Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и кибербезопасности
Высшая школа программной инженерии

КУРСОВОЙ ПРОЕКТ
МЕТОДЫ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ
по дисциплине «Архитектура программных систем»

Выполнил
студент гр. 5130904/30107 Л. А. Марков

Руководитель А. В. Гончаров

Санкт-Петербург
2025

Оглавление

Введение

Имитационное моделирование — метод, позволяющий строить модели, описывающие процессы так, как они проходили бы в действительности. Такую модель можно «проиграть» во времени как для одного испытания, так и заданного их множества. При этом результаты будут определяться случайным характером процессов. По этим данным можно получить достаточно устойчивую статистику.

К имитационному моделированию прибегают, когда:

- дорого или невозможно экспериментировать на реальном объекте;
- невозможно построить аналитическую модель: в системе есть время, причинные связи, последствия, нелинейности, стохастические (случайные) переменные;
- необходимо сымитировать поведение системы во времени.

Целью данного курсового проекта является спроектировать имитационную модель вычислительной системы. Одним из наиболее эффективных способов реализации таких моделей является представление системы в виде системы массового обслуживания (СМО). За основу была взята система городской доставки YA.rover (подробнее о системе изложено ниже).

Расшифровка варианта и представление СМО

Вариант №10: ИБ ИЗ2 ПЗ1 Д10З3 Д10О2 Д2П2 Д2Б5 ОР1 ОД1

- ИБ - Бесконечный источник
- ИЗ2 - Равномерный закон распределения времени генерации заявок источником
- ПЗ1 - Экспоненциальный закон распределения времени обслуживания заявки прибором
- Д10З3 - Постановка в буфер происходит на первое свободное место от начала
- Д10О2 - Дисциплина отказа по номеру. (Приоритеты по источникам: 1 > 2 > 3, где 1 - самый приоритетный источник)
- Д2П2 - Выбор прибора происходит по кольцу
- Д2Б5 - Выбор заявки из буфера по номеру источника. (Приоритеты по источникам: 1 > 2 > 3, пакетная выборка)
- ОР1 - Автоматический режим - сводная таблица результатов
- ОД1 - Пошаговый режим - календарь событий, буфер и текущее состояние.

Источники. Ими будут заявки на доставку. Число пользователей сервиса - не ограничено, с-но мы имеем бесконечный источник заявок. Заказы поступают в любое время дня, без сильных пиков и снижений - равномерный закон распределения.

Буфер памяти. В роли буфера будет выступать сервер “Центра управления Роверами” - очередь заказов, ожидающих свободного робота.

ДП (Диспетчер Постановки). Это “Менеджер приема и размещения заказов” - алгоритм, встроенный в “Центр управления Роверами”. Кладет новую заявку на первое свободное место. Если буфер переполнен, то в отказ отправляется старейшая заявка с наименьшим приоритетом.

Приборы. В их роли будут выступать роботы-курьеры (или Яндекс.Роверы). Имеется определенный ограниченный парк этих роботов. Время доставки (выполнение прибором одной задачи) - неизвестно, но более вероятны

быстрые заказы (так как вряд ли их будут отправлять куда-то дальше, чем текущий район).

ДВ (Диспетчер Выборки). “Планировщик заданий для роботов” - мозг "Центра управления Роверами", отдельный фоновый процесс, который постоянно отслеживает два состояния: 1) свободен ли кто-то из роботов? и 2) есть ли что в буфере?

Согласно варианту, выборка из БП осуществляется исходя из приоритета “пакета” источника. Мы будем иметь 3 источника:

- Источник 1 (наивысший приоритет) - заказы из сервиса “Яндекс.Здоровье”.
- Источник 2 (средний приоритет) - заказы из сервиса “Яндекс.Еда”.
- Источник 3 (низший приоритет) - заказы из сервиса “Яндекс.Лавка”.

Выбор прибора будет осуществляться “по кольцу”, т. е. ближайший к текущему значению указателя свободный робот.

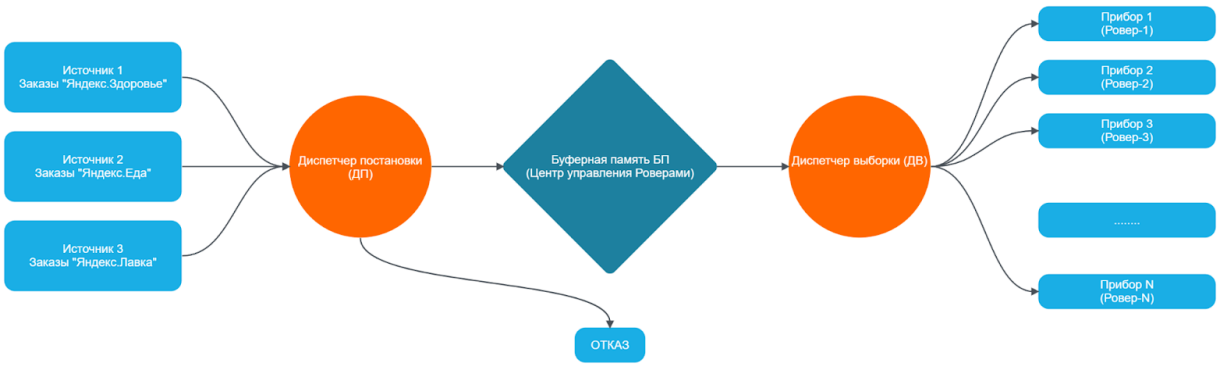


Рисунок 1: Визуализация СМО

Архитектура приложения

Диаграмма классов

https://github.com/mrkvv/YA.rover/blob/main/png/classDiagram.png

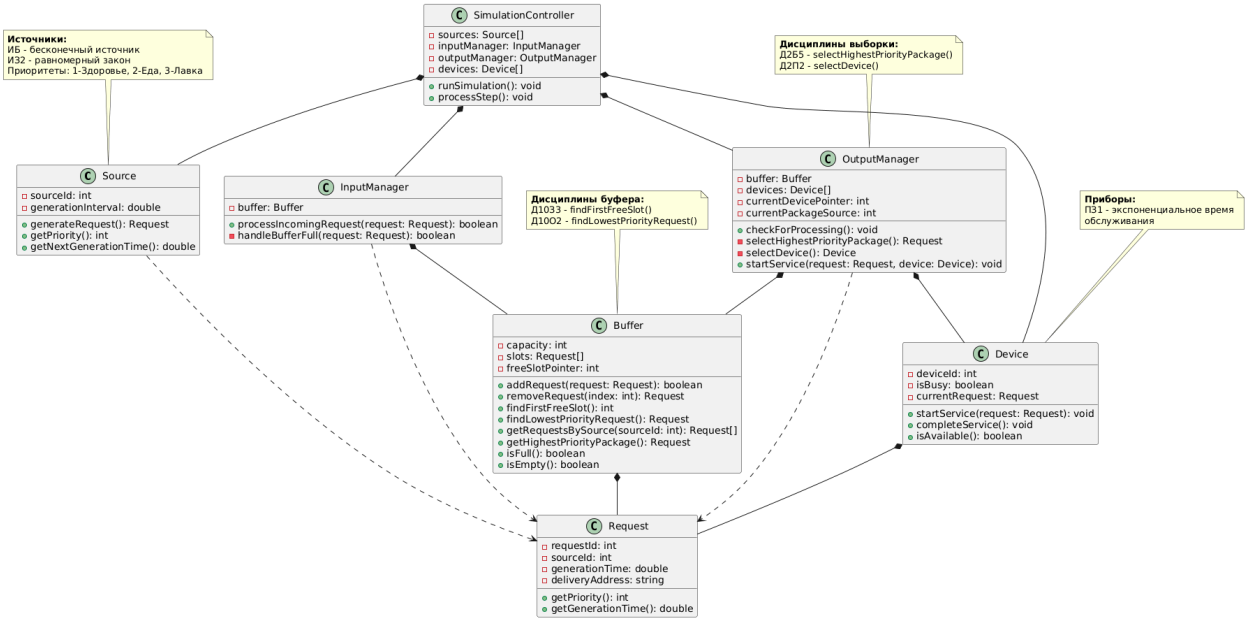


Рисунок 2: Диаграмма классов

Сиквенс диаграмма

https://github.com/mrkvv/YA.rover/blob/main/png/sequenceDiagram.png

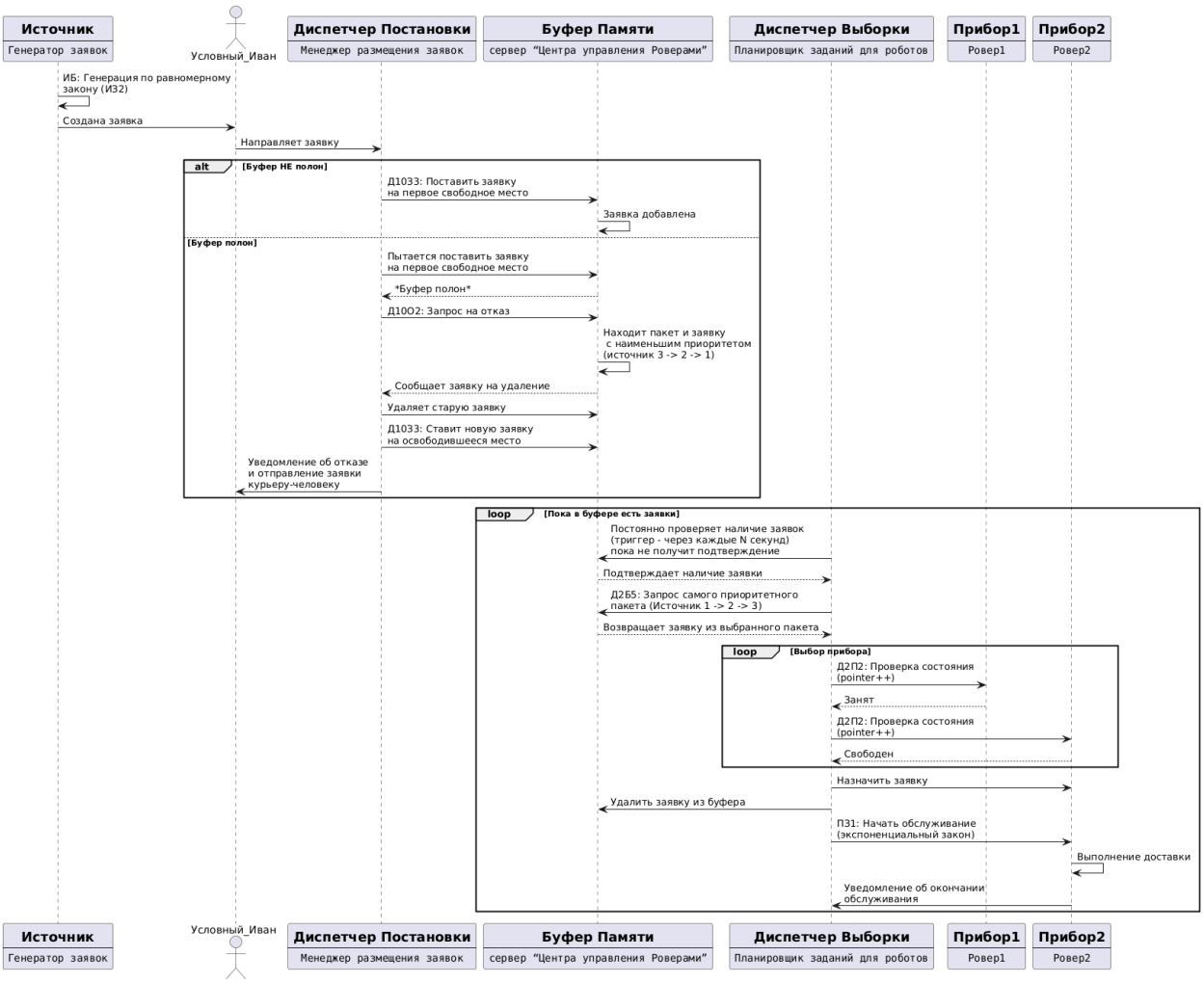


Рисунок 3: Сиквенс диаграмма

Flow-chart

https://github.com/mrkvv/YA.rover/blob/main/png/flowchart.png

Законы распределения

Равномерный закон (генерация заявок источником) — заявки могут появиться равновероятно в любой момент времени.

Экспоненциальный закон (время обслуживания заявки прибором) — более вероятны значения близкие к 0. Величину этой вероятности задает параметр лямда:

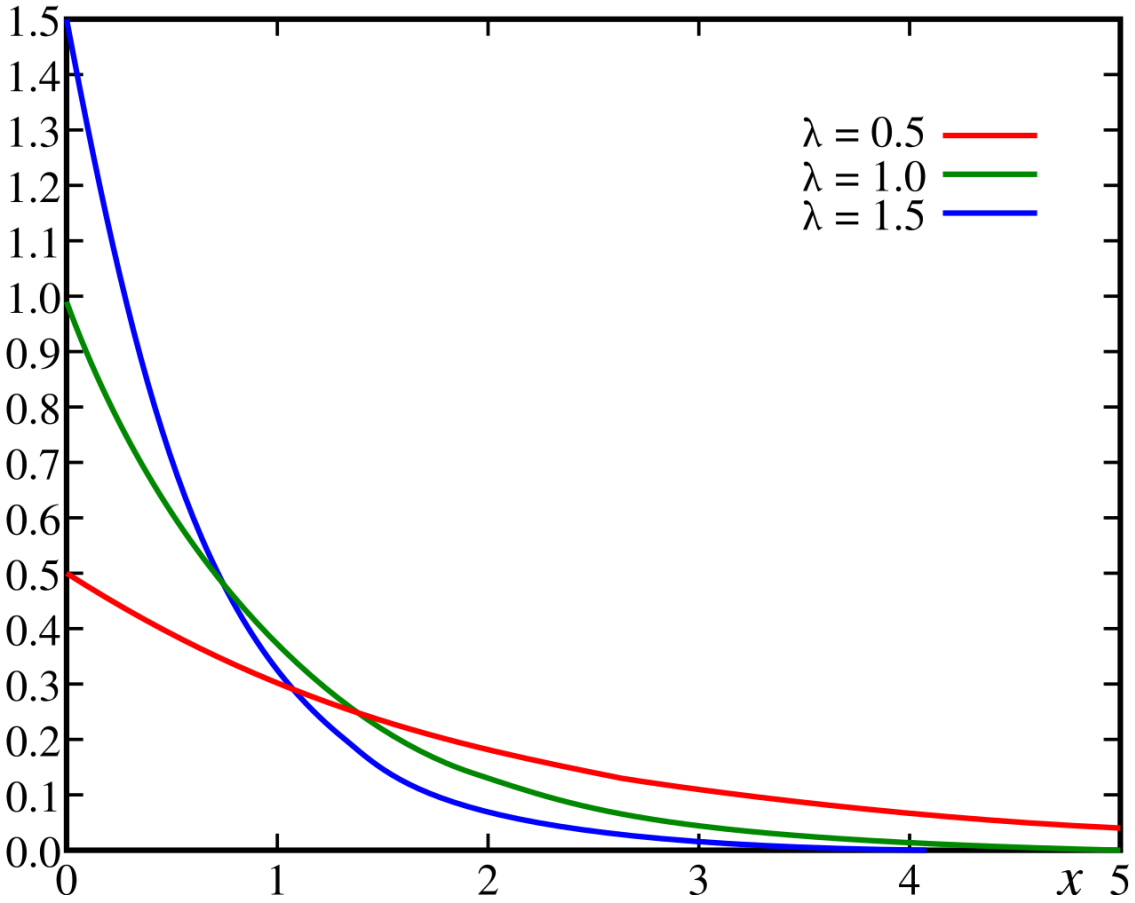


Рисунок 4: Функция плотности экспоненциального закона

В данном проекте параметр лямда был взят равным 1.0. Таким образом, мы еще и коррелируем со здравым смыслом, говоря, что приборы (наши яндекс.роверы) не ходят на большие дистанции, а доставляют заказы на ближайшие к ним точки.

Ограничения и требуемые характеристики

Количество реализаций, необходимое для получения нужной точности при заданной доверительной вероятности, можно оценивать по формуле:

$$N = \frac{t_a^2(1-p)}{p\sigma}$$

, где p—вероятность отказа заявкам в обслуживании, t =1.643 для α = 0.9, σ = 0.1 — относительная точность.

Было получено, что в большинстве случаев для достижения заданной точности необходимо около 2000 заявок.

Программная реализация C++

СМО была реализована на языке C++ и состоит из классов, описанных в диаграмме классов. Дополнительно для отображения результатов был введен класс EventCalendar, который обладает двумя методами:

- addEvent(Event event) — принимает на вход всю необходимую информацию о событии и сохраняет ее.
- printCalendar() – выводит весь имеющийся календарь в файл с именем «event_calendar.txt»

Константы, влияющие на симуляцию и их расположение:

SimulationController.h:

- START_TIME = 0 — время начала симуляции
- END_TIME = 20 — время окончания симуляции
- REQUESTS_PER_SOURCE_AMOUNT = 20 — количество заявок для поступления от каждого из источников

Buffer.h:

- CAPACITY = 10 – вместимость буфера

Rover.cpp:

- Rover::lambda = 1.0 – параметр экспоненциального распределения времени выполнения заявки прибором

OutputManager.h:

- ROVERS_AMOUNT = 4 – количество приборов (роверов) в парке

main.cpp:

- MODE = true – режим симуляции. True – автоматика, false – пошаговый.

Результаты работы программы

Пошаговый режим

В пошаговом режиме по каждому нажатию на клавишу “Enter” будет проходить одна единица времени программы, а так же будет выводится текущее состояние отслеживаемых компонентов: заполненность и внутренности буфера, занятость приборов (роверов), указатель на следующего ровера и текущий пакет к выполнению.

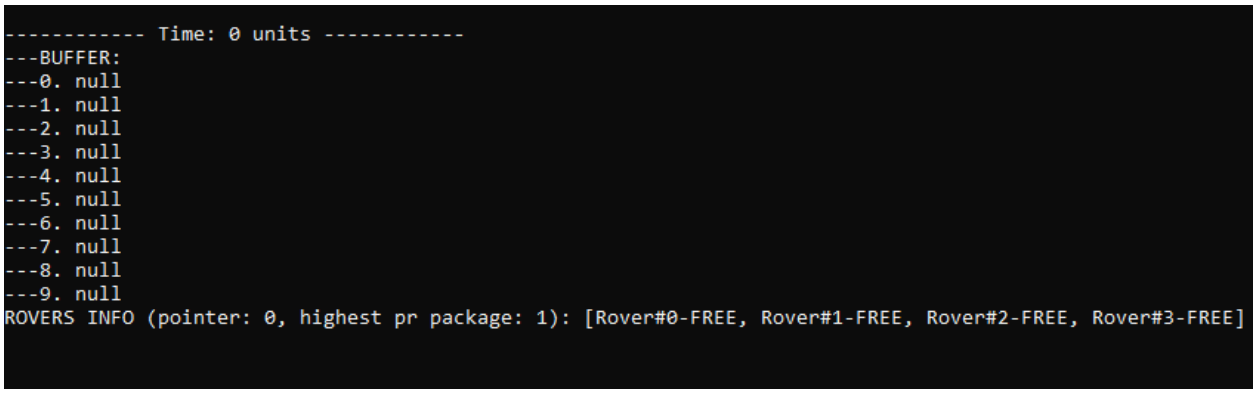


Рисунок 5: Состояние системы в момент запуска

Генерация заявок отображается в консоли синим цветом, а взятие заявок отображается желтым.

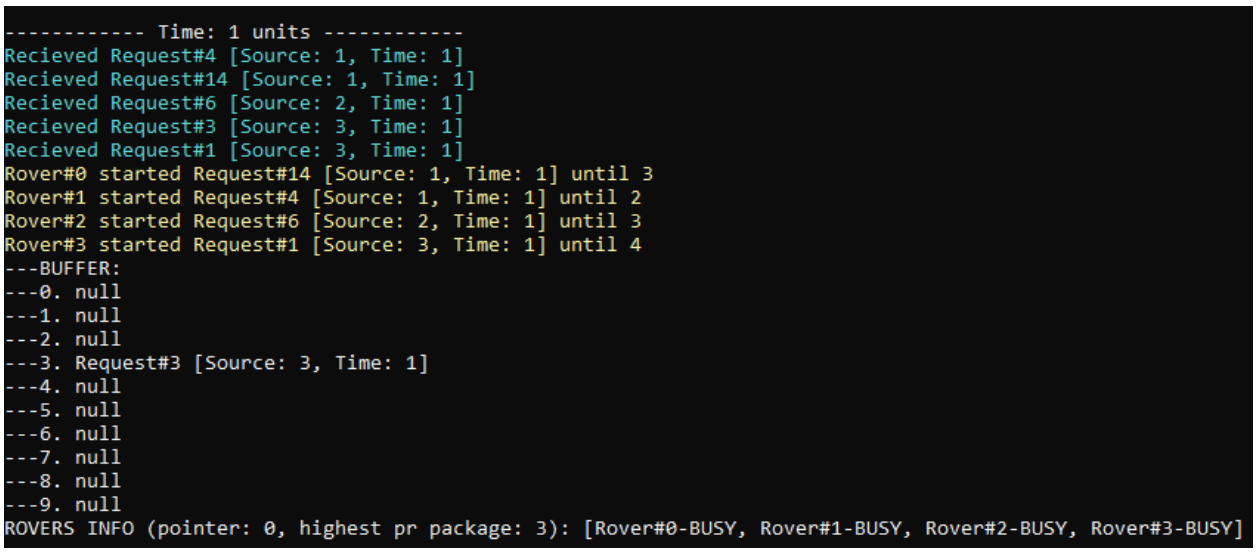


Рисунок 6: Состояние системы после первого "Enter'а" в одном из запусков программы

Окончание выполнения заявки обозначается зеленым цветом.

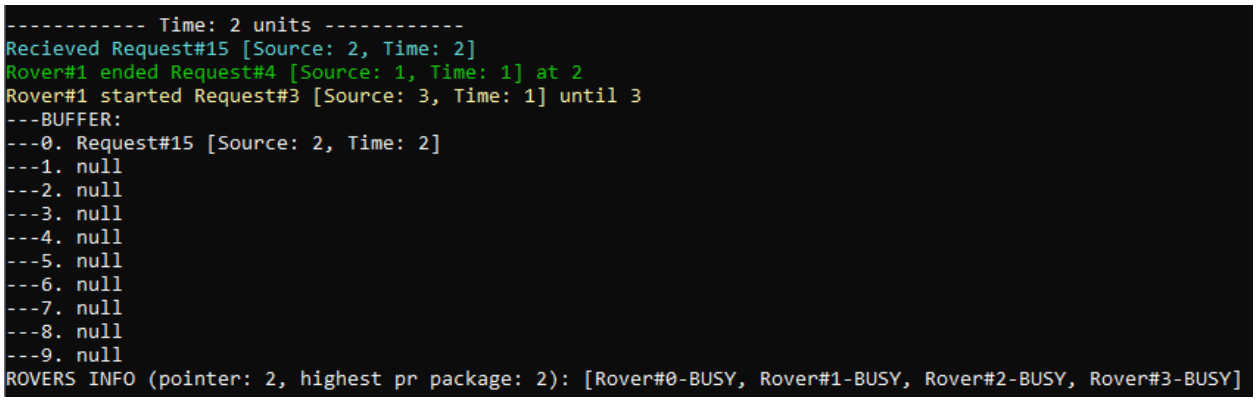


Рисунок 7: Завершение выполнения заявки (зеленый)

В ходе симуляции может случится выбивание заявки из буфера. В таком случае программа оповестит о выбивании и сообщит, какая заявка вместо какой была поставлена в буфер:

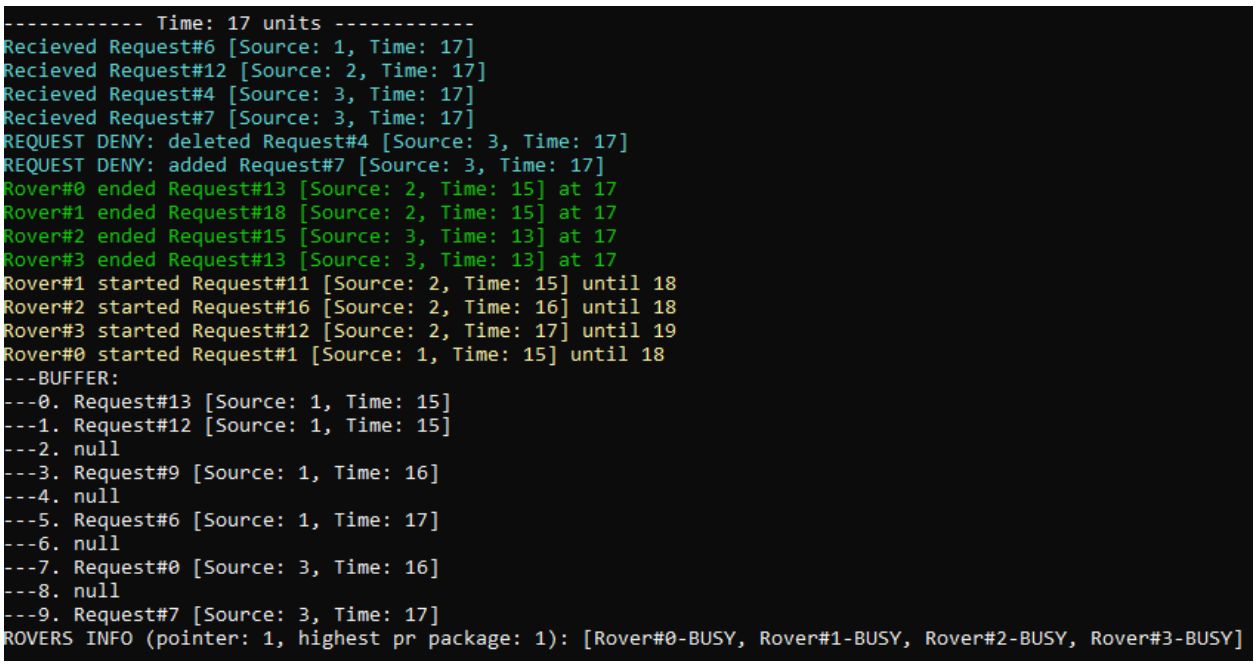


Рисунок 8: Выбивание заявки из буфера

После прохода всех шагов симуляции в корне проекта появится файл «event_calendar.txt» с таблицей следующего вида:

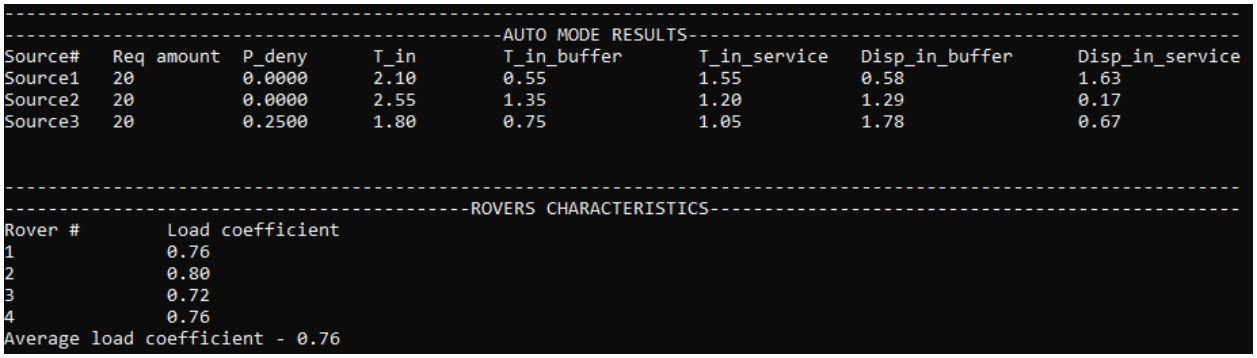
КАЛЕНДАРЬ СОБЫТИЙ

Время	Тип	Компонент	Описание
1	Создание заявки	Источник	Source#1 Request#4
1	Создание заявки	Источник	Source#1 Request#14
1	Создание заявки	Источник	Source#2 Request#6
1	Создание заявки	Источник	Source#3 Request#3
1	Создание заявки	Источник	Source#3 Request#1
1	Добавление заявки в буфер	Диспетчер постановки	Добавлена заявка Source#2 Request#6
1	Добавление заявки в буфер	Диспетчер постановки	Добавлена заявка Source#1 Request#4
1	Добавление заявки в буфер	Диспетчер постановки	Добавлена заявка Source#3 Request#3
1	Добавление заявки в буфер	Диспетчер постановки	Добавлена заявка Source#3 Request#1
1	Добавление заявки в буфер	Диспетчер постановки	Добавлена заявка Source#1 Request#14
1	Выбор заявки на обработку	Диспетчер выборки	Из буфера взята заявка Source#1 Request#14
1	Выбор заявки на обработку	Диспетчер выборки	Из буфера взята заявка Source#1 Request#4
1	Выбор заявки на обработку	Диспетчер выборки	Из буфера взята заявка Source#2 Request#6
1	Выбор заявки на обработку	Диспетчер выборки	Из буфера взята заявка Source#3 Request#1
1	Отправка заявки на обработку	Диспетчер выборки	Заявка Source#1 Request#14 отправлена на выполнение для Rover#0
1	Отправка заявки на обработку	Диспетчер выборки	Заявка Source#1 Request#4 отправлена на выполнение для Rover#1
1	Отправка заявки на обработку	Диспетчер выборки	Заявка Source#2 Request#6 отправлена на выполнение для Rover#2
1	Отправка заявки на обработку	Диспетчер выборки	Заявка Source#3 Request#1 отправлена на выполнение для Rover#3
2	Создание заявки	Источник	Source#2 Request#15
2	Добавление заявки в буфер	Диспетчер постановки	Добавлена заявка Source#2 Request#15
2	Выбор заявки на обработку	Диспетчер выборки	Из буфера взята заявка Source#3 Request#3
2	Отправка заявки на обработку	Диспетчер выборки	Заявка Source#3 Request#3 отправлена на выполнение для Rover#1
3	Создание заявки	Источник	Source#1 Request#8
3	Создание заявки	Источник	Source#1 Request#11
3	Создание заявки	Источник	Source#3 Request#19
3	Создание заявки	Источник	Source#3 Request#6
3	Создание заявки	Источник	Source#1 Request#19
3	Создание заявки	Источник	Source#1 Request#3
3	Создание заявки	Источник	Source#1 Request#5
3	Добавление заявки в буфер	Диспетчер постановки	Добавлена заявка Source#3 Request#6
3	Добавление заявки в буфер	Диспетчер постановки	Добавлена заявка Source#3 Request#19
3	Добавление заявки в буфер	Диспетчер постановки	Добавлена заявка Source#1 Request#19

Рисунок 9: Календарь событий

Анализ результатов автоматического режима

В результате симуляции в автоматическом режиме мы получим следующие таблицы:



AUTO MODE RESULTS

Source#	Req amount	P_deny	T_in	T_in_buffer	T_in_service	Disp_in_buffer	Disp_in_service
Source1	20	0.0000	2.10	0.55	1.55	0.58	1.63
Source2	20	0.0000	2.55	1.35	1.20	1.29	0.17
Source3	20	0.2500	1.80	0.75	1.05	1.78	0.67

ROVERS CHARACTERISTICS

Rover #	Load coefficient
1	0.76
2	0.80
3	0.72
4	0.76

Average load coefficient - 0.76

Рисунок 10: Автоматический режим: результаты

Растолковать результаты конкретно этой выборки просто:

- *Req amount*: Каждый источник сгенерировал по 20 заявок
- *P_deny*: Вероятность отказа в обслуживании (выбивания заявки) у источников 1 и 2 составляет 0%, то есть все заявки были обслужены в данной выборке. А у источника 3 вероятность отказа составляет целых 25%, что говорит о том, что 5 заявок от 3-го источника были выбиты.
- *T_in:* Среднее время нахождения в системе у заявок от источников. В среднем наибольшее время в системе у заявок источника 2. Теоретически, заявки от источника 3 должны быть в системе наибольшее количество времени, но по данной выборке вышло совсем наоборот. Объяснение этому кроется в следующих двух параметрах.
- *T_in_buffer*: Среднее время нахождения заявки в буфере. У заявок источника 1 это время наименьшее, вероятно в следствии их приоритетности перед другими, тут все штатно. Однако вторым по времени является источник 3, возникает вопрос почему? — тут свое влияние оказали выбивания из буффера — целых 25% от всех заявок 3-го источника были выбиты, а следовательно выбитые заявки и повлияли на таковой результат.
- *T_in_service*: Среднее время обслуживания заявки. Теоретически, это время должно быть у всех одинаковым, так как на время выполнения не влияет приоритетность источника, а влияет только экспоненциальный закон распределения и его параметр лямда. Но по данной выборке мы получаем неимоверно быстрое выполнение заявок от 3-го источника, что лишь является результатом статистики и вероятности (среди десяти бросков монеты навряд-ли будут 5 орлов и 5 решек в силу ограниченности выборки).
- *Disp_in_buffer*: Дисперсия величины T_in_buffer, показывает наличие выбросов, считая от среднего.
- *Disp_in_service*: Дисперсия величины T_in_service.

И говоря о занятости приборов, мы получаем среднее значение коэффициента равное 0.76 по выборке. Такой результат мог появиться вследствие статистики и техники подсчета (общее время занятости ровера / [общее время симуляции + время до завершения всех текущих заказов]). В основном все роверы работали все время, за исключением «доработок» после основного времени, когда работала только лишь часть из них.

Анализ наилучшей конфигурации и подсчет средней стоимости

Необходимо подобрать 2 основных параметра:

1. Количество роверов
2. Вместимость буфера

так, чтобы:

- Количество приборов -> min
- Количество мест в буфере -> min
- Среднее значение коэффициента занятости приборов > 0.9
- Вероятность отказа < 0.1
- Среднее время пребывания заявки -> min

Прикинем стоимость в удельных единицах (у. е.):

- 1 яндекс доставщик (ровер) ~ 200 у.е. закупка + 20 у. е./мес
- 1 место в буфере ~ 3 у. е./мес
- Диспетчер постановки ~ 500 у.е. закупка + 20 у. е./мес
- Диспетчер выборки ~ 800 у.е. закупка + 25 у. е./мес
- Интеграция с сервисами Я.Лавка, Я.Еда, Я.Здоровье ~ 300 у. е.
- Так же стоит учитывать, что могут быть непредвиденные расходы и вообще вся эта прикидка лишь прикидка.

Пусть R – количество роверов, а B – количество мест в буфере, тогда, финансово это обойдется в:

- Единоразово: R*200 + 500 + 800 + 300 = 1600 + R*200
- Ежемесячно: R*20 + B*3 + 20 + 25 = 45 + R*20 + B*3

Далее, в подсчете оптимальной конфигурации, будем сравнивать только подчеркнутые части суммы, так как все остальные — константы.

Подбор наиболее оптимальной конфигурации.

Теперь необходимо подобрать *наиболее оптимальную конфигурацию*. Будем считать, что приоритетнее снизить количество роверов, так как они много дороже в закупке и месячном обслуживании.
Так же, стоит уточнить, что количество источников фиксировано, то есть их всегда 3 и в таблице оно присутствует для наглядности.

Будем смотреть на 2 основных параметра: Емкость буфера и кол-во роверов. Допустимые конфигурации для нас, это такие, в которых

- $P\_deny < 0.1$
- $T\_in \rightarrow min$
- Занятость приборов > 0.9

Будем отмечать:

- красным — недопустимые значения, а значит и вся конфигурация недопустима.
- зеленым — допустимые к рассмотру конфигурации.
- желтым — граничные значения, от которых мы будем смотреть их «окрестность», чтобы найти конфигурации, которые можно будет пометить зеленым.
- красным шрифтом — допустимые, но по финансовым или логическим соображениям не оптимальные варианты.

Будем заполнять столбцы P_deny, T_in и Занятость приборов средними значениями по результатам измерений из нескольких выборок.

Рассмотрим R = 3:

Кол-во Заявок	Кол-во Источников	B	R	P_deny	T_in	Занятость приборов
60	3	10	3	0.25	3	0.92
60	3	12	3	0.2	3.5	0.97
60	3	18	3	0.1	4	0.99
60	3	22	3	0.02	4.4	1.00

Получили:

- Разово: 3*200 = 600 у.е.
- Ежемесячно: 3*20 + 22*3 = 126 у.е./мес

Теперь для R = 2:

Кол-во Заявок	Кол-во Источников	B	R	P_deny	T_in	Занятость приборов
60	3	30	2	0.1	11	1.00
60	3	40	2	0	15	1.00
60	3	34	2	0.04	14	1.00

Получили:

- Разово: 2*200 = 400 у.е.
- Ежемесячно: 2*20 + 34*3 = 142 у.е./мес

Разница с R = 3 составляет всего 200 у.е. и 16 у.е./мес, но при этом мы жертвуем ~10 единиц времени на обработку заявки в среднем. Таким образом, данный вариант нерентабелен и нам не подходит.

Лучший вариант (пока): R = 3, B = 22

Пусть R = 4:

Кол-во Заявок	Кол-во Источников	B	R	P_deny	T_in	Занятость приборов
60	3	10	4	0.05	3	0.92

С первой попытки нашлась оптимальная конфигурация для R=4, получили:

- Разово: 4*200 = 800 у.е.
- Ежемесячно: 4*20 + 10*3 = 110 у.е./мес

Разница с R = 3 составляет 200 у.е. и 16 у.е./мес, что со временем будет рентабельнее. Так же, стоит отметить, что мы выигрываем по среднему времени обработки заявки, она изменилась на 1.4. Таким образом, теперь эта конфигурация является наилучшей.

Лучший вариант (пока): R = 4, B = 10

Рассматривать R = 5 нет смысла, так как ежемесячное обслуживание составит минимум 100 у.е. в месяц, что отличается от текущего лучшего результата всего на 10 у.е./мес, но прибавляет целых 200 у.е. в единоразовую стоимость, что не рентабельно. Аналогично R > 5.

Таким образом, конфигурация R = 4. B = 10 – лучшая при таких исходных данных (60 заявок, что является оптимальным по отношению к реальным условиям). **Попробуем повысить количество заявок** и посмотреть, что будет и как нужно на это реагировать:

Кол-во Заявок	Кол-во Источников	B	R	P_deny	T_in	Занятость приборов
90	3	10	4	0.3	2.4	0.96
90	3	26	4	0.2	6	1.00
90	3	20	5	0.1	4.1	1.00
90	3	26	5	0.06	4.6	1.00

Конечно же, если оставить конфигурацию, рассчитанную на 60 заявок, то вероятность отказа взлетает пропорционально количеству заявок.
В таком случае, попробуем повысить B, получаем при B = 26: P_deny = 0.2, что сразу нам говорит, что дальше повышать B попросту нерентабельно и все же лучше сразу докупить приборы (R).
При R = 5, B = 20 получаем граничный результат по P_deny.
Тогда добавив 6 мест в буфере получаем необходимый результат.

Таким образом, был рассмотрен алгоритм подбора значений B и R под текущую нагрузку на систему. Если мы хотим увеличить быстродействие системы, то следует докупить приборы (и подсократить места в буфере, для экономности). Если мы хотим уменьшить вероятность отказа, то следует:

- либо увеличить B, пожертвовав быстродействием, но получив финансовую выгоду (места в буфере намного дешевле, чем целый ровер),
- либо увеличить R, пожертвовав выгодой, но получив быстродействие,
- либо балансированно увеличить оба показателя, что, собственно, и рекомендуется.

В данном же проекте, мы остановимся на 60 заявках и будем считать, что это оптимальное количество, которое и будет встречено на практике.

Заключение

В результате выполнения курсового проекта была смоделирована система массового обслуживания городской доставки по мотивам YA.rover, а так же посчитаны примерные затраты для приобретения таковой системы, запуска ее в работу и месячного содержания.
Итого, получилось, что единоразовые и ежемесячные затраты составят (при R = 4, B = 10):

- Единоразово: 1600 + 4*200 = 2400 у.е.
- Ежемесячно: 45 + 4*20 + 10*3 = 155 у.е./мес

Данное вычисление было проведено на момент декабря 2025 года из расчета 1 у.е. = 100 руб.

Так же был разработан и продемонстрирован алгоритм масштабирования системы при увеличении нагрузки на нее на 50%. При необходимости можно воспользоваться данным алгоритмом и подобрать под абсолютно любую нагрузку.