**«Описание процессов жизненного цикла АСОИУ»**

**Методические указания к семинару  
«Настройка имитационной модели с фреймворком SimPy»**

В этом примере моделируется процесс обслуживания покупателей в магазине.

В этом процессе ресурсом обслуживания является касса, в нашем случае число касс, которые обслуживают покупателей. В этой модели мы полагаем, что единица модельного времени соответствует 1 секунде реального времени. Длительность покупки (время обслуживания, ST) оценивается как равномерно распределенная случайная величина на интервале от 0 до 20 мин. Интервал появления нового посетителя (IAT) оценивается как равномерно распределенная случайная величина на интервале от 0 до 4 мин. После 10 часов работы магазин закрывается на вход, и обслуживаются только оставшиеся покупатели.

П

process\_man

К

source\_men

Man

X

Ключевыми характеристиками работы магазина в этой упрощенной модели будут:

- максимальная длина очереди;

- необходимое число касс, чтобы очереди не становились слишком длинными;

- время ожидания в очереди.

# -\*- coding: utf-8 -\*-

# Импорт библиотеки SimPy

import simpy

# Для генерации случайных чисел

import random

# Инициализация среды моделирования

env = simpy.Environment()

# Ресурс обслуживания; в данном случае - capacity - число касс,

# которые обслуживают покупателей

bcs = simpy.Resource(env, capacity=6)

# в этой модели полагаем, что единица модельного

# времени соответствует 1 секунде реального времени

SERVICE\_DURATION = 20 \* 60 # максимальная длительность обслуживания 20 мин

# время обслуживания - равномерно распределенная случайная величина

# на интервале [0, 20 \* 60]

# интервал между появлением нового посетителя

ARRIV\_INTER = 4 \* 60

# время появления нового посетителя -равномерно распределенная

# случайная величина на интервале [0, 4 \* 60]

# После 10 часов работы магазин закрывают и обслуживают только оставшихся

CONSUMER\_TIME = 60 \* 60 \* 10

# ------- Служебные параметры для сбора статистики --------

myquelen = 0 # Текущая длина очереди

queue = [] # массив очередей, для построения графика

maxwaits = [] # массив времен ожидания в очереди, для построения графика

timelist\_q = [] # массив времен, соответствующий очереди queue

timelist\_w = [] # массив времен, соответствующий временам ожидания

# -----------------------------------------------------------------

# Объект класса Man - посетитель магазина

class Man(object):

def \_\_init\_\_(self, env, res, name='№\_0'):

self.name = name # Имя посетителя, нам надо их различать

self.env = env # Среда моделирования

self.res = res # используемый ресурс, в данном случае - касса

def run(self):

# сделаем счетчики статистики глобальными

global myquelen, maxwaits, timelist\_q,timelist\_w

# человек пришел и встал в очередь: она увеличилась на 1

myquelen += 1

print(u"Добрый день! Меня зовут {0} и я прибыл в магазин в {1} \

(время)".format(self.name, self.env.now))

# Запомним время, чтобы посчитать потом время пребывания в очереди

time = self.env.now

# Запрос свободной кассы...

with self.res.request() as req:

# Нет ничего свободного... в очередь...

yield req

# Свободная касса появилась...

# Человек поступает на обслуживание, и очередь уменьшается на 1

myquelen -= 1

# запомним текущую длину очереди для графика

queue.append(myquelen)

# запомним текущее время для графика

timelist\_q.append(self.env.now)

# переменная wait-time - время, проведенное в очереди

wait-time = self.env.now

# время обслуживания - случайное число, генерируем его

serving\_duration = random.randint(0, SERVICE\_DURATION)

# обслуживаемся в кассе...

yield self.env.timeout(serving\_duration)

# Обслужились

print(u"Я {0}; обслуживался {1} единиц времени, и ждал в \

очереди {2} ".format(self.name, serving\_duration, wait-time))

# Запомним время проведенное в очереди для графика

maxwaits.append(wait-time)

# Запомним текущее время

timelist\_w.append(self.env.now)

print(u"Меня обслужили и сейчас ({0} - текущее время) я \

покидаю магазин.".format(self.env.now))

# Процесс-источник посетителей со временем между клиентами

# случайная величина на интервале [0, 4 \* 60]

def source\_men(env):

ind = 0

# Посетители приходят 10 часов CONSUMER\_TIME = 3600\*10

while env.now < (CONSUMER\_TIME - ARRIV\_INTER):

ind += 1

yield env.timeout(random.randint(0, ARRIV\_INTER))

man = Man(env, bcs, name='№%s' % ind)

env.process(man.run())

# Добавляем в модель процесс появления в магазине посетителей

env.process(source\_men(env))

# Запускаем процесс моделирования, полагая, что

# один шаг моделирования - 1 секунда реального времени;

# процесс моделирования составляет 12 часов; посетители входят 10 часов

# Далее обслуживаются оставшиеся в очереди, если таковые имеются

env.run(until=12 \* 60 \* 60)

Результат запуска модели будет такой (результаты могут несколько отличаться от эксперимента к эксперименту, т.к. мы не фиксировали поведение генератора случайных чисел):

Добрый день! Меня зовут №1 и я прибыл в магазин в 36 (время)

Добрый день! Меня зовут №2 и я прибыл в магазин в 152 (время)

Добрый день! Меня зовут №3 и я прибыл в магазин в 224 (время)

Я №1; обслуживался 189 единиц времени, и ждал в очереди 0 единиц времени

Меня обслужили и сейчас (225 -- текущее время) я покидаю магазин.

Добрый день! Меня зовут №4 и я прибыл в магазин в 235 (время)

Добрый день! Меня зовут №5 и я прибыл в магазин в 282 (время)

Я №4; обслуживался 191 единиц времени, и ждал в очереди 0 единиц времени

Меня обслужили и сейчас (426 -- текущее время) я покидаю магазин.

Добрый день! Меня зовут №6 и я прибыл в магазин в 485 (время)

Я №5; обслуживался 245 единиц времени, и ждал в очереди 0 единиц времени

Меня обслужили и сейчас (527 -- текущее время) я покидаю магазин.

Я №3; обслуживался 353 единиц времени, и ждал в очереди 5 единиц времени

Меня обслужили и сейчас (577 -- текущее время) я покидаю магазин.

Добрый день! Меня зовут №7 и я прибыл в магазин в 639 (время)

Добрый день! Меня зовут №8 и я прибыл в магазин в 711 (время)

Добрый день! Меня зовут №9 и я прибыл в магазин в 786 (время)

Я №6; обслуживался 336 единиц времени, и ждал в очереди 10 единиц времени

Меня обслужили и сейчас (821 -- текущее время) я покидаю магазин.

И т.д.

Запустим сначала модель со следующими параметрами (config1) :

capacity = 6 #(количество касс)

SERVICE\_DURATION = 20 \* 60 #(макс. время обслуживания клиента на кассе)

ARRIV\_INTER = 4 \* 60 #(макс.интервал прихода нового клиента)

Тогда, несмотря на значительное число касс, и неплохое начало обслуживания (т.е. в начале рабочего дня очереди нулевые), где-то в середине рабочего дня образуются очереди до 7 человек, со временем ожидания в очереди для отдельных клиентов до 10 минут. Важно, что в начале дня очереди нет, и может показаться, что так будет всегда (например, понаблюдали 2 часа - количество касс достаточно, справляются, но модель показывает, что возможны небольшие "удлинения" очереди).

Изменим наши условия, и уменьшим число касс на единицу при сохранении остальных параметров (config2):

capacity = 5 #(количество касс)

SERVICE\_DURATION = 20 \* 60 #(макс. время обслуживания клиента на кассе)

ARRIV\_INTER = 4 \* 60 #(макс.интервал прихода нового клиента)

Если уменьшить число касс на 1, появляются уже очереди из 15-20 человек, где люди должны ожидать около 40 минут. Теперь уменьшим еще на 1 (config3):

capacity = 4 #(количество касс)

SERVICE\_DURATION = 20 \* 60 #(макс. время обслуживания клиента на кассе)

Теперь очереди возрастут до 60 человек, и некоторые из посетителей будут ожидать в них около 3 часов. Одна из причин – достаточно большое максимальное время обслуживания (оно может достигать 20 минут); если этот параметр уменьшить (до 10 мин), то и количество касс можно будет сократить (config4):

capacity = 4 #(количество касс)

SERVICE\_DURATION = 10 \* 60 #(макс. время обслуживания клиента на кассе)

Для отображения динамики состояния очереди лучше построить ее график.

# --------------------- Выводим результаты моделирования в виде графиков

# Должен быть установлен пакет matplotlib,

# если нет, то используем canvas

try:

from pylab import \*

#

font = {'family' : 'Normal',

'weight' : 'normal',

'size' : 22}

matplotlib.rc('font', \*\*font)

figure()

# График длины очереди

plot(timelist\_q, queue, label='queue\_len')

title('Queue length')

xlabel(u'Simulation time, sec')

ylabel(u'Current queue length, #')

figure()

# График времени ожидания в очереди

plot(timelist\_w, maxwaits)

title(u'Waiting time')

xlabel(u'Simulation time, sec')

ylabel(u'Somebody waits..., sec')

show()

# ----------------------------------------------------------------------------

except ImportError:

print('without matplotlib - use tkinter')

from tkinter import \*

#

tk = Tk()

tk.title("График очереди")

#

button = Button(tk); button["text"]="Закрыть"; button["command"]= tk.destroy

button["font"]='Arial'

button.pack()

#

canva = Canvas(tk)

canva["height"]=600; canva["width"]=800; canva["background"]="#eeddff"

canva["borderwidth"]=2

canva.pack()

#

y0=500; x0=10; x1=600; dx=1

# рисуем ось У

y\_axe=[];

yy=(x0,0); y\_axe.append(yy)

yy=(x0,y0); y\_axe.append(yy)

canva.create\_line(y\_axe,fill='black',width=2,arrow=FIRST)

# делаем подписи на оси

for i in range(y0):

if (i>0) and (i%100==0):

k = y0 - i

canva.create\_line(x0, k, x0+10, k, width = 0.5, fill = 'black')

canva.create\_text(x0+20, k-10,text=str(i\*2),fill='purple')

# рисуем ось х

x\_axe=[]

xx=(x0,y0); x\_axe.append(xx)

xx=(2\*x0+x1,y0); x\_axe.append(xx)

canva.create\_line(x\_axe,fill='black',width=2,arrow=LAST)

# делаем подписи на оси

for i in range(x1):

if (i % 50 == 0):

k = i

canva.create\_line(x0+i, y0, x0+i, y0+10, width = 0.5, fill = 'black')

canva.create\_text(x0+i, y0+20, text=str(i\*100),fill='purple')

canva.create\_text(2\*x0+x1,y0+20,text="время",fill='purple')

canva.create\_text(2\*x0+x1,50,text='макс.',fill='black', font=('Arial','14'))

# рисуем График длины очереди

points=[]

for n in range(0,len(queue)):

pp=(timelist\_q[n]/100,y0-queue[n]\*30)

points.append(pp)

canva.create\_line(points,fill="blue",smooth=0,tags="queue count")

canva.create\_text(2\*x0+x1,y0-200,text=str(max(queue)),fill='blue')

canva.create\_text(2\*x0+x1,y0-220,text="длина",fill='blue')

# рисуем График времени ожидания в очереди

points=[]

for n in range(0,len(maxwaits)):

pp=(timelist\_w[n]/100,y0-maxwaits[n]/2)

points.append(pp)

canva.create\_line(points,fill="red",smooth=0,tags="wait time")

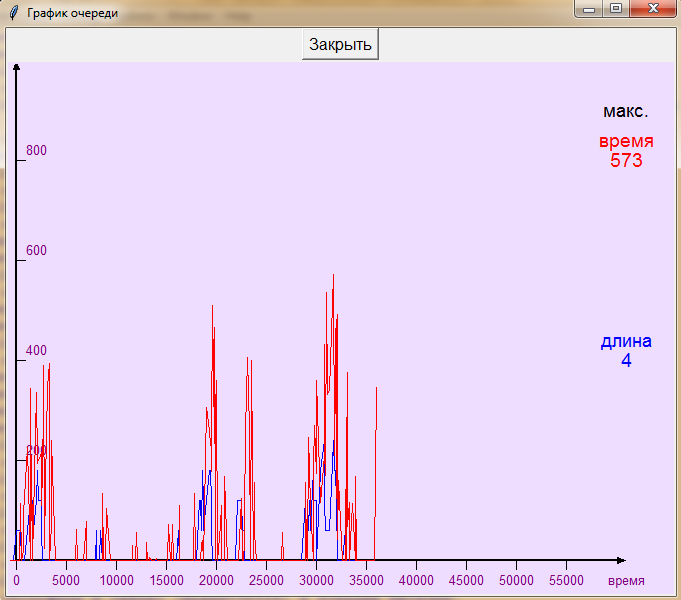
canva.create\_text(2\*x0+x1,y0-400,text=str(max(maxwaits)),fill='red')

canva.create\_text(2\*x0+x1,y0-420,text="время",fill='red',font=('Arial','14'))

#

tk.mainloop()

График при использовании tkinter будет примерно такого вида:



Постройте графики для 4х вариантов конфигурации модели – config1, config2, config3, config4. Добавьте на график общее число обслуженных покупателей. Сохраните графики в документ word, добавьте в отчет текст модели на python и пришлите на почту преподавателя.