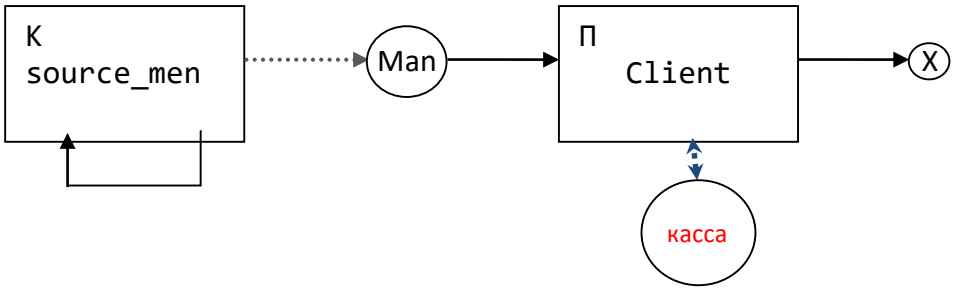
Методические указания к семинару

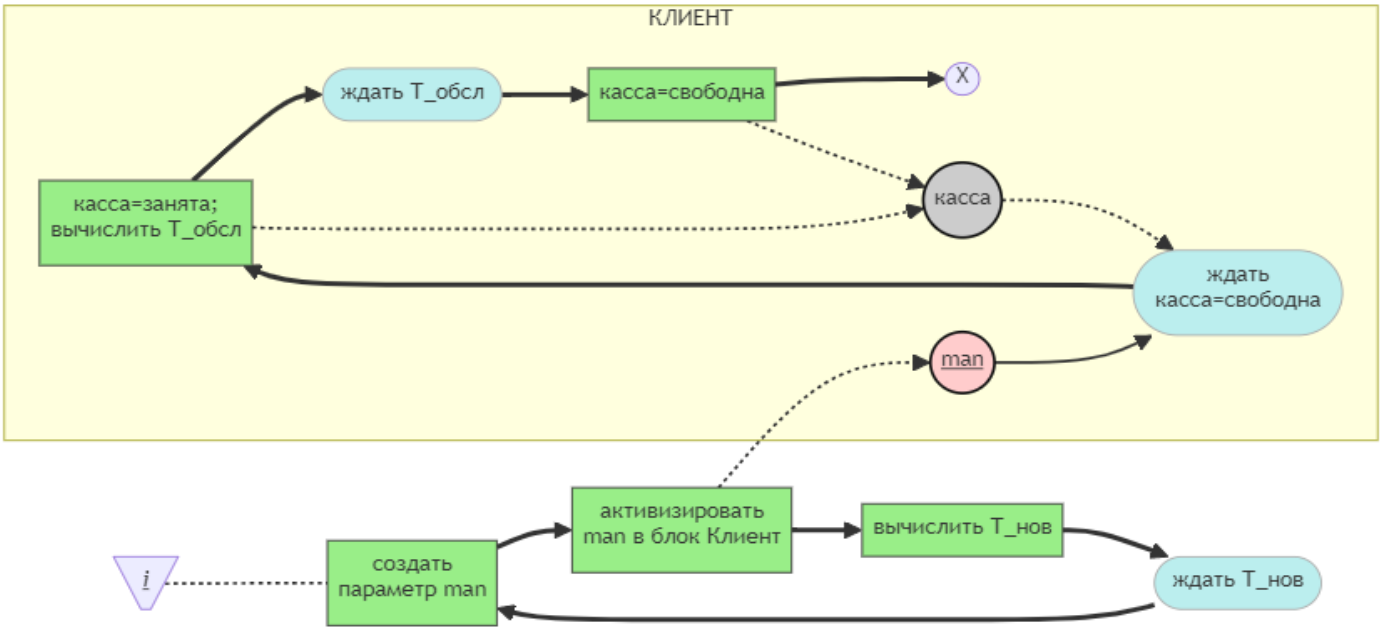
Настройка имитационной модели системы массового обслуживания на основе фреймворка Simpy

В этом примере рассмотрим модель процесса обслуживания покупателей на кассе в магазине.

В этом процессе ресурсом обслуживания является касса, в нашем случае число касс, которые обслуживают покупателей. Обслуживание происходит из одной общей очереди на любой свободной кассе.

Предположим, что единица модельного времени соответствует 1 секунде реального времени. Длительность оплаты на кассе (время обслуживания ST) оценивается как треугольно распределенная случайная величина на интервале от 30 сек до 20 мин с модой 7 мин. Интервал появления нового покупателя (IAT) оценивается как равномерно распределенная случайная величина на интервале от 0 до 4 мин. После 10 часов работы магазин закрывается на вход, но дообслуживаются все покупатели.





Ключевыми характеристиками работы магазина в этой упрощенной модели будут:

- максимальная длина очереди;
- необходимое число касс, чтобы очередь не становились слишком длинной;
- время ожидания в очереди.

Код модели:

```
# -*- coding: utf-8 -*-
# Импорт среды SimPy
import simpy
# Для генерации случайных чисел
import random
# Единица модельного времени = 1 секунда реального времени
# Длительность получения услуги
MAX_SERVICE_DURATION = 20 * 60
# т.е. максимальная длительность обслуживания 20 мин
MODE_SERVICE_DURATION = 7 * 60
# индивидуальное время обслуживания треугольно распределенной
# случайной величине на интервале [30, 20*60, 7*60]

# макс.интервал до появления нового покупателя
ARRIV_INTER = 3 * 60
# время появления нового покупателя равномерно распределенная
# случайная величина на интервале [0, 3*60]

# Время работы магазина (10 часов)
# После 10 часов работы магазин закрывают и обслуживают только оставшихся
CONSUMER_TIME = 3600 * 10

# ------------ Служебные параметры для статистики --------
myquelen = 0  # Текущая длина очереди
quelog = [] # журнал длины очереди
tservlog =[] # журнал пребывания в системе
# Класс - покупатель магазина - логический блок-процессор
class Client(object):
    def __init__(self, env, res, name='client'):
        self.name = name  # Имя покупателя, чтобы их различать
        self.env = env  # Среда моделирования
        self.res = res  # используемый при моделировании ресурс- касса

    def run(self):
        # ссылка на глобальные счетчики статистики
        # для графиков после моделирования
        global myquelen, quelog, tservlog
        # клиент пришел и встал в очередь: она увеличилась на 1
        # Запомним время, чтобы посчитать потом время пребывания в магазине
        timeq = self.env.now
        print(f"Привет! Это {self.name}, и я прибыл в магазин в {timeq} (сек)")
        # Запрос свободной кассы
        with self.res.request() as req:
            # Нет свободной кассы? в очередь
            yield req
            # Свободная касса появилась!
            # клиент поступает на обслуживание и очередь уменьшается на 1
            # запомним текущую длину очереди и текущее время события
            wait = self.env.now
            quelog.append((wait, len(self.res.queue)))
            # время обслуживания - случайное число, генерируем его
```

```
            serving_duration=int(random.triangular(30,MAX_SERV_DURAT,MODE_SERV_DURAT))
            # обслуживаемся в кассе!
            yield self.env.timeout(serving_duration)
            # Обслужились и освободили кассу
            # переменная (wait-timeq) - время, проведенное в очереди
            print(f"Я {self.name}, обслуживался {serving_duration} секунд, \
                    ждал в очереди {wait-timeq} секунд")
            # Запомним время проведенное в магазине
            # Запомним текущее время события - клиент ушел
            tservlog.append((self.env.now, self.env.now-timeq))
            print(f"Меня обслужили, и сейчас (время={self.env.now}) я ушёл.")
# Инициализация среды моделирования
env = simpy.Environment()

# Ресурс обслуживания - capacity - число касс,
# которые обслуживают покупателей
cashier = simpy.Resource(env, capacity=6)

# Источник покупателей предполагает, что покупатели приходят
# в течение 10 часов от начала работы магазина
def source_men(env):
    ind = 0
    while env.now < (CONSUMER_TIME - ARRIV_INTER):
        ind += 1
        yield env.timeout(random.randint(0, ARRIV_INTER))
        man = Client(env, cashier, name='клиент%s' % ind)
        env.process(man.run())

# инициализация датчика случайных чисел
random.seed(13971)
# инициализация журналов статистики
quelog = []; tservlog = []
# Добавляем процесс прихода покупателей в модель
env.process(source_men(env))
# Запускаем процесс моделирования, полагая, что
# процесс моделирования составляет 12 часов;
env.run(until=12 * 60 * 60)
```

Результат запуска модели будет примерно такой (результаты могут несколько отличаться от эксперимента к эксперименту):

```
Привет! Это клиент1, и я прибыл в магазин в 14 (сек)
Привет! Это клиент2, и я прибыл в магазин в 38 (сек)
Привет! Это клиент3, и я прибыл в магазин в 210 (сек)
Привет! Это клиент4, и я прибыл в магазин в 274 (сек)
Привет! Это клиент5, и я прибыл в магазин в 363 (сек)
Привет! Это клиент6, и я прибыл в магазин в 491 (сек)
Привет! Это клиент7, и я прибыл в магазин в 513 (сек)
Привет! Это клиент8, и я прибыл в магазин в 564 (сек)
Я клиент1, обслуживался 622 секунд, ждал в очереди 0 секунд
Меня обслужили, и сейчас (время=636) я ушёл.
Я клиент2, обслуживался 610 секунд, ждал в очереди 0 секунд
Меня обслужили, и сейчас (время=648) я ушёл.
Привет! Это клиент9, и я прибыл в магазин в 691 (сек)
```

```
Я клиент7, обслуживался 131 секунд, ждал в очереди 123 секунд
Меня обслужили, и сейчас (время=767) я ушёл.
Я клиент3, обслуживался 561 секунд, ждал в очереди 0 секунд
Меня обслужили, и сейчас (время=771) я ушёл.
Привет! Это клиент10, и я прибыл в магазин в 852 (сек)
Привет! Это клиент11, и я прибыл в магазин в 884 (сек)
Привет! Это клиент12, и я прибыл в магазин в 895 (сек)
Я клиент4, обслуживался 624 секунд, ждал в очереди 0 секунд
Меня обслужили, и сейчас (время=898) я ушёл.
Я клиент6, обслуживался 472 секунд, ждал в очереди 0 секунд
Меня обслужили, и сейчас (время=963) я ушёл.
Привет! Это клиент13, и я прибыл в магазин в 986 (сек)
 и т.д.
```

Запустим сначала модель со следующими параметрами (**config1**):

```
capacity = 6 #(количество касс)
```

Тогда, несмотря на значительное число касс, и неплохое начало обслуживания (т.е. в начале рабочего дня очередь отсутствует), где-то к середине рабочего дня образуется очередь со временем ожидания 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).

Для отображения динамики состояния очереди лучше построить ее график. Выводим результаты моделирования в виде графиков, если есть установленный пакет matplotlib: один график – ступенчатый для длины очереди, второй график – точечный для времен пребывания клиентов в системе.

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

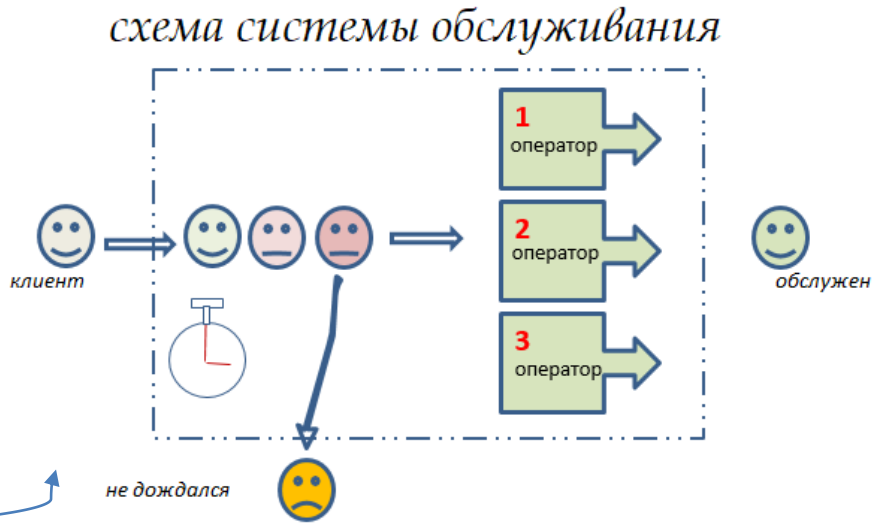
```
capacity = 5 #(количество касс)
```

Если уменьшить число касс на 1, появляется очередь. Теперь уменьшим число касс еще на 1 (**config3**):

```
capacity = 4 #(количество касс)
```

Теперь очередь еще больше возрастет. Одна из причин – достаточно большое максимальное время обслуживания (оно может достигать 20 минут).

Рассмотрим вариант обслуживания с учетом возможного ухода клиентов из очереди без обслуживания. Этот вариант процесса был рассмотрен ранее в примере модели в формате excel.



В таком варианте обслуживания нужно будет в модели предусмотреть и другой вариант получения/неполучения ресурса в формате simpy:

```
# Определим время допустимого ожидания
patience = random.randint(120, 300)
# Нет свободной кассы? в очередь
# клиент ждет или выходит по таймауту
results = yield (req | env.timeout(patience))
# Свободная касса появилась или терпение кончилось!
```

При сохранении основных параметров модели зададим время допустимого ожидания `patience` и количество касс =3 (**config4**):

```
capacity = 3 #(количество касс)
```

Задание

1) Постройте графики для 4х вариантов конфигурации модели – config1, config2, config3, config4. Добавьте на график общее число обслуженных покупателей и максимальный размер очереди.

2) Сравните состояние графиков для config4 при учете допустимого ожидания (man1) и без учета (man). Оцените в этих двух вариантах конфигураций долю клиентов с временем пребывания не более 5 минут.

Пришлите результат (Jupiter notebook) на почту преподавателя.