Preliminär specifikation för programutvecklingsprojekt

# Nätverksbaserat krigsspel för forskningsändamål


Uppdragsgivare

Joel Brynielsson Klas Wallenius

Gruppmedlemmar

Henrik Bäärnhielm Andreas Enblom
Jing Fu Zi Niklas Hallenfur
Karl Hasselström Henrik Hägerström
Oskar Linde Jon Åslund


14 maj 2001

## 1 Problembeskrivning

### 1.1 Bakgrund

Inom försvarsforskning använder man sig av simuleringar för att öva personal, ta fram optimala truppsammansättningar, simulera tänkta stridsförlopp etc.

Trots det stora utbudet av datorspel på den kommersiella marknaden visar studier på Försvarshögskolan att det idag inte finns något bra datorbaserat krigsspel lämpat för forskningsändamål.

### 1.2 Syfte

Projektet går ut på att ta fram ett generellt och utbyggbart system som låter användarna spela olika typer av krigsspel mot varandra. Det ska vara enkelt att skapa nya spel och scenarier för existerande spel. En stor del av projektet kommer att gå ut på att i samråd med uppdragsgivarna utarbeta hur systemet ska vara uppbyggt och vilken funktionalitet som ska stödjas.

### 1.3 Krav och avgränsningar

Forskningsvärlden behöver ett nätverksbaserat datorspel med öppen och väldokumenterad källkod skriven i Java. Dokumentation och kommentarer i källkoden skrivs på engelska. Spelet skall vara av strategityp där man flyttar enheter på en geografisk karta.

Mycket kraft måste läggas på att definiera spelet så att det blir så allmänt som möjligt så att alla tänkbara datorspel kan spelas. All indata gällande grafik, terrängens uppförande, antal enheter etc. etc. skall lätt kunna genereras av icke programmeringskunnig person via t.ex. textfiler eller editorer. En del kommersiella krigsspel erbjuder möjligheten att generera egna scenarior och kan därmed tjäna som bra inspiration.

Vi ser två huvuddelar som skall utvecklas, en serverdel och en klientdel. Servern, eller själva spelmotorn, sköter spelförloppet. Till servern kopplas ett godtyckligt antal spelklienter.

Inom detta projekt begränsar vi oss till att en sorts spelklient där en spelare manuellt flyttar sina enheter tillverkas. I framtiden tänker vi oss dock att forskare får laborera med att skapa egna smarta fiender". Dessa består av klienter som flyttar enheter automatiskt enligt kluriga algoritmer, t.ex. algoritmer baserade på maskininlärning (inom Försvarshögskolan har försök med genetiska algoritmer gjorts).

### 1.4 Funktioner

Systemet kommer att utvecklas i Java och kan delas upp i tre delar; server, kommunikation och klient. I allmänhet körs servern och den delen av kommunikationen som inte byggs in i klientprogrammen endast på en dator medan flera användare ska kunna använda olika typer av klienter (t.ex. spelare i det gröna laget, spelare i det röda laget och övervakare) på ett antal olika datorer. Systemet kommer att kräva att de inblandade datorerna är ganska snabba och utan problem kan använda de mest avancerade funktionerna i Java. Kommunikationen mellan klienten och servern kommer att realiseras med HLA-systemet där det finns många färdiga implementationer för dataöverföring. Systemet kommer också att kräva att datorerna kan kommunicera med varandra med TCP/IP.

Användare av systemet kommer att vara försvarsforskare och militärer under utbildning.

## 2 Förslag till lösning

I figur 1 visas en skiss över alla de komponenter som ska ingå i systemet.

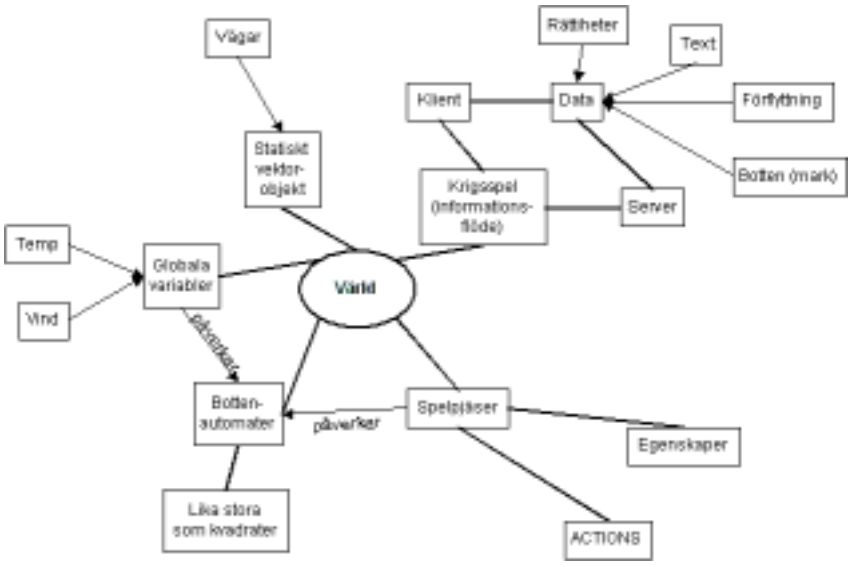
Globala parametrar kan t.ex. vara vind eller temperatur. Globala parametrar berör hela spelplanen. Vi har alltså *en* vindriktning i hela Sverige om simuleringen visar en karta över Sverige.

Systemet kommer att ha tre lager:

1. Spelpjäser, vektorbaserade, har egenskaper, kan flyttas osv.
2. Statisk vektorobjekt (SVO), vektorbaserade, står still, t.ex. vägar.
3. Marken, automatbaserat, kan ändras beroende på regler

Diskussioner gällande SVO förs fortfarande. Vi är inte övertygade om att de verkligen är nödvändiga för produkten. Det enda speciella användningsområde vi hittintills funnit är för att modellera vägar.

För att lösa problemet med olika klienter som styr olika enheter i spelet så kommer en modul för att hantera rättigheter inkluderas i plattformen. Denna




Figur 1: Alla komponenter i systemet.

har exempelvis ansvaret för att två motståndarklienter inte kan styra varandras pjäser, medan en loggklient ska kunna läsa allt som händer i spelet.

Spelpjäser har egenskaper och actions. Egenskaper är t.ex. hälsa, attackvärde, försvarsvärde, snabbhet. Actions definierar vad en specifik spelpjäserna kan göra, t.ex. flytta, attackera. Actions kan påverka andra spelpjäser/automater/svo.

## 2.1 Terräng/automater

Terrängen kommer att bestå av en rutnät där varje ruta motsvarar en viss terrängtyp. Varje ruta kommer också att fungera som en tillståndsautomat. En noggrannare beskrivning av vad automaterna/terrängenheterna innebär följer nedan.

- Kvadrater, så att man kan måla ut dem i ett bitmappsprogram.
- Varje ruta är en automat. Då den uppdateras tittar den på sitt eget tillstånd och omgivande rutors tillstånd, och ändrar sitt tillstånd utifrån regler definierade av spelkonstruktören.
- Automaterna ställer sig i kö för att bli uppdaterade om de befinner sig i ett dynamiskt tillstånd. Exempelvis skall skog som brinner så småningom brinna ner helt och övergå i ett annat tillstånd. När en automat uppdateras kan den utifrån det aktuella tillståndet samt de av spelkonstruktören givna reglerna avgöra om den inom en snar framtid måste uppdateras igen. Den kö som automaterna läggs in i är en prioritetskö. Automater exekveras i tur och ordning sorterade efter tidpunkt de ville uppdateras på.
- En automat kan, då den uppdateras, lägga in närliggande automater i kön så att de uppdateras, t.ex. för att brand skall kunna spridas från automat till automat.

- Anledningen till kön är att stora områden bara behöver uppdateras sällan, om ens någonsin (berg, sjöar). Vi vill kunna använda oss av ett stort antal automater, möjligtvis så många som en miljon, och att uppdatera alla dem en gång i sekunden är inte möjligt. Dessutom vill vi att ett snabbt brandförlopp ska uppdateras mer än en gång i sekunden.

## 2.2 Spelpjäser/enheter

- Det finns två typer av enheter, atomära enheter och grupper. Grupper består av atomära enheter eller grupper. Enheter har vissa värden såsom attackvärde, hälsa, försvarsvärde. Vilka de är och hur de används definieras i spelreglerna.

- För atomära enheter definierar reglerna direkt vad värdena ska vara (t.ex. hälsa 15 från början, attackvärde två gånger hälsan, etc.). För grupper definierar reglerna hur värdena ska beräknas från värdena på de ingående atomära enheterna eller grupperna (t.ex. att gruppens hälsa ska vara summan av de ingående enheternas hälsa).

- Reglerna definierar olika slags atomära enheter, t.ex. plutoner och tanks. Varje typ har sin egen uppsättning värden. De definierar också olika typer av grupper. Varje typ har sina egna regler för hur delarna kombineras till en helhet, och sina egna regler för vilka delar som får ingå. Exempelvis kanske en bataljon måste bestå av mellan två och fem plutoner och inget annat, medan en kvataljon måste bestå av minst två bataljoner och inget annat.

- Enheter kan kanske ha flera olika radier. En radie för utbredning, en för attack, en för synfält, etc. Grupper har nog oftast större radie än de ingående enheterna, men mindre än summan av dessa.

- Enheter förflyttas genom att man med musen klickar på ett antal punkter. Enheten gör sedan i räta linjer till punkt efter punkt. Ett specialfall är vägar. Klickar man ”tillräckligt” nära en väg (så nära att muspekaren ändrats för att indikera att man klickar tillräckligt nära) så ska enheten då den går mellan två sådana punkter inte gå i en rät linje, utan följa vägen istället.

- Objekt som vägsegment och byggnader (eller delar av byggnader) är stationära enheter som kan skadas och explodera precis som andra enheter, men beskrivs av polygoner istället. Eftersom enheter har bara en hälsa t.ex., måste saker som vägar och stora byggnader beskrivas som flera enheter för att man ska kunna förstöra bara bitar av dem.

## 2.3 Interaktion mellan enheter och terräng

- Terräng påverkar enheter. En bataljon som står på en ruta med brinnade skog blir skadad. En bataljon kan röra sig fortare över en prärie än genom en djungel, och inte alls över vatten eller Kebnekajse. Tänk också på att en enhet inte antingen står i en ruta eller inte. Snarare är det så att enheten och rutan överlappar varandra mer eller mindre (möjligen inte alls).

- Enheter påverkar terräng. Bomber får skog att börja brinna, brandbilar får brinnande skog att slockna, etc. Enheter som påverkar tillståndet på en automat genom att utföra en action av något slag medför att den berörda automaten läggs in i kön för omedelbar uppdatering.

## 2.4 Actions

Varje spelpjäs ska ha möjlighet att utföra ett antal verb. Dessa benämns som actions.

Actions kan hanteras på två sätt:

1. Sådana som utförs på en gång (t.ex. att i detta nu släppa bomber från flygplanet på den plats det befinner sig över).

2. Sådana som utförs i en viss ordning (t.ex. förflyttning längs en rutt bestående av ett antal punkter, för att sedan desertera och ansluta sig till fienden). Dessa placeras lämpligtvis i en kö.

Då en action av typ två är färdigutförd, svarar servern med ”lycka”, ”misslyckad” eller ”target lost”. Vad dessa svar betyder framgår av nedanstående resonemang.

Några actions som bör finnas med är:

- Förflyttning. En förflyttning innefattar bara förflyttning längs en rät linje mellan två punkter. Om man vill förflytta en spelpjäs längs någon rutt får man göra en förflyttning i taget och då servern svarar att förflyttningen lyckades (med svaret ”lyckad”, se ovan), skickar man över nästa förflyttning längs en rät linje. Om det inte gick att förflytta spelpjäsen i enlighet med den senaste förflyttningen (t.ex. om det låg en sjö i vägen för kavalleriet) svarar servern ”misslyckad” och klienten tömmer kön för den spelpjäsen. Det kan finnas olika typer av förflyttning (t.ex. galopp eller skritt för kavalleriet), med olika egenskaper.

- Attack (beskjutning). När man väljer detta kan det med fördel visas en radie inom vilken attack är möjlig (för flygplan beror detta på bränslet, och för artilleri på skjutvapnens räckvidd), och då spelaren klickar på en trupp eller på en markbit (beroende på vilket truppslag vi talar om) utförs ordern direkt (för t.ex. den oflyttbara kanontruppen) eller så läggs ordern (förhoppningsvis först) i kön.

Hur man löser några specialfall på detta sätt:

- Vägar. Om både startpunkten och slutpunkten för en viss förflyttning ligger på samma väg, tillfrågas spelaren huruvida denne vill förflytta sig längs vägen, och i så fall förflyttas enheten på vägen tills dess att slutpunkten är nådd. Det är upp till servern att veta att en trupp kan hålla högre hastighet på en väg än i annan terräng.

- Att attackera en trupp långt bort. När man väljer att attackera en trupp långt bort (som man vet finns där eftersom någon spelpjäs ser den) skickas actionen ”attackera trupp” till servern. Servern ser till att den attackerande truppen förflyttas mot truppen långt borta. Om servern märker att spelaren inte längre ser truppen, avslutas attacken och ”target lost” skickas

tillbaka. Klienten väljer då att lägga en förflyttning till den punkten först i kön (vilket man får göra i vår definition av en kö) och varje gång någon av ens egna spelpjäser får syn på en motståndartrupp gör en bedömning av om denna nya trupp kan vara den man tänkte anfalla förrut, och i så fall ändrar man förflyttningen till samma typ av attack som ovan.

### 2.5 Användargränssnitt

Användargränsnittet kommer att bestå av en karta, och ett antal informationsrutor.

På kartan presenteras terrängen och spelpjäserna. För att ge en eller flera av sina spelpjäser instruktioner markeras dessa med musen (shift-tangenten hålls nere för att markera flera). Sedan klickar man med högerknappen (eller alt-klickar på Macintosh) för att få upp en meny med actions som de markerade enheterna kan utföra. När en action väljs utförs den omedelbart om den inte kräver några argument, och annars visas en aktionsradie för just den actionen och användaren klickar på kartan eller på andra spelpjäser för att ange de argument som denna action kräver.

Informationsrutorna kommer att innehålla värdet på de globala parameterna, systemmeddelanden och information om markerade spelpjäser.

## 3 Administration

### 3.1 Ansvarsfördelning

Projektledare är Andreas Enblom och Henrik Hägerström. Vidare delas medlemmarna in i tre grupper, servergruppen (Henrik Bäärnihielm, Karl Hasselström och Henrik Hägerström), kommunikationsgruppen (Niklas Hallenfur och Jon Åslund) och klientgruppen (Andreas Enblom, Jing Fu Zi och Oskar Linde).

### 3.2 Tidsplanering

| Period | Arbete |
|---|---|
| 31 jan – 15 feb | Inledande studier och dikussioner, studiebesök på försvarshögskolan. |
| 15 feb – 9 april | Noggrann specifikatin och design. |
| 9 april – 27 april | Implementation. |
| 27 april – | Slutdokumentation, avstämning, uppstädning och presentation. |

### 3.3 Möten

Gruppen träffas i stort sett varje onsdag, oftast tillsammans med uppdragsgivarne för att arbeta med systemlösningar och specifikation av systemet. Utöver detta träffas också de mindre grupperna för att diskutera och designa sina delar.

## 4 Riskanalys

De största riskerna är (sorterade efter, som kursledaren uttrycker det, sannolikhet):

1. Att automattekniken visar sig bli för långsam med teknologier som Java. Lite försök som gjorts här visar dock att det bör gå bra. Om det trots allt skulle bli ett problem får man lösa det genom att göra färre automater eller genom att minska uppdateringsfrekvensen.

2. Att uppdragsgivarna inte tycker att spelet är tillräckligt generallt och lätt att utöka. Genom en ständig dialog med dessa försöker man undvika detta problem.

3. Att spelet inte blir färdigt. Uppdragsgivarna har uttryckt att det är viktigare att det finns en generell, väldokumenterad och utbyggbar grund än att spelet blir helt färdigt. Vid tidsbrist kommer vi att koncentrera oss på att skapa en bra kommunikationslösning m.h.a. HLA och på att implementera serverns mest kritiska delar. På detta sätt löser vi de implementationsdelar där uppdragsgivarna har uttryckt att de har sämst kunskaper och ger dem därmed en bra grund för fortsatt arbete.

# A Internetrefereser

Gruppen har ett e-postalias som är `pr01-krigsspel@nada.kth.se`. Projektledarna Andreas Enblom och Henrik Hägerström nås via e-post på `d98-aen@nada.kth.se` resp. `d98-hha@nada.kth.se`.

Grupen har en hemsida på `http://www.student.nada.kth.se/~d98-jas/prupwiki/` där i stort sett all skriftligt dokumentation hamnar förr eller senare. Hemsidan använder sig av WikiWikiWeb-teknik för att alla lätt ska kunna skapa specifikationer och andra dokument tillsammans.