Kravspecifikation

Redaktör: Johan Eliasson

Version 1.0

I ett större projekt är projektgruppen i högsta grad inblandade i att skriva kravspecifikationen. Eftersom detta är en första kontakt med en projektstyrningsmodell och ett mindre projekt så anses kravspecifikationen vara behandlad och klar. Eventuella omformuleringar/tolkningar av krav måste dock dokumenteras i projektrapporten.

Innehåll

Dokumenthistorik

version	datum	utförda förändringar	utförda av	granskad
1.0	2009-03-26	Första versionen	Johan Eliasson	

1 Inledning

Projektet går ut på att skapa en fungerande routerprogramvara som gör att den fungerar tillfredsställande tillsammans med simuleringsprogramvaran SimNet. Simuleringsprogramvaran används eftersom tillverkarna inte vill använda routern i skarpt läge förrän programvaran är färdigutvecklad.

I detta dokument beskrivs alla krav med en tabellrad enligt nedan. Kravnummer är löpande genom hela dokumentet, i kolumn 1. Kravnumren låses till den numrering som gäller för den godkända kravspecifikationen. I kolumn 2 finns lydelsen av kravtexten och i kolumn 3 finns kravets prioritet angiven. Krav med prioritet 1 skall levereras i projektet, krav med prioritet 2 skall levereras om det ryms inom projektets tid- och kostnadsbudget. Krav med prioritet 3 kan eventuellt behöva levereras, dock endast om frågan initieras av beställaren.

Normalt finns också en kolumn i tabellen som används för hänvisning till beslut och/eller orsak när ett krav förändras. Denna är utesluten i detta dokument eftersom eventuella omformuleringar/tolkningar av krav måste godkännas av kontaktpersonerna (handledarna) och noga dokumenteras i projektrapporten.

Krav nr x	**Kravtext för krav nr X**	**prioritet**

1.1 Parter

Beställare är Johan Eliasson, Institutionen för datavetenskap, Umeå Universitet, 90187 Umeå.

Kontaktpersoner under projektets gång är

- Lucas Lindström (lucasl@cs.umu.se), rum D418 i MIT-huset, plan 4
- Tor Sterner-Johansson (tors@cs.umu.se), rum D415 i MIT-huset, plan 4
- Mikael Öhman (mikaelo@cs.umu.se), rum C225 i MIT-huset, plan 2

Projektet utförs av utsedda grupper av studenter som läser kursen Datastrukturer och algoritmer, 5DV043 våren 2009 vid för datavetenskap, Umeå Universitet, 90187 Umeå.

1.2 Syfte och Mål

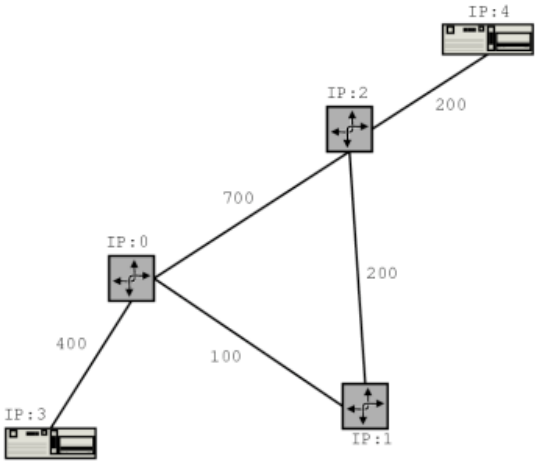
Projektets syfte är att skapa mjukvara till en nyutvecklad router. Målet är att skapa routerprogramvara som fungerar tillfredsställande tillsammans med simuleringsprogramvaran SimNet. Simuleringsprogramvaran används eftersom tillverkarna inte vill använda routern i skarpt läge förrän programvaran är färdigutvecklad.

2 Översikt av systemet

Ett datornätverk är uppbyggt av bl.a. flera knutpunkter (noder) som fungerar som trafikdirigerare. En sådan knutpunkt kallas router. En host (tex. en persondator eller webbserver) kan skicka och ta emot data (kallas också meddelanden eller paket och i SimNet:s miljö kallas de Datagram). Ett nätverk innehåller flera ”host”:ar och dessa är sammankopplade via minst en router.

En routers främsta uppgift är att styra hur paket/meddelanden/Datagram skickas i nätet. När till exempel host:en med IP-adress 3 (se figur 1) skickar ett paket till host:en med IP-adress 4, skickas paketet först till routern med IP-adress 0 som i sin tur avgör vilken router den skall skicka vidare paketet till. Skillnaden mellan en host och en router är att en host bara har en länk kopplad till sig medan en router har många länkar inkopplade. Det är också enbart hostar som skickar meddelanden (till andra hostar).

Vilken väg som ett paket färdas avgörs av tidsåtgången som i sin tur beror på länkens fördröjning (propagation delay) och den kräsne användaren vill ju givetvis att paketet skall färdas den snabbaste vägen. För att detta skall fungera behöver routern kunna beräkna den optimala vägen en del av projektet består av att utföra denna uppgift med hjälp av Dijkstras algoritm. Informationen om den kortaste vägen från en router till alla noder i nätverket sparas i en så kallad routingtabell.



Figur 1: Exempel på hur ett enkelt nätverk kan se ut

Datornätverket är ett dynamiskt nät där fördröjningen i en länk kan förändras. Dessutom kan länkar gå ned helt och hållet (någon banarbetare på stambanan råkar gräva av en kabel). Detta är faktorer som en router måste ta hänsyn till och därför måste den kunna räkna om sin routingtabell.

Uppgiften är att implementera klassen Router och de hjälpklasser som behövs, (alla klasser i SimNet är givna och ska inte skrivas av er).

2.1 Ingående delsystem

Eftersom routern ska fungera tillsammans med en existerande simuleringsprogramvara så kommer denna kravspecifikation att behandla routern som ett eget delsystem. Det krävs flera olika datatyper (graf, kö/prioritetskö och tabell) för att man ska kunna implementera en Router. Var och en av dessa kommer också att utgöra ett delsystem som beskrivs i de kommande kapitlen.

2.2 SimNet

SimNet™ tar en implementation av en Router och simulerar dess beteende i ett nätverk. För att kunna genomföra simuleringen krävs förutom implementationen av routern också två textfiler som beskriver nätverkets uppbyggnad samt hur/vilken information som ska skickas i nätverket. Notera att ni inte ska skriva någon kod i SimNet utan den är given. Däremot krävs en förståelse av vad som händer i SimNet för att kunna använda det. Detaljerad information om de ingående klasserna samt hur man använder SimNet och startar upp en simulering finns på webbadressen http://www.cs.umu.se/kurser/5DV043/VT09/ou/projekt/simnet_java.html.

2.3 Generella krav på hela systemet

Det övergripande målet och kravet på projektet är att programvaran ska fungera felfritt tillsammans med den givna simuleringsmiljön SimNet.

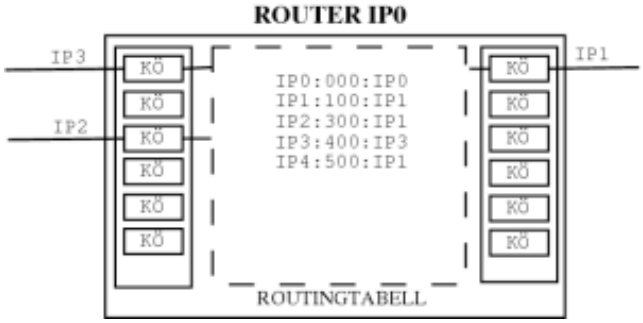
Krav nr 1	**Routerprogramvaran ska fungera felfritt tillsammans med den givna simuleringsmiljön SimNet.**	**1**

3 Router

Detta delsystem består av klassen `Router`. Klassen representerar en router i nätverket och för att klassen skall kunna integreras i simuleringsmiljön behöver den följa en specificerad gränsyta. Detta görs genom att implementera gränssnittet `RouterInterface` som beskrivs i avsnitt 3.2. Interfacet `RouterInterface.java` finns i distributionen av SimNet.

3.1 Routerns uppbyggnad

En bild på hur en router är uppbyggd finns i Figur 3. Den har ett antal fysiska kopplingar/länkar till andra noder i nätverket. Dessa kopplingar kallas **portar** och det finns maximalt 24 portar i en router. Varje port har en in- och en utkö. Dessa köer har en begränsad lagringskapacitet som anges av Routerns attribut `bufSize`. Anledningen till att vi har en maximal buffertstorlek är att vi vill kunna simulera "blockeringar" i nätet på grund av överfulla portar och många paket i nätet.



Figur 2: En skiss på routerns uppbyggnad med portar, köer och routingtabeller.

Krav nr 2	**Antalet portar i en Router ska vara begränsat till 24**	**1**
Krav nr 3	**Varje port ska ha en in- och en utkö.**	**1**
Krav nr 4	**In- och utköerna i porten ska vara prioritetsköer.**	**3**
Krav nr 5	**In- och utköerna ska ha begränsad lagringskapacitet som ska anges vid konstruktionen av en router genom attributet `bufSize`.**	**1**
Krav nr 6	**Projektet måste diskutera hur fulla in- och utköer ska hanteras och noga dokumentera hanteringen.**	**1**

Routingtabellen dirigerar trafiken från routern och ser till att meddelandet tar kortast möjliga väg till sin destination. I routingtabellen i Figur 2 ser man till exempel att ett meddelande som kommit till router 0 på sin väg till router 4 ska fortsätta via router 1 för att nå kortast väg. Jämför detta med grafen i Figur 1!

För att kunna skapa en routingtabell som dirigerar paketet rätt väg behöver routern veta hur nätet ser ut. Den informationen skickas från SimNet till Routern med hjälp av ett `RIPDatagram` som innehåller ett RIP-meddelande. Detta meddelande kommer ej på en port utan metoden `generateTable()` blir anropad med ett RIPDatagram som argument. När kontakten kopplas in och ur (dvs. metoderna `connect` och `disconnect` anropas) så skickas också ett RIPDatagram från SimNet. Du behöver alltså inte räkna om routingtabellen i dessa metoder. Mer information om RIP-meddelanden finns i avsnitt RIPDatagram i SimNet-beskrivningen (http://www.cs.umu.se/kurser/5DV043/VT09/ou/projekt/simnet_java.html) där en algoritm för hur man tolkar ett RIP-meddelande också ges.

Krav nr 7	**Routerklassens `generateTable` ska hantera RIPDatagram enligt den algoritm som ges i dokumentationen av SimNet.**	**1**

I varje tidssteg i simuleringen körs en uppdateringsmetod (dvs. SimNet anropar metoden) i routern som heter `processRouting()`. Den gör följande (pseudokod):

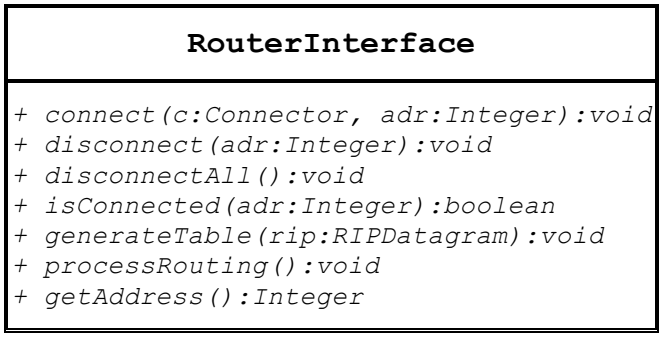
```
1. Upprepa för alla portar i routern:
   1.1. Skicka ev. paket som finns i utkön (om inte länken är upptagen)
   1.2. Om det finns paket på väg in till porten
        1.2.1 Lägg dem i inkön.
2. Upprepa för alla portar i routern:
   2.1. Så länge det finns data i inkön
        2.1.1. Använd routingtabellen för att finna vilken utport datat
               ska skickas till
        2.1.2. Om utportens utkö inte är full
               2.1.2.1 lägg datat i utportens utkö
```

Uppdateringsmetoden utför allt den kan göra i ett tidsteg, dvs alla paket som finns på inportarna dirigeras i detta steg och läggs till i utportens kö. Observera att datagrammet ej läggs ut på länken förrän vid nästa anrop av uppdateringsmetoden.

Krav nr 8	**Routerklassen måste i metoden `processRouting` hantera hämtande och skickande av meddelanden via en länk på det sätt som anges under rubriken Connector i SimNet-dokumentationen.**	**1**
Krav nr 9	**Varje meddelande måste i metoden `processRouting` stämplas på det sätt som anges under rubriken Datagram i SimNet-dokumentationen.**	**1**

3.2 Gränssnitt

Gränssnittet för Routern beskrivs i figuren nedan samt i tabellen på nästa sida. RouterInterface.java finns medlagd i distributionen av Simnet och kan laddas ned från webbsidan http://www.cs.umu.se/kurser/5DV043/VT09/ou/projekt/simnet_java.html).



Figur 3: RouterInterface

Signatur	Beskrivning
void connect(Connector c, Integer adr)	Koppla in en kontakt (Connector) i routern.
void disconnect(Integer adr)	Koppla ifrån en redan ansluten router.
void disconnectAll()	Kopplar ifrån alla anslutna routrar.
boolean isConnected(Integer adr)	Testar om routern är ansluten till en annan specifik router. Med andra ord, testar om routern har en port som är direkt ansluten till en annan specifik router med adressen adr
void generateTable(RIPDatagram rip)	Generera routingtabellen utifrån en given graf specificerad i ett RIP-meddelande. Anropas när nätet byggs upp första gången eller när nätet förändras på något sätt. Här byggs en graf upp och utifrån denna skapas sedan en routingtabell.
void processRouting()	Routerns uppdateringsmetod. Se pseudokoden nedan för vad som ska hända här.
Integer getAddress()	Hämta routerns adress.
och en konstruktor som ser ut på följande sätt:	
Router(Integer address, String switchFabric, int bufSize)	Konstruktorns signatur.

Er routerklass måste också importera SimNet-filerna. Detta görs genom att skriva följande överst i Router-klassen: `import se.umu.simnet.*;`

Krav nr 10	**Routerklassen ska implementera det givna gränssnittet RouterInterface.**	1

4 Graf

För att kunna använda Dijkstras kortaste-vägen algoritm måste routern känna till nätverket den ingår i. Detta nätverk representeras av en oriktad graf. Implementera den utifrån följande gränsyta (klassen Node får du själv bestämma vad den ska innehålla men det minimala är nodens adress som är av typen Integer):

GraphInterface

```
+ insertNode(n:Node):void
+ insertEdge(src:Node, dest:Node, weight:int):void
+ isEmpty():boolean
+ hasNoEdges():boolean
+ neighbours(n:Node):Vector
+ getNodes():Vector
+ getWeight(src:Node, dest:Node):int
+ deleteNode(n:Node):void
+ deleteEdge(src:Node, dest:Node):void
+ setWeight(src:Node, dest:Node, weight:int):void
```

Signatur	Beskrivning
void insertNode(Node n)	Sätter in noden n i grafen
void insertEdge(Node src, Node dest, int weight);	Sätter in en kant med vikten weight i grafen mellan noderna src och dest. Det förutsätts att noderna som anges ingår i grafen.
boolean isEmpty();	Testar om grafen är tom, dvs saknar noder.
boolean hasNoEdges();	Testar om grafen saknar kanter.
Vector neighbours(Node n);	Mängden av noder som är granne till noden n.
Vector getNodes();	Mängden av alla noder i grafen
int getWeight(Node src, Node dest);	Vikten på en kant mellan nod src och dest.
void deleteNode(Node n);	Tar bort en nod ur grafen.
void deleteEdge(Node src, Node dest);	Tar bort en kant mellan nod src och dest ur grafen.
void setWeight(Node src, Node dest, int weight);	Ändra vikten på en kant mellan nod src och dest i grafen.

Krav nr 11	**Grafklassen ska implementera det givna gränssnittet.**	**1**
Krav nr 12	**Används en annan variant av Dijkstras kortaste vägen-algoritm än den som beskrivs på kursen ska detta diskuteras och motiveras i projektdokumentationen.**	**1**

5 Kö

Det behövs en kö dels för kortaste-vägen-algoritmen och dels för att kunna hantera portarna i routern. Notera att den version av Dijkstras algoritm som vi går igenom på föreläsningen använder sig av en prioritetskö (se också krav 12 ovan)! Projektet får själva avgöra om portarnas köer ska hanteras som vanliga köer eller som prioritets-köer. Gränsytorna för kön/köerna ska följa kursbokens gränsytor (se föreläsningsanteckningarna om ni saknar kursbok).

Krav nr 13	**Gränsytorna för kön/köerna ska följa kursboken.**	**1**
Krav nr 14	**Valet av kö i portarna ska dokumenteras och motiveras.**	**1**
Krav nr 15	**Om prioritetsköer används i portarna så måste de implementeras så att högt värde ger hög prioritet och tvärtom.**	**2**

6 Tabell

För representationen av routingtabellen krävs en tabell. Ni bör återanvända koden från OU2 och annars motivera noga i dokumentationen varför ni inte gör det. Lookuptiden i tabellen får ej överstiga 1000 millisekunder.

Krav nr 16	**Asymptotisk komplexitetsanalys för metoden `LookUp` i projektets tabell måste genomföras och dokumenteras.**	**1**
Krav nr 17	**Om man frångår sin implementation från OU2 ska detta noga motiveras. Valet mellan de tre varianterna i OU2 ska också motiveras.**	**1**

7 Testning av systemet

Det förutsätts att kontinuerlig testning av systemet sker. Var och en av delsystemen ska testas utförligt enskilt och inbyggt i det färdiga systemet.

Krav nr 18	**För var och en av delsystemen ska en testplan upprättas, test genomföras och resultat dokumenteras.**	**1**

8 Utvecklingsmetodik

Krav nr 19	**Programvaran ska utvecklas i Java och följa en god objektorienterad programmeringsmetodik.**	**1**

9 Ekonomi och resursbegränsningar

Projektet har inga ekonomiska resurser. Det innebär att ev. resor, utskriftskostnader, mm betalas av projektmedlemmarna själva. Om det föreligger behov för ekonomiskt bidrag, skall detta diskuteras med handledaren. Projektet har maximalt 100 timmar till sitt förfogande. Projektet skall drivas med ett mål att inte använda mer än 75 timmar (inklusive möten, handledning, stöd, förberedande och genomförande av presentation mm).

Krav nr 20	**Projektet skall genomföras med en ekonomisk budget om 0 kronor**	**1**
Krav nr 21	**Projektets maximalt tillåtna tid är 100 timmar för en grupp om 2 personer**	**1**
Krav nr 22	**Projektets mål är att inte använda mer än 75 timmar och redovisa i vecka 22.**	**2**
Krav nr 23	**Fördelningen i nedlagd tid mellan projektdeltagarna skall vara inom +/- 10% från medeltiden som deltagarna lagt ned.**	**2**

10 Leveranskrav och delleveranser

Alla dokument bör utformas med hjälp av (modifierade) mallar i LIPS-modellen. De modifierade mallarna kan hittas via kurshemsidan [1], där man även kan hitta länkar till ursprungliga mallar i LIPS-modellen. Det är dessutom viktigt att dokument får relevanta och spårbara namn, och att det är enkelt att se på dokumentet, till vilken projektgrupp det hör.

Krav nr 24	**Alla dokument skall ha ett enhetligt utseende (layout, typsnitt, etc.).**	**1**
Krav nr 25	**Alla dokument skall utformas med hjälp av (modifierade) mallar i LIPS-modellen, se kurshemsidan [1].**	**2**
Krav nr 26	**Gruppen skall göra en projektplan (som även innehåller tidplan, enligt nivå 1 i LIPS-modellen), som skall levereras senast 20/4 12.00.**	**1**
Krav nr 27	**Gruppen skall göra en gemensam muntlig redovisning av arbetet så långt 14/5 15.15-17**	**1**
Krav nr 28	**Gruppen ska lämna in en reviderad projektplan senast den 14/5 17.00**	**1**
Krav nr 29	**Gruppen skall lämna in en slutrapport som presenterar och analyserar resultaten från arbete. Ytterligare krav på rapporten finns specificerat i krav nr 30-43. Rapporten skall lämnas in helst under vecka 22 men absolut senast 3/6 12.00.**	**1**

11 Dokumentation

Här definieras krav på den dokumentation som lämnas in under projektets gång. Den dokumentation som det finns mallar till (tex projektplan) ska följa de anvisningar som ges i mallarna.

Dokument	Syfte	Målgrupp	Format/ media
Projektplan	Beskrivning av planeringen av projektet	Projektets medlemmar, kontaktpersoner och beställare	Pappersformat (doc-mall finns)
Tidplan	Planering och rapportering av tid i projektet	Projektets medlemmar, kontaktpersoner och beställare	Pappersformat (excel-mall finns)
Efterstudie	Analys och diskussion kring projektet och dess resultat	Kontaktpersoner och beställare	Pappersformat (doc-mall finns)
Slutrapport	Avrapportering av projektet och dess resultat	Första års student i datavetenskap som ej gått kursen eller hört talas om projektet tidigare.	Pappersformat

Slutrapporten ska innehålla följande

- En inledning som beskriver projektet och vad det går ut på. Beskrivningen måste skrivas på en sådan nivå att en första års student i datavetenskap som ej gått kursen eller hört talas om projektet tidigare förstår vad projektets syfte och mål är samt i stora drag få en översikt över vad som gjorts i projektet.

- En övergripande beskrivning av er programvara (inklusive ett klassdiagram) ska finnas i slutrapporten. En fullständing systembeskrivning över alla ingående klasser, metoder och attribut ska beskrivas med hjälp av en webbaserad API. Om alla kommentarer skrivs med Javadoc så kan man i BlueJ skapa html-filerna genom att i menyn ”Tools” använda kommandot ”Project documentation”. Ni som inte använder BlueJ hänvisas till informationen på sidan http://java.sun.com/j2se/javadoc/ eftersom det exakta tillvägagångssättet varierar beroende på vilken miljö man arbetar i.

- En redovisning av komplexitetsanalysen för LookUp i tabellen som används ska genomföras.

- Ett avsnitt där ni diskuterar kravspecifikationen och huruvida ni uppfyller alla krav i den. Lista alla krav som inte har uppfyllts och motivera noga varför. På samma sätt om något krav har modifierats under projektets gång ska även det diskuteras i rapporten.

- En diskussion kring begränsningar och möjlighet till utveckling av er lösning ska finnas med. Kanske delar av er kod kan optimeras och beskriv i så fall vad som borde optimeras.

- Ett avsnitt som beskriver hur ni testat programvaran under projektets gång. Utförliga och välkommenterade testkörningar av hela systemet för minst två fall ska finnas. Man ska enkelt kunna följa vad som testas och vad resultatet blev. Redovisa hur nätverket ser ut, vilka resultat ni borde få och varför, samt vilka resultat ni fick.

- Väl strukturerad, kommenterad och indenterad källkod utskriven i en icke-proportionerlig typsnitt (tex. courier) ska finnas med i slutrapporten.

- Källor ska refereras korrekt. Det ska finnas referenser till alla informationskällor som använts under projektet. Referenserna ska vara enhetligt utformade och följa riktlinjerna på sidorna
  http://www.ub.umu.se/global/biblref.htm eller
  http://www.cs.umu.se/education/examina/InternetCite/citera.htm

Krav nr 30	**Den dokumentation som det finns mallar till (tex projektplan) ska följa de anvisningar som ges i mallarna.**	**1**
Krav nr 31	**All dokumentation ska vara genomläst av båda medlemmarna i projektgruppen och stavningskontrollerad.**	**1**
Krav nr 32	**Slutrapporten ska kunna läsas helt fristående från projektdirektiv och kravspecifikation.**	**2**
Krav nr 33	**I slutdokumentet ska en beskrivning av projektet finnas som förklarar för läsare i den tänkta målgruppen projektets syfte och mål och i stora drag vad som gjorts i projektet.**	**1**
Krav nr 34	**En övergripande beskrivning av er programvara (inklusive ett klassdiagram) ska finnas i slutrapporten.**	**1**
Krav nr 35	**Den fullständiga systembeskrivningen ska göras som ett webbaserat API och redovisas i slutrapporten med en webbadress till API:et.**	**1**
Krav nr 36	**Den genomförda asymptotiska komplexitetsanalysen för metoden LookUp i projektets tabell ska redovisas och diskuteras.**	**1**
Krav nr 37	**Slutrapporten ska ha ett avsnitt där uppfyllelsen av kravspecifikationen diskuteras. Eventuella tolkningar och/eller godkända omformuleringar av krav ska finnas i dokumentationen.**	**1**
Krav nr 38	**Diskussioner kring lösning, vilka begränsningar och möjligheter finns. Finns utrymme för optimering?**	**2**
Krav nr 39	**Ett avsnitt i slutrapporten ska ägnas åt att beskriva de test av delsystem och det slutliga systemet som gjorts.**	**1**
Krav nr 40	**Noggrann genom gång av minst två testkörningar av det färdiga systemet ska redovisas.**	**1**
Krav nr 41	**Källkoden ska vara utskriven i ett icke-proportionellt typsnitt, t.ex. `courier`**	**1**
Krav nr 42	**Källkoden ska vara väl strukturerad, indenterad och kommenterad**	**1**
Krav nr 43	**Referenser ska finnas till de källor som använts under projektet. Referenserna ska vara enhetligt utformade och följa riktlinjerna på sidorna http://www.ub.umu.se/global/biblref.htm eller http://www.cs.umu.se/education/examina/InternetCite/citera.htm**	**1**

Referenser