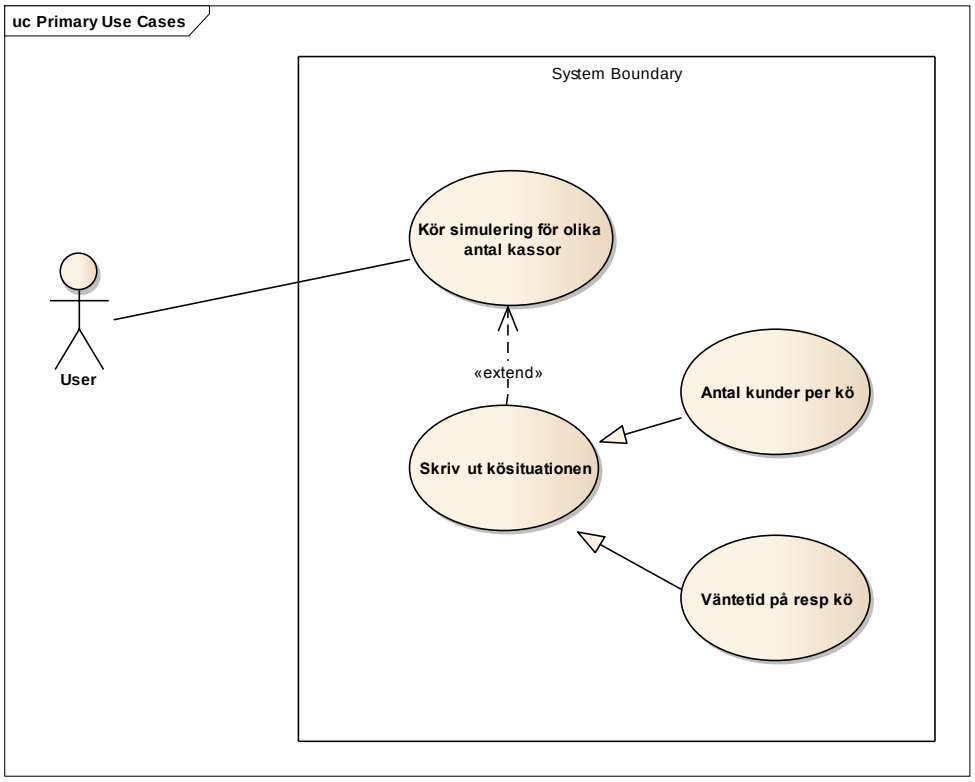
Enkel rapport för Inlämningsuppgift nr 6 - Henrik Axelsson, 750328-4999

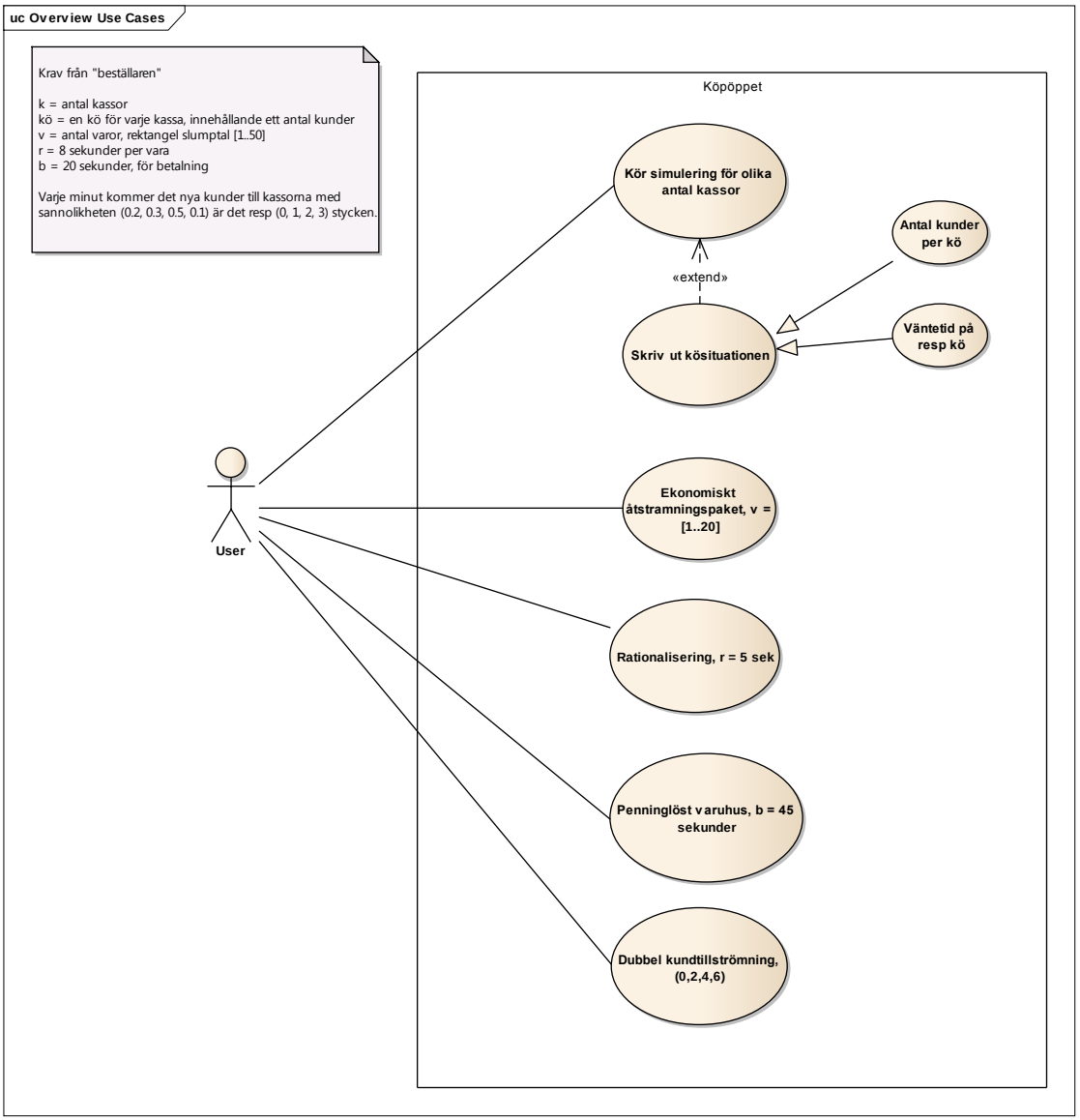
Kravanalys

Gjorde precis som förra gången en OO-analys med hjälp av Enterprise Architect som ”ritbräda” (i UML). Började med att analysera kraven och lägga upp de enklaste användarfallen, se Figur 1.



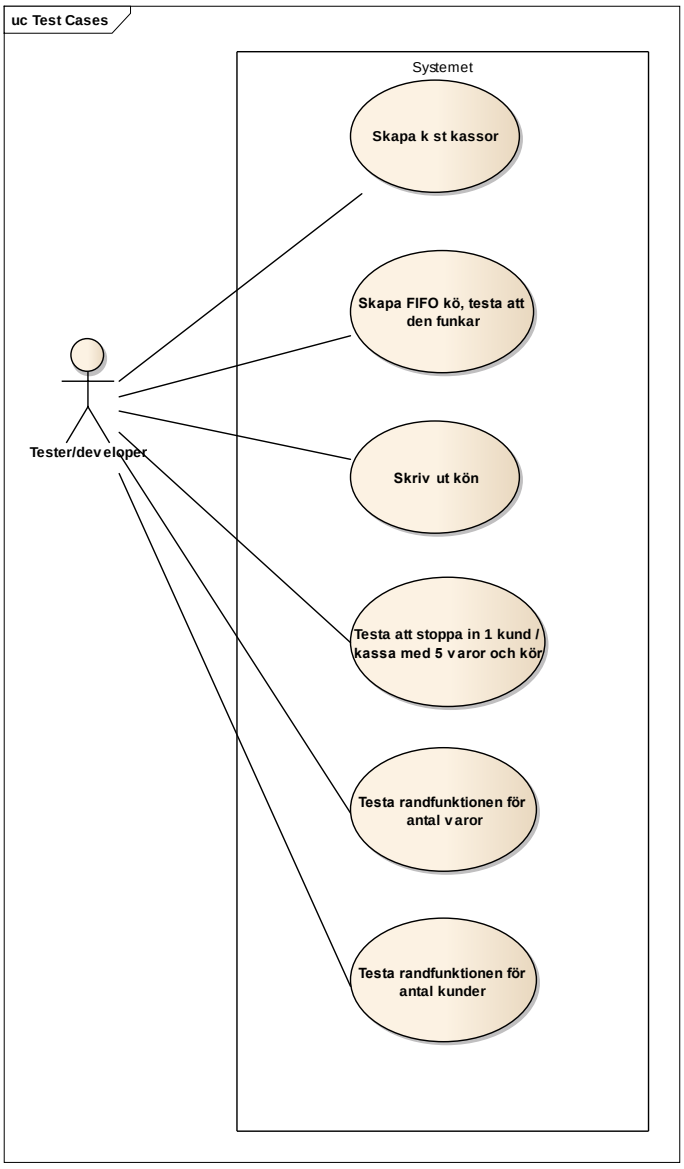
Figur 1. Use Case diagram.

Jag gjorde även en mer komplett vy som visade alla användarfallen enligt projektförslaget:



Figur 2. Mer komplett Use Case diagram.

Parallellt med OO-analysen arbetade jag igenom vilka test-case som jag behövde för att stegvis implementera och förfina programmet. Se exempel nedan i Figur 3.



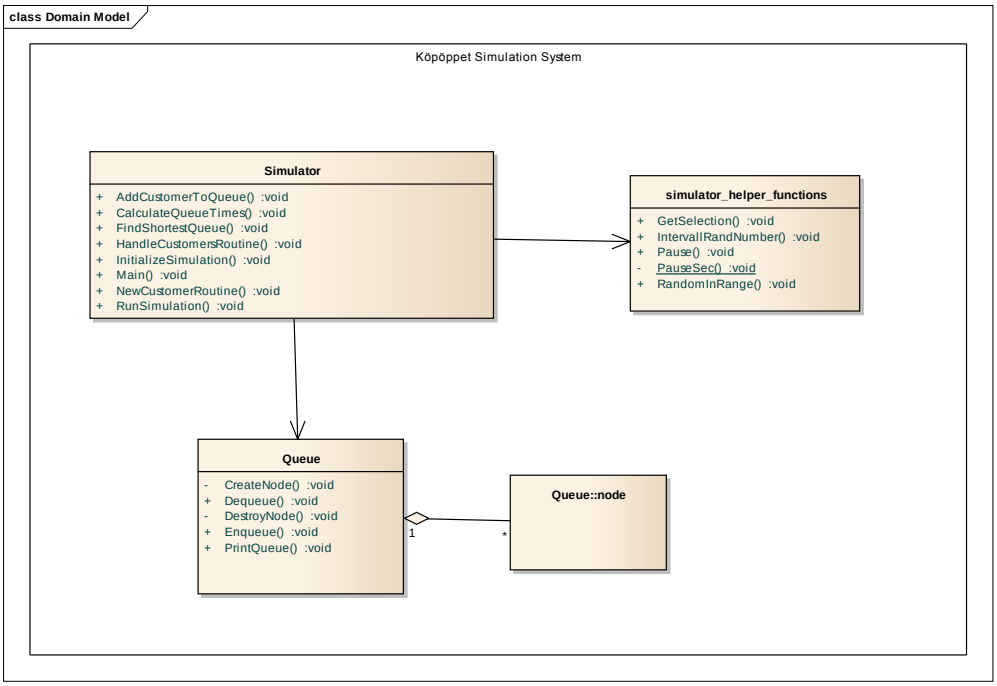
Figur 3. Exempel på testfall.

Systemanalys och design

Utifrån den fördefinierade uppgiften skapade jag ett klassdiagram med klasser enligt inlämningsuppgiften samt ett antal metoder och attribut för de respektive klasserna. Skapade även några hjälpmetoder. Detta fick jag senare revidera kraftigt då jag från början hade en annan struktur, men valde att förenkla det till det resultat som syns i Figur 4.

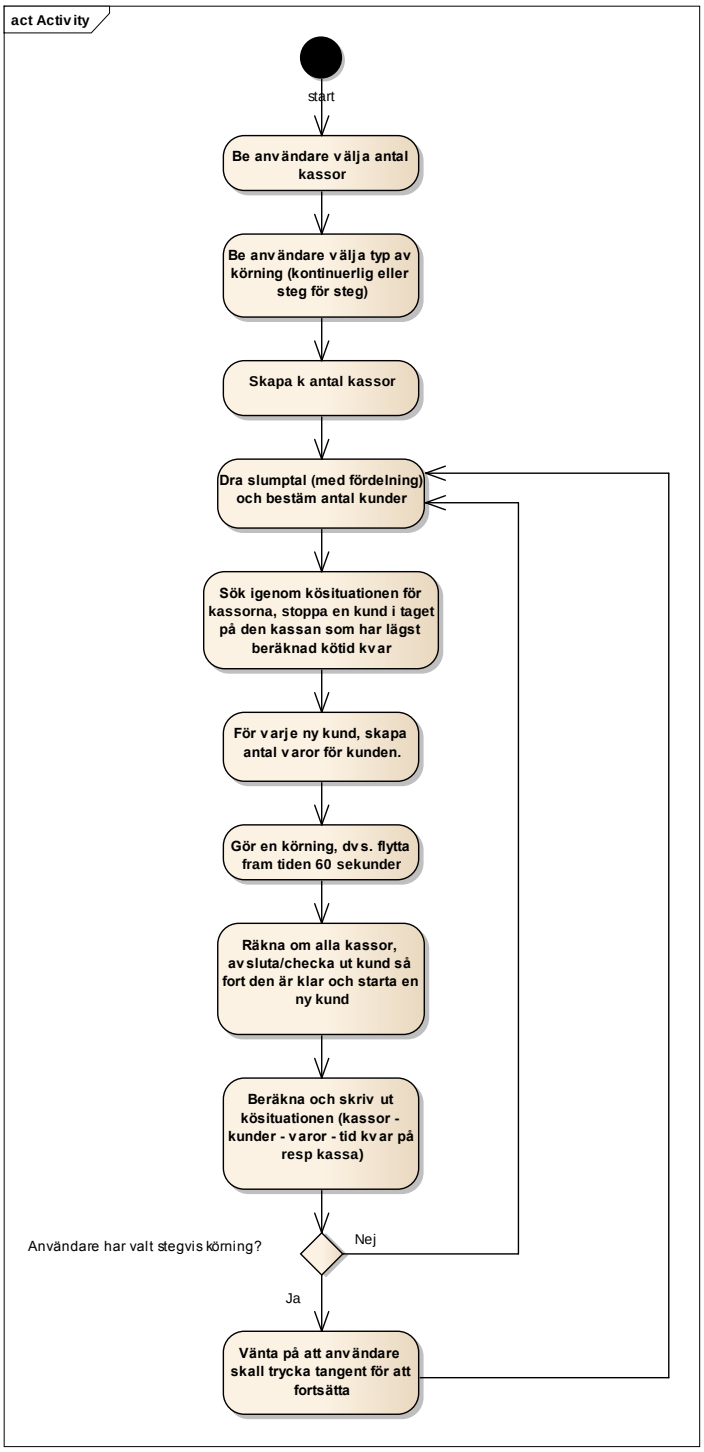
Första designbeslutet var om jag skulle använda C++ eller C, vilket landade i att använda C, av två anledningar: den ena för att detta är en C-kurs och den andra för att jag har mest nytta av C i mitt arbete.

Det hela landade i en simulatorkomponent, där även main-funktionen körs. Vissa hjälpfunktioner placerade jag i en separat header. Köfunktioner återanvände och förädlade jag från tidigare projekt och skrev om en del.



Figur 4. Översiktligt strukturdiagram för systemet.

Jag gjorde även ett aktivitetsdiagram för att visa ungefärligt ordningen av exekveringen som jag tänkt mig i systemet (inte på en detaljerad nivå). Se Figur 5.



Figur 5. Översiktligt strukturdiagram för systemet.

Implementation, problem längs vägen

De största problemen härrörde från skapandet av datastrukturen och att få det hela med pekare att fungera. Gjorde även en lärdom längs med vägen att det är lättare att deklarera variabler och strukturer i globalt scope, eftersom jag först ville ha arrayen med nodpekarelement dynamiskt bestämd till sin storlek. Då fick jag skicka med pekare in i alla underfunktioner och deras

underfunktioner och tappade bort mig på vägen. Valde därför att skapa en fast storlek på arrayen, det är endast 10 element och inget krav på att hålla nere minnesförbrukningen i detta system.

Andra strul på vägen var att helt enkelt hålla tungan rätt i munnen i alla detaljerna i implementationen. Använder mig mycket av utskrifter för att se vad programmet gör och var det hänger sig.

Flyttade även om funktionalitet en del och kommenterade bort vissa funktioner under utvecklingens gång. Jobbade i stort sett bottom-down med nedbrytning.

Körning av systemet
Jag testkörde systemet och dokumenterade resultaten i Excel. Körde inte så många gånger på varje, utan tittade mer på trender för att se hur köerna utvecklade sig.

Körde först igenom alla 16 parametervarianterna med 5 kassor och såg då vilka varianter som inte gav en acceptabel kösituation. Vissa fall tycktes ackumulera köer, medan andra knappt gjorde det alls.

Maximala antalet varor påverkar såklart resultatet mycket, vilket syns då 5 kassor räcker för alla fall, möjligen med reservation för det fallet som gulmarkerats i tabell 1.

Självklart påverkar det mycket med dubbelt antal tillströmmande kunder, men även rationaliseringen kunde påverka situationen avsevärt (jfr. exv. variant 1 och 5).

Provade att köra med 3 kassor och valde då fall som kunde tänkas ligga på gränsen att tippa över till en orimlig situation om antalet kassor minskades (gröna eller gula i fallet med 5 kassor). Mycket riktigt var det 3 av dessa fall som inte klarades med 3 kassor (alla hade dubbelt antal kunder tillströmmande).

Ökade även på antalet kassor till 7st, vilket löste situationen för två rödmarkerade och ett hult fall (jfrt med 5 kassor).

För de 4 fall som fortfarande var röda med 7 kassor körde jag simuleringen med 10 kassor. Resultatet blev att 2 till grönmarkerades.

Det värsta fallet, föga oväntat, är när max antal varor är 50, ingen rationalisering införts samt att kunden betalar med kreditkort och att kundtillströmningen är dubbel. Alla dessa parametrar pekar ju mot ökad behandlingstid samt fler kunder att behandla. I detta fallet verkar tiden halveras (grov uppskattning!) om man tar bort det penninglösa varuhuset och använder kontanter.

Avslutning
Systemet skulle kunna vidareutvecklas genom att automatisera körningarna samt få längre och större körningar för att slumptalsgeneratorn inte skall påverka resultaten. Det verkar vara en hel del svängning i dessa resultat (jag har dock testat generatorn med ett stort antal körningar och då blev fördelningen mycket nära det den skall bli).

/Henrik Axelsson

						5-10 körningar per styck
Variant	Kassor	Ekonomiskt	Rationalisering	Penninglöst	Dubbla kunder	Kötider efter 120 min
1	5	0	0	0	0	ca 11-30 minuter
2	5	0	0	0	1	ca 160-190 minuter
3	5	0	0	1	0	ca 24-60 minuter
4	5	0	0	1	1	ca 180-220 minuter
5	5	0	1	0	0	ca 0-2 minuter
6	5	0	1	0	1	ca 90-120 minuter
7	5	0	1	1	0	ca 0-3 minuter
8	5	0	1	1	1	ca 90-130 minuter
9	5	1	0	0	0	ca 0-1 minuter
10	5	1	0	0	1	ca 0-10 minuter
11	5	1	0	1	0	ca 0-1 minuter
12	5	1	0	1	1	ca 0-10 minuter
13	5	1	1	0	0	ca 0-1 minuter
14	5	1	1	0	1	ca 0-10 minuter
15	5	1	1	1	0	ca 0-1 minuter
16	5	1	1	1	1	ca 4-17 minuter
17	3	0	1	0	0	ca 0-1 minuter
18	3	0	1	1	0	ca 0-3 minuter
19	3	1	0	0	0	ca 0-1 minuter
20	3	1	0	0	1	ca 80-110 minuter
21	3	1	0	1	1	ca 70-90 minuter
22	3	1	1	0	0	ca 0-3 minuter
23	3	1	1	0	1	ca 70-90 minuter
24	3	1	1	1	0	ca 0-3 minuter
25	7	1	1	1	1	ca 0-1 minuter
26	7	0	0	0	0	ca 0-1 minuter
27	7	0	0	0	1	ca 60-120 minuter
28	7	0	0	1	0	ca 0-1 minuter
29	7	0	0	1	1	ca 90-120 minuter
30	7	0	1	0	1	ca 30-50 minuter
31	7	0	1	1	1	ca 30-50 minuter
32	10	0	0	0	1	ca 15-25 minuter
33	10	0	0	1	1	ca 35-60 minuter
34	10	0	1	0	1	ca 0-3 minuter
35	10	0	1	1	1	ca 0-6 minuter

Tabell 1. Ungefärliga resultat av körningen.