Hermun skautaskála hjá Ísáli

Gunnarr Baldursson & Ragnar Gísli Ólafsson

Apríl 2011

Útdráttur

Ble! Abstract

Efnisyfirlit

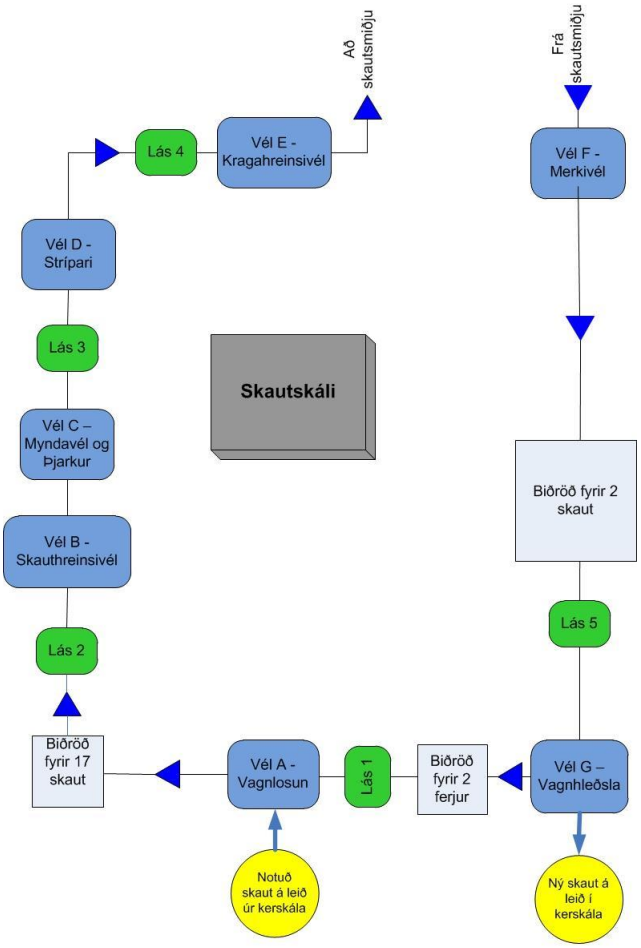
1 Inngangur

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. Þessu 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 sólarhring nemur framleiðsla um það bil 44 skautum á klukkustund. Í inntaki eru skilgreindar tvær heiltölubreytur, önnur er fyrir lágmarksfjölda bilana á sólarhring og hin fyrir hámarksfjölda bilana á sólarhring. Þessar breytur mynda því bil, og fyrir hverja heiltölu á þessu bili er hermunin framkvæmt. Í inntaki eru þessar breytur 3 og 10. Í grein 3.1 er talað um að lágmarksframleiðsla skautskála á sólarhring fyrir straumhækkun eru 404 skaut. Eftir hækkun er talan 444 skaut. Tímabilið sem hermt var yfir eru þrír mánuðir þar sem að 2 vaktir eru unnar á sólarhring og miðast niðurstöðurnar við átta bilanir á sólarhring.

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 sólarhring. 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]

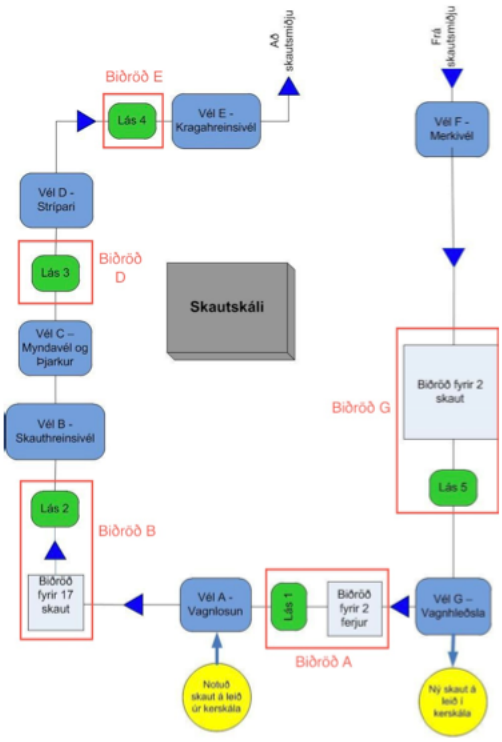
Þegar Mynd 2 er skoðuð má sjá að hægt er að skipta Skautskála upp í tvo helminga sem hefur hvor sitt inntak og sitt úttak. Inntak í vinstri helming kemur frá vél A, og úttak hans fer frá vél E. Inntak í hægri helming kemur frá vél F, en sú nálgun er gerð að þar sé ávallt nóg af nýjum skautum að taka, 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 sólarhring, 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 sólarhring. 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

```c
/*
 *  isal.c
 *
 *
 *  Created by Gunnarr Baldursson & Ragnar Gisli Olafsson on 4/18/11.
 *  Copyright 2011 Haskoli Islands. All rights reserved.
 *
 */

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include "simlib/rndlib.h"
#include "simlib/simlib.h"

// EVENTS
#define EVENT_WAGEN_UNLOAD_ARRIVAL   1
#define EVENT_WAGEN_UNLOAD_DEPARTURE 2
#define EVENT_SKAUT_ARRIVAL 3
#define EVENT_SKAUT_DEPARTURE 4
#define EVENT_MACHINE_FAILURE 5
#define EVENT_MACHINE_FIXED 6
#define EVENT_END_SIMULATION 7
#define EVENT_END_WARMUP   8
#define EVENT_GENERATE_FAILURES 9

// STREAMS
#define STREAM_WAGEN_ARRIVAL 1

//Other constants
```

```
33 #define NUM_MACHINES 7
34 #define SHIFT_LENGTH 57600.0;
35 #define WAGEN_LOAD 14
36 #define MACHINES_ON_THE_LEFT_SIDE 5
37 #define MACHINES_ON_THE_RIGHT_SIDE 2
38 #define OPTIMAL_THROUGHPUT 52
39 #define ACTUAL_THROUGHPUT 40
40 #define TRANSFER_ARRAY_LENGTH 11
41 #define PREP_TIME 20.0
42
43 typedef struct
44 {
45     float failtime;
46     float downtime;
47     int machine_nr;
48 } breakdown;
49
50
51 //#define LOADING_TIME_PER_SKAUT
52
53 // Global variables
54 int number_of_machines, min_productivity, min_no_failures, max_no_failures,
       skaut_throughput;
55 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;
56
57
58 int sampst_delays, throughput_time; // variable for queue delays and throughput
       time
59 time_t dummy;
60 unsigned int skaut_id, stream, failure_nr;
61 int queue_size[NUM_MACHINES +1];
62 float machine_broken[NUM_MACHINES +1];
63 breakdown *fail_list;
64 int fail_index;
65
66 int is_machine_busy[NUM_MACHINES +1],
67     queue_size[NUM_MACHINES +1];
68
69 float work_time[NUM_MACHINES + 1],
70     transfer_time[NUM_MACHINES +1]; // +1 is the less preferable simlib
          indexing scheme
71
72
73 float temp_transfer[TRANSFER_ARRAY_LENGTH];
74
75 FILE *infile, *outfile;
76
77 /* Function signatures */
78
79 // Usage: create_machine_fail_events()
80 // Pre:   init_twister must be called for random number generation
81 // Post:  scheduled events have been created for machines
82 void create_machine_fail_events();
83
84
85 // Usage: push_array();
86 // Pre:   we expect that correct values are in transfer array
87 // Post:  our temp_transfer array now has the values in transfer_array
88 void push_array();
89
90 // Usage: pop_array();
91 // Pre:   we expect that correct values are in transfer_temp array
92 // Post:  our transfer array now has the values in transfer_temp
93 void pop_array();
94
95 // Usage: wagen_arrival();
96 // Pre:   EVENT_WAGEN_UNLOAD_ARRIVAL is the next event to be processed
```

```
97 // Post:  14 EVENT_SKAUT_ARRIVAL events are next to be processed on the event
       list.
98 void wagen_unload_arrival();
99
100 // Usage: skaut_arrival();
101 // Pre:   EVENT_SKAUT_ARRIVAL is the next event to be processed
102 // Post:  a skaut has been processed by a machine or put in it's queue.
103 //       subsequent events may have been scheduled
104 void skaut_arrival();
105
106 // Usage: skaut_departure();
107 // Pre:   EVENT_SKAUT_DEPARTURE is the next event to be processed
108 // Post:
109 void skaut_departure(); // do we need an event for departure?
110
111 // Usage: machine_failure();
112 // Pre:   EVENT_MACHINE_FAILURE is the next event to be processed
113 // Post:
114 void machine_failure();
115
116 // Usage: machine_fixed();
117 // Pre:   EVENT_MACHINE_FIXED is the next event to be processed
118 // Post:
119 void machine_fixed();
120
121 // Usage: end_warmup();
122 // Post:  SIMLIB statistical variables have been cleared
123 void end_warmup();
124
125 // Usage: parse_input(input_filename_data,input_filename_time);
126 // Pre:   input_filename_data,input_filename_time of type char[],
127 //       global variables from the input file exist.
128 // Post:  the global variables were assigned values from input_filename,
129 //
130 void parse_input(char[] ,char[]);
131
132 // Usage: x = N(muy, sigma, stream);
133 // Pre:   muy and sigma are of type float
134 //       stream is of type int
135 // Post:  x is a random gaussian distributed variable of type float
136 //       with mean muy and std sigma
137 float N(float muy, float sigma, int stream);
138
139 // Usage: report("the_report.out");
140 // Pre:   the values to be reported have values
141 // Post:  a report on program values and simlib statistics
142 //       have been APPENDED to "the_report.out"
143 void report();
144
145 // Usage: schedule_failures(i);
146 // Pre:   the global variable end_simulation_time has a value, i is of type int
147 // Post:  i failures have been scheduled uniformly on machines
148 //       with ?random? repair times on the interval [min_machine_repair_time,...
      max_machine_repair_time]
149 //       uniformly distributed over the interval 0...end_simulation_time
150 void schedule_failures(int i);
151
152 void queue_is_full();
153
154 int main()
155 {
156 // load datafiles
157     parse_input("adal_inntak.in","velar_og_bidradir.in");
158
159     // initialize arrays and variables
160     if((fail_list = malloc(sizeof(breakdown)*max_no_failures))==NULL) {
161   printf("Allocation Error\n");
162   exit(1);
163     }
164
```

```
165
166
167     int b;
168 /*     for (b=1; b <= number_of_machines; b++) {
169   printf("transfer_time[%d] = %f\n", b,transfer_time[b] );
170   printf("busy %d broken %f \n",is_machine_busy[b],machine_broken[b]);
171   }*/
172     // We perform simulation for "a few" failures per day
173
174     for (failure_nr = min_no_failures; failure_nr<= max_no_failures; failure_nr
          ++) {
175   stream = (unsigned int)time(NULL) % 100;
176
177   memset( is_machine_busy,0, NUM_MACHINES +1 );
178   memset( machine_broken,0, NUM_MACHINES +1);
179   memset( fail_list ,0, sizeof(breakdown)*NUM_MACHINES+1);
180   fail_index = 0;
181   skaut_throughput = 0;
182   sampst_delays = number_of_machines +1;
183   throughput_time = number_of_machines +2;
184
185
186   skaut_id = 1;
187   skaut_throughput = 0;
188
189
190   // Initialize rndlib
191   init_twister();
192
193   // Initialize simlib
194   init_simlib();
195
196   maxatr = 6; // how many attributes do we need?
197
198   /* Schedule first wagen arrival */
199   event_schedule( 10.0, EVENT_WAGEN_UNLOAD_ARRIVAL );
200
201   /* Schedule end of warmup time */
202   event_schedule( end_warmup_time, EVENT_END_WARMUP );
203   event_schedule(end_warmup_time, EVENT_GENERATE_FAILURES );
204   /* Schedule simulation termination */
205   event_schedule(end_simulation_time , EVENT_END_SIMULATION );
206
207   next_event_type = 0;
208
209
210
211   while (next_event_type != EVENT_END_SIMULATION) {
212
213       timing();
214       /*                    printf("event_type = %d, transfer[3] = %f\n",
          next_event_type, transfer[3]);
215           int k;
216           for (k = 1; k <= number_of_machines; k++)
217           printf("Items in machines/queues %d:  %d, %d\n", k, list_size[k],
                list_size[number_of_machines +k]);
218           printf("\n");
219       */
220
221
222       switch (next_event_type) {
223       case EVENT_WAGEN_UNLOAD_ARRIVAL:
224     wagen_unload_arrival();
225     break;
226       case EVENT_SKAUT_ARRIVAL:
227     skaut_arrival();
228     break;
229       case EVENT_SKAUT_DEPARTURE:
230     skaut_departure();
231     break;
```

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

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

```
353     event_schedule(PREP_TIME + sim_time + transfer_time[(int)(transfer[3]) -1],
            EVENT_SKAUT_ARRIVAL);
354       }
355
356
357       if (list_size[number_of_machines + current_unit] != 0) {
358     pop_array();
359
360     list_file(FIRST,current_unit); // first equals last because size should only
            be 1
361     pop_array();
362
363     list_remove(FIRST, number_of_machines + current_unit);
364     pop_array();
365
366     sampst(sim_time - transfer[5], sampst_delays);
367     sampst(sim_time - transfer[5], current_unit);
368     event_schedule(PREP_TIME + sim_time + work_time[current_unit],
            EVENT_SKAUT_DEPARTURE);
369       }
370 }
371
372
373 void parse_input(char inputfile_data[], char inputfile_time[])
374 {
375
376
377       if ((infile = fopen (inputfile_data, "r")) == NULL) {
378     printf("Could not open file %s\n",inputfile_data);
379       }
380
381       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);
382       fclose(infile);
383
384
385       if ((infile = fopen (inputfile_time, "r")) == NULL) {
386     printf("Could not open file %s\n",inputfile_time);
387       }
388       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);
389
390       int counter = 1;
391       while (!feof(infile)) {
392     fscanf(infile, "%f %d %f", &transfer_time[counter], &queue_size[counter], &
            work_time[counter] );
393     printf("%f %d %f\n", transfer_time[counter], queue_size[counter], work_time[
            counter] );
394     counter++;
395       }
396       fclose(infile);
397
398 }
399
400 void end_warmup()
401 {
402       sampst(0.0, 0);
403       timest(0.0, 0);
404       skaut_throughput = 0;
405 }
406
407 void report()
408 {
409
410       int i;
411       float total_downtime = 0.0;
```

```
412     printf("\n**************************************************\n");
413     printf("Report for %d failures per day\n",failure_nr);
414
415     for (i=0; i <NUM_MACHINES; i++) {
416   printf("--Breakdown nr %d--\n", i+1);
417   printf("Number of fails\t Downtime \t\n");
418   printf("\t %d\t", fail_list[i].machine_nr);
419   printf("%.3f sec / %.3f min\t", fail_list[i].downtime,fail_list[i].downtime
        /60.0);
420   printf("\n");
421   total_downtime+=fail_list[i].downtime;
422     }
423     printf("\n\n");
424
425
426     printf("Total downtime was %.3lf seconds or %.3lf minutes\n",total_downtime
          , total_downtime/60.0);
427
428     printf("----------------\nMachine load\n----------------\n");
429     for (i=1; i <= number_of_machines; i++) {
430   printf("Machine %d\t", i);
431     }
432     printf("\n");
433     for (i=1; i <= number_of_machines; i++) {
434   printf("%f\t", filest(i) );
435     }
436     printf("\n\n");
437
438     printf("------------------------\nAverage delay in queues\n
          ------------------------\n");
439     for (i=1; i <= number_of_machines; i++) {
440   printf("Queue %d \t", i);
441     }
442
443     printf("\n");
444
445     for (i=1; i <= number_of_machines; i++) {
446   printf("%f\t", sampst(0.0, -i));
447     }
448     printf("\n\n");
449     printf("Average queue delay: %f\n", sampst(0.0, -sampst_delays));
450     printf("System throughput: %d\n", skaut_throughput );
451     printf("Average throughput time: %f\n", sampst(0.0, -throughput_time));
452     printf("Min throughput time: %f\n", transfer[4]);
453     printf("Random seed: %d\n", stream);
454
455
456 }
457
458 void push_array() {
459
460     memcpy(temp_transfer,transfer,TRANSFER_ARRAY_LENGTH*sizeof(float));
461 }
462
463 void pop_array() {
464     memcpy(transfer,temp_transfer,TRANSFER_ARRAY_LENGTH*sizeof(float));
465 }
466
467 void create_machine_fail_events() {
468     int i;
469     float a[20],shift_length;
470     shift_length = (float)SHIFT_LENGTH;
471     int n = failure_nr;
472     memset(a,0,20*sizeof(float));
473     float span = shift_length / (float)n+1.0;  //max time between machine
          failures
474     float current_span = 0.0;
475     int machine;
476     float repair_time ;
477     float breakdown_time;
```

```
478
479     for (i = 0;i<n;i++) {
480   current_span+=span;
481   machine = (int)unirand(1,number_of_machines+1,stream);
482   breakdown_time = unirand(0.0,current_span,stream);
483   repair_time =(5.0 + expon(log(max_machine_repair_time -
          min_machine_repair_time),stream))*60.0;
484   if (a[machine]<breakdown_time) {  //
485       a[machine] = breakdown_time+repair_time;
486   }
487   else { // if breakdown_time clashes with the same machine then let the
          breakdown happen after the machine goes up again
488       breakdown_time = a[machine] + 1.0;
489       a[machine] = breakdown_time+repair_time;
490   }
491   transfer[3] = repair_time;
492   transfer[4] = (float)machine;
493   fail_list[machine-1].downtime+= repair_time;
494   fail_list[machine-1].machine_nr++;
495   event_schedule(sim_time + breakdown_time, EVENT_MACHINE_FAILURE );
496     }
497
498     event_schedule(sim_time + shift_length, EVENT_GENERATE_FAILURES );
499 }
500
501 void machine_failure(){
502     float repair_time = transfer[3];
503     int   machine     = (int)transfer[4];
504     machine_broken[machine] = repair_time;
505 //    printf(" Machine %d broke down and it takes %f to repair\n", machine,
      repair_time/60.0);
506
507     event_schedule(sim_time + repair_time, EVENT_MACHINE_FIXED);
508 }
509
510 void machine_fixed(){
511
512     int   machine     = (int)transfer[4];
513     machine_broken[machine] = 0.0;
514 }
```

5.2 Inntaksgögn líkans

```
7 404 3 10 30.0 2.0 5.0 180.0 1000.0 5185000.0
num min min max mean std min max warmup simulationtime
prod- no no wagen wagen repair repair
uction fail- fail- arrival arrival time time
ures ures
```

```
129.83  14  70.0
122.82  17 21.79
18.98 0 12.70
22.74 1 67.41
0.1 1 69.75
0.1 0 81.85
0.1 2 70.33
```

5.3 Keyrsluskýrslur

```
7 404 3 10 30.000000 2.000000 5.000000 180.000000 1000.000000 5185000.000000
129.830002 14 70.000000
122.820000 17 21.790001
18.980000 0 12.700000
22.740000 1 67.410004
0.100000 1 69.750000
```

```
0.100000 0 81.849998
0.100000 2 70.330002
0.000000 0 0.000000

**************************************************
Report for 3 failures per day
--Breakdown nr 1--
Number of fails  Downtime
    35 19672.229 sec / 327.870 min
--Breakdown nr 2--
Number of fails  Downtime
    37 23301.904 sec / 388.365 min
--Breakdown nr 3--
Number of fails  Downtime
    39 22888.336 sec / 381.472 min
--Breakdown nr 4--
Number of fails  Downtime
    33 19278.080 sec / 321.301 min
--Breakdown nr 5--
Number of fails  Downtime
    42 24431.441 sec / 407.191 min
--Breakdown nr 6--
Number of fails  Downtime
    36 20377.275 sec / 339.621 min
--Breakdown nr 7--
Number of fails  Downtime
    48 28950.250 sec / 482.504 min


Total downtime was 158899.500 seconds or 2648.325 minutes
----------------
Machine load
----------------
Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
0.654763  0.296186  0.235536  0.628533  0.636136  0.000000  0.000000

------------------------
Average delay in queues
------------------------
Queue 1   Queue 2   Queue 3   Queue 4   Queue 5   Queue 6   Queue 7
558.539059  0.000000  0.000000  0.752615  0.921827  0.000000  0.000000

Average queue delay: 112.032943
System throughput: 73152
Average throughput time: 1313.980430
Min throughput time: 716.020005
Random seed: 80

**************************************************
Report for 4 failures per day
--Breakdown nr 1--
Number of fails  Downtime
    50 27900.582 sec / 465.010 min
--Breakdown nr 2--
Number of fails  Downtime
    55 30530.309 sec / 508.838 min
--Breakdown nr 3--
Number of fails  Downtime
    48 27365.381 sec / 456.090 min
--Breakdown nr 4--
Number of fails  Downtime
    50 28820.166 sec / 480.336 min
--Breakdown nr 5--
Number of fails  Downtime
    47 33093.066 sec / 551.551 min
--Breakdown nr 6--
Number of fails  Downtime
    49 29884.803 sec / 498.080 min
--Breakdown nr 7--
Number of fails  Downtime
```

```
   61 37425.289 sec / 623.755 min


Total downtime was 215019.594 seconds or 3583.660 minutes
----------------
Machine load
----------------
Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
0.655876  0.293562  0.235888  0.624951  0.630135  0.000000  0.000000

-------------------------
Average delay in queues
-------------------------
Queue 1   Queue 2   Queue 3   Queue 4   Queue 5   Queue 6   Queue 7
568.505439  0.000000  0.000000  0.711261  0.930494  0.000000  0.000000

Average queue delay: 114.024413
System throughput: 72350
Average throughput time: 1317.489711
Min throughput time: 716.020005
Random seed: 80

**************************************************
Report for 5 failures per day
--Breakdown nr 1--
Number of fails  Downtime
   60 33213.066 sec / 553.551 min
--Breakdown nr 2--
Number of fails  Downtime
   74 40202.219 sec / 670.037 min
--Breakdown nr 3--
Number of fails  Downtime
   67 39338.594 sec / 655.643 min
--Breakdown nr 4--
Number of fails  Downtime
   55 32902.844 sec / 548.381 min
--Breakdown nr 5--
Number of fails  Downtime
   64 36943.953 sec / 615.733 min
--Breakdown nr 6--
Number of fails  Downtime
   67 40575.629 sec / 676.260 min
--Breakdown nr 7--
Number of fails  Downtime
   63 38466.336 sec / 641.106 min


Total downtime was 261642.625 seconds or 4360.710 minutes
----------------
Machine load
----------------
Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
0.654380  0.290900  0.234669  0.621559  0.625087  0.000000  0.000000

-------------------------
Average delay in queues
-------------------------
Queue 1   Queue 2   Queue 3   Queue 4   Queue 5   Queue 6   Queue 7
569.814654  0.001096  0.000000  1.074564  1.292770  0.000000  0.000000

Average queue delay: 114.433455
System throughput: 71620
Average throughput time: 1334.215767
Min throughput time: 716.020005
Random seed: 81

**************************************************
Report for 6 failures per day
--Breakdown nr 1--
Number of fails  Downtime
```

```
   73 44437.770 sec / 740.629 min
--Breakdown nr 2--
Number of fails  Downtime
   74 43240.008 sec / 720.667 min
--Breakdown nr 3--
Number of fails  Downtime
   73 46937.207 sec / 782.287 min
--Breakdown nr 4--
Number of fails  Downtime
   88 53451.793 sec / 890.863 min
--Breakdown nr 5--
Number of fails  Downtime
   80 48910.336 sec / 815.172 min
--Breakdown nr 6--
Number of fails  Downtime
   81 48909.609 sec / 815.160 min
--Breakdown nr 7--
Number of fails  Downtime
   71 41589.074 sec / 693.151 min


Total downtime was 327475.812 seconds or 5457.930 minutes
---------------
Machine load
---------------
Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
0.656314  0.287679  0.235063  0.621714  0.619336  0.000000  0.000000

-------------------------
Average delay in queues
-------------------------
Queue 1   Queue 2   Queue 3   Queue 4   Queue 5   Queue 6   Queue 7
579.043716  0.001564  0.000000  1.281874  1.592186  0.000000  0.000000

Average queue delay: 116.386480
System throughput: 70848
Average throughput time: 1368.833296
Min throughput time: 716.020005
Random seed: 81

**************************************************
Report for 7 failures per day
--Breakdown nr 1--
Number of fails  Downtime
   101  56887.988 sec / 948.133 min
--Breakdown nr 2--
Number of fails  Downtime
   76 46014.801 sec / 766.913 min
--Breakdown nr 3--
Number of fails  Downtime
   81 47521.797 sec / 792.030 min
--Breakdown nr 4--
Number of fails  Downtime
   106  64224.965 sec / 1070.416 min
--Breakdown nr 5--
Number of fails  Downtime
   81 44091.891 sec / 734.865 min
--Breakdown nr 6--
Number of fails  Downtime
   91 55643.684 sec / 927.395 min
--Breakdown nr 7--
Number of fails  Downtime
   94 54704.039 sec / 911.734 min


Total downtime was 369089.156 seconds or 6151.486 minutes
---------------
Machine load
---------------
Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
```

```
0.656378   0.285759   0.231099   0.618292   0.614869   0.000000   0.000000

-------------------------
Average delay in queues
-------------------------
Queue 1    Queue 2    Queue 3    Queue 4    Queue 5    Queue 6    Queue 7
586.661686   0.000501   0.000000   1.524263   1.595263   0.000000   0.000000

Average queue delay: 117.950359
System throughput: 70246
Average throughput time: 1377.780016
Min throughput time: 716.020005
Random seed: 81

**************************************************
Report for 8 failures per day
--Breakdown nr 1--
Number of fails  Downtime
    110  71093.250 sec / 1184.888 min
--Breakdown nr 2--
Number of fails  Downtime
    101  62205.086 sec / 1036.751 min
--Breakdown nr 3--
Number of fails  Downtime
    111  65479.945 sec / 1091.332 min
--Breakdown nr 4--
Number of fails  Downtime
    100  61106.328 sec / 1018.439 min
--Breakdown nr 5--
Number of fails  Downtime
    113  70937.930 sec / 1182.299 min
--Breakdown nr 6--
Number of fails  Downtime
    98 56875.254 sec / 947.921 min
--Breakdown nr 7--
Number of fails  Downtime
    87 52393.074 sec / 873.218 min


Total downtime was 440090.844 seconds or 7334.847 minutes
---------------
Machine load
---------------
Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
0.656147   0.281196   0.232280   0.612183   0.606929   0.000000   0.000000

-------------------------
Average delay in queues
-------------------------
Queue 1    Queue 2    Queue 3    Queue 4    Queue 5    Queue 6    Queue 7
593.587457   0.005425   0.000000   1.776352   1.885498   0.000000   0.000000

Average queue delay: 119.439326
System throughput: 69212
Average throughput time: 1393.676346
Min throughput time: 716.020005
Random seed: 81

**************************************************
Report for 9 failures per day
--Breakdown nr 1--
Number of fails  Downtime
    117  69005.820 sec / 1150.097 min
--Breakdown nr 2--
Number of fails  Downtime
    126  74216.391 sec / 1236.940 min
--Breakdown nr 3--
Number of fails  Downtime
    107  69218.562 sec / 1153.643 min
--Breakdown nr 4--
```

```
Number of fails  Downtime
   125  74980.672 sec / 1249.678 min
--Breakdown nr 5--
Number of fails  Downtime
   103  61685.805 sec / 1028.097 min
--Breakdown nr 6--
Number of fails  Downtime
   104  67950.938 sec / 1132.516 min
--Breakdown nr 7--
Number of fails  Downtime
   128  77454.203 sec / 1290.903 min


Total downtime was 494512.375 seconds or 8241.873 minutes
----------------
Machine load
----------------
Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
0.657259  0.281046  0.232867  0.614676  0.602645  0.000000  0.000000

------------------------
Average delay in queues
------------------------
Queue 1   Queue 2   Queue 3   Queue 4   Queue 5   Queue 6   Queue 7
596.864645  0.008671  0.000000  1.903298  1.707945  0.000000  0.000000

Average queue delay: 120.099683
System throughput: 68826
Average throughput time: 1399.454525
Min throughput time: 716.020005
Random seed: 82

**************************************************
Report for 10 failures per day
--Breakdown nr 1--
Number of fails  Downtime
   144  89020.391 sec / 1483.673 min
--Breakdown nr 2--
Number of fails  Downtime
   128  78905.758 sec / 1315.096 min
--Breakdown nr 3--
Number of fails  Downtime
   120  74155.375 sec / 1235.923 min
--Breakdown nr 4--
Number of fails  Downtime
   127  75887.453 sec / 1264.791 min
--Breakdown nr 5--
Number of fails  Downtime
   130  74679.445 sec / 1244.657 min
--Breakdown nr 6--
Number of fails  Downtime
   106  65326.320 sec / 1088.772 min
--Breakdown nr 7--
Number of fails  Downtime
   145  88014.719 sec / 1466.912 min


Total downtime was 545989.500 seconds or 9099.825 minutes
----------------
Machine load
----------------
Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
0.659111  0.279224  0.232834  0.608286  0.596649  0.000000  0.000000

------------------------
Average delay in queues
------------------------
Queue 1   Queue 2   Queue 3   Queue 4   Queue 5   Queue 6   Queue 7
606.011902  0.008728  0.000000  2.021353  2.104044  0.000000  0.000000

```

```
357 Average queue delay: 122.028500
358 System throughput: 68096
359 Average throughput time: 1421.318794
360 Min throughput time: 716.020005
361 Random seed: 82
```