Semestrálna práca č. 2
Stolárska dielňa Najlepší nábytok, s.r.o

Vypracoval: Marek Magula

Študijná skupina: 5ZIS11

Predmet: Diskrétna simulácia

Cvičiaci: Ing. Peter Jankovič, PhD

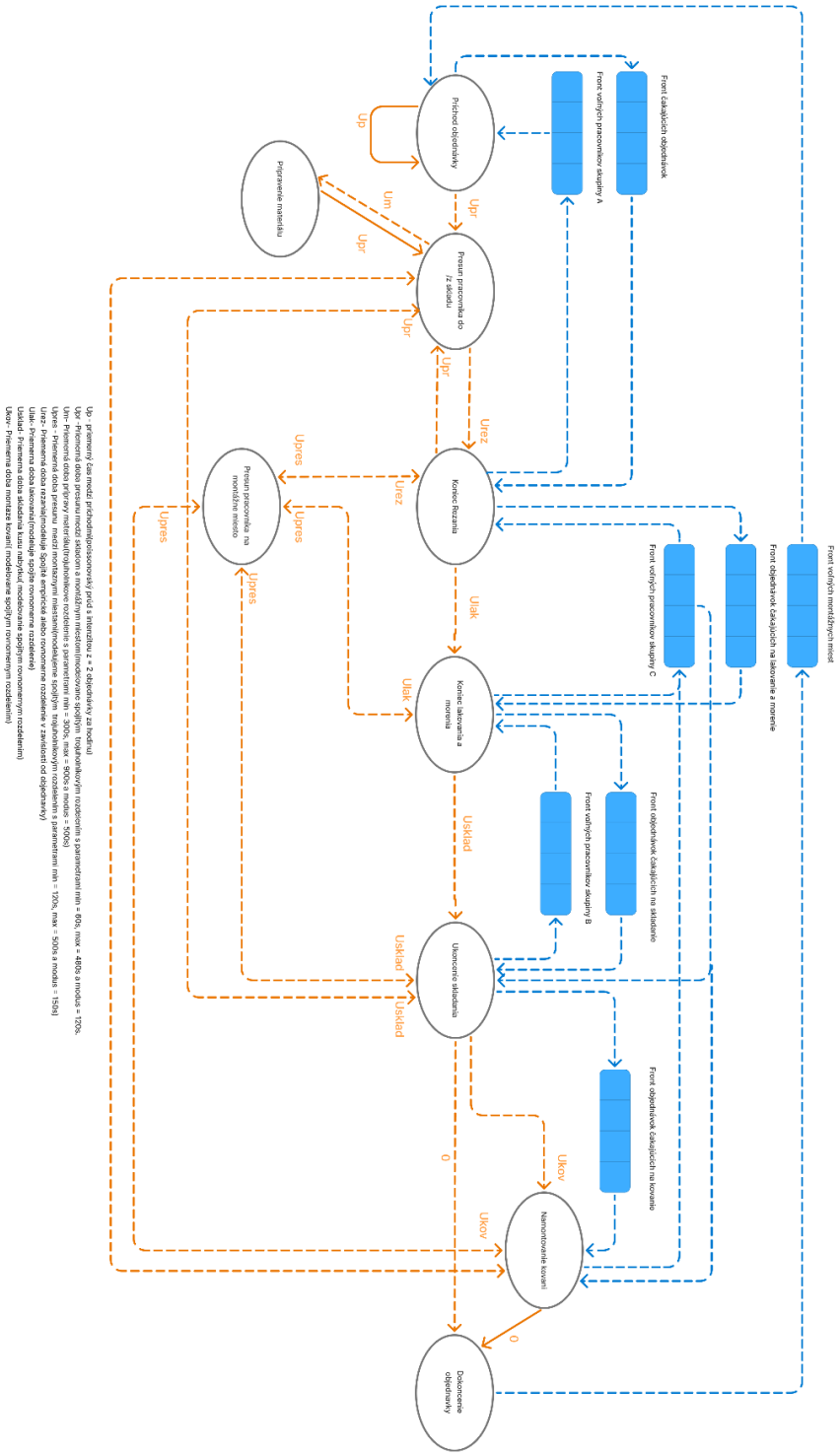
Obsah

Zoznam tabuliek

Zoznam obrázkov

Zoznam príloh

Udalostný diagram



Obrázok 1 Udalostný diagram

Popis implementovaných udalostí

Príchod objednávky(OrderArrival)

Udalosť modeluje príchod objednávky. Udalosť vykonáva nasledovnú postupnosť krokov:

1. Vytvorí novú objednávku
2. Skontroluje dostupnosť pracovníkov skupiny A.
3. Ak je voľný pracovník skupiny A:
    a. Ak je v dielni voľné montážne miesto priradí ho objednávke.
    b. Ak nie je voľné montážne miesto vytvorí nové montážne miesto a priradí ho objednávke
    c. Na základe aktuálnej pozície pracovníka(sklad / montážne miesto) naplánuje udalosť presunu(Presun pracovníka z skladu / Presun pracovníka na montážne miesto) pracovníka na montážne miesto priradené objednávke.
4. Ak nie je voľný žiadny pracovník skupiny A:
    a. Zaradí objednávku do fronty čakajúcich objednávok
5. Naplánuje ďalšiu udalosť príchodu novej objednávky

Presun pracovníka do/z skladu(MoveToStorage)

Udalosť modeluje presun pracovníka medzi skladom a montážnym miestom v oboch smeroch.

Udalosť obdrží z vstupných parametrov pracovníka a objednávku. Skontroluje aktuálny stav objednávky a na základe tohto stavu naplánuje ďalší technologický krok potrebný pre objednávku(Prípravu materiálu,Rezanie, Lakovanie, Skladanie, Montáž Kovaní) .

Presun pracovníka na montážne miesto(MoveToStation)

Udalosť modeluje presun pracovníka medzi dvomi montážnymi miestami v dielni.

Udalosť obdrží z vstupných parametrov pracovníka a objednávku. Skontroluje aktuálny stav objednávky a na základe tohto stavu naplánuje ďalší technologický krok potrebný pre objednávku(Lakovanie, Skladanie, Montáž Kovaní) .

Príprava materiálu(PrepareMaterial)

Udalosť modeluje prípravu potrebného materiálu v sklade pre vyhotovenie objednávky. Po pripravení potrebného materiálu udalosť naplánuje presun pracovníka s pripraveným materiálom zo skladu k montážnemu miestu kde pracovník materiál nareže.

Koniec rezania(CuttingEnd)

Udalosť modeluje vykonanie technologického kroku rezania materiálu z ktorého bude objednávka pozostávať. Po dokončení rezania materiálu pre objednávku udalosť skontroluje, či sa vo fronte čakajúcich objednávok nachádza objednávka. Ak áno, tak udalosť čakajúcej objednávke priradí montážne miesto rovnakým spôsobom ako v udalosti príchodu objednávky a zamestnancovi, ktorý práve dokončil rezanie objednávky rovno naplánuje udalosť presunu do skladu, kde bude opäť

pripravovať materiál pre danú objednávku. Ak vo fronte nečaká žiadna objednávka, tak je tento pracovník zaradený do frontu voľných pracovníkov skupiny A.

Následne udalosť kontroluje dostupnosť pracovníkov skupiny C. Ak existuje voľný pracovník skupiny C tak:

1. Prioritne skontroluje, či vo fronte objednávok čakajúcich na montáž kovaní čaká objednávka
2. Ak vo fronte čaká objednávka tak:
    a. Na základe aktuálnej pozície voľného pracovníka skupiny C naplánuje udalosť presunu pracovníka na montážne miesto na ktorom sa nachádza čakajúca objednávka
3. Ak vo fronte nečaká žiadna objednávka, tak objednávka ktorá sa vrámci tejto udalosti dorezala bude posunutá na lakovanie a voľnému pracovníkovi skupiny C sa na základe jeho aktuálnej pozície naplánuje presun na montážne miesto, kde sa táto objednávka nachádza. Ak sa pracovník už nachádza na danom montážnom mieste tak rovno začne proces lakovania a morenia.

Ak neexistuje dostupný pracovník skupiny C tak je dorezaná objednávka zaradená do fronty objednávok čakajúcich na lakovanie a morenie.

Koniec lakovania a morenia(VarnishingEnd)

Udalosť modeluje vykonanie technologického kroku lakovania a morenia materiálu z ktorého bude objednávka pozostávať. Po dokončení lakovania a morenia materiálu pre objednávku udalosť skontroluje, či sa vo fronte čakajúcich objednávok na montáž kovaní nachádza objednávka. Ak áno tak túto objednávku pracovník ktorý práve dokončil lakovanie a morenie začne rovno spracovať a presunie sa na montážne miesto, kde sa nachádza čakajúca objednávka. Ak vo fronte nečaká žiadna objednávka, tak sa skontroluje front objednávok čakajúcich na lakovanie a morenie, ak sa v ňom nachádza objednávka tak ju tento pracovník začne spracovávať a presunie sa na montážne miesto, kde sa nachádza čakajúca objednávka. Ak ani v tomto fronte nečaká žiadna objednávka tak je tento pracovník pridaný do fronty voľných pracovníkov skupiny C.

Následne udalosť kontroluje dostupnosť pracovníkov skupiny B. Ak existuje voľný pracovník skupiny B tak tento pracovník začne skladať objednávku ktorá bola práve nalakovaná a namorená a v závislosti od aktuálnej pozície voľného pracovníka buď rovno začne objednávku skladať, alebo sa naplánuje udalosť jeho presunu na správne montážne miesto a následne začne objednávku skladať. Ak neexistuje voľný pracovník skupiny B, tak je nalakovaná objednávka priradená do frontu objednávok čakajúcich na skladanie.

Ukončenie skladania(AssemblyEnd)

Udalosť modeluje vykonanie technologického kroku skladania objednávky. Po dokončení skladania udalosť skontroluje či vo fronte objednávok čakajúcich na skladanie čaká objednávka. Ak áno tak sa pracovník, ktorý práve doskladal objednávku presunie na montážne miesto, kde sa táto objednávka nachádza a následne ju poskladá. Ak je front prázdny tak je tento pracovník zaradený do frontu voľných pracovníkov skupiny B.

Následne udalosť kontroluje typ objednávky. Pokiaľ sa jedná o skriňu tak skontroluje dostupnosť pracovníkov skupiny C. Ak je dostupný pracovník skupiny C, tak začne na skriňu montovať kovania alebo sa presunie na dané montážne miesto ak sa tam ešte nenachádza. Ak nie je dostupný žiadny pracovník skupiny C tak je táto objednávka zaradená do frontu objednávok čakajúcich na montáž kovaní.

Pokiaľ sa nejedná o skriňu tak je objednávka dokončená.

Namontovanie kovaní(Fitting)

Udalosť modeluje vykonanie technologického kroku montáže kovaní. Po dokončení montáže kovaní na skriňu je objednávka dokončená a udalosť skontroluje, či sa vo fronte objednávok čakajúcich na montáž kovaní nachádza objednávka. Ak áno, tak sa pracovník ktorý práve dokončil montáž kovania presunie na montážne miesto na ktorom sa nachádza čakajúca objednávka a začne s montážou kovaní. Ak v tejto fronte nečaká žiadna objednávka, tak sa skontroluje front objednávok čakajúcich na lakovanie a morenie. Ak sa v tejto fronte nachádza čakajúca objednávka, tak sa pracovník ktorý práve dokončil montáž kovania presunie na montážne miesto na ktorom sa nachádza čakajúca objednávka a začne s lakovaním. Ak v tejto fronte nečaká žiadna objednávka, tak je pracovník umiestnený do frontu voľných pracovníkov skupiny C.

Dokončenie objednávky(FinalizeOrder)

Udalosť predstavuje dokončenie procesu výroby objednávky. Montážne miesto na ktorom sa objednávka spracovávala je uvoľnené.

Generátory

V rámci semestrálnej práce sme implementovali generátor pre trojuholníkové rozdelenie a generátor pre exponenciálne rozdelenie.

TriangularGenerator

Trieda TriangularGenerator generuje náhodné hodnoty podľa trojuholníkového rozdelenia pravdepodobnosti, definovaného minimom (min), maximom (max) a najpravdepodobnejšou hodnotou (mode).

Metóda getSample()

Metóda vráti náhodné číslo z trojuholníkového rozdelenia určeného parametrami min, max a mode získanými v konštruktore.

```
public Double getSample() {
    double rand = probabilityGenerator.nextDouble();
    double c = (mode - min) / (max - min);
    if (rand < c) {
      return min + Math.sqrt(rand * (max - min) * (mode - min));
    } else {
      return max - Math.sqrt((1 - rand) * (max - min) * (max - mode));
    }
  }
```

Priloha 1Metóda getSample generátora trojuholníkového rozdelenia

ExponentialGenerator

Trieda ExponentialGenerator generuje náhodné hodnoty podľa exponenciálneho rozdelenia pravdepodobnosti so zadanou strednou hodnotou (mean).

Metóda getSample()

Metóda vráti náhodné čislo z exponenciálneho rozdelenia určeného parametrom mean získaným v konštruktore.

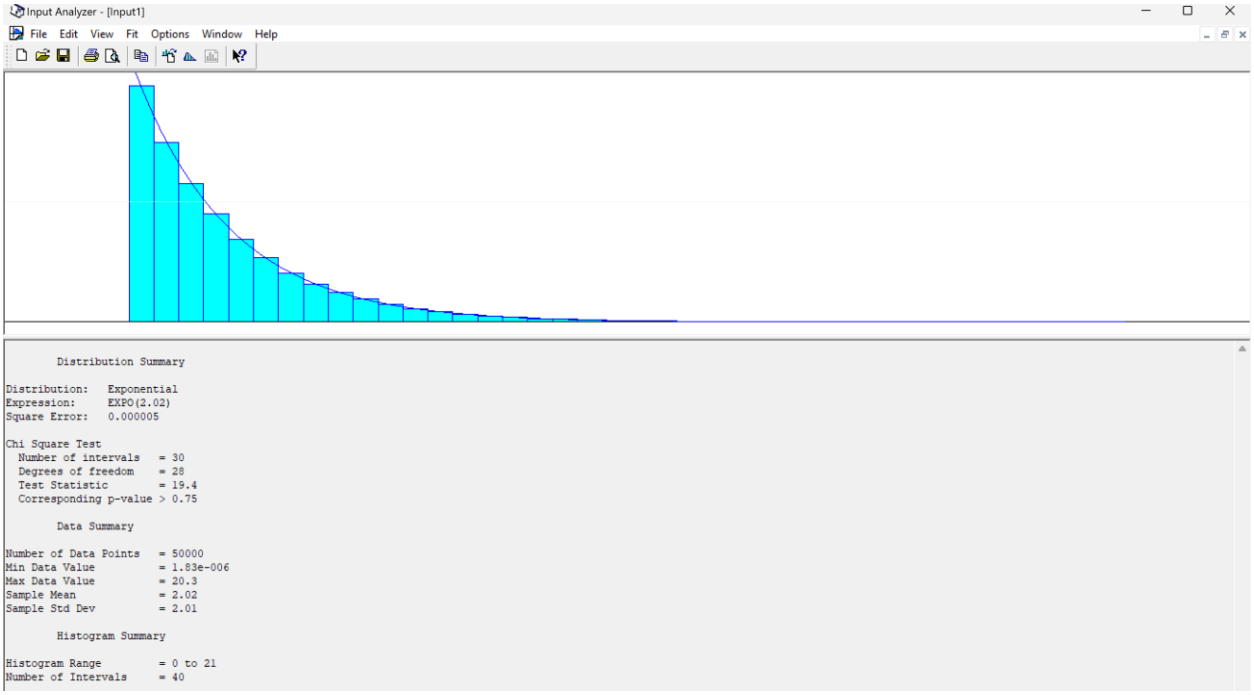
```
public Double getSample() {
    double rand = probabilityGenerator.nextDouble();
    return -Math.log(1 - rand) / lambda;
  }
```

Priloha 2 Metóda getSample generátora exponenciálneho rozdelenia

Testovanie generátorov

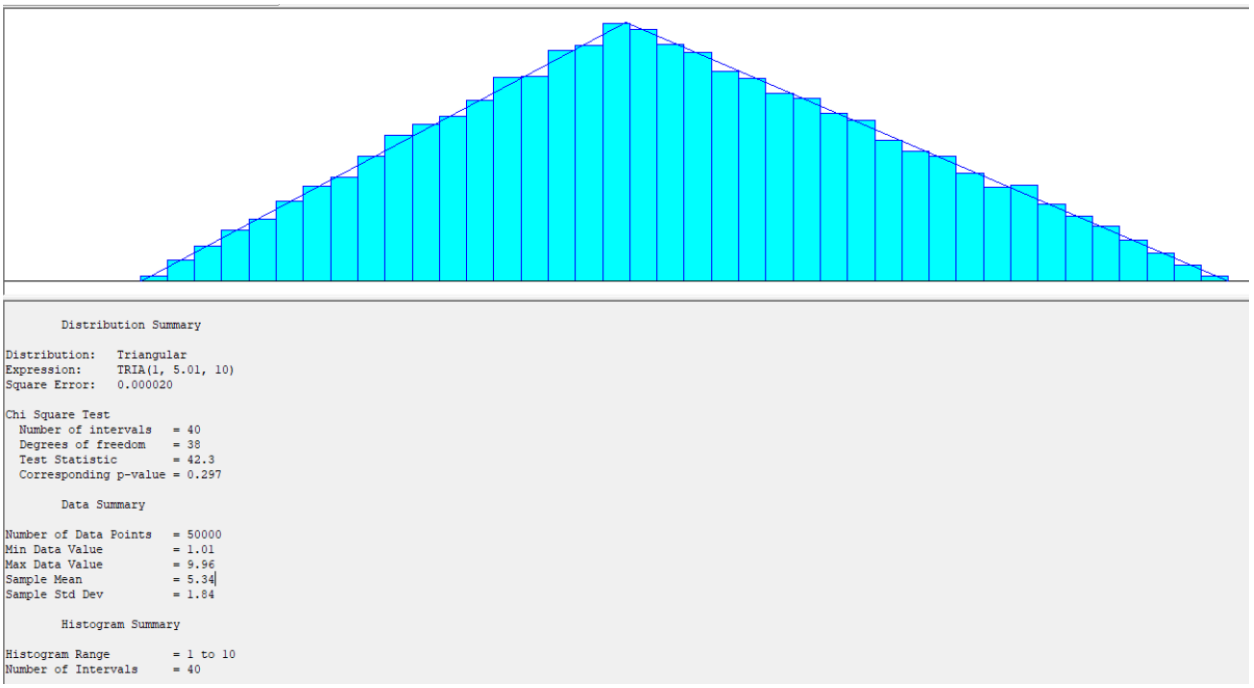
Správnu implementáciu generátorov sme otestovali pomocou programu Input Analyzer a štatistického testu chí-kvadrát.

ExponentialGenerator



Obrázok 2 Výsledok štatistického testu generátora exponenciálneho rozdelenia

TriangularGenerator



Obrázok 3 Výsledok štatistického testu generátora trojuholníkového rozdelenia

Simulačné jadro

EventSimulationCore

Implementovali sme triedu EventSimulationCore, ktorá predstavuje všeobecné simulačné jadro pre udalostne orientované simulácie. Triedu sme implementovali ako potomka triedy SimulationCore, ktorá predstavuje všeobecné simulačné jadro, ktoré sme implementovali v prvej semestrálnej práci a prekryli sme metódu executeSimRun, ktorá implementuje vykonanie jednej replikácie simulačného behu.

Potomok rozširuje triedu o kalendár udalostí implementovaný prioritným frontom, atribúty pre aktuálny a maximálny simulačný čas , faktor spomalenia simulácie a boolean atribúty pre pozastavenie a predčasné ukončenie simulácie.

Simulačné jadro podporuje predčasné zastavenie simulačného behu, dočasné pozastavenie simulačného behu a zmenu rýchlosti priebehu simulačného behu.

Simulačné jadro podporuje pomocou návrhového vzoru Observer akékoľvek užívateľské rozhranie, ktoré implementuje rozhranie UserInterface a jeho metódu refresh.

executeSimRun

```
protected void executeSimRun() {
    while (eventCalendar.size() > 0 && currentTime < maxSimulationTime && !stop) {
      while (pause) {
        try {
          Thread.sleep(200);
        } catch (InterruptedException e) {
          e.printStackTrace();
        }
      }
      Event event = eventCalendar.poll();
      if (event.getTime() < maxSimulationTime) {
        if (event.getTime() >= currentTime - Constants.epsilon) {
          currentTime = event.getTime();
          event.execute();

          if (numberOfReplications == 1) {
            refreshGUI();
          }
        } else {
          throw new RuntimeException("Event time is less than current time: " + event.getTime() + " < " + currentTime);
        }

      }
    }
  }
```

Priloha 3 Hlavná metóda executeSimRun udalostného simulačného jadra

Udalosti využité s týmto simulačným jadrom musia byť potomkom triedy Event.

Event

Trieda predstavuje všeobecnú udalosť, pre udalostne orientované simulačné jadro.

```
public abstract class Event implements Comparable<Event> {
   private double time;
private EventSimulationCore simulationCore;
   public Event(double time, EventSimulationCore simulationCore) {
      this.simulationCore = simulationCore;
      this.time = time;
   }
   public double getTime() {
      return time;
   }
   abstract public void execute();
   public EventSimulationCore getSimulationCore() {
      return simulationCore;
   }
}
```

Priloha 4 Abstraktná trieda Event

SystemEvent

Trieda predstavuje udalosť slúžiacu na spomaľovanie a zrýchlovanie simulácie.

Vykonanie udalosti sa plánuje na každú jednotku simulačného času v prípade, že simulácia nebeží maximálnou rýchlosťou.

```
public void execute() {
    double timeFactor = getSimulationCore().getTimeFactor();
    if (timeFactor > 0) {
      try {
        Thread.sleep((long)(1000/timeFactor));
      }
      catch (InterruptedException e) {
        e.printStackTrace();
      }
      getSimulationCore().refreshGUI();
      SystemEvent newEvent = new SystemEvent(getTime() + 1, getSimulationCore());
      getSimulationCore().addEvent(newEvent);
    }
  }
```

Priloha 5 Metóda execute triedy SystemEvent

Štatistiky

Na zber a vyhodnocovanie štatistík využívame v semestrálnej práci triedu Statistic a WeightStatistic. Tieto triedy využívame na zber hodnôt a následný výpočet priemerných hodnôt a intervalov spoľahlivosti. Pre výpočet intervalu spoľahlivosti využívame knižnicu org.apache.commons.math3.distribution a vrámci tejto knižnice triedu TDistribution pre studentovo rozdelenie.

V semestrálnej práci sme sledovali nasledovné štatistiky:

1. Priemerný čas dokončenia objednávky
2. Priemerný počet nezačatých objednávok.
3. Percentuálne pracovné vyťaženie jednotlivých pracovníkov zo skupiny A,B a C.
4. Percentuálne pracovné vyťaženie pre celú skupinu pracovníkov A, B a C.

Simulácia stolárskej dielne

Simuláciu stolárskej dielne sme implementovali v triede FurnitureCompany, ktorá je potomkom všeobecného udalostného simulačného jadra.

Trieda obsahuje generátory pre rozdelenia definované v zadaní práce:

Tabuľka 1 Evidencia generátorov pre stolársku dielňu

Názov generátora	Typ rozdelenia	Parametre	Poznámka
orderArrivalGen	Exponenciálne	mean = 1800	30 minút medzi príchodmi (v sekundách)
storageMoveTimeGen	Trojuholníkové	min = 60, max = 480, mode = 120	Čas presunu medzi skladom a montážnym miestom
stationMoveTimeGen	Trojuholníkové	min = 120, max = 500, mode = 150	Čas presunu medzi montážnymi miestami
materialPrepTimGen	Trojuholníkové	min = 300, max = 900, mode = 500	Príprava materiálu
tableCutTimeGen	Spojité empirické	intervaly: [600–1500], [1500–3000] pravdepodobnosti: 0.6, 0.4	Rezanie stolov
tableVarnTimeGen	Spojité rovnomerné	interval: [12000–36600]	Lakovanie stolov
tableAssembleTimeGen	Spojité rovnomerné	interval: [1800–3600]	Skladanie stolov
chairCutTimeGen	Spojité rovnomerné	interval: [720–960]	Rezanie stoličiek
chairVarnTimeGen	Spojité rovnomerné	interval: [12600–32400]	Lakovanie stoličiek
chairAssembleTimeGen	Spojité rovnomerné	interval: [840–1440]	Skladanie stoličiek
wardrobeCutTimeGen	Spojité rovnomerné	interval: [900–4800]	Rezanie skríň
wardrobeVarnTimeGen	Spojité rovnomerné	interval: [36000–42000]	Lakovanie skríň
wardrobeAssembleTimeGen	Spojité rovnomerné	interval: [2100–4500]	Skladanie skríň
wardrobeFittingTimeGen	Spojité rovnomerné	interval: [900–1500]	Montáž kovaní

Trieda ďalej obsahuje potrebné fronty, ktoré sú implementované spôsobom FIFO. Simulačný čas v tejto triede je uvádzaný v sekundách. Trieda rovnako obsahuje atribúty pre zber potrebných štatistík.

Hlavné metódy triedy

beforeSimRun

Inicializuje východzí stav entít simulácie pred každou replikáciou a resetuje štatistiky zbierané pre každú replikáciu.

afterSimRun

Aktualizuje zber globálnych štatistík vyhodnocovaných pre celú simuláciu.

Experimenty

Úlohou semestrálnej práce bolo po implementácii vykonať experimenty a odporučiť minimálny počet stolárov jednotlivých skupín pri ktorom priemerný čas vybavenia objednávky nepresiahne 16 hodín.

Keďže našou úlohou je minimalizovať počet zamestnancov každej skupiny , tak prvotný experiment sme vykonali s nasledovnou konfiguráciou počtu zamestnancov v jednotlivých skupinách:

A=2, B=2, C=2

Sledovaním priebehu tejto simulácie sme si všimli veľké množstvo objednávok čakajúcich na lakovanie, čo bolo spôsobené tým, že technologický krok lakovania pri všetkých typoch nábytku má najdlhšie trvanie. Zároveň zamestnanci skupiny C ako jediný vykonávajú 2 technologické kroky a to lakovanie a montáž kovaní na skrine.

V tomto prípade bol priemerný čas spracovania objednávky 879 hodín.

Následne sme postupným navyšovaním množstva pracovníkov skupiny C hľadali konfiguráciu, pri ktorej priemerný čas potrebný pre spracovanie objednávky klesne pod požadovanú hranicu 16 hodín.

Postupným navyšovaním sme získali nasledovnú konfiguráciu:

A=2, B=2, C=18

Pri tejto konfigurácii priemerný čas spracovania objednávky klesol na 13.2504h, čo spĺňa požiadavku neprekročenia 16 hodín.

Zozbierané štatistiky pre túto konfiguráciu:

Tabuľka 2 Štatistiky pre objednávky konfigurácie A=2 B=2 C=18

Štatistika	Hodnota	Interval spoľahlivosti
Priemerný čas spracovania objednávky(h)	13.2504	<13.1902, 13.3106>
Priemerný počet čakajúcich objednávok:	1.3911	<1.3751, 1.4072>

Tabuľka 3 Štatistiky pre skupiny pracovníkov konfigurácie A=2 B=2 C=18

Skupina	Vyťaženosť(%)	Interval spoľahlivosti
A	80.3277	<80.2394, 80.4161>
B	80.9071	<80.8218, 80.9924>
C	91.9678	<91.8685, 92.0670>

Táto konfigurácia spĺňa vybavenie objednávky do 16 hodín. Ďalej sme preskúmaním okolia tohto riešenia skúmali možnosti minimalizovať počet pracovníkov v jednotlivých skupinách.

Otestovali sme nasledovné riešenia:

A=2, B=2, C=17

V tejto konfigurácii prekročil priemerný čas spracovania objednávky 16 hodín, takže riešenie nie je prúpustné.

Štatistiky:

Tabuľka 4 Štatistiky pre objednávky konfigurácie A=2 B=2 C=17

Štatistika	Hodnota	Interval spoľahlivosti
Priemerný čas spracovania objednávky(h)	19.3802	<19.0372, 19.7233>
Priemerný počet čakajúcich objednávok:	1.3827	<1.3673, 1.3980>

Tabuľka 5 Štatistiky pre skupiny pracovníkov konfigurácie A=2 B=2 C=17

Skupina	Vyťaženosť(%)	Interval spoľahlivosti
A	80.3134	<80.2318, 80.3951>
B	80.5666	<80.4973, 80.6358>
C	96.9759	<96.8940, 97.0577>

A=2, B=1, C=18

V tejto konfigurácii prekročil priemerný čas spracovania objednávky 16 hodín, takže riešenie nie je prúpustné.

Tabuľka 6 Štatistiky pre objednávky konfigurácie A=2 B=1 C=18

Štatistika	Hodnota	Interval spoľahlivosti
Priemerný čas spracovania objednávky(h)	392.5164	<391.5324, 393.5003>
Priemerný počet čakajúcich objednávok:	1.3774	<1.3616, 1.3933>

Tabuľka 7 Štatistiky pre skupiny pracovníkov konfigurácie A=2 B=1 C=18

Skupina	Vyťaženosť(%)	Interval spoľahlivosti
A	80.3600	<80.2772, 80.4428>
B	99.5268	<99.5222, 99.5314>
C	91.4184	<91.3302, 91.5067>

A=1, B=2, C=18

V tejto konfigurácii prekročil priemerný čas spracovania objednávky 16 hodín, takže riešenie nie je prúpustné.

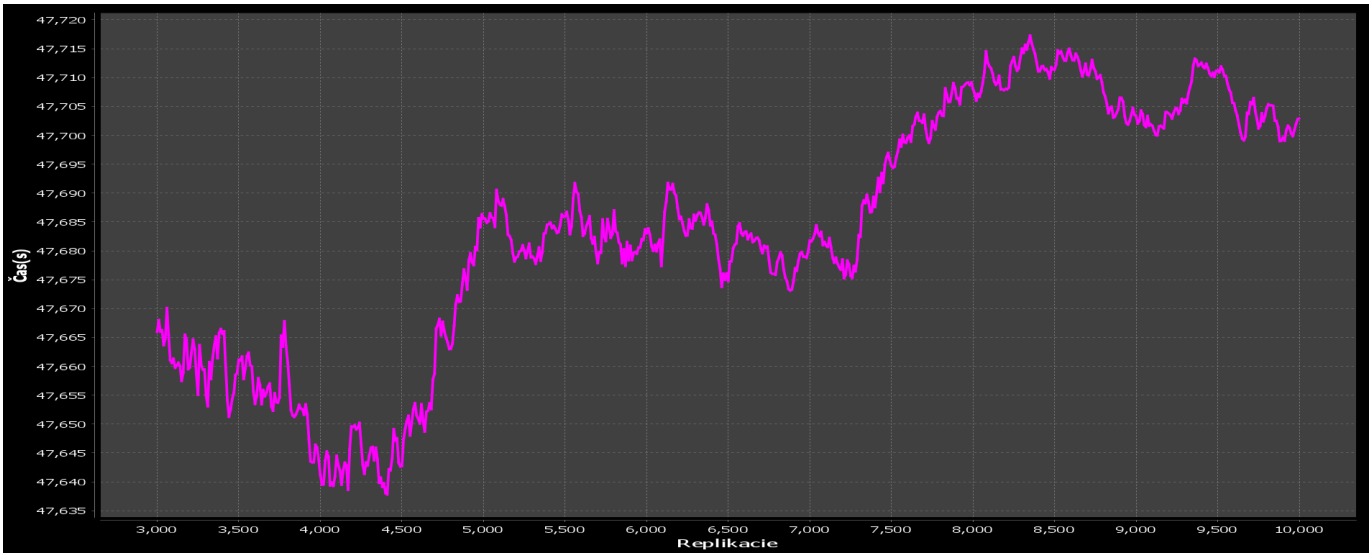
Tabuľka 8 Štatistiky pre objednávky konfigurácie A=1 B=2 C=18

Štatistika	Hodnota	Interval spoľahlivosti
Priemerný čas spracovania objednávky(h)	385.0666	<384.1067, 386.0265>
Priemerný počet čakajúcich objednávok:	753.4864	<751.1599, 755.8130>

Tabuľka 9 Štatistiky pre skupiny pracovníkov konfigurácie A=1 B=2 C=18

Skupina	Vyťaženosť(%)	Interval spoľahlivosti
A	99.9540	<99.9512, 99.9568>
B	50.4050	<50.3808, 50.4292>
C	57.2916	<57.2658, 57.3173>

Na základe vykonaných experimentov sme zistili, že minimálny počet pracovníkov v jednotlivých skupinách za podmienky priemerného času spracovania objednávky nepresahujúceho 16 hodín je nasledovný: A=2, B=2, C=18.



Obrázok 4 Ustáľovanie času spracovania objednávky