

Кафедра вычислительной техники Группа Р3317

КУРСОВАЯ РАБОТА

по дисциплине «Моделирование»

на тему «Моделирование дискретных систем»

Автор(ы) Плюхин Д.А.
(Фамилия, И.О.)

Руководитель Соснин В.В., к.т.н.
(Фамилия, И.О., ученое звание, степень)

Курсовая работа выполнена с оценкой _____

Дата защиты “ ____ ” _____ 20 ____ г.

Санкт-Петербург, 2017 г.

Оглавление

Разработка концептуальной модели объекта исследования	3
Описание модели	3
Схема модели.....	5
Цель моделирования	5
Разработка имитационной модели Anylogic	6
Основные дополнительные упрощения и допущения	6
Скриншоты модели	6
Сценарии работы модели	28
Обычный режим работы.....	28
Режим работы во время эпидемии гриппа.....	30
Режим работы в разгар лета.....	32
Режим работы в случае образования повышенного количества модников	34
Некоторые обобщения.....	36
Разработка имитационной модели SimPy	37
Дополнительные замечания	37
Исходный код.....	38
Сценарии работы модели	66
Обычный режим работы.....	66
Режим работы во время эпидемии гриппа:	67
Режим работы в разгар лета.....	69
Режим работы в случае образования повышенного количества модников	71
Некоторые обобщения.....	73

Разработка концептуальной модели объекта исследования

Описание модели

В данной курсовой работе объектом моделирования является система обслуживания парикмахерской, поток заявок неоднородный, каждый транзакт представляет собой модель клиента парикмахерской, таким образом в рамках моделирования используются три класса заявок: клиенты, которые стригутся под одну насадку, клиенты, желающие модельную стрижку и клиенты, которые пришли на покраску волос. Также реализуется система приоритетов - право на обслуживание вне очереди имеют пенсионеры и инвалиды.

В реализуемой модели парикмахерской 6 узлов:

- 1) Касса у входа в парикмахерскую
- 2) Зал стрижки под одну насадку
- 3) Зал модельных стрижек
- 4) Зал покраски волос
- 5) Зал ожидания после покраски
- 6) Стол с книгой отзывов и предложений

В процессе выполнения работы были приняты следующие допущения и использованы следующие предположения:

- 1) Модельная стрижка для женщин занимает не менее одного часа (источник - парикмахер с 15-летним стажем с форума <http://www.woman.ru/psycho/career/thread/3841557/>), для мужчин, поскольку стрижка проще, составляет как минимум 40 минут.
- 2) Стрижка под одну насадку занимает в среднем 20-25 минут (информация с того же форума)
- 3) Покраска волос занимает около 10 минут (информация с форума <https://otvet.mail.ru/question/67178187>)
- 4) После покраски нужно ждать около 30 минут (информация с того же форума) в зале ожидания
- 5) После ожидания после покраски нужно ждать около 5 минут производить просушку (источник тот же)
- 6) Парикмахерская расположена во Фрунзенском районе Санкт-Петербурга и её клиентская база составляет 2% населения этого района, то есть, около 8120 человек (информация о населении района в 2017 году взята с Википедии), из которых 3700 человек - мужчины, остальные - женщины (процентное соотношение соответствует процентному соотношению мужчин и женщин в Санкт-Петербурге по информации с Википедии)
- 7) С учетом средней скорости роста волос 1-1,5 см и максимального размера насадки в 2 см (информация с сайта alergan.ru) *мужчины, которые стригутся под насадку, для поддержания стрижки будут приходить раз в 1 - 2 месяца. Это значит, что в среднем в день приходит 20-30 человек при условии, что количество людей, стригущихся под одну насадку, по наблюдением автора курсовой, значительно меньше числа людей, требующих модельную стрижку.*
- 8) С учетом средней скорости роста волос 1-1,5 см и максимального размера насадки в 2 см (информация с сайта alergan.ru) *мужчины, которые носят модельную стрижку, с учетом большей её стоимости, для поддержания стрижки будут приходить раз в 2 - 3 месяца. Это значит, что в среднем в день приходит 5-*

10 человек. Что касается женщин - для них отдельные виды модельной стрижки стоят еще дороже, однако необходимость поддержания прически острее, поэтому в среднем общее количество женщин в день будет составлять примерно в полтора раза большую долю, чем мужчин и окажется на уровне 40-60 человек в день с учетом того, что часть из них приходит только на покраску.

9) Поскольку покрасить волосы самостоятельно значительно проще, чем самому постричься, женщины по большей части красят волосы дома и лишь небольшая их доля приходят для этого в парикмахерскую (в том числе и женщины, которым нужна не однотонная покраска). Их количество составит около 15-25 человек в день.

10) В случае занятости стола с книгой отзывов и предложений клиент уходит, так и не оставив свое мнение

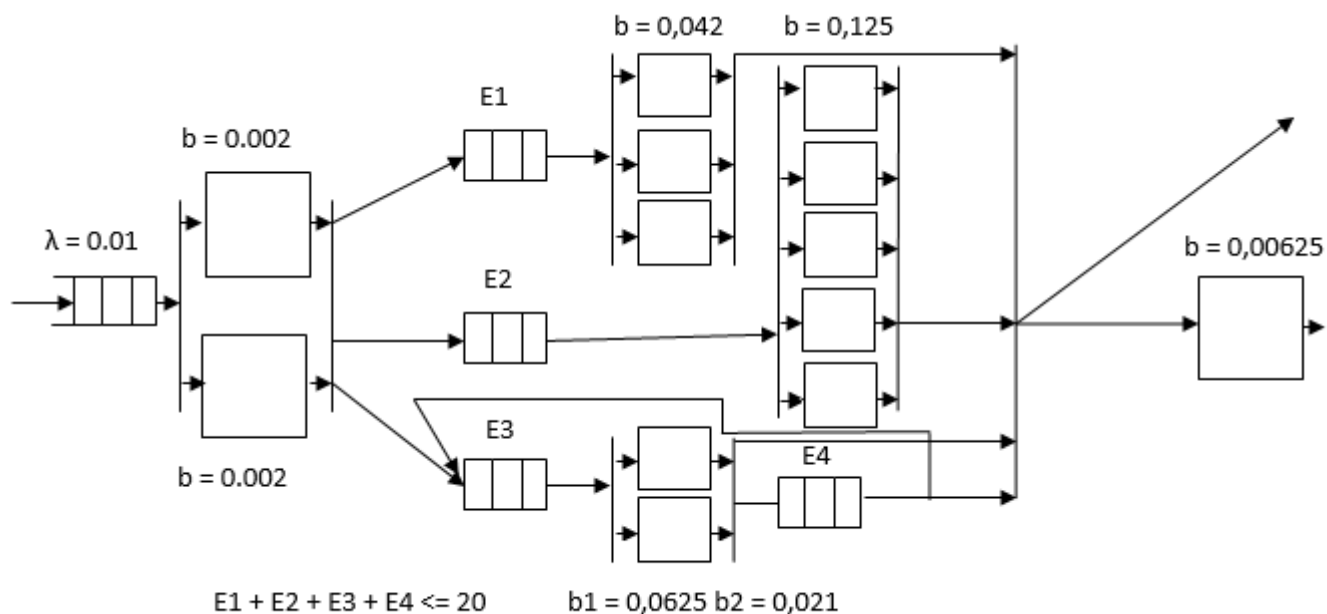
11) По минимуму обслуживание клиента включает приветствие (1-2 секунды), вопрос - ответ о желаемой стрижке (около 5 секунд), ожидание платежных средств от клиента (от 5 до 10 секунд) и проведение операции с кассой (также от 5 до 10 секунд) и составляет в целом 15-30 секунд. Однако нельзя не предусмотреть желания клиента проконсультироваться по какому-либо вопросу, на который либо тут же будет дан ответ, либо будет произведена постановка на очередь в зал с последующей консультацией у парикмахера, после которой клиент, вероятно, не удовлетворится ответом и покинет парикмахерскую (с не очень большой вероятностью, поскольку количество новых клиентов значительно меньше числа клиентов, состоящих в клиентской базе). Тем не менее, отведем на консультацию 30 секунд - именно в течение этого промежутка может быть задан вопрос и дан ответ, если вопрос является достаточно простым. В итоге получаем минимальное время обслуживания в кассе 45-60 секунд без учета того, что под конец рабочего дня кассир постепенно устает и начинает обслуживать клиентов медленнее приблизительно в 2 раза. С учетом этого фактора, а также необходимых коротких перерывов, получим разброс во времени обслуживания от 1 минуты (0,017 часа) до 5 минут (0,083 часа) со значительным смещением в область быстрого обслуживания.

12) Количество касс - 2, количество парикмахеров в зале стрижки под насадку - 3, в зале модельной стрижки - 5, в зале покраски - 2, количество столов с книгой отзывов и предложений - 1

13) Интервалы времени между новыми заявками и время обслуживания заявок в каждом узле распределены по экспоненциальному закону.

14) Очередь на кассу бесконечна, а общая очередь на все виды стрижек для обеспечения комфорта клиентов не может превышать 20 человек и обусловлена размерами зала ожидания с мягкими диванами и телевизором.

Схема модели



Цель моделирования

критерий эффективности

= количество отзывов в день — количество работающих мастеров — размер очереди в зале ожидания
— размер очереди в кассу

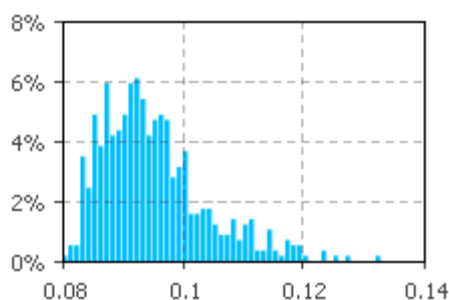
Сценарии модернизации системы обслуживания:

1. Перевод мастеров из одних залов в другие, увольнение лишних мастеров
2. Изменение количества касс до оптимального значения
3. Добавление второго стола с книгой отзывов и предложений для сбора большего количества мнений клиентов о предприятии, возможно, за счет уменьшения свободной площади в зале ожидания (уменьшения максимальной суммарной очереди)

Разработка имитационной модели Anylogic

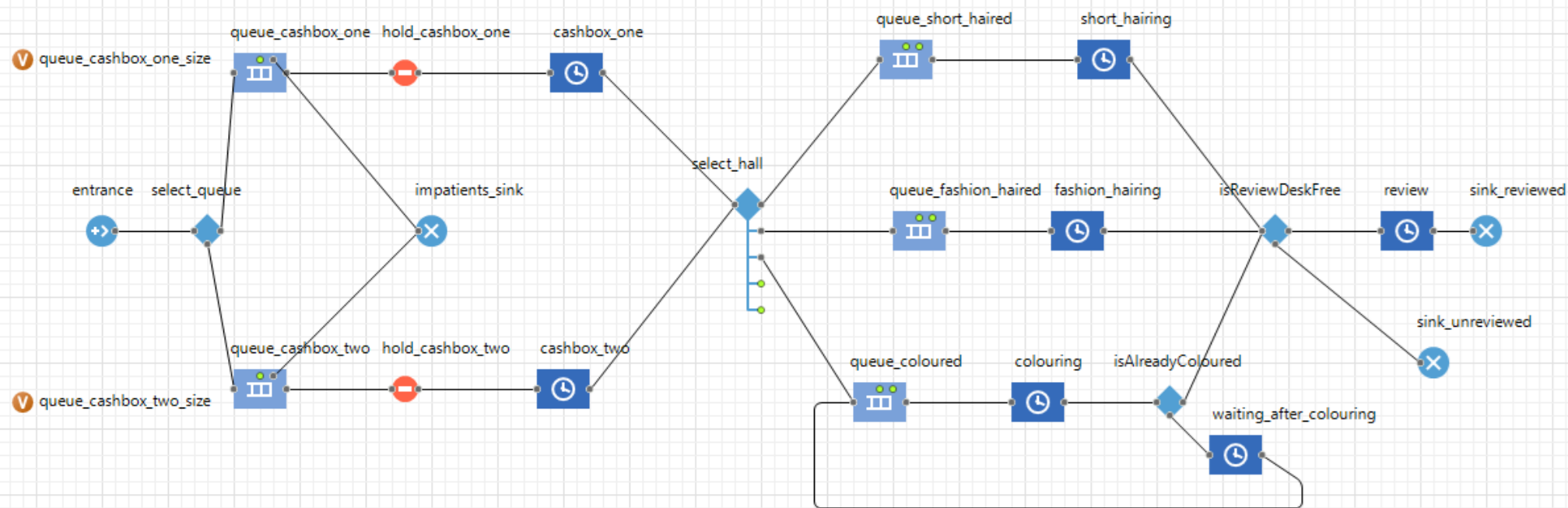
Основные дополнительные упрощения и допущения

- 1) Классы заявок кодируются в поле type типа данных Request целыми числами:
 - a. Мужчины, пришедшие на стрижку под насадку – 1
 - b. Женщины и мужчины, пришедшие на модельную стрижку – 2
 - c. Женщины, пришедшие на покраску – 3
- 2) Соотношение количества заявок этих типов составляет в среднем $25 : (50 - 20 + 7) : 20 = 25:37:20$, то есть, вероятности появления заявок каждого типа составят соответственно 0.30, 0.45, 0.25.
- 3) Приоритет заявок кодируется в поле priority типа данных Request целыми числами:
 - a. Инвалиды и пенсионеры – 1
 - b. Остальные клиенты – 0
- 4) По информации с сайта http://sptoday.ru/2013_11_28/petrostat-naschital-v-peterburge-700-tysyach-invalidov-iz-nix-14-tysyach-deti/ количество инвалидов в Санкт-Петербурге без учета пенсионеров составляет около 120 тыс. человек. Пенсионеров же в Санкт-Петербурге, по информации с сайта <http://www.the-village.ru/village/city/situation/246267-pensioner>, почти 1 360 тыс. человек. Примем, что вероятность появления высокоприоритетной заявки равна соотношению суммарного количества пенсионеров и инвалидов в Санкт-Петербурге и населения Санкт-Петербурга, то есть, равна 0.28.
- 5) В соответствии с предыдущими выводами, в день будет приходить от 60 до 100 человек – это значит, что интервал между заявками находится в пределах 4.8 (0.08 часа) – 8 (0.13 часа) минут. Будем исходить из предположения, что в этом диапазоне интервал между заявками имеет вид гамма – распределения с параметрами $\alpha = 3$; $\beta = 0.3$.



- 6) Будем исходить из предположения, что все времена обслуживания также распределены по гипер-экспоненциальному закону
- 7) Клиент в первую очередь пытается пройти на первую кассу, но, если во второй очереди оказывается меньше человек, он идет туда.
- 8) Будем исходить из предположения, что время обслуживания заявок в каждой из 2 касс имеет вид гамма – распределения с параметрами $\alpha = 1$; $\beta = 0.8$, поскольку быстрое обслуживание клиентов значительно более вероятно, однако, редкие перерывы в работе кассира не невозможны.
- 9) Как только суммарное количество человек в зале ожидания парикмахерской достигнет 20, люди из очередей в кассы перестают обслуживаться до тех пор, пока место в зале не освободится.
- 10) По информации с сайта <http://udoktora.net/pochemu-prebyivanie-v-ocheredi-nas-trevozhit-29476/> через 5 минут ожидания в очереди половина людей покидают её. На основании этого наблюдения будем предполагать, что половина людей покидает очередь в интервале 4 – 5 минут, и в обе стороны оставшееся количество распределено одинаково (до 1 минуты и 9 минут соответственно), так что по виду напоминает линейную зависимость. Для моделирования распределения времени покидания в очередь будем использовать эмпирическое распределение.

Скриншоты модели



Имя: write_results ☒ Отображать имя ☐ Исключить

Видимость: ☒ да

☒ Действие (не возвращает ничего)

☐ Возвращает значение

Аргументы

Имя	Тип
index	int
sindex	int
sheet_name	String

Тело функции

```
experiments_results.setCellValue(entrance.count(),sheet_name+"!" +column_names_set.get(sindex).get(index)+"1");
experiments_results.setCellValue(cashbox_one_impatients_counter/(cashbox_one_patients_counter + cashbox_one_impatients_counter),
sheet_name+"!" +column_names_set.get(sindex).get(index)+"2");
experiments_results.setCellValue(cashbox_two_impatients_counter/(cashbox_two_patients_counter + cashbox_two_impatients_counter),
sheet_name+"!" +column_names_set.get(sindex).get(index)+"3");
experiments_results.setCellValue(review_impatients_counter/(review_counter + review_impatients_counter),
sheet_name+"!" +column_names_set.get(sindex).get(index)+"4");

experiments_results.setCellValue(queue_cashbox_one_input_intensity.getYMean()/cashbox_one_service_intensity.getYMean(),
sheet_name+"!" +column_names_set.get(sindex).get(index)+"6");
if (cashbox_two_service_intensity.getYMean() != 0){
    experiments_results.setCellValue(queue_cashbox_two_input_intensity.getYMean()/cashbox_two_service_intensity.getYMean(),
    sheet_name+"!" +column_names_set.get(sindex).get(index)+"7");
}
experiments_results.setCellValue(queue_short_haired_input_intensity.getYMean()/short_haired_service_intensity.getYMean(),
sheet_name+"!" +column_names_set.get(sindex).get(index)+"8");
experiments_results.setCellValue(queue_fashion_haired_input_intensity.getYMean()/fashion_haired_service_intensity.getYMean(),
sheet_name+"!" +column_names_set.get(sindex).get(index)+"9");
experiments_results.setCellValue(queue_coloured_input_intensity.getYMean()/coloured_service_intensity.getYMean(),
sheet_name+"!" +column_names_set.get(sindex).get(index)+"10");
experiments_results.setCellValue(review_input_intensity.getYMean()/review_service_intensity.getYMean(),
sheet_name+"!" +column_names_set.get(sindex).get(index)+"11");

experiments_results.setCellValue(queue_cashbox_one.statsSize.mean(),sheet_name+"!" +column_names_set.get(sindex).get(index)+"13");
experiments_results.setCellValue(queue_cashbox_two.statsSize.mean(),sheet_name+"!" +column_names_set.get(sindex).get(index)+"14");
experiments_results.setCellValue(queue_short_haired.statsSize.mean(),sheet_name+"!" +column_names_set.get(sindex).get(index)+"15");
experiments_results.setCellValue(queue_fashion_haired.statsSize.mean(),sheet_name+"!" +column_names_set.get(sindex).get(index)+"16");
experiments_results.setCellValue(queue_coloured.statsSize.mean(),sheet_name+"!" +column_names_set.get(sindex).get(index)+"17");
```

Специфические

☐ Статическая

Уровень доступа: по умолчанию ▼

☐ Единицы измерения (сист. динамика):

Описание

Имя: ☒ Отображать имя ☐ Исключить

Тип: ☒ Определенное время
☐ Пока не вызван метод stopDelay()

Время задержки: минуты

Вместимость:

Максимальная вместимость:

Место агентов:

▼ Специфические

Выталкивать агентов:

Вернуть агента в исходную точку:

Включить сбор статистики:

▼ Действия

При входе:

При подходе к выходу:

При выходе:

При извлечении:

▼ Специфические

Тип агента:

☒ Одиночный агент ☐ Популяция агентов

Модель/Библиотека: Библиотека моделирования процессов [\(Изменить...\)](#)

Видимость: ☒ да

☐ Отображается на верхнем уровне

☒ Записывать лог в базу данных
[Включить логирование выполнения модели](#)

► Описание

Имя: ☒ Отображать имя ☐ Исключить

Видимость: ☒ да

☒ Действие (не возвращает ничего)
☐ Возвращает значение

▼ Аргументы

Имя	Тип

▼ Тело функции

```
review_counter_per_day += 1;
if (time() - time_of_last_reset_review_counter >= 8){
    reviews_per_day.add(review_counter_per_day);
    review_counter_per_day = 0;
    time_of_last_reset_review_counter = time();
}
```

Имя: unblock_holds_if_needed ☒ Отображать имя ☐ Исключить
Видимость: ☒ да
☒ Действие (не возвращает ничего)
☐ Возвращает значение

Аргументы

Имя	Тип

Тело функции

```
if (queue_short_haired_size + queue_fashion_haired_size + queue_coloured_size + waiting_after_colouring_size <= 19) {  
    if (hold_cashbox_one.isBlocked()){  
        hold_cashbox_one.unblock();  
    }  
    if (hold_cashbox_two.isBlocked()){  
        hold_cashbox_two.unblock();  
    }  
}
```

Имя: sink_unreviewed ☒ Отображать имя ☐ Исключить

Действия

При входе: `time_presence_in_system.add(60*(time() - agent.timeEnteringSystem));`

Специфические

Тип агента: ☒ Request

☒ Одиночный агент ☐ Популяция агентов

Модель/Библиотека: Библиотека моделирования процессов [\(Изменить...\)](#)

Видимость: ☒ да

☐ Отображается на верхнем уровне

☒ Записывать лог в базу данных
[Включить логирование выполнения модели](#)

[Показать презентацию](#)

Имя: sink_reviewed ☒ Отображать имя ☐ Исключить

Действия

При входе: `time_presence_in_system.add(60*(time() - agent.timeEnteringSystem));`

Специфические

Тип агента: ☒ Request

☒ Одиночный агент ☐ Популяция агентов

Модель/Библиотека: Библиотека моделирования процессов [\(Изменить...\)](#)

Видимость: ☒ да

☐ Отображается на верхнем уровне

☒ Записывать лог в базу данных
[Включить логирование выполнения модели](#)

[Показать презентацию](#)


Имя: short_hairing ☒ Отображать имя ☐ Исключить

Тип: ☒ Определенное время
☐ Пока не вызван метод stopDelay()

Время задержки: минуты

Вместимость:

Максимальная вместимость: ☐

Место агентов: 

▼ Специфические

Выталкивать агентов: ☐

Вернуть агента в исходную точку: ☒

Включить сбор статистики: ☐

▼ Действия


При входе:

При подходе к выходу:

При выходе:

При извлечении:

▼ Специфические

Тип агента:  Request

☒ Одиночный агент ☐ Популяция агентов

Модель/Библиотека: Библиотека моделирования процессов [\(Изменить...\)](#)

Видимость: ☒ да

☐ Отображается на верхнем уровне

☒ Записывать лог в базу данных
[Включить логирование выполнения модели](#)

► Описание

Имя: ☒ Отображать имя ☐ Исключить

Выход true выбирается: ☐ Заданной вероятностью
☒ При выполнении условия

Условие:

▼ Действия

При входе:

При выходе (true):

При выходе (false):

▼ Специфические

Тип агента: ▼

☒ Одиночный агент ☐ Популяция агентов

Модель/Библиотека: Библиотека моделирования процессов [\(Изменить...\)](#)

Видимость: ☒ да

☐ Отображается на верхнем уровне

☒ Записывать лог в базу данных
[Включить логирование выполнения модели](#)

► Описание

Имя: ☒ Отображать имя ☐ Исключить

Использовать: ☐ Вероятности
☒ Условия
☐ Номер выхода

Условие 1:

Условие 2:

Условие 3:

Условие 4:

▼ Действия

При входе:

При выходе 1:

При выходе 2:

При выходе 3:

При выходе 4:

При выходе 5:

▼ Специфические

Тип агента: ▼

☒ Одиночный агент ☐ Популяция агентов

Модель/Библиотека: Библиотека моделирования процессов [\(Изменить...\)](#)

Видимость: ☒ да

☐ Отображается на верхнем уровне

☒ Записывать лог в базу данных
[Включить логирование выполнения модели](#)

► Описание

Имя: review ☒ Отображать имя ☐ Исключить

Тип: ☒ Определенное время ☐ Пока не вызван метод stopDelay()

Время задержки: минуты

Вместимость:

Максимальная вместимость: ☐

Место агентов:

▼ Специфические

Выталкивать агентов: ☐

Вернуть агента в исходную точку: ☒

Включить сбор статистики: ☐

▼ Действия

При входе:

При подходе к выходу:

При выходе:

При извлечении:

▼ Специфические

Тип агента:

☒ Одиночный агент ☐ Популяция агентов

Модель/Библиотека: Библиотека моделирования процессов [\(Изменить...\)](#)

Видимости: ☒ да

☐ Отображается на верхнем уровне


☒ Записывать лог в базу данных [Включить логирование выполнения модели](#)

► Описание

Имя: queue_short_haired ☒ Отображать имя ☐ Исключить

Вместимость:

Максимальная вместимость:

Место агентов: 

▼ Специфические

Очередь:

Разрешить уход по таймауту: ☐

Разрешить вытеснение: ☐

Вернуть агента в исходную точку: ☒

Включить сбор статистики: ☐

▼ Действия

При входе:

При подходе к выходу:

При выходе:

При извлечении:

▼ Специфические

Тип агента:

☒ Одиночный агент ☐ Популяция агентов

Модель/Библиотека: Библиотека моделирования процессов [\(Изменить...\)](#)

Видимость: ☒ да


☐ Отображается на верхнем уровне

☒ Записывать лог в базу данных [Включить логирование выполнения модели](#)

► Описание

Имя: queue_fashion_haired ☒ Отображать имя ☐ Исключить


Вместимость:  100


Максимальная вместимость: 


Место агентов:  


▼ Специфические

Очередь:  FIFO


Разрешить уход по таймауту:  ☐

Разрешить вытеснение:  ☐

Вернуть агента в исходную точку:  ☒


Включить сбор статистики:  ☐

▼ Действия

При входе: 

```
queue_fashion_haired_size += 1;
queue_fashion_haired_input_intensity.add(1/((time() - queue_fashion_haired_last_agent_time)*60));
queue_fashion_haired_last_agent_time = time();
block_holds_if_needed();
agent.serviceStart = time();
agent.timePaymentStart = time();
add_value_to_congestion_waiting_hall();
```

При подходе к выходу: 

При выходе: 

```
queue_fashion_haired_size -= 1;
unblock_holds_if_needed();
fashion_haired_waiting_time.add(60*(time() - agent.serviceStart));
add_value_to_congestion_waiting_hall();
```

При извлечении: 

▼ Специфические


Тип агента:  Request

☒ Одиночный агент ☐ Популяция агентов

Модель/Библиотека: Библиотека моделирования процессов [\(Изменить...\)](#)

Видимость: ☒ да

☐ Отображается на верхнем уровне


 ☒ Записывать лог в базу данных
[Включить логирование выполнения модели](#)

► Описание

Имя: queue_coloured ☒ Отображать имя ☐ Исключить

Вместимость:

Максимальная вместимость:

Место агентов: 

▼ Специфические

Очередь:

Разрешить уход по таймауту: ☐

Разрешить вытеснение: ☐

Вернуть агента в исходную точку: ☒

Включить сбор статистики: ☐

▼ Действия

При входе:

При подходе к выходу:

При выходе:

При извлечении:

▼ Специфические

Тип агента:

☒ Одиночный агент ☐ Популяция агентов

Модель/Библиотека: Библиотека моделирования процессов [\(Изменить...\)](#)

Видимость: ☒ да

☐ Отображается на верхнем уровне

☒ Записывать лог в базу данных [Включить логирование выполнения модели](#)

► Описание

Имя: queue_cashbox_two ☒ Отображать имя ☐ Исключить

Вместимость:

Максимальная вместимость:

Место агентов: 

▼ Специфические

Очередь:

Приоритет агента:

Разрешить уход по таймауту: ☒

Таймаут:

Разрешить вытеснение: ☐

Вернуть агента в исходную точку: ☒

Включить сбор статистики: ☐

▼ Действия

При входе:

```
queue_cashbox_two_size += 1;
queue_cashbox_two_input_intensity.add(1/((time() - queue_cashbox_two_last_agent_time)*60));
queue_cashbox_two_last_agent_time = time();
agent.serviceStart = time();
```

При подходе к выходу:

При выходе:

```
queue_cashbox_two_size -= 1;
cashbox_two_patients_counter += 1;
cashbox_two_waiting_time.add(60*(time() - agent.serviceStart));
```

При уходе по таймауту:

```
cashbox_two_impatients_counter += 1;
queue_cashbox_two_size -= 1;
```

При извлечении:

▼ Специфические

Тип агента: ☒ Request ☐

☒ Одиночный агент ☐ Популяция агентов

Модель/Библиотека: Библиотека моделирования процессов [\(Изменить...\)](#)

Видимость: ☒ да

☐ Отображается на верхнем уровне


☒ Записывать лог в базу данных
[Включить логирование выполнения модели](#)

► Описание

Имя: queue_cashbox_one ☒ Отображать имя ☐ Исключить

Вместимость:

Максимальная вместимость:

Место агентов: 

▼ Специфические

Очередь:

Приоритет агента:

Разрешить уход по таймауту: ☒

Таймаут:

Разрешить вытеснение: ☐

Вернуть агента в исходную точку: ☒

Включить сбор статистики: ☐

▼ Действия

При входе:

При подходе к выходу:

При выходе:

При уходе по таймауту:

При извлечении:

▼ Специфические

Тип агента:

☒ Одиночный агент ☐ Популяция агентов

Модель/Библиотека: Библиотека моделирования процессов [\(Изменить...\)](#)

Видимость: ☒ да

☐ Отображается на верхнем уровне

☒ Записывать лог в базу данных [Включить логирование выполнения модели](#)

► Описание

Имя: leavingQueueTimeDistributi ☒ Отображать имя ☐ Исключить

Видимость: ☒ да

Тип: ☒ Непрерывное
☐ Дискретное
☐ Значение

Режим задания: ☒ Интервалы
☐ Частотная таблица
☐ Набор наблюдений

▼ Данные

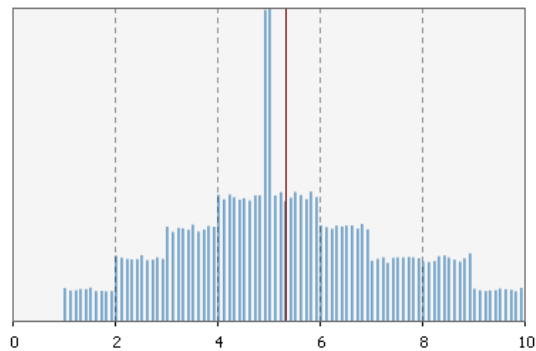
☐ Загружается из базы данных

Начало интервала	Конец интервала	Число наблюдений
1.0	2.0	5.0
2.0	3.0	10.0
3.0	4.0	15.0
4.0	4.9	20.0
4.9	5.1	50.0
5.1	6.0	20.0
6.0	7.0	15.0
7.0	9.0	10.0
9.0	10.0	5.0

+

×

▼ Предв. просмотр



▼ Специфические

☐ Статический

► Описание

Имя: ☒ Отображать имя ☐ Исключить

Выход true выбирается: ☐ Заданной вероятностью
☒ При выполнении условия

Условие:

▼ Действия

При входе:

При выходе (true):

При выходе (false):

▼ Специфические

Тип агента: ☐ Request

☒ Одиночный агент ☐ Популяция агентов

Модель/Библиотека: Библиотека моделирования процессов ([Изменить...](#))

Видимость: ☒ да

☐ Отображается на верхнем уровне

☒ Записывать лог в базу данных
[Включить логирование выполнения модели](#)

► Описание

Имя: ☒ Отображать имя ☐ Исключить

Выход true выбирается: ☒ Заданной вероятностью
☐ При выполнении условия

Вероятность:

▼ Действия

При входе:

При выходе (true):

При выходе (false):

▼ Специфические

Тип агента: ☐ Request

☒ Одиночный агент ☐ Популяция агентов

Модель/Библиотека: Библиотека моделирования процессов ([Изменить...](#))

Видимость: ☒ да

☐ Отображается на верхнем уровне

☒ Записывать лог в базу данных
[Включить логирование выполнения модели](#)

► Описание

Имя:
☒ Отображать имя
☐ Исключить

Действия

При входе:

Специфические

Тип агента:

☒ Одиночный агент
☐ Популяция агентов

Модель/Библиотека: Библиотека моделирования процессов [\(Изменить...\)](#)

Видимость: ☒ да

☐ Отображается на верхнем уровне

☒ Записывать лог в базу данных
[Включить логирование выполнения модели](#)

Описание

Имя:
☒ Отображать имя
☐ Исключить

Режим:

Изначально заблокирован: ☐

Действия

При входе:

Специфические

Тип агента:

☒ Одиночный агент
☐ Популяция агентов

Модель/Библиотека: Библиотека моделирования процессов [\(Изменить...\)](#)

Видимость: ☒ да

☐ Отображается на верхнем уровне

☒ Записывать лог в базу данных
[Включить логирование выполнения модели](#)

Имя:
☒ Отображать имя
☐ Исключить

Режим:

Изначально заблокирован: ☐

Действия

При входе:

Специфические

Тип агента:

☒ Одиночный агент
☐ Популяция агентов

Модель/Библиотека: Библиотека моделирования процессов [\(Изменить...\)](#)

Видимость: ☒ да

☐ Отображается на верхнем уровне

☒ Записывать лог в базу данных
[Включить логирование выполнения модели](#)

Имя: ☒ Отображать имя ☐ Исключить

Видимость: ☒ да

☐ Действие (не возвращает ничего)

☒ Возвращает значение

Тип:

Аргументы

Имя	Тип
alpha	double
beta	double
min	double
max	double

Тело функции

```
double result = 0; do { result = gamma(alpha, beta, min); } while (result > max); return result;
```

Имя: ☒ Отображать имя ☐ Исключить

Тип: ☒ Определенное время
☐ Пока не вызван метод stopDelay()

Время задержки:

Вместимость:

Максимальная вместимость:

Место агентов:

Специфические

Выталкивать агентов: ☐

Вернуть агента в исходную точку: ☒

Включить сбор статистики: ☐

Действия

При входе:

При подходе к выходу:

При выходе:

При извлечении:

Специфические

Тип агента:

☒ Одиночный агент ☐ Популяция агентов

Модель/Библиотека: Библиотека моделирования процессов [\(Изменить...\)](#)

Видимость: ☒ да

☐ Отображается на верхнем уровне

☒ Записывать лог в базу данных
[Включить логирование выполнения модели](#)

Описание

Имя: ☒ Отображать имя ☐ Исключить

Прибывают согласно:

Время между прибытиями:

Считать параметры агентов из БД: ☐

За 1 раз создается несколько агентов: ☐

Ограниченное кол-во прибытий: ☐

Новый агент:

Местоположение прибытия:

▼ Специфические

Установить время начала: ☐

Добавить агентов в: ☒ Популяцию по умолчанию ☐ Другую популяцию агентов

Выталкивать агентов: ☒

▼ Действия

До прибытия:

При подходе к выходу:

```
double rnd_prior = Math.random();
double rnd_type = Math.random();
if (rnd_prior < 0.28) {agent.priority = 1;}
if (rnd_type >= part_short_hairing + part_fashion_hairing) {agent.type = 3;}
else if (rnd_type >= part_short_hairing) {agent.type = 2;}
```

При выходе:

```
if (entrance.count() > max_number_of_clients) {finishSimulation();}
if ((entrance.count() % 4900 == 0) && (entrance.count() != 0)) {
    //write_results(column_name_index,4,"fashion_scenario");
    column_name_index += 1;
}
agent.timeEnteringSystem = time();
```

▼ Специфические

Тип агента:

☒ Одиночный агент ☐ Популяция агентов

Модель/Библиотека: Библиотека моделирования процессов [\(Изменить...\)](#)

Видимость: ☒ да

☐ Отображается на верхнем уровне

☒ Записывать лог в базу данных [Включить логирование выполнения модели](#)

► Описание

Имя: colouring ☒ Отображать имя ☐ Исключить

Тип: ☒ Определенное время
☐ Пока не вызван метод stopDelay()

Время задержки: минуты

Вместимость:

Максимальная вместимость:

Место агентов: 

▼ Специфические

Выталкивать агентов: ☐

Вернуть агента в исходную точку: ☒

Включить сбор статистики: ☐

▼ Действия

При входе:

При подходе к выходу:

При выходе:

При извлечении:

▼ Специфические

Тип агента:

☒ Одиночный агент ☐ Популяция агентов

Модель/Библиотека: Библиотека моделирования процессов [\(Изменить...\)](#)

Видимость: ☒ да

☐ Отображается на верхнем уровне

☒ Записывать лог в базу данных

[Включить логирование выполнения модели](#)

► Описание

Имя: cashbox_two ☒ Отображать имя ☐ Исключить

Тип: ☒ Определенное время
☐ Пока не вызван метод stopDelay()

Время задержки: минуты

Вместимость:

Максимальная вместимость:

Место агентов: 

▼ Специфические

Вытаскивать агентов: ☐

Вернуть агента в исходную точку: ☒

Включить сбор статистики: ☐

▼ Действия

При входе:

При подходе к выходу:

При выходе:

При извлечении:

▼ Специфические

Тип агента:

☒ Одиночный агент ☐ Популяция агентов

Модель/Библиотека: Библиотека моделирования процессов [\(Изменить...\)](#)

Видимость: ☒ да

☐ Отображается на верхнем уровне

☒ Записывать лог в базу данных

[Включить логирование выполнения модели](#)

► Описание

Имя: cashbox_one ☒ Отображать имя ☐ Исключить

Тип: ☒ Определенное время
☐ Пока не вызван метод stopDelay()

Время задержки: минуты

Вместимость:

Максимальная вместимость:

Место агентов:

▼ Специфические

Выталкивать агентов: ☐

Вернуть агента в исходную точку: ☒

Включить сбор статистики: ☐

▼ Действия

При входе:

При подходе к выходу:

При выходе:

При извлечении:

▼ Специфические

Тип агента:

☒ Одиночный агент ☐ Популяция агентов

Модель/Библиотека: Библиотека моделирования процессов [\(Изменить...\)](#)

Видимость: ☒ да

☐ Отображается на верхнем уровне

☒ Записывать лог в базу данных
[Включить логирование выполнения модели](#)

► Описание

Имя: block_holds_if_needed ☒ Отображать имя ☐ Исключить

Видимость: ☒ да

☒ Действие (не возвращает ничего)
☐ Возвращает значение

▼ Аргументы

Имя	Тип

▼ Тело функции

```

if (queue_short_haired_size + queue_fashion_haired_size + queue_coloured_size + waiting_after_colouring_size > 19){
    if (hold_cashbox_one.isBlocked() == false){
        hold_cashbox_one.block();
    }
    if (hold_cashbox_two.isBlocked() == false){
        hold_cashbox_two.block();
    }
}

```







Имя: add_value_to_congestion_w ☒ Отображать имя ☐ Исключить

Видимость: ☒ да

☒ Действие (не возвращает ничего)

☐ Возвращает значение

▼ Аргументы

Имя	Тип
<div>     </div>	

▼ Тело функции

```
congestion_waiting_hall.add(waiting_after_colouring_size + queue_short_haired_size + queue_fashion_haired_size + queue_coloured_size);
```

▼ Специфические

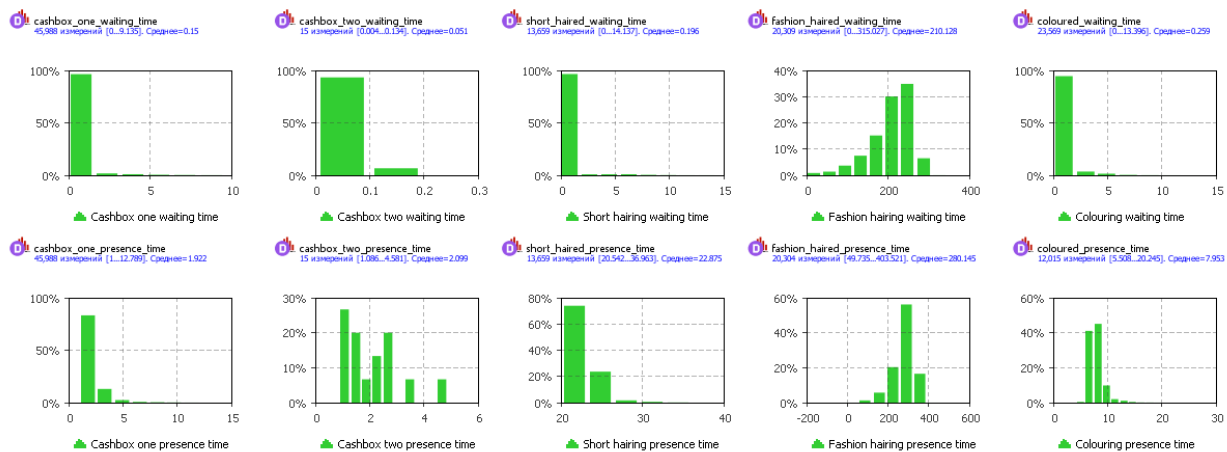
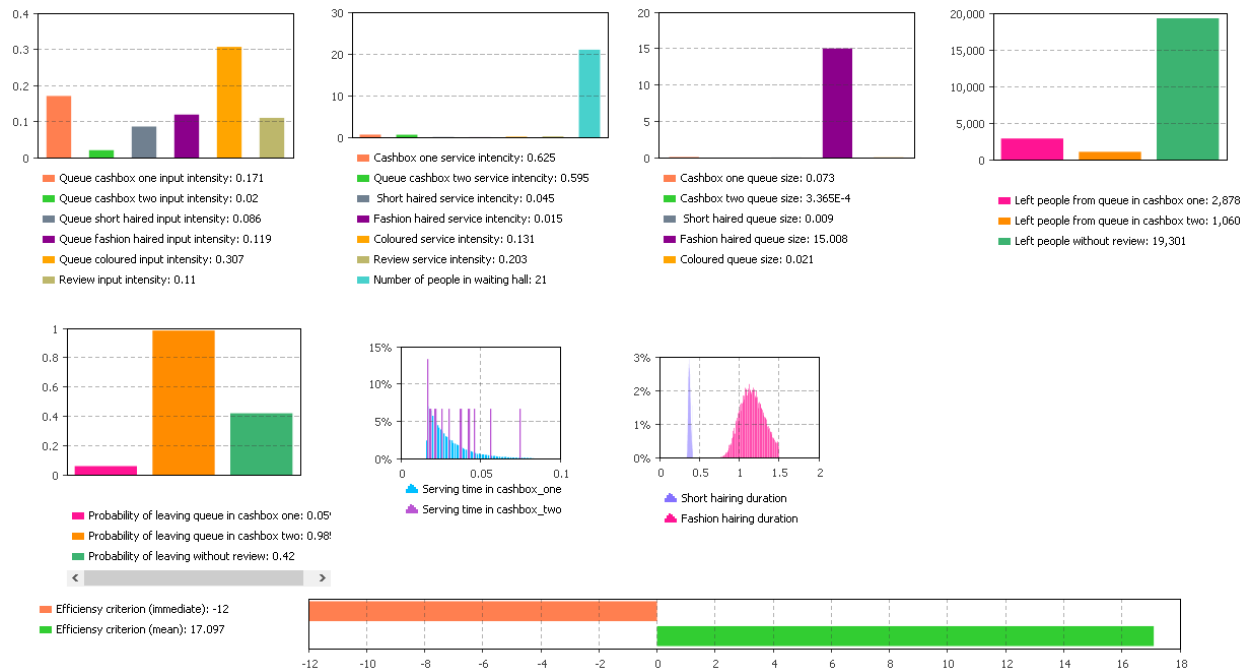
☐ Статическая

Уровень доступа: по умолчанию ▼

☐ Единицы измерения (сист. динамика):

Сценарии работы модели

Обычный режим работы

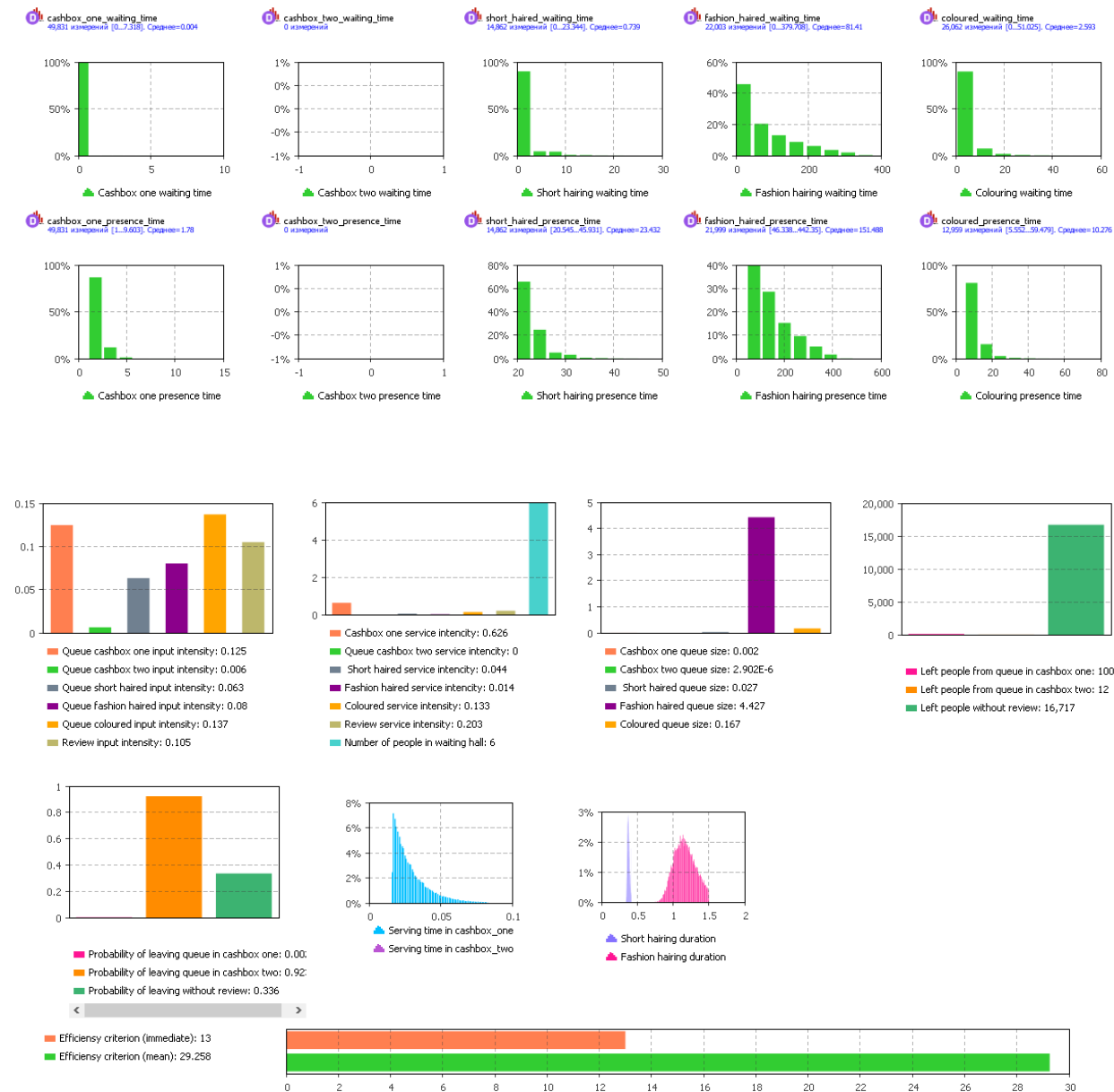


Результаты подсчетов ширины доверительного интервала характеристик модели (в процентах)

Number of bids	4900	9800	14700	19600	24500	29400	34300	39200	44100	49000
Probability of leaving queue in cashbox one	29,594595	17,937086	10,065245	7,0951679	7,187276	5,6205774	4,2107948	4,2400325	4,1849998	4,1407306
Probability of leaving queue in cashbox two	0,7115334	0,5061969	0,5060607	0,5058407	0,3038132	0,4048943	0,40501	0,4047811	0,3035642	0,3034457
Probability of leaving without review	2,4184887	1,4483964	1,2085915	0,9638535	0,7222212	0,4820579	0,7235812	0,4823827	0,7241306	0,4829101
Utilization cashbox one	2,2435352	0,3672295	0,3670365	0,3668231	0,366907	0,367133	0,3670639	0,3671557	0,3673266	0,3674011
Utilization cashbox two	35,445828	41,054898	36,242916	24,26509	18,422092	20,889556	14,584246	14,589164	11,549286	11,327814
Utilization short haired	1,9475989	1,3947447	2,1755079	1,6656006	1,8080914	1,8547359	1,6236585	1,9916721	1,8053558	1,7140498
Utilization fashion haired	1,1672044	1,3380182	1,3868321	1,1695599	0,5464884	0,2791284	0,2923514	0,3178246	0,381485	0,2800384
Utilization coloured	48,129015	165,97593	140,69338	119,77566	112,06577	102,83761	97,852703	92,901597	88,724347	77,20026
Utilization review	1,2591285	0,5383485	0,5386756	0,5395727	0,3595865	0,5401254	0,3601061	0,3599294	0,3601002	0,360247
Cashbox one queue size	13,163109	15,745144	8,1790704	5,7607712	5,8311641	5,6672212	4,5314507	4,5570493	4,4999193	3,3439462
Cashbox two queue size	259,06816	262,40514	256,82948	253,30905	255,61345	247,32505	245,45843	245,69438	242,12089	238,97133
Short haired queue size	34,686825	10,868277	20,589483	10,408475	10,224034	10,436122	10,239781	10,227633	10,30235	10,301901
Fashion haired queue size	6,9402785	2,5709111	1,7244868	0,9006046	1,6606285	1,7688712	1,632081	1,4488487	1,3989409	1,3006051
Coloured queue size	17,870397	5,2095111	5,0195725	5,1221525	10,278146	5,124914	5,0765869	5,0624732	5,0843801	10,096583
Min	0,7115334	0,3672295	0,3670365	0,3668231	0,3038132	0,2791284	0,2923514	0,3178246	0,3035642	0,2800384
Average	32,474693	37,668559	34,680452	30,846301	30,384976	28,828428	27,668417	27,33178	26,557648	25,727947
Max	259,06816	262,40514	256,82948	253,30905	255,61345	247,32505	245,45843	245,69438	242,12089	238,97133

Режим работы во время эпидемии гриппа

3 парикмахера заболели, по одному из каждого зала, интервал между посетителями увеличился в 1.5 раза.

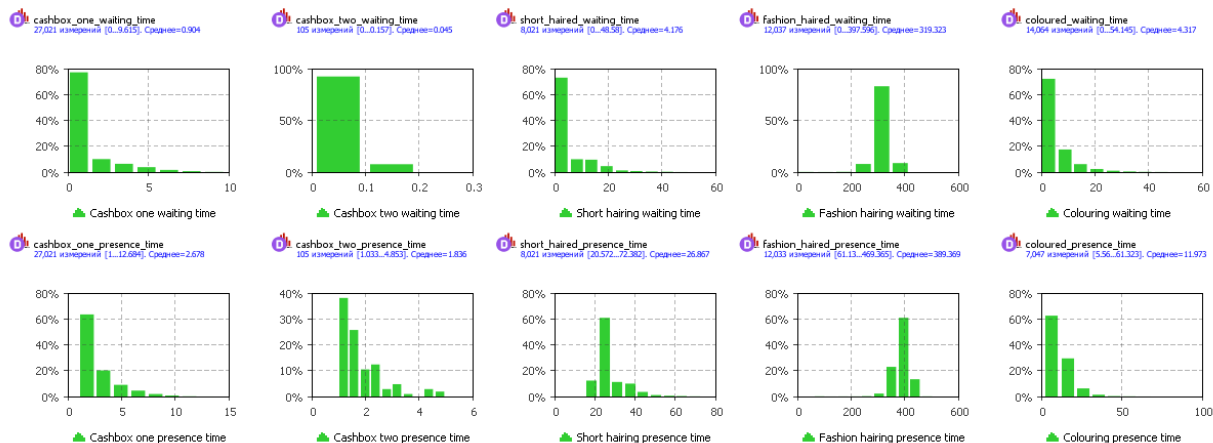
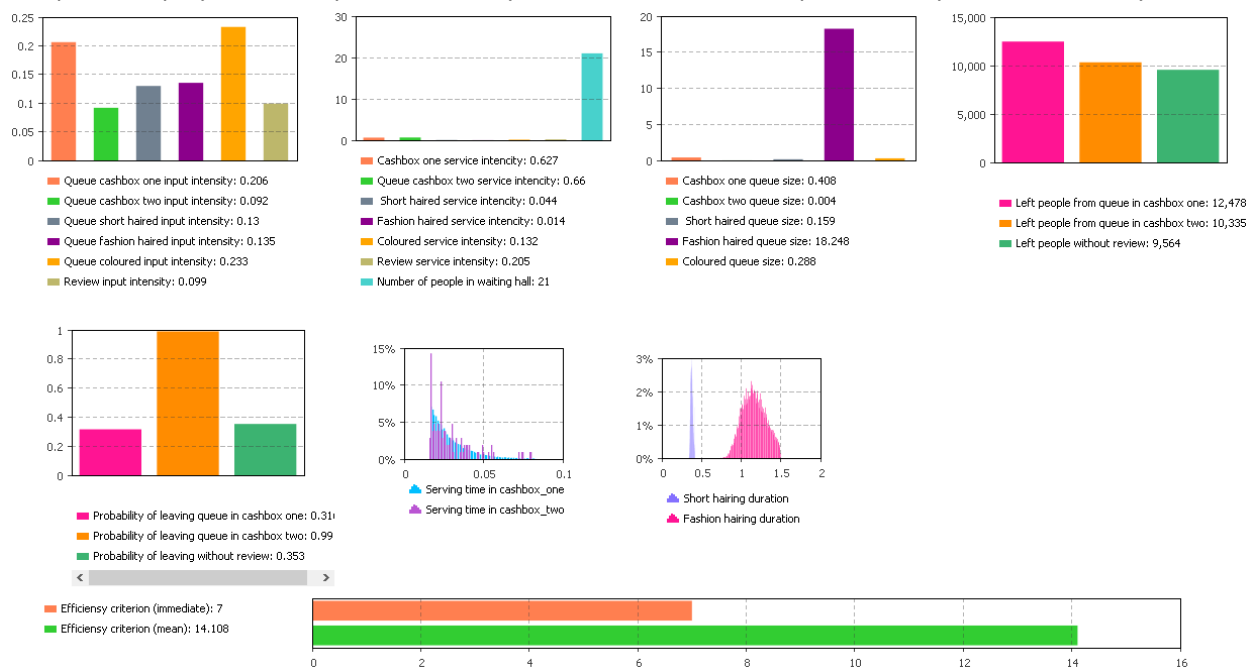


Результаты подсчетов ширины доверительного интервала характеристик модели (в процентах)

Number of bids	4900	9800	14700	19600	24500	29400	34300	39200	44100	49000
Probability of leaving queue in cashbox one	161,07899	104,24845	94,618116	94,213709	46,742284	46,141584	24,285854	24,161218	22,961794	20,529545
Probability of leaving queue in cashbox two	158,84615	153,40909	28,428928	30,294118	17,466235	13,881356	11,342536	6,1850876	4,8305306	3,9729801
Probability of leaving without review	0,8944524	0,5936732	0,8895087	0,592283	0,5940633	0,5948311	0,5941105	0,5939671	0,2972185	0,2970327
Utilization cashbox one	5,081487	0,5181132	0,517603	0,5177991	0,5176285	0,5175929	0,5176372	0,517451	0,517317	0,5175427
Utilization cashbox two	0	0	0	0	0	0	320,87954	311,31283	274,1276	277,07466
Utilization short haired	2,4338266	1,2869327	0,6761816	0,407255	0,7477237	0,6792756	0,6782304	0,7466747	0,6108305	0,6113509
Utilization fashion haired	1,1334453	0,213078	0,2305452	0,1952283	0,194975	0,3367528	0,3371573	0,3014266	0,1950163	0,2302827
Utilization coloured	26,419952	6,6591531	21,582414	17,110007	13,56296	35,344769	33,122659	29,748975	27,740864	27,197601
Utilization review	0,3972022	0,3968767	0,3969923	0,1984563	0,3969154	0,3968502	0,3968782	0,3968548	0,1983864	0,1983961
Cashbox one queue size	150,27825	129,39774	117,33033	77,345539	55,123983	53,411708	27,935658	27,912781	26,71094	23,665011
Cashbox two queue size	21161,949	24916,76	38236,904	44920,043	29150,425	29639,484	32917,214	29351,236	27518,31	24879,754
Short haired queue size	14,137975	10,558211	6,9756322	7,1781781	10,803632	7,1480063	7,0974686	7,1828346	3,6073872	7,1948518
Fashion haired queue size	22,7859	23,487844	17,580626	16,339057	10,654303	7,1440549	7,4536714	7,6082731	9,534356	8,9888085
Coloured queue size	24,37273	18,356674	14,991744	10,505842	9,4462349	7,1316153	6,3929989	6,3727057	6,3449929	6,3865204
Min	0	0	0	0	0	0	0,3371573	0,3014266	0,1950163	0,1983961
Average	1552,1292	1811,849	2752,9373	3226,7815	2094,0483	2129,4438	2382,7321	2126,734	1992,5705	1804,0442
Max	21161,949	24916,76	38236,904	44920,043	29150,425	29639,484	32917,214	29351,236	27518,31	24879,754

Режим работы в разгар лета

3 парикмахера ушли в отпуск, по одному из каждого зала, интервал между посетителями уменьшился на 30%.

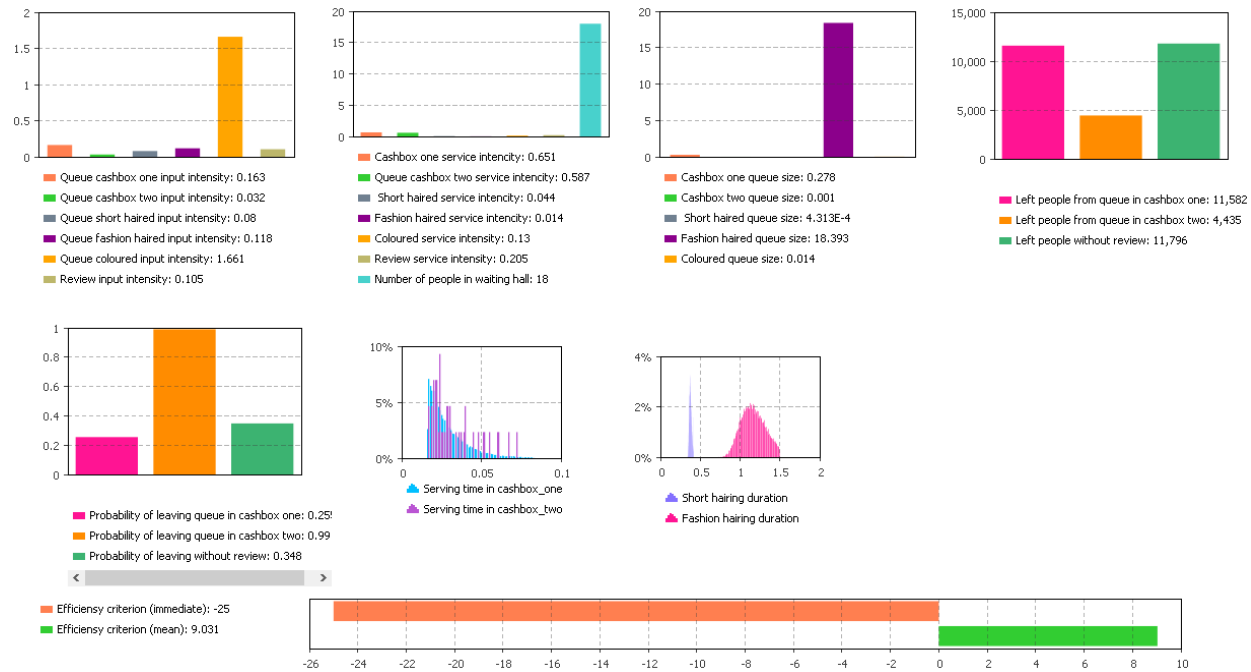


Результаты подсчетов ширины доверительного интервала характеристик модели (в процентах)

Number of bids	4900	9800	14700	19600	24500	29400	34300	39200	44100	49000
Probability of leaving queue in cashbox one	1,8864704	4,1305401	2,5232579	2,5126062	2,1983096	1,8737246	1,8659591	1,867282	1,8707286	1,8702367
Probability of leaving queue in cashbox two	0,4042905	0,202125	0,1009921	0,2019478	0,2018397	0,2018239	0,1009163	0,2018521	0,2018723	0,2019036
Probability of leaving without review	2,5929067	1,996754	0,8535794	1,4213399	0,8514284	0,8505176	0,8529821	0,852553	0,8511663	0,5674593
Utilization cashbox one	4,2418762	0,9259293	0,6171101	0,6165628	0,6163726	0,6168376	0,3084801	0,3084302	0,3083776	0,3085004
Utilization cashbox two	20,464358	7,3666519	7,3811384	7,4234108	5,9932324	5,9451803	6,0238834	5,2546811	4,49069	5,200418
Utilization short haired	3,6451118	4,9952301	8,1879478	7,1452118	6,3842461	10,636065	9,0189414	22,538327	19,824396	17,872167
Utilization fashion haired	9,9661488	6,6972457	4,2735314	3,4695627	3,1652653	2,2749543	4,4486658	3,8929688	3,4674363	2,9200645
Utilization coloured	31,522567	26,577341	19,786775	16,599841	37,215809	31,464515	29,13906	26,25257	25,555676	21,231808
Utilization review	1,3630351	1,1690015	0,7813631	0,7841988	0,587771	0,5887159	0,5891063	0,7855525	0,7852725	0,5890698
Cashbox one queue size	1,7265872	2,9597182	1,965462	1,7128576	1,2228954	1,217697	1,2152285	1,2139948	1,2140873	0,9713771
Cashbox two queue size	22,900437	22,861148	22,706977	22,591202	22,668124	22,529534	22,529977	22,52422	22,537854	22,527551
Short haired queue size	18,901773	19,404244	9,0349797	11,069647	9,7549316	10,509111	10,154417	10,086444	8,718987	8,1107038
Fashion haired queue size	0,6442472	0,7409817	0,519867	0,4368387	0,3167261	0,2618603	0,1797794	0,2233848	0,2070642	0,1580256
Coloured queue size	26,028617	22,325874	12,498279	10,718636	9,0022033	6,9908428	4,2280663	6,299882	6,6445509	4,5510299
Min	0,4042905	0,202125	0,1009921	0,2019478	0,2018397	0,2018239	0,1009163	0,2018521	0,2018723	0,1580256
Average	10,449173	8,7394846	6,5165186	6,1931331	7,1556539	6,8543842	6,4753903	7,3072959	6,9055827	6,2200224
Max	31,522567	26,577341	22,706977	22,591202	37,215809	31,464515	29,13906	26,25257	25,555676	22,527551

Режим работы в случае образования повышенного количества модников

Доля заявок на стрижку под одну насадку снижена с 0.30 до 0.15; доля заявок, требующих модельную стрижку увеличена с 0.45 до 0.60.



Результаты подсчетов ширины доверительного интервала характеристик модели (в процентах)

Number of bids	4900	9800	14700	19600	24500	29400	34300	39200	44100	49000
Probability of leaving queue in cashbox one	2,8620116	2,3528661	2,7130788	2,3253137	3,0961322	2,7062239	2,3236803	1,9406862	1,9440285	1,555297
Probability of leaving queue in cashbox two	0,911425	0,5052662	0,4038044	0,3029582	0,2019739	0,1009827	0,1009508	0,1009437	0,2018771	0,2018997
Probability of leaving without review	1,4335866	0,576048	1,1530261	0,8630623	1,1528308	0,5767699	0,8635326	0,8631531	0,8627949	0,8629071
Utilization cashbox one	4,9401862	0,7649358	0,7654684	0,7647952	0,7645128	0,3820616	0,7643429	0,3820521	0,3820902	0,3821883
Utilization cashbox two	16,417084	10,036321	14,089339	12,152107	10,365818	12,592489	10,422919	10,25036	10,236695	10,131342
Utilization short haired	9,5922059	4,4432199	2,6031273	2,4337369	2,5001457	1,9944212	2,7847578	3,6406726	3,7193565	3,7293368
Utilization fashion haired	9,7815998	3,6934209	2,7831257	2,3979435	1,9347965	1,8533669	2,1681167	1,6926335	2,0257514	1,743427
Utilization coloured	55,325075	33,147944	25,564407	16,37307	14,539638	13,61823	12,642252	21,19362	18,210127	16,413045
Utilization review	0,6070931	1,0174677	1,0198326	0,8162101	0,6120341	0,6117262	0,6118281	0,4076936	0,4076961	0,2037337
Cashbox one queue size	2,996498	1,8062847	2,1446239	2,50107	2,852067	2,4945751	2,4976079	2,1449928	1,7895655	1,0733678
Cashbox two queue size	73,176794	71,031574	70,552615	70,622797	70,824553	71,043964	71,485699	71,70741	71,590436	71,474309
Short haired queue size	263,62503	243,37572	247,23649	246,39566	216,85018	201,1336	209,40647	199,53883	198,77203	203,28512
Fashion haired queue size	0,4557328	0,2996613	0,2118894	0,2171711	0,2387025	0,2494572	0,2440654	0,2006801	0,1790101	0,1790022
Coloured queue size	27,656642	22,753025	22,741608	22,894121	15,006589	14,727842	14,509739	14,408697	14,33614	14,325192
Min	0,4557328	0,2996613	0,2118894	0,2171711	0,2019739	0,1009827	0,1009508	0,1009437	0,1790101	0,1790022
Average	33,555783	28,271697	28,141602	27,218573	24,352855	23,148979	23,630426	23,462316	23,189828	23,254298
Max	263,62503	243,37572	247,23649	246,39566	216,85018	201,1336	209,40647	199,53883	198,77203	203,28512

Некоторые обобщения

Так, в соответствии с критерием эффективности, система показала себя наилучшим образом в период эпидемии – это косвенно указывает на необходимость рассмотрения увольнения одного или нескольких мастеров.

Что касается доверительного интервала – ни в одном случае его величина по среднему значению, а также по максимуму недостаточно мала, вероятно, это связано с недостаточным количеством пропущенных заявок, дальнейшее увеличение которого невозможно в связи с ограничением Anylogic используемой версии (максимум 50 000 сгенерированных заявок).

Тем не менее, при максимальном измеренном числе заявок ширина доверительного интервала минимальна в случае моделирования системы в летний период, максимальна – в период эпидемии, что недвусмысленно указывает на то, что ширина доверительного интервала находится в обратной зависимости от интервала между клиентами.

Разработка имитационной модели SimPy

Дополнительные замечания

- 1) Поскольку генераторы библиотеки `scipy` при передаче тех же параметров, что и в среде Anylogic выдают очень отличающиеся распределения от используемых на предыдущем этапе, параметры были изменены в сторону получения близких по характеристикам распределений.
- 2) Добавлена задержка при открытии входа для новых посетителей для учета того, что это открытие происходит не моментально, а также для обеспечения достоверности того факта, что зал ожидания успеет разгрузиться перед новым открытием
- 3) Эмпирическое распределение, используемое на предыдущем этапе, было аппроксимировано с помощью гамма-функции.
- 4) Расчет доверительного интервала критерия эффективности при запуске в систему N заявок происходит следующим образом:
 - a. Система запускается 5 раз с количеством заявок, равным N , сохраняются полученные значения коэффициентов эффективности
 - b. С использованием библиотек `scipy` и `numpy` рассчитывается математическое ожидание и величина доверительного интервала при помощи T -распределения Стьюдента
 - c. Полученное математическое ожидание усредняется с учетом предыдущих запусков системы, в которых количество заявок было соответственно равно $N-K$, $N-2K$, $N-3K$, $N-4K$, где K - некоторый шаг, с которым производится поиск оптимального числа заявок. В результате данного этапа получаем усредненное математическое ожидание коэффициента эффективности, а также ширину доверительного интервала относительно результатов запуска системы с N , $N-K$, $N-2K$, $N-3K$, $N-4K$, заявками (расчет производится так же, как и на этапе a)
 - d. Результирующая ширина доверительного интервала в процентах получается суммированием ширины доверительного интервала, полученной при пятикратном запуске системы с количеством заявок, равным N с шириной доверительного интервала, полученной при запуске системы с количеством заявок N , $N-K$, $N-2K$, $N-3K$, $N-4K$, и делением полученной суммы на математическое ожидание коэффициента эффективности, вычисленное при усреднении значений, полученных при запуске системы с количеством заявок, равным N , $N-K$, $N-2K$, $N-3K$, $N-4K$.

Исходный код

Файл *barbershop.py*

```
1. import numpy
2. import generators
3.
4. import constants
5. import statistics
6. import entities
7. from scipy import stats
8. import math
9. import winsound
10.
11. waiting_hall_fill = 0
12.
13. blocked = False
14.
15. #rqs = [0,0]
16.
17. def get_services(customer_class):
18.     services = []
19.     if (customer_class == 1):
20.         services.append((entities.short_hairing_hall, " short hairdressing ", generators.get_service_short_hairing_interval))
21.     elif (customer_class == 2):
22.         services.append((entities.fashion_hairing_hall, " fashion hairdressing ", generators.get_service_fashion_hairing_interval))
```

```

23.     else:
24.         services.append((entities.colouring_hall, " colouring ", generators.get_service_colouring_interval))
25.         services.append((entities.waiting_after_colouring, " waiting after colouring ", generators.get_waiting_after_colouring_interval))
26.         services.append((entities.colouring_hall, " drying ", generators.get_service_colouring_interval))
27.     return services
28.
29. def get_cashbox():
30.     if (statistics.get_queue_length(entities.cashbox_two) < statistics.get_queue_length(entities.cashbox_one)):
31.         return entities.cashbox_two
32.     return entities.cashbox_one
33.
34. def source(env, quantity):
35.     global waiting_hall_fill
36.     global blocked
37.     global rqs
38.     for i in range(quantity):
39.         c = customer(env,
40.                     'Customer%02d' % i,
41.                     get_cashbox(),
42.                     get_services(generators.get_class_id()),
43.                     entities.review_desk,
44.                     generators.get_random_priority())
45.
46.     env.process(c)

```

```

47.         yield env.timeout(generators.get_interval_before_new_customer_summer())
48.
49.
50.
51. def switch_blocked_state_if_necessary():
52.     global blocked
53.     global waiting_hall_fill
54.     if (waiting_hall_fill >= constants.waiting_hall_max_fullness) and (not blocked):
55.         env.process(blocker(entities.cashbox_one, entities.unblock_event))
56.         env.process(blocker(entities.cashbox_two, entities.unblock_event))
57.     elif (waiting_hall_fill < constants.waiting_hall_max_fullness) and blocked:
58.         entities.unblock_event.succeed()
59.         entities.unblock_event = entities.env.event()
60.
61. def blocker(resource, unblock_event):
62.     global blocked
63.     with resource.request(priority = constants.staff_priority_id) as req:
64.         yield req
65.         yield env.timeout(constants.time_of_switching_entrance)
66.         blocked = True
67.         yield (entities.env.timeout(constants.max_blocking_interval) | unblock_event)
68.         yield env.timeout(constants.time_of_switching_entrance)
69.         blocked = False
70.

```



```
71. def try_print(message):
72.     if constants.verbous:
73.         print(message)
74.
75. def increase_waiting_hall_fullness():
76.     global waiting_hall_fill
77.     statistics.waiting_hall_fills.append(waiting_hall_fill)
78.     waiting_hall_fill += 1
79.
80. def decrease_waiting_hall_fullness():
81.     global waiting_hall_fill
82.     statistics.waiting_hall_fills.append(waiting_hall_fill)
83.     waiting_hall_fill -= 1
84.
85. def fix_entering_queue(resource, is_it_waiting_hall):
86.     statistics.increase_queue_length(resource)
87.     statistics.append_queue_length(resource)
88.     if (is_it_waiting_hall):
89.         increase_waiting_hall_fullness()
90.         switch_blocked_state_if_necessary()
91.
92. def fix_leaving_queue(resource, is_it_waiting_hall):
93.     statistics.decrease_queue_length(resource)
94.     statistics.append_queue_length(resource)
```

```

95.     if (is_it_waiting_hall):
96.         decrease_waiting_hall_fullness()
97.         switch_blocked_state_if_necessary()
98.
99. def update_reviews_per_day():
100.     statistics.reviews_per_day += 1
101.     if entities.env.now - statistics.last_time_writing_reviews >= statistics.day_length:
102.         statistics.reviews_per_day_set.append(statistics.reviews_per_day)
103.         statistics.reviews_per_day = 0
104.         statistics.last_time_writing_reviews = entities.env.now
105.
106. def fix_arriving(resource):
107.     last_seen_input_time = statistics.get_last_seen_input_time(resource)
108.     if last_seen_input_time > 0:
109.         statistics.append_intensity_component(resource, 1/(env.now - last_seen_input_time))
110.     statistics.set_last_seen_input_time(resource, env.now)
111.
112. def fix_stop_serving(resource, start_serving_time):
113.     statistics.append_service_intensity_component(resource, 1/(entities.env.now - start_serving_time))
114.
115. def customer(env, name, cashbox, services, review_desk, customer_priority):
116.     if constants.statistics_enable:
117.         fix_arriving(cashbox)
118.     try_print('%7.4f %s arrived' % (env.now, name))

```

```

119. arriving_timestamp = env.now
120. starting_serving_timestamp = env.now
121. with cashbox.request(priority = customer_priority) as req:
122.     fix_entering_queue(cashbox, False)
123.     results = yield req | env.timeout(generators.get_waiting_interval())
124.     fix_leaving_queue(cashbox, False)
125.     if req in results:
126.         if constants.statistics_enable:
127.             statistics.append_waiting_time(cashbox, env.now - arriving_timestamp)
128.             handling_started = env.now
129.             yield env.timeout(generators.get_service_cashbox_interval())
130.
131.             try_print('%7.4f %s served in cashbox' % (env.now, name))
132.             if constants.statistics_enable:
133.                 statistics.append_presence_time(cashbox, env.now - arriving_timestamp)
134.                 fix_stop_serving(cashbox, handling_started)
135.         else:
136.             try_print('%7.4f %s left without serving' % (env.now, name))
137.             statistics.increase_lost_quantity()
138.         return
139.
140. for service in services:
141.     if constants.statistics_enable:
142.         fix_arriving(service[0])

```

```

143.     arriving_timestamp = env.now
144.     try_print('%7.4f %s arrived at %s queue' % (arriving_timestamp, name, service[1]))
145.     fix_entering_queue(service[0], True)
146.     with service[0].request() as req:
147.         results = yield req
148.         if constants.statistics_enable:
149.             statistics.append_waiting_time(service[0], env.now - arriving_timestamp)
150.             handling_started = env.now
151.             fix_leaving_queue(service[0], True)
152.             yield env.timeout(service[2]())
153.             if constants.statistics_enable:
154.                 statistics.append_presence_time(service[0], env.now - arriving_timestamp)
155.                 fix_stop_serving(service[0], handling_started)
156.                 try_print('%7.4f %s got %s' % (env.now, name, service[1]))
157.
158.     with review_desk.request() as req:
159.         if constants.statistics_enable:
160.             fix_arriving(review_desk)
161.             arriving_timestamp = env.now
162.             results = yield req | env.timeout(0)
163.
164.     if req in results:
165.         yield env.timeout(generators.get_writing_review_interval())
166.         if constants.statistics_enable:

```

```

167.         statistics.append_presence_time(review_desk, env.now - arriving_timestamp)
168.         fix_stop_serving(review_desk, arriving_timestamp)
169.         update_reviews_per_day()
170.     else:
171.         statistics.increase_lost_reviews_quantity()
172.
173.     try_print('%7.4f %s successfully served' % (env.now, name))
174.     statistics.serving_times.append(env.now - starting_serving_timestamp)
175.     return
176.
177. def reset():
178.     global waiting_hall_fill
179.     global blocked
180.
181.     statistics.reset_statistics()
182.     waiting_hall_fill = 0
183.     blocked = False
184.
185. def get_efficiency_criteria():
186.     return (numpy.mean(statistics.reviews_per_day_set) -
187.            (constants.short_hairing_masters_quantity +
188.             constants.fashion_hairing_masters_quantity +
189.             constants.colouring_masters_quantity) -
190.            numpy.mean(statistics.waiting_hall_fills) -

```

```

191.         (numpy.mean(statistics.get_queue_lengths(entities.cashbox_one)) +
192.          numpy.mean(statistics.get_queue_lengths(entities.cashbox_two))))
193.
194. def get_reliability_interval_relative_width(values):
195.     t_distribution = stats.t(len(values)-1)
196.     left_bound_of_reliability_interval = t_distribution.ppf(1-constants.student_parameter/2)
197.
198.     mean = numpy.mean(criterias)
199.     reliability_interval = (left_bound_of_reliability_interval*numpy.std(values)/math.sqrt(len(criterias)))
200.     return reliability_interval/mean, mean, reliability_interval
201.
202. def increase_index(index, maximum):
203.     index += 1
204.     if index < maximum:
205.         return index
206.     else:
207.         return 0
208.
209. if (constants.find_optimal_number_of_clients):
210.     previous_means = []
211.     previous_means_index = 0
212.     print("%20s | %20s | %22s" % ("number of clients", "interval width (%)",
213.                                   "efficiency criterion"))
214.     print("-"*68)

```

```
215.     counter = 1
216.     accuracy = 1
217.     prev_accuracy = 1
218.     prev_prev_accuracy = 1
219.     general_accuracy = 1
220.     general_interval_width = 1
221.     general_mean = 1
222.     common_accuracy = 1
223.     common_prev_accuracy = 1
224.     common_prev_prev_accuracy = 1
225.     while (counter < constants.number_of_considered_means) or \
226.           (common_accuracy > constants.minimal_accuracy) or \
227.           (common_prev_accuracy > constants.minimal_accuracy) or \
228.           (common_prev_prev_accuracy > constants.minimal_accuracy):
229.
230.         prev_prev_accuracy = prev_accuracy
231.         prev_accuracy = accuracy
232.
233.         common_prev_prev_accuracy = common_prev_accuracy
234.         common_prev_accuracy = common_accuracy
235.         criterias = []
236.         for i in range(5):
237.
238.             env = entities.env
```

```

239.         env.process(source(env, constants.number_of_clients))
240.         env.run()
241.         criteria = get_efficiency_criteria()
242.         criterias.append(criteria)
243.         reset()
244.
245.     accuracy, mean, interval_width = get_reliability_interval_relative_width(criterias)
246.     #print("-")
247.
248.     if counter <= constants.number_of_considered_means:
249.         previous_means.append(mean)
250.         print("-")
251.     else:
252.         previous_means[previous_means_index] = mean
253.         previous_means_index = increase_index(previous_means_index, constants.number_of_considered_means)
254.         general_accuracy, general_mean, general_interval_width = get_reliability_interval_relative_width(previous_means)
255.         common_accuracy = (general_interval_width+interval_width)/general_mean
256.         print("%20i | %20.4f | %22s" % (constants.number_of_clients, common_accuracy*100,
257.                                         "%7.4f ± %7.4f" % (general_mean,general_interval_width+interval_width)))
258.     if (common_accuracy > constants.minimal_accuracy):
259.         winsound.Beep(500, 1000)
260.     else:
261.         winsound.Beep(2500, 1000)
262.     constants.number_of_clients += constants.step_number_of_clients

```



```

263.         counter += 1
264.     print("Optimal number of clients is %i" % (constants.number_of_clients - constants.step_number_of_clients*3))
265. else:
266.     env = entities.env
267.     env.process(source(env, constants.number_of_clients))
268.     env.run()
269.
270.
271. if (constants.statistics_enable):
272.     statistics.save_histogram(statistics.serving_times, 100,
273.                             "Serving times", "length of serving (minutes)", "quantity of clients")
274.     statistics.save_histogram(statistics.get_waiting_times(entities.cashbox_one), 50,
275.                             "Waiting time in cashbox one queue", "length of waiting (minutes)", "quantity of clients")
276.     statistics.save_histogram(statistics.get_waiting_times(entities.cashbox_two), 10,
277.                             "Waiting time in cashbox two queue", "length of waiting (minutes)", "quantity of clients")
278.     statistics.save_histogram(statistics.get_waiting_times(entities.short_hairing_hall), 50,
279.                             "Waiting time in short hairing hall queue", "length of waiting (minutes)", "quantity of clients")
280.     statistics.save_histogram(statistics.get_waiting_times(entities.fashion_hairing_hall), 50,
281.                             "Waiting time in fashion hairing hall queue", "length of waiting (minutes)", "quantity of clients")
282.     statistics.save_histogram(statistics.get_waiting_times(entities.colouring_hall), 50,
283.                             "Waiting time in colouring hall queue", "length of waiting (minutes)", "quantity of clients")
284.
285.     statistics.save_histogram(statistics.get_presence_times(entities.cashbox_one), 50,
286.                             "Presence time in cashbox one", "length of presence (minutes)", "quantity of clients")

```

```

287. statistics.save_histogram(statistics.get_presence_times(entities.cashbox_two), 10,
288.     "Presence time in cashbox two", "length of presence (minutes)", "quantity of clients")
289. statistics.save_histogram(statistics.get_presence_times(entities.short_hairing_hall), 50,
290.     "Presence time in short hairing hall", "length of presence (minutes)", "quantity of clients")
291. statistics.save_histogram(statistics.get_presence_times(entities.fashion_hairing_hall), 50,
292.     "Presence time in fashion hairing hall", "length of presence (minutes)", "quantity of clients")
293. statistics.save_histogram(statistics.get_presence_times(entities.colouring_hall), 50,
294.     "Presence time in colouring hall queue", "length of presence (minutes)", "quantity of clients")
295.
296. print("%f" % numpy.mean(statistics.get_queue_lengths(entities.cashbox_one)))
297. print("%f" % numpy.mean(statistics.get_queue_lengths(entities.cashbox_two)))
298. print("%f" % numpy.mean(statistics.get_queue_lengths(entities.short_hairing_hall)))
299. print("%f" % numpy.mean(statistics.get_queue_lengths(entities.fashion_hairing_hall)))
300. print("%f" % numpy.mean(statistics.get_queue_lengths(entities.colouring_hall)))
301.
302. print("%f" % numpy.mean(statistics.get_intensity_components(entities.cashbox_one)))
303. print("%f" % numpy.mean(statistics.get_intensity_components(entities.cashbox_two)))
304. print("%f" % numpy.mean(statistics.get_intensity_components(entities.short_hairing_hall)))
305. print("%f" % numpy.mean(statistics.get_intensity_components(entities.fashion_hairing_hall)))
306. print("%f" % numpy.mean(statistics.get_intensity_components(entities.colouring_hall)))
307. print("%f" % numpy.mean(statistics.get_intensity_components(entities.review_desk)))
308.
309. print("%f" % numpy.mean(statistics.get_waiting_times(entities.cashbox_one)))
310. print("%f" % numpy.mean(statistics.get_waiting_times(entities.cashbox_two)))

```

```

311.     print("%f" % numpy.mean(statistics.get_waiting_times(entities.short_hairing_hall)))
312.     print("%f" % numpy.mean(statistics.get_waiting_times(entities.fashion_hairing_hall)))
313.     print("%f" % numpy.mean(statistics.get_waiting_times(entities.colouring_hall)))
314.
315.     print("%f" % numpy.mean(statistics.get_presence_times(entities.cashbox_one)))
316.     print("%f" % numpy.mean(statistics.get_presence_times(entities.cashbox_two)))
317.     print("%f" % numpy.mean(statistics.get_presence_times(entities.short_hairing_hall)))
318.     print("%f" % numpy.mean(statistics.get_presence_times(entities.fashion_hairing_hall)))
319.     print("%f" % numpy.mean(statistics.get_presence_times(entities.colouring_hall)))
320.
321.     print("%f" % numpy.mean(statistics.get_service_intensity_components(entities.cashbox_one)))
322.     print("%f" % numpy.mean(statistics.get_service_intensity_components(entities.cashbox_two)))
323.     print("%f" % numpy.mean(statistics.get_service_intensity_components(entities.short_hairing_hall)))
324.     print("%f" % numpy.mean(statistics.get_service_intensity_components(entities.fashion_hairing_hall)))
325.     print("%f" % numpy.mean(statistics.get_service_intensity_components(entities.colouring_hall)))
326.     print("%f" % numpy.mean(statistics.get_service_intensity_components(entities.review_desk)))
327.
328.     print("%f" % (statistics.lost_reviews/constants.number_of_clients))
329.     print("%f" % (statistics.lost/constants.number_of_clients))
330. #show_histogram(statistics.cashbox_queue_waiting_times[0], 100, "Cashbox one queue waiting times", "length of waiting (minutes)", "quantity of clients")
331. #show_histogram(statistics.cashbox_queue_waiting_times[1], 100, "Cashbox two queue waiting times", "length of waiting (minutes)", "quantity of clients")
332.

```

Файл constants.py

```
1. short_hairing_masters_quantity = 3
2. fashion_hairing_masters_quantity = 5
3. colouring_masters_quantity = 2
4.
5. waiting_hall_max_fullness = 20
6.
7. short_hairing_client_probability = 0.3
8. fashion_hairing_client_probability = 0.45
9.
10. priority_client_probability = 0.28
11.
12. short_hairing_client_class_id = 1
13. fashion_hairing_client_class_id = 2
14. colouring_client_class_id = 3
15.
16. staff_priority_id = 0
17. important_client_priority_id = 1
18. regular_client_priority_id = 2
19.
20. max_blocking_interval = 1000
21.
22. verbous = True
```

```
23. statistics_enable = True
24.
25. number_of_clients = 500
26. step_number_of_clients = 100
27.
28. find_optimal_number_of_clients = False
29. student_parameter = 0.05
30.
31. minimal_accuracy = 0.05
32. minimal_stability = 0.01
33. number_of_considered_means = 5
34.
35. time_of_switching_entrance = 100
36.
```

Файл entities.py

```
1. import simpy
2. import constants
3.
4. env = simpy.Environment()
5.
6. #devices
7. unblock_event = env.event()
8. cashbox_one = simpy.PriorityResource(env, capacity=1)
9. cashbox_two = simpy.PriorityResource(env, capacity=1)
10.
```

```

11. short_hairing_hall = simpy.Resource(env, capacity=constants.short_hairing_masters_quantity)
12. fashion_hairing_hall = simpy.Resource(env, capacity=constants.fashion_hairing_masters_quantity)
13. colouring_hall = simpy.Resource(env, capacity=constants.colouring_masters_quantity)
14. waiting_after_colouring = simpy.Resource(env, capacity=constants.waiting_hall_max_fullness*2)
15.
16. review_desk = simpy.Resource(env, capacity=1)
17.

```

Файл generators.py

```

1. import numpy
2. import constants
3.
4. def get_class_id():
5.     num = numpy.random.rand()
6.     if (num < constants.short_hairing_client_probability):
7.         return constants.short_hairing_client_class_id
8.     if (num < constants.short_hairing_client_probability + constants.fashion_hairing_client_probability):
9.         return constants.fashion_hairing_client_class_id
10.    return constants.colouring_client_class_id
11.
12. def get_random_priority():
13.    num = numpy.random.rand()
14.    if (num < constants.priority_client_probability):
15.        return constants.important_client_priority_id
16.    return constants.regular_client_priority_id
17.

```

```
18. def get_interval_before_new_customer():
19.     return gamma(35,0.002,0.08,0.13)*60
20.
21. def get_interval_before_new_customer_epidemic():
22.     return gamma(45,0.002,0.12,0.195)*60
23.
24. def get_interval_before_new_customer_summer():
25.     return gamma(30,0.002,0.056,0.091)*60
26.
27. def get_writing_review_interval():
28.     return gamma(4,0.7,3,5)
29.
30. def get_waiting_after_colouring_interval():
31.     return gamma(40,0.8,25,40)
32.
33. def get_service_colouring_interval():
34.     return gamma(12,0.8,5,15)
35.
36. def get_service_fashion_hairing_interval():
37.     return gamma(12,5,40,90)
38.
39. def get_service_short_hairing_interval():
40.     return gamma(22,0.85,20,25)
41.
```

```
42. def get_service_cashbox_interval():
43.     return gamma(20,0.3,1,5)
44.
45. def get_waiting_interval():
46.     k = 0.033
47.     num = numpy.random.rand()
48.     if num <= k * 1:
49.         return gamma(2, 5, 1, 2)
50.     elif num <= k * 3:
51.         return gamma(2, 5, 2, 3)
52.     elif num <= k * 6:
53.         return gamma(2, 5, 3, 4)
54.     elif num <= k * 10:
55.         return gamma(2, 5, 4, 4.9)
56.     elif num <= k * 20:
57.         return gamma(2, 5, 4.9, 5.1)
58.     elif num <= k * 24:
59.         return gamma(2, 5, 5.1, 6)
60.     elif num <= k * 27:
61.         return gamma(2, 5, 6, 7)
62.     elif num <= k * 29:
63.         return gamma(2, 5, 7, 9)
64.     else:
65.         return gamma(2, 5, 9, 10)
```



```

66.     return num
67.
68. def gamma(shape, size, min, max):
69.     result = numpy.random.gamma(shape,size)
70.     while (result < min) or (result > max):
71.         result = numpy.random.gamma(shape,size)
72.     return result

```

Файл statistics.py

```

1. import matplotlib.pyplot as plt
2.
3. def reset_statistics():
4.     global cashbox_queue_lengths
5.     global serving_times
6.     global cashbox_queue_length_sets
7.     global cashbox_queue_waiting_times
8.
9.     global lost
10.    global lost_reviews
11.
12.    global queue_lengths
13.    global waiting_times
14.    global queue_length
15.
16.    global reviews_per_day_set
17.

```

```
18.     global waiting_hall_fills
19.
20.     global reviews_per_day
21.
22.     global last_time_writing_reviews
23.     global presence_times
24.     global intensity_components
25.     global service_intensity_components
26.     global figures_counter
27.
28.     cashbox_queue_lengths = [0,0]
29.     serving_times = []
30.     cashbox_queue_length_sets = [[],[]]
31.     cashbox_queue_waiting_times = [[],[]]
32.     figures_counter = 0
33.     lost = 0
34.     lost_reviews = 0
35.
36.     queue_lengths = {}
37.     waiting_times = {}
38.     queue_length = {}
39.     presence_times = {}
40.     intensity_components = {}
41.     service_intensity_components = {}
```

```
42.
43.     reviews_per_day_set = []
44.
45.     waiting_hall_fills = []
46.
47.     reviews_per_day = 0
48.
49.     last_time_writing_reviews = 0
50.
51. figures_counter = 0
52. cashbox_queue_lengths = [0,0]
53. serving_times = []
54. cashbox_queue_length_sets = [[],[]]
55. cashbox_queue_waiting_times = [[],[]]
56.
57. lost = 0
58. lost_reviews = 0
59.
60. last_seen_input_time = {}
61. queue_lengths = {}
62. waiting_times = {}
63. queue_length = {}
64. presence_times = {}
65. intensity_components = {}
```

```
66. service_intensity_components = {}
67.
68. reviews_per_day_set = []
69.
70. waiting_hall_fills = []
71.
72.
73. reviews_per_day = 0
74. day_length = 480
75.
76. last_time_writing_reviews = 0
77.
78. def get_last_seen_input_time(resource):
79.     try:
80.         return last_seen_input_time[resource]
81.     except:
82.         last_seen_input_time[resource] = -1
83.         return last_seen_input_time[resource]
84.
85. def set_last_seen_input_time(resource, value):
86.     last_seen_input_time[resource] = value
87.
88. def get_queue_length(resource):
89.     try:
```

```
90.         return queue_length[resource]
91.     except:
92.         queue_length[resource] = 0
93.         return queue_length[resource]
94.
95. def increase_queue_length(resource):
96.     try:
97.         queue_length[resource] += 1
98.     except:
99.         queue_length[resource] = 1
100.
101. def decrease_queue_length(resource):
102.     try:
103.         queue_length[resource] -= 1
104.     except:
105.         queue_length[resource] = 0
106.
107. def append_queue_length(resource):
108.     if (queue_length[resource] == 0):
109.         return
110.     append_value_to_collection(resource, queue_lengths, queue_length[resource])
111.
112. def append_presence_time(resource, time):
113.     if (time == 0):
```

```
114.         return
115.     append_value_to_collection(resource, presence_times, time)
116.
117. def get_presence_times(resource):
118.     return get_values_from_collection(resource, presence_times)
119.
120. def append_intensity_component(resource, intensity_component):
121.     if (intensity_component == 0):
122.         return
123.     append_value_to_collection(resource, intensity_components, intensity_component)
124.
125. def get_intensity_components(resource):
126.     return get_values_from_collection(resource, intensity_components)
127.
128. def append_service_intensity_component(resource, intensity_component):
129.     if (intensity_component == 0):
130.         return
131.     append_value_to_collection(resource, service_intensity_components, intensity_component)
132.
133. def get_service_intensity_components(resource):
134.     return get_values_from_collection(resource, service_intensity_components)
135.
136. def get_queue_lengths(resource):
137.     return get_values_from_collection(resource, queue_lengths)
```

```
138.
139. def append_waiting_time(resource, time):
140.     if (time == 0):
141.         return
142.     append_value_to_collection(resource, waiting_times, time)
143.
144. def get_waiting_times(resource):
145.     return get_values_from_collection(resource, waiting_times)
146.
147. def append_value_to_collection(resource, collection, value):
148.     try:
149.         collection[resource].append(value)
150.     except:
151.         collection[resource] = []
152.         collection[resource].append(value)
153.
154. def get_values_from_collection(resource, collection):
155.     try:
156.         return collection[resource]
157.     except:
158.         return [0]
159.
160. def show_histogram(collection, number_of_intervals, title, xlabel, ylabel):
161.     global figures_counter
```

```
162. plt.hist(collection,number_of_intervals)
163. plt.title(title)
164. plt.xlabel(xlabel)
165. plt.ylabel(ylabel)
166. plt.grid(True)
167. plt.show()
168.
169.def save_histogram(collection, number_of_intervals, title, xlabel, ylabel):
170.     global figures_counter
171.     plt.hist(collection,number_of_intervals)
172.     plt.title(title)
173.     plt.xlabel(xlabel)
174.     plt.ylabel(ylabel)
175.     plt.grid(True)
176.     plt.savefig("figure_%i.png" % figures_counter)
177.     figures_counter += 1
178.     plt.gcf().clear()
179.
180.def increase_lost_quantity():
181.     global lost
182.     lost += 1
183.
184.def increase_lost_reviews_quantity():
185.     global lost_reviews
```



```
186.     lost_reviews += 1
```

Сценарии работы модели

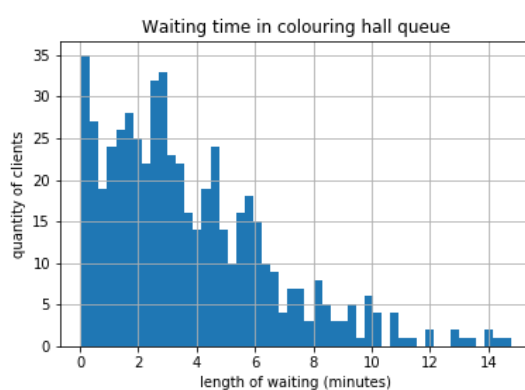
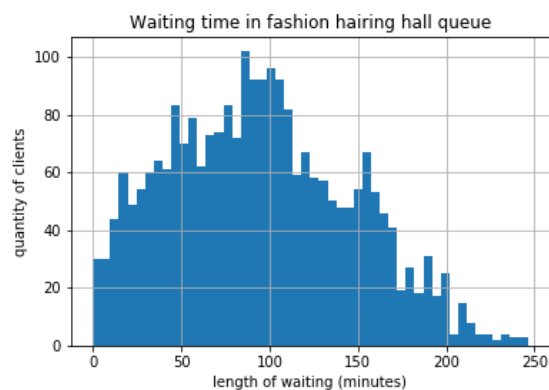
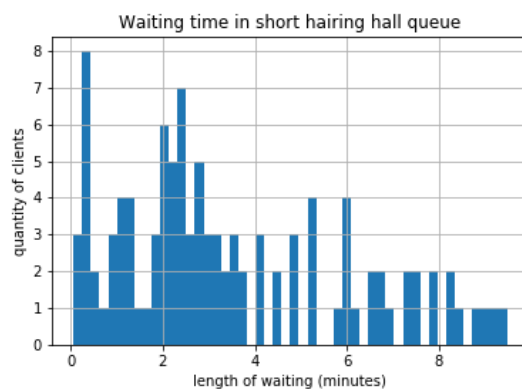
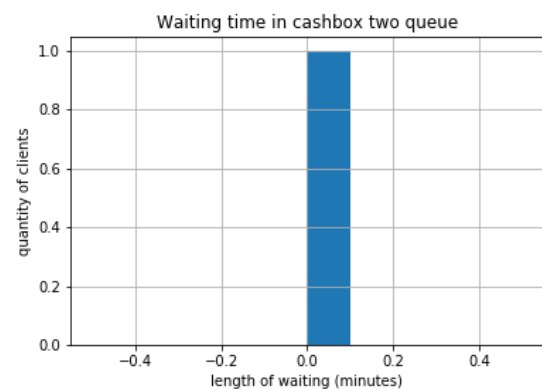
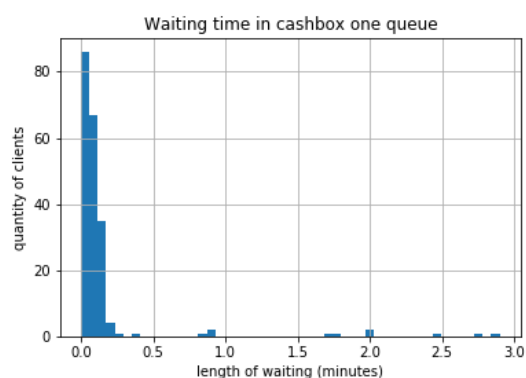
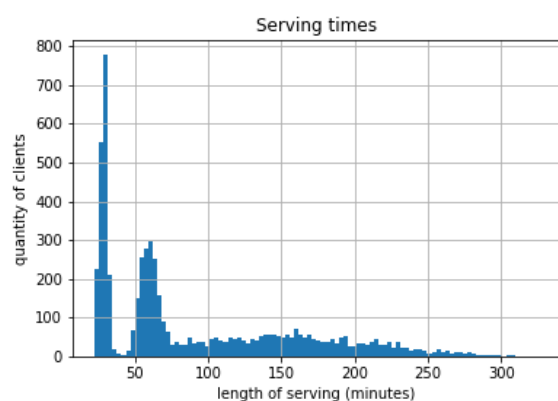
Обычный режим работы

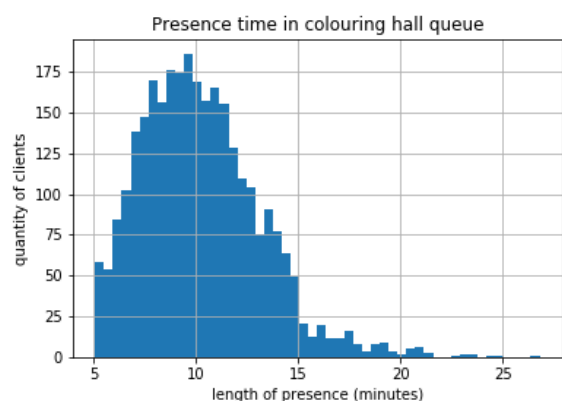
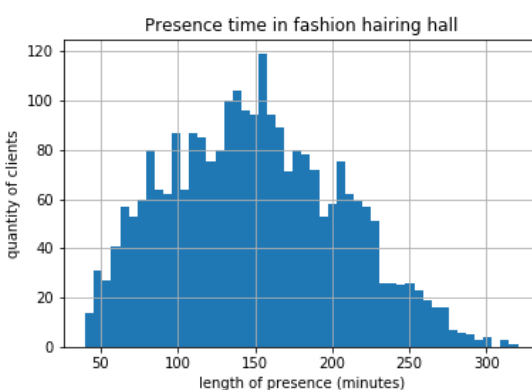
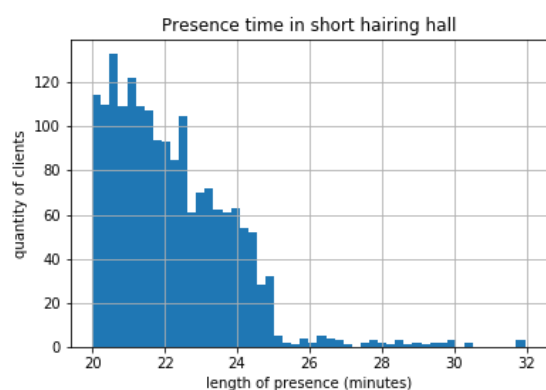
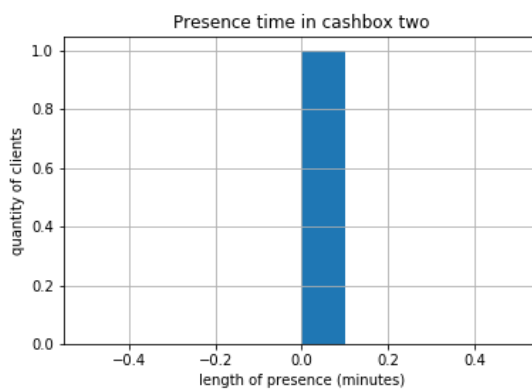
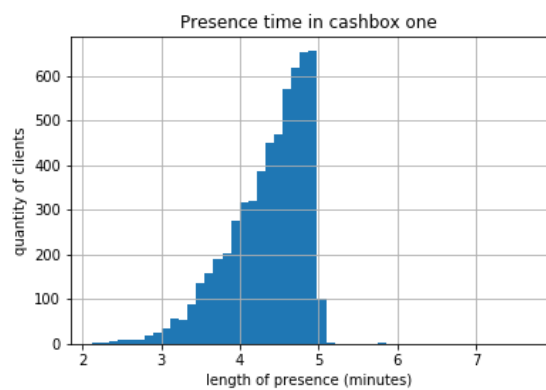
Поиск оптимального числа заявок

number of clients	interval width (%)	efficiency criterion
5500	6.0635	33.4395 ± 2.0276
5600	4.6881	34.6189 ± 1.6230
5700	4.0890	33.5378 ± 1.3714
5800	8.0270	34.1508 ± 2.7413
5900	5.8752	35.1945 ± 2.0678
6000	6.8349	33.3783 ± 2.2814
6100	6.8719	33.4548 ± 2.2990
6200	3.7946	34.0274 ± 1.2912
6300	4.6896	34.5365 ± 1.6196
6400	3.7223	33.3876 ± 1.2428

Optimal number of clients is 6200

Измеренные характеристики системы при оптимальном числе клиентов





Режим работы во время эпидемии гриппа:

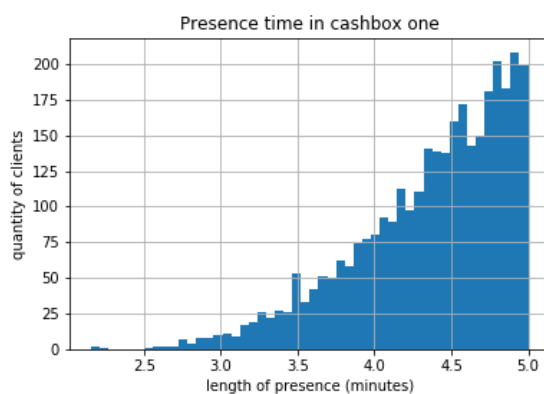
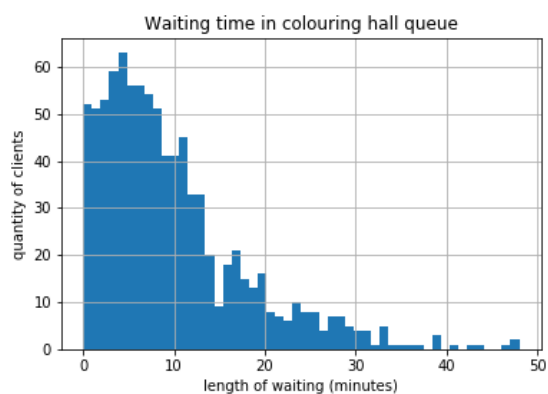
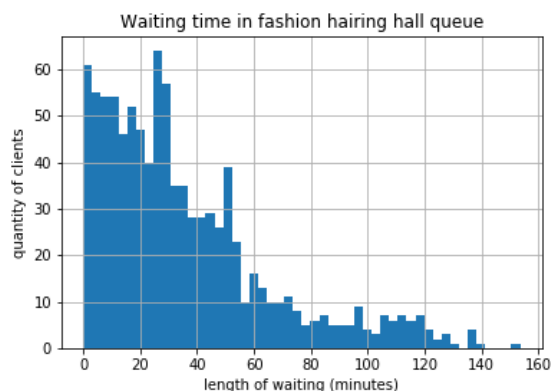
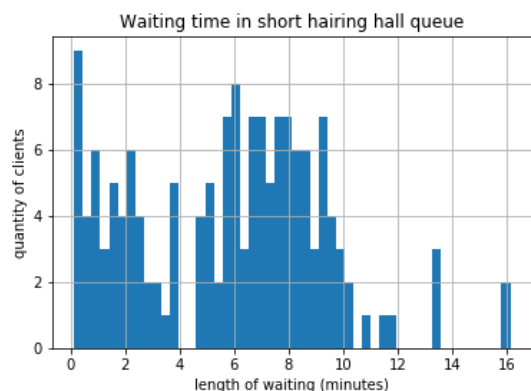
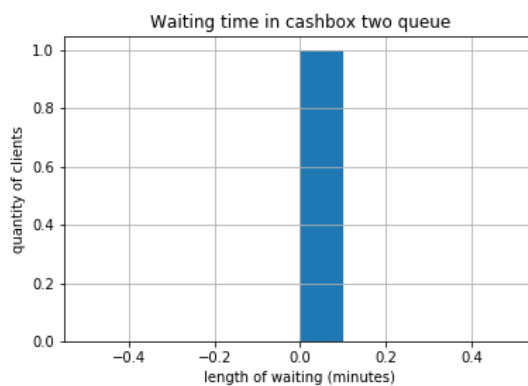
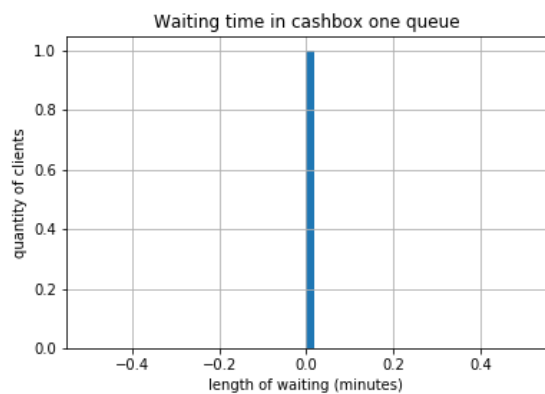
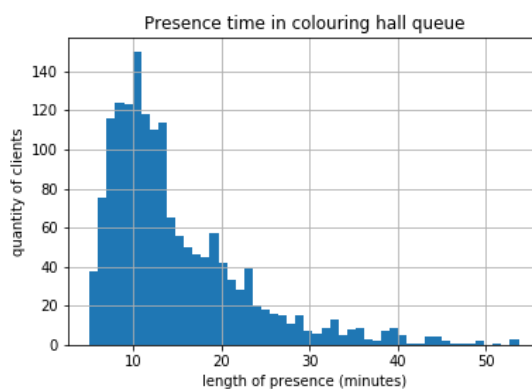
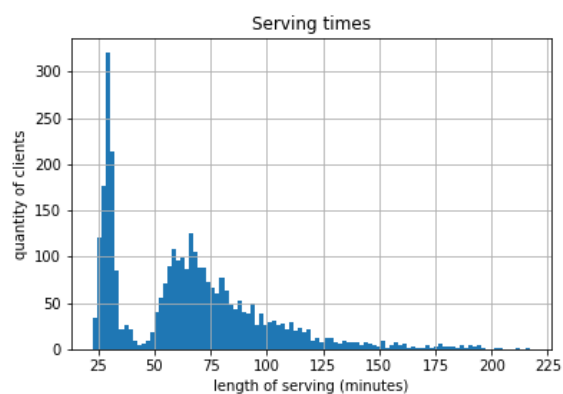
3 парикмахера заболели, по одному из каждого зала, интервал между посетителями увеличился в 1.5 раза.

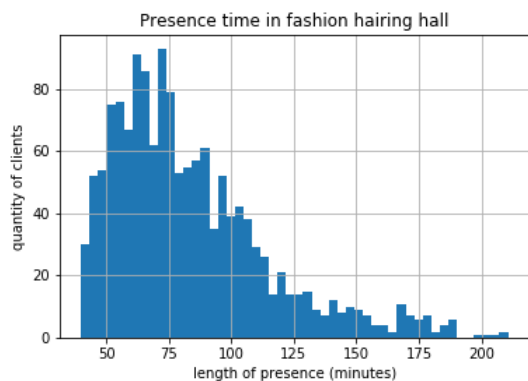
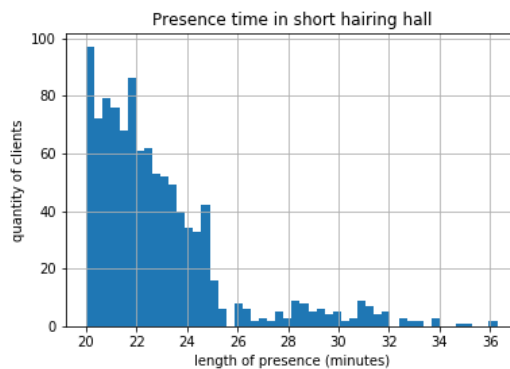
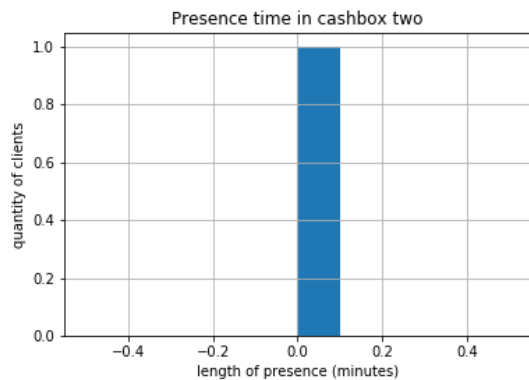
Поиск оптимального числа заявок

number of clients	interval width (%)	efficiency criterion
3000	4.2692	33.1622 ± 1.4158
3100	3.6132	33.8246 ± 1.2221
3200	5.3429	33.0615 ± 1.7664
3300	3.4134	34.0194 ± 1.1612
3400	2.7694	33.5984 ± 0.9305
3500	4.7767	33.6798 ± 1.6088

Optimal number of clients is 3300

Измеренные характеристики системы при оптимальном числе клиентов





Режим работы в разгар лета

3 парикмахера ушли в отпуск, по одному из каждого зала, интервал между посетителями уменьшился на 30%.

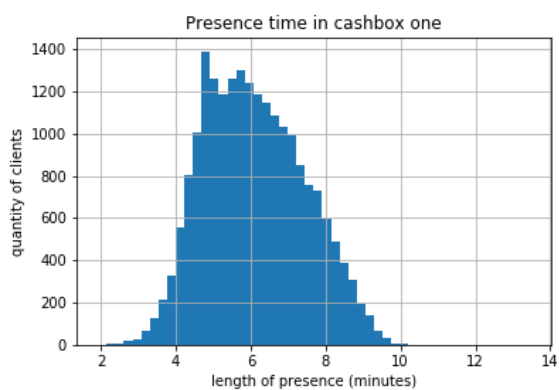
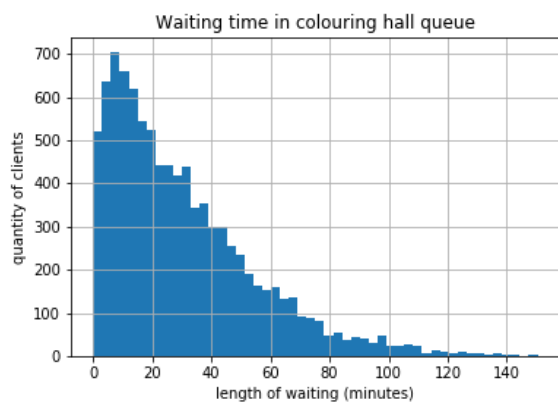
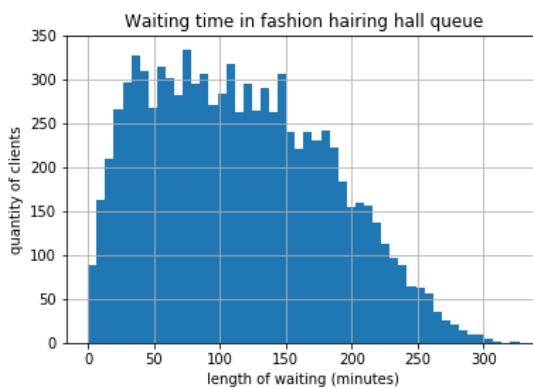
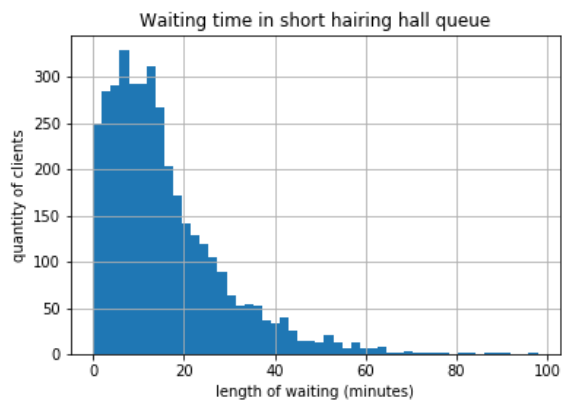
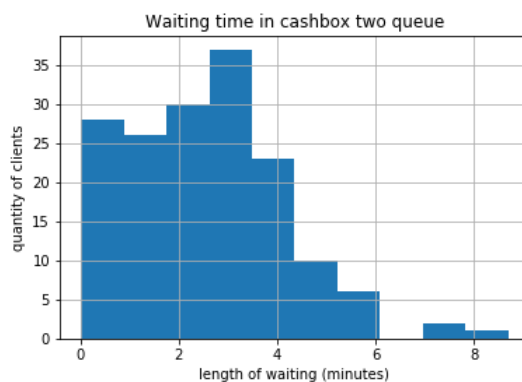
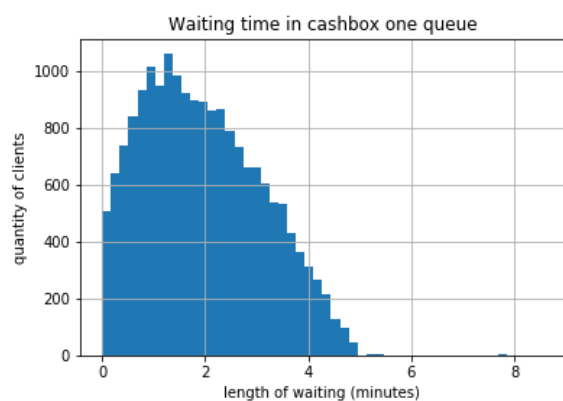
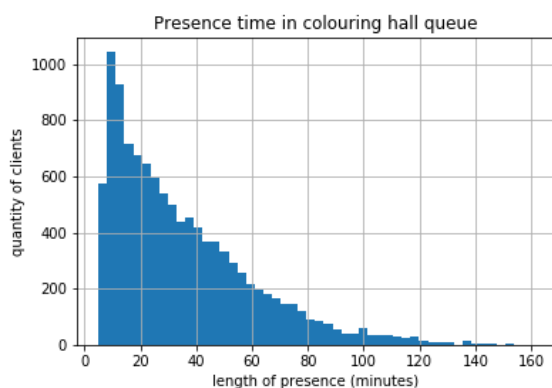
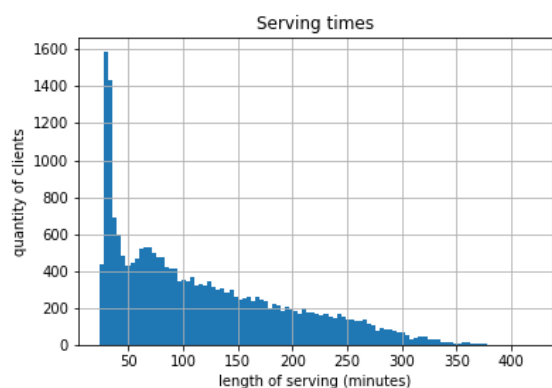
Поиск оптимального числа заявок

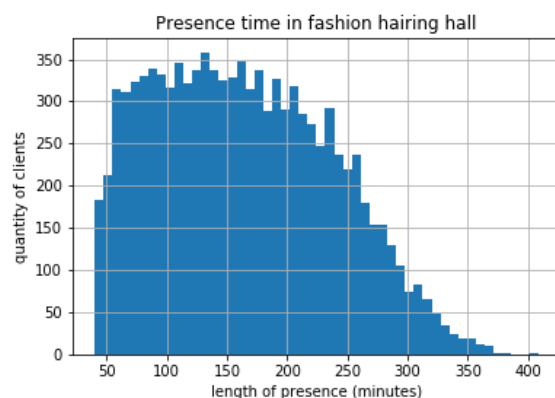
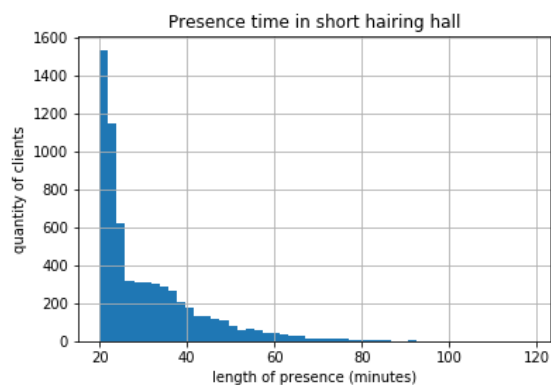
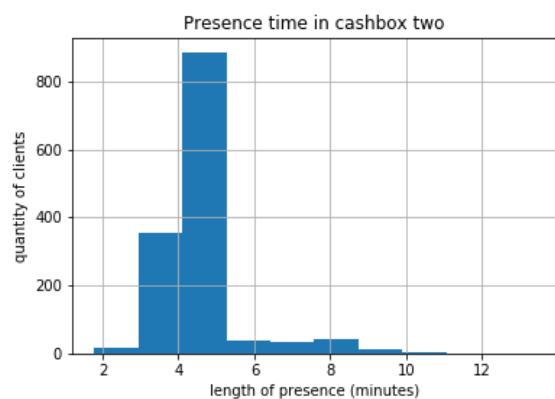
number of clients	interval width (%)	efficiency criterion
3000	20.7462	18.2746 ± 3.7913
3100	20.1571	18.1064 ± 3.6497
5000	8.9720	17.3493 ± 1.5566
5100	10.2057	17.8869 ± 1.8255
5200	11.6641	16.5774 ± 1.9336
5300	16.6203	16.8946 ± 2.8079
5400	10.3480	17.7466 ± 1.8364
5500	11.5326	18.1267 ± 2.0905
7000	9.4263	18.3037 ± 1.7254
7100	13.1935	18.2914 ± 2.4133
7200	10.4200	18.4799 ± 1.9256
10000	7.4065	18.0783 ± 1.3390
10100	13.2020	17.6059 ± 2.3243
10200	12.6470	18.0992 ± 2.2890
10300	5.6850	18.7040 ± 1.0633
10400	13.1255	17.5835 ± 2.3079
10500	9.5163	17.6184 ± 1.6766
20000	13.3549	18.2895 ± 2.4425
20100	9.3674	18.1265 ± 1.6980
25000	8.2466	18.0026 ± 1.4846
25100	5.3658	18.2404 ± 0.9788
31250	4.2431	17.7361 ± 0.7526
31500	5.2997	18.6601 ± 0.9889
31750	4.4788	18.6077 ± 0.8334
32000	7.4289	18.0245 ± 1.3390
32250	9.2563	17.3545 ± 1.6064
32500	7.6549	17.9321 ± 1.3727
61250	3.6824	18.5151 ± 0.6818
61500	2.2746	18.5766 ± 0.4225
61750	5.6559	18.3723 ± 1.0391
62000	4.2760	18.1464 ± 0.7759

62250	2.5097	18.6183 ± 0.4673
62500	5.1572	18.5216 ± 0.9552
62750	4.9001	17.8839 ± 0.8763
63000	3.8539	18.6843 ± 0.7201
63250	4.9366	18.0637 ± 0.8917

Optimal number of clients is 62750

Измеренные характеристики системы при оптимальном числе клиентов





Режим работы в случае образования повышенного количества модников

Доля заявок на стрижку под одну насадку снижена с 0.30 до 0.15; доля заявок, требующих модельную стрижку увеличена с 0.45 до 0.60.

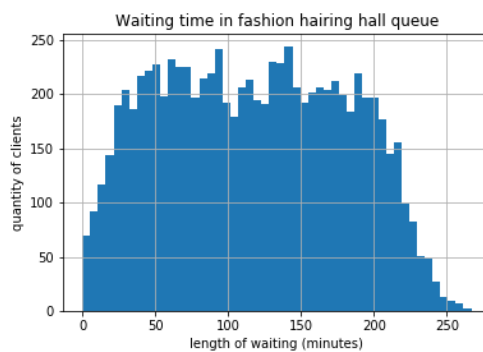
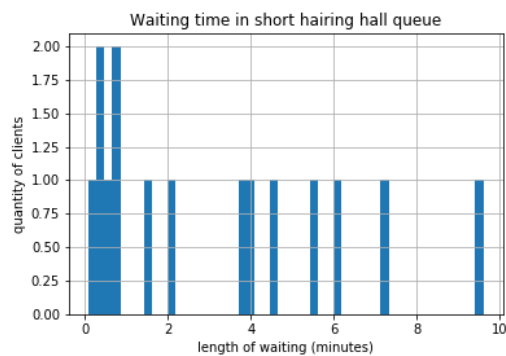
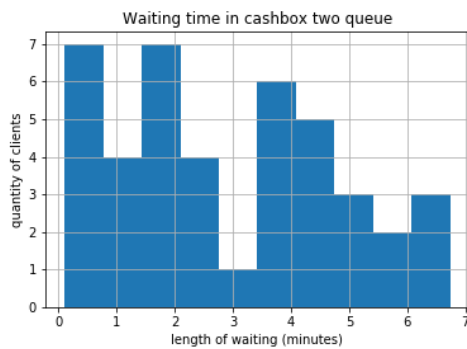
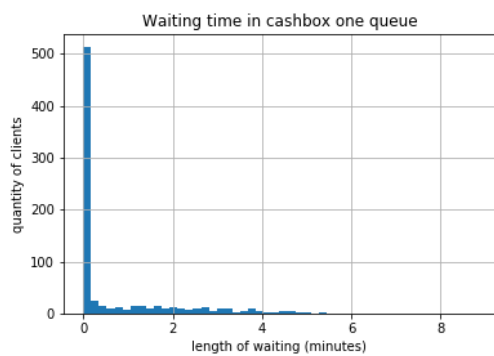
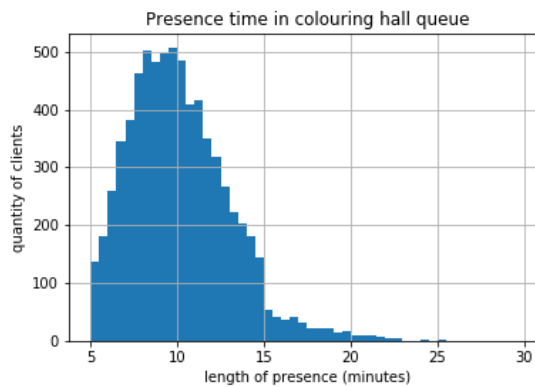
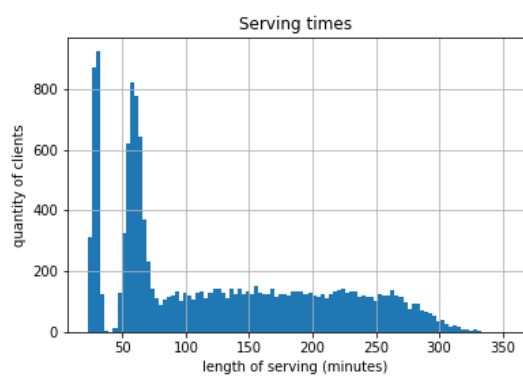
Поиск оптимального числа заявок

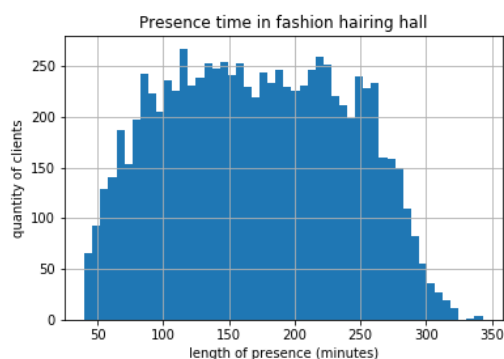
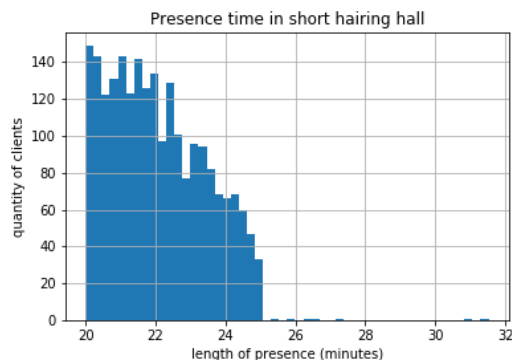
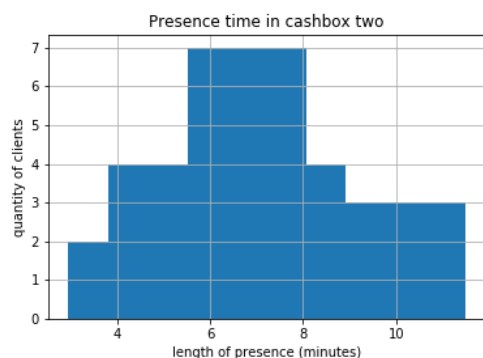
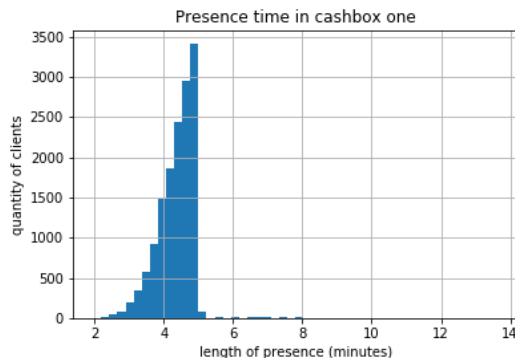
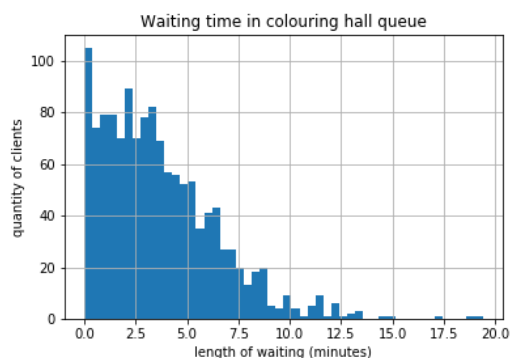
number of clients	interval width (%)	efficiency criterion
3000	5.2651	17.8285 ± 0.9387
3100	12.9264	17.4162 ± 2.2513
3200	8.5325	18.2032 ± 1.5532
3300	9.9808	16.4352 ± 1.6404
3400	14.2154	16.9418 ± 2.4083
3500	12.7466	18.4998 ± 2.3581
6000	8.0882	17.3188 ± 1.4008
6100	4.7593	18.0748 ± 0.8602
6200	13.6645	18.6536 ± 2.5489
6300	11.3666	17.7266 ± 2.0149
6400	10.0514	17.8764 ± 1.7968
6500	7.4623	18.6127 ± 1.3889
6600	6.8202	17.5608 ± 1.1977
6700	7.4759	18.4870 ± 1.3821
6800	7.4180	18.5020 ± 1.3725
10000	5.0245	17.8445 ± 0.8966
10100	5.3053	18.3432 ± 0.9732
10200	3.9081	18.6175 ± 0.7276
10300	8.0052	18.0538 ± 1.4452
10400	8.1634	17.8992 ± 1.4612
10500	5.4792	18.5270 ± 1.0151
10600	7.8271	18.8564 ± 1.4759
10700	3.8935	18.3663 ± 0.7151
10800	5.5459	18.4413 ± 1.0227
10900	5.6793	18.1924 ± 1.0332
15000	5.9441	18.6859 ± 1.1107
15100	5.4004	18.5546 ± 1.0020
15200	5.0974	18.3401 ± 0.9349

15300	6.6393	18.3915 ± 1.2211
15400	4.2194	18.2298 ± 0.7692
15500	2.7368	18.6355 ± 0.5100
15600	5.6867	18.7816 ± 1.0680
15700	4.0901	18.9797 ± 0.7763
15800	7.3974	18.6054 ± 1.3763
15900	9.1174	18.7630 ± 1.7107
25000	6.7003	18.2818 ± 1.2249
25100	3.3734	18.7770 ± 0.6334
25200	4.0850	18.8018 ± 0.7680
25300	3.8377	18.8195 ± 0.7222

Optimal number of clients is 25100

Измеренные характеристики системы при оптимальном числе клиентов





Некоторые обобщения

Анализ полученных данных

Так, в соответствии с тем, что в период эпидемии гриппа коэффициент эффективности системы снизился всего на 0.1 %, можно сделать вывод, что следует рассмотреть сокращение количества работающих мастеров – однако следует учитывать тот факт, что в период эпидемии снизился поток заявок в предприятие – это говорит о том, что увольнение сразу нескольких людей может негативно сказаться на работе системы.

Более того, средняя длина очереди во вторую кассу во всех случаях не превышает одного человека, а интенсивность входного потока заявок в первую кассу в среднем на 25 % меньше интенсивности обслуживания – отсюда можно сделать вывод о том, что во втором кассире отсутствует особая необходимость.

При моделировании режима работы при повышенном количестве модников коэффициент эффективности резко упал на 45 %, а вероятность потери клиента возросла более чем в 7 раз. Это говорит о том, что следует проводить мониторинг классов поступающих заявок и рассмотреть стратегию перераспределения мастеров по залам в периоды повышенного потока желающих модную стрижку.

Во время моделировании работы парикмахерской летом коэффициент эффективности также снизился более чем на 45 % по сравнению с максимальным значением, а вероятность потери клиента возросла более чем в 11 раз, что указывает на невысокую эффективность работы кассира при условиях повышенного потока клиентов. Помимо всего прочего полученные результаты, в частности, повышенные времена ожиданий в очередях, говорят о том, что, либо не следует предоставлять отпуск сразу нескольким мастерам, по крайней мере, в период повышенного потока клиентов, либо на периоды их отсутствия нанимать каких-либо других мастеров, согласных на временную работу. Также следует рассмотреть стратегию перераспределения мастеров по залам в периоды непредвиденного отсутствия двух и более работников.

Обобщающая таблица характеристик системы для разных сценариев работы при оптимальном числе заявок

	Обычный режим	Период эпидемии	Летний период	Повышенное число модников
Average cashbox one queue length	1.000000	1.000000	1.049582	1.000545
Average cashbox two queue length	1.000000	0.000000	1.000000	1.000000
Average short hairing queue length	1.004449	1.012393	1.587058	1.000447
Average fashion hairing queue length	7.796621	2.377376	7.349668	9.435068
Average colouring queue length	1.046305	1.315251	3.359909	1.045177
Cashbox one input intensity	0.189957	0.132391	0.201974	0.177511
Cashbox two input intensity	0.060121	0.000000	0.109073	0.066453
Short hairing hall input intensity	0.100061	0.070771	0.216079	0.063206
Fashion hairing hall input intensity	0.125173	0.085672	0.361698	0.150223
Colouring hall input intensity	0.584678	0.348231	0.714066	0.864544
Review desk input intensity	1.560285	1.593569	1.406799	162.941138
Average cashbox one waiting time	0.159993	0.000000	1.960015	0.758964
Average cashbox two waiting time	0.000000	0.000000	2.564269	2.951080
Average short hairing waiting time	3.468677	5.793693	15.752023	3.151969
Average fashion hairing waiting time	94.999977	35.470946	115.162002	116.952828
Average colouring waiting time	3.751529	9.840127	30.490369	3.671648
Average cashbox one presence time	4.363492	4.356536	6.097309	4.392072
Average cashbox two presence time	0.000000	0.000000	4.627210	7.101309
Average short hairing presence time	22.195015	22.921981	31.087208	22.091295
Average fashion hairing presence time	150.268871	83.482217	164.249811	171.227737
Average colouring presence time	10.252613	14.877537	35.408044	10.163618
Average cashbox one service intensity	0.233011	0.233051	0.233148	0.233366
Average cashbox two service intensity	0.000000	0.000000	0.234984	0.237466
Average short hairing service intensity	0.045637	0.045482	0.045572	0.045477
Average fashion hairing service intensity	0.017158	0.017306	0.017140	0.017076
Average colouring service intensity	0.111338	0.112324	0.112925	0.112192
Average review desk service intensity	0.269068	0.267748	0.269286	0.269318
Losing review probability	0.358548	0.290303	0.115044	0.187849
Losing client probability	0.057258	0.000000	0.645976	0.416932
Efficiency criterion	34.0274 ± 1.2912	34.0194 ± 1.1612	17.8839 ± 0.8763	18.7770 ± 0.6334