Hermun skautaskála hjá Ísáli

Gunnarr Baldursson & Ragnar Gísli Ólafsson

Apríl 2011

Útdráttur

Ble! Abstract

Efnisyfirlit

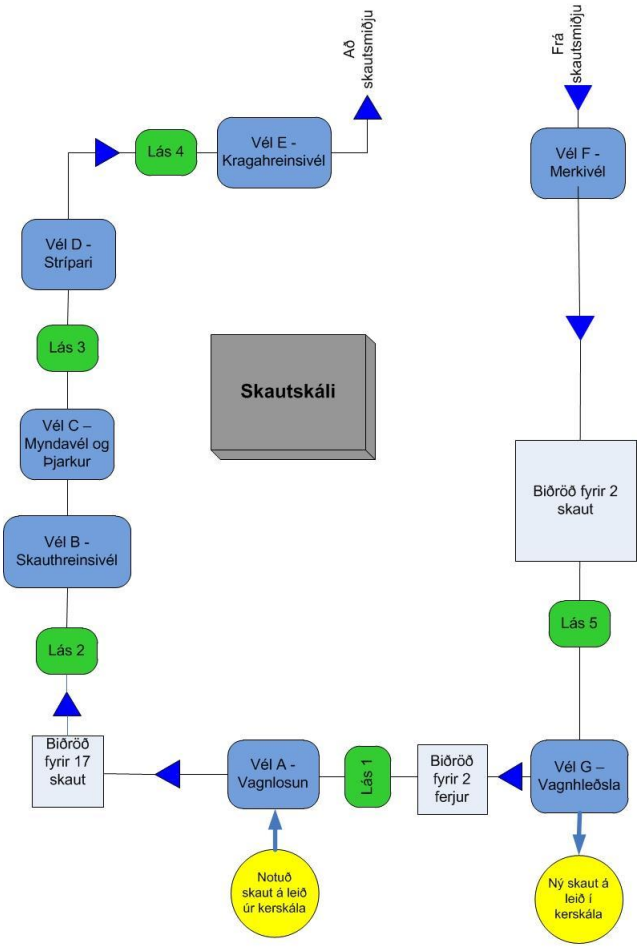
1 Inngangur

Alcan á Íslandi hf, betur þekkt sem Ísál, er hluti af Rio Tinto Alcan, fjölþjóðlegu fyrirtæki sem er stærsti álframleiðandi í heimi. Ísal rekur álverið í Straumsvík sem er ellefta stærsta álverið innan samsteypunnar. Framleiðslugetan er um 185 þúsund tonn og starfsmennirnir eru um 450; vélvirkjar, verkfræðingar, stóriðjugreinar, rafvirkjar, verkafólk, tæknifræðingar, málarar, skrifstofufólk, bifvélavirkjar, viðskiptafræðingar, múrarar, matreiðslumenn, rafeindavirkjar, smiðir o.fl

Ísal rekur þrjá kerskála þar sem að svonefnd skaut eru notuð til rafgreiningar áls. Kerin eru númeruð frá 1001 til 3160 þar sem fyrsta talan stendur fyrir númer skála, og næstu þrjár númer kers í skálanum. Hvert ker hefur 24 skaut sem að slitna með tímanum. Þess vegna þarf að skipta um þau á 26 til 30 daga fresti en það er mismunandi eftir skálum. Skáli 3 hefur stærri skaut og hærri straum, 165 kA og þar endast skaut í 26 daga. Skálar 1 og 2 hafa lægri straum, 133 kA og minni skaut sem að endast í 30 daga. Daglega þarf að skipta út um það bil 404 skautum.

Í kerskála er unnið á þremur vöktum allan sólarhringinn alla daga vikunnar og fer fram skautskipting á hverri vakt. Hver vakt nemur 8 klukkustundum, næturvaktin byrjar á miðnætti, dagvaktin klukkan átta og kvöldvaktin klukkan fjögur. Hver vakt skiptir því um $404/3 = 104$ skaut. Starfsmenn kerskála taka brunnin skaut úr kerum og setja ný skaut í kerin í staðinn. Síðan kemur starfsmaður skautskálans og nær í brunnu skautin sem bíða á vögnum í kerskálanum og flytur þau á sérstakan kæligang. Þar skilur hann þau eftir og nær í staðinn í brunnin skaut sem eru orðin köld og fer með þau í skautskála til hreinsunar. Í hvert skipti sem starfsmaður skautskála sækir brunnin úr kerskála kemur hann með ný skaut. Því er alltaf jafn fjöldi vagna sem fer inn í skautskálann og út úr honum.

Skautin eru flutt á tveimur tengdum vögnum með 12-14 skautum á í einu. Ferðir frá skautskála til kerskála eru aðeins farnar á dagvöktum og kvöldvöktum. Skaut sem þarf að nota á næturna eru því keyrð til kerskála á dag - og kvöldvöktum. Meðaltal fjölda ferða frá skautskala til kerskála eru



Mynd 1: Ferli skautskála

um það bil 30 á sólarhring, eða 15 á vakt. Skautskáli reynir að framleiða þann fjölda skauta sem nemur skautafjölda tveggja vakta hjá kerskála á hverri vakt, og er því tveimur vöktum á undan.

Fræðileg hámarks afkastageta skautskála eru 52 skaut á klukkustund en vegna bilanna er afkastageta á hverri vakt í besta falli um það bil 40 skaut á klukkustund. Skálinn er framleiðslulína sem að samtímis tekur skautleifar af vögnum og hreinsar ásamt því að taka á móti nýjum skautum og setja á vagnanna. Lestun nýrra skauta og losun brunninna skauta er samtengt ferli, ef ekki er hægt að taka brunnin skaut af vagni þá er heldur ekki hægt að setja ný skaut á vagn.

Framleiðsluferli skautskála hefst þegar skautleifar koma á vögnum að vél A, sem að hífir þær af vögnum. Eftir það fara þær í gegnum vélar B til E þar sem að leifarnar eru hreinsaðar þannig að gaffallinn stendur einn eftir. Gaffallinn heldur síðan áfram inn í aðra byggingu sem að nefnist skautsmiðja, þar sem hann er skoðaður, réttur af og sandblásinn áður en hann fer í steypun þar sem að ný kol eru steypt við hann. Þá er hann tilbúinn sem nýtt skaut. Þegar þessu ferli er lokið kemur skautið að vél F þar sem það er merkt og sent til vélar G. Vél G lestar skautið á vagn, sem er síðar keyrður til kerskála. Þetta ferli er lýst á Mynd 1.

Skautið er tekið inn í ferlið þannig að það er hengt á ferju sem að er dregin áfram af keðju, sem að fer í gegnum allan skautskálann og inn í skautsmiðjuna og til baka. Ferlið er raðgengt svo ef vélr er að afgreiða skaut þarf skautið á eftir að bíða þangað til að vélin hefur lokið sér af. Til að stýra þessu flæði eru svokallaðir lásar staðsettir með regulegu millibili á keðjunni og kúpla þeir ferjum út til að stöðva þær. Þannig geta sum skaut verið á hreyfingu á meðan önnur eru kyrrstæð því að keðjan sjálf stöðvar ekki nema slökkt sé á henni handvirkt. Á bak við sumar vélar eru biðraðir en þar bíða skaut eftir afgreiðslu ef að vélin er upptekin. Lása og biðraðir má sjá á Mynd 1. Lásar á undan fullum biðröðum mega ekki sleppa sýnum skautum þangað til að það rúmast til í röðinni. Skaut geta ekki farið framhjá vélum þannig að ef að vél bilar lengi og röð hennar fyllist heldur sá lás sem kemur þar á undan sýnu skauti föstu og þannig koll af kolli. Þannig getur löng bilun stöðvað skautahreinsiferlið

í einhvern tíma þó að keðjan sem ber ferjurnar haldi áfram keyrslu. Hún er þá eins og bílvél með einhvern snúningshraða sem er í hlutlausum gír.

Það er nokkuð slembið hvaða vélar stoppa nema vél F sem að bilar nánast aldrei. Þegar stærri bilanir eiga sér stað þarf að kalla út viðgerðarmenn en í flestum tilfellum tekur það 5 til 30 mínútur að koma bilaðri vél aftur af stað. Skakkt skaut í vél flokkast sem bilun og þá þarf starfsmaður að bakka því út úr vélinni, leiðrétta það og senda inn aftur. Svoleiðis atvik eiga sér stað nokkrum sinnum á sólarhring og er helsta ástæða þess að afköst skautskála nema um það bil 40 skautum á klukkustund. Ef viðgerðartímar eru þeim mun lengri á einhverri vakt þá þarf vaktin sem kemur á eftir að vinna upp framleiðslutapið. Skautskáli keyrir aðeins á dagvöktum og kvöldvöktum.

Framkvæmdir eru hafnar við að auka strauminn í kerum 1 og 2 sem veldur dræmri endingartíma skauta, og koma þau þá til með að endast í 26 til 28 daga eftir straumhækkun. Gerð verður sú nálgun að alltaf sé nóg til af nýjum skautum í skautsmiðju sem koma að vél F. Verkefnið er að herma ferli skautskála með eftirfarandi vangaveltur í huga:

1. Hversu mikið af töfum (í mínútum talið) þolir skautskálinn til að ná lágmarksafköstum?
2. Er það ráðlegt að stækka biðraðir eða bæta við biðröðum?
3. Hve miklu munar það fyrir ferlið ef að starfsmenn koma vélum af stað eins fljótt og þeir geta?
4. Ef tafir eru litlar, hvenær hefur vakt náð lágmarksafköstum?
5. Hversu fljótur er skálinn að vinna upp langar viðgerðatafir?
6. Hvaða áhrif hefur hækkun straums á ferlið?

Þar sem að meðaltími, besti og versti tími skauta, meðalhámarkslengd biðraða og nýtni véla verða til hliðsjónar.

2 Niðurstöður

Í grein 3.2 kemur fram að talan nokkrar bilanir séu 8 bilanir. Í heimsókn til Ísal [3] kom fram að vegna bilanna nemi raunframleiðsla skautskála um það bil 44 skautum á klukkustund þrátt fyrir að skálin geti fræðilega afkastað meiru. Inntaksgögn líkans (sjá grein 5.2) er þannig að við átta bilanir á vakt nemur framleiðsla um það bil 44 skautum á klukkustund. Í inntaki eru skilgreindar tvær heiltölubreytur, önnur er fyrir lágmarksfjölda bilana á vakt og hin fyrir hámarksfjölda bilana á vakt. Þessar breytur mynda því bil, og fyrir hverja heiltölu á þessu bili er hermunin framkvæmt. Í inntaki eru þessar breytur 3 og 9. Í grein 3.1 er talað um að lágmarksframleiðsla skautskála á sólarhring fyrir straumhækkun eru 404 skaut. Eftir hækkun er talan 444 skaut.

3 Forsendur og Líkan

Til að komast að niðurstöðum smíðuðum við líkan sem að hermir eftir ferli skautskála. Í næstu undirgreinum er forsendum líkansins lýst.

3.1 Umskipting skauta og afköst Skautskála

- Fyrir straumhækkun þá þarf að skipta um skaut í skálum 1 og 2 á 30 daga fresti, og í skála 3 á 26 daga fresti. Það gerir $\frac{24 \cdot 160}{26} + \frac{24 \cdot 160}{30} + \frac{24 \cdot 160}{30} = 403.69$ skaut á dag, þar sem að allir skálar hafa 24 skaut í í hverju keri og 160 ker eru í hverjum skála. Nefnararnir í formúlunni eru endingadagar skauta í viðeigandi kerskála. Sú tala er námunduð upp í 404 skaut á dag og eru það lágmarksafköst skautskála.
- Eftir straumhækkun þarf að skipta um skaut í öllum skálum á 26 daga fresti. Það gerir $\frac{24 \cdot 160}{26} + \frac{24 \cdot 160}{26} + \frac{24 \cdot 160}{26} = 443.07$ skaut á dag. Sú tala er námunduð upp í 444 skaut á dag og er það lágmarks afkastageta skautskála eftir straumhækkun.
- Fræðileg hámarksafköst skála eru 52 skaut á klst, eða $16 \cdot 62 = 832$ skaut á sólarhring (Skautskálinn starfar aðeins í tvær vaktir, eða 16 klukkustundir á sólarhring).
- Raunveruleg hámarksafköst skála eru 40 skaut á klst, eða $16 \cdot 40 = 640$ skaut á sólarhring (Skautskálinn starfar aðeins í tvær vaktir, eða 16 klukkustundir á sólarhring).

3.2 Bilanir

Samkvæmt verkefnislýsingunni [1] er það nokkuð slembið hvaða vélar bila, að vél F undanskilinni, og að bilanir eiga sér stað nokkrum sinnum á vakt. Gildið á tölunni *nokkrum sinnum* er illa skilgreint en höfundar sammæltust um töluna 8. Fyrir flestar bilanir er viðgerðartíminn 5 til 30 mínútur. Í einhverjum tilfellum þarf að ræsa út viðgerðarmann ef um stórar bilanir er að ræða og slíkar bilanir geta varað í nokkrar klukkustundir. Engin önnur gögn liggja fyrir um bilanir eða tíðni þeirra og þar sem að gögnin eru ekki nákvæmari voru eftirfarandi forsendur gefnar:

- Allir viðgerðartímar liggja á bilinu 5 til 30 mínútur.
- Viðgerðartímar eru veldisdreyfðir þannig að mestar líkur eru á viðgerð taki 5 mínútur og minnstar líkur eru á 30 mínútuna viðgerð. Þar sem að bilanir og tafir vegna skakkra skauta í vélum má flokka undir sama hatt þykir höfundum líklegast að um slíkar tafir sé að ræða frekar en vélræna bilun.
- Tímasetningar bilana á sólarhring eru uniform dreifðar.
- Ef að vél A eða G bila eru engar ferðir farnar frá Skautskála til Kerskálanna meðan á viðgerð stendur.

3.3 Einingar og undirkerfi

Þættir skautskála eru dregnar saman í undirkerfi eins og sjá má á eftirfarandi töflu:

Eining	**Þættir**	**Hlutverk**
A	Vél A, 14 skauta biðröð í formi vagna	Vagnlosun
B	Vél B, biðröð og lás sem geyma 17 skaut	Skauthreinsivél
C	Vél C	Myndavél og Þjarkur
D	Vél D, einn lás	Strípari
E	Vél E, einn lás	Kragahreinsivél
F	Vél F, einn lás	Merkivél
G	Vél G, biðröð fyrir 2 skaut	Vagnhleðsla

Þessu er lýst á Mynd 2. Vélar B og C geta unnið tvö skaut í einu. [1]

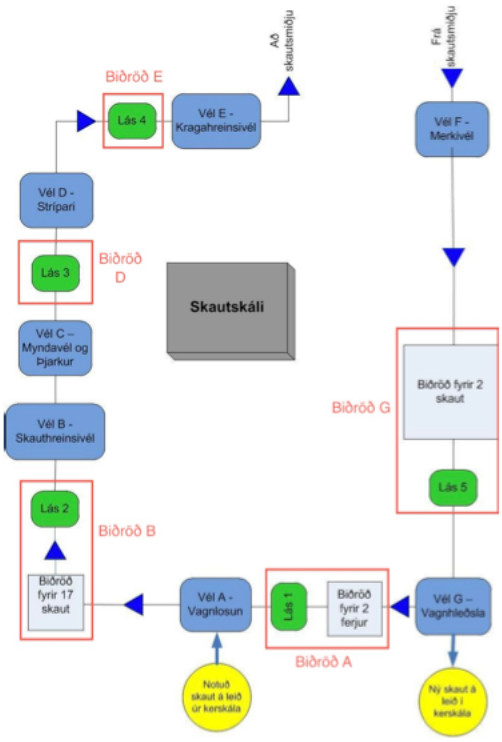
Þegar Mynd 2 er skoðuð má sjá að hægt er að skipta Skautskála upp í tvo helminga sem hefur hvor sitt inntak og sitt úttak. Inntak í vinstri helming kemur frá vél A, og úttak hans fer frá vél E. Inntak í hægri helming kemur frá vél F, en sú nálgun er gerð að þar sé ávallt nóg af nýjum skautum að taka, og úttak þess helmings er vél G, sem að hleður nýju skautunum á vagna. Við heimsókn í Ísal [3] kom fram að bilanir og tafir megi sjaldnast rekja til hægri helmingsins. Þar eru aðeins tvær vélar meðan vinstri hliðin hefur fimm vélar sem að vinna flóknari verk. Af þeim ástæðum er vél F undanskilin hermun. Vél G getur bilað, og ef það gerist stöðvar lestun og losun skauta um þann tíma sem það tekur að gera við bilunina.

Þegar vagn kemur með skautaleyfar til vélar *A* bíður hann á meðan leyfarnar eru hýfðar af honum. Því næst er hann er hann hlaðinn með nýjum skautum [1]. Gert er ráð fyrir því að fjöldi skautaleyfða sem hýfðar eru af vagni og fjöldi nýrra skauta sem lestuð eru á vagn sé um það bil sá sami.

3.4 Vélar og vinnslutímar

Vinnslutími vélar er sá tími sem líður milli þess að skaut kemur að lausri vél og fer frá vélinni aftur. Færslutími er sá tími sem líður milli þess að skaut fer frá vél og kemur að næstu vél. Þeir eru reiknaðir út samkvæmt gagnaskjali, [2]. Þar sem að vélar B og C geta unnið tvö skaut í einu er vinnslutími þeirra helmingaður.

Vél	**Vinnslutími**	**Færslutími**
A	70	129.83
B	21.798	122.83
C	12.7	18.98
D	67.41	22.74
E	69.75	0



Mynd 2: Einingar líkans

3.5 Atburðir og kjarnavirkni líkans

Líkanið er atburðadrifið: einhver atburður á sér stað sem að getur skrásett annan atburð og þannig koll af kolli þangað til að keyrslu er lokið. Kjarni líkansins er atburðavinnslan sjálf, hvernig það bregðast skal við þeim atburðum sem að skilgreindir eru. Eftirfarandi atburði skal skilgreina:

- Vagn kemur með brunnin skaut að vél A
- Skaut kemur að vél
- Skaut fer frá vél
- Vél bilar
- Vél löguð
- Endir upphitunartíma
- Endir hermunar

Næstu undirgreinar útskýra hvernig bregðast þarf við þessum viðburðum.

Vagn kemur með brunnin skaut

Vagnar sem koma með hrein skaut geta flutt 12 til 14 skaut saman lagt og er það uniform dreifð slembitala. Fyrir hvert skaut þarf að framkalla atburðinn *skaut kemur að vél*, þar sem að vélin er vél *A*. Hver slíkur atburður þarf að innihalda eftirfarandi gögn:

- Tímasetningin þegar atburðurinn á sér stað
- Staðsetning skauts í ferlinu
- Tímasetningin þegar skautið kemur fyrst í kerfið
- Raðnúmer skauts

Loks þarf að skrásetja annan *vagn kemur með brunnin skaut* atburð. Þar sem að vagnar koma á um það bil 32 mínútna fresti að meðaltali á dag, meðan unnið er í Skautskála [1], er eðlilegt að koma þeirra sé uniform slembitala milli 28 og 30. Ef að vélar A eða G eru bilaðar þarf að fresta komu næsta vagns um þann tíma sem það tekur við að gera við vélarnar af því að vagnlosunin og lestun eru samtengd ferli [1].

Skaut kemur að vél

Þegar skaut kemur að vél þarf að huga að ýmsu.

- **Er vélin upptekin?**
  Ef að vélin er laus skal merkja að skautið hafi fengið þjónustu umsvifalaust. Svo skal skrásetja *Skaut fer frá vél* atburð sem að inniheldur tímasetningu brottfarar og vél sem að farið er frá. Annars skal vista komutíma skauts og setja það í röð þeirrar einingar sem skautið kemur að, ef einhver er.
- **Hefur vélin röð og ef svo er, er röðin full?**
  Ef að vélin hefur enga röð eða röðin er full þarf að fresta komu þessa skauts, lásinn sem að heldur því má í rauninni ekki sleppa því þangað til að það rúmast til í röðinni.
- **Er vélin biluð?**
  Ef að vélin er biluð þarf að fresta þessum atburði um þann tíma sem að samsvarar viðgerðartímanum.

Skaut fer frá vél

Þegar *skaut fer frá vél* atburður er meðhöndlaður hefur vél unnið sitt verk á skautinu.

- Ef að vélin sem skautið fer frá er biluð þarf að fresta atburðinum um þann tíma sem það tekur að gera við vélina.
- Ef að vélin sem skautið fer frá er vél E þarf að hækka teljara sem telur hversu mörg skaut hafa farið frá vél A til E. Sú forsenda var gerð að fjöldi skautaleyfa sem losuð eru af vögnum við A sé um það bil sá sami og fjöldi nýrra skauta sem lestuð eru við G má hækka þennan teljara um tvo. Annars skal skrásetja *Skaut kemur að vél* atburð frá núverandi vél sem á að meðhöndla eftir færslutímann að næstu vél.
- Ef að röð vélarinnar sem farið er frá er ekki tóm þarf að vinna fremsta skautið í vélinni, halda utan um hve lengi skautið þurfti að bíða eftir þjónustu og skrásetja þá *Skaut fer frá vél* atburð eftir vinnslutíma vélarinnar.

Vél bilar

Þegar að vél bilar þarf að merkja hana bilaða, áætla viðgerðartíma fyrir hana og skrásetja *Vél löguð* atburð eftir viðgerðartímann. Sjá undirgrein 3.2 fyrir umfjöllun um bilanir að ofan.

Vél löguð

Þegar að biluð vél er löguð þarf að merkja vélina lagaða, svo að hún valdi ekki lengur töfum í ferlinu.

Endir upphitunartíma

Nú þarf að endurstilla skautateljara og ýmsar tölfræðibreytur sem að halda utan um tafir raða og nýtni véla.

Endir hermunar

Nú þarf að prenta út skýrslu á skjá eða í skjal með niðurstöðum hermunar.

4 Sannreyning Líkans

Höfundar beittu fjórum aðferðum til að sannreyna það að líkanið sé gott og gilt fyrir gögnin sem þeim voru gefin.

Samanburður við þekktar niðurstöður

Fræðileg hámarks framleiðslugeta skautskála eru 52 skaut á klukkustund [1], en vegna bilanna eru raunafköst nær 44 skautum á klukkustund [3] vegna bilana. Þegar frumgerð líkans var keyrð upphaflega, áður en rökfræði sem snýr að bilunum far smíðuð, náðu bestu afköst líkansins að meðaltali 56 skautum á klukkustund. Þar er vissulega skekkja um 4 skaut á klukkustund en hvort sú skekkja stafar af mistökum í líkanagerð eða dræmum gögnum [2] um vinnslutíma véla og færslutíma skauta

á milli þeirra er erfitt að segja til um. Hinsvegar eru 56 skaut / klst nokkuð nákvæmt og gefur það til kynna að líkanið og gögnin séu ásættanlega lýsandi fyrir ferlið sjálft.

Eftir að rökfræði bilana var smíðuð eru afköst um það bil 46 skaut / klst, miðað við 8 bilanir á vakt, sem er 4 skautum meira en raun ber vitni um. [1]. Niðurstöðurnar eru samt sem áður nærri lagi og þykja því bera vitni um áreiðanleika líkansins og gagnana sem það notar.

Ferlið rakið

Þegar á þróun líkansins stóð var staða véla og raða prentuð í skrá við hvern atburð sem var meðhöndlaður. Þannig var hægt að sjá það skref fyrir skref hvernig skaut ferðuðust milli véla og biðraða til að ganga úr skugga um að allt væri samkvæmt settum reglum.

Mismunandi inntaksgögn

Ferlið var hermt fyrir mismunandi margar bilanir á vakt. Útkoman var eins og við var að búast; bein tenging er á milli fjölda bilanna og afkastagetu á skautskála klukkustund.

Extreme Programming

Höfundar smíðuðu líkan í *Extreme Programming* stíl sem stuðlar að hraðvirkari uppgötvun villa.

Heimildir

[1] Starfsmaður Ísal, *HermunIsal_2011_r2.pdf*. 2011.

[2] Starfsmaður Ísal, *Millitimar.xls*. 2011.

[3] *Heimsókn til Ísal*. 21. mars 2011.

5 Viðauki

5.1 Frumgerð í forritunarmálinu C

```
1  /*
2   *  isal.c
3   *
4   *
5   *  Created by Gunnarr Baldursson & Ragnar Gisli Olafsson on 4/18/11.
6   *  Copyright 2011 Haskoli Islands. All rights reserved.
7   *
8   */
9
10 #include <stdio.h>
11 #include <string.h>
12 #include <stdlib.h>
13 #include <math.h>
14 #include <time.h>
15 #include "simlib/rndlib.h"
16 #include "simlib/simlib.h"
17
18 // EVENTS
19 #define EVENT_WAGEN_UNLOAD_ARRIVAL   1
20 #define EVENT_WAGEN_UNLOAD_DEPARTURE 2
21 #define EVENT_SKAUT_ARRIVAL 3
22 #define EVENT_SKAUT_DEPARTURE 4
23 #define EVENT_MACHINE_FAILURE 5
24 #define EVENT_MACHINE_FIXED 6
25 #define EVENT_END_SIMULATION 7
26 #define EVENT_END_WARMUP   8
27
28 // STREAMS
29 #define STREAM_WAGEN_ARRIVAL 1
30
31 //Other constants
32 #define NUM_MACHINES 7
```

```
33 #define WAGEN_LOAD 14
34 #define MACHINES_ON_THE_LEFT_SIDE 5
35 #define MACHINES_ON_THE_RIGHT_SIDE 2
36 #define OPTIMAL_THROUGHPUT 52
37 #define ACTUAL_THROUGHPUT 40
38 #define TRANSFER_ARRAY_LENGTH 11
39 #define PREP_TIME 20.0
40
41 typedef struct
42 {
43     float failtime;
44     float downtime;
45     int machine_nr;
46 } breakdown;
47
48
49 //#define LOADING_TIME_PER_SKAUT
50
51 // Global variables
52 int number_of_machines, min_productivity, min_no_failures, max_no_failures,
       skaut_throughput;
53 float mean_wagen_arrival, std_wagen_arrival, mean_failures, std_failures,
       min_machine_repair_time, max_machine_repair_time, end_warmup_time,
       end_simulation_time;
54
55
56 int sampst_delays, throughput_time; // variable for queue delays and throughput
       time
57 time_t dummy;
58 unsigned int skaut_id, stream, failure_nr;
59 int queue_size[NUM_MACHINES +1];
60 float machine_broken[NUM_MACHINES +1];
61 breakdown *fail_list;
62 int fail_index;
63
64 int is_machine_busy[NUM_MACHINES +1],
65     queue_size[NUM_MACHINES +1];
66
67 float work_time[NUM_MACHINES + 1],
68     transfer_time[NUM_MACHINES +1]; // +1 is the less preferable simlib
          indexing scheme
69
70
71 float temp_transfer[TRANSFER_ARRAY_LENGTH];
72
73 FILE *infile, *outfile;
74
75 /* Function signatures */
76
77 // Usage: create_machine_fail_events(number_of_failures)
78 // Pre:   init_twister must be called for random number generation
79 // Post:  scheduled events have been created for machines
80 void create_machine_fail_events(int);
81
82
83 // Usage: push_array();
84 // Pre:   we expect that correct values are in transfer array
85 // Post:  our temp_transfer array now has the values in transfer_array
86 void push_array();
87
88 // Usage: pop_array();
89 // Pre:   we expect that correct values are in transfer_temp array
90 // Post:  our transfer array now has the values in transfer_temp
91 void pop_array();
92
93 // Usage: wagen_arrival();
94 // Pre:   EVENT_WAGEN_UNLOAD_ARRIVAL is the next event to be processed
95 // Post:  14 EVENT_SKAUT_ARRIVAL events are next to be processed on the event
       list.
96 void wagen_unload_arrival();
```

```
97
98  // Usage: skaut_arrival();
99  // Pre:   EVENT_SKAUT_ARRIVAL is the next event to be processed
100 // Post:  a skaut has been processed by a machine or put in it's queue.
101 //        subsequent events may have been scheduled
102 void skaut_arrival();
103
104 // Usage: skaut_departure();
105 // Pre:   EVENT_SKAUT_DEPARTURE is the next event to be processed
106 // Post:
107 void skaut_departure(); // do we need an event for departure?
108
109 // Usage: machine_failure();
110 // Pre:   EVENT_MACHINE_FAILURE is the next event to be processed
111 // Post:
112 void machine_failure();
113
114 // Usage: machine_fixed();
115 // Pre:   EVENT_MACHINE_FIXED is the next event to be processed
116 // Post:
117 void machine_fixed();
118
119 // Usage: end_warmup();
120 // Post:  SIMLIB statistical variables have been cleared
121 void end_warmup();
122
123 // Usage: parse_input(input_filename_data,input_filename_time);
124 // Pre:   input_filename_data,input_filename_time of type char[],
125 //        global variables from the input file exist.
126 // Post:  the global variables were assigned values from input_filename,
127 //
128 void parse_input(char[] ,char[]);
129
130 // Usage: x = N(muy, sigma, stream);
131 // Pre:   muy and sigma are of type float
132 //        stream is of type int
133 // Post:  x is a random gaussian distributed variable of type float
134 //        with mean muy and std sigma
135 float N(float muy, float sigma, int stream);
136
137 // Usage: report("the_report.out");
138 // Pre:   the values to be reported have values
139 // Post:  a report on program values and simlib statistics
140 //        have been APPENDED to "the_report.out"
141 void report();
142
143 // Usage: schedule_failures(i);
144 // Pre:   the global variable end_simulation_time has a value, i is of type int
145 // Post:  i failures have been scheduled uniformly on machines
146 //        with ?random? repair times on the interval [min_machine_repair_time,...
        max_machine_repair_time]
147 //        uniformly distributed over the interval 0...end_simulation_time
148 void schedule_failures(int i);
149
150 void queue_is_full();
151
152 int main()
153 {
154 // load datafiles
155     parse_input("adal_inntak.in","velar_og_bidradir.in");
156
157     // initialize arrays and variables
158     if((fail_list = malloc(sizeof(breakdown)*max_no_failures))==NULL) {
159   printf("Allocation Error\n");
160   exit(1);
161     }
162
163
164
165     int b;
```

```
166 /*     for (b=1; b <= number_of_machines; b++) {
167   printf("transfer_time[%d] = %f\n", b,transfer_time[b] );
168   printf("busy %d broken %f \n",is_machine_busy[b],machine_broken[b]);
169   }*/
170     // We perform simulation for "a few" failures per day
171     stream = (unsigned int)time(NULL) % 100;
172     for (failure_nr = min_no_failures; failure_nr < max_no_failures; failure_nr
          ++) {
173
174   memset( is_machine_busy,0, NUM_MACHINES +1 );
175   memset( machine_broken,0, NUM_MACHINES +1);
176   memset( fail_list ,0, sizeof(breakdown)*max_no_failures);
177   fail_index = 0;
178   skaut_throughput = 0;
179   sampst_delays = number_of_machines +1;
180   throughput_time = number_of_machines +2;
181
182
183   skaut_id = 1;
184   skaut_throughput = 0;
185
186
187   // Initialize rndlib
188   init_twister();
189
190   // Initialize simlib
191   init_simlib();
192
193   maxatr = 6; // how many attributes do we need?
194
195   /* Schedule machine breakdown time */
196   create_machine_fail_events(failure_nr);
197
198   /* Schedule first wagen arrival */
199   //transfer[3] = 1.0;
200   event_schedule( 10.0, EVENT_WAGEN_UNLOAD_ARRIVAL );
201
202   /* Schedule end of warmup time */
203   event_schedule( end_warmup_time, EVENT_END_WARMUP );
204
205   /* Schedule simulation termination */
206   event_schedule( end_simulation_time, EVENT_END_SIMULATION );
207
208   next_event_type = 0;
209
210
211
212   while (next_event_type != EVENT_END_SIMULATION) {
213
214       timing();
215       /*                    printf("event_type = %d, transfer[3] = %f\n",
          next_event_type, transfer[3]);
216           int k;
217           for (k = 1; k <= number_of_machines; k++)
218           printf("Items in machines/queues %d:  %d, %d\n", k, list_size[k],
              list_size[number_of_machines +k]);
219           printf("\n");
220       */
221
222
223       switch (next_event_type) {
224       case EVENT_WAGEN_UNLOAD_ARRIVAL:
225     wagen_unload_arrival();
226     break;
227       case EVENT_SKAUT_ARRIVAL:
228     skaut_arrival();
229     break;
230       case EVENT_SKAUT_DEPARTURE:
231     skaut_departure();
232     break;
```

```
233         case EVENT_MACHINE_FAILURE:
234       machine_failure();
235       break;
236         case EVENT_MACHINE_FIXED:
237       machine_fixed();
238       break;
239         case EVENT_END_WARMUP:
240       end_warmup();
241       break;
242         case EVENT_END_SIMULATION:
243       report();
244       break;
245         }
246     }
247       }
248 }
249
250 void wagen_unload_arrival()
251 {
252
253       int i;
254       int current_unit = 0;
255       float wagen_arrival_zeit = unirand((mean_wagen_arrival-std_wagen_arrival)
              *60.0,(mean_wagen_arrival+std_wagen_arrival)*60.0,stream);
256
257       for (i = 1; i<NUM_MACHINES+1; i++) {   //delay unload of skaut by the time
              it takes to repair
258       if (machine_broken[i] > 0.0) {
259         event_schedule(sim_time + machine_broken[i], EVENT_WAGEN_UNLOAD_ARRIVAL);
260         return;
261       }
262       }
263
264       if (list_size[number_of_machines + 1] != 0) { // ef allt er enn fullt Ã¿Ãą
              koma meÃř nÃęsta vagn eftir uÃ¿b hÃąlftÃŋma
265     event_schedule(sim_time + wagen_arrival_zeit, EVENT_WAGEN_UNLOAD_ARRIVAL);
266     return;
267       }
268
269       int vagn_magn =  WAGEN_LOAD-((int)unirand(0.0,3.0,stream));   //12 - 14
              skaut Ãą hverjum vagni
270       for (i=1; i <= vagn_magn; i++) {
271
272     transfer[3]=1.0;
273     transfer[4] = sim_time + (i * 0.01); // skaut entering system time
274     transfer[6] = (float) skaut_id++;
275     //printf("tr4 in wagen: %f\n", transfer[4]);
276     event_schedule( sim_time + ( i* 0.01),  EVENT_SKAUT_ARRIVAL);
277       }
278
279       event_schedule(sim_time+wagen_arrival_zeit, EVENT_WAGEN_UNLOAD_ARRIVAL);
280 }
281
282
283 void skaut_arrival()
284 {
285       push_array();
286       int current_unit = (int)transfer[3];
287       int i;
288
289       for (i = NUM_MACHINES; i>=current_unit; i--) {   //add delay if there is a
              broken machine before current one
290     if (machine_broken[i] > 0.0) {
291         if ((list_size[1+number_of_machines + current_unit] < queue_size[1+
                current_unit])||queue_size[1+current_unit] == 0) {   // if current
                machine is broken then delay it.x
292       event_schedule(sim_time +  machine_broken[i] + work_time[current_unit],
              EVENT_SKAUT_ARRIVAL); //also if next queue is full then delay it.
293       return;
294       }
```

```
295     }
296       }
297
298       // check if machine is not busy
299       if (list_size[current_unit] == 0 && machine_broken[current_unit] == 0.0) {
300     sampst(0.0, sampst_delays);
301     sampst(0.0, current_unit);
302
303     list_file(FIRST, current_unit); // last := first here because there are only
          to be 0 or 1 items in machine
304
305     // schedule departure after machine processing time
306     pop_array();
307     event_schedule(PREP_TIME + sim_time + work_time[current_unit],
          EVENT_SKAUT_DEPARTURE);
308       } else {
309
310     if (list_size[number_of_machines + current_unit] == queue_size[current_unit])
           {
311
312         event_schedule(PREP_TIME + sim_time + work_time[current_unit],
              EVENT_SKAUT_ARRIVAL); //also if queue is full then delay it.
313
314     } else {
315         transfer[5] = sim_time;
316         list_file(LAST, number_of_machines + current_unit);
317         //printf("puting skaut in queue: %d\n", current_unit);
318     }
319
320       }
321
322 }
323
324 void skaut_departure()
325 {
326     push_array();
327     int current_unit = (int) transfer[3];
328     int i = 0;
329     for (i = NUM_MACHINES; i>=current_unit; i--) {  //add delay if machine is
          broken or there is a broken machine before current one
330   if (machine_broken[i] > 0.0) {
331       if ((i == current_unit)  || (list_size[1+number_of_machines +
            current_unit] < queue_size[1+current_unit])) {  // if current machine
             is broken then delay it.
332     event_schedule(sim_time + machine_broken[i], EVENT_SKAUT_DEPARTURE); //also
           if next queue is full then delay it.
333     return;
334       }
335 //       printf("Size of next queue %d, limit of next queue %d\n",list_size[1+
      number_of_machines + current_unit], queue_size[1+current_unit]);
336       break;
337   }
338     }
339
340     if (current_unit == MACHINES_ON_THE_LEFT_SIDE) {
341   skaut_throughput += 2;
342   sampst(sim_time - transfer[4], throughput_time);
343   list_remove(FIRST,current_unit);
344     } else {
345   list_remove(FIRST,current_unit);
346   pop_array();
347   transfer[3]++;
348   event_schedule(PREP_TIME + sim_time + transfer_time[(int)(transfer[3])-1],
          EVENT_SKAUT_ARRIVAL);
349     }
350
351
352     if (list_size[number_of_machines + current_unit] != 0) {
353   pop_array();
354
```

```
355     list_file(FIRST,current_unit); // first equals last because size should only
            be 1
356     pop_array();
357
358     list_remove(FIRST, number_of_machines + current_unit);
359     pop_array();
360
361     sampst(sim_time - transfer[5], sampst_delays);
362     sampst(sim_time - transfer[5], current_unit);
363     event_schedule(PREP_TIME + sim_time + work_time[current_unit],
            EVENT_SKAUT_DEPARTURE);
364       }
365 }
366
367
368 void parse_input(char inputfile_data[], char inputfile_time[])
369 {
370
371
372     if ((infile = fopen (inputfile_data, "r")) == NULL) {
373     printf("Could not open file %s\n",inputfile_data);
374     }
375
376     fscanf (infile, "%d %d %d %d %f %f %f %f %f %f", &number_of_machines, &
            min_productivity, &min_no_failures, &max_no_failures, &
            mean_wagen_arrival, &std_wagen_arrival, &min_machine_repair_time, &
            max_machine_repair_time, &end_warmup_time, &end_simulation_time);
377     fclose(infile);
378
379
380     if ((infile = fopen (inputfile_time, "r")) == NULL) {
381     printf("Could not open file %s\n",inputfile_time);
382     }
383     printf( "%d %d %d %d %f %f %f %f %f %f\n", number_of_machines,
            min_productivity, min_no_failures, max_no_failures, mean_wagen_arrival,
             std_wagen_arrival, min_machine_repair_time, max_machine_repair_time,
            end_warmup_time, end_simulation_time);
384
385     int counter = 1;
386     while (!feof(infile)) {
387     fscanf(infile, "%f %d %f", &transfer_time[counter], &queue_size[counter], &
            work_time[counter] );
388     printf("%f %d %f\n", transfer_time[counter], queue_size[counter], work_time[
            counter] );
389     counter++;
390     }
391     fclose(infile);
392
393 }
394
395 void end_warmup()
396 {
397     sampst(0.0, 0);
398     timest(0.0, 0);
399     skaut_throughput = 0;
400 }
401
402 void report()
403 {
404
405     int i;
406     float total_downtime = 0.0;
407     printf("\n*****************************************************\n");
408     printf("Report for %d failures per day\n",failure_nr);
409
410     for (i=0; i <fail_index; i++) {
411     printf("--Breakdown nr %d--\n", i+1);
412
413
414     printf("  Machine nr \t Fail time\t Downtime \t\n");
```

```
415     printf("\t %d\t", fail_list[i].machine_nr);
416     printf("%.3f\t", fail_list[i].failtime);
417     printf("%.3f sec / %.3f min\t", fail_list[i].downtime,fail_list[i].downtime
            /60.0);
418     printf("\n\n");
419     total_downtime +=fail_list[i].downtime;
420       }
421
422
423
424       printf("Total downtime was %.3lf seconds or %.3lf minutes\n",total_downtime
              , total_downtime/60.0);
425
426       printf("----------------\nMachine load\n----------------\n");
427       for (i=1; i <= number_of_machines; i++) {
428     printf("Machine %d\t", i);
429       }
430       printf("\n");
431       for (i=1; i <= number_of_machines; i++) {
432     printf("%f\t", filest(i) );
433       }
434       printf("\n\n");
435
436       printf("-------------------------\nAverage delay in queues\n
              -------------------------\n");
437       for (i=1; i <= number_of_machines; i++) {
438     printf("Queue %d \t", i);
439       }
440
441       printf("\n");
442
443       for (i=1; i <= number_of_machines; i++) {
444     printf("%f\t", sampst(0.0, -i));
445       }
446       printf("\n\n");
447       printf("Average queue delay: %f\n", sampst(0.0, -sampst_delays));
448       printf("System throughput: %d\n", skaut_throughput );
449       printf("Average throughput time: %f\n", sampst(0.0, -throughput_time));
450       printf("Min throughput time: %f\n", transfer[4]);
451       printf(": %d\n", stream);
452
453
454 }
455
456 void push_array() {
457
458     memcpy(temp_transfer,transfer,TRANSFER_ARRAY_LENGTH*sizeof(float));
459 }
460
461 void pop_array() {
462     memcpy(transfer,temp_transfer,TRANSFER_ARRAY_LENGTH*sizeof(float));
463 }
464
465 void create_machine_fail_events(int n) {
466     int i;
467     float a[20];
468     memset(a,0,20*sizeof(float));
469     float span = (float)(end_simulation_time - end_warmup_time) / (float) n
            +1.0;  //max time between machine failures
470     float current_span = 0.0;
471     int machine;
472     float repair_time ;
473     float breakdown_time;
474     for (i = 0;i<n;i++) {
475   current_span+=span;
476   machine = (int)unirand(1,number_of_machines+1,stream);
477   breakdown_time = unirand(0.0,current_span,stream);
478   repair_time =(5.0 + expon(log(max_machine_repair_time -
          min_machine_repair_time),stream))*60.0;
479   if (a[machine]<breakdown_time) {  //
```

```
480         a[machine] = breakdown_time+repair_time;
481     }
482     else { // if breakdown_time clashes with the same machine then let the
            breakdown happen after the machine goes up again
483         breakdown_time = a[machine] + 1.0;
484         a[machine] = breakdown_time+repair_time;
485     }
486     transfer[3] = repair_time;
487     transfer[4] = (float)machine;
488     fail_list[fail_index].failtime = breakdown_time+end_warmup_time;
489     fail_list[fail_index].downtime = repair_time;
490     fail_list[fail_index].machine_nr = machine;
491     fail_index++;
492     event_schedule(breakdown_time+end_warmup_time, EVENT_MACHINE_FAILURE );
493       }
494 }
495
496 void machine_failure(){
497     float repair_time = transfer[3];
498     int   machine     = (int)transfer[4];
499     machine_broken[machine] = repair_time;
500 //     printf(" Machine %d broke down and it takes %f to repair\n", machine,
        repair_time/60.0);
501
502     event_schedule(sim_time + repair_time, EVENT_MACHINE_FIXED);
503 }
504
505 void machine_fixed(){
506
507     int   machine     = (int)transfer[4];
508     machine_broken[machine] = 0.0;
509 }
```

5.2 Inntaksgögn líkans

```
1 7 404 3 10  30.0  2.0 5.0 180.0 1000.0  58600.0
2 num min min max mean  std min max warmup  simulationtime
3   prod- no  no  wagen wagen repair  repair
4   uction  fail- fail- arrival arrival time  time
5     ures  ures
```

```
1 129.83  14  70.0
2 122.82  17 21.79
3 18.98 0 12.70
4 22.74 1 67.41
5 0.1 1 69.75
6 0.1 0 81.85
7 0.1 2 70.33
```

5.3 Keyrsluskýrslur

```
1  7 404 3 10 30.000000 2.000000 5.000000 180.000000 1000.000000 58600.000000
2  129.830002 14 70.000000
3  122.820000 17 21.790001
4  18.980000 0 12.700000
5  22.740000 1 67.410004
6  0.100000 1 69.750000
7  0.100000 0 81.849998
8  0.100000 2 70.330002
9  0.000000 0 0.000000
10
11 ************************************************
12 Report for 3 failures per day
13 --Breakdown nr 1--
14   Machine nr    Fail time    Downtime
```

```
    6   3601.294   1032.031 sec / 17.201 min

--Breakdown nr 2--
  Machine nr    Fail time    Downtime
    6   5876.547   1375.157 sec / 22.919 min

--Breakdown nr 3--
  Machine nr    Fail time    Downtime
    7   13732.224 610.088 sec / 10.168 min

Total downtime was 3017.276 seconds or 50.288 minutes
----------------
Machine load
----------------
Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
0.614992   0.277149   0.218000   0.618717   0.601450   0.000000   0.000000

-------------------------
Average delay in queues
-------------------------
Queue 1    Queue 2    Queue 3    Queue 4    Queue 5    Queue 6    Queue 7
575.573996   0.000000   0.000000   0.362356   0.508830   0.000000   0.000000

Average queue delay: 114.989977
System throughput: 774
Average throughput time: 1324.542431
Min throughput time: 716.020005

**************************************************
Report for 4 failures per day
--Breakdown nr 1--
  Machine nr    Fail time    Downtime
    6   2951.004   1032.031 sec / 17.201 min

--Breakdown nr 2--
  Machine nr    Fail time    Downtime
    6   4657.474   1375.157 sec / 22.919 min

--Breakdown nr 3--
  Machine nr    Fail time    Downtime
    7   10549.334 610.088 sec / 10.168 min

--Breakdown nr 4--
  Machine nr    Fail time    Downtime
    3   6618.717   545.539 sec / 9.092 min

Total downtime was 3562.815 seconds or 59.380 minutes
----------------
Machine load
----------------
Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
0.651750   0.282788   0.221319   0.609036   0.606225   0.000000   0.000000

-------------------------
Average delay in queues
-------------------------
Queue 1    Queue 2    Queue 3    Queue 4    Queue 5    Queue 6    Queue 7
596.349629   0.000000   0.000000   0.745432   0.342073   0.000000   0.000000

Average queue delay: 119.242757
System throughput: 780
Average throughput time: 1343.736855
Min throughput time: 716.020005

**************************************************
Report for 5 failures per day
--Breakdown nr 1--
  Machine nr    Fail time    Downtime
    6   2560.831   1032.031 sec / 17.201 min

```

```
--Breakdown nr 2--
   Machine nr    Fail time    Downtime
    6   3926.030   1375.157 sec / 22.919 min

--Breakdown nr 3--
   Machine nr    Fail time    Downtime
    7   8639.600   610.088 sec / 10.168 min

--Breakdown nr 4--
   Machine nr    Fail time    Downtime
    3   5495.052   545.539 sec / 9.092 min

--Breakdown nr 5--
   Machine nr    Fail time    Downtime
    2   11851.744 545.307 sec / 9.088 min

Total downtime was 4108.123 seconds or 68.469 minutes
----------------
Machine load
----------------
Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
0.661850   0.280051   0.247659   0.608238   0.605939   0.000000   0.000000

-------------------------
Average delay in queues
-------------------------
Queue 1    Queue 2    Queue 3    Queue 4    Queue 5    Queue 6    Queue 7
572.872516   0.000000   0.000000   1.675616   0.433246   0.000000   0.000000

Average queue delay: 114.051340
System throughput: 778
Average throughput time: 1350.133985
Min throughput time: 716.020005

************************************************
Report for 6 failures per day
--Breakdown nr 1--
   Machine nr    Fail time    Downtime
    6   2300.715   1032.031 sec / 17.201 min

--Breakdown nr 2--
   Machine nr    Fail time    Downtime
    6   3438.401   1375.157 sec / 22.919 min

--Breakdown nr 3--
   Machine nr    Fail time    Downtime
    7   7366.443   610.088 sec / 10.168 min

--Breakdown nr 4--
   Machine nr    Fail time    Downtime
    3   4745.941   545.539 sec / 9.092 min

--Breakdown nr 5--
   Machine nr    Fail time    Downtime
    2   10043.276 545.307 sec / 9.088 min

--Breakdown nr 6--
   Machine nr    Fail time    Downtime
    7   56158.137 2049.004 sec / 34.150 min

Total downtime was 6157.127 seconds or 102.619 minutes
----------------
Machine load
----------------
Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
0.663490   0.264631   0.207108   0.588357   0.566059   0.000000   0.000000

-------------------------
Average delay in queues
-------------------------
```

```
Queue 1    Queue 2    Queue 3    Queue 4    Queue 5    Queue 6    Queue 7
621.410040   0.000000   0.000000   1.280168   1.225756   0.000000   0.000000

Average queue delay: 124.305862
System throughput: 728
Average throughput time: 1390.151979
Min throughput time: 716.020005

**************************************************
Report for 7 failures per day
--Breakdown nr 1--
   Machine nr    Fail time    Downtime
    6   2114.918   1032.031 sec / 17.201 min

--Breakdown nr 2--
   Machine nr    Fail time    Downtime
    6   3147.948   1375.157 sec / 22.919 min

--Breakdown nr 3--
   Machine nr    Fail time    Downtime
    7   6457.046   610.088 sec / 10.168 min

--Breakdown nr 4--
   Machine nr    Fail time    Downtime
    3   4210.863   545.539 sec / 9.092 min

--Breakdown nr 5--
   Machine nr    Fail time    Downtime
    2   8751.514   545.307 sec / 9.088 min

--Breakdown nr 6--
   Machine nr    Fail time    Downtime
    7   48279.223 2049.004 sec / 34.150 min

--Breakdown nr 7--
   Machine nr    Fail time    Downtime
    7   56746.000 353.151 sec / 5.886 min

Total downtime was 6510.277 seconds or 108.505 minutes
----------------
Machine load
----------------
Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
0.668178   0.266992   0.244490   0.593908   0.572547   0.000000   0.000000

--------------------------
Average delay in queues
--------------------------
Queue 1    Queue 2    Queue 3    Queue 4    Queue 5    Queue 6    Queue 7
634.761883   0.046638   0.000000   1.561620   0.356272   0.000000   0.000000

Average queue delay: 126.793143
System throughput: 736
Average throughput time: 1417.806163
Min throughput time: 716.020005

**************************************************
Report for 8 failures per day
--Breakdown nr 1--
   Machine nr    Fail time    Downtime
    6   1975.570   1032.031 sec / 17.201 min

--Breakdown nr 2--
   Machine nr    Fail time    Downtime
    6   3008.601   1375.157 sec / 22.919 min

--Breakdown nr 3--
   Machine nr    Fail time    Downtime
    7   5774.999   610.088 sec / 10.168 min

```

```
--Breakdown nr 4--
  Machine nr    Fail time    Downtime
    3   3809.554   545.539 sec / 9.092 min

--Breakdown nr 5--
  Machine nr    Fail time    Downtime
    2   7782.693   545.307 sec / 9.088 min

--Breakdown nr 6--
  Machine nr    Fail time    Downtime
    7   42370.039 2049.004 sec / 34.150 min

--Breakdown nr 7--
  Machine nr    Fail time    Downtime
    7   49778.598 353.151 sec / 5.886 min

--Breakdown nr 8--
  Machine nr    Fail time    Downtime
    6   56913.910 1529.975 sec / 25.500 min

Total downtime was 8040.252 seconds for 134.004 minutes
---------------
Machine load
---------------
Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
0.601259  0.256615  0.200969  0.587983  0.549249  0.000000  0.000000

-------------------------
Average delay in queues
-------------------------
Queue 1    Queue 2    Queue 3    Queue 4    Queue 5    Queue 6    Queue 7
623.912249  0.000000  0.000000  0.569042  0.622996  0.000000  0.000000

Average queue delay: 124.596773
System throughput: 706
Average throughput time: 1384.680592
Min throughput time: 716.020005

**************************************************
Report for 9 failures per day
--Breakdown nr 1--
  Machine nr    Fail time    Downtime
    6   1867.188   1032.031 sec / 17.201 min

--Breakdown nr 2--
  Machine nr    Fail time    Downtime
    6   2900.219   1375.157 sec / 22.919 min

--Breakdown nr 3--
  Machine nr    Fail time    Downtime
    7   5244.517   610.088 sec / 10.168 min

--Breakdown nr 4--
  Machine nr    Fail time    Downtime
    3   3497.425   545.539 sec / 9.092 min

--Breakdown nr 5--
  Machine nr    Fail time    Downtime
    2   7029.165   545.307 sec / 9.088 min

--Breakdown nr 6--
  Machine nr    Fail time    Downtime
    7   37774.008 2049.004 sec / 34.150 min

--Breakdown nr 7--
  Machine nr    Fail time    Downtime
    7   44359.508 353.151 sec / 5.886 min

--Breakdown nr 8--
  Machine nr    Fail time    Downtime
```

```
    6   50702.113 1529.975 sec / 25.500 min

--Breakdown nr 9--
   Machine nr    Fail time    Downtime
    7   44713.660 505.864 sec / 8.431 min

Total downtime was 8546.116 seconds or 142.435 minutes
---------------
Machine load
---------------
Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
0.590062   0.252427   0.197562   0.573658   0.540893   0.000000   0.000000

-------------------------
Average delay in queues
-------------------------
Queue 1    Queue 2    Queue 3    Queue 4    Queue 5    Queue 6    Queue 7
560.800766   0.000000   0.000000   1.226853   1.478608   0.000000   0.000000

Average queue delay: 112.958626
System throughput: 696
Average throughput time: 1336.759305
Min throughput time: 716.020005
```