktiga kopior, betydligt mindre. Att tillgängliggöra kompillerade klasser hjälper också till med att öka tillförlitligheten för det totala projektet. Genom att accelerera tillgängligheten av prototypklasser till projektet, kan hela projektet bli mer robust och samtidigt mer tidseffektivt.

2.1.3.2 Kollaborativ modellexekvering

Webben kan dessutom erbjuda stora möjligheter för kollaborativ exekvering av modeller. Användare med tillgång till nätverket kan koppla upp sig mot varandra och i realtid följa en simulering som alla dessutom kan interagera med, t.ex. lägga till nya enheter, ändra förutsättningar mm. Detta skulle vara mycket värdefullt för t.ex. distribuerade stabsövningar. Denna teknik existerar idag och utnyttjas i högsta grad av spelindustrin. Nätverksbaserade datorspel har inom de senaste åren varit en enorm kommersiell succé. Nätverksbaserade spel som Ultima Online, Everquest, Doom, Quake, Diablo II och många andra har engagerat miljontals spelare över hela världen. I den här typen av spel har varje deltagare en version av spelet installerat på sin dator. En komplettering i det här fallet vore att möjliggöra för användarna att komma åt spelen via webben, och härmed slippa installera någon programvara.

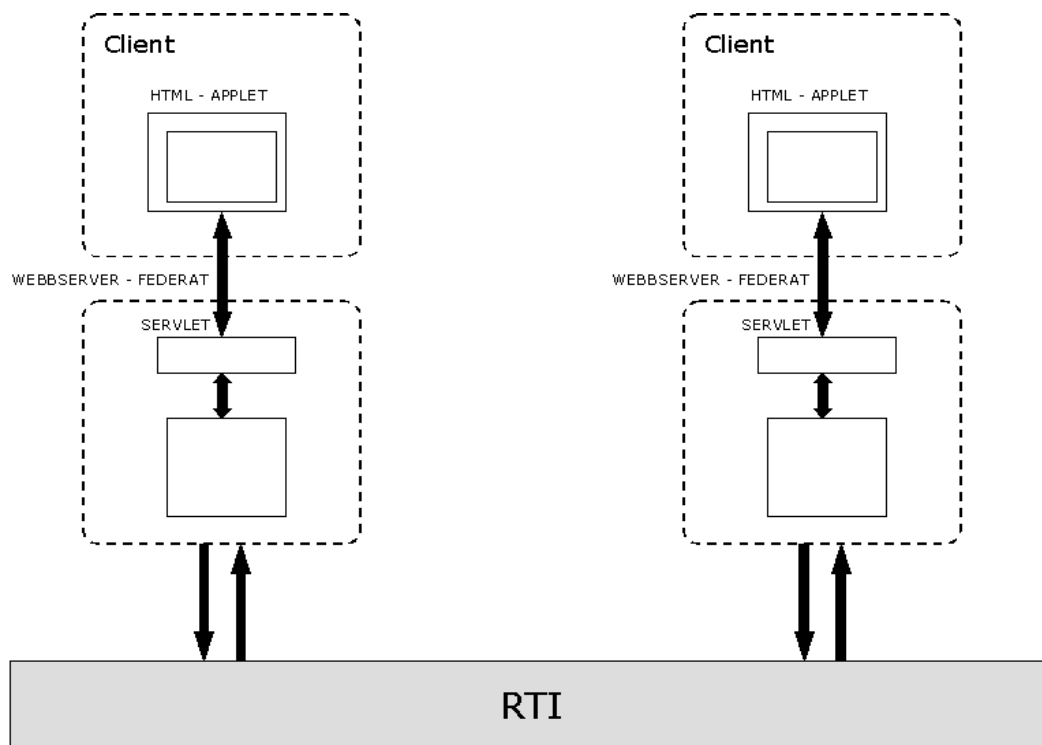
2.1.3.3 Implementering

Inom projektet studerades olika möjligheter för att implementera kollaborativt stöd i den webbaserade miljö som utvecklades. Eftersom HLA, Java och JXTA redan används i miljön beaktades dessa tekniker som alternativa metoder för implementering.

HLA – RTI

HLA-arkitekturen och RTI skulle kunna användas för att implementera kollaborativt stöd. Detta skulle kunna ske genom att låta klienter agera federater i en federation. Federationen utgörs här av gruppen användare i den aktuella tillämpningen. I det här fallet kopplar en användare upp sig mot WWW och hämtar en HTML-sida för applikationen. När användaren loggar in på sidan går den automatiskt in som en federat i en särskild federation, en användargrupp. Denna federation fungerar i sammanhanget precis som en "peer group", se figur 4 nedan. Federationen, förutom dessa federater, består av applikationer som erbjuder ett antal tjänster. Tjänsterna kan till exempel vara att leta efter eller publicera resurser och motsvarande, samt tillåta samverkan (kollaboration) vid vissa uppgifter etc.

För att klienten ska kunna utgöra en federat kan flera metoder användas. En är att låta webbservern hos klienten utgöra federaten. Det vill säga webbservern modifieras så att hela servern blir en federat. Klienten har i sin tur kontakt med federaten på liknande sätt som exempelvis användes i WebSim 2001, det vill säga en applet har kontakt med en servlet som i sin tur upprättar kontakt med federaten i webbservern och får således kontakt med federationen. På detta sätt kan användare arbeta kollaborativt med stöd från RTI.



Figur 4. HLA/RTI används som mellanlager för kollaboration

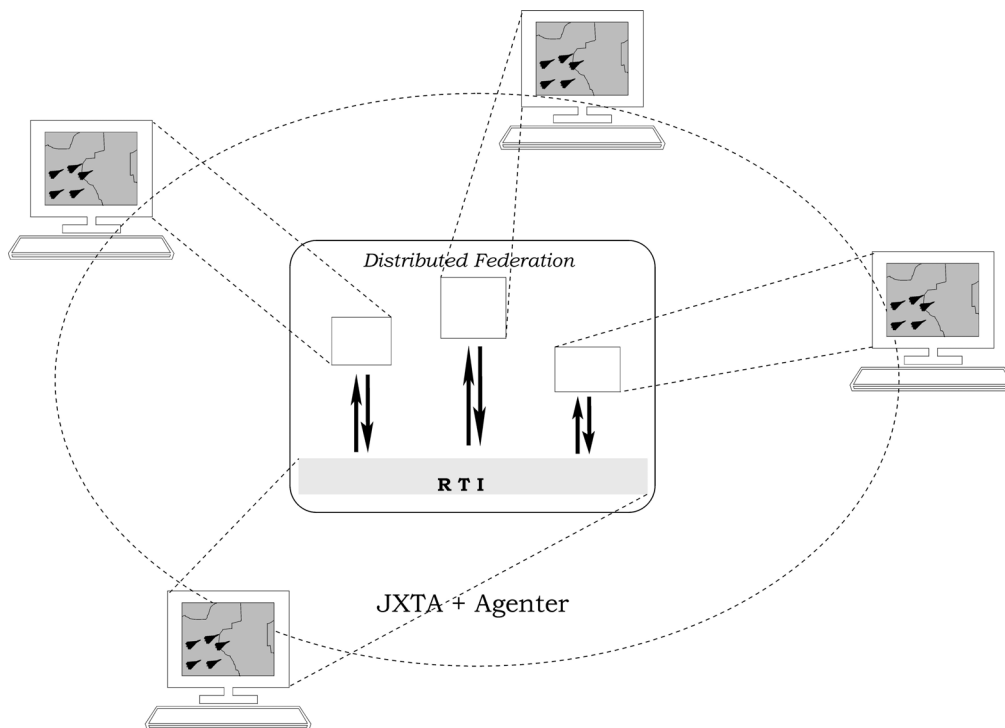
JXTA / JXTA + agenter

Ett annat alternativ för att åstadkomma M&S-samarbete kan vara att bygga vidare på, och/eller modifiera, ett P2P-nätverk som redan finns och är ”open-source”⁶. En lämplig P2P-teknik kan till exempel vara JXTA. I JXTA finns tjänster som tillåter publicering av och sökning efter resurser, skapande av grupper mm. Däremot finns idag inga tjänster som tillåter kollaborativ utveckling eller kollaborativ exekvering och inte heller funktioner för synkronisering etc..

JXTA skulle behöva modifieras för att stödja kollaborativ utveckling. Detta skulle kunna göras genom att antingen utöka JXTA med en ”kollaborationsmodul”, som innehåller de tjänster som behövs, eller så kan någon annan teknik, såsom JavaBeans eller agenter, användas i kombination med JXTA. Båda metoderna är intressanta att titta på. Viktiga aspekter som bör tas i beaktning vid ett beslut om detta är: Hur arbetsamt är det att bygga vidare på JXTA respektive hur mycket vinner man på att använda någon annan särskild metod.

En idé är att använda JXTA i grunden för de tjänster som JXTA redan erbjuder och tillföra ett lager med agenter som sköter koordinering och synkronisering av deltagare mm. Att använda agenter möjliggör här utbyggbara och återanvändbara komponenter, som samverkar för att kunna erbjuda klienterna tjänster som till exempel samarbetsstöd. Agenterna är inte heller bundna till JXTA-tekniken varför de även kan kopplas bort och till en annan teknik om det skulle visa sig att JXTA inte är önskvärd.

⁶ ”open-source” betyder att källkoden till applikationen är öppen så att en person vid behov kan modifiera koden för sitt eget syfte.



Figur 5. JXTA och agenter utgör kärnan för kollaborativt arbete

Nackdelar med och frågeställningar kring diskuterade modeller

Att använda HLA/RTI som ett mellanlager för att implementera kollaborativt stöd är ett intressant alternativ och det vore även intressant att se om liknande arbete har utförts tidigare. Den frågeställning som denna metod medför är huruvida det verkligen är fördelaktigt att använda HLA och RTI, eller om det är enklare och mer praktiskt i längden att använda en annan teknik, såsom P2P. HLA erbjuder inte alla de tjänster som kan tänkas behövas i miljön, vilket medför att dessa måste implementeras separat som ett tillägg till RTI eller klienterna. Å andra sidan kanske det är en fördel att HLA används, då även simuleringsmiljön baseras på HLA och RTI.

En P2P-teknik som JXTA kan medföra begränsningar i implementering och arkitektur, då teknikerna inte är utvecklade för detta specifika syfte. Men en sådan teknik kan erbjuda tjänster som ändå behövs och måste implementeras och det blir därför mer tidseffektivt att använda en sådan.

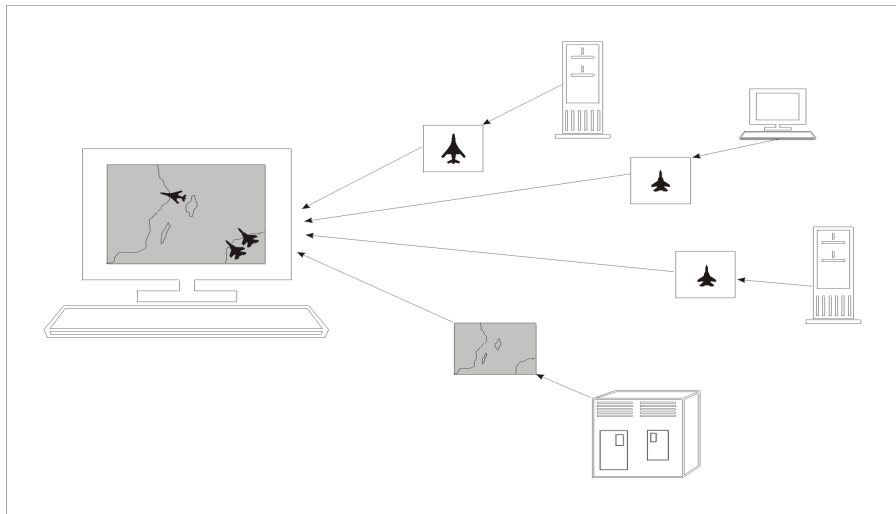
Att istället för att bygga vidare på JXTA utveckla stödet för kollaborativt arbete som ett separat lager baserat på någon annan teknik, exempelvis agenter, ger en mer återanvändbar metod. Det extra lagret (agenterna) blir mindre beroende av en särskild teknik (JXTA) och detta är positivt. Men frågan är om det blir för komplext eller tidskrävande att utveckla detta i jämförelse med att vidareutveckla JXTA.

2.2 Webbaserade HLA-federationer

2.2.1 Introduktion

En distribuerad simulering är en simulering som körs från flera olika platser i ett nätverk, se figur 6. Det vill säga simuleringen delas upp i delsimuleringar som var och en körs på en egen plats (i en nod i nätverket), men som tillsammans bildar den totala simuleringen. Simuleringar kan på detta sätt köras antingen över ett lokalt nätverk eller över Internet.

För att delsimuleringar tillsammans ska kunna delta i en gemensam simulering och interagera med varandra, måste de kopplas ihop via ett gemensamt, standardiserat gränssnitt. Vidare är det önskvärt att ha en koppling till någon central enhet, eller programvara, som håller ihop den totala simuleringen. Detta medför att simuleringsmodeller måste vara utvecklade enligt en standardiserad metod och den sammanfogande programvaran måste i allmänhet vara installerad i varje nod.



Figur 6. Distribuerad simulering

Om simuleringsmiljön vore webbaserad skulle förhållandena se annorlunda ut. Då skulle istället en klient endast behöva en webbläsare och tillgång till ett nätverk, i vilket simuleringen finns, för att kunna köra simuleringen. Simuleringen skulle exekveras på någon central och betydligt kraftfullare dator och klienten skulle kunna utveckla och styra simuleringen från sin webbläsare. Detta betyder dels att klienten inte behöver ha den specifika programvaran för simuleringen installerad och även att klienten kan utnyttja resurser från en annan plats i nätverket.

En webbaserad miljö, som den ovan beskrivna, medför inte bara fördelar som plattformsoberoende, utan ökar även tillgängligheten för simuleringsmodeller och möjligheterna till återanvändning m.m. Härmed kan arbete och kostnader för simuleringsutveckling minska betydligt, förutsatt att modellerna är väldokumenterade. Fördelarna med webbaserade simuleringar tas upp i bl.a. [Chien-Chung Shen, 1998]. Shen diskuterar tre problem som ofta uppstår med simuleringar idag: Simuleringarna är inte överförbara – de är exempelvis språkberoende; de är inte interoperabla – de är t.ex. plattformsoberoende; och sist men inte minst så är simuleringarna inte webbportabla – det vill säga de går inte att köra på webben. Han diskuterar i sitt papper en möjlig lösning på dessa problem, men involverar inte distribuerade simuleringar.

En annan intressant aspekt med en webbaserad simuleringsmiljö är att utöka miljön till en kollaborativ utvecklingsplats. Det vill säga klienter skulle kunna utveckla simuleringar tillsammans och samtidigt. Detta skulle innebära att om en klient adderar en simuleringskomponent till simuleringen så ser alla andra klienter förändringen och vice versa. Resultatet blir en interaktiv kollaborativ utvecklingsmiljö för simuleringar.

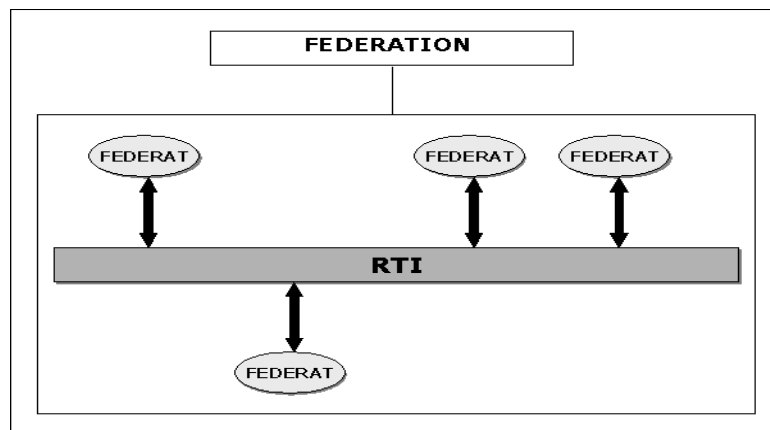
2.2.2 Nyttan med webbaserade HLA-federationer

Simuleringar i ett system likt det ovan beskrivna bör vara uppbyggda av standardiserade komponenter för att vara interoperabla. För detta syfte finns idag flera

lösningar på objektorienterat baserade distribuerade simuleringar, och den kanske största utvecklade strukturen för detta är DMSO's "High Level Architecture" (HLA). Denna teknik har studerats och med fördel använts i flera projekt, främst militära, för att erhålla strikt objektorienterade simuleringar där federater sammanfogas till en federation.

Att använda HLA i webbaserade simuleringar har diskuterats tidigare bl.a. i [Page, 1998] och eftersom HLA redan är studerat och även i viss utsträckning används på FOI, och eftersom arkitekturen är öppen, så valdes HLA-standarden för den tänkta miljön. Enligt HLA utvecklas fristående federater med ett standardiserat HLA-gränssnitt. Federaterna kopplas sedan samman till en federation via det gemensamma gränssnittet med hjälp av en programvara som HLA tillhandahåller, d.v.s. RTI (Runtime Infrastructure). Federationen hålls härmed samman av RTI och all kommunikation i scenariot går därigenom och inte direkt mellan federaterna, se figur 7 nedan. För mer information om HLA, se [DMSO HLA].

Språket som valdes för implementeringen av testmiljön var Java eftersom det är ett webbkompatibelt och plattformsoberoende programmeringsspråk. Java har ofta valts vid andra liknande försök, till exempel till det som beskrivs i [Chien-Chung Shen, 1998].



Figur 7. Schematisk bild av federater och RTI i HLA

HLA är idag väl utvecklad och dokumenterad varför den passar mycket bra som teststruktur för projektet. Nackdelar kan finnas med att binda sig till en standard, men denna nackdel går ännu inte att komma ifrån. Viktiga frågor som uppkommer med användningen av HLA-federater i den webbaserade miljön, är bl.a. hur federaterna görs webbaserade. Vilka tekniker och metoder finns att tillgå och vilka är mest respektive minst fördelaktiga. Projektets HLA-inriktade del inriktar sig på att svara på dessa frågor och att även studera möjligheter med användning av just HLA-komponenter i en webbaserad miljö. Bland annat så studeras möjlig utveckling av s.k. superfederationer där federationer byggs upp av federationer, d.v.s. superfederationer bestående av federater som i sig är federationer. Försök till implementation av den webbaserade HLA-miljön görs också enligt idén med applets och servlet. I detta arbete ingår även ett examensarbete som utförts på FOI för KTH och som beskrivs mer utförligt i kap. 3.

2.2.3 Komponentbaserade simuleringar

Simuleringar består ofta av många olika delsimuleringar. Ett exempelscenario kan vara ett flygplan som söker av ett område och hittar ett mål som beskjuts. Scenariot består av ett plan, en miljö - terräng och luftrum - och ett mål. Men planet består kanske

i sig av en plattform, flygplansdynamik (rörelsesystem), sensor, vapensystem etc. Precis som i den objektorienterade världen kan delar av simuleringen - sensorn, plattformen etc. - separeras från varandra till delsimuleringar – s.k. federater. Federaterna kan köras helt individuellt och även tillsammans med andra federater än de från början tänkta. Dessa federater måste då vara standardiserade och ha ett interoperabelt gränssnitt för att uppnå den objektorienterade strukturen. I [Page, 1999] diskuteras fördelarna med att ha komponentbaserade simuleringar och predikteras även den fortsatta utvecklingen inom simuleringsvärlden som komponentbaserad.

Att bygga upp sammansatta simuleringar (s.k. federationer) av federater har många fördelar, varav den främsta är att komponenterna (federaterna) kan återanvändas. Men fortfarande måste de datorer där federaterna körs ha den programvara som krävs för att koppla ihop komponenterna. Om simuleringskomponenterna i stället har ett webbaserat gränssnitt så uppstår inte detta problem. Komponenterna kan tänkas ligga på en server och klienten (användaren) behöver bara en webbläsare för att använda och koppla ihop dem. Hela simuleringen körs därefter på servern och klienten ser resultatet som om den kördes i klientens egen dator. Fördelen med komponentbaserade simuleringar är också att federaterna kan köras på olika platser i nätverket (distribuerade simuleringar) bl.a. för att maximera resursutnyttjande. Detta skulle kunna uppnås exempelvis med användning av Java-teknikerna applets och servlets.

Klienten har en webbläsare och har tillgång till ett nätverk, t.ex. ett decentraliserat nätverk, och utgör alltså en nod i nätverket. När klienten ansluter sig laddas en applet ner hos klienten, se stycke 2.3.2 för mer information om applets. I appleten finns det användargränssnitt i vilket klienten utvecklar sin simulering, dvs. kopplar ihop federater till en federation. Appleten kommunicerar med en servlet som ligger någonstans i nätverket, till exempel på en server. När användaren utvecklar simuleringen finns tillgång till alla komponenter inom nätverket, oavsett vilken plats de ligger på. Servleten letar upp önskade komponenter som användaren vill ha i sin federation och kör federaterna i deras respektive noder. Detta innebär att resurserna blir distribuerade och komponenterna återanvändbara, något som är mycket önskvärt i simuleringsvärlden idag.

2.3 Webbanpassade Simuleringar

2.3.1 Nyttan med webbanpassade simuleringar

Inom FM finns idag ett stort antal simuleringsmodeller som utnyttjas föga. Dessa modeller anses inte vara olämpliga för simulering, men av olika anledningar förblir de outnyttjade. Detta kan bero på att programmen endast finns tillgängliga på en ”maskin” (plattform) som få har kunskap om, eller för att programmen är komplicerade och svåransvända. Oavsett anledningen resulterar detta i att simuleringsmodellen får liten spridning hos användare, och ekonomiska medel som har satsats på utveckling förräntas dåligt.

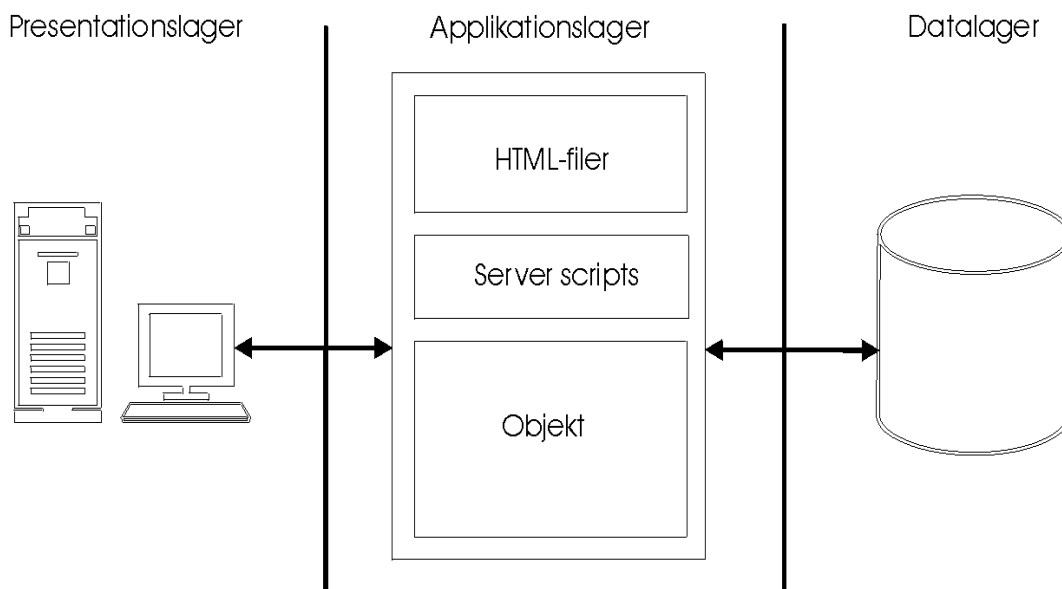
För att sprida simuleringsmodeller till en bredare användarkrets kan webbt teknik utnyttjas. Målet är att göra webbaserade versioner av programmen som därmed kan exekveras från en vanlig webbläsare. I och med detta krävs mycket lite i form av datorresurser på klient-sidan för att utföra en simulering. En dator eller mobiltelefon med anslutning till Internet och med tillgång till en webbläsare, kan välja att köra en viss typ av simulering. Simuleringen belastar inte klienten som vill utföra simuleringen, utan exekveras av webbservern. Efter eller under exekvering skickar webbservern simuleringsresultatet till klienten, där det presenteras på lämpligt sätt.

En annan viktig aspekt av att göra webbaserade versioner av simuleringsmodeller, är att användandet kan effektiviseras. Ett svårhanterligt program med begränsat användargränssnitt kan med små medel göras mer användarvänligt. Användargränssnittet kan integreras med hjälpfunktioner som förenklar för användaren att utnyttja systemet och minskar felaktig användning. Förutom att ett bra användargränssnitt är mer visuellt tilltalande, kan det även bidra till att tiden det tar för att förbereda en simulering reduceras.

2.3.2 Lämpliga webbt tekniker

För att webbanpassa en simuleringsmodell krävs att den omformas till en serverbaserad applikation. Beroende på modellens natur kräver omvandlingen olika grad av förändring i programmet. En simuleringsmodell med ett rikt grafiskt gränssnitt kräver generellt sett större ingrepp. Följande avsnitt ger en översiktlig bild av tekniker som kan utnyttjas för serverbaserad applikationsutveckling. Ingen hänsyn tas dock till när en teknik är mer lämplig än andra för webbanpassning av en viss typ av simuleringsmodell.

Figur 8 visar en generell struktur för webbaserade applikationer bestående av presentations-, applikations- och datalager. I presentationslagret finns klienten (webbläsare) där information presenteras i form av html-sidor och/eller applets. Nästa steg är applikationslagret som utgörs av en webserver innehållande html-sidor, JSP-sidor, ASP-sidor, CGI-script, servlets, osv (dessa beskrivs nedan). Här genereras det innehåll som presenteras för klienten. Det tredje lagret består av databaser av skilda slag som betjänar applikationslagret med data.



Figur 8. Generell struktur för webbaserade applikationer, bestående av presentations-, applikations- och datalager.

En serverbaserad applikation har i stort sett samma egenskaper som ett vanligt program, men med några undantag. Programmet skall kunna aktiveras av en webserver. Webbservern måste kunna skicka formulärdata till programmet. Efter bearbetning av data måste programmet förpacka och sända resultatet tillbaka till webbservern, som i sin tur sänder det vidare till webbläsaren. För ett antal år sedan utnyttjades främst "Common Gate Interface" (CGI) för att bygga serverbaserade applikationer. Microsoft och Netscape skapade båda särskilda API:s ("Application

Programming Interface”) som kunde utnyttjas för webbutveckling med CGI. På senare år har tekniker som Active Server Pages (ASP), Java servlets och Java Server Pages (JSP) börjat utnyttjas i allt större omfattning [Patzner, 1999].

CGI

Ett CGI-program har fördelen att i stort sett kunna utvecklas i vilket språk som helst. I praktiken sker det oftast i PERL. Webbservern som implementerar CGI fungerar som en sluss mellan förfrågan från användaren och data som erfordras. För varje enskild förfrågan skapar webbservern en ny process. Tillvägagångssättet skapar skalningsproblem om ett stort antal förfrågningar sker simultant. Alternativa lösningar har skapats för att kringgå problemet (FastCGI etc.), där en instans har förmågan att delas av flertalet förfrågningar [Patzner, 1999].

ASP

ASP kombinerar HTML, skript och serverbaserade komponenter i en och samma fil. Stöd finns för ”JScript” (script-språk baserat på Java) och ”VBScript” (script-språk baserat på Visual Basic) som kan utnyttjas tillsammans med COM-objekt (Component Object Model). När en klient begär en ASP-fil returnerar webbservern en webbsida till webbläsaren. En fördel med ASP är att allt exekveras på servern, vilket innebär att webbsidan är oberoende av webbläsare. Nackdelar existerar i form av beroendet av Microsoft-teknologi, vilket inte främjar plattformsoberoende [Patzner, 1999].

Applets

Applets är små applikationer som laddas ner från webbservern, transporteras via Internet och slutligen installeras och exekveras i en webbläsare (klienten). För att erhålla en tillfredställande säkerhet är en applets resurser begränsade i klientdatorn. Det är t.ex. inte möjligt för en applet att skriva till klientdatorns hårddisk, eller ha fullständig kontroll över nätverksanslutningar.

Servlets

Servlets är server-baserade program som behandlar ”*http-requests*” och returnerar ”*http-responses*”. En servlet är en generisk utökning av en server och kan liknas vid en applet utan grafiskt gränssnitt. Applets och servlets har liknande livscyklar och exekveras båda i en ”Java Virtual Machine” (JVM). Till skillnad från appleten, som är ett litet program som laddas ner och körs hos klienten, så körs servleten som ett program på servern och utökar härmed serverns funktioner. Servlets är också till skillnad från CGI byggda kring ett flertrådat system med en instans av applikationen. Utöver detta medför Java-teknologin att servlets är plattformsoberoende [Patzner, 1999].

JSP

JSP liknar ASP till viss del. En JSP-sida kan innehålla HTML, JAVA-kod och JavaBeans. När en klient begär en JSP-fil skapar webbservern en motsvarande servlet, som aktiveras och returnerar en webbsida till webbläsaren. I JSP är det möjligt att konstruera anpassade taggar (”*custom tags*”) som kan utnyttjas för att utöka JSP’s tagbibliotek. På detta sätt kan utveckling förenklas genom att XML-liknande taggar används istället för skriptkod [SUN].

2.4 Relaterade projekt och miljöer

Nedan följer en översiktlig beskrivning av ett antal intressanta nationella och internationella projekt inom området webbaserade simuleringar.

2.4.1 Simulation Based Acquisition (SBA) Demonstrator

SBA-projektets syfte har varit att visa på möjligheterna med modellering och simulering som stöd för materielanskaffningsprocessen, samt bygga en teknisk miljö för

detta ändamål. Projektet var en FMV-beställning som utfördes under hösten 2000 av Pitch AB och SaabTech Systems AB. Visionen med projektet har varit att skapa en integrerad miljö som stöd till materielanskaffningsprocessen, där kravställare (FM), beställare (FMV) och leverantörer kan samarbeta och samverka såväl inom olika faser av ett projekt som mellan olika projekt.

Projektet behandlar både process-, kultur- och teknikfrågor. För att lösa tekniska frågor har projektet implementerat en utvecklingsmiljö som består av fyra delar, en konstruktionsmiljö, en exekveringsmiljö, en utvärderingsmiljö och ett bibliotek av befintliga modeller. Aktörerna skall kunna söka efter delmodeller i biblioteket och på ett komponentbaserat sätt utveckla modeller som sedan kan köras i en distribuerad datormiljö. Exekverade modeller skall också kunna analyseras och utvärderas.

2.4.2 ALICE (Advanced, Scalable and Integrated Computing Engine)

ALICE är ett pågående projekt vid universitetet i Singapore, vars mål är att skapa en miljö för effektivt utnyttjande av nätverksresurser, d.v.s. minnes- och bearbetningsresurser. Projektets nytta definieras enligt följande:

- Att minimera kostnader genom att maximera användningen av befintliga datorresurser.
- Lösa kortsiktiga behov av kraftiga datorresurser utan att behöva göra stora investeringar.
- Att erbjuda tillgång till dyra datorresurser och utrustningar, och stora och svår-distribuerade datamängder.

Miljön består av tre huvuddelar, en klientdel, en serverdel och en resursfördelare. Klienter konsumerar datorresurser. Servrarna erbjuder lediga datorresurser och resursfördelarna koordinerar allokering av lediga resurser för att svara mot klienternas prestanda-behov. Systemet har också algoritmer för att hantera distribuerad lastbalansering. Detta innebär att det finns mekanismer för att fördela uppgifter jämnt över befintliga servrar [ALICE].

2.4.3 SIMULTAAN (holländska för simultan)

Simultaan är ett holländskt samarbetsprojekt med sex deltagande företag. Projektets fokus har varit att skapa en infrastruktur för komponentbaserad utveckling av träningssimulatorer för komplexa applikationer inom bl. a. transportindustrin och försvar, samt skapa ett bibliotek av komponenter. Projektets mål har varit att:

- Utveckla en generisk Simultaan Simulator Arkitektur (SSA)
- Utveckla en permanent kunskapsinfrastruktur
- Skapa en permanent nätverksstruktur
- Flytta kunskap mellan olika partner inom Simultaan

Projektet har utvecklat en objektorienterad design för SSA, som består av ett antal (generiska) simulatorkomponenter som kan kommunicera med varandra. Komponenterna i SSA följer en bestämd standard och kan därmed utvecklas samtidigt (Simultaan) på olika håll. Olika partner kan därmed utveckla och underhålla sina delar av infrastrukturen och lokala infrastrukturer kan integreras i en stor virtuell infrastruktur, där alla partners produkter blir tillgängliga för varandra. Detta görs inom en så kallad "Simultaan Virtual Simulator Factory".

Simultaans simuleringsmiljö består av ett antal nätverksbaserade simulatorer med tillhörande scenariohanteringssystem. Gränssnittet mellan olika komponenter utgörs av HLA, och kommunikationen sker via RCI (Runtime Communication Infrastructure), som är baserad på RTI [Simultaan].

2.4.4 NanoComp

Detta är ett pågående forskningsprojekt vid Delft tekniska högskola i Holland. Målet med projektet är att skapa en generisk, kollaborativ miljö för forskning och utveckling. Avsikten är att underlätta tillgång till, samt distribution och körning av modeller. I miljön finns stöd för modeller i alla slags former och formalismer. Och slutligen vill man integrera existerande heterogena verktyg till ett sammanhållet beräkningsstöd. Motivationen är att tillhandahålla simuleringsjänster via webben, att kunna dela och återanvända heterogena verktyg som är spridda på många olika platser, att åstadkomma ett transparent modellutnyttjande bland alla samarbetsparter, samt att se till att man får en enkel utbyggbarhet.

I systemets front-end-lager finns en agentmiljö, som innehåller förmedlingsagenter och gränssnittsagenter. Här finns också användargränssnitt o.s.v. I närmast underliggande lager finns en "Processing Platform Controller", som synkroniserar alla användare. Dessutom finns här ett system för distribution av objekt. I det tredje lagret finns slutligen alla de verktyg och modeller som är tillgängliga på nätet [NanoComp].

2.4.5 SimKit

SimKit är en uppsättning Java-klasser för att skapa händelsestyrda och diskreta simuleringsmodeller. Programmet är utvecklat vid "Naval Postgraduate School" i USA och erbjuder en miljö med enkla simuleringsverktyg i ett objektorienterat språk som kan köras på olika typer av datorer. Modellerna i SimKit kan antingen utvecklas som fristående modeller eller Java-applets. Målet med projektet har varit att skapa ett simuleringsverktyg som:

- Är enkelt att använda, utan behov av professionell kompetens eller hjälp
- Är tillförlitligt nog för inte alltför stora projekt
- Kan erbjuda och förenkla återanvändning av modeller
- Är kostnadseffektivt både vad gäller tid och resurser
- Är kapabelt för att lösa verkliga problem
- Tillåter utnyttjande och användning av avancerade simuleringskoncept, som distribuerad simulering

Verktyget består av tre Java-paket (Javasim, AWT och Util), där vart och ett innehåller en mängd Java-klasser. Javasim är kärnan i SimKit och innehåller klasser för att bygga diskreta modeller. AWT erbjuder en begränsad mängd färdiga grafiska element som används av Javasim. Och Util-paketet innehåller stödfunktioner för simuleringsmodeller. Detta inkluderar till exempel datainsamling, slumpvalsgeneratorer och datastrukturer som lagrar statistisk data.

2.4.6 Silk®

Silk® är ett Java-baserat modelleringsverktyg som används för simulering, studier och förbättring av dator-, tillverkning- eller tjänstesystem. Silk® är ett generellt simuleringsspråk som är baserat på Java och är utvecklat vid Threadtec.com i USA. Silk® slår samman bekanta processororienterade modelleringsstrukturer med objektorienterade språkegenskaper i en intelligent design. Den erbjuder enkel

modellstruktur och återanvändbarhet genom utveckling och visuell sammansättning av modelleringskomponenter i en JavaBeans-baserad grafisk miljö.

Silk[®] är ett exempel på hur man kan använda Javas plattformsoberoende och komponentbaserade egenskaper för att bygga kraftfulla webbaserade utvecklingsmiljöer [Silk[®]].

2.4.7 Nätverksbaserad Modellering och Simulering (NBMS)

NBMS är ett strategiskt projekt vid FOI, som började 2002 med målet att undersöka möjligheter och förutsättningar och utveckla/identifiera metoder/tekniker/verktyg för att bygga en miljö för design, utveckling, dokumentation, exekvering och analys av Web- och komponentbaserade simuleringsmodeller, samt bygga en demonstrator för att visa på fördelarna med en sådan miljö. Miljön skall vara en integrerad infrastruktur för nätverksbaserad modellering och simulering (NBMS), som stödjer processer som FEDEP (Federation Development and Execution Process), SBM (Simulation-Based Methodology) och VV&A (Verification, Validation and Accreditation) [NätSim, 2002].

HLA är den etablerade arkitekturen inom distribuerad simulering. Det främsta syftet med HLA är att[®] möjliggöra samverkan mellan och återanvändbarhet av modeller. Idag är det osäkert om HLA är en fullständig arkitektur och kan agera som kärnan inom framtida NBMS system. Idag är t ex HLA inte alltid lämplig för tidskritiska eller dynamiska simuleringar. Projektet undersöker möjligheter och förutsättningar och har börjat med att, baserad på en HLA-kärna, bygga upp ett försvarsmaktsgemensamt system och miljö för konstruktion, utveckling, dokumentation, exekvering samt verifiering och validering av nätverksbaserade simuleringsmodeller. På så sätt försöker projektet, genom de krav som miljön ställer på den grundläggande infrastrukturen samt jämförelse med andra tillgängliga teknologier, identifiera ev. brister i HLA arkitekturen och bidra till en vidareutveckling av den. Arbetet innebär också undersökning av senaste tekniker inom HLA, som BOM-teknologin och dynamiska FOM:ar.

3 Utveckling av en webbaserad simuleringsmiljö

3.1 Webbanpassad utveckling av webbaserade HLA-federationer

Det här kapitlet presenterar resultatet från vårt arbete med att webbanpassa utvecklingen av HLA-federationer, det vill säga utveckling av webbanpassade federater som sedan exekveras som en federation via en webbaserad miljö.

3.1.1 Metodik

I kapitel 2.2 diskuterades ett antal fördelar som kan vinnas med att inte bara göra simuleringar webbanpassade utan att även webbanpassa utvecklingsprocessen för simuleringar. Det vill säga, en användare skulle kunna utveckla en simulering via ett webbgränssnitt. En metod för att implementera detta är att använda komponentbaserade simuleringar, där delsimuleringar (federater) kopplas ihop till en hel simulering (federation). Ihopkopplingen sker genom ett webbgränssnitt där objekten (federaterna) är kopplade till verkliga simuleringsmodeller i bakgrunden.

Projektet valde att undersöka metoden att via webben utveckla komponentbaserade simuleringar och för att möjliggöra detta valdes arkitekturen HLA som grund⁷. I detta fall måste federaterna förutom att vara HLA-anpassade även vara webbaserade och ha ett webbgränssnitt. Syftet med denna del i projektet var främst att undersöka möjligheterna för att utveckla webbaserade HLA-komponenter och även göra försök till sådan utveckling. En viktig del av arbetet utfördes inom ramen för ett examensarbete. I samarbete utvecklades och genomfördes sex deluppgifter enligt nedan:

- Identifiera lämpliga tekniker för att utveckla webbaserade HLA-federationer.
- Analysera de tekniker som identifierades i punkt 1 och utvärdera vilken som är bäst lämpad för syftet.
- Utveckla testfederater som kan användas för utveckling av webbaserade HLA-federationer.
- Utveckla en webbaserad miljö där flera federater kan kopplas ihop till en federation som sedan exekveras.
- Analysera resultatet av testmiljön.
- Identifiera och undersöka metoder och tekniker för att koppla samman flera federationer till en ny federation – hierarkisk federationsutveckling.

Mer detaljerad beskrivning av examensarbetet och resultaten från detta finns i en examensrapport [Svensson, 2002].

3.1.2 Utveckling av prototyp

För att studera och testa möjligheterna med att göra konstruktionen av HLA-federationer webbaserad utvecklades en prototyp till en sådan utvecklingsmiljö. Först valdes ett antal federater som skulle användas i en testfederation. Sedan studerades och valdes de webbtekniker som var tänkta att utnyttjas och hur dessa skulle implementeras. Parallellt med detta utvecklades webbgränssnittet. Samtliga delar gjordes av projektmedlemmar i nära samarbete med examensarbetaren.

⁷ För information om HLA se kap 2.2 och [DMSO HLA].

3.1.2.1 Testfederater

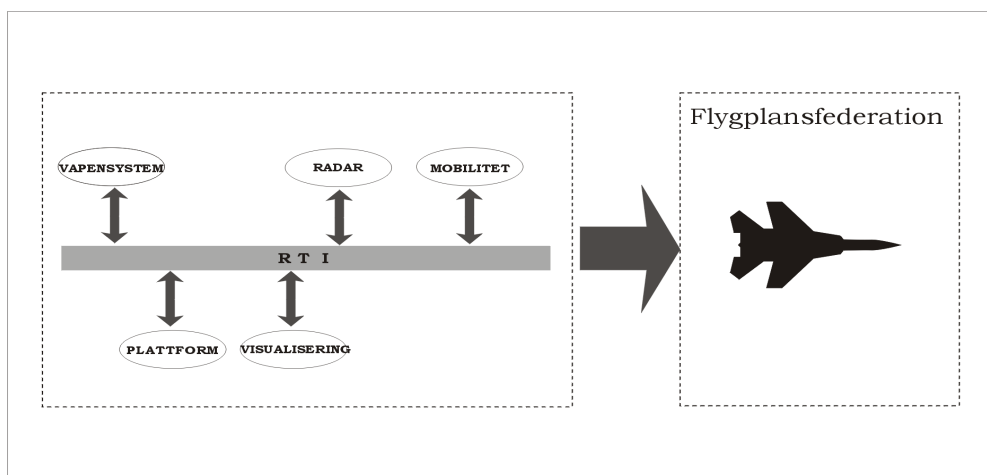
Testfederaterna var inte tänkta att vara komplexa eller särskilt generella. Det var tekniker och metoder som var centrala i arbetet kring den webbaserade simuleringsutvecklingen. Vid implementeringen av federaterna valdes programmeringsspråket Java. Att just Java valdes har många anledningar, men de främsta var att språket till stor del är plattformsoberoende och även utvecklat för att vara lämpligt i just webb- och nätverksbaserade miljöer. Det finns många, som t.ex. Healy [Page, 1999] som anser att Java är standardspråket för Internet-baserade applikationer. Det RTI som valdes för testmiljön, pRTI⁸, är dessutom utvecklat i Java så det kändes naturligt att fortsätta med det.

Det scenario som var tänkt att simuleras var mycket enkelt. Det baserades på att en användare kan välja ut federater i ett användargränssnitt i en webbläsare, se stycke 3.1.2.3. Från webbläsaren kan användaren starta federationen som då exekveras på en server någonstans. I webbfönstret presenteras resultatet av simuleringen.

För att strukturera upp arbetet valdes ett antal testfederater som tillsammans skulle bilda en federation i form av en flygplansfederation, se figur 9 nedan. De federater som valdes för implementeringen var som följer:

- Plattform – Generisk plattform som används som bas för den modell som ska ingå i federationen.
- Radar – Detekterar objekt i omvärlden.
- Mobilitet – Styr federationens rörelse.
- Visualiseringsfederat – Den del som visualiserar federationens rörelse.
- Vapensystem – Generell federat för verkan och vapensystem.

Mer detaljer om federaterna och utförligt hur de fungerar och interagerar med varandra finns beskrivet i examensrapporten [Svensson, 2002].



Figur 9. Testfederationen: Flygplansfederation

3.1.2.2 Lämpliga webbt tekniker

Ett antal webbt tekniker studerades och utvärderades utifrån deras lämplighet i sammanhanget. Eftersom federaterna utvecklades i Java och eftersom språket har just de fördelar som nämndes ovan, var det mest intressant att titta på Java-tekniker. De

⁸ pRTI – "Portable RTI" är utvecklat av Pitch, se www.pitch.se.

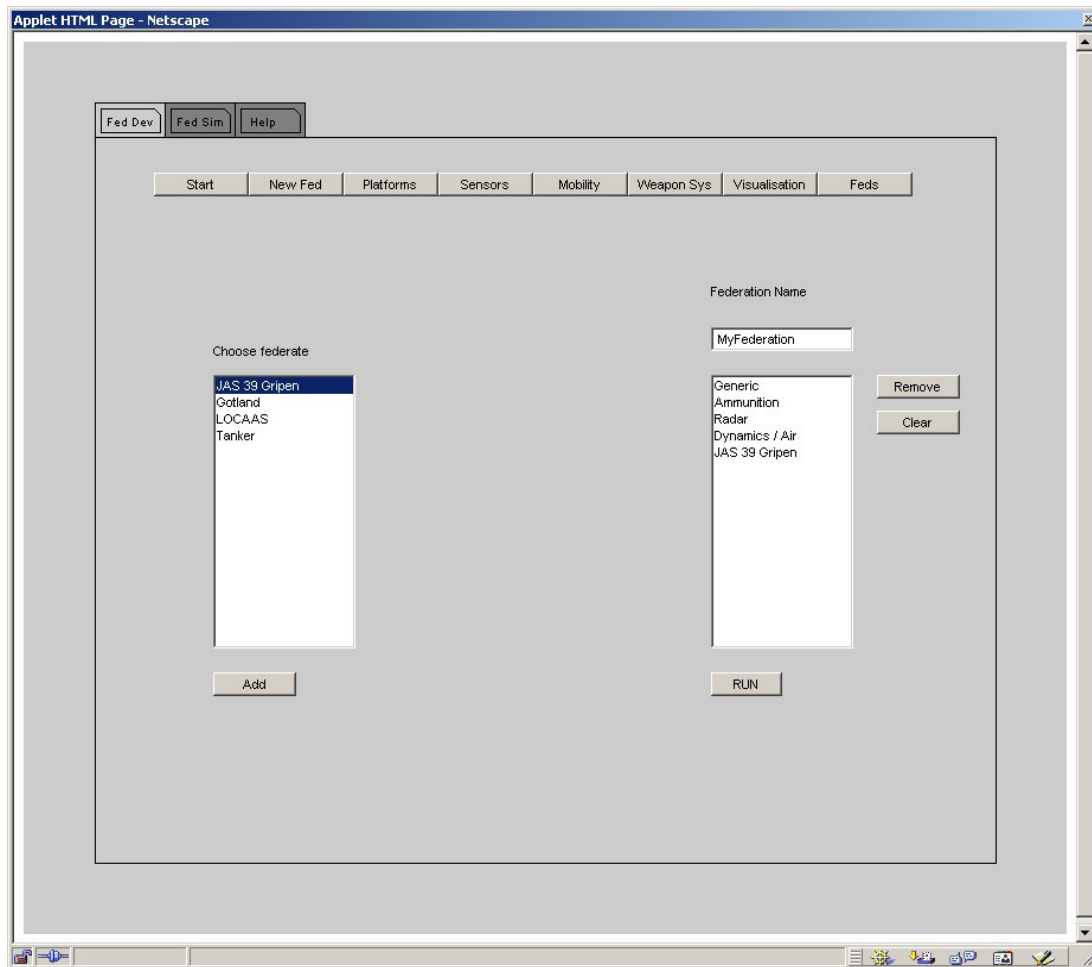
olika tekniker som har följt med Java-språket har de fördelar som även Java-språket har, men också flera andra.

De webbt tekniker som studerades beskrivs utförligt i [Svensson, 2002], och några av dessa (servlets, CGI och dylika) har tidigare presenterats i kapitel 2.3.2. Därmed kommer vi endast att ge en kort beskrivning av den enda återstående här aktuella tekniken JavaBeans. Punkter som beaktades särskilt väl vid val av tekniker var t.ex. plattformsoberoende och att klientens kapacitet inte behövde ha betydelse, utan att så lite som möjligt av klientens resurser skulle användas, d.v.s. klienten skulle vara så tunn som möjligt.

JavaBeans

Med Java-bönor menas här Enterprise JavaBeans (EJB) och inte de ursprungliga JavaBeans, för beskrivning av skillnaden mellan de båda se [Svensson, 2002]. En Java-böna är en komponent som kan pluggas in till en server som en s.k. "server-side" komponent. Det vill säga att den fungerar som en utökning av servern. Java-bönans komponentmodell är dessutom kombinerad med teknologi för objektdistribuering vilket ökar användningsområdena för den.

Egenskaper hos bönan gör att den med fördel skulle kunna användas för att skapa federater. Det vill säga att en böna skulle kunna motsvara en federat. Detta är en mycket intressant idé som diskuteras mer i [Svensson, 2002]. I förstudien och prototypen valdes dock inte Java-bönan som teknik. Den främsta orsaken till detta var att en böna kräver lite mer kodning än klassiska federater som programmeras i Java. De federater som finns tillgängliga idag är dessutom inte gjorda som Java-bönor. Därför valdes som ett första steg Java-bönor bort ur skapandet av prototypen och istället valdes en teknik där Java-federater används tillsammans med servlets. Vidare studier i att använda Java-bönor i detta syfte vore däremot intressanta p.g.a. Java-bönans alla fördelar.



Figur 10. Webbgränssnittet

3.1.2.3 Webbgränssnitt

För att göra en prototyp av en miljö, likt den ovan beskrivna, finns många olika metoder. En struktur för miljön arbetades fram och beskrivs nedan. Utseendet på användargränssnittet i webbfönstret visas i figur 10.

En av komponenterna i prototypen måste vara ett webbgränssnitt där användaren på ett enkelt sätt kan komma åt och exekvera simuleringar. Här är det även tänkt att användaren ska kunna komponera sin egen federation av federater som går att välja ur exempelvis listor. För att enklast åstadkomma detta används en applet. Här väljs de federater som ska exekveras i federationen. Användaren måste även kunna se resultatet av exekveringen i gränssnittet. Användaren kommer åt appleten via en vanlig webbsida. Eventuellt kan detta göras med personlig inloggning för att göra användandet av miljön säkert och för att bistå med personliga profiler.

Federaterna ligger i sin tur på exempelvis en server, eller distribuerade över nätverket. Användaren måste kunna komma åt federaterna via appleten. Detta kan exempelvis göras med hjälp av en servlet. Servleten finns på den webserver som appleten är uppkopplad mot och fungerar som mellanhand mellan gränssnittet och de verkliga federaterna. Det är också servleten som startar exekveringen av federationen. Federaterna måste därför ha standardiserade, plattformsoberoende gränssnitt för att tillsammans kunna kopplas till en federation.

3.1.2.4 Implementering

Här sammanfattas hela implementationen av miljön, de delar som ingår och hur de fungerar. Detta är beskrivet mer i detalj i [Svensson, 2002] och visas schematiskt i figur 11 nedan.

Applet

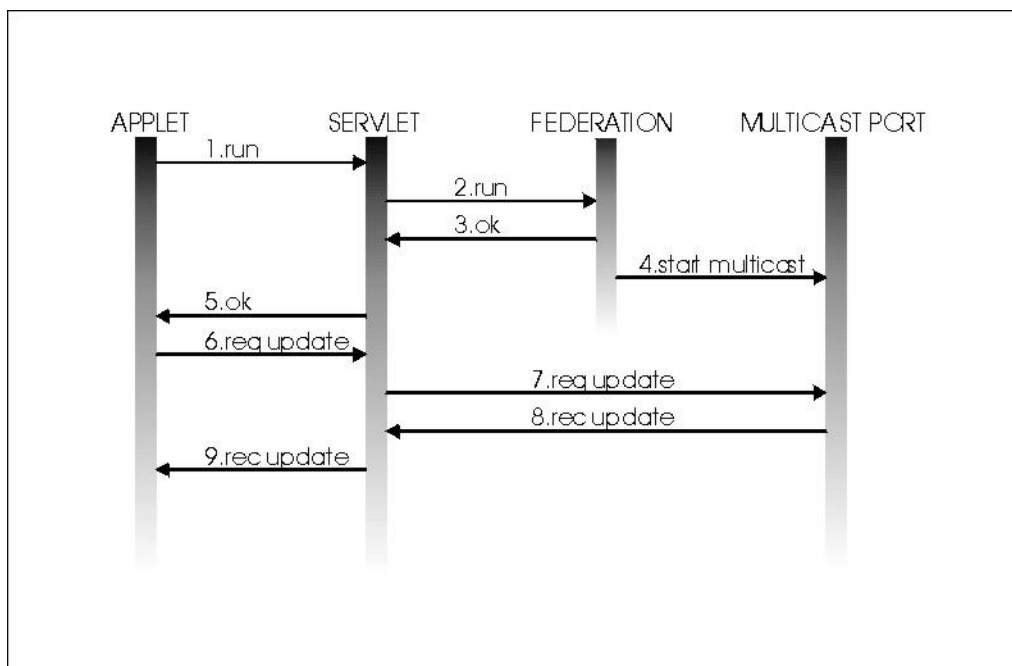
Användargränssnittet implementerades i Java som en applet. En lite äldre version av Java valdes (Java 1.0) för att passa även de webbläsare som ännu inte stödjer de senaste versionerna av Java. I appleten kan användaren välja federater ur menylister som grupperas efter olika federattyper, såsom plattformar och vapensystem. När användaren trycker på knappen "Run" skickas en lista med valda federater till en servlet som tar emot den och i sin tur uppmanar federaterna, som ligger på servern, att starta som en federation.

Servlet

Servleten tar emot information om vilka federater som användaren vill ska exekveras som en federation och vad federationen ska heta. Ett kommando skickas som startar upp den tänkta federationen. När federationen exekveras uppdateras servleten kontinuerligt om federationens agerande (i det här fallet flygplanets rörelse, position). Servleten vidareförmedlar informationen till appleten som uppdaterar visualiseringen av scenariot (simuleringen). När exekveringen är färdig upphör kommunikationen och användaren kan börja om.

Federater

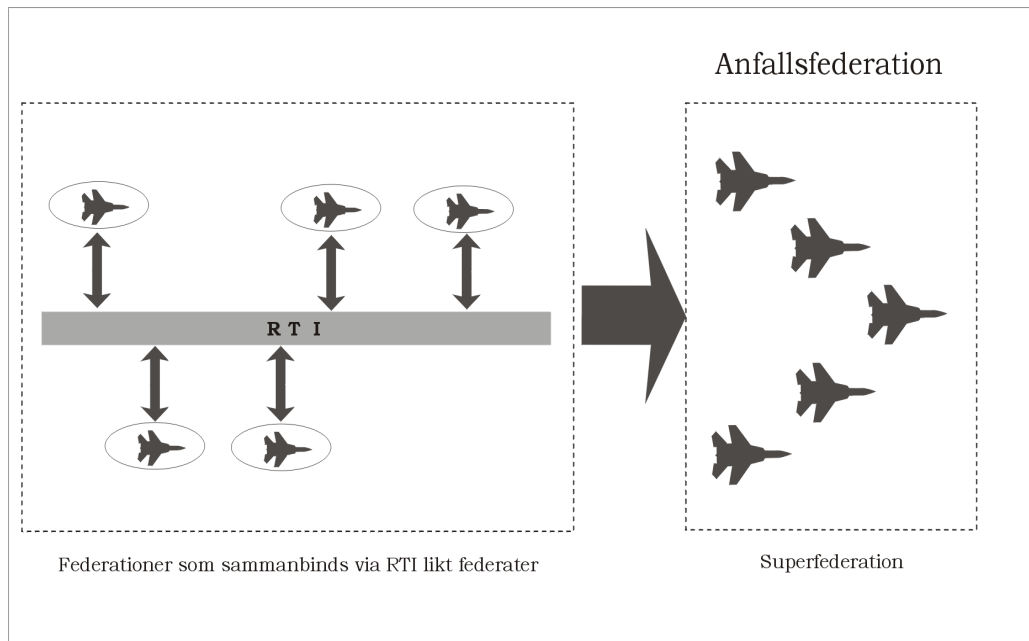
Federaterna är skrivna i Java och är HLA-standardiserade. För att koppla ihop federationen användes pRTI från Pitch. Federaterna är lagrade på samma server som servleten och deras funktioner är beskrivna ovan i stycke 3.1.3.1. För mer detaljerad information om federaterna och kommunikationen med RTI och servlet hänvisas till [Svensson, 2002].



Figur 11. Den webbaserade miljön

3.1.3 Hierarkisk federationsutveckling

I projektet studerades även möjligheter kring att bygga HLA-federationer på ett inte bara komponentbaserat utan även hierarkiskt sätt. Tanken är att det ska gå att utveckla hierarkiska federationer, d.v.s. federationer som består av federater som i sig är federationer, se figur 12 för illustrativ beskrivning. Ett flygplan består exempelvis av många olika komponenter som t.ex. en sensor, en plattform mm. Det kan även tänkas att flygplanet ingår i en simulering där det interagerar med andra federater, som flygplan, luftvärnssimuleringar, etc. I projektet gjordes försök till att implementera denna funktion och dessa presenteras kort nedan och mer genomförligt i [Svensson, 2002].



Figur 12. Illustrativ bild av hierarkisk federation

3.1.3.1 Metoder och tekniker

HLA baseras strikt på att all information som kommuniceras går igenom RTI. För att få federationer att agera som federater i en hierarkisk federation behövs därför någon slags metod, en brygga, mellan de hierarkiska federaterna. Bryggan måste bestå av i alla fall en extra federat som sköter kommunikationen mellan delfederationerna i den hierarkiska federationen. Denna federat kallas här "kommunikationsfederat", eller förkortat kom-federat (FP enligt [Liu et al., 2002]), och måste för sin funktion vara medlem i både de lokala delfederationerna och den hierarkiska federationen. För att åstadkomma detta studerades två alternativ. Det första alternativet baseras på en centraliserad kom-federat. Denna hanterar all kommunikation mellan de olika federaterna (delfederationerna) som är kopplade till den hierarkiska federationen. Det andra alternativet som studerades innebär istället att varje delfederation har en kommunikationsfederat var, vilket betyder att kom-federaterna är distribuerade (distribuerad FP, enligt [Cramp & Oudshoorn, 2002]).

Centraliserad kommunikationsfederat

En centraliserad kom-federat behöver en RTI-ambassadör för varje delfederation som är med i den hierarkiska federationen. Detta betyder att kom-federaten är komplex, särskilt då antalet delfederationer ökar.

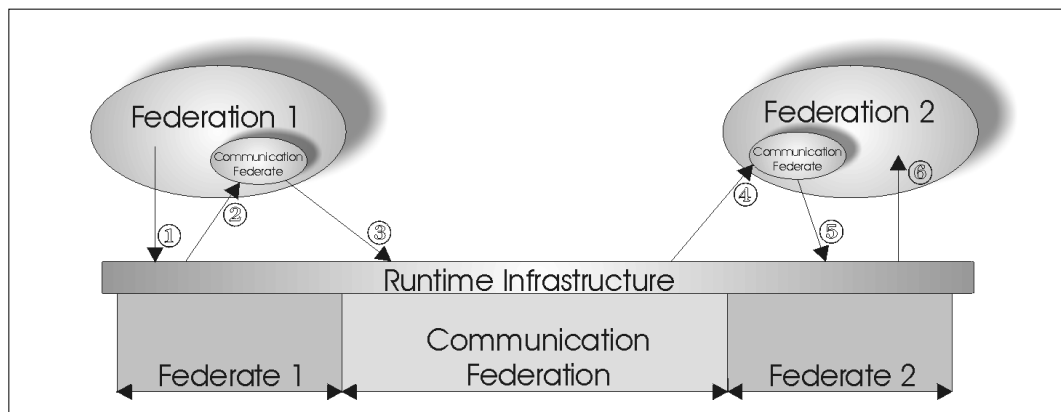
Distribuerad kommunikationsfederat

Det distribuerade tillvägagångssättet ökar visserligen det totala antalet federater som är med i simuleringen, men det förenklar implementeringen för varje kom-federat eftersom de inte skiljer sig så mycket från simulering till simulering oavsett hur många delfederationer som deltar. Detta är också orsaken till att denna teknik valdes för testimplementering.

3.1.3.2 Implementering

Likt implementeringen av den webbaserade testfederationen så valdes inte en komplex federation för att testa idén med utveckling av hierarkiska federationer. Det viktiga här var att implementera och undersöka minst en av de alternativa metoder som studerats. Den teknik som valdes baserades på idén om distribuerade kommunikationsfederater, där varje delfederation har sin egen kom-federat. Se figur 13 för schematisk beskrivning.

En kommunikationsfederat implementerades inom ramen för examensarbetet. Kom-federaten testades och kunde ansluta sig till både sin delfederation och till den hierarkiska federationen. Dessutom kunde delfederationen publicera och prenumerera på data inom den hierarkiska federationen på det sätt som det var tänkt.



Figur 13. Den webbaserade miljön

3.2 Webbanpassning av simuleringsmodeller

3.2.1 Val av simuleringsmodeller

Två simuleringsmodeller, Tavast2 och Arteva, valdes ut för att representera två olika typer av applikationer. Tavast2 med ett grafiskt användargränssnitt valdes för att belysa hur ett program med relativt komplicerad grafik kan få en motsvarighet på nätet. Arteva är ett textbaserat program som exekveras från en kommandotolk (DOS-fönster) och som därmed skulle kunna göras mer lättanvänd med enkel webb-teknik.

3.2.1.1 Tavast2

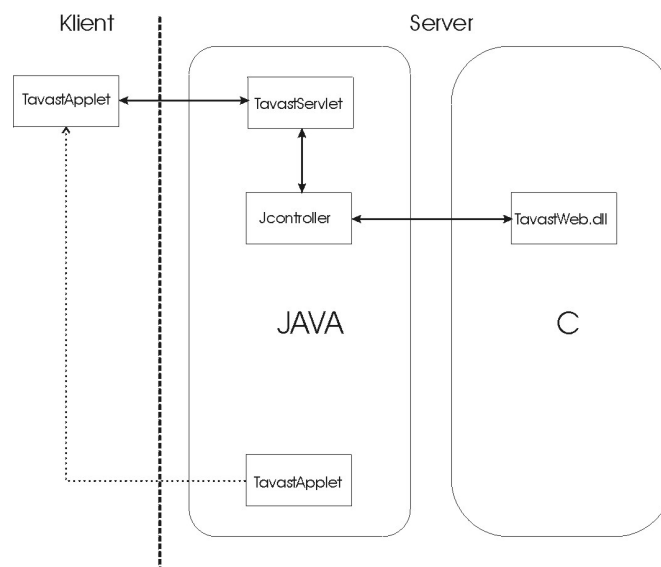
Tavast2 står för TAtisk Verkan Av STörning och är ett verktyg för simulering av störverkan mot kommunikationsutrustning. Radiomottagare och sändare placeras ut i en terräng som representeras av en kartbakgrund. Parametrar som uteffekt, frekvens och antennhöjd kan alterneras för de enskilda radioapparaterna. Vidare placeras störsändare ut i samma terräng och kommunikation mellan sändare och mottagare simuleras. Kommunikationsmöjligheterna redovisas för de enskilda radioapparaterna i form av siffror och grafik (färgsättning av radioapparater) [Tavast2, 1999].

3.2.1.2 Arteva

Arteva simulerar verkan av artilleri mot sammansatta mål, bestående av ett godtyckligt antal soldater, stridsvagnar och pansarbandvagnar. En simulering genomförs genom konstruktion av en textfil med ett antal modellspecifika parametrar, samt därefter exekvering från en kommandotolk. I parameterfilen specificeras bland annat måltyp, ammunitionstyp, terrängtyp och väderförhållanden. Arteva-körningen resulterar i en textfil där verkansvärden för målen redovisas [Arteva, 2001].

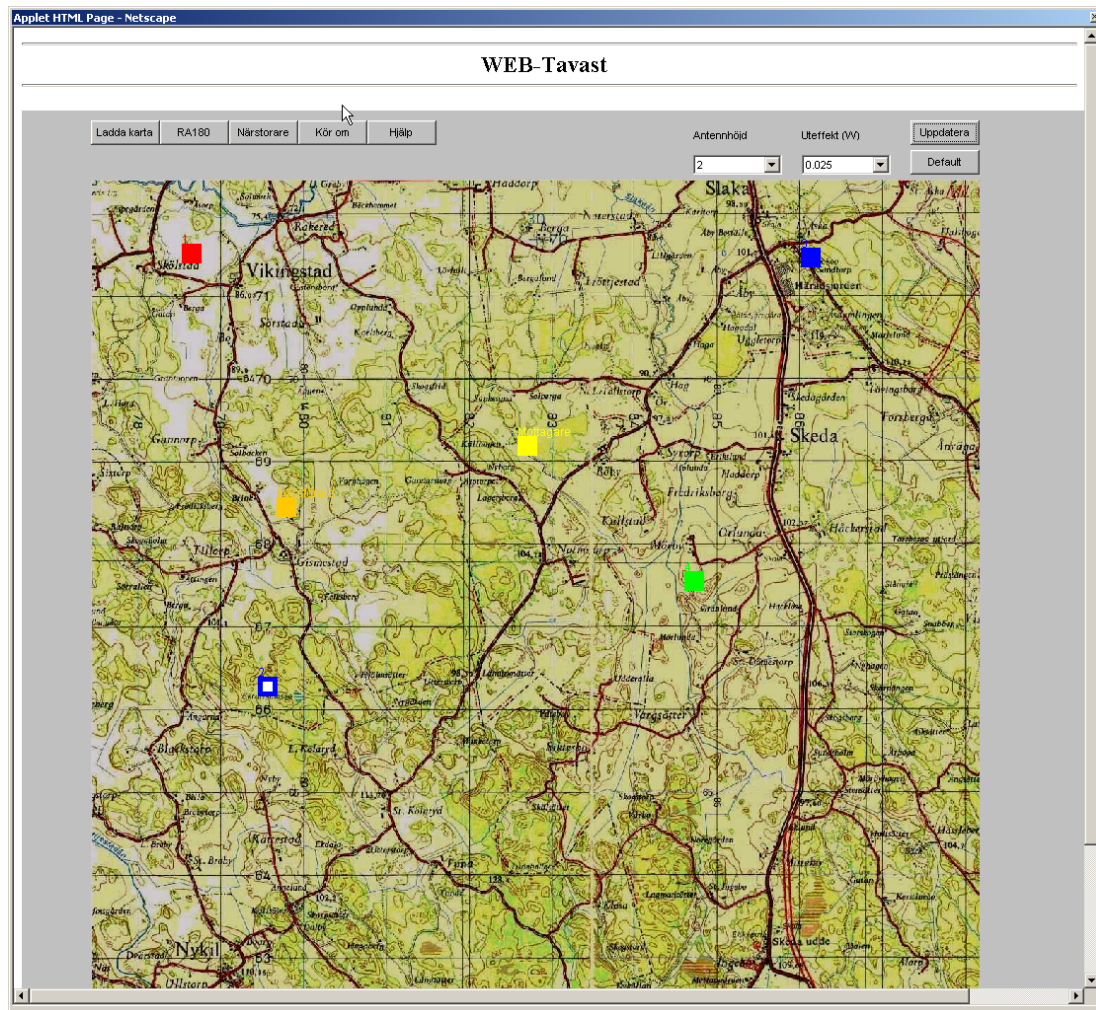
3.2.2 Implementering av Webb-Tavast

Figur 14 redovisar uppbyggnaden av Webb-Tavast, vilket är den webbanpassade versionen av Tavast2 som utvecklats inom projektet, med ingående objekt och inbördes kommunikation mellan dessa. Det grafiska användargränssnittet i Tavast2 representeras i Webb-Tavast av en applet, TavastApplet. En applet är en mindre applikation som nås via en webbserver, transporteras över Internet och som automatiskt installeras och körs i en webbläsare. Hos klienten har appleten begränsad tillgång till datorns resurser, för att på så sätt erhålla en tillfredställande säkerhet [Schildt, 2001].



Figur 14. Uppbyggnaden av Webb-Tavast. Pilarna visar hur kommunikationen sker mellan de ingående objekten.

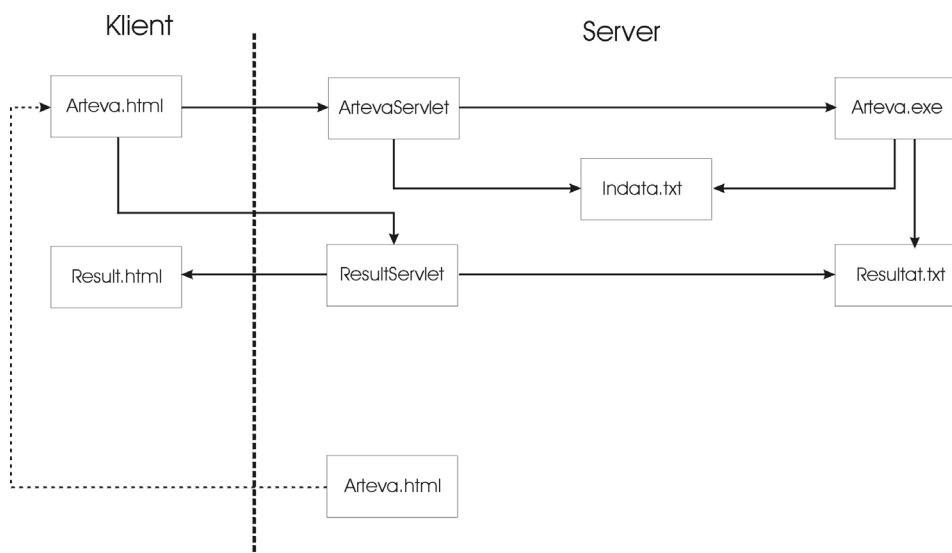
En körning av Webb-Tavast går till på följande sätt: klienten begär att få en kopia av TavastApplet som sedan installeras och körs i webbläsaren. TavastApplet initierar kontakt med TavastServlet som skapar en instans av JController. JController har i sin tur kontakt med TavastWeb.dll som utgör själva simuleringsmodellen. Varje förändring som användaren gör i gränssnittet (TavastApplet), lägger till en radio, tar bort en radio osv., resulterar i en kontakt med TavastServlet som via JController kör simuleringsmodellen och erhåller det nya "tillståndet" (dvs. om radiokontakt är möjlig). TavastServlet vidarebefordrar resultatet till TavastApplet som uppdaterar grafiken. Figur 15 visar användargränssnittet (TavastApplet) till Webb-Tavast.



Figur 15. Användargränssnitt till Webb-Tavast (TavastApplet) Den gula ikonerna representerar mottagaren, störaren representeras av den orangefärgade ikonerna. De övriga ikonerna representerar sändare där färgen (grön, blå eller röd) indikerar signal-störförhållandet till mottagaren.

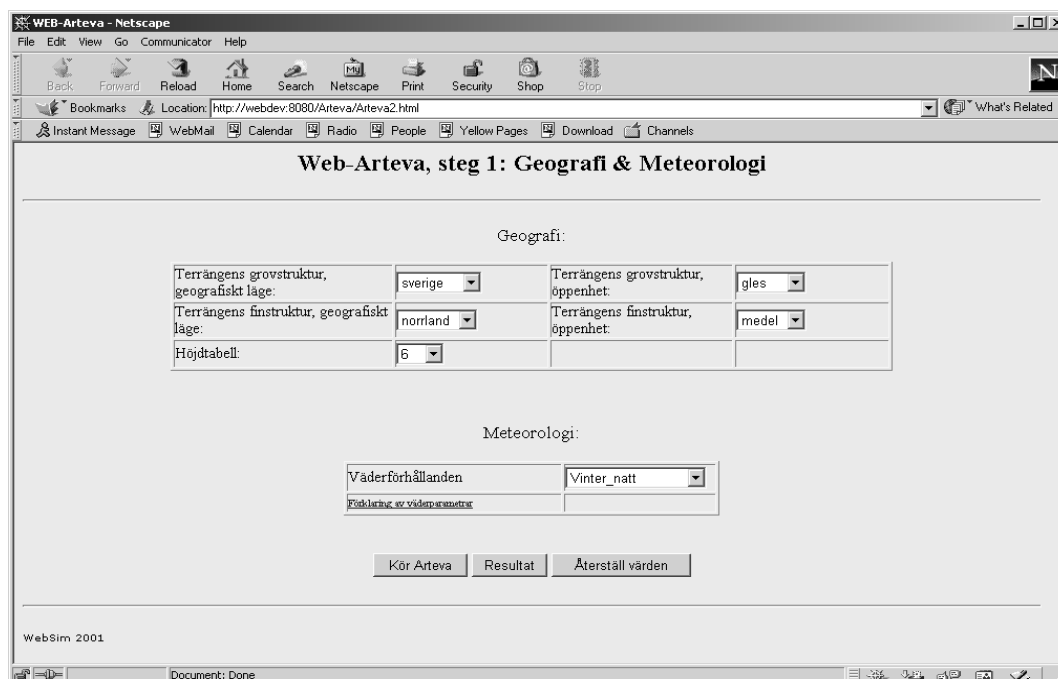
3.2.3 Implementering av Webb-Arteva

Figur 16 redovisar uppbyggnaden av Webb-Arteva, vilket är den webbanpassade versionen av Arteva som utvecklats inom projektet, med ingående objekt och inbördes kommunikation mellan dessa. För att exekvera Arteva krävs att en parameterfil som specificerar mål, ammunition, väderförhållanden osv. finns tillgänglig. Arbetet med att göra Arteva tillgängligt via Internet innebar i praktiken att ett webbaserat användargränssnitt, med uppgift att skapa denna fil, byggdes. Inga förändringar av Arteva krävdes.



Figur 16. Uppbyggnaden av Webb-Arteva. Pilarna visar hur kommunikationen sker mellan de ingående objekten.

Användargränssnittet utgörs av en HTML-sida, Arteva.html, där parametrar specificeras via rullgardinsmenyer, se figur 17. När konfigurationen av simuleringen är klar skickas informationen till webbservern, där en servlet tar emot resultatet, ArtevaServlet. ArtevaServlet skapar en textfil (indata.txt), innehållande informationen som skickades från HTML-sidan, och exekverar därefter Arteva med den skapade filen som input. Körningen av Arteva genererar en textfil som utdata, resultat.txt. Innehållet i denna redovisas på begäran av användaren genom en knapptryckning på huvudsidan. På denna begäran öppnas en ny webbläsare, varvid en servlet, ResultServlet, öppnar och skickar informationen från utdatafilen till denna.



Figur 17. Användargränssnittet till Webb-Arteva (Arteva.html) där geografiska och meteorologiska parametrar specificeras i rullgardinsmenyer.

3.3 Distribuerad exekveringsmiljö

3.3.1 Design av miljön

Den distribuerade exekveringsmiljön, här benämnd FOI_EXE, är en delmängd av den simuleringsmiljö som utvecklas inom ramen för NätSim-projektet. Målet med denna simuleringsmiljö är att stödja nätverksbaserad, kollaborativ modellutveckling och exekvering. Exekveringen är för användaren av simuleringsmiljön en transparent process, dvs. exekveringsmiljön är dold för användaren [NätSim, 2002].

De grundläggande förutsättningarna för FOI_EXE är att den skall kunna utnyttja beräkningsresurser i ett nätverk av arbetsstationer ("Network of Workstations", NOW), använda ett distribuerat fildelningssystem (modellbibliotek), bygga på ett decentraliserat koncept (P2P-system) och slutligen skall de simuleringar som exekveras vara HLA-baserade. Nätverket är i detta fall begränsat till ett lokalt nätverk (LAN). För de beräkningsresurser (noder) som reserveras för uppgifter av FOI_EXE gäller en icke-statisk allokering, dvs. tillgängligheten på resurser under en simulering är dynamisk. Anslutning av noder till exekveringsmiljön sker på "Ad-Hoc"-basis, dvs. nätverkstopologin kan förväntas att förändras snabbt och oförutsägbart.

Nyttjande av NOW för distribuerad simulering har under senare tid ökat i popularitet. Detta kan bland annat hänföras till det faktum att ett NOW ofta finns tillgängligt inom en organisation, till skillnad från stöd för alternativa metoder för distribuerad simulering, dvs. specialutvecklad hårdvara för parallell exekvering [Luthi & Großmann, 2001]. I [Luthi & Großmann, 2001] presenteras ett system benämnt Resource Sharing System (RSS), som till stor del uppfyller de krav som specificerats för FOI_EXE ovan. En avgörande skillnad mellan systemen är att FOI_EXE inte bygger på centraliserad kontroll, vilket RSS gör. RSS har en central komponent (Manager) som koordinerar systemet samt en ftp-server som lagrar modeller (modellbibliotek). I FOI_EXE kan varje enskild nod i nätverket anta rollen som koordinator för simuleringen och modellbiblioteket är en distribuerad filarea som delas av alla ingående noder.

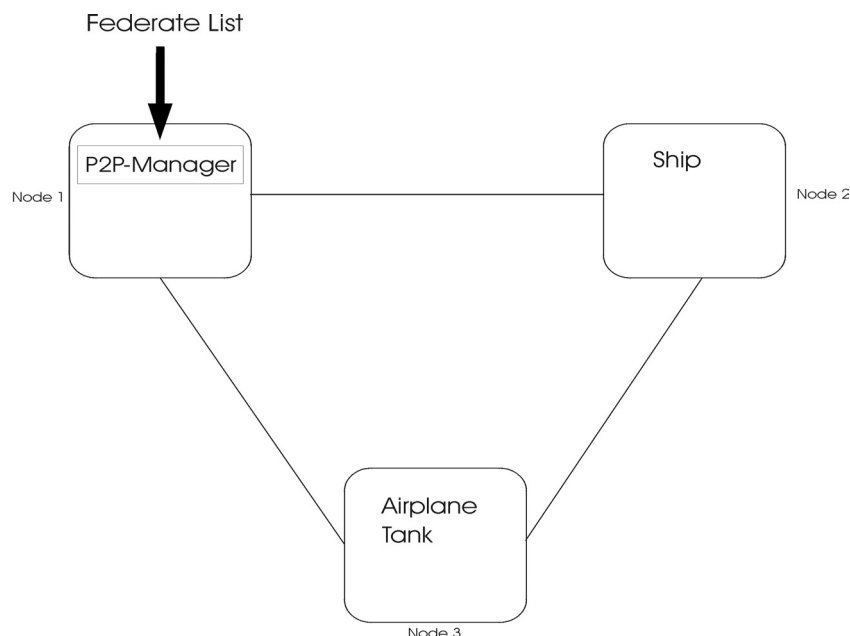
För utnyttjande av beräkningsresurser inom det lokala nätverket förutsätts ett aktivt deltagande från datoranvändare. Konceptet liknar det tankesätt som bland annat [SETI@home] har anammat, dvs. att använda beräkningsresurser hos datorer som för tillfället inte utnyttjas. Med aktivt deltagande avses att användare laddar ner och startar en applikation som under vissa förutsättningar gör processorkraft tillgänglig för någon typ av uppgift. I FOI_EXE kan applikationen ha två huvudsakliga uppgifter; vara tillgänglig för exekvering av HLA-federater och/eller ingå i ett distribuerat filsystem. Graden av inblandning specificeras av användaren i applikationens gränssnitt.

FOI_EXE har två distinkt skilda delar, nämligen; en infrastruktur för simulering (HLA) och en miljö för administration inför och under en simulering. Den senare benämns JXTA-miljön i följande text. Centralt för JXTA-miljön är det faktum att alla noder som ansluter till systemet har en gemensam funktionalitet. Vid exekveringen av en federation antar en av noderna rollen som administratör för exekveringen, benämnd P2P-Manager i följande text. Under denna fas antar systemet en konfiguration som mer liknar en traditionell client-server-topologi. I syfte att redovisa den funktionalitet som FOI_EXE innehåller, redovisas översiktligt de steg som en exekvering omfattar.

Steg 1 (Figur 18)

Figur 18 visar hur en JXTA-miljö kan vara konfigurerad inför en exekvering. I detta fall är tre noder, på tre skilda arbetsstationer, anslutna till JXTA-miljön. När en användare av simuleringsmiljön har definierat den federation som skall exekveras, skickas en lista med ingående federater till JXTA-miljön, eller snarare JXTA-

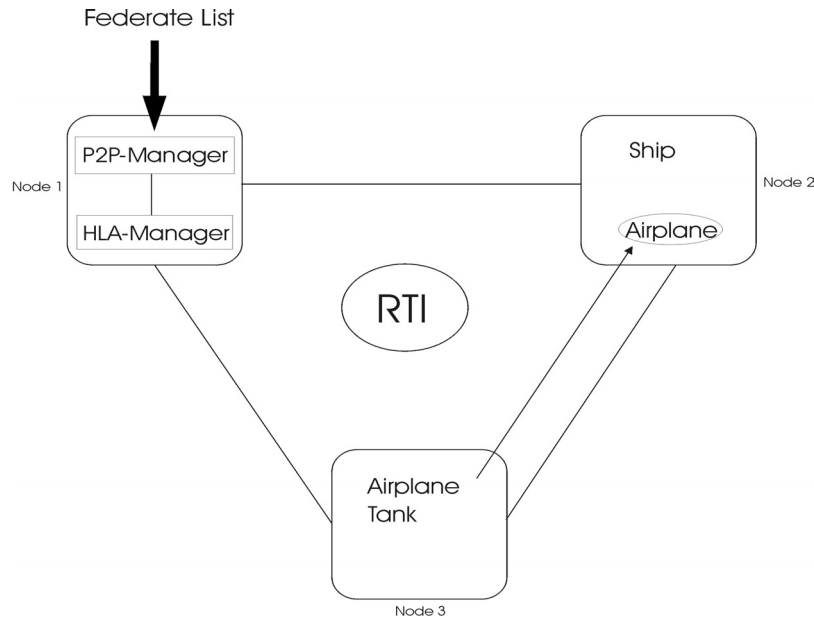
exekveringsmiljön. Som nämnts tidigare kan i detta skede godtycklig nod inom JXTA-miljön anta rollen som koordinator för exekveringen. I detta exempel antar nod 1 rollen som administratör för simuleringen genom dess P2P-Manager. De noder som momentant utgör en JXTA-miljö bygger upp ett distribuerat filsystem där federater finns lagrade. FOI_EXE förutsätter att de filer som anges i "Federate list" finns tillgängliga från filsystemet. Detta är ett rimligt antagande eftersom användare av simuleringsmiljön utgår från de federater som finns lagrade i filsystemet vid skapande av federationer. Eftersom JXTA-miljön är ett dynamiskt system är det möjligt att resurser (federater) försvinner från filsystemet mellan specifikation och exekvering av en federation. Detta är dock förhållandevis enkelt att hantera genom att införa mer permanenta noder i miljön, samt genom att replikera federater när dessa laddas upp, dvs. låta mer än en nod lagra en federat. I exemplet som framgår av bilden finns tre federater i fildelningsystemet; "Airplane", "Tank" och "Ship". I nästa exempel utgörs federationen som skall exekveras av federaterna "Airplane" och "Tank".



Figur 18. Exekveringsmiljöns konfiguration inför en simulering. I detta exempel ingår tre noder i JXTA-nätverket inom vilket tre federater delas (Airplane, Tank och Ship).

Steg 2 (figur 19)

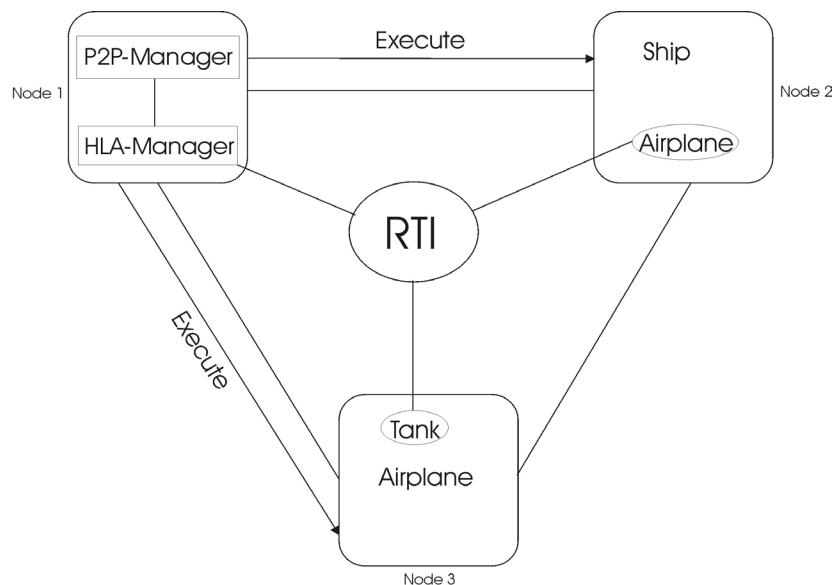
I nästa steg söker P2P-Managern efter beräkningsresurser inom JXTA-miljön. Som nämnts tidigare kan noder anta två olika roller; deltagare i det distribuerade filsystemet och/eller beräkningsresurs (exekvering av federater). I detta exempel står både nod 2 och 3 till förfogande för exekvering. Utifrån federationsspecifikationen i "Federate List" och antalet lediga beräkningsresurser inom JXTA-miljön, fördelar P2P-Managern arbetet mellan noderna. I exemplet exekveras Airplane-federaten på nod 2 och Tank-federaten på nod 3. I praktiken innebär detta beslut att nod 2 laddar ner Airplane-federaten från det gemensamma filsystemet. Under detta skeende startar P2P-Managern även upp en HLA-Manager som fungerar som gränssnitt mellan federationen och P2P-Managern (HLA-Managern är medlem av federationen). Vidare startar även P2P-Managern ett RTI på någon nod inom JXTA-miljön. För att underlätta förståelsen av systemet illustreras RTI i figur 19 som fristående från noderna i JXTA-miljön. I själva verket startas RTI upp på en av de ingående JXTA-noderna.



Figur 19. P2P-Managern söker efter lediga beräkningsresurser inom JXTA-nätverket och startar en HLA-Manager vars uppgift är att administrera en HLA-federation.

Steg 3 (figur 20)

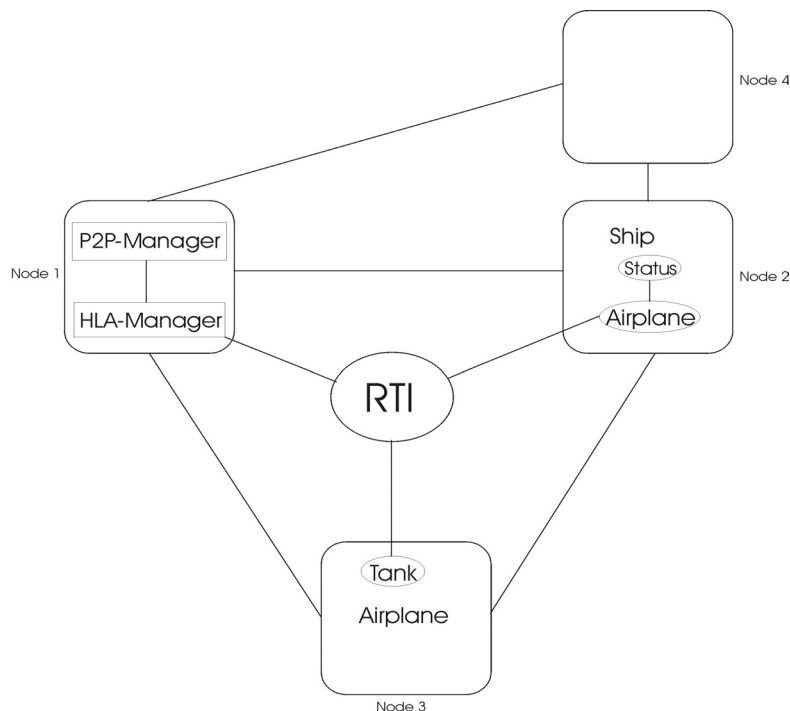
När federaterna har distribuerats till noderna och ett RTI finns tillgängligt inom nätverket, ansluter HLA-Managern till RTI och skapar en ny federationsexekvering. P2P-Managern skickar därpå ett exekveringskommando till de noder som skall delta i exekveringen, i detta fall till nod 2 och 3. När noderna nås av detta kommando exekverar nod 2 Airplane-federaten och nod 3 Tank-federaten som båda ansluter till federationen. När initiering av federationen är avklarad är P2P-Managern fri att starta simuleringen (avancera federationens tid) med hjälp av HLA-Managern.



Figur 20. När federater i den federation som skall exekveras har distribuerats ut bland noderna i JXTA-nätverket skickar P2P-Managern ett exekveringskommando, varpå federaterna startas upp och ansluter till RTI.

Steg 4 (figur 21)

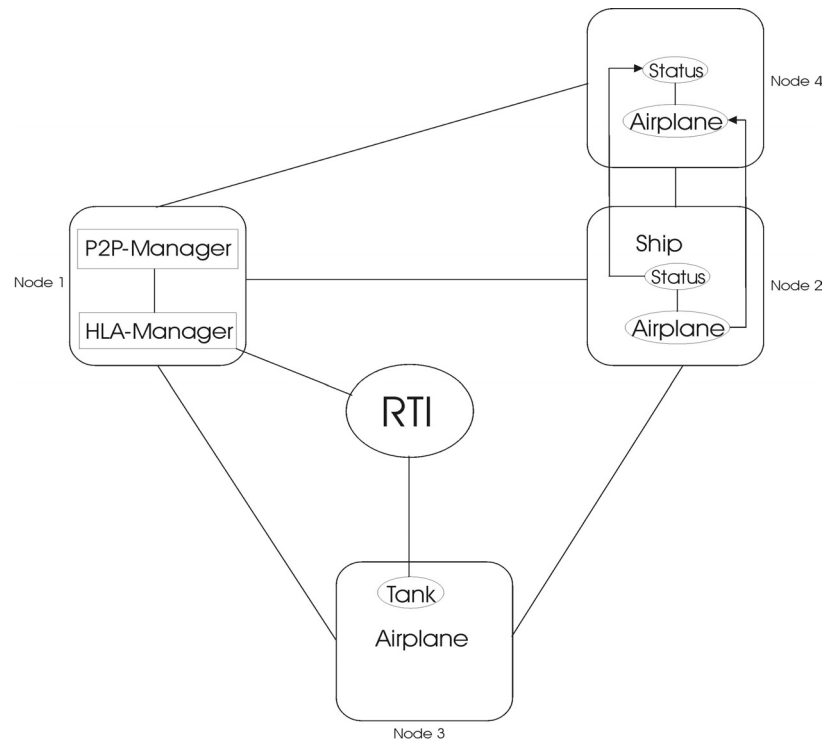
Anslutning och bortfall av noder inom JXTA-miljön sker på "Ad-Hoc"-basis. Om en datoranvändare under ett exekveringsförlopp beslutar sig för att inte tillgängliggöra sina datorresurser längre, underrättas P2P-Managern om detta. Detta får till följd att P2P-Managern via HLA-Managern "fryser" simuleringen (slutar att avancera federationens tid), samt meddelar den federat som exekveras på aktuell nod att spara aktuell simuleringsdata. I figur 21 benämns sparad simuleringsdata "Status". När federaten har skapat en status-fil laddas denna upp till det gemensamma filsystemet. I detta exempel har under exekveringens förlopp ytterligare en nod anslutit till JXTA-miljön, nod 4 i figur 21. Denna nod är inte enbart del av det distribuerade fildelningssystemet, utan har även tillgängliggjort beräkningsresurser.



Figur 21. Allteftersom federationen exekveras kan noder ansluta och kopplas från JXTA-nätverket på "Ad-Hoc"-basis.

Steg 5 (figur 22)

När federaten har skapat statusfilen går den ur den inaktiva federationen och federatens process avslutas. Från det distribuerade fildelningssystemet laddar nod 4 ner federaten som skall flyttas (Airplane), samt status-filen för federaten (sparad simuleringsdata).

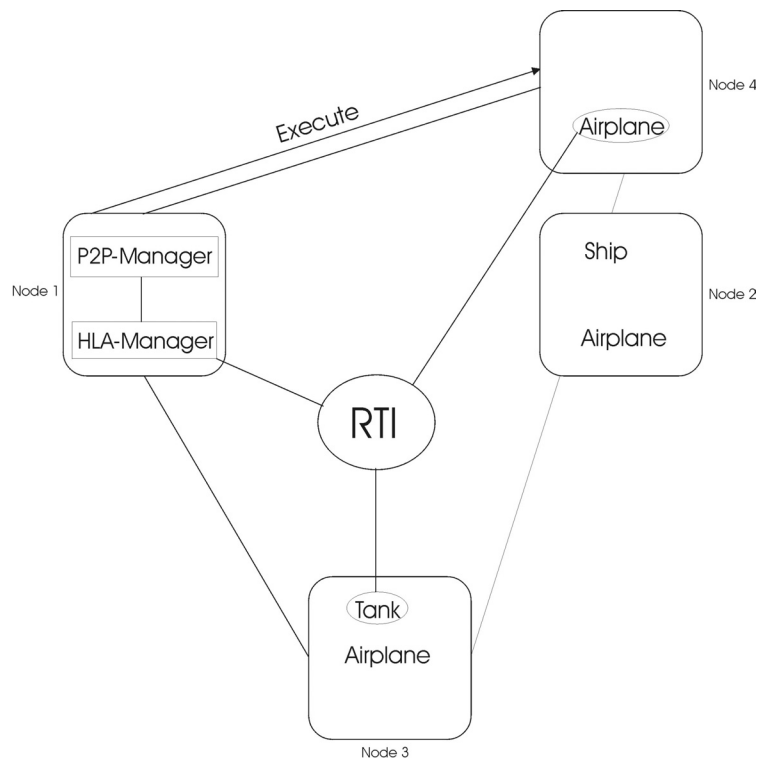


Figur 22. Den nod som övertar ansvaret för att exekvera "Airplane" laddar ner federaten plus den aktuella statusfilen från det gemensamma filsystemet.

Steg 6 (figur 23)

När de aktuella filerna har laddats ned från filsystemet skickar P2P-Managern ett exekveringskommando till nod 4. Airplane-federaten initialiseras med informationen från status-filen och ansluter till den inaktiva federationen. Efter ytterligare initialisering (information från RTI) underrättas HLA-Managern av den nyligen anslutna federaten (via RTI) att den är redo för fortsatt simulering. P2P-Managern är i och med detta åter fri att avancera federationens tid med hjälp av HLA-Managern.

Steg 4 till 6 kan ske godtyckligt antal gånger under ett exekveringsförlopp varefter nätverkstopologin varierar. När federationen har nått sitt slutgiltiga mål, går deltagande federater ur federationen, federationsexekveringen tas bort från RTI och RTI stoppas. Eventuella "lokala" kopior av federater som skapades under exekveringen tas bort från systemet. Enbart de kopior som aktivt laddas upp till det gemensamma filsystemet av användare tillåts att existera permanent. Annars är det omöjligt att administrera innehållet i fildelningssystemet, exempelvis uppdatering av redan existerande federater.



Figur 23. När federaten har laddats ner skickas ett exekveringskommando från P2P-Managern till nod 4. Federaten (Airplane) startas då upp och ansluter till RTI.

3.3.2 Implementering av exekveringsmiljön

3.3.2.1 JXTA

En central del av JXTA-miljön utgörs av Content Management Systems (CMS), som är ett ramverk för delning av filer mellan JXTA-noder. CMS medför att noder kan dela ut filer (göra filer tillgängliga för andra noder), söka efter utdelade filer, samt ladda ner filer från andra noder [Brookshier et al., 2002]. CMS har för närvarande begränsningar beträffande sökning efter resurser och optimering av filöverföringen, men infördes ändå som del av JXTA-miljön för att minimera komplexiteten i utvecklingen av FOI_EXE. De tjänster som CMS erbjuder är tillräckliga för att uppfylla de grundläggande krav som ställdes på exekveringsmiljön. Följande funktionalitet uppfylls med hjälp av CMS:

- Federater kan delas ut från alla noder i JXTA-miljön.
- Inför exekveringen av en federation skapas lokala kopior av federaterna hos de noder som skall delta.
- När en federat flyttas, under en pågående exekvering, sparas simuleringsdata i en fil. Denna fil överförs mellan noder med hjälp av CMS.

Utöver den funktionalitet som CMS erbjuder implementerades tjänster för att lokalisera resurser i ett JXTA-nätverk. Inom JXTA-miljön finns två olika typer av resurser definierade; Computing Resource (CR) och Manager Resource (MR). En CR exekverar federater medan en MR hanterar exekveringen av en federation. Inför exekveringen av en federation skickas en allmän förfrågan ut till anslutna noder angående tillgång på resurser. Utifrån svaret på denna förfrågan allokeras noder till att

utföra en specifik uppgift, MR eller CR. En av noderna bli MR och benämns i följande text P2P-Manager. Efter att resurser har allokerats inför en exekvering sker all slags kommunikation via JXTA-pipes, se avsnitt 2.1.2 för en översiktlig beskrivning av pipes. I [Sparf, 2003] presenteras JXTA-miljön mer utförligt, med en detaljerad beskrivning av implementeringen, JXTA-teknologin i allmänhet etc.

3.3.2.2 HLA

De speciella förutsättningar som JXTA-miljön medför, framförallt vad gäller föränderlig nätverkstopologi, men även det faktum att federater exekveras på arbetsstationer med varierande prestanda, innebär att två anpassade HLA-komponenter utvecklades; en federationsadministratör och en generisk HLA-federat. Dessa komponenter benämns HLA-Manager respektive GenericHLAFederate i följande text.

3.3.2.2.1 HLA-Manager

En HLA-Manager är federat i en federation som exekveras. Inför varje exekveringsförfrågan skapas en unik HLA-Manager som administrerar simuleringen utifrån ett antal tjänster som blir tillgängliga för P2P-Managern (JXTA-Miljön). Dessa tjänster kan indelas i följande huvudkategorier:

- Federationsinitiering
- Synkronisering
- Tidshantering
- Avbrottsshantering

3.3.2.2.2 Federationsinitiering

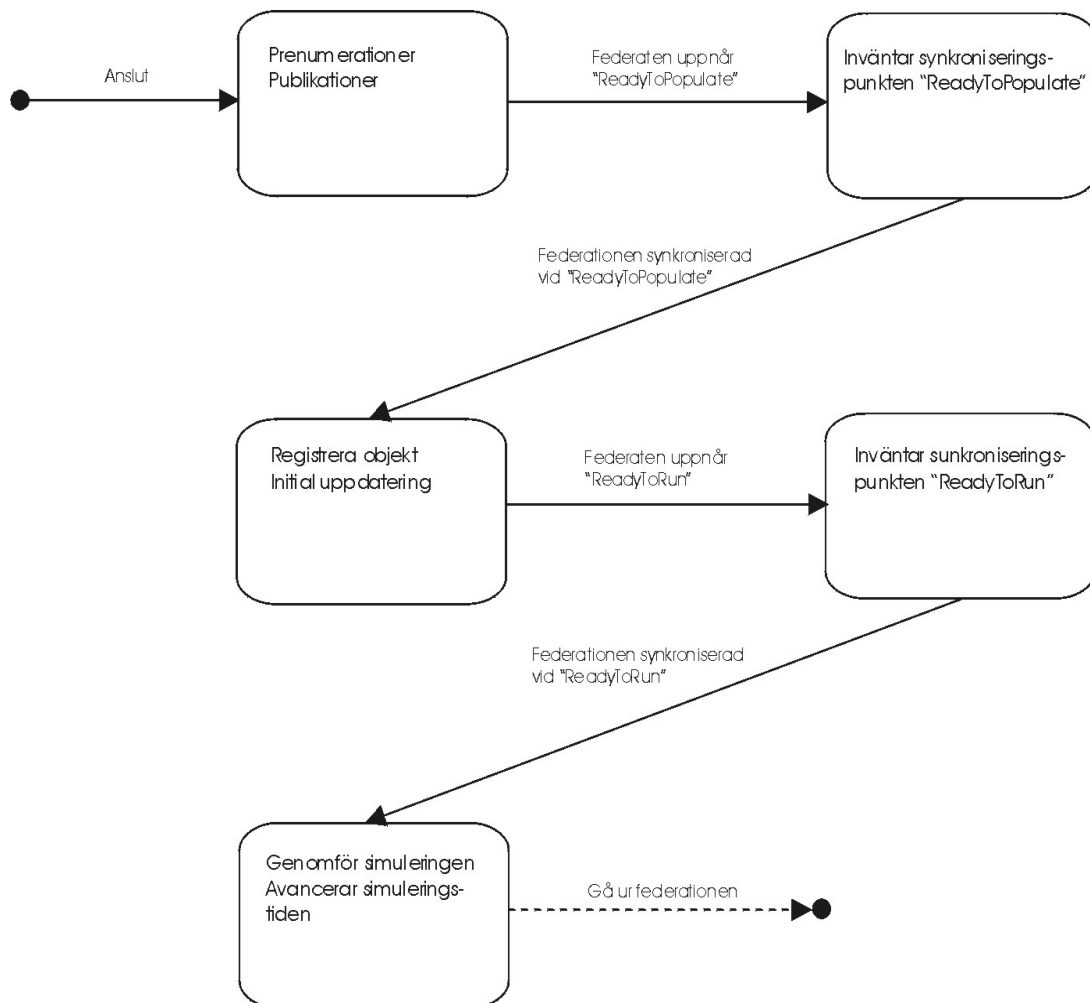
Federationsinitiering innebär att HLA-Managern ansluter till det RTI som anges av P2PManagern och skapar en federation med ett unikt namn. I denna process bifogas en FED-fil som representerar den federation som skall exekveras, samt antalet federater som förväntas ansluta till federationen. Information om antalet federater som ingår i en federation är avgörande för federationssynkroniseringen.

Vid distribuerad simulering är det viktigt att åstadkomma en kontrollerad och synkroniserad start av federationen. Det går inte på förhand att explicit specificera när olika federater väntas ansluta till federationen. Detta faktum är av stor betydelse för FOI_EXE, där det är omöjligt att på förhand känna till vilka resurser som är tillgängliga vid en viss tidpunkt. För att uppnå en koordinerad initiering används synkroniseringspunkter. Detta betyder att initieringen inte tillåts passera specifika punkter förrän alla federater har uppnått samma tillstånd.

Det finns generella rekommendationer, vilka beskriver i vilken ordning federater skall utföra uppgifter som berör initieringen av en federation. Dessa rekommendationer måste tillgodoses för att säkerställa att alla federater får tillgång till varje efterfrågad information från övriga federater. [Kuhl et al., 1999] redovisar en användbar strategi för detta. Detta koncept, som implementerats i HLA-Managern och i GenericHLAFederate, utnyttjar synkroniseringspunkter för att fastställa följande hållpunkter:

- Alla federater har anslutit
- Alla federater har genomfört publiceringar/prenumereringar
- Berörda federater har registrerat instanser och genomfört initial uppdatering av dessa
- Federationen är klar för exekvering
- Federationen kan avslutas

Figur 24 redovisar ett tillståndsdigram över hur initieringen av en federation fortgår. Värt att notera är att federationens avslut inte redovisas i denna figur (streckad pil). En federat ansluter till en federation och börjar med att specificera prenumerationer och publikationer. När detta är genomfört underrättas RTI om att federaten har uppnått "ReadyToPopulate" och federaten försätts i vänteläge. Efter att alla federater har anslutit till federationen annonserar HLA-Managern att den har uppnått "ReadyToPopulate", varpå federaterna registrerar eventuella objekt och genomför initial uppdatering av dessa. När registreringen är slutförd underrättas RTI att federaterna har uppnått "ReadyToRun" och federaten försätts i vänteläge. När detta moment är avklarat från alla federaters sida är synkroniseringspunkten "ReadyToRun" uppnådd och simuleringen kan börja.

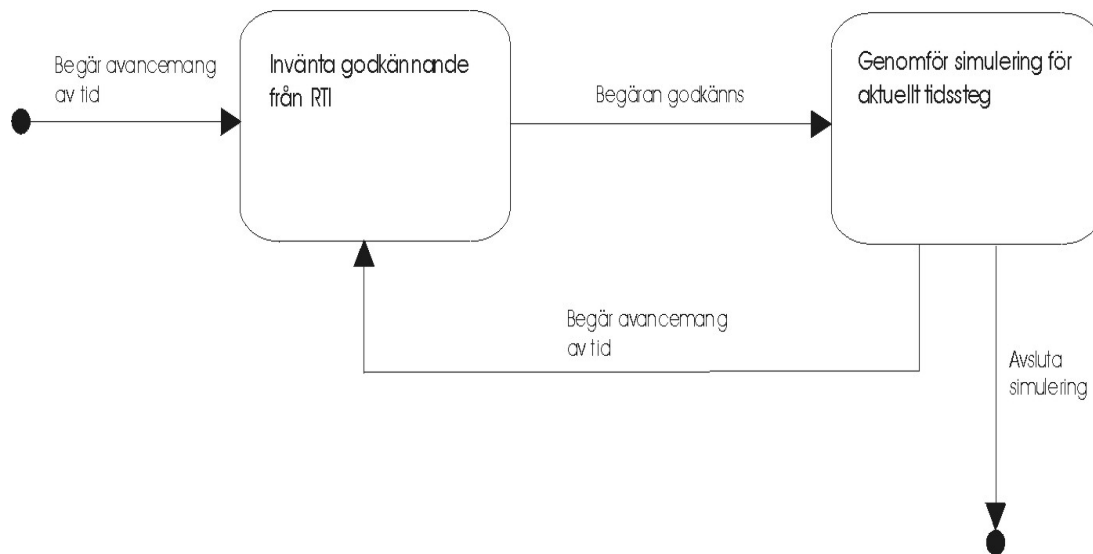


Figur 24. Principen för hur federationsinitiering hanteras.

3.3.2.2.3 Tidshantering

En federationsexekvering inom ramen för FOI_EXE förutsätter att enskilda federater med stor sannolikhet kommer att exekveras på datorer med varierande prestanda. Däröver tillkommer fluktuationer i nätverkets belastning som påverkar effektiviteten i kommunikationen mellan federater och RTI. Eftersom händelser i en simulering måste ske i tidsordning, har en konservativ tidssynkronisering införts som krav på federater inom FOI_EXE. Konservativ tidssynkronisering innebär att federater i en federation är

tidsbundna och tidsreglerande. Avancering av federationens tid fortgår därav enligt följande princip (se figur 25): federater begär avancemang av simuleringstiden från RTI. När alla federater har gjort denna begäran godkänner RTI denna och federaterna är fria att genomföra beräkningar för aktuellt tidssteg. Detta upprepas cykliskt till dess att simuleringen avslutas, för enskild federat eller för hela federationen. Denna implementering medför att HLA-Managern har förmågan att kontrollera avancering av federationers tid. Kontrollen omfattas även av paus i federationsexekveringen, samt förmågan att påverka exekveringshastigheten. Den maximala hastigheten begränsas dock av enskilda noders beräkningskapacitet.



Figur 25. Principen för hur konservativ tidshantering fungerar.

3.3.2.2.4 Avbrottshantering

En viktig delmängd av FOI_EXE är, som tidigare har beskrivits, att exekvering av federater sker i en dynamisk miljö där användare momentant har förmågan att tillgängliggöra/otillgängliggöra beräkningsresurser. På grund av detta måste en federat kunna flyttas från den nod där den för tillfället exekveras till en ny, ledig nod, utan att det påverkar federationen i övrigt. Detta medför att HLA-Managern har förmågan att "frysa" en federation (sluta avancera tiden), samt uppmana en enskild federat att spara aktuell simuleringsdata till fil och gå ur federationen.

Det finns tjänster implementerade i RTI som kan användas för att spara eller återskapa en federation. Dessa tjänster är dock inriktade mot att just spara/återskapa en hel federation och inte en enskild federat. På grund av detta ansågs det mer effektivt (framförallt vad gäller mängden nätverkskommunikation som krävs) att utöka federationernas FOM till att omfatta stöd för avbrottshantering. Denna utökning omfattas av två interaktionsklasser; "Sleep" och "Awake". "Sleep" används då en federat flyttas från en nod till en annan. HLA-Managern skickar en "Sleep"-interaktion på begäran från P2P-Managern. Interaktionen håller namnet på den federat som skall flyttas som parameter. Berörd federat sparar aktuell simuleringsdata till fil och går ur federationen. Federaten startas därefter upp på en ledig nod och initieras med den föregående federatens sparade data. När initieringen är avklarad skickar den nya federaten en "Awake"-interaktion till HLA-Managern. När HLA-Managern mottar denna är allt klart för vidare simulering och tiden börjar därmed avanceras igen.

3.3.2.2.5 Krav på deltagande federater

Den funktionalitet som FOI_EXE erbjuder medför vissa krav på deltagande federater. I den nuvarande versionen av FOI_EXE utgörs dessa krav av följande:

- Federater måste implementera utökningen av federationens objekt modell (FOM), dvs. prenumerera på interaktionen "Sleep" och publicera interaktionen "Awake". Detta sker automatiskt då GenericHLAFederate utnyttjas som grund för en federat, se nedan.
- Hantering av interaktionerna "Sleep" och "Awake", vilket omfattas av att federaten måste implementera metod för generering av fil utifrån aktuell simuleringsstatus, samt metod för inläsning av simuleringsdata från fil.
- Federater är tidsreglerande ("time-regulating") and tidsbundna ("time-constrained").
- Federater följer ett strikt sykroniseringsschema vid start av en federation.

3.3.2.2.6 GenericHLAFederate

Under arbetet med FOI_EXE utreddes möjligheten för att skapa en generell HLA-federat, här benämnd GenericHLAFederate. Syftet med detta arbete motiverades delvis av att isolera de krav som ställdes på federater av FOI_EXE, se ovan, men även av det faktum att en generell HLA-federat minskar implementeringsarbetet vid utveckling av nya federater. För närvarande finns följande funktionalitet implementerat i GenericHLAFederate:

- Funktionalitet som krävs av exekveringsmiljön:
 - Vid anslutning till en federation utförs automatiskt prenumeration på interaktionen "Sleep" och publicering av interaktionen "Awake" pga. de antaganden om federationen som redovisats ovan. Vidare blir federaten pga. dessa antaganden även tidsreglerande och tidsbundna vid anslutning.
 - Inför flytt av en federat mellan två noder sparas aktuell simuleringsdata till en fil (på begäran av en HLA-Manager). För närvarande kan enbart enkla datatyper sparas av den metod som hanterar detta. Efter flytt av en federat mellan noder initieras federaten med sparad simuleringsdata. Inläsning av simuleringsdata omfattas av samma datatyper, dvs. enkla datatyper. En förhållandevis enkel utökning av dessa metoder är att möjliggöra sparande av objekt .
 - Vid start av en federation följer federater det synkroniseringsschema som specificerats ovan. Hanteringen av detta är delvis implementerad i GenericHLAFederate.
 - Eftersom federaterna är tidsbundna och tidsreglerande finns funktionalitet för tidshantering inkluderad. Tidshanteringen sker enligt den metod som redovisats ovan.

- Generell funktionalitet:
 - Anslutning till den federation som anges av FOI_EXE. Parametrar som krävs för anslutning är; namn på federationen, adressen till RTI, samt det unika namn som federaten har i federationen.
 - Metoder för publicering/prenumerering på interaktioner.
 - Metoder för publicering/prenumerering på objekt, samt registrering och uppdatering av objekt.

3.3.3 Resultat

FOI_EXE uppfyller de krav som specificerades inför utvecklingsarbetet, nämligen; systemet skall utnyttja beräkningsresurser i ett nätverk av arbetsstationer ("Network of Workstation", NOW), bygga på ett decentraliserat koncept (P2P) och slutligen exekvera HLA-baserade simuleringar. Vissa aspekter av FOI_EXE kräver dock vidareutveckling, då den grundläggande funktionaliteten i JXTA inte alltid kan leverera en komplett lösning.

En rad problem relaterat till utformningen av fildelningstjänsten (CMS) i JXTA upptäcktes allteftersom FOI_EXE implementerades, men även efter tester av miljön. En svaghet är hantering av metadataassociation till de resurser som delas av systemet. Idag kan en sökning efter resurser baseras på nyckelord (textsträng med nyckelord) eller text som specificeras vid utdelning av en resurs. Vad den övergripande simuleringsmiljön kräver av ett fildelningssystem är nyckelord knutna till specifika kategorier. Detta innebär att ett "advertisement" för en resurs bör inkludera fördefinierade fält (typ plattform, organisation etc.) som definieras vid utdelning av en resurs (federat). Detta lägger grunden till en mer flexibel sökning och logisk indelning av resurser. Vidare är det enbart möjligt att använda ett sökord vid sökning i CMS, vilket begränsar användbarheten. Vid en sökning är även hantering av filtermekanismen något begränsad. Istället för att inkludera ett sökfilter i den förfrågan som skickas ut bland deltagande noder, filtreras svaret på förfrågan lokalt hos den initierande noden. Detta innebär att nätverket måste hantera stora mängder kommunikation helt i onödan.

Även om GenericHLAFederate endast implementerar en delmängd av HLAs gränssnittsspecifikation, visar konceptet att utveckling av federater baserat på återanvändbara komponenter är fördelaktigt. Ett bra exempel är hur hantering av tjänster inom "Declaration management" har förenklats radikalt.

4 Diskussion

4.1 Webbanpassade HLA federationer

Den miljö som diskuteras i rapporten är inte komplett utan är ett första steg i utvecklingen av en webbaserad utvecklingsmiljö för komponentbaserade simuleringar. Miljön är tänkt att vara en startpunkt för vidare forskning inom området och skulle med fördel kunna utvecklas vidare i framtiden. Till exempel vore det intressant att studera vidare användningen av servleten och hur den kan utnyttjas för webbanpassning av HLA-federationer. Det finns försök, t.ex. [Housand & Hudgins, 2001], där själva servleten har använts som federat. Det vore även intressant att arbeta vidare med utveckling av "hierarkiska federationer" i miljön.

Den webbanpassade HLA-miljön ovan saknar i nuläget scenariorhantering. Det skulle vara bra att införa scenariorhantering som tillåter användare att stoppa, pausa och backa simuleringen etc. Att dessutom införa någon slags tidshantering i miljön vore bra. Idag finns ingen direkt tidshantering utan allt sker i "realtid", se [Svensson, 2002]. Det skulle kanske vara en fördel om ett scenario kunde spelas in och sedan spelas upp i en för användaren önskad hastighet etc.

En åtgärd som skulle kunna göra en sådan komplett utvecklingsmiljö än mera fördelaktig vore att göra den officiell. Det vill säga att låta alla inom försvarsmakten ha tillgång till simuleringsnätverket och själva kunna lägga in modeller i en databas som håller reda på resurser och komponenter inom nätverket – såsom t.ex. HLA-federater. Det skulle sedan kunna gå att leta i databasen via webbläsare efter alla tillgängliga komponenter vilket skulle medföra återanvändning av modeller och stor besparing av arbetstid och -kostnad.

4.2 Webbanpassade simuleringsmodeller

Endast en delmängd av de funktioner som finns tillgängliga i Tavast2 implementerades i Webb-Tavast. Det är dock tillräckligt för att visa att den valda tekniken lämpar sig väl för ändamålet. De möjligheter som finns inom Java för att utveckla grafiska gränssnitt är tillräckliga för att implementera en fullständig version av Webb-Tavast. Arbetet med att göra Tavast2 webbaserat krävde förhållandevis stora ingrepp i programmets källkod, något som är önskvärt att minimera i så stor utsträckning som möjligt. Detta varierar med stor sannolikhet från fall till fall beroende på strukturen i programmet som ska modifieras.

Endast en del av de parametrar som kan specificeras i Arteva har gjorts möjliga att ändra i Webb-Arteva. Trots detta är Webb-Arteva ett bra exempel på hur en förhållandevis enkel webbt teknik kan utnyttjas för att ge ett grafiskt användargränssnitt till en textbaserad simuleringsmodell. En fullt utbyggd version av Webb-Arteva skulle erbjuda ett långt mer användarvänligt och lättillgängligt verktyg än den tidigare versionen. En fördel med systemet är att detta kan implementeras utan att simuleringsmodellen (Arteva) behöver genomgå några förändringar.

I första versionen av Webb-Arteva kan endast parametrar utan inbördes relation specificeras. Vid en fullständig implementering måste hänsyn tas till de inbördes beroenden som existerar mellan vissa parametrar. Om en viss typ av artillerienhet väljs kan t. ex. endast en delmängd av ammunitionstyperna fungera. Detta kräver en mer dynamisk programmeringsteknik än den som utnyttjas i första versionen av Webb-Arteva.

Beslutet att utnyttja Java-baserad teknik för utveckling av Webb-Tavast och Webb-Arteva grundas på en rad faktorer. Eftersom servlets utvecklas i Java kan dessa

användas över flertalet operativsystem och servrar, tekniken är plattformsoberoende. Utveckling i Java medför även en högre grad av säkerhet, jämfört med t. ex. C++, när det gäller minneshantering och felhantering. Vidare bidrar servlet-tekniken till en bättre skalbarhet av webbaserade applikationer jämfört med andra tekniker. När en servlet används för första gången laddas den till serverns minne, där en instans av objektet betjänar alla förfrågningar. Servleten förblir i serverns minne och kan därför bibehålla kopplingar till utomstående resurser som t. ex. databaser. I och med detta blir webblösningen även mer tidseffektiv.

4.3 Distribuerad exekveringsmiljö

Valet av JXTA som grundläggande plattform för FOI_EXE var inte självklar. Det existerar en rad andra lösningar som i många fall bygger på en mer mogen teknologi än JXTA, exempelvis JINI [Jini]. Idag lider JXTA mer av bristfällig implementering av de protokoll som utgör standarden, än av den grundläggande filosofin bakom konceptet, vilket bl.a. har påpekats av [Halepovic & Deters, 2002]. Till detta tillkommer en förhållandevis sparsam dokumentation av plattformen vilket utelämnar utvecklaren till att granska källkoden i syfte att förstå en bakomliggande process. Många av de svårigheter som har uppkommit under detta arbete kan hänföras till implementeringen av fildelningstjänsten CMS i JXTA. Det skall dock tilläggas att CMS inte tillhör kärnan av tjänster i JXTA och det var på förhand känt att vissa aspekter av CMS var otillräckliga. CMS valdes ändå till att fungera som ramverk för fildelning i FOI_EXE då det innebar en klar reduktion av utvecklingsarbetet. CMS är även ett "öppet" system som i framtiden kan vidareutvecklas för att uppfylla våra syften bättre.

I nuvarande version av FOI_EXE finns inte något stöd för hantering av hierarkiska federationer, men HLA-Managern skulle kunna förses med funktionalitet som stödjer detta. Typen av hierarkiskt federationsstöd skulle då vara en s.k. Federate Proxy (FP). I detta sammanhang är det intressant att notera begreppet distribuerad FP (DFP), som introducerats av [Cramp & Oudshoorn, 2002]. I detta system existerar en "lokal" FP för varje enskild federation istället för en "global" FP som är gemensam för alla federationer. Vinsten med denna lösning är bland annat effektivare filtrering av data, samt inkapsling av en federations interna information. Givet två federationer som existerar inom två skilda LAN, filtreras data innan den lämnar den lokala domänen, vilket begränsar mängden information som måste transporteras över Internet. Vidare har en metod för konservativ tidshantering utvecklats för detta system [Oudshoorn et al., 2002] vilken stämmer överens med den strategi som har implementerats i HLA-Managern och GenericHLAFederate.

Tillgång till generiska och återanvändbara HLA-komponenter kan avsevärt förenkla implementering av nya federater. Genom att isolera simuleringslogik från HLA-implementering kan utvecklingskostnader minskas, nyttjandet av HLA som standard för distribuerad simulering ökas och återanvändning av kod maximeras. Genom att tillämpa en komponentbaserad programmeringsmodell vid utveckling av federater kan detta uppnås. Ett exempel på implementering av en komponentbaserad programmeringsmodell för HLA presenteras i [Radeski & Parr, 2002]. I denna artikel föreslås en Simulation Component Model (SCM) för HLA, som bygger på välbeprövade koncept som EJB och COM/.NET, i syfte att lägga grunden till en infrastruktur för utveckling av HLA federater. Även andra studier har påvisat betydelsen av att skapa en alternativ programmeringsmodell för HLA. I [Cengiz & Oguztuzun, 2002] presenteras en samling återanvändbara HLA-komponenter baserade på Microsofts COM-teknologi. [Cusack & Hoare, 1999] och [Cusack et al., 2000] presenterar komponentmodeller för HLA baserade på JavaBeans. Allt detta pekar på

betydelsen av att i projektet skapa ett ramverk (komponentmodell) för utveckling av federater och/eller anpassning av existerande simuleringsmodeller, till FOI_EXE. Detta ramverk bör inkludera de krav som FOI_EXE ställer på deltagande federater samt implementera den funktionalitet som definieras i HLAs gränssnittsspecifikation.

4.4 Framtidsvision

Det långsiktiga målet för projektet har varit att erhålla en djupgående förståelse för området webbaserade simuleringar. Vår strävan har varit att identifiera hur och hur mycket webbt teknologin påverkar modellering och simulering, samt för- och nackdelar med denna kombination. Vår framtidsvision är ett system med bl. a. följande egenskaper:

- Ett distribuerat nätverk med noder av olika slag, t. ex. sensorer, databaser, ledningscentraler, soldater utrustade med mobila informationsterminaler (handdatorer, mobiltelefoner, etc.), osv.
- Noderna kan anpassas för individuella användare så att nätverket kan ha olika funktioner.
- Noderna skall kunna erbjuda tjänster av olika slag t. ex. simuleringsmodeller, sensordata, geografiska informationssystem, osv.
- Den verkliga världen skall vara integrerad med en virtuell värld.
- Den virtuella världen skall kunna befolkas med datorgenererade styrkor på olika aggregationsnivåer för att simulera fiender och olika taktiska förhållanden.

En fortsättning på det arbete som har utförts inom projektet är att studera närmare de frågor och aspekter på webb och M&S som har identifierats. Områden av särskilt intresse är kollaborativ M&S (framförallt kollaborativ exekvering av modeller), P2P (resursallokering och -utnyttjande, decentraliserade arkitekturer, etc.), komponentbaserad M&S och alternativa arkitekturer till HLA (analogier med programvaru- och systemutvecklingsprocesser, återanvändning och interoperabilitet), distribuerat modellbibliotek och generella webbaserade visualiseringsmoduler (Java3D, VRML, etc.). Samtidigt är det av stor vikt att undersöka påverkan av webb på andra M&S-områden, som t. ex. VV&A (verifiering, validering och ackreditering), CMMS (Conceptual Models of the Mission Space) och CGF (Computer Generated Forces). Vi anser att dessa kan byggas in och realiseras i form av en miljö bestående av två huvuddelar, en konstruktionsmiljö och en exekveringsmiljö.

4.4.1 Konstruktionsmiljö

Konstruktionsdelen skall vara en webbaserad simuleringsmiljö som erbjuder möjligheten att olika organisationer, eller olika enheter i samma organisation, skall kunna konstruera, utveckla och testa modeller, samt lägga dessa i ett webbaserat bibliotek. Modellbiblioteket kan antingen vara centraliserat eller distribuerat på flera datorer, med kravet att modellerna är tillgängliga för alla engagerade parter via WWW.

I denna miljö skall modellutvecklaren kunna använda existerande (del-) modeller och komponera nya modeller. Därmed kommer modellutvecklingsprocessen att delvis vara komponentbaserad, där t. ex. COTS-produkter (Commercial-Off-The-Shelf) kan användas i utvecklingsprocessen.

Utvecklingsmiljön skall dessutom ha stöd för dokumentering (baserad på standarder som XML), för att bl. a. stödja VV&A- och CMMS-processer, samt återanvändning och interoperabilitet.

4.4.2 Exekveringsmiljö

En annan viktig del av det tänkta systemet är en vidareutveckling av den framtagna exekveringsmiljön. Den skall kunna erbjuda två olika sätt att exekvera modeller. Modeller skall kunna exekveras:

- På en dator (t. ex. PC)
- I en ”Peer to Peer”-miljö

Det enklaste fallet är att exekvera modeller på en dator, men i många fall behöver modellerna större datorresurser (t. ex. beräkningskapacitet och minne) än vad som kan erbjudas av en ensam PC. I det andra fallet kan man använda en Peer-to-Peer-miljö, där modellexekveringen distribueras på ett antal utnyttjade datorer. Syftet med P2P är att försöka skapa en infrastruktur för optimalt utnyttjande av befintliga nätverksresurser (såsom minne och bearbetningskapacitet) i en organisation, genom dynamisk resursallokering. Målet med denna del av projektet är att undersöka vilka fördelar P2P kan erbjuda nätverksbaserade simuleringar. Miljön skall dessutom vara kopplad till det distribuerade modellbiblioteket, samt stödja kollaborativt exekvering av modeller.

5 Slutsatser

Våra slutsatser från arbetet bekräftar våra preliminära antaganden om fördelarna med kopplingen mellan webb och HLA-federationer och simuleringar. Vi tror att området har stor potential och att webbaserade simuleringar skulle kunna ha en central roll i framtida nätverksbaserade försvarssystem. För att kunna dra nytta av området bör dock framförallt metodiken och tekniken utvecklas vidare.

Ett krav för att arbeta med webbaserad M&S är att de plattformsoberoende miljöer och tekniker som används har hög säkerhet och är anpassade för webben. Javateknologin är den mest lämpliga för tillfället, eftersom den erbjuder ett standardiserat, konsistent, säkert och plattformsoberoende ramverk för utveckling av skalbara, webbaserade och distribuerade applikationer.

Komponentbaserad modellutveckling, som är den givna utvecklingstrenden efter objektorienterad modellutveckling, erbjuder ett kostnads- och tidseffektivt sätt att bygga modeller och simuleringar. Det är dock viktigt att välja ändamålsenlig arkitektur. Samtidigt krävs det mycket arbete för att effektivt kunna utnyttja tekniken. HLA är den arkitektur som har lanserats inom M&S-världen, vilken tillsammans med tillhörande processer, som FEDEP och CMMS har potentialen att förverkliga visionen om komponentbaserad M&S. Hierarkiska federationer inom HLA skulle kunna vara ett sätt att ytterligare stödja denna vision. Det behövs dock mer arbete för att kunna dra sådana slutsatser. Här skulle vi dessutom vara mycket betjänta av att undersöka och följa utvecklingen av det senaste inom programvaruutvecklingsmetodik, -teknik och -verktyg. Många av de problem som vi möter idag inom komponentbaserad M&S, hanterats med framgång inom programvaruutvecklingsdomänen.

Att webbanpassa befintliga simuleringar kan leda till att de blir både användarvänligare och mer lättillgängliga, men det kan i vissa fall innebära stora ändringar i koden för att lyckas med uppgiften. Om det dessutom inte används lämplig teknik eller simuleringar med tung grafik, kan det uppstå för stora fördröjningar och tidskrävande körningar.

Den största fördelen med exekvering av federationer inom en P2P-miljö är att arbetet fördelas jämnt mellan noder, vilket eliminerar möjliga flaskhalsar i systemet. Resultaten från detta arbete visar att P2P-system har många fördelar jämfört med centraliserade system, men att de är mer komplexa att implementera. Slutligen måste federater som skall exekveras av miljön anpassas till systemet, vilket gör utvecklingen av federater mindre generell.

6 Referenser

- [ALICE] http://www.comp.nus.edu.sg/~ngyewkwo/hons_proj/ (senast besökt 2002-01-14).
- [Arteva, 2001] Manual för Arteva (Windows-version), 2001, FOI.
- [Brookshier et al., 2002] Brookshier D., Govoni D., Krishnan N., *JXTA: JAVA P2P Programming*, SAMS, Indianapolis, US.
- [Cengiz & Oguztuzun, 2002] Cengiz Z.S., Oguztuzun H., *A COM Component for HLA Interface Specification*, Proceedings of the Fall 2002 Simulation Interoperability Workshop, Orlando, Florida.
- [Chien-Chung Shen, 1998] Chien-Chung Shen, *Discrete-Event Simulation on the Internet and the Web*, Proceedings of the Western Multi-Conference (WMC), 1998.
- [Cramp & Oudshoorn, 2002] Cramp A., Oudshoorn M., *Employing Hierarchical Federation Communities in the Virtual Ship Architecture*, Twenty-Fifth Australasian Computer Science Conference, Melbourne, Australia.
- [Cusack & Hoare, 1999] Cusack M., Hoare P., *Component based development in HLA using Java*, Proceedings of the Spring 1999 Simulation Interoperability Workshop.
- [Cusack et al., 2000] Cusack M., Exon N., Hoare P., Hoyle R., *Visual construction of JavaBean federates*, Proceedings of the Spring 2000 Simulation Interoperability Workshop.
- [Dahman et al., 1997] Dahman J.S., *The Departement of Defence High Level Architecture*, Proceedings of the 1997 Winter Simulation Conference.
- [DMSO HLA] HLA forskning vid DMSO, <http://www.dmsomil/index.php?page=64> (senast besökt 2001-12-20).
- [Fishwick, 1996] Fishwick P.A., 1996, *Web-Based Simulation: Some Personal Observations*, In: Proceedings of the 1996 Winter Simulation Conference, Coronado, CA.
- [Gong, 2001] Gong L., 2001, *Project JXTA: A technology Overview*, http://www.jxta.org/project/www/white_papers.html.
- [HLA Interface Specification] HLA sidor hos DMSO, <https://www.dmsomil/public/transition/hla/> (senast besökt 2002-12-03).
- [HLA Rules] HLA sidor hos DMSO, <https://www.dmsomil/public/transition/hla/> (senast besökt 2002-12-03).
- [Housand & Hudgins, 2001] Housand C., Hudgins B., 2001, *Bringing HLA to the Web: The Incorporation of Enterprise Java Technology into HLA Federations*, The AEGIS technologies group inc.
- [Jini] Projekt Jini, <http://www.jini.org> (senast besökt 2002-12-03).
- [JXTA Home Page] Projekt JXTA, <http://www.jxta.org> (senast besökt 2001-11-07).
- [JxtaSpaces] JXTA, Projekt JxtaSpaces, <http://www.jxta.org> (senast besökt 2001-10-15).

- [Kayl, 2001] Kayl K., 2001, *JOY POSES JXTA INITIATIVE, Pushing the Boundaries of Distributed Computing*, <http://java.sun.com/features/> (senast besökt 2001-11-07).
- [Kuhl et al., 1999] Kuhl F., Weatherly R., Dahman J., *Creating Computer Simulation Systems*, Prentice Hall, Upper Saddle River, US.
- [Liu et al., 2002] Liu L., Turner S.J., Lee B.S., Li G., *DDM Implementation in Hierarchical Federations*, Proceedings of the Fall 2002 Simulation Interoperability Workshop, Orlando, Florida.
- [Luthi, 2000] Lüthi J., Großmann, S., *The Resource Sharing System: Dynamic Federate Mapping for HLA-based Distributed Simulation*, Proceedings: The Fifteenth Workshop on Parallel and Distributed Simulation (PADS 2001), Lake Arrowhead, CA, US.
- [NanoComp] <http://www.kbs.twi.tudelft.nl/Research/Projects/CollSim/view.html> (senast besökt 2001-12-08).
- [NätSim, 2002] *Nätverksbaserad Modellering och Simulering*, Projekttrapport, FOI, 2002.
- [Oudshoorn et al., 2002] Oudshoorn M., Cramp A., Best J., *Time Management in Hierarchical Federation Communities*, Proceedings of the Fall 2002 Simulation Interoperability Workshop, Orlando, Florida.
- [Page, 1998] Page E.H., 1998, *The Rise of Web-based Simulation: Implications for the High Level Architecture*, In: Proceedings of the 1998 Winter Simulation Conference, Washington, DC.
- [Page, 1999] Page E.H., 1999, *Investigating the Application of Web-Based Simulation Principles within the Architecture for a Next-Generation Computer Generated Forces Model*, The MITRE Corp.
- [Parlor] JXTA, Projekt Parlor, <http://www.jxta.org> (senast besökt 2001-10-15).
- [Patzner, 1999] Patzner A., 1999, *Web Application Development*, In: Professional Java Server Programming, Wrox Press, Birmingham.
- [Radeski & Parr, 2002] Radeski A., Parr S., *Towards a Simulation Component Model for HLA*, Proceedings of the Fall 2002 Simulation Interoperability Workshop, Orlando, Florida.
- [Schildt, 2001] Schildt H., 2001, *Java2, The Complete Reference*, Osborne/McGraw-Hill, Berkley, California.
- [SETI@Home] Projekt SETI@Home, <http://setiathome.ssl.berkeley.edu/> (senast besökt 2002-12-09).
- [Silk®] <http://www.threadtec.com> (senast besökt 2001-12-17).
<http://www.cs.lr.tudelft.nl/projects/simultaan/> (senast besökt 2001-12-18).
- [Sparf, 2003] Sparf M., *Peer-to-Peer as a Solution to Resource Discovery and Allocation in Web-based Simulation*, Examensarbete DSV.
- [Sparf, 2002] Sparf M., *Kortfattad reserapport från "Second International Conference on Peer-to-Peer Computing" 5-7 September 2002 Linköping, Sweden*, FOI Systemmodellering.

- [SUN, 1999] Sun Microsystems, 1999, "Comparing JavaServer Pages Technology and Microsoft Active Server Pages", <http://www.sun.com>, (senast besökt 2001-11-25).
- [Sundsted, 2001] Sundsted T., 2001, *The practice of peer-to-peer computing: Introduction and history*, <http://www-106.ibm.com/developerworks> (senast besökt 2001-05-15).
- [Svensson, 2002] Svensson O., 2002, "Web-Based HLA Federations and Simulations", FOI, under utgivning (februari 2002).
- [Tavast2, 1999] Manual för Tavast2, 1999, FOI.
- [Ulriksson et al., 2002] Ulriksson J., Moradi F., Svensson O., *A Web-based Environment for building Distributed Simulations*, Proceedings of the European Simulation Interoperability Workshop, ESIW'02, London, England, June 2002.