**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Университет ИТМО

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](#_Toc500605696)

[Описание модели 3](#_Toc500605697)

[Схема модели 4](#_Toc500605698)

[Цель моделирования 5](#_Toc500605699)

[Разработка имитационной модели Anylogic 6](#_Toc500605700)

[Основные дополнительные упрощения и допущения 6](#_Toc500605701)

[Скриншоты модели 6](#_Toc500605702)

[Сценарии работы модели 28](#_Toc500605703)

[Обычный режим работы 28](#_Toc500605704)

[Режим работы во время эпидемии гриппа 30](#_Toc500605705)

[Режим работы в разгар лета 32](#_Toc500605706)

[Режим работы в случае образования повышенного количества модников 34](#_Toc500605707)

[Некоторые обобщения 36](#_Toc500605708)

[Разработка имитационной модели SimPy 37](#_Toc500605709)

[Дополнительные замечания 37](#_Toc500605710)

[Исходный код 38](#_Toc500605711)

[Сценарии работы модели 66](#_Toc500605712)

[Обычный режим работы 66](#_Toc500605713)

[Режим работы во время эпидемии гриппа: 67](#_Toc500605714)

[Режим работы в разгар лета 69](#_Toc500605715)

[Режим работы в случае образования повышенного количества модников 71](#_Toc500605716)

[Некоторые обобщения 73](#_Toc500605717)

# 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 *alerana.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 *alerana.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

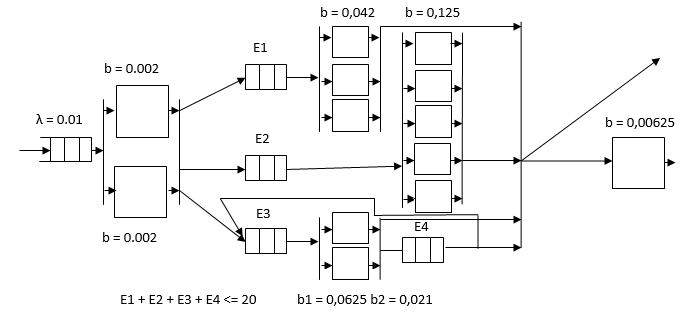
11) По минимуму обслуживание клиента включает приветствие (1-2 секунды), вопрос - ответ о желаемой стрижке (около 5 секунд), ожидание платежных средств от клиента (от 5 до 10 секунд) и проведение операции с кассой (также от 5 до 10 секунд) и составляет в целом 15-30 секунд. Однако нельзя не предусмотреть 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 просты. 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**



Сценарии модернизации системы обслуживания:

1. Перевод мастеров из одних залов в другие, увольнение лишних мастеров

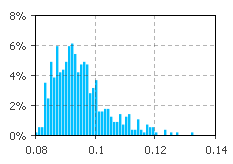
2. Изменение количества касс до оптимального значения

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

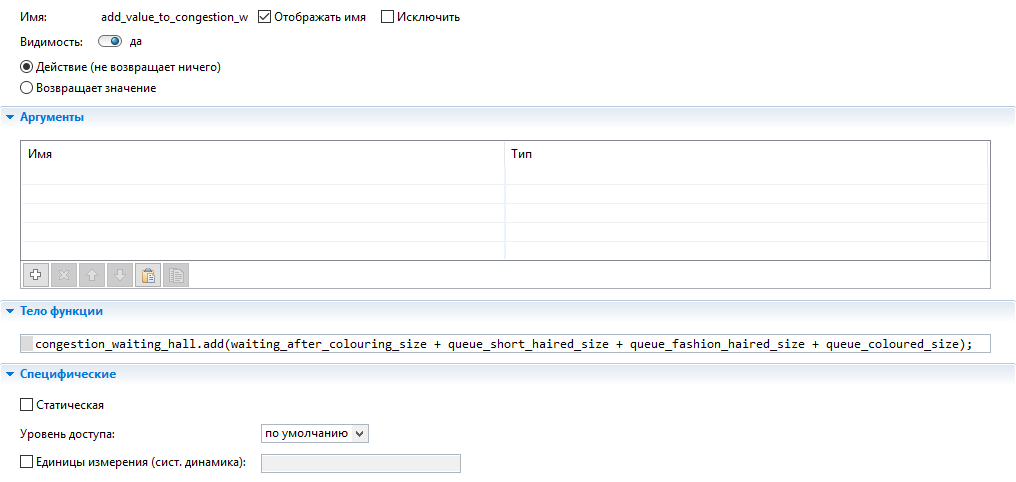
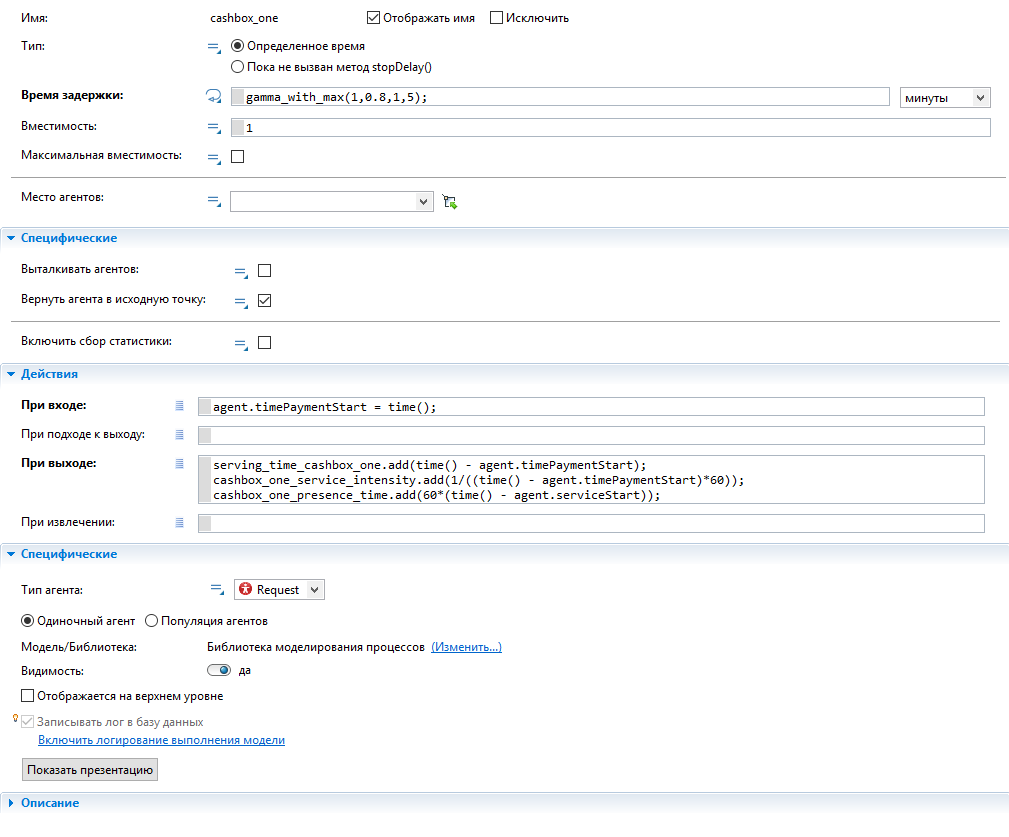
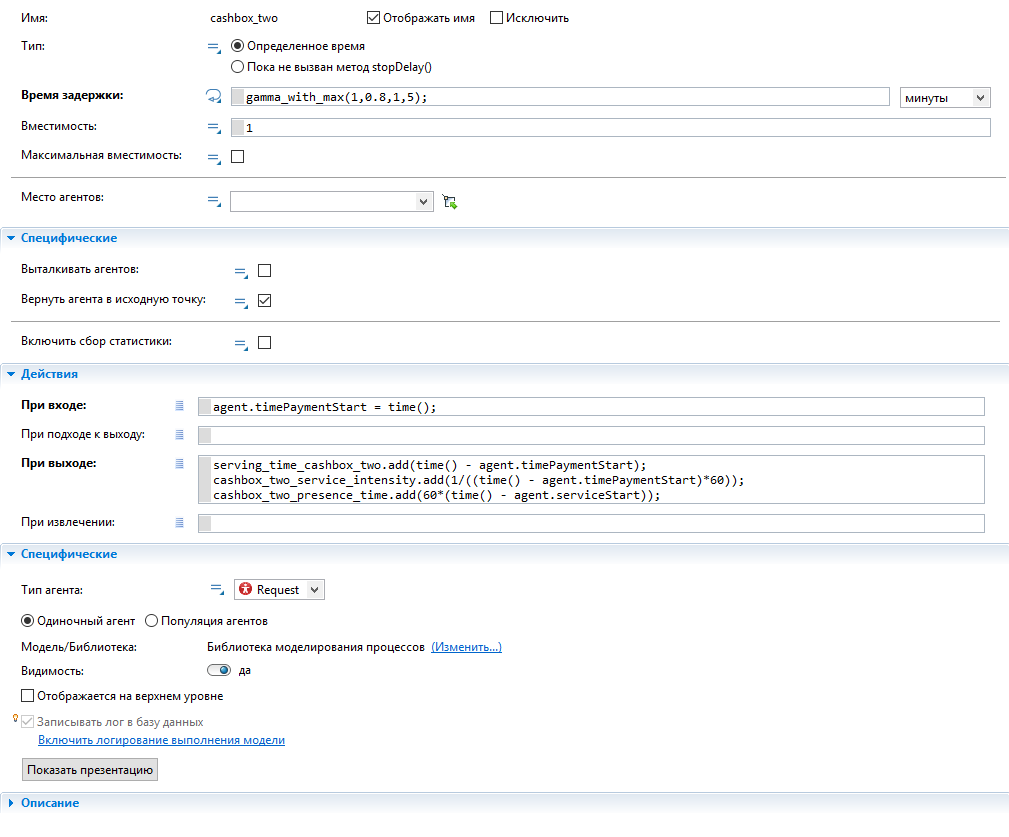
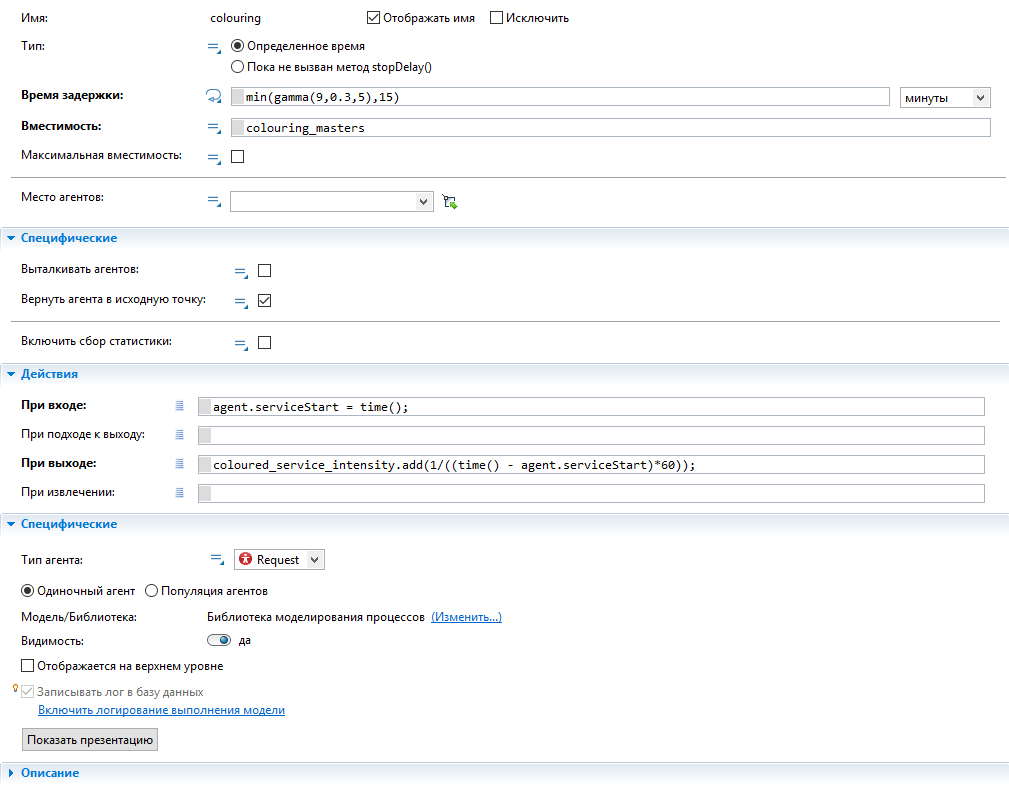
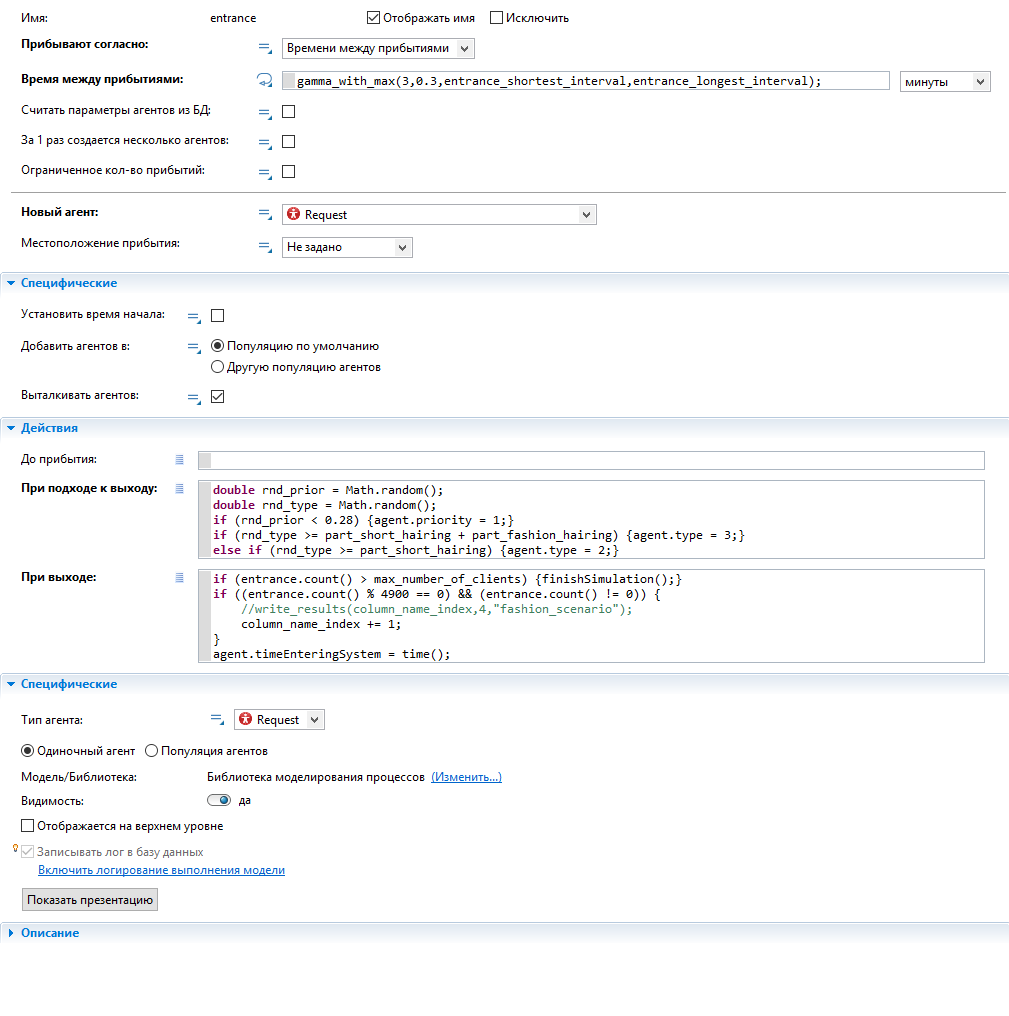
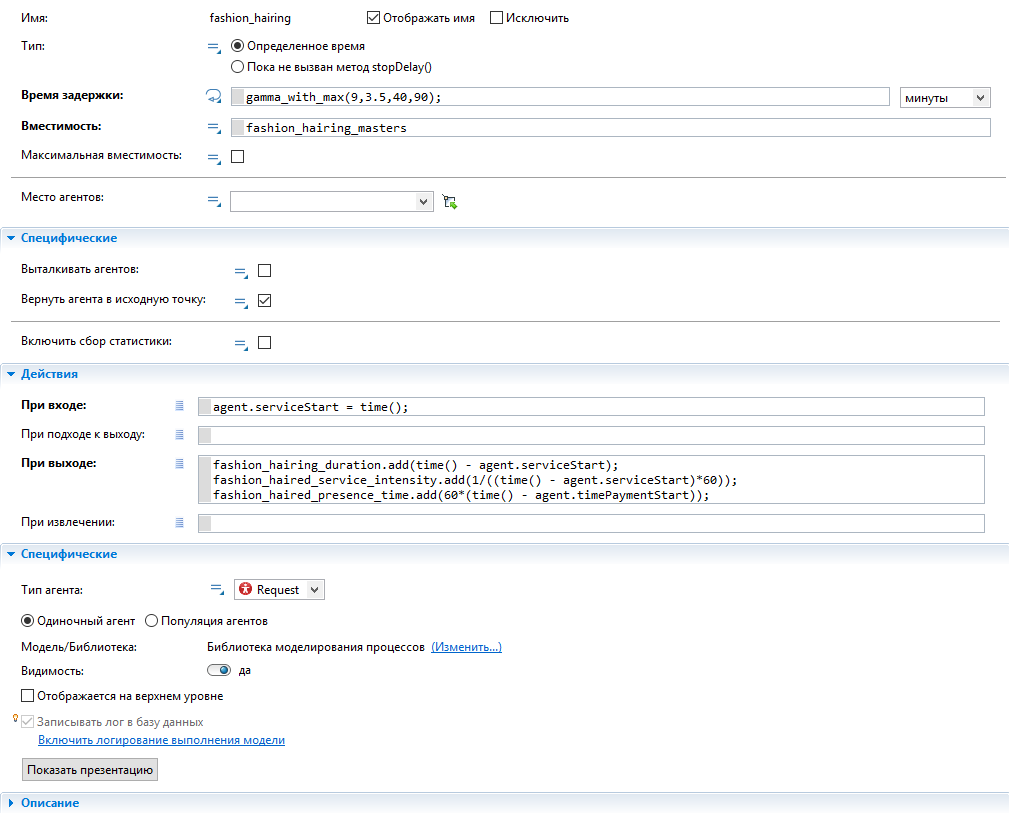
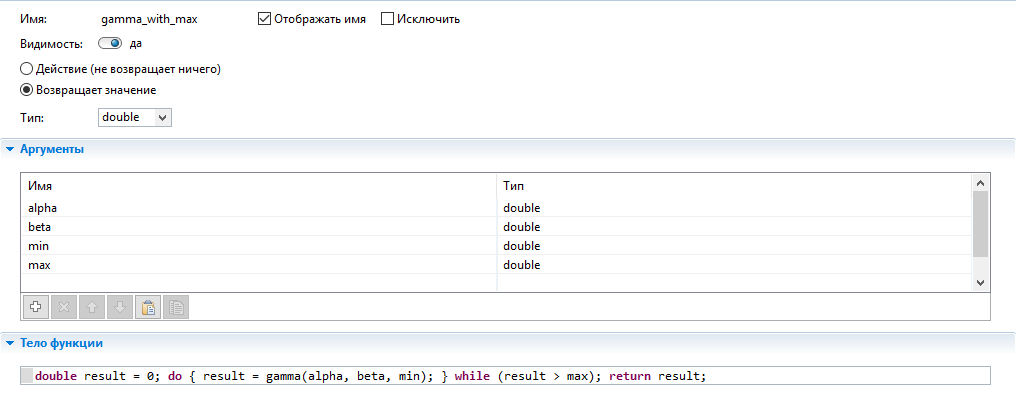
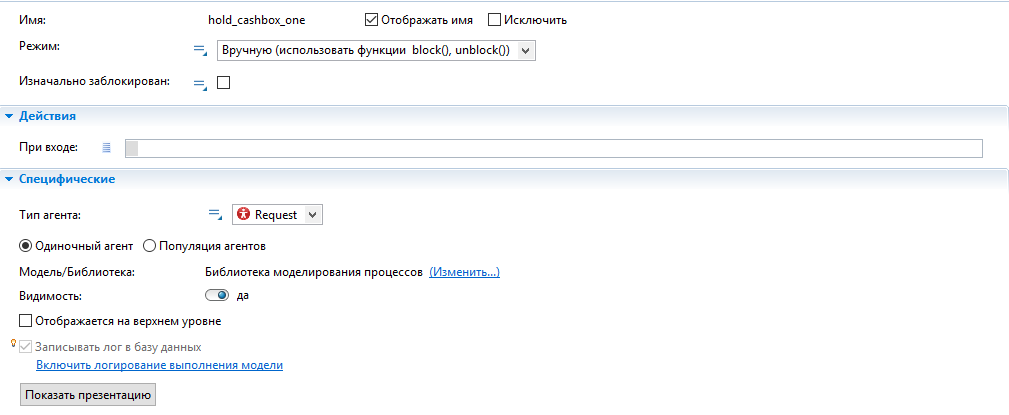
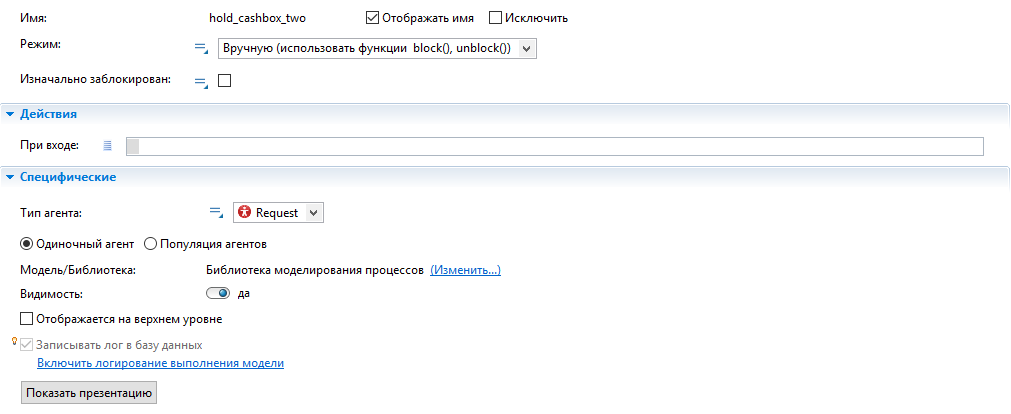
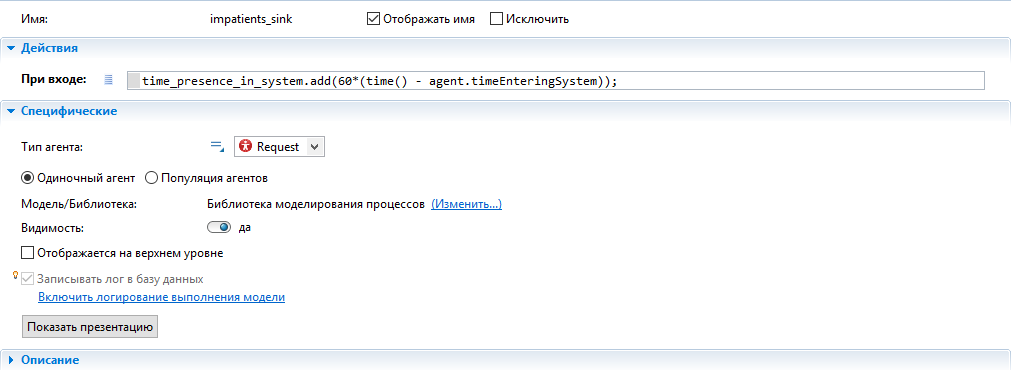
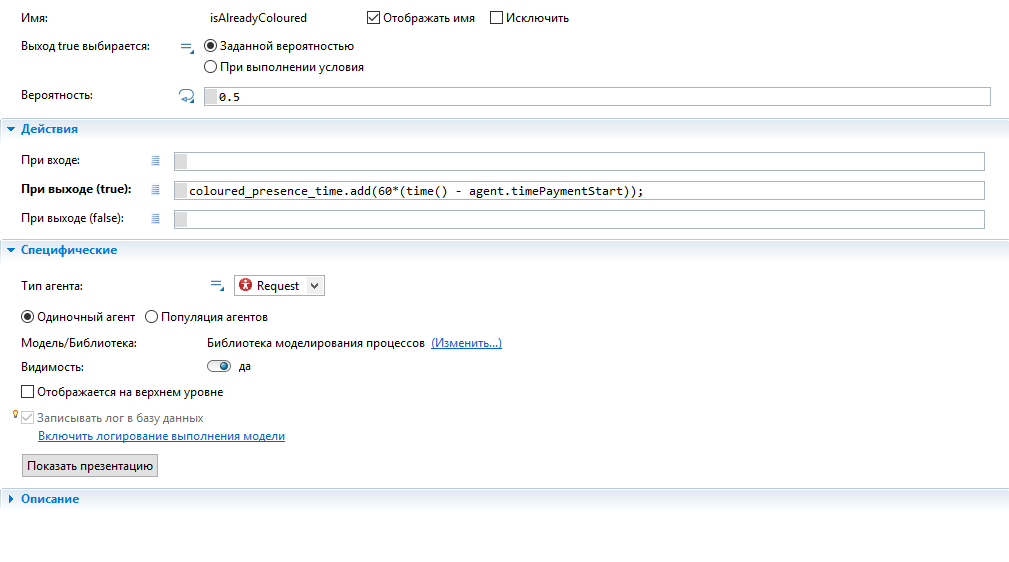
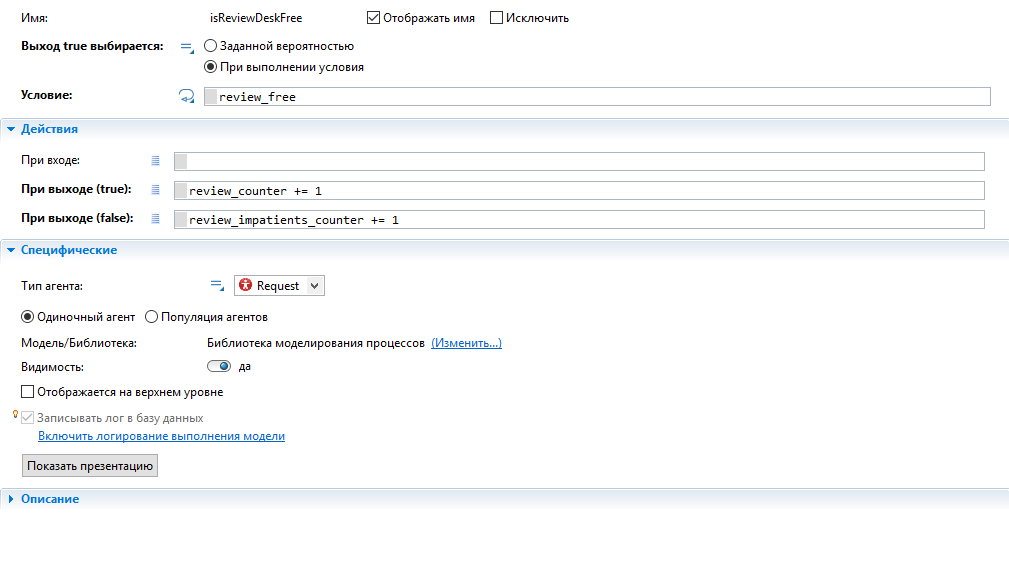
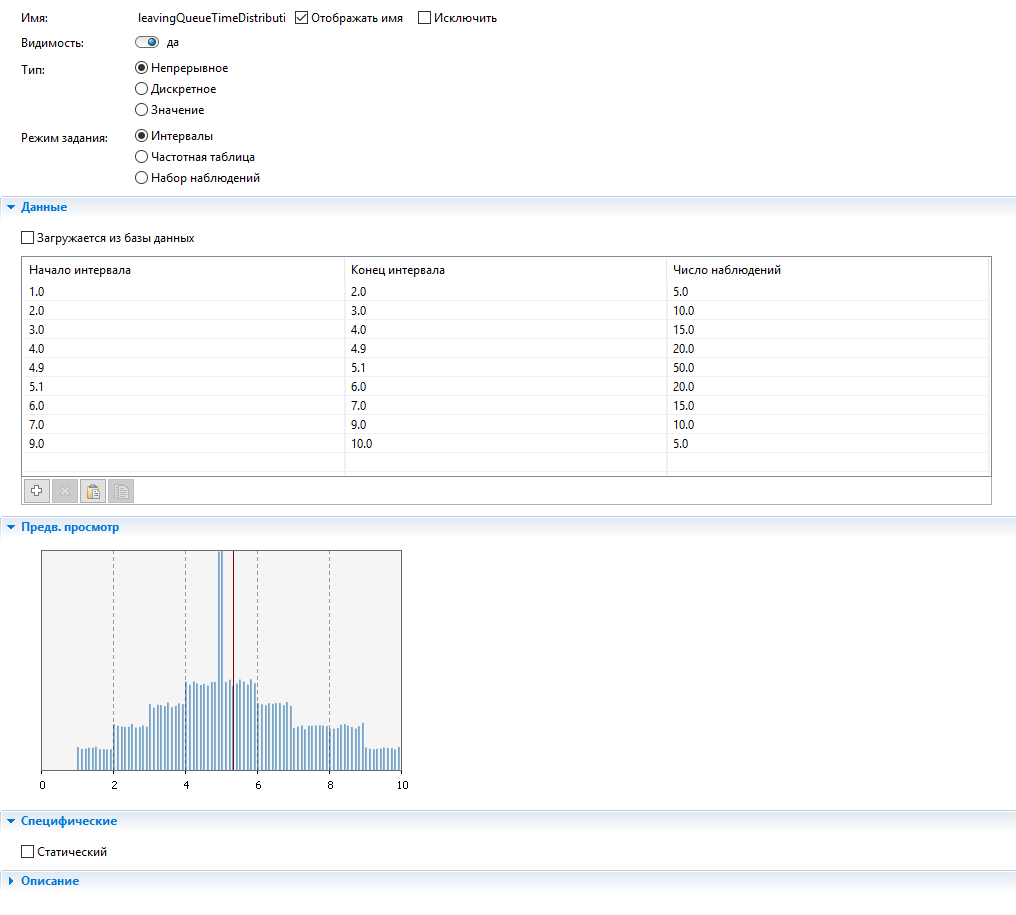
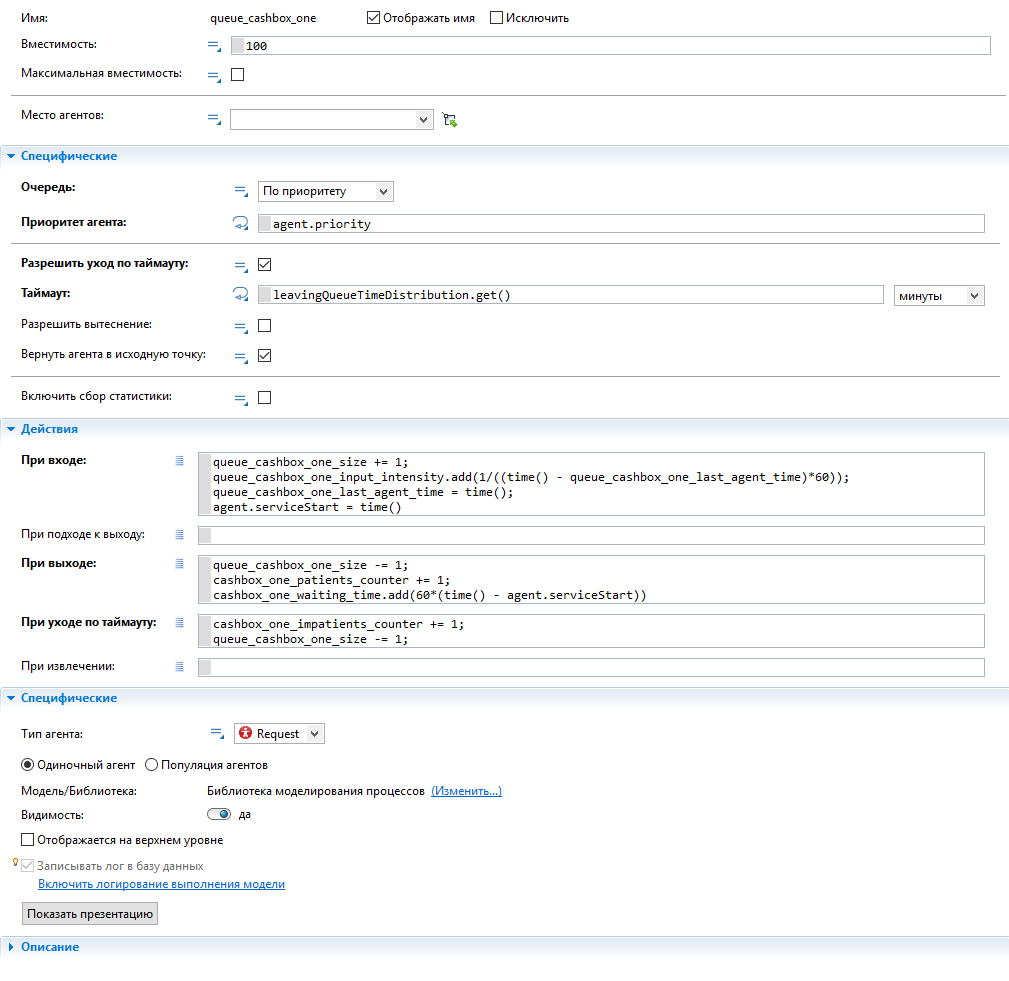
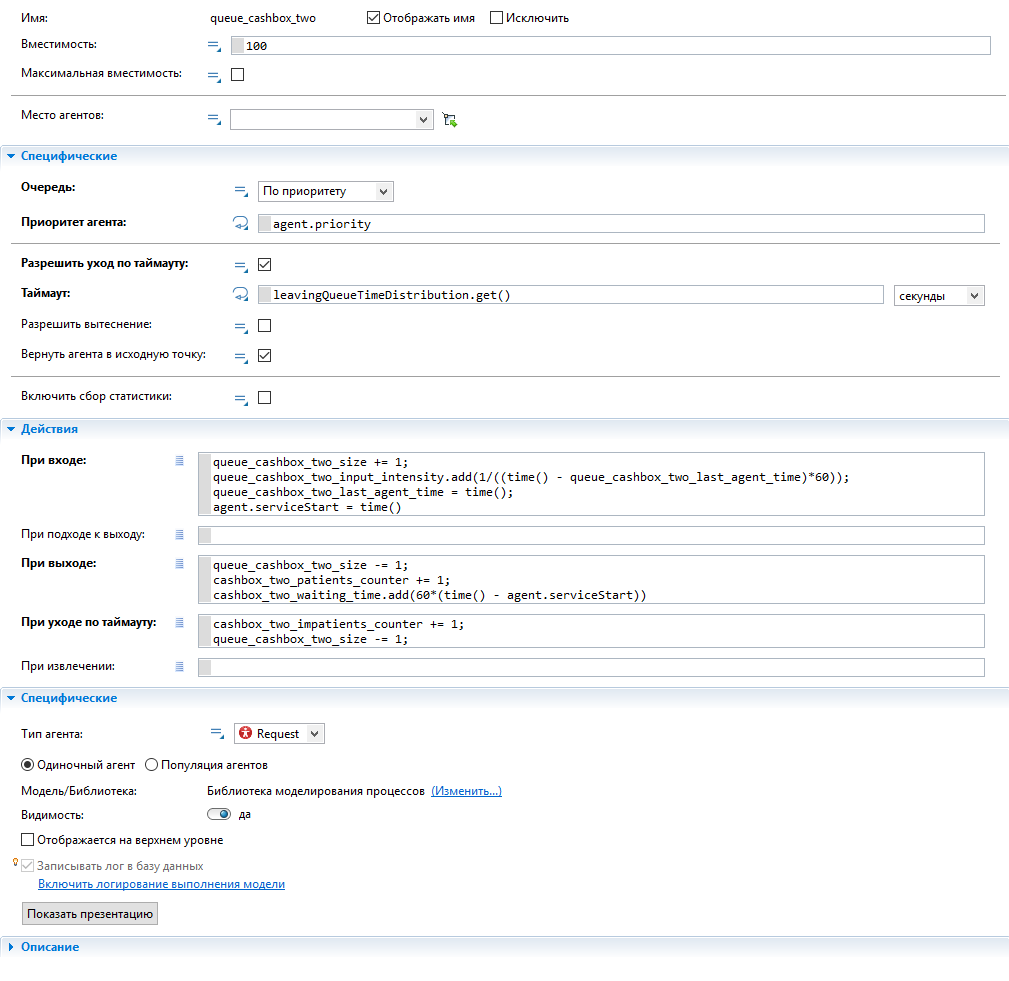
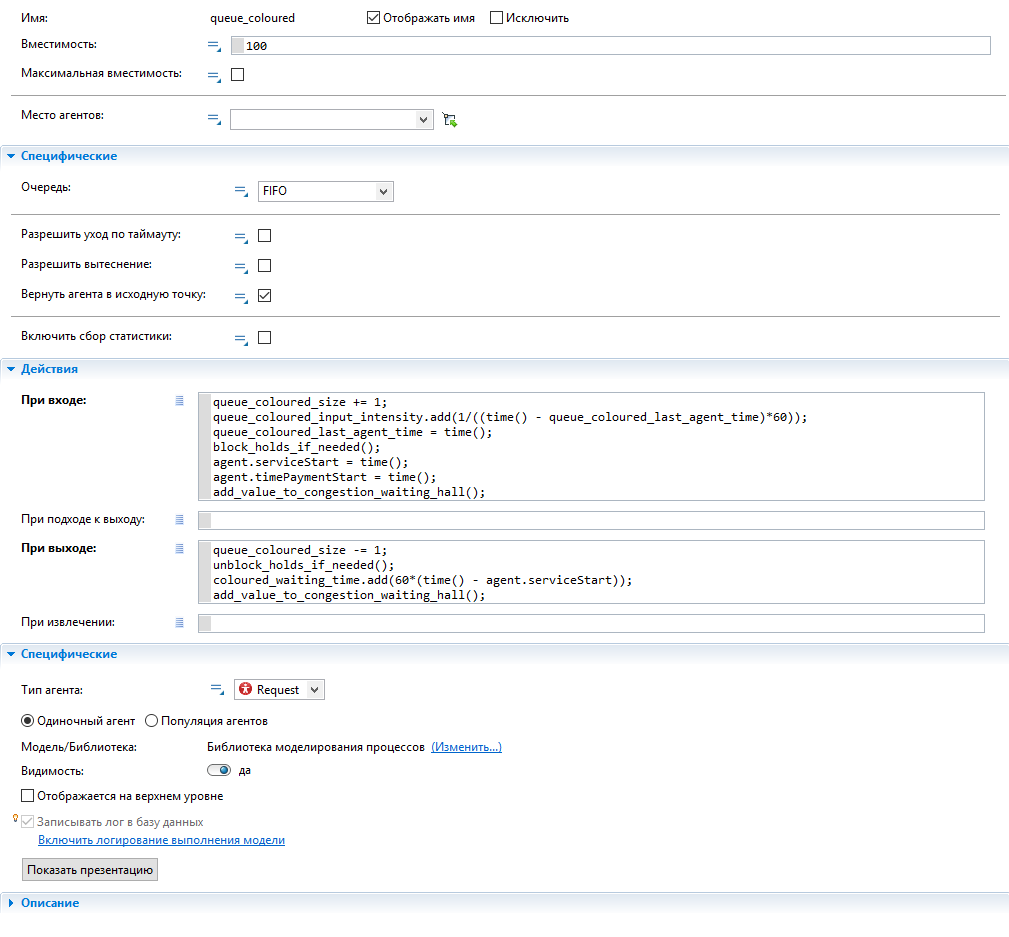
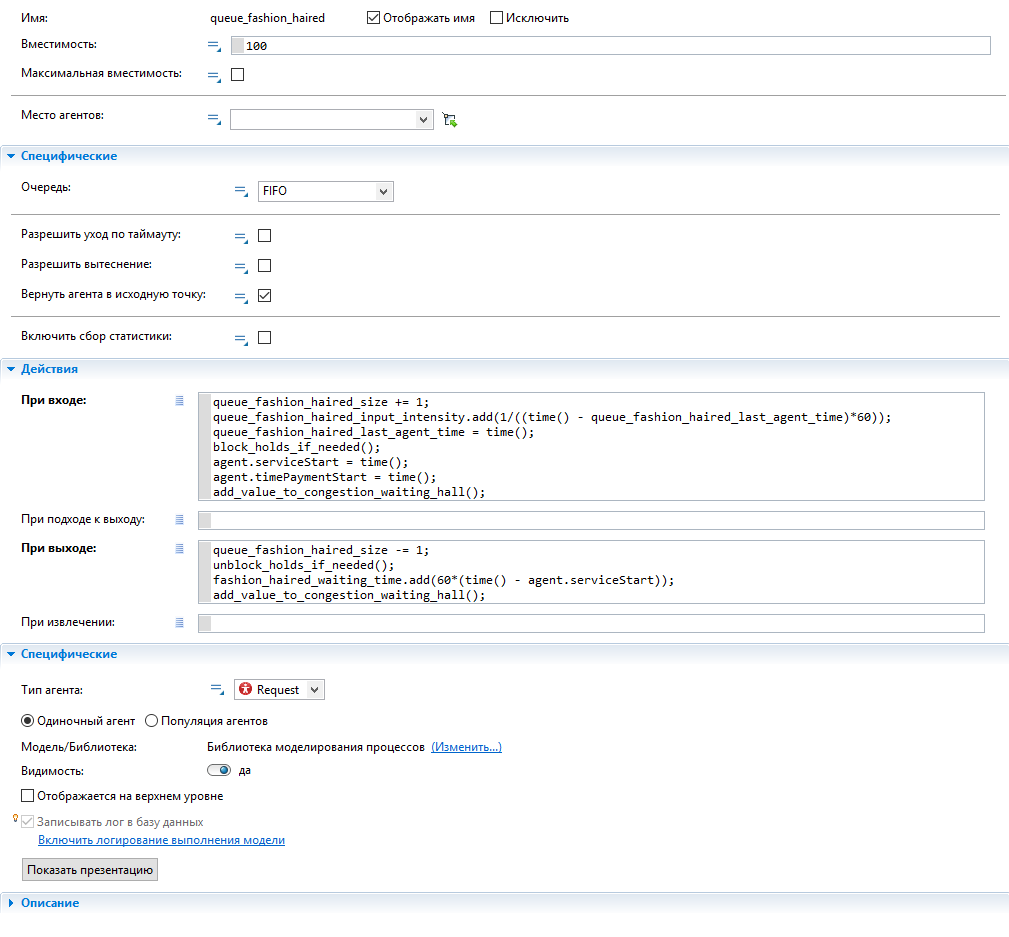
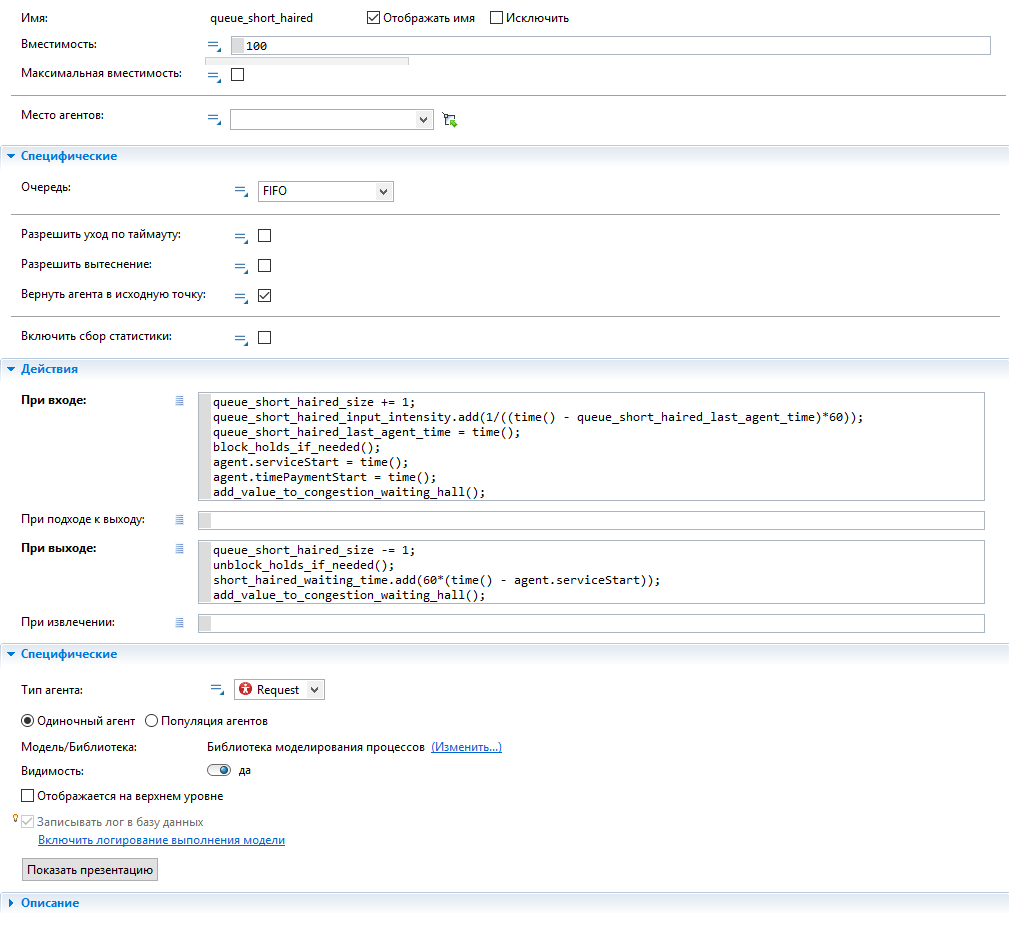
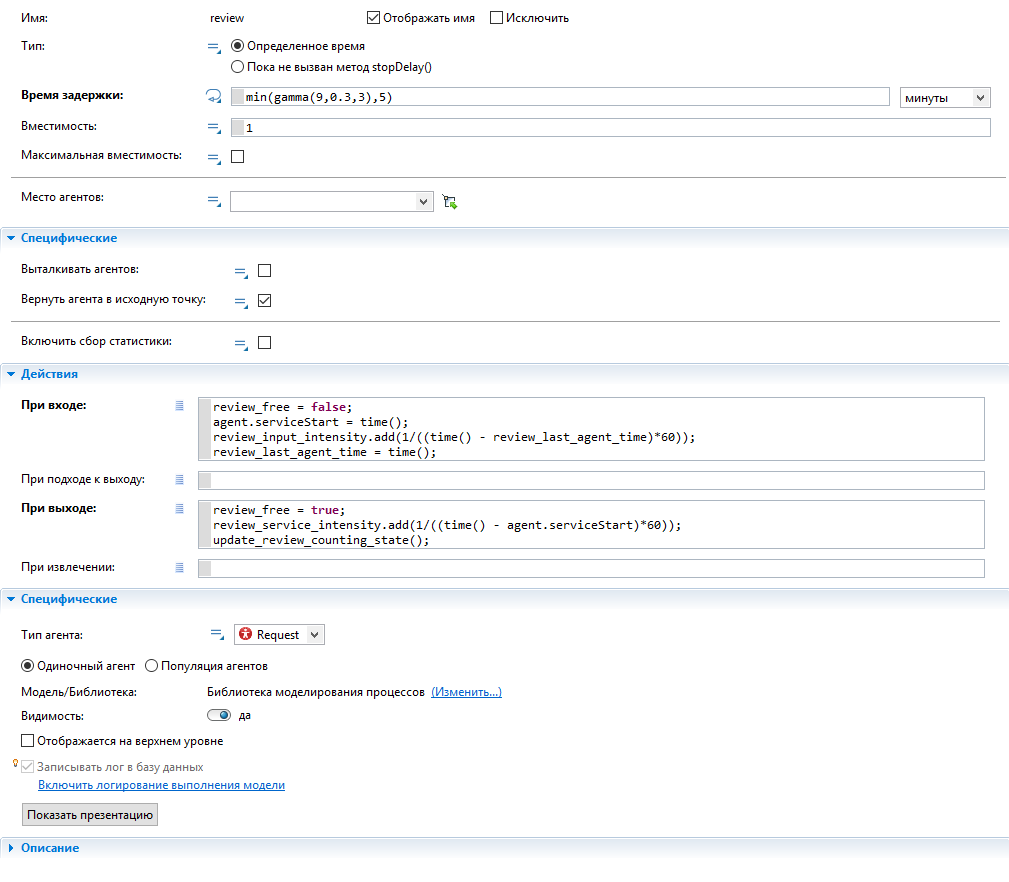
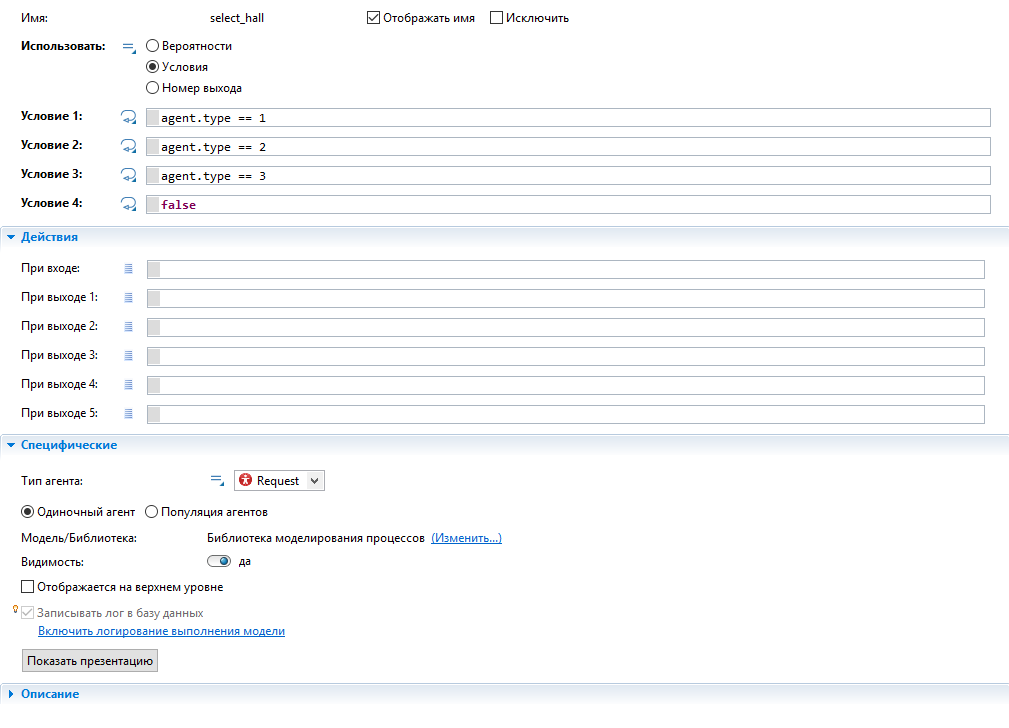
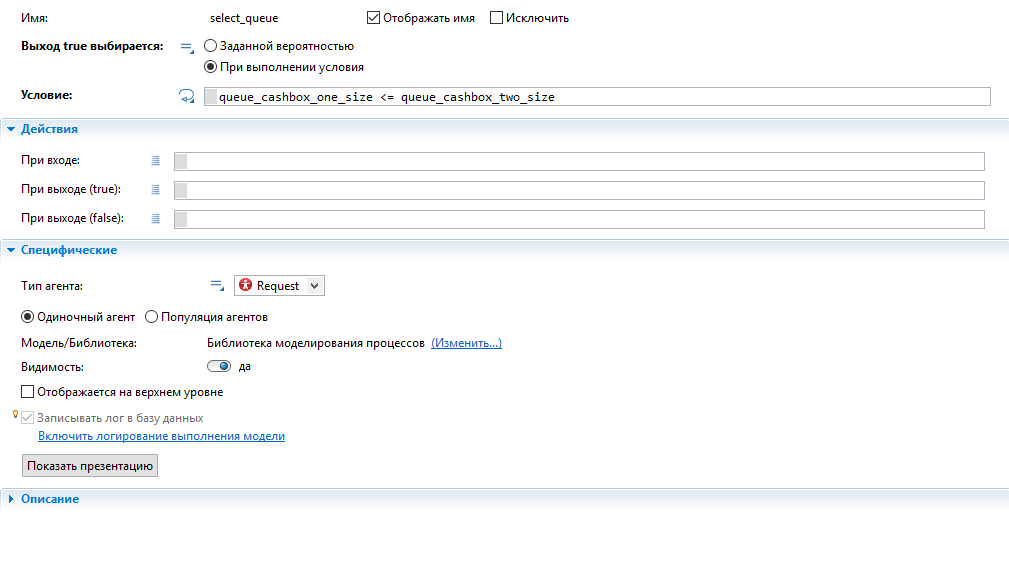
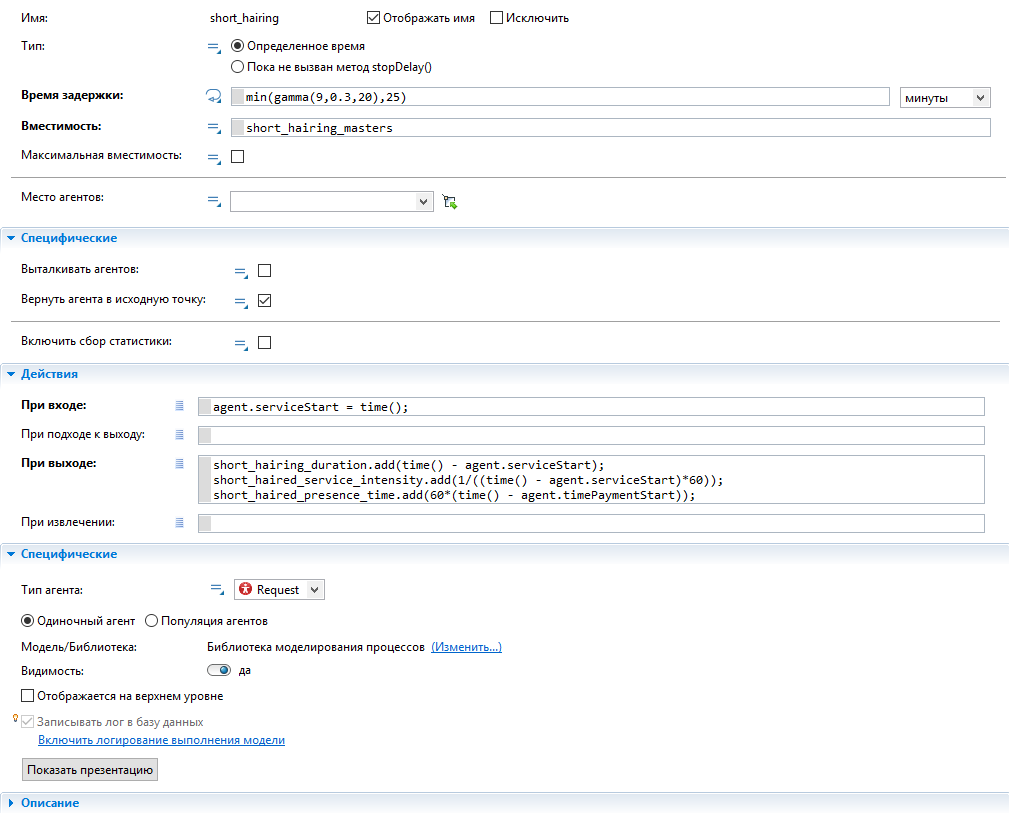
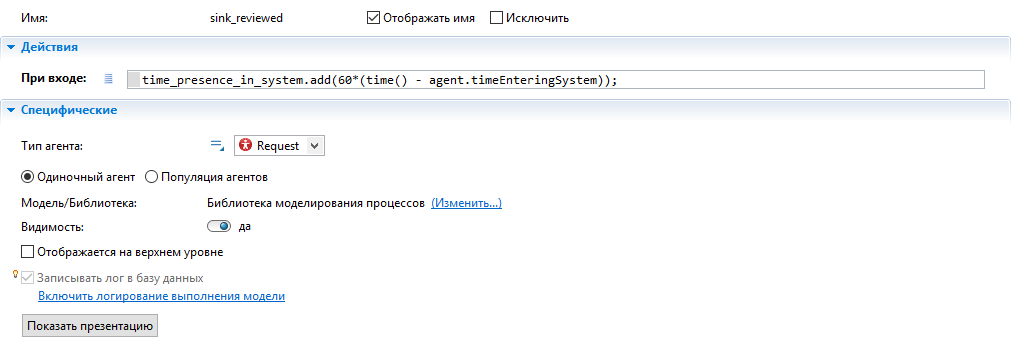
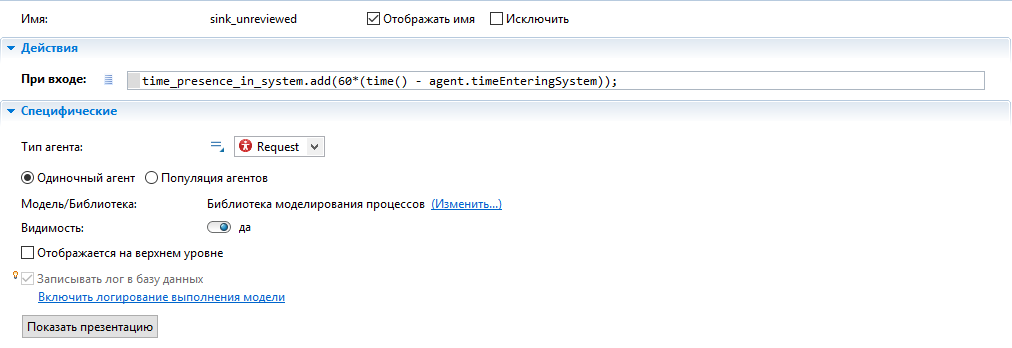
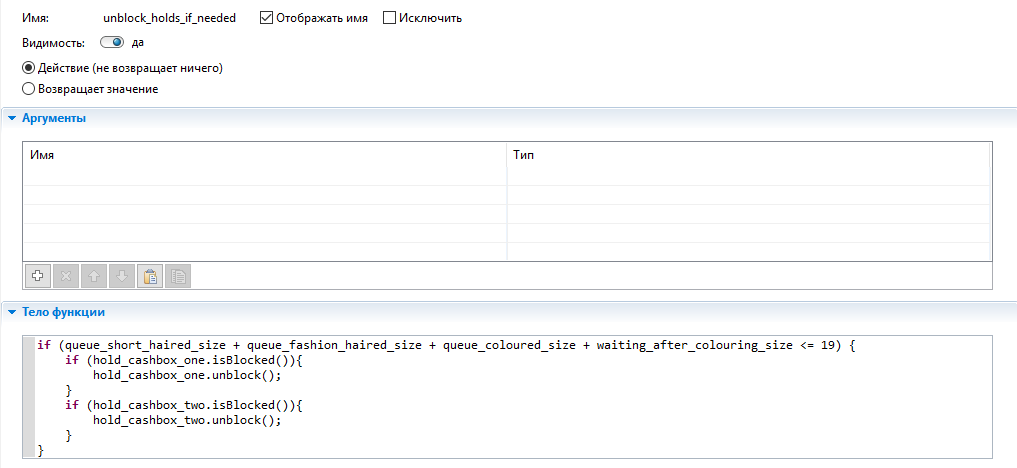
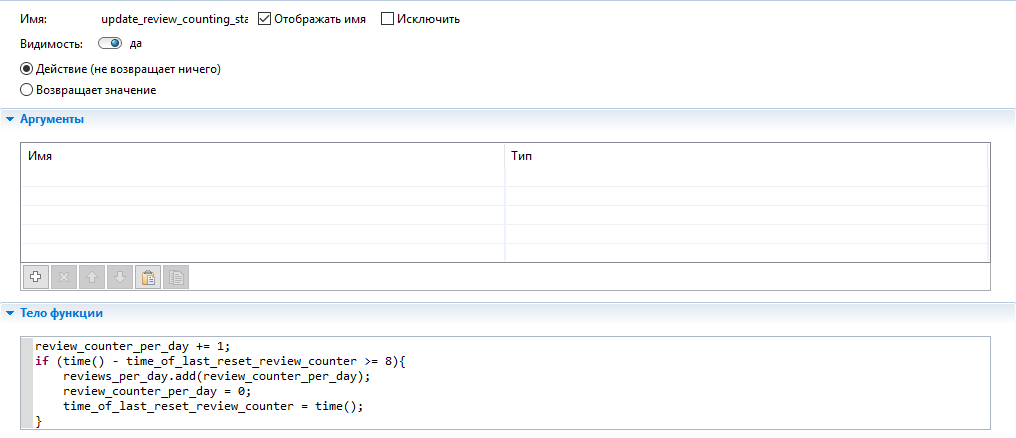
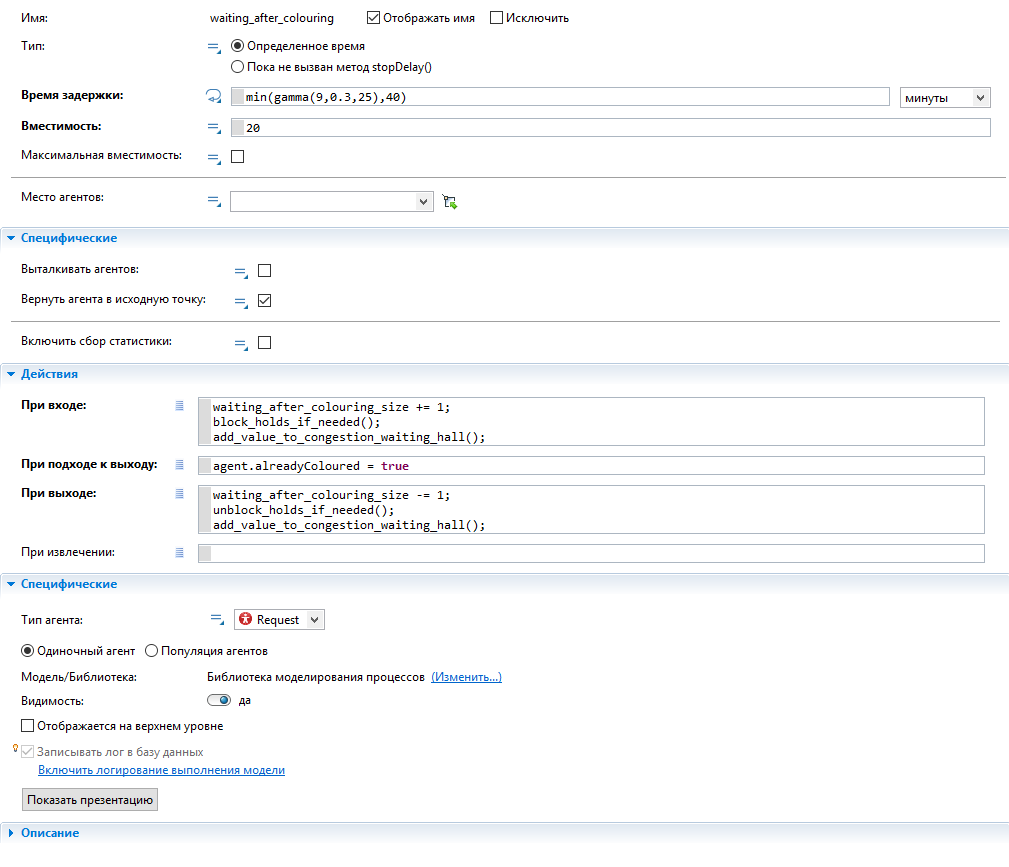
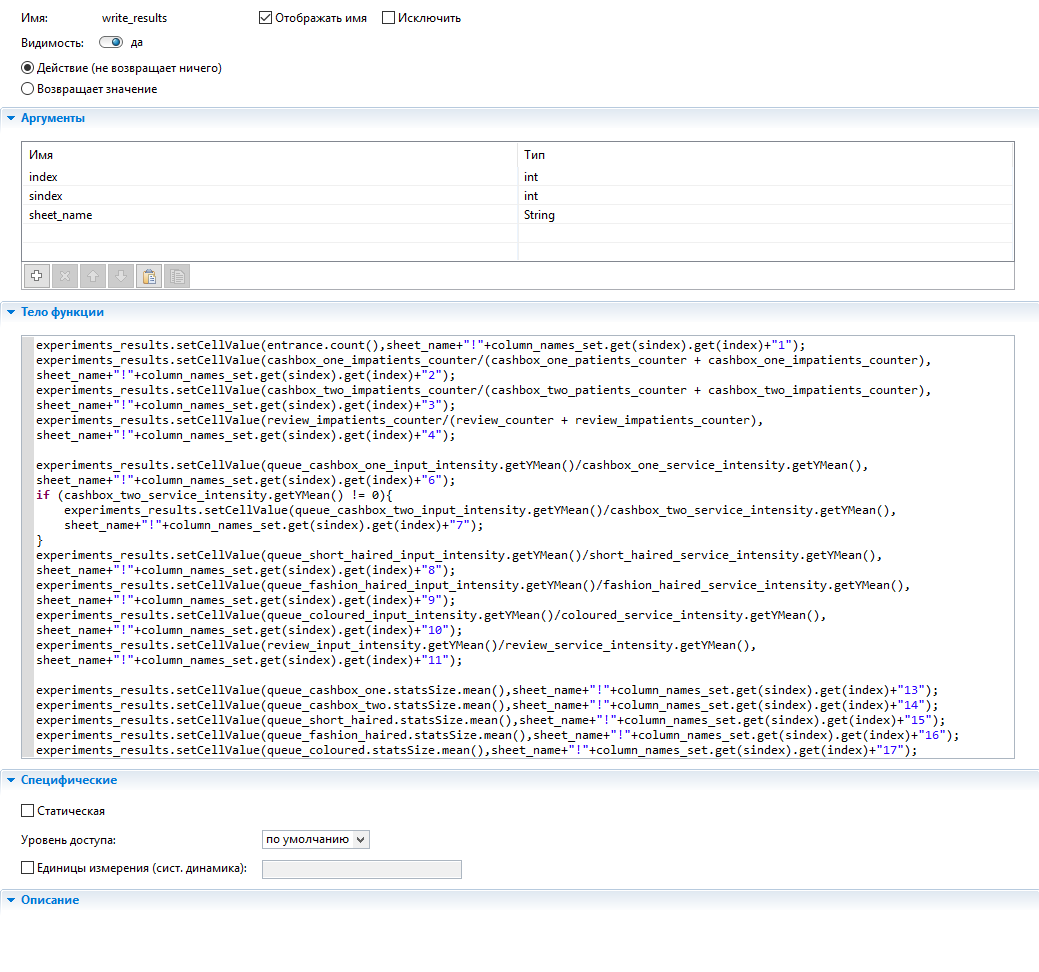
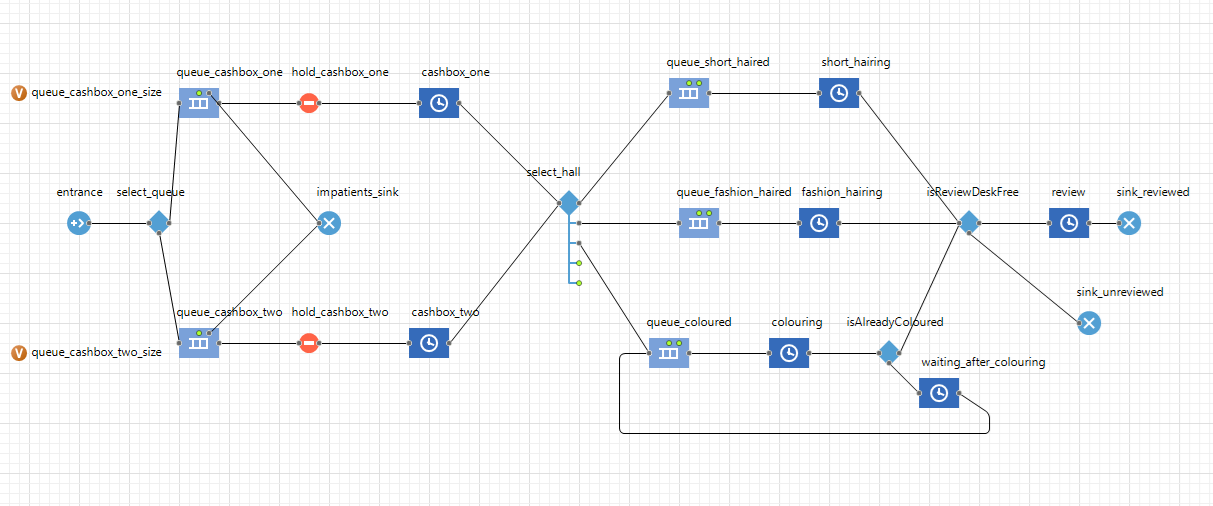
**Основные дополнительные упрощения и допущения**

1. Классы заявок кодируются в поле type типа данных Request целыми числами:
   1. Мужчины, пришедшие на стрижку под насадку – 1
   2. Женщины и мужчины, пришедшие на модельную стрижку – 2
   3. 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:
   1. Инвалиды и пенсионеры – 1
   2. 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 α = 3; β = 0.3.



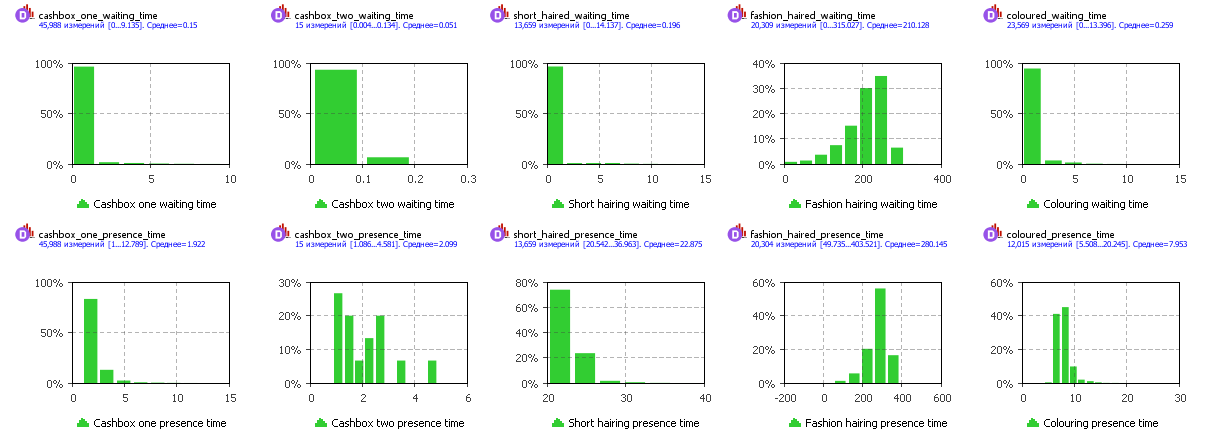
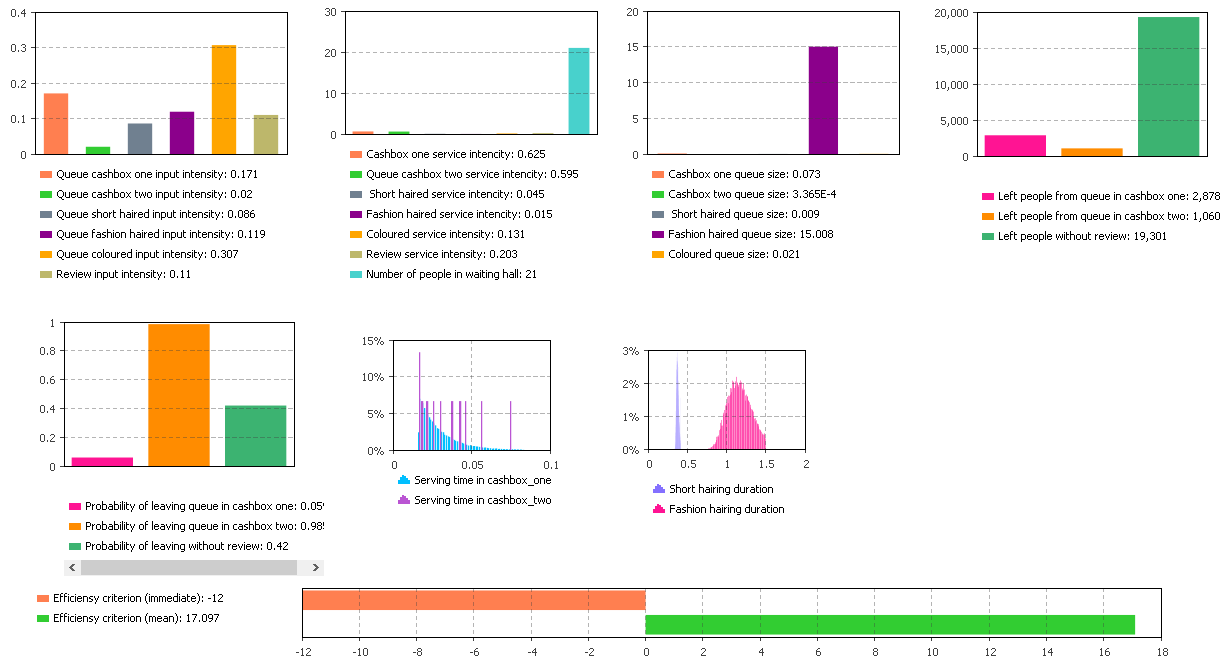
1. Будем исходить из предположения, что все времена обслуживания также распределены по гипер-экспоненциальному закону
2. Клиент в первую очередь пытается пройти на первую кассу, но, если во второй очереди оказывается меньше человек, он идет туда.
3. Будем исходить из предположения, что время обслуживания заявок в каждой из 2 касс имеет вид гамма – распределения с параметрами α = 1; β = 0.8, поскольку быстрое обслуживание клиентов значительно более вероятно, однако, редкие перерывы в работе кассира не невозможны.
4. 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.
5. 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**



**Сценарии работы модели**

**Обычный режим работы**

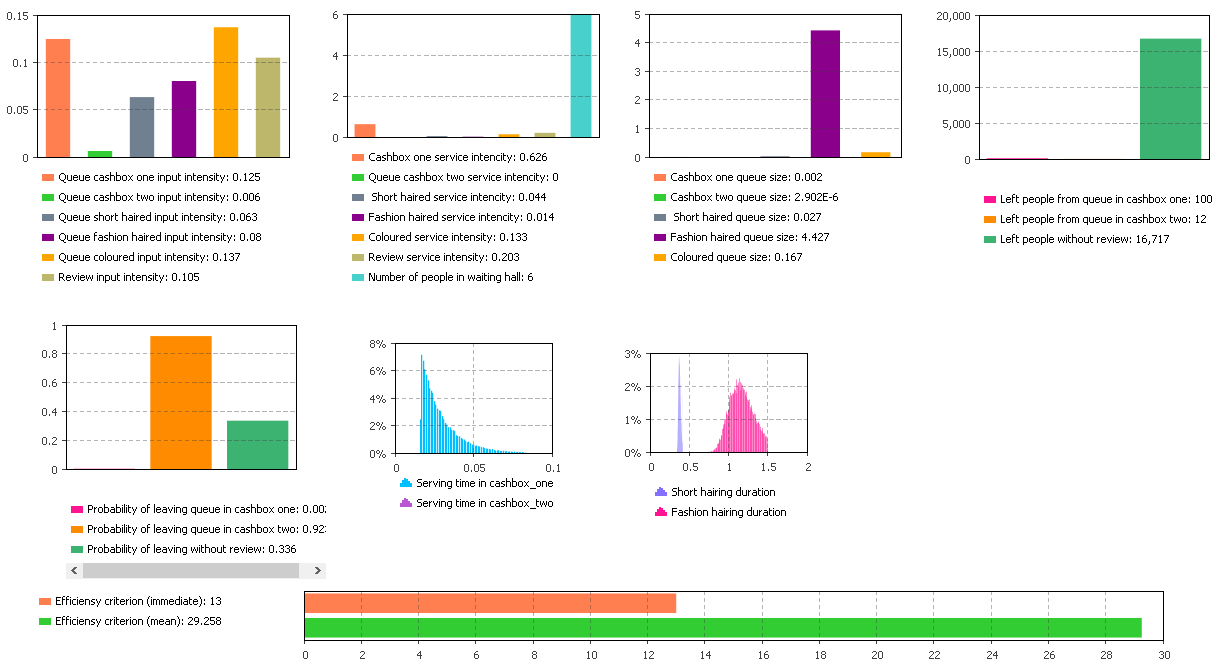
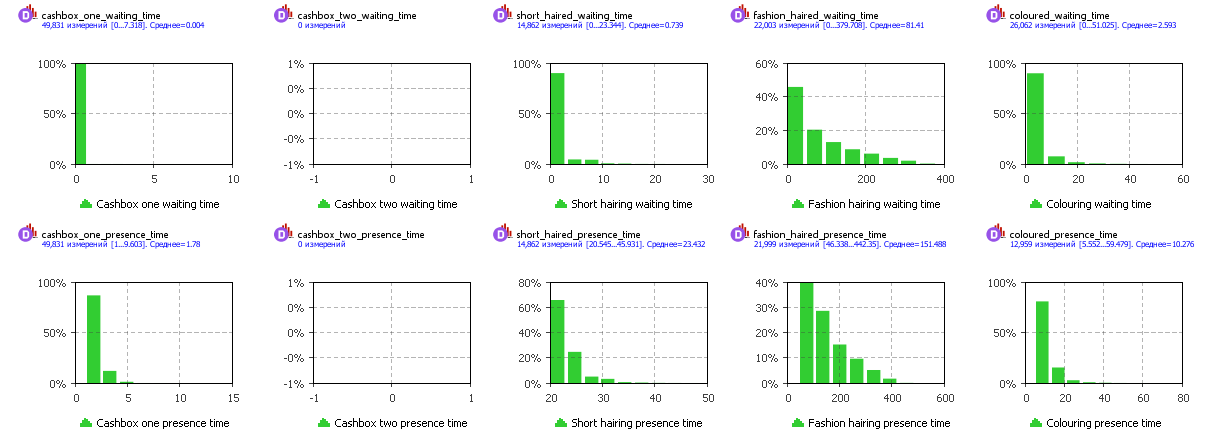


**Результаты подсчетов ширины доверительного интервала характеристик модели (в процентах)**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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.

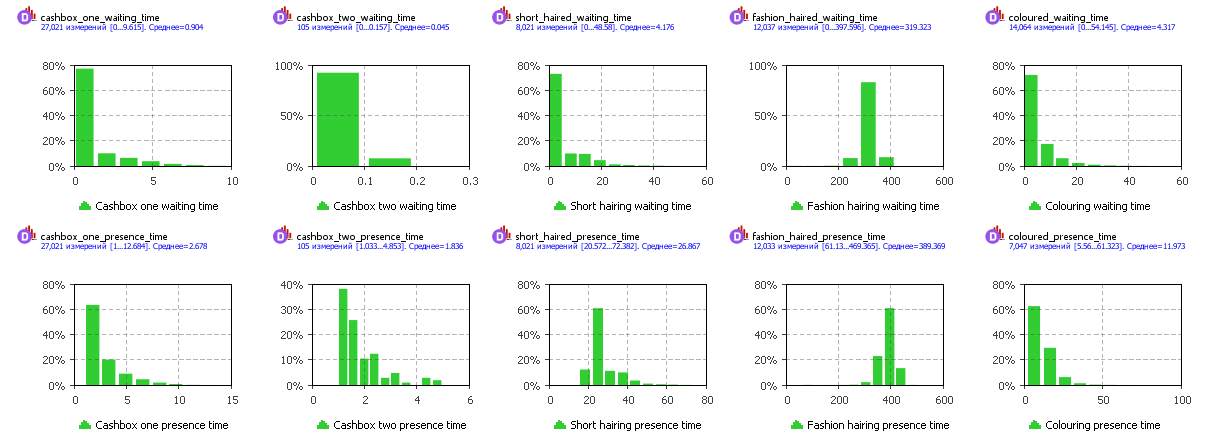
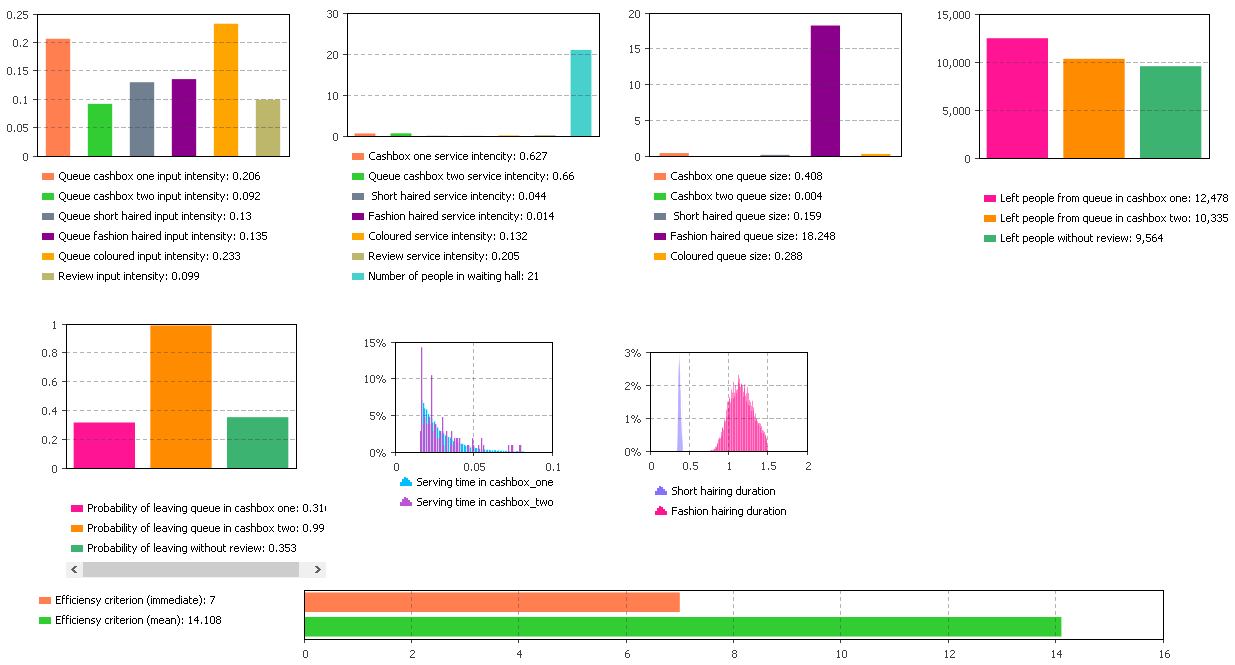


**Результаты подсчетов ширины доверительного интервала характеристик модели (в процентах)**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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%.

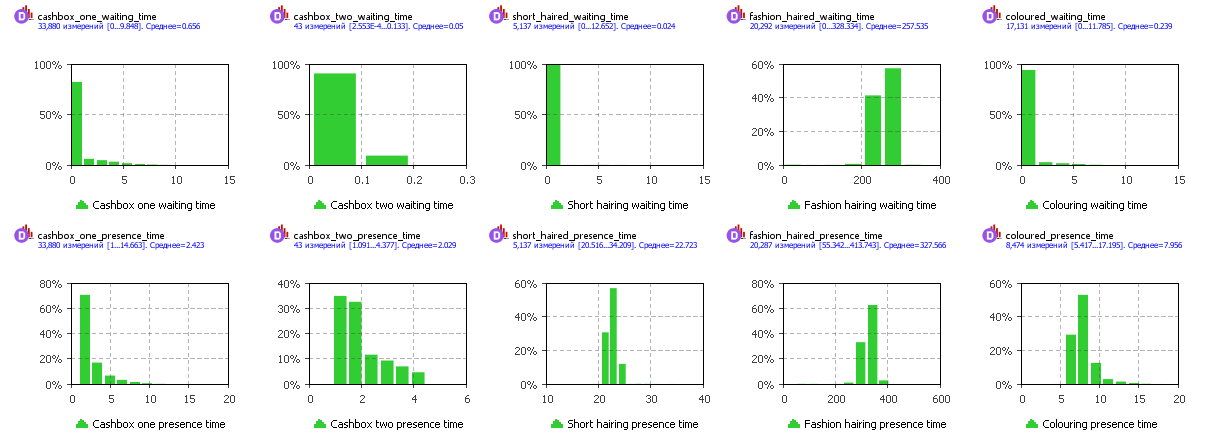
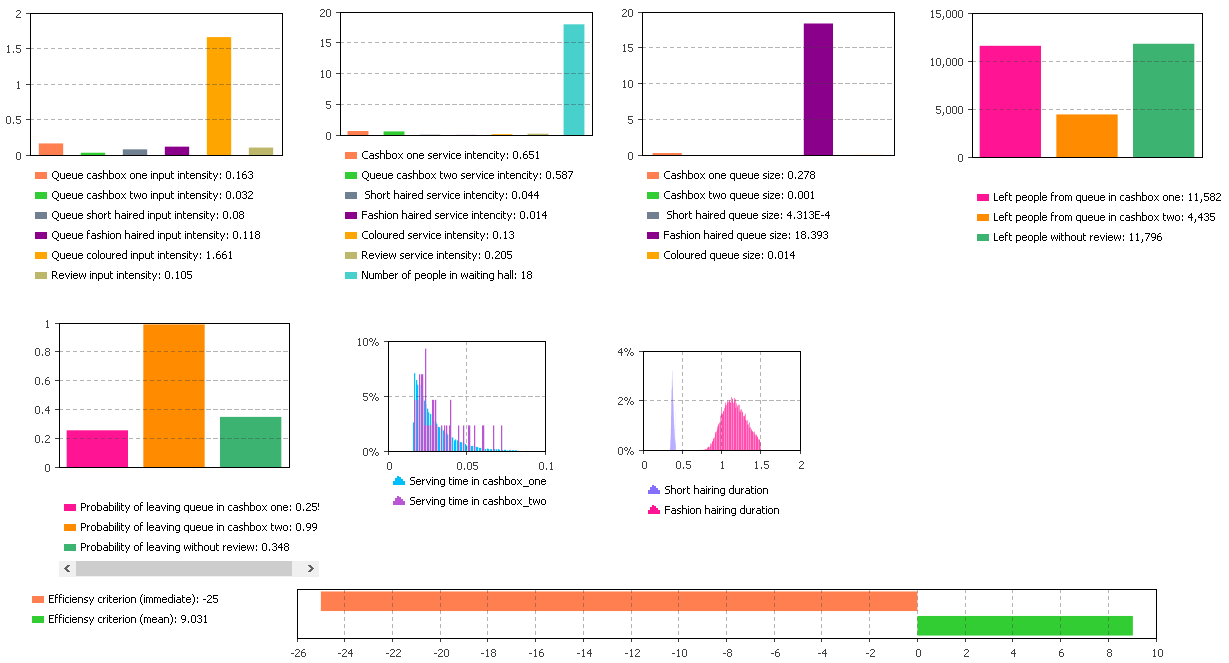


**Результаты подсчетов ширины доверительного интервала характеристик модели (в процентах)**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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:
   1. Система запускается 5 раз с количеством заявок, равным N, сохраняются полученные значения коэффициентов эффективности
   2. С использованием библиотек scipy и numpy рассчитывается математическое ожидание и величина доверительного интервала при помощи T-распределения Стьюдента
   3. 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)
   4. 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
4. **import** constants
5. **import** statistics
6. **import** entities
7. **from** scipy **import** stats
8. **import** math
9. **import** winsound
11. waiting\_hall\_fill = 0
13. blocked = False
15. *#rqs = [0,0]*
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
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
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())
46. env.process(c)
47. **yield** env.timeout(generators.get\_interval\_before\_new\_customer\_summer())


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()
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
71. **def** try\_print(message):
72. **if** constants.verbous:
73. **print**(message)
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
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
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()
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()
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
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)
112. **def** fix\_stop\_serving(resource, start\_serving\_time):
113. statistics.append\_service\_intensity\_component(resource, 1/(entities.env.now - start\_serving\_time))
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())
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**
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]))
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)
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()
173. try\_print('%7.4f %s successfully served' % (env.now, name))
174. statistics.serving\_times.append(env.now - starting\_serving\_timestamp)
175. **return**
177. **def** reset():
178. **global** waiting\_hall\_fill
179. **global** blocked
181. statistics.reset\_statistics()
182. waiting\_hall\_fill = 0
183. blocked = False
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))))
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)
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
202. **def** increase\_index(index, maximum):
203. index += 1
204. **if** index < maximum:
205. **return** index
206. **else**:
207. **return** 0
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):
230. prev\_prev\_accuracy = prev\_accuracy
231. prev\_accuracy = accuracy
233. common\_prev\_prev\_accuracy = common\_prev\_accuracy
234. common\_prev\_accuracy = common\_accuracy
235. criterias = []
236. **for** i **in** range(5):
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()
245. accuracy, mean, interval\_width = get\_reliability\_interval\_relative\_width(criterias)
246. *#print("-")*
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()

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")
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")
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)))
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)))
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)))
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)))
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)))
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")*

***Файл constants.py***

1. short\_hairing\_masters\_quantity = 3
2. fashion\_hairing\_masters\_quantity = 5
3. colouring\_masters\_quantity = 2
5. waiting\_hall\_max\_fullness = 20
7. short\_hairing\_client\_probability = 0.3
8. fashion\_hairing\_client\_probability = 0.45
10. priority\_client\_probability = 0.28
12. short\_hairing\_client\_class\_id = 1
13. fashion\_hairing\_client\_class\_id = 2
14. colouring\_client\_class\_id = 3
16. staff\_priority\_id = 0
17. important\_client\_priority\_id = 1
18. regular\_client\_priority\_id = 2
20. max\_blocking\_interval = 1000
22. verbous = True
23. statistics\_enable = True
25. number\_of\_clients = 500
26. step\_number\_of\_clients = 100
28. find\_optimal\_number\_of\_clients = False
29. student\_parameter = 0.05
31. minimal\_accuracy = 0.05
32. minimal\_stability = 0.01
33. number\_of\_considered\_means = 5
35. time\_of\_switching\_entrance = 100

***Файл entities.py***

1. **import** simpy
2. **import** constants
4. env = simpy.Environment()
6. *#devices*
7. unblock\_event = env.event()
8. cashbox\_one = simpy.PriorityResource(env, capacity=1)
9. cashbox\_two = simpy.PriorityResource(env, capacity=1)
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)
16. review\_desk = simpy.Resource(env, capacity=1)

***Файл generators.py***

1. **import** numpy
2. **import** constants
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
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
18. **def** get\_interval\_before\_new\_customer():
19. **return** gamma(35,0.002,0.08,0.13)\*60
21. **def** get\_interval\_before\_new\_customer\_epidemic():
22. **return** gamma(45,0.002,0.12,0.195)\*60
24. **def** get\_interval\_before\_new\_customer\_summer():
25. **return** gamma(30,0.002,0.056,0.091)\*60
27. **def** get\_writing\_review\_interval():
28. **return** gamma(4,0.7,3,5)
30. **def** get\_waiting\_after\_colouring\_interval():
31. **return** gamma(40,0.8,25,40)
33. **def** get\_service\_colouring\_interval():
34. **return** gamma(12,0.8,5,15)
36. **def** get\_service\_fashion\_hairing\_interval():
37. **return** gamma(12,5,40,90)
39. **def** get\_service\_short\_hairing\_interval():
40. **return** gamma(22,0.85,20,25)
42. **def** get\_service\_cashbox\_interval():
43. **return** gamma(20,0.3,1,5)
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
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
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
9. **global** lost
10. **global** lost\_reviews
12. **global** queue\_lengths
13. **global** waiting\_times
14. **global** queue\_length
16. **global** reviews\_per\_day\_set
18. **global** waiting\_hall\_fills
20. **global** reviews\_per\_day
22. **global** last\_time\_writing\_reviews
23. **global** presence\_times
24. **global** intensity\_components
25. **global** service\_intensity\_components
26. **global** figures\_counter
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
36. queue\_lengths = {}
37. waiting\_times = {}
38. queue\_length = {}
39. presence\_times = {}
40. intensity\_components = {}
41. service\_intensity\_components = {}
43. reviews\_per\_day\_set = []
45. waiting\_hall\_fills = []
47. reviews\_per\_day = 0
49. last\_time\_writing\_reviews = 0
51. figures\_counter = 0
52. cashbox\_queue\_lengths = [0,0]
53. serving\_times = []
54. cashbox\_queue\_length\_sets =[[],[]]
55. cashbox\_queue\_waiting\_times =[[],[]]
57. lost = 0
58. lost\_reviews = 0
60. last\_seen\_input\_time = {}
61. queue\_lengths = {}
62. waiting\_times = {}
63. queue\_length = {}
64. presence\_times = {}
65. intensity\_components = {}
66. service\_intensity\_components = {}
68. reviews\_per\_day\_set = []
70. waiting\_hall\_fills = []

73. reviews\_per\_day = 0
74. day\_length = 480
76. last\_time\_writing\_reviews = 0
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]
85. **def** set\_last\_seen\_input\_time(resource, value):
86. last\_seen\_input\_time[resource] = value
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]
95. **def** increase\_queue\_length(resource):
96. **try**:
97. queue\_length[resource] += 1
98. **except**:
99. queue\_length[resource] = 1
101. **def** decrease\_queue\_length(resource):
102. **try**:
103. queue\_length[resource] -= 1
104. **except**:
105. queue\_length[resource] = 0
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])
112. **def** append\_presence\_time(resource, time):
113. **if** (time == 0):
114. **return**
115. append\_value\_to\_collection(resource, presence\_times, time)
117. **def** get\_presence\_times(resource):
118. **return** get\_values\_from\_collection(resource, presence\_times)
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)
125. **def** get\_intensity\_components(resource):
126. **return** get\_values\_from\_collection(resource, intensity\_components)
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)
133. **def** get\_service\_intensity\_components(resource):
134. **return** get\_values\_from\_collection(resource, service\_intensity\_components)
136. **def** get\_queue\_lengths(resource):
137. **return** get\_values\_from\_collection(resource, queue\_lengths)
139. **def** append\_waiting\_time(resource, time):
140. **if** (time == 0):
141. **return**
142. append\_value\_to\_collection(resource, waiting\_times, time)
144. **def** get\_waiting\_times(resource):
145. **return** get\_values\_from\_collection(resource, waiting\_times)
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)
154. **def** get\_values\_from\_collection(resource, collection):
155. **try**:
156. **return** collection[resource]
157. **except**:
158. **return** [0]
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()
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()
180. **def** increase\_lost\_quantity():
181. **global** lost
182. lost += 1
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

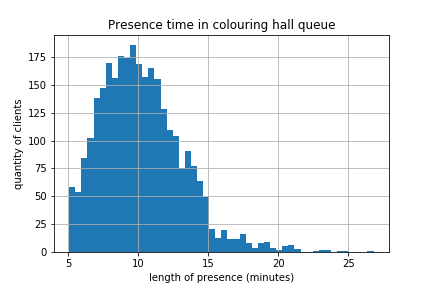
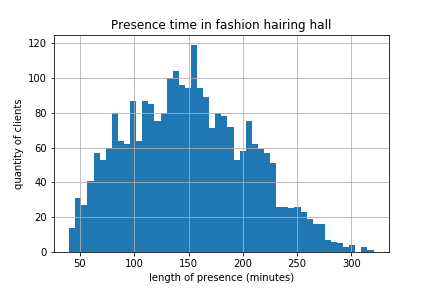
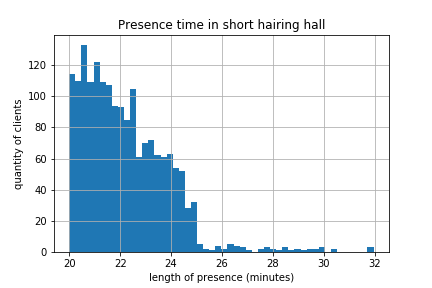
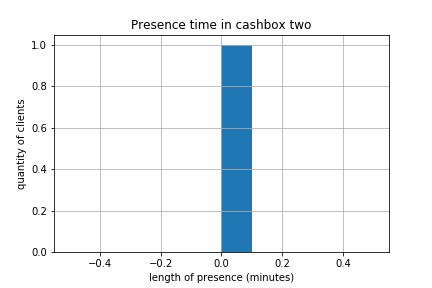
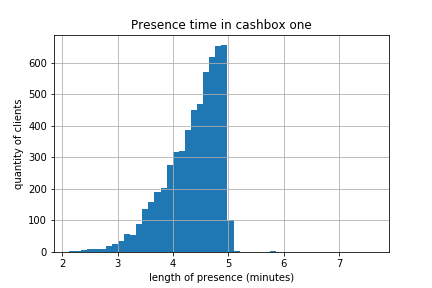
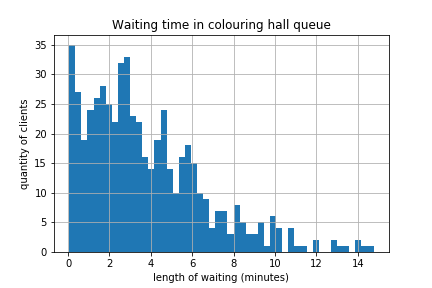
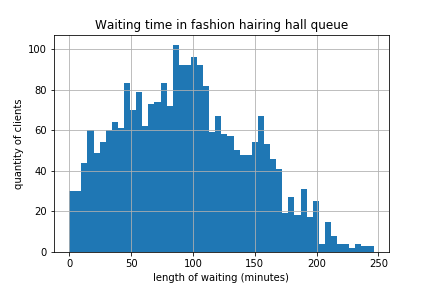
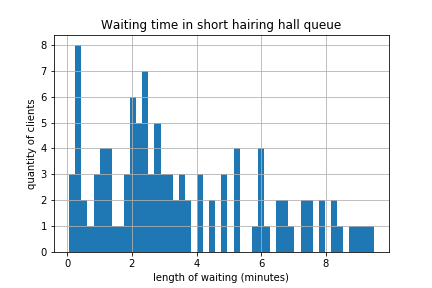
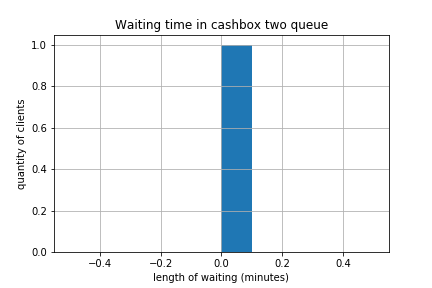
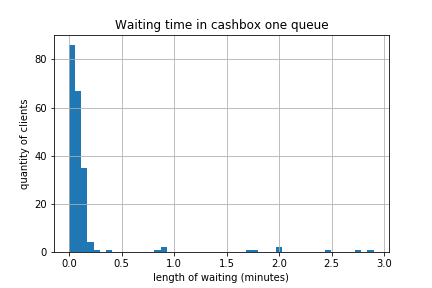
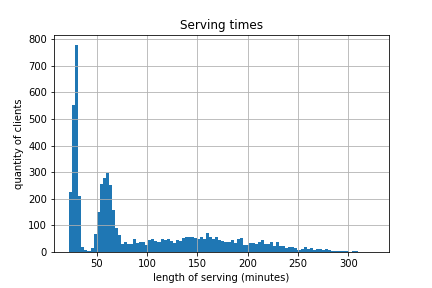
6200 | 3.7946 | 34.0274 ± 1.2912

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

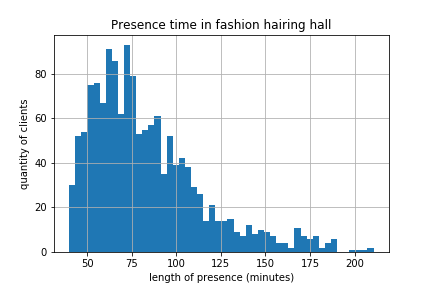
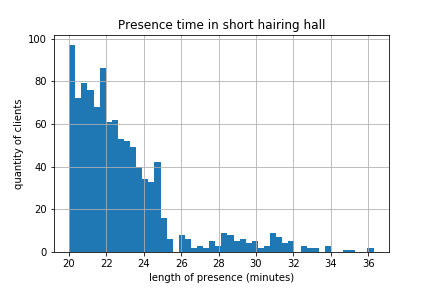
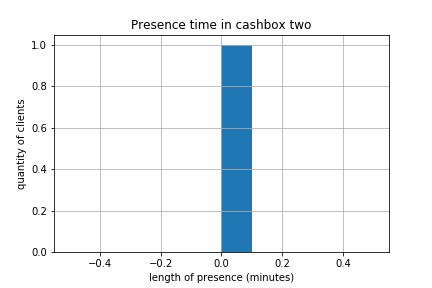
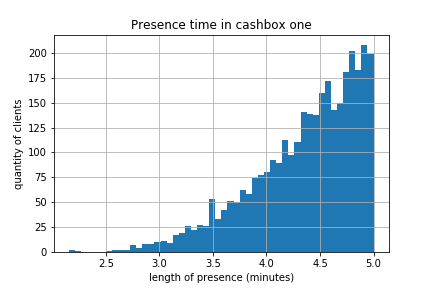
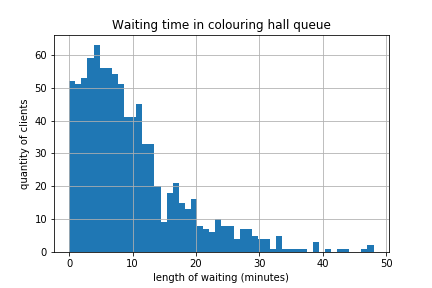
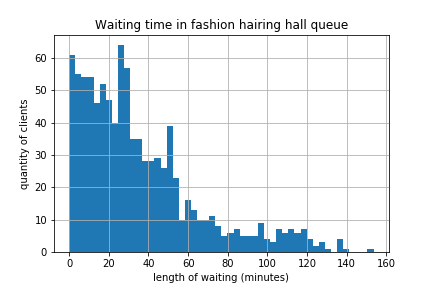
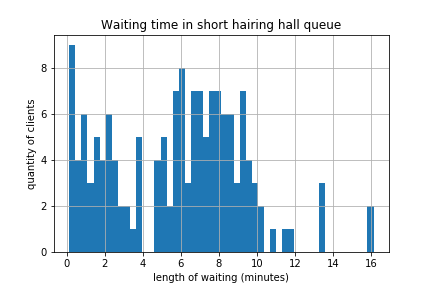
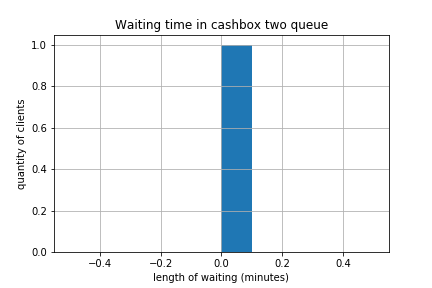
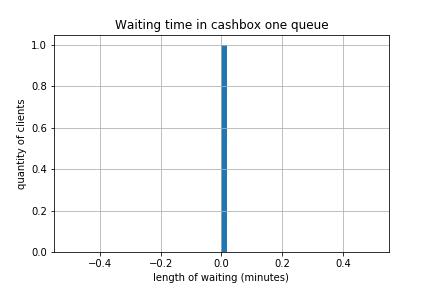
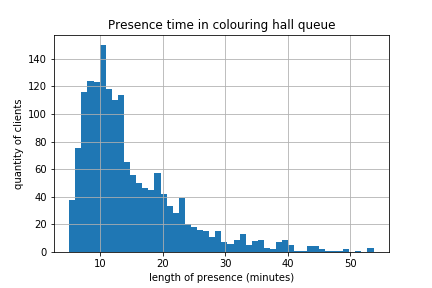
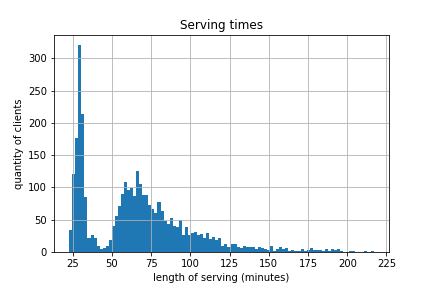
3300 | 3.4134 | 34.0194 ± 1.1612

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

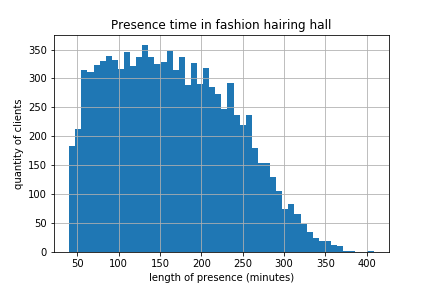
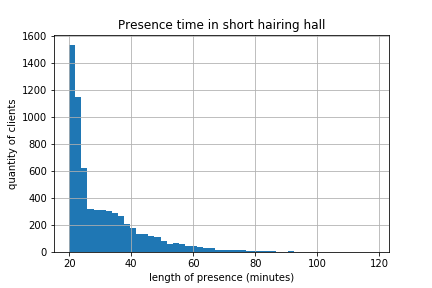
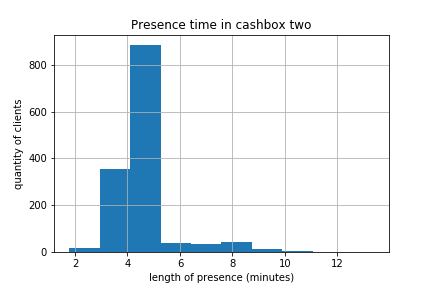
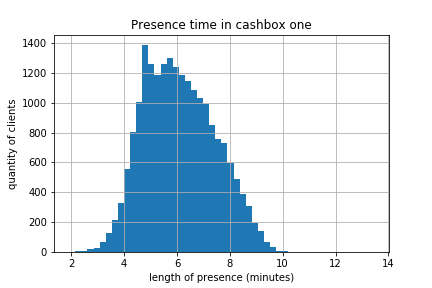
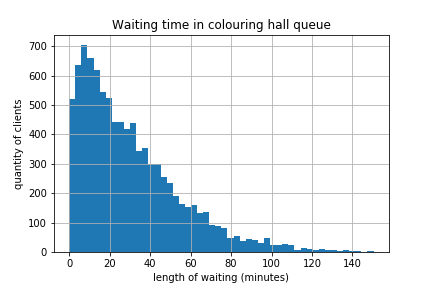
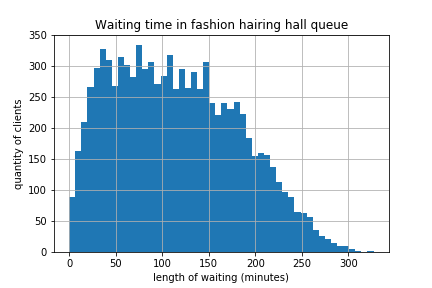
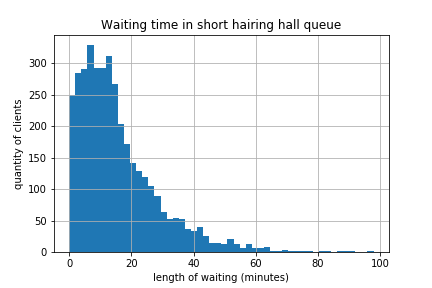
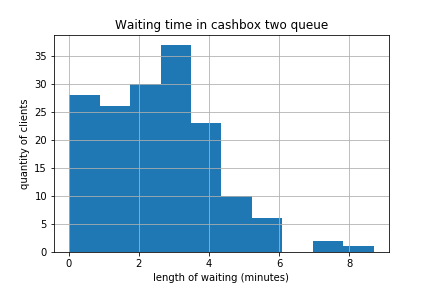
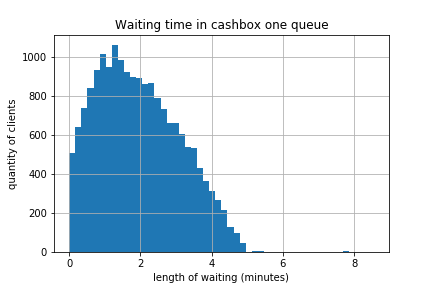
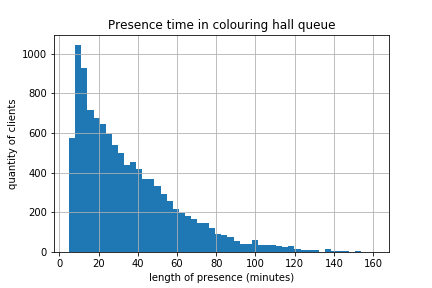
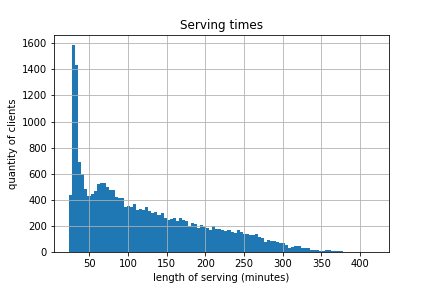
62750 | 4.9001 | 17.8839 ± 0.8763

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

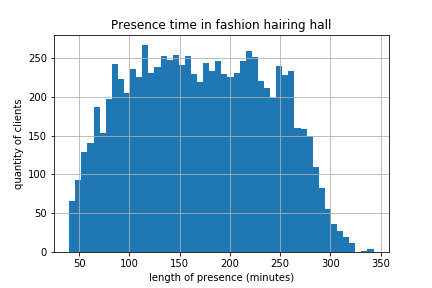
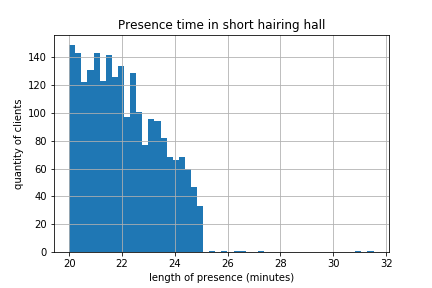
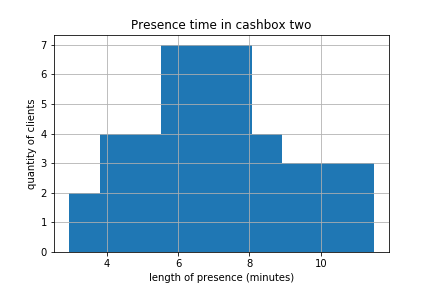
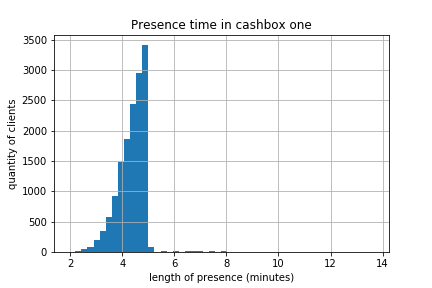
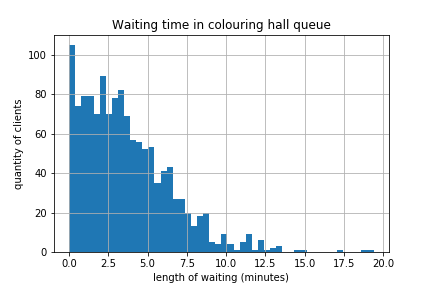
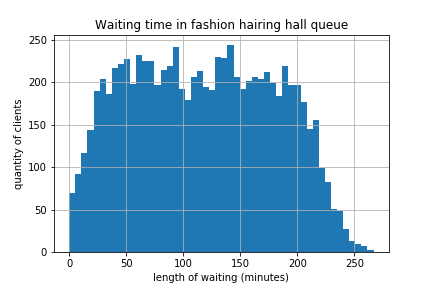
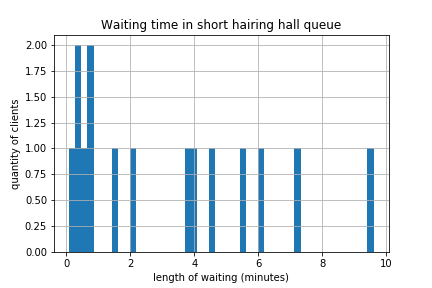
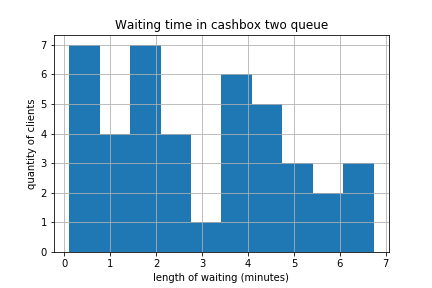
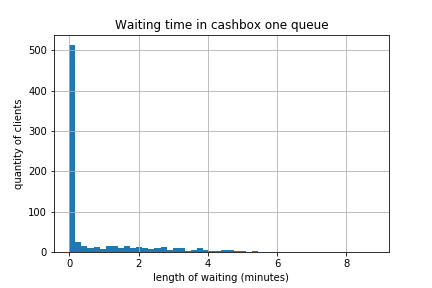
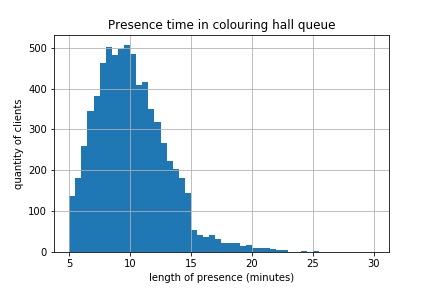
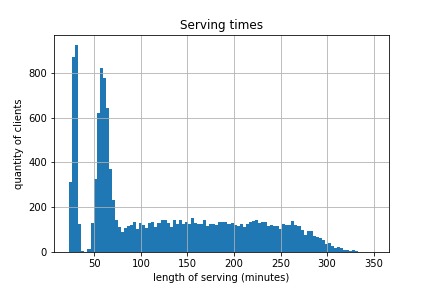
25100 | 3.3734 | 18.7770 ± 0.6334

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** |