|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство образования и науки Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Робототехники и комплексной автоматизации

КАФЕДРА Системы автоматизированного проектирования (РК-6)

**ОТЧЕТ ПО ДОМАШНЕМУ ЗАДАНИЮ ПО КУРСУ**

**“Аналитические модели и имитационное моделирование на системном уровне”**

Студент Попонкин Дмитрий

Группа РК6-82

Вариант 13

Тип задания Домашнее задание

*Москва, 2019 г.*

# Содержание

[Содержание 2](#_Toc7451385)

[Задание 3](#_Toc7451386)

[Решение на GPSS 4](#_Toc7451387)

[Реализация на языке программирования 6](#_Toc7451388)

[Вывод 7](#_Toc7451389)

# Задание

В транспортной компании N=200 машин. Карта города представляет собой квадрат, разделённый регулярной сеткой на 10000 квадратных зон. В начальный момент времени все машины находятся на складе, расположенном в левом нижнем углу карты. В диспетчерскую поступают запросы от заказчиков (интервалы распределены экспоненциально со средним 2 минуты, место назначения определяется случайным образом). Диспетчер направляет на вызов первую свободную машину, находящуюся на складе. Если свободных машин нет, клиент ставится в очередь. Машина загружается товаром только одного заказчика и полностью разгружается у него, после чего возвращается на склад. Погрузка и разгрузка машины занимают по 10±2 минуты. Поездка совершается только по «горизонтальным» и «вертикальным» улицам, время на перемещение в соседний квадрат распределено экспоненциально со средним 5 минут, но не менее 2-х.

Промоделировать работу в течение 8 часов. Определить средние долю времени простоя машин. Определить параметры очереди клиентов.

# Решение на GPSS

CARS STORAGE 200

GENERATE(EXPONENTIAL(1,0,2#60))

QUEUE Q\_ORDER

ENTER CARS

DEPART Q\_ORDER

ADVANCE(UNIFORM(2,8#60,12#60))

ASSIGN 1,(DUNIFORM(3,0,100))

ASSIGN 2,(DUNIFORM(4,0,100))

ASSIGN 3,(P1+P2)

TEST G P3,0,SKIP\_GO\_ROAD

GO\_ONE\_ROAD ADVANCE(EXPONENTIAL(6,2#60,5#60))

ASSIGN 3-,1

TEST LE P3,0,GO\_ONE\_ROAD

SKIP\_GO\_ROAD ADVANCE(UNIFORM(5,8#60,12#60))

ASSIGN 4,(P1+P2)

TEST G P4,0,SKIP\_GO\_BACK\_ROAD

GO\_BACK\_ONE\_ROAD ADVANCE(EXPONENTIAL(7,2#60,5#60))

ASSIGN 4-,1

TEST LE P4,0,GO\_BACK\_ONE\_ROAD

SKIP\_GO\_BACK\_ROAD LEAVE CARS

TERMINATE

GENERATE 28800;

TERMINATE 1;

START 1;

Решение на GPSS содержит в себе очередь заказов клиентов Q\_ORDER и многоканальное устройство CARS (парк автомашин). Симуляция езды по городу, загрузки и разгрузку происходит с помощью блока ADVANCE с различными случайно-сгенерированными зависимостями по времени.

Средствами GPSS сгенерирован отчет, включающий в себя множество показателей. Некоторые из них, примененные для вычислений или выводов в отчете приведены ниже.

Параметры очереди клиентов (из GPSS):

QUEUE MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME AVE.(-0) RETRY

Q\_ORDER 30 30 236 206 1.889 230.469 1813.020 0

Параметры многоканального устройства CARS (из GPSS):

STORAGE CAP. REM. MIN. MAX. ENTRIES AVL. AVE.C. UTIL. RETRY DELAY

CARS 200 0 0 200 206 1 111.761 0.559 0 30

Долю простоя машин можно определить как

1 – 0.559 = 0.441

В данном случае, обращая внимание на параметры очереди клиентов, можно установить, что все машины к концу симуляции заняты развозом заказов (очередь полна). Следовательно, логично предположить, что с увеличением времени симуляции утилизация многоканального устройства CARS будет стремиться к 1, а доля простоя машин к 0. Проверим, увеличив время симуляции до 24 часов.

Параметры очереди клиентов (из GPSS) для 24 часов:

QUEUE MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME AVE.(-0) RETRY

Q\_ORDER 424 423 706 206 156.850 19195.278 27103.733 0

Параметры многоканального устройства CARS (из GPSS) для 24 часов:

STORAGE CAP. REM. MIN. MAX. ENTRIES AVL. AVE.C. UTIL. RETRY DELAY

CARS 200 0 0 200 283 1 170.587 0.853 0 423

Действительно, очередь клиентов растет, как и утилизация машин.

В таком случае необходимо найти оптимальное число машин в автопарке чтобы удовлетворять потребностям бизнес-процесса.

Опытным путем получено, что при утилизации машин, равной порядка 0.7 (выбрано исходя из соображений запаса), необходимо около 700 машин (симуляция на 48 часов):

Параметры очереди клиентов (из GPSS) для 48 часов и 700 машин:

QUEUE MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME AVE.(-0) RETRY

Q\_ORDER 26 21 1439 1176 1.408 169.090 925.172 0

Параметры многоканального устройства CARS (из GPSS) для 48 часов и 700 машин:

STORAGE CAP. REM. MIN. MAX. ENTRIES AVL. AVE.C. UTIL. RETRY DELAY

CARS 700 0 0 700 1418 1 495.442 0.708 0 21

Доля простоя машин в таком случае равна

1 – 0.708 = 0.292

# Реализация на языке программирования

В качестве языка для разработки был выбран JavaScript с интерпретатором NodeJs v.11.6.

Исходный код можно найти на github: https://github.com/PDmitryA/imitational\_modeling.

Архитектура решения для симуляции включает в себя несколько принципиальных модулей:

1. time.mjs – модуль симуляции времени. Имеет методы: subscribeOnDelta (подписаться на событие в будущем), tick (переход и исполнение событий на следующем ближайшем временном значении), addGenerator (добавление генератора).
2. queue.mjs – модуль симуляции очереди клиентов. Имеет методы: getCar (получить свободный автомобиль), releaseCar (вернуть автомобиль в автопарк).
3. generator.mjs – модуль, содержащий генераторы u01 (uniform от 0 до 1), uniform (uniform от min до max), exponential (экспоненциальный генератор c математическим ожиданием mean), exponentialMin (экспоненциальный генератор с математическим ожиданием mean и смещением min).
4. car.mjs – модуль, содержащий в себе класс CarTransact, реализующий путем подписки на события в будущем модуля time.mjs логику доставки клиенту товара.
5. index.mjs – модуль управления.

Класс CarTransact реализует методы:

* takeLoad – получение товара на складе.
* goToClient – поездка до клиента.
* deliverLoad – разгрузка товара у клиента.
* goBack – поездка обратно на склад.

Асинхронность функций класса CarTransact позволяет декларативно описывать путь каждого транзакта:

**await** clientQueue.getCar();

**await** carTransact.takeLoad();  
**let** {value: x} = xCoordGen.next();  
**let** {value: y} = yCoordGen.next();  
x = Math.ceil(x);  
y = Math.ceil(y);   
**await** carTransact.goToClient({x: x, y: y});  
**await** carTransact.deliverLoad();  
**await** carTransact.goBack();

**await** clientQueue.releaseCar();

Результаты симуляции для 8 часов моделирования и 200 машин:

Параметры очереди:

Средняя длина очереди: 5.582191508961899

Максимальная длина очереди: 48

Параметры автопарка:

Среднее число машин на задании: 118.99473743445888

Утилизация: 118.99473743445888 / 200 = 0.558805

Результаты симуляции близки к значениям, полученным из GPSS, что лишний раз подтверждает точность симуляции.

# Вывод

Язык имитационного моделирования GPSS предоставляет средства для моделирования бизнес-систем, позволяет анализировать достаточность ресурсов предприятия, а главное формирует отчет, содержащий множество показателей для дальнейшего анализа. Конечно, симуляцию можно производить и на языке программирования общего назначения, но это повлечет за собой трату дополнительных ресурсов для разработки цикла событий, отладки и последующему анализу результатов.

# Приложение

## ЦТС и ЦБС

Пусть 1-ый транзакт направляется в ячейку карты (1;1), а 2-ой в (1;0).

Обозначения: [Номер транзакта. Время движения. Номер текущего блока. Уровень приоритета. Номер следующего блока]

|  |  |  |
| --- | --- | --- |
| Модельное время | Цепь текущих событий | Цепь будущих событий |
| До фазы ввода | Пусто | Пусто |
| После фазы ввода | Пусто | [1.1.нет.0.2] [2.2000.нет.0.2] |
| 1 (после фазы коррекции таймера) | [1.КМР.нет.0.2] | [2.2000.нет.0.2] |
| 1 (после фазы просмотра) | Пусто | [2.2000.нет.0.2] [1.10000.нет.0.6] |
| 2000 (после фазы коррекции таймера) | [2.КМР.нет.0.3] | [1.10000.нет.0.6] |
| 2000 (после фазы просмотра) | Пусто | [1.10000.нет.0.6][2.10000.нет.0.6] |
| 10000 (после фазы коррекции таймера) | [1.КМР.нет.0.6][2.КМР.нет.0.6] | Пусто |
| 10000 (после фазы просмотра) | Пусто | [1.15000.нет.0.11][2.16000.нет.0.11] |
| 15000 (после фазы коррекции таймера) | [1.КМР.нет.0.11] | [2.16000.нет.0.11] |
| 15000 (после фазы просмотра) | Пусто | [2.16000.нет.0.11] [1.21000.нет.0.11(переместился на 1 и запланировал перемещение еще на 1)] |
| 16000 (после фазы коррекции таймера) | [2.КМР.нет.0.11] | [1.21000.нет.0.11] |
| 16000 (после фазы просмотра) | Пусто | [1.21000.нет.0.11] [2.28000.нет.0.14 (переместился на 1 и запланировал разгрузку)] |
| 21000 (после фазы коррекции таймера) | [1.КМР.нет.0.11] | [2.28000.нет.0.14] |
| 21000 (после фазы просмотра) | Пусто | [2.28000.нет.0.14] [1.26000.нет.0.14 (запланировал разгрузку)] |
| 26000 (после фазы коррекции таймера) | [1.КМР.нет.0.14] | [2.28000.нет.0.14] |
| 26000 (после фазы просмотра) | Пусто | [2.28000.нет.0.14] [1.35000.нет.0.17 (разгрузился и планирует обратный путь)] |
| 28000 (после фазы коррекции таймера) | [2.КМР.нет.0.14] | [1.35000.нет.0.17] |
| 28000 (после фазы просмотра) | Пусто | [1.35000.нет.0.17] [2.35000.нет.0.17 (разгрузился и планирует обратный путь)] |
| 35000 (после фазы коррекции таймера) | [1.КМР.нет.0.17] [2.КМР.нет.0.17 (вернулся в парк)] | Пусто |
| 35000 (после фазы просмотра) | Пусто | [1.37000.нет.0.17 (осталась 1 клетка)] |
| 37000 (после фазы коррекции таймера) | [1.КМР.нет.0.17] | Пусто |
| 37000 (после фазы просмотра) | Пусто | Пусто |