Analys av callcentersimulering

SEBASTIAN HEW
och TONI HUOTARI

KTH Datavetenskap
och kommunikation

Examensarbete

Analys av callcentersimulering

S E B A S T I A N H E W
o c h T O N I H U O T A R I

Examensarbete i datalogi om 15 högskolepoäng
vid Programmet för datateknik
Kungliga Tekniska Högskolan år 2011
Handledare på CSC var Mads Dam
Examinator var Mads Dam

URL: www.csc.kth.se/utbildning/kandidatexjobb/datateknik/2011/
hew_sebastian_OCH_huotari_toni_K11002.pdf

Kungliga tekniska högskolan
Skolan för datavetenskap och kommunikation

KTH CSC
100 44 Stockholm

URL: www.kth.se/csc

Analys av callcentersimulering

Sammanfattning

Ett callcenter är ett servicesystem som många företag har för att via telefon erbjuda hjälp till sina kunder. Syftet med den här rapporten var att utveckla och analysera en enkel simulering av ett callcenter. Analysen utfördes genom att jämföra prestandaskillnader av olika testfall. Detta genom att jämföra servicenivån genom påverkan av olika parametrar så som betjäningstider och ankomstintensiteter av ärenden samt uppdelning av arbetskraft. Testerna pekade på en betydlig prestandaskillnad i vissa fall.

Analysis of Call Center Simulation

Abstract

Call centers are telephone services that companies have to offer assistance to their customers. The purpose of this paper was to produce and analyze a simple simulation of a call center. The analysis was done by comparing the performance of different test cases of call centers. The main parameters to be analyzed were service time and arrival rate of calls as well as the staffing of the labor. The tests indicated a significant difference in performance of the call centers in some cases.

Fördelning av arbete

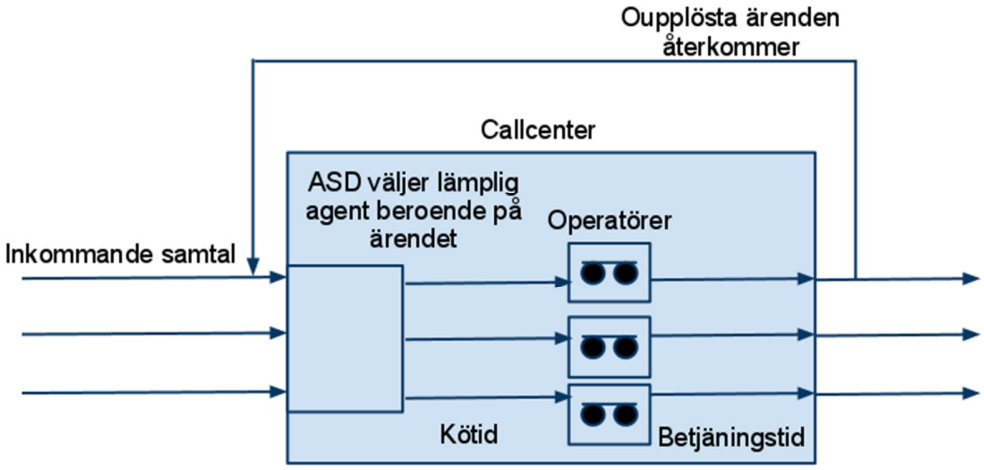
Största delen av arbetet utfördes tillsammans av båda författarna, exempelvis var båda lika delaktiga i skrivningen av rapporten. Toni var huvudansvarig för strukturen i hur arbetet skulle utföras och strukturen av rapporten. Toni var även mer delaktig i skapandet av testfallen och var mer ansvarig när det kom till teori om callcenter. Sebastian var mer ansvarig för att sätta sig in i simuleringsteori (Monte Carlo-metoden och sannolikhetsfördelningar) och var även huvudansvarig för utvecklingen av simulatorn. Sebastian utförde även körningarna av de flesta experimenten.

Innehållsförteckning

1. Introduktion

1.1 Bakgrund

Många företag har idag ett så kallat callcenter. Det är ett servicesystem som via telefon kan erbjuda hjälp till kunderna. Callcenter kan till exempel erbjuda support för företagets produkter, andra tjänster som kan erbjudas inkluderar faktura- och produktfrågor. Ett callcenter består av ett antal anställda, så kallade operatörer, som sköter samtalen. Vanligt förekommande med callcenter är att de tar hand om flera olika områden och callcentret måste i sådana fall kunna koppla kunden till en operatör med rätt kompetens. Exempelvis om man ska ringa sin telefonoperatör känner nog de flesta igen att man först får svara på några frågor, såsom om man redan är kund, om det gäller telefoni, bredband eller TV, för att sedan kopplas till lämplig operatör. Det här automatiserade systemet sköts av en så kallad ASD (automatisk samtalsdistribuerare). Det intressanta är att operatörerna ofta har kompetens inom flera olika områden och detta möjliggör mångsidiga callcenter. Med sådana callcenter dyker det upp optimeringsmöjligheter för fördelning av arbetskraft. Callcenter som bara tar hand om ett område eller flera områden kommer att kallas för enkla respektive mångsidiga callcenter.



Figur 1: Modell över mångsidigt callcenter

Figur 1 åskådliggör en modell över ett simpelt mångsidigt callcenter. Samtal kommer in med en viss intensitet och lämplig operatör väljs ut med hjälp av ASD. Därefter kommer kunden hamna i en kö om ingen lämplig operatör finns tillgänglig. När kunden har kommit fram till en operatör så betjänas de

under en viss betjäningstid. Kunden kommer i vissa fall behöva återkomma då alla ärenden inte löses under första samtalet.

1.2 Termer

När man pratar om effektiviteten hos ett callcenter finns det ett antal termer att ta hänsyn till. Till resultatet kommer fokus att ligga på tre olika termer; SF (servicefaktor), GV (genomsnittsväntetiden) samt SFK (sannolikhet för kö). Till servicefaktorn hör även AV (acceptabel väntetid). Man brukar säga att man ska svara en viss procent av alla samtal inom en viss väntetid. Industristandarden är att 80% av alla samtal ska svaras inom 20 sekunder (Koole, 2007, s. 10), vilket brukas benämnas 80/20 SF, där 20 sekunder av kunden betraktas som en acceptabel väntetid. GV anger genomsnittsväntetiden för hur länge en kund behöver vänta innan en operatör kan ta hand om samtalet. SFK är sannolikheten för att när man ringer inte direkt får hjälp med sitt ärende, utan måste köa en viss tid.

1.3 Ordlista

Arbetsgrupp	En grupp med operatörer där samtliga operatörer besitter samma kompetenser.
ASD	Automatisk samtalsdistribuerare, automatisk tjänst som tilldelar alla inkommande samtal till lämpliga operatörer beroende på tillgänglighet samt kunskapsområde.
AV	Acceptabla väntetiden, den acceptabla tiden i kön innan betjäning.
Beta	β, beteckning på betjäningstiden.
Callcenter	Center som tar hand om bland annat supportrelaterade frågor dit kunden kan ringa.
GV	Genomsnittsväntetiden, genomsnittstiden för hur länge kunden får stå i kö.
Inaktivitetstilldelning	Metod för att välja den operatör som varit sysslolös den längsta perioden sen det senaste samtalet.
Kompetens	Se område.

Lambda	λ, beteckning på ankomstintensiteten.
Område	Kompetens som en operatör besitter för en viss samtalstyp.
Operatör	En operatör på callcentret som betjänar kunder. Kan ha kompetens inom ett eller flera områden.
Samtal	Se ärende.
Samtalstyp	Typen av ett samtal, har ett motsvarande område.
Servicenivå	Mått på callcenters service, kan mätas upp på olika sätt t.ex väntetid, SF.
SF	Servicefaktor, hur många procent av ett samtal som svaras inom AV.
SFK	Sannolikhet för kö, sannolikheten att en kund som ringer till callcentret hamnar i kön och behöver vänta för att bli betjänad.
Ärende	Ett samtal med en viss samtalstyp. Samtalet kan bara tas hand om av en operatörer som har kompetens inom motsvarande område för samtalstypen.

1.4 Erlang C

En klassisk modell för beräkning av callcenter-matematik är Erlang C-modellen, vilken har fått sitt namn av den danska matematikern Agner Krarup Erlang. Ett callcenter har en viss uppringningsintensitet, det vill säga i snitt hur många inkommande samtal man får under en viss tidsperiod, uppringningsintensiteten betecknas λ. Den genomsnittliga betjäningstiden, hur lång tid det i snitt tar för en operatör att behandla en kund, betecknas β. Trafiktätheten a definieras som $a = \lambda * \beta$ och har enheten Erlang. Antalet operatörer som finns i systemet betecknas med s.

Erlang C-formeln används för att beräkna SFK givet värden på antal operatörer i systemet och trafiktätheten . När man sedan har räknat ut SFK kan det värdet sedan användas för att räkna ut SF och GV.

$$SFK = \frac{a^s}{(s-1)!\,(s-a)}\left[\sum_{j=0}^{s-1}\frac{a^j}{j!} + \frac{a^s}{(s-1)!\,(s-a)}\right]^{-1}$$

Formel 1: Erlang C-formeln

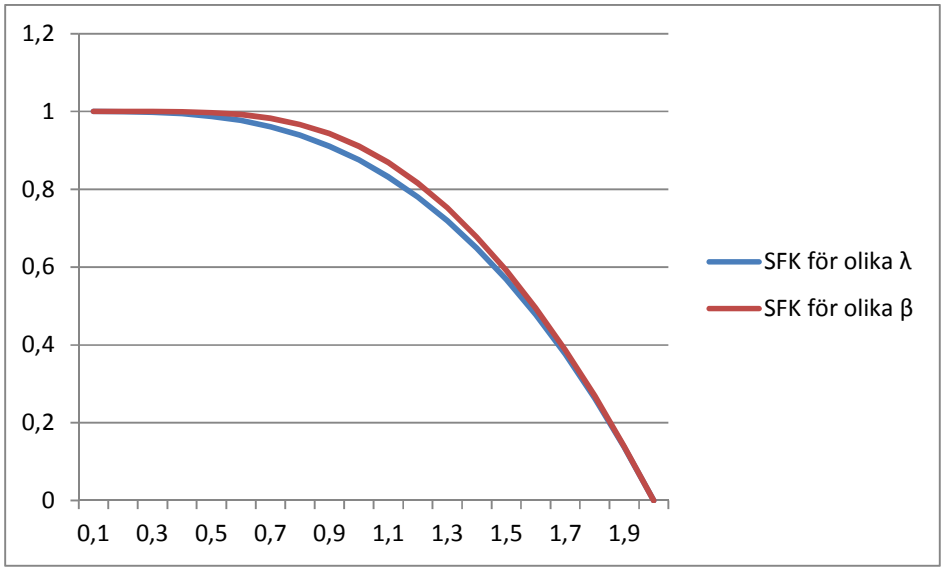
$$SF = 1 - SFK * e^{-(s-a)\frac{AV}{\beta}}$$

Formel 2: Uträkning av servicefaktor

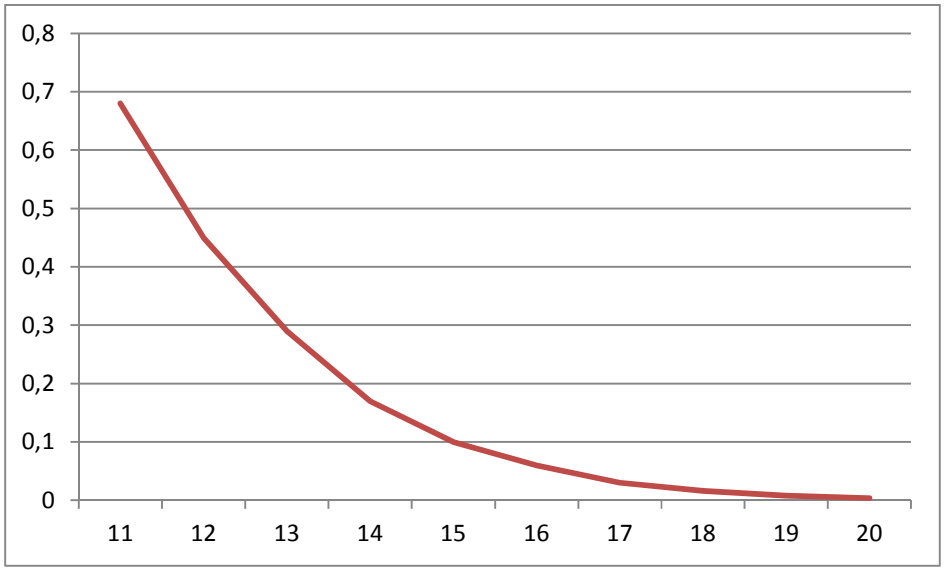
$$GV = \frac{SFK * \beta}{s-a}$$

Formel 3: Uträkning av den genomsnittliga väntetiden

Formel 1 (Koole, 2007, s.65) är Erlang C-formeln och beräknar SFK, formel 2 (Koole, 2007, s. 16) och formel 3 (Koole, 2007, s. 17) visar hur man använder Erlang C-formeln för att beräkna SF och GV. En förutsättning för att formlerna ska fungera är att trafiktätheten a är mindre än antalet operatörer. Rent praktiskt skulle det resultera i en samtalskö som bara blir större och större utan att operatörerna hinner med att ta hand om kunderna. I formlerna är det tydligast i fallet där trafiktätheten är lika stor som antalet operatörer, då man får division med noll både i formel 1 och 3. I fallet där a är större än s får man vissa negativa värden vilket ger oönskade resultat. Till exempel då λ=3, β=3 och s=6 fås $\frac{a^s}{(s-1)!(s-a)}$=-1476,225 som resulterar i SFK=2,55, vilket uppenbart är ett felaktigt resultat då SFK är en sannolikhet och därmed bör vara ett tal mellan 0 och 1. En annan begränsning är att Erlang C-formeln inte tillåter flera olika samtalstyper, därav kan formeln bara användas för att beskriva enkla callcenters (Koole, 2007, s. 67).



Figur 2: Ändring av SFK för olika värden på λ och β, den undre axeln är λ/β och den övre visar hur SFK påverkas.



Figur 3: Ändring av SFK för olika värden på s, den undre axeln är s och den övre visar hur SFK påverkas

Figur 2 och 3 visar hur värdet på SFK uträknat med Erlang C-formeln ändras när man ändrar på de olika parametrarna. I figur 2 är λ eller β (beroende på vilken av de som varierar) satt till 2 samtal/minut respektive 2 minuter. Antalet operatörer är 4 för båda fallen. X-axeln visar hur λ/β ändras från 0,1 till 2,0 och y-axeln visar hur SFK påverkas. I figur 3 är λ=2 och β=5. X-axeln

visar hur många operatörer som används och utgår från 11 och går upp till 20. Y-axeln visar hur SFK påverkas av de ändringarna.

1.5 Fördelningar

När man inom den matematiska statistiken pratar om köteori, det vill säga system där kunder anländer och kan betjänas av ett antal betjäningsstationer, är det främst två olika fördelningar som är aktuella. Den första fördelningen är poissonfördelningen, vilken beskriver inträffandet av oberoende händelser (Weisstein, n.d.).

$$f(k,\lambda) = \frac{\lambda^k e^{-\lambda}}{k!}$$

Formel 4: Poissonfördelning

Formel 4 visar hur man för ett givet väntevärde λ (observera att λ inte är ankomstintensiteten i den här formeln) på antalet inträffade händelser under ett visst intervall (kan vara tid, avstånd, volym etc) kan beräkna sannolikheten att just k inträffade händelser sker, där k är ett icke-negativt heltal. Det här gör poissonfördelningen till en väldigt bra fördelning att använda till att räkna ut hur många nya samtal som inkommer under en viss tidsperiod.

Exponentialfördelningen är likt poissonfördelningen minneslös och används för att beräkna tiden mellan det att två händelser inträffar (Weissstein, n.d.). Exponentialfördelningen är därför användbar för att beräkna hur lång tid ett visst samtal kommer ta.

2. Problembeskrivning

2.1 Mål

Syftet med rapporten är att utveckla ett program som kan simulera hur ett callcenter fungerar. Med hjälp av programmet ska olika typer av callcenter undersökas, bland annat genom att jämföra servicenivåer från olika körningar mot deterministiska värden uträknade med hjälp av Erlang C-formeln. Resultaten från simulatorn ska ritas ut i grafer för att enkelt kunna överblicka utvecklingen av de genomsnittliga värderna. Fokus kommer ligga på beräkningar av servicefaktorn. Anledningen är att servicefaktorn är ett relevant sätt att räkna ut servicenivån på och dessutom enkel att arbeta med. Förutom servicefaktorn kommer programmet även räkna ut de genomsnittliga väntetiderna samt sannolikheten att en kund hamnar i kö. Första delen av callcenteranalysen kommer att fokusera på simuleringar av enkla callcenter då det är lättare att jämföra resultaten från dessa mot deterministiska värden för att undersöka simuleringens korrekthet. Andra delen av analysen kommer omfatta påverkan från olika parametrar. I simulatorn vill vi kunna ändra på parametrar som intensiteter för inkommande samtal, förväntade samtalslängder, operatörers kompetenser och operatörer per område.

2.2 Utvecklingsmiljö

Angående de tekniska detaljerna ska utvecklingsmiljön Visual Studio 2010 med .NET4 användas och koden kommer att skrivas i språket C#. Skälet till valet av den plattformen är för att det är väldigt enkelt att skapa grafiska gränssnitt i Visual Studio. Programmeringsspråket C# väljs på grund av det omfattande utbud av biblioteksfunktioner som finns tillgängliga.

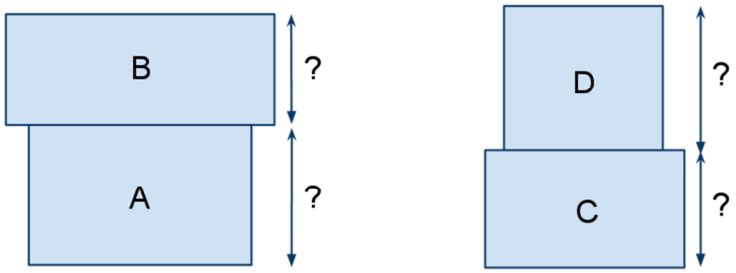
3. Simulering

Simuleringsprogrammet går på diskret tid och inte kontinuerlig, det innebär att programmet har en loop och där det efter varje iteration har gått en viss tid. Varje iteration är 20 sekunder, vilket är valt då AV är 20 sekunder och när man beräknar SF är det fördelaktigt att AV är ett jämnt antal iterationer.

Det första som sker i varje iteration är att data för operatörer och samtal uppdateras, bland annat ökas inaktivitetstiden för operatörer utan samtal och väntetiden för samtal i kön. Nästa steg är att ta emot nya samtal till systemet, man använder först en poissonmetod för att fastställa hur många nya samtal som ska inkomma, och sedan skapar man dessa samtal där var och en får en egen betjäningstid som är exponentialfördelad. Samtalen läggs i kö direkt när de kommer in till systemet. Nästa steg är att försöka tilldela samtal till en ledig operatör, vilket sker med en algoritm som beskrivs i kapitel 3.3. Samtal som inte lyckades bli tilldelade en operatör ligger kvar i kön till nästa iteration där man försöker tilldela dem en operatör igen. Det sista som inträffar är ta fram värden på SF, GV samt SFK. Detta görs inte deterministiskt med Erlang C-formeln utan genom att man kollar på den data som finns i systemet, till exempel för SF räknar man ihop hur många samtal som svarades innan AV och delar det med det antal samtal man har svarat på. Resultaten sparas och skickas när simuleringen är färdig till en klass som ritar upp grafer för de olika uträkningarna. Grafritandet sköts med ett tredjepartsbibliotek kallat ZedGraph som är utvecklat av J. Champion (2007).

3.1 Monte Carlo

Monte Carlo-metoden är en metod som kan användas för att approximera värden med hjälp av slumpmässigt indata och är även vanligt förekommande vid simulering av data. Generellt kan man säga att Monte Carlo-metoden går till som så att man först definierar ett intervall på sin indata, sedan genererar man slumpmässigt fram sin indata enligt någon sannolikhetsfördelning, för att slutligen utföra någon typ av beräkning och få fram ett resultat (Wittwer, 2004). Som ett exempel kan man tänka sig fyra byggklossar; A, B, C och D. Man har inga mått på hur stora klossarna är, men uppskattar ungefär vilka värden deras höjd kan tänkas ligga mellan. Om man staplar A och B på varandra samt C och D på varandra kan man då med Monte Carlo-metoden approximera sannolikheten att stapeln AB är högre än stapeln CD (se figur 2).



Figur 4: Klossar med okänd höjd staplade på varandra

Då man har ett intervall på hur höga samtliga klossar är kan man slumpa fram ett tal inom intervallet för var och en av klossarna, man ställer sedan upp modellen (A+B)-(C+D) och kollar om resultatet är positivt eller negativt. Repeterar man genereringen av de slumpmässiga värdena och uträkningen av modellen ett par tusen gånger kan man med hjälp av den datan räkna ut en approximation på sannolikheten.

I simuleringsprogrammet används Monte Carlo-metoden för att generera värden på ankomstintensiteten till systemet, samt betjäningstiden. Ankomstintensiteterna är, som tidigare nämnt, poissonfördelade och betjäningstiderna för samtalen är exponentialfördelade. Med hjälp av ett tredjepartsbibliotek utvecklat av S. Troschuetz (2007) genereras slumpmässiga siffror med olika fördelningar.

För varje iteration i simuleringen använder vi en poissonmetod för att få fram en siffra på hur många nya samtal som inkommer. Poissonmetoden använder ankomstintensiteten som λ för att generera fram en slumpmässig siffra. Sannolikheten att just siffra x slumpas fram är sannolikheten för att få just x nya samtal under en iteration, för det givna värdet på λ. I metoden finns en loop som kör 10000 gånger (kan göras fler gånger för bättre precision), och för varje iteration i loopen lägger den in ett poissonslumpat tal i en lista. Om det till exempel kommer 5 nya samtal med sannolikhet 20% kommer det då finnas ungefär 2000 instanser av siffran 5 i listan när loopen är klar. Metoden väljer sen helt slumpmässigt fram ett tal från listan och returnerar det. I exemplet ovan kommer då alltså siffran 5 returneras med sannolikhet 20%. Exponentialmetoden fungerar på ungefär samma sätt för att få fram värden på alla samtalens samtalslängder. Skillnaden är att exponentialfördelningen är kontinuerlig och man får först tillbaka ett reellt värde som behöver avrundas för att kunna användas i simulatorn som kör i diskret tid. Om betjäningstiden avrundas till 0 sätts den istället till 1 för att samtal inte ska kunna avslutas i samma iteration som de blir besvarade. Pseudokoden här nedan visar hur metoderna fungerar. DistributionGenerator är alltså poisson- eller exponentialgenereraren beroende på vilken metod man befinner sig i. Först sätter man väntevärdet

och sedan fyller man en lista med slumpade tal enligt den aktuella fördelningen för att till sist returnera ett slumpmässigt valt tal ur listan.

```
distributionGenerator.Lambda = lambda
i = 1
for(i <= 10000; i++)
{
      list.Add(distributionGenerator.Next)
}
return list.getRandom
```

3.2 Antaganden

Som en följd av att analysen bara betraktar vissa aspekter av callcenter kan flera antaganden göras. Undersökningen som behandlar enkla callcenter kommer bara ta hänsyn till ankomstintensiteterna och betjäningstiderna för samtal. Det intressanta när det kommer till mångsidiga callcenter kommer enbart, alltså utöver det som redan nämnts för enkla callcenter, att ta hänsyn till påverkan av antalet arbetsgrupper(en arbetsgrupp är en grupp bestående av operatörer med samma kompetenser) i callcentret. På grund av detta kan följande antas:

- Callcentret överbelastas inte, likt Erlang C-formeln förutsätts att trafiktätheten är mindre än antalet operatörer.
- Ärenden uppklaras alltid, inga kunder övergiver callcentret innan satisfierad betjäning.
- Betjäningstiden är oberoende av samtalstyp, alltså kommer simuleringen inte att skilja på olika typer av ärenden mer än att namnge dessa.
- Operatörers kompetenser är inte prioriterade eller ordnade, alltså betraktas alla kompetenser som likvärdiga.

3.3 Algoritm för samtalsdistribuering

Följande algoritm för samtalsdistribuering liknar den algoritm som används i studien av Rodney B. Wallace och Ward Whitt (2004). Algoritmen tar hänsyn till två olika fall vid simuleringen: När ett samtal inträffar och när en operatör blir ledig. I simuleringen kommer operatörer bli lediga i en och samma iteration som nya samtal tilldelas operatörer, händelserna sker alltså samtidigt och vi har på grund av detta valt att inte använda oss av algoritmen i fallet då en operatör blir ledig. En ofta upprepad rutin i algoritmen är inaktivitetstilldelning, rutinen är så som att för ärendet väljs den operatör som varit sysslolös den längsta perioden sen det senaste samtalet.

Algoritmen för att tilldela nya samtal till operatörer går till som så att först bestäms typen av samtalet, säg att samtalet är av typ k. Tilldelningen av samtalet bestäms sen genom att tillämpa inaktivitetstilldelning på de grupper med operatörer som har kompetens för typen k.

```
foreach(call in callQueue)
{
      foreach(group in employeeGroups)
      {
            if(group.Skills.Contains(call.Issue))
            {
                  list.Add(group)
            }
      }
      agent = FindLongestIdle(list)
      Assign(agent, call)
}

FindLongestIdle(list)
{
      int idle = -1
      returnAgent = null
      foreach(employeeGroup in list)
      {
            foreach(agent in employeeGroup)
            {
                  if(agent.IdleTime > idle)
                  {
                     returnAgent = agent
                     idle = agent.IdleTime
                  }
            }
      }
      return returnAgent
}
```

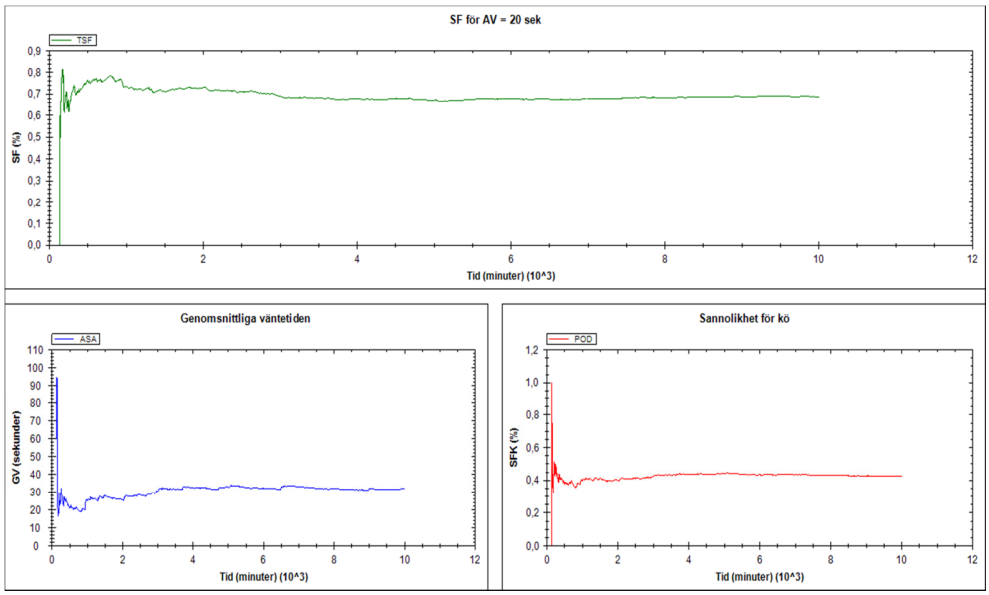
Det som sker i pseudokoden är att man går igenom samtalskön ett samtal i taget. Sedan går man igenom arbetsgrupperna, och om operatörerna i arbetsgruppen har kompetens att ta hand om samtalet läggs gruppen till i en lista. Listan skickas sedan till en metod som returnerar den operatör som har varit sysslolös under längst tid. Operatören tilldelas sedan samtalet. Det som sker i FindLongestIdle är att man går igenom alla operatörer för alla

arbetsgrupper som skickas in till metoden. När man hittar en operatör som har varit sysslolös längst av de som man hittills har gått igenom så sparar man undan den operatören om det skulle vara den som ska returneras. Man sparar även undan tiden operatören har varit sysslolös för att kunna jämföra med de operatörer som är kvar att gå igenom. När man är klar returnerar man den operatör som finns sparad.

4. Resultat

De två olika typer av callcenter som ska analyseras med simuleringen är enkla och mångsidiga callcenter. Enkla callcenter består av enbart en typ av ärenden som alla operatörer kan ta hand om. Mångsidiga callcenter består av ett antal olika samtalstyper. Operatörerna i mångsidiga callcenter kan ha kompetens för att ta hand om flera olika samtalstyper. Först ska enkla callcenter analyseras för att jämföra värden från simuleringar med deterministiska värden. Sedan undersöks fler enkla callcenter för att se påverkan av olika ankomstintensiteter och betjäningstider. Till slut analyseras mångsidiga callcenter för att se påverkan av olika antal av samtalstyper och oberoende arbetsgrupper.

Resultaten fås av simulatorn i form av grafer. Värdena som används tas i slutet av graferna där den i de flesta fall har hållit sig på ungefär samma värde under en längre tid. Graferna kan se ut som i figur 5 där övre grafen är för SF, den nedre till vänster är för GV och den nedre till höger till SFK.



Figur 5: Utdata från simulatorn

Samtliga testfall har körts under en period av 10000 minuter inne i simulatorn. På grund av att samtliga operatörer är lediga under starten av en simulering, vilket kan leda till mindre noggranna resultat, börjar resultat inte registreras förrän efter 400 minuter. Samtliga tester körs totalt 3 gånger varpå ett medelvärde, $\bar{x}$, räknas ut. När korrektheten för simulatorn testas används även en Erlang C-kalkylator för att beräkna deterministiska värden på SF, GV och SFK. Ett mått på korrekheten görs genom att beräkna

skillnaden, Δ, mellan de simulerade medelvärderna och de determinstiskt uträknade värderna.

4.1 Korrekthet

Då enkla callcenter är så förenklade är de inte intressanta från en praktisk synvinkel, däremot är de intressanta för att undersöka simuleringens korrekthet. Fördelen med enkla callcenter är att det går enkelt att räkna ut deterministiska värden för SF, GV och SFK med Erlang C. I följande testfall jämförs värden som fås ur simuleringen med värden från en Erlang C-kalkylator.

4.1.1 Litet callcenter

Varaktighet av simulering: 400-10000 minuter
s: 4
λ: 3 samtal per minut
β: 1 minut

Tabell 1 visar värden från simuleringen av ett litet callcenter med 4 operatörer, ankomstintensitet på 3 samtal per minut och betjäningstiden 1 minut. Det är tydligt att värdena från simuleringen stämde väl överens med de deterministiska värdena. SF avvek mellan simuleringarna med endast några procentenheter och medelvärdet från de tre simuleringarna var med två värdesiffrors nogrannhet exakt samma som det värde som räknades ut med Erlang C-formel. GV avvek med ungefär 10 sekunder per körning och SFK avvek med några procentenheter.

	Erlang C	**Sim 1**	**Sim 2**	**Sim 3**	$\overline{x}$	**Δ**
SF	0,63	0,60	0,63	0,65	0,63	0
GV[s]	30,57	44,75	39,79	37,32	40,62	10,05
SFK	0,51	0,52	0,50	0,48	0,5	0,01

Tabell 1: Värden från ett litet callcenter.

4.1.2 Större callcenter

Varaktighet av simulering: 400-10000 minuter
s: 65
λ: 6 samtal per minut
β: 10 minuter

I Tabell 2 ser man värden från ett större callcenterexempel med 65 operatörer, 6 inkommande samtal per minut och betjäningstid på 10 minuter. SF från simuleringarna avvek med ungefär 5 procentenheter, GV avvek med runt 4 sekunder i medel och SFK avvek med 5 procentenheter i medel.

	Erlang C	**Sim 1**	**Sim 2**	**Sim 3**	$\overline{x}$	**Δ**
SF	0,64	0,70	0,68	0,70	0,69	0,053
GV[s]	50,41	41,00	52,76	46,29	46,68	3,73
SFK	0,42	0,37	0,38	0,36	0,37	0,05

Tabell 2: Värden från ett större callcenter.

4.1.3 Låg ankomstintensitet, Hög betjäningstid

Varaktighet av simulering: 400-10000 minuter
s: 65
λ: 4 samtal per minut
β: 15 minuter

Tabell 3 visar värden från ett callcenter med 65 operatörer, 4 inkommande samtal i minuten och 15 minuters betjäningstid. SF avvek med 3 procentenheter i medel från de deterministiska värdena, GV avvek med runt 4 sekunder och SFK avvek med runt 3 procentenheter.

	Erlang C	**Sim 1**	**Sim 2**	**Sim 3**	$\overline{x}$	**Δ**
SF	0,62	0,60	0,66	0,69	0,65	0,03
GV[s]	75,61	96,57	65,87	53,03	71,82	3,79
SFK	0,42	0,43	0,38	0,35	0,39	0,03

Tabell 3: Värden från ett callcenterexempel med betydligt högre λ än β.

4.1.4 Hög ankomstintensitet, låg betjäningstid

Varaktighet av simulering: 400-10000 minuter
s: 65
λ: 15 samtal per minut
β: 4 minuter

Tabell 4 visar värden från ett callcenter med 65 operatörer, 15 inkommande samtal i minuten och 4 minuters betjäningstid. I medel avvek SF med 5 procentenheter, GV avvek med några sekunder och SFK avvek med runt 7 procentenheter.

	Erlang C	Sim 1	Sim 2	Sim 3	$\overline{X}$	Δ
SF	0,72	0,78	0,77	0,76	0,77	0,05
GV[s]	20,16	19,95	25,84	20,44	22,08	1,92
SFK	0,42	0,34	0,35	0,36	0,35	0,07

Tabell4: Värden från ett callcenterexempel med betydligt lägre λ än β.

4.1.5 Specialfall av mångsidigt callcenter

Varaktighet av simulering: 400-10000 minuter
s: 30
λ: 9 samtal per minut
β: 3 minuter
Arbetsgrupper: 10
Samtalstyper: 10

I callcentret finns det tio arbetsgrupper med 3 operatörer vardera. Varje arbetsgrupp har hand om samtliga 10 samtalstyper som existerar i systemet. Det här ger ett mångsidigt callcenter som kan jämföras med ett enkelt callcenter och på grund av detta går det bra att testa simulatorn med värden från Erlang C-kalkylatorn. I tabell 5 ser man värdena från simuleringen respektive de deterministiska värdena. I medel avvek SF med 5 procentenheter, GV avvek med runt 1 sekund och SFK avvek med runt 6 procentenheter.

	Erlang C	Sim 1	Sim 2	Sim 3	$\overline{x}$	Δ
SF	0,66	0,72	0,69	0,71	0,71	0,05
GV[s]	28,28	27,65	32,79	26,60	29,01	0,73
SFK	0,47	0,39	0,42	0,41	0,41	0,06

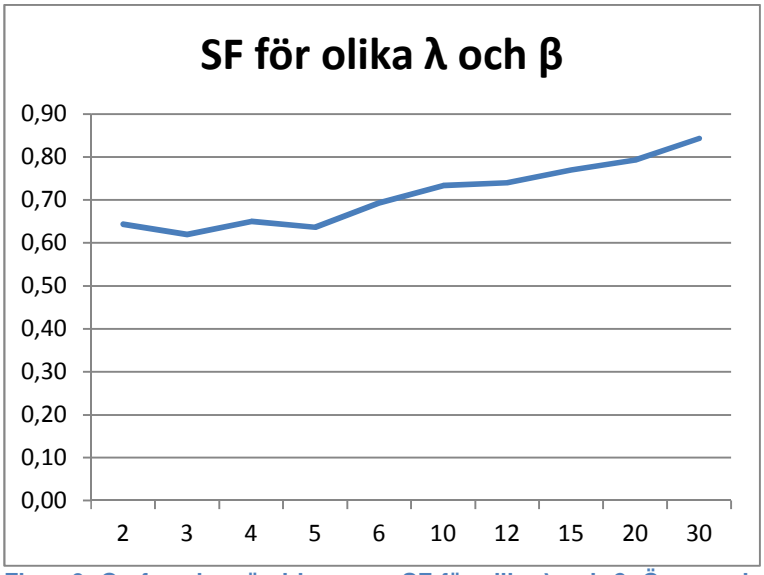
Tabell 5: Värden från specialfall av mångsidigt callcenter.

4.2 Ankomstintensiteter och betjäningstider

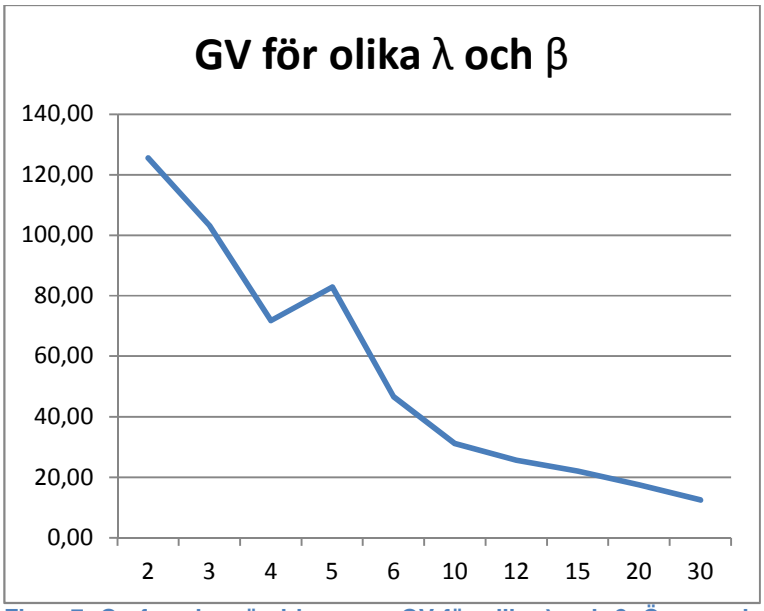
I det här fallet testas påverkan av olika ankomstintensiteter och betjäningstider utan att ändra trafiktätheten för callcentren. Antalet operatörer är 65 för samtliga mätningar och trafiktätheten är konstant 60 Erlang. Ankomstintensiteten börjar på 2 och går upp till 30, vilket betyder att betjäningstiden börjar på 30 och går ner till 2 för att multiplicerat med ankomstintensiteten alltid bli 60. I Tabell 6 ser man värden från de 10 olika simuleringsförsöken. SF ökade med 20 procentenheter från ha gått från 0,64 (lägst λ och högst β) till 0,84 (högst λ och lägst β). GV slutade som 1/10 av det ursprungliga värdet genom att gå från 125,58 sekunder till bara 12,51 sekunder. Figur 6 och figur 7 visar ändringen av SF respektive GV mer överskådligt.

λ	β	SF	GV [s]
2	30	0,64	125,58
3	20	0,62	103,24
4	15	0,65	71,82
5	12	0,64	82,95
6	10	0,69	46,68
10	6	0,73	31,09
12	5	0,74	25,59
15	4	0,77	22,08
20	3	0,79	17,52
30	2	0,84	12,51

Tabell 6: Värden på SF och GV beroende på olika värden på λ och β utan att ändra den totala trafiktätheten.



Figur 6: Grafen visar ändringen av SF för olika λ och β. Övre axeln är SF och undre axeln är λ, β ändras också i den undre axeln enligt tabell 6 för värden på λ.

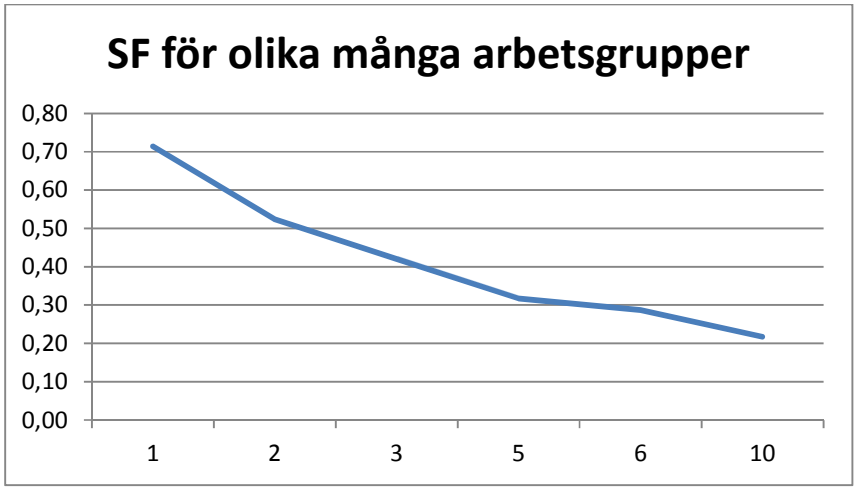


Figur 7: Grafen visar ändringen av GV för olika λ och β. Övre axeln är GV och undre-axeln är λ, β ändras också i den undre axeln enligt tabell 6 för värden på λ.

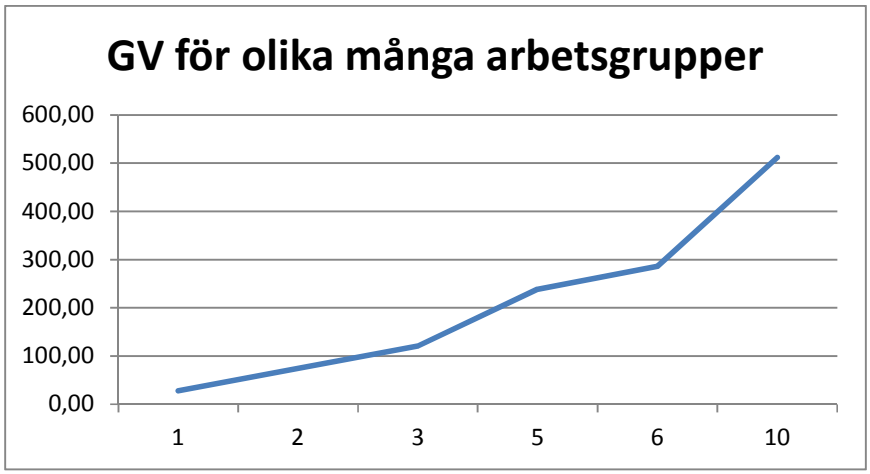
4.3 Arbetsgrupper i mångsidiga callcenter

Här testas hur antalet arbetsgrupper påverkar resultaten. Antalet operatörer är konstant 30, uppdelat på ett antal olika oberoende arbetsgrupper med lika många operatörer. Ankomstintensiteten och betjäningstiden är även de

konstanta och satta till 9 respektive 3, vilket ger en trafiktäthet på 27 Erlang. Alla arbetsgrupper har endast hand om ett område var och ingen grupp har hand om samma område som någon annan. I Figur 8 och 9 åskådliggörs förändringen av SF respektive GV för olika många arbetsgrupper. Det är tydligt att SF sänks respektive GV ökar när antalet arbetsgrupper och samtalstyper ökar.



Figur 8: Grafen visar ändringen av SF för olika antal arbetsgrupper. Undre axeln är antalet arbetsgrupper från 0 till 10 och övre axeln är SF.



Figur 9: Grafen visar ändringen av GV för olika antal arbetsgrupper. Undre axeln är antalet arbetsgrupper från 0 till 10 och övre axeln är GV.

5. Diskussion

I diskussionen kommer först tas upp felkällor vid simuleringen, där ligger fokus i att försöka förklara simuleringens svagheter. Sedan diskuteras resultaten från experimenten i ordningen: korrekthet, ankomstintensiteter och betjäningstider och till sist arbetsgrupper i mångsidiga callcenter.

5.1 Felkällor

Den främsta felkällan är varaktigheten, alltså tiden som simuleringen körs. Längre körtid av simuleringen resulterade i mer konvergerande resultat, men på grund av komplexiteten av callcenter-simuleringen kunde körtiden bli för lång. Det ansågs ändå vara tillräcklig konvergens i resultaten efter 30000 iterationer i varje testfall. I början av simuleringen får man värden som skiljer sig mycket från de värden som simuleringen har när den är närmare stabilitet. Eftersom det är callcentrens konvergensvärden som är intressanta, registreras resultat inte förrän en bit in i körningen. Men då alla körningar skiljer sig åt så nås stabilitetsnivån vid olika tidpunkter för olika körningar, vilket kan påverka slutresultatet även om det tas hänsyn till instabila startvärden.

Erlang C antar att ankomstintensiteten och betjäningstider är konstanta, vilket inte är fallet i simuleringen, därav tas inte hänsyn till perioder där samtal inkommer med mycket högre intensitet och där samtalen tar mycket längre tid att tas hand om (Tanner, 2004).

Korrektheten för simuleringen av mångsidiga callcenter kan inte verifieras genom att jämföra med deterministiska värden då Erlang C inte gäller för mångsidiga callcenter. Däremot om simuleringen visar sig vara korrekt för enkla callcenter kan man anta att den stämmer för mångsidiga om resultaten verkar rimliga. Det finns även specialfall av mångsidiga callcenter där alla arbetsgrupper kan ta hand om alla samtalstyper, eftersom dessa tekniskt sett är enkla callcenter kan man jämföra värden från dessa med deterministiska värden.

När betjäningstiderna i simuleringen räknas ut används som tidigare nämnt exponentialfördelningen, vilket är en kontinuerlig fördelning och metoden returnerar därmed ett reellt tal. Det reella talet måste avrundas för att kunna användas i simulatorn som kör i diskret tid. Då samtal inte heller kan avslutas i samma iteration som det blir besvarade så sätts betjäningstiden till 1 i de fallen den avrundas till 0.

5.2 Korrekthet

Det är tydligt att simuleringens värden stämde väl överens med de deterministiskt uträknade med Erlang C med några mindre avvikelser. Generellt avvek både SF och SFK med runt 5% i simuleringarna. Det är en acceptabel avvikelse då värdena från simuleringar generellt borde avvika från deterministiska värden med tanke på den slumpmässiga komponenten i simuleringen.

GV var den som avvek mest, värst var det i det lilla callcenter exemplet där GV skiljde sig i en av simuleringarna med 15 sekunder från det deterministiska värdet som låg på ungefär 30 sekunder. En möjlig förklaring till detta kan vara att betjäningstider som avrundas till 0 istället sätts till 1. I mindre callcenter, där betjäningstiderna är låga, får man fler betjäningstider som egentligen skulle vara 0 jämfört med större callcenter. Mindre callcenter får alltså betjäningstider vars medelvärde är större än vad som egentligen skulle vara fallet.

5.3 Ankomstintensiteter och betjäningstider

En klar trend, både vad gäller SF och GV, är att callcentret presterar bättre när trafiktätheten är uppdelad som så att λ är större än β. I fallet där man har låg λ må man inte få så många samtal men de samtal som kommer in till systemet stannar kvar väldigt länge. När man hamnar i det läge där alla operatörer är upptagna är det inte osannolikt att situationen förblir densamma under en längre period, vilket leder till att väntetiderna blir högre. Ser man istället på det andra fallet där man har ett större värde på λ får man istället in massor av mindre samtal till callcentret. För varje iteration i systemet kommer en hel del gamla samtal uppklaras, vilket lämnar plats till nya samtal. De många nya samtalen behöver med andra ord inte vänta länge i kön innan det öppnas upp en plats för dem, vilket leder till att väntetiderna sjunker drastiskt. Då båda systemen kommer ha identisk SFK (formeln använder bara trafiktätheten och antalet operatörer) är det logiskt att det systemet som har en lägre GV även har högre SF. Har man en låg GV kan man dra slutsatsen att fler av de som väntar kommer kunna bli tilldelade en operatör inom AV än om man har en högre GV.

5.4 Arbetsgrupper i mångsidiga callcenter

Av resultaten ser man att callcentret får en kraftig prestandasänkning när antalet arbetsgrupper stiger. SF sjunker från runt 70% för endast en arbetsgrupp till strax över 20% för 10 arbetsgrupper. Ännu större skillnad ser man på den genomsnittliga väntetiden som ligger på cirka 30 sekunder för

en arbetsgrupp men som ligger på över 500 sekunder med 10 grupper. Det här beror på slumpmässigheten i simulatorn som skapar obalans mellan grupperna. När man har många oberoende grupper med få operatörer blir resultatet att några av grupperna har fullt upp under en längre period medan andra grupper istället kan vara sysslolösa. Om man rent teoretiskt skulle tänka sig att samtal av de olika samtalstyperna inkommer till callcentret i ordning så att man hela tiden får lika många samtal av alla olika typer, samt att antal inkommande samtal och betjäningstiderna hela tiden ligger på det givna väntevärdet så torde resultaten inte variera som de gör nu. Om det kommer in ett samtal som får en väldigt hög betjäningstid påverkar det hela systemet mer när man har grupper med färre operatörer, till exempel i fallet då man endast har tre operatörer per grupp så kommer gruppen bara ha 2/3 av sin kapacitet under tiden som en av operatörerna får ta hand om det långa samtalet, jämfört med 29/30 i fallet då man har alla 30 operatörer i samma grupp. Det här gör att grupper som får en eller flera väldigt långa samtal försvagas rejält och många fler samtal av den typen gruppen tar hand om kommer fastna i kön, vilket försämrar prestandan för hela callcentret. På grund av att slumpgeneratorn, som används för att slumpa fram vilken samtalstyp ett visst samtal kommer ha, inte är perfekt blir effekten ännu starkare om det kommer ett långt samtal med en av de samtalstyper som kommer in mer frekvent till systemet.

6. Slutsatser

Resultaten pekar på att simulatorn var korrekt. Det kan dock behövas mer omfattande testning av mångsidiga callcenter då värden från dessa inte generellt kunde jämföras med deterministiska värden. Den största felkällan som kunde ha påverkat resultaten var varaktigheten av simuleringarna. De två möjliga lösningarna för den här felkällan är att optimera simuleringen eller att låta simuleringen köra en längre tid. Samtidigt är det inte heller fullt realistiskt att köra simulatorn i all oändlighet då ett riktigt callcenter inte är öppet för nya samtal dygnet runt. I riktiga callcenter kommer även ankomstintensiteten variera väldigt mycket under dagen. Resultaten blev dock tillräckligt bra för att kunna påstå att en längre körtid inte skulle behövas.

Man kan jämföra resultaten från experimenten med antal arbetsgrupper med påverkan av olika ankomstintensiteter och betjäningstider då båda kan leda till att systemet överbelastas. När Ankomstintensiteten är låg och betjäningstiden hög finns risken att det dyker upp flera omfattande samtal samtidigt som överbelastar delar eller hela callcentret. När det finns många olika oberoende arbetsgrupper som enskilt jobbar med sina områden finns risken att många av dessa arbetsgrupper blir sysslolösa under tiden som de andra arbetsgrupperna överbelastas. Man kan anse dessa som rimliga resultat, även om testfallen är något ringa, eftersom generellt sett är det alltid bättre att ha en överblick av hela systemet gentemot att arbeta oberoende och ovetande av varandra. Som fortsatt fördjupning av testningen av simuleringen skulle det vara av intresse att försöka bekräfta den allmänt accepterade principen inom flexibel tillverkning ”Man kommer långt med lite flexibilitet” (R. Wallace and W. Whitt, 2005). Detta skulle kunna utföras genom att skapa testfall där callcentren har flera arbetsgrupper där operatörerna i arbetsgrupperna tar hand om några enstaka, men väl utvalda, områden. Avslutningsvis kan man säga att simuleringen var lyckad och man fann prestandaskillnader i de olika testfallen.

7. Referenser

G. Koole, 2007, Call Center Mathematics. [e-book] http://www.math.vu.nl/~koole/ccmath/book.pdf [Hämtad 2011-02-09]

J. Champion, 2007, A flexible charting library for .NET. [online] http://www.codeproject.com/KB/graphics/zedgraph.aspx [Hämtad 2011-04-09]

M. Tanner, 2004, Limitations of Erlang-C. [online] http://www.mitan.co.uk/erlang/elgcprob.htm#ovrstaff [Hämtad 2011-04-09]

R. Wallace and W. Whitt, 2005, A staffing algorithm for call centers with skill-based routing. [e-book] http://pages.stern.nyu.edu/~gjanakir/WhittPaper.pdf [Hämtad 2011-03-08]

S. Troschuetz, 2007, .NET random number generators and distribution. [online] http://www.codeproject.com/KB/recipes/Random.aspx [Hämtad 2011-03-22]

J. Wittwer, 2004, “Monte Carlo Simulation Basics” från Vertex42.com. [online] http://vertex42.com/ExcelArticles/mc/MonteCarloSimulation.html [Hämtad 2011-03-23]

E. Weisstein, "Poisson Distribution." från *MathWorld*--A Wolfram Web Resource. [online] http://mathworld.wolfram.com/PoissonDistribution.html [Hämtad 2011-04-02]

E. Weisstein, “Exponential Distribution.” från MathWorld—A Wolfram Web Resource. [online] http://mathworld.wolfram.com/ExponentialDistribution.html [Hämtad 2011-04-02]

Appendix A – Översättningar

Engelska	Svenska	Förkortning(sv.)
ACD(Automatic Call Distributor)	Automatisk samtalsdistribuerare	ASD
Agent	Operatör	-
ASA(Average Speed of Answer)	Genomsnittsväntetiden	GV
AWT(Acceptable waiting time)	Acceptabla väntetiden	AV
Call Center	Callcenter	-
LIAR(longest-idle-agent-routing)	Inaktivitetstilldelning	-
Multi-skilled callcenter	Mångsidigt callcenter	-
Probability of delay	Sannolikhet för kö	SFK
SBR(Skill Based Routing)	Kompetensbaserad tilldelning	KBT
SL(Service Level)	Servicenivå	-
TSF(Telephone Service Factor)	Servicefaktor	SF