ISSN 2658-5014 (Print) СИИТ ISSN 2686-7044 (Online)
http://siit.ugatu.su 2027. Т. 9, № 1(30)
СИСТЕМНАЯ ИНЖЕНЕРИЯ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ
Научная статья

УДК 004.8 DOI: 10.54708/SIIT-2027-no1-p18 EDN: CDMGPR

Моделирование системы обработки запросов в гибридном контакт-центре банка

И. А. Бурханов • А. Ф. Валеева • Р. С. Валеев

Уфимский университет науки и технологий

Статья посвящена моделированию процесса обработки входящих запросов в гибридном контакт-центре банка, включающем трехуровневую систему обслуживания клиентов, состоящую из чат-ботов, операторов первой и второй линий поддержки. Поступление запросов описывается неоднородным пуассоновским процессом. Разработан алгоритм, моделирующий приход запросов на различные уровни с учетом нерешенных клиентских запросов. При этом вычисляются показатели, позволяющие оценить эффективность работы контакт-центра, а именно: среднее время пребывания запроса в системе, время ожидания в очередях, коэффициенты загрузки операторов и доли решенных запросов на каждом уровне. Предложенный алгоритм позволяет руководству банка оптимизировать работу контакт-центра без проведения экспериментов в реальном времени.

Трехуровневая система обслуживания клиентов; система массового обслуживания с устройствами; соединенными параллельно; неоднородный пуассоновский процесс; дискретно-событийная имитационная модель.

Введение

Современные банки обрабатывают большое количество клиентских запросов, поступающих через различные каналы связи. Для повышения качества обслуживания и снижения нагрузки на операторов широко применяются гибридные контакт-центры, в которых процесс обработки запросов представляет собой трехуровневую схему принятия решения. Так первичная обработка запросов выполняется чат-ботами (1 уровень), сложные вопросы решаются операторами первой и второй линий поддержки (соответственно 2-й и 3-й уровни). При этом необходимо решить вопросы о количестве операторов, производительности чат-ботов, распределении нагрузки между операторами. Подобные задачи исследовались в [Law19, Bei19, Fak24, Cha22a, Cha22b, Ani21, Ros23], для их решения применялось дискретно-событийное имитационное моделирование.

На рынке программного обеспечения (ПО) предлагаются как коммерческие пакеты (AnyLogic [Bor21], Simio [Smi24], Arena [Kel23]), так и пакеты с открытым исходным кодом (SimPy [Sim26], Salabim [Van23, Van18]). Проведенный анализ перечисленного ПО показал:

- для необходимой в рассматриваемой задаче реализации сменного графика операторов с отложенным удалением занятых сотрудников, в AnyLogic и SimPy требуется написания большого низкоуровнего кода (Java для AnyLogic, пользовательских классов для SimPy) с риском допущения ошибок синхронизации. При этом в разработанном алгоритме эта проблема решается прозрачно с помощью флага и функции отложенного удаления;
- в AnyLogic и SimPy отсутствует функция для генерации неоднородного пуассоновского процесса, поддерживается лишь функция генерации однородного пуассоновского процесса. Реализация неоднородного пуассоновского процесса в перечисленных выше пакетах

Бурханов И. А., Валеева А. Ф., Валеев Р. С. Моделирование системы обработки запросов в гибридном контакт-центре банка // СИИТ. 2027. Т. 9, № 1(30). С. 18-30. DOI: 10.54708/SIIT-2027-no1-p18. EDN: CDMGPR.

Burhanov I. A., Valeeva A. F., Valeev R.S. “The modeling of the query processing system in a bank's hybrid contact center” // SIIT. 2027. Vol. 9, no. 1(30), pp. 18-30. DOI: 10.54708/SIIT-2027-no1-p18. EDN: CDMGPR. (In Russian).

приводит к существенному росту времени моделирования, большая часть которого связана с реализацией виртуальной машины Java и с визуализацией, даже при отключении последней;

- для расчета оценок эффективности работы гибридного контакт-центра банка необходима детальная информация о каждом запросе клиента. В связи с этим в SimPy, например, возникла бы необходимость в создании собственных контейнеров данных, что привело бы к реализации «с нуля» Фреймворка с дополнительными накладными расходами на вызовы методов SimPy;
- готовая библиотека как в коммерческих пакетах, так и в пакетах с открытым кодом, представляют собой «чёрные ящики», предложенная же реализация решения задачи с помощью стандартных модулей Python (heapq, random, math) позволяет контролировать каждый шаг алгоритма, исключает скрытые допущения, упрощает верификацию и может быть адаптирована к любой среде без установки дополнительных пакетов.

В статье предлагается дискретно-событийная имитационная модель системы обработки запросов в гибридном контакт-центре банка; алгоритм, базирующийся на этой модели и позволяющий проанализировать эффективность работы системы с неоднородным потоком запросов клиентов с одного уровня принятия решений на другой с некоторой вероятностью, реализованный стандартными средствами языка Python.

Постановка задачи и имитационный алгоритм обработки запросов

Гибридный контакт-центр банка занимается обработкой запросов клиентов, в котором процесс обработки запросов состоит из трех уровней. На первом уровне запрос от клиента поступает на обработку чат-боту. Если ответ на запрос не удовлетворил клиента, то запрос передается на второй уровень оператору первой линии поддержки для обработки нерешенных запросов. В противном случае, если запрос остался нерешенным, он передается на третий уровень оператору второй линии поддержки. Запросы поступают операторам первой и второй линии поддержки по одному в случайные моменты времени, дисциплина их обслуживания FIFO («первый пришел, первый ушел»), время обслуживания запросов случайная переменная, распределенная по экспоненциальному закону распределения. Интенсивность λ поступления запросов в систему в течение рабочего дня составляет 1.8 запросов в минуту. На рис. 1 изображена логическая схема обработки запросов клиентов. Требуется исследовать работу гибридного контакт-центра в течение некоторого заранее заданного промежутка времени T с целью адаптации работы операторов 1-й и 2-й линий под реальную нагрузку.

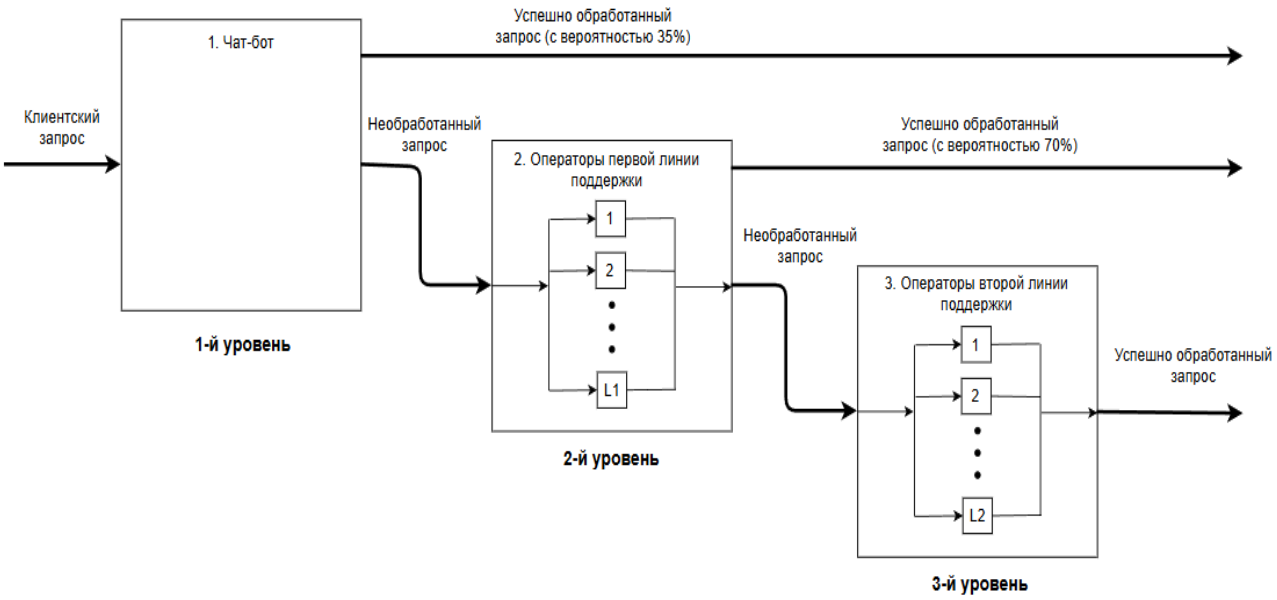


Рис. 1. Логическая схема обработки запроса клиентов

Рассматриваемая задача обработки клиентских запросов представляет собой систему массового обслуживания с устройствами, соединенными параллельно, с последовательной обработкой запросов в системе из трех уровней, очередями с дисциплиной FIFO и переводом запроса между уровнями с некоторой вероятностью, с ожиданием и неограниченной очередью и устройствами (операторами). Для моделирования процесса обработки запросов используется дискретно-событийная имитационная модель, в которой события – это поступления запросов к чат-ботам и их обработка. При этом моделью процесса поступления запросов на первый уровень является неоднородный пуассоновский процесс с экспоненциальным временем обслуживания запросов операторами и с функцией интенсивности $\lambda(t)$, $t \in [t_0, T]$, t_0 – время начала процесса моделирования, T – время конца моделирования. Функция интенсивности $\lambda(t)$ входящего потока запросов задается дискретно с 15-минутными интервалами в течение рабочего дня с 8:00 до 20:00 (интервал моделирования T = 720 мин). Функция интенсивности $\lambda(t)$ представлена на рис. 2.

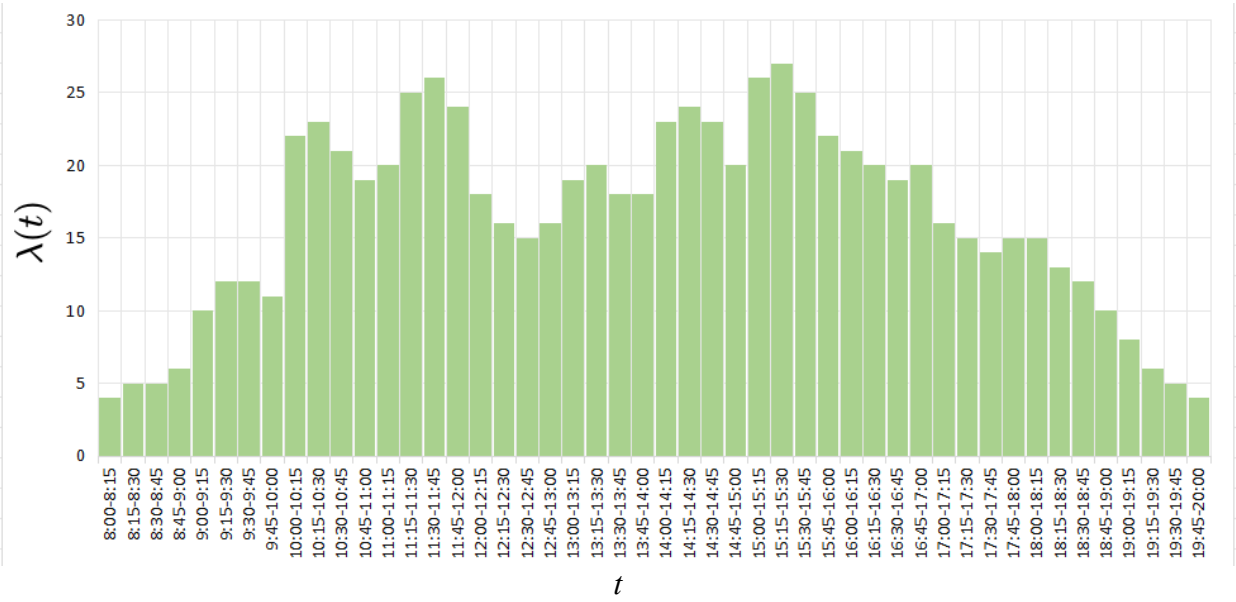


Рис. 2. Функция интенсивности поступления запросов в течение рабочего дня с 15 минутными интервалами

Для анализа эффективности работы системы обработки клиентских запросов применяются следующие оценки:

- среднее время нахождения запроса в системе:

$$S_t = \frac{\sum_{i=1}^{NA}(t_depart_i - A_i)}{NA}; \quad (1)$$

- среднее время задержки запроса в очереди на втором уровне:

$$S_{tq1} = \frac{\sum_{i=1}^{Nq1}(t_endL1_i - t_startL1_i)}{Nq1}; \quad (2)$$

- среднее время задержки запроса в очереди на третьем уровне:

$$S_{tq2} = \frac{\sum_{i=1}^{Nq2}(t_endL2_i - t_startL2_i)}{Nq2}; \quad (3)$$

- коэффициент занятости устройства k на втором уровне:

$$\rho_{1,k} = \frac{B1_k}{t_now} \; ; \tag{4}$$

- средний коэффициент занятости устройств на втором уровне:

$$\rho_1 = \frac{\sum_{k=1}^{m1} \rho_{1,k}}{m1} \; ; \tag{5}$$

- коэффициент занятости устройства k на третьем уровне:

$$\rho_{2,k} = \frac{B2_k}{t_now} \; ; \tag{6}$$

- средний коэффициент занятости устройств на третьем уровне:

$$\rho_2 = \frac{\sum_{k=1}^{m2} \rho_{2,k}}{m2} \; ; \tag{7}$$

- доля решенных запросов на первом уровне:

$$R_{bot} = \frac{C_{bot}}{ND} \times 100\% \; ; \tag{8}$$

- доля решенных запросов на втором уровне:

$$R_{L1} = \frac{C_{L1}}{ND} \times 100\% \; ; \tag{9}$$

- доля решенных запросов на третьем уровне:

$$R_{L2} = \frac{C_{L2}}{ND} \times 100\% \; ; \tag{10}$$

- доля запросов, переведенных на второй уровень:

$$E_1 = \frac{N_{q1}}{ND} \times 100\% \; ; \tag{11}$$

- доля запросов, переведенных на третий уровень:

$$E_2 = \frac{N_{q2}}{ND} \times 100\%, \tag{12}$$

где T – период моделирования; $m1$ – количество операторов первой линии поддержки; $m2$ – количество операторов второй линии поддержки; NA – количество запросов, прибывших к моменту времени t; ND – количество запросов, обработанных к моменту времени t; A_i – время прибытия в систему запроса i, $i = 1, \ldots, NA$; t_depart_i – время ухода из системы запроса i, $i = 1, \ldots, NA$; $t_startL1_i$ – время начала ожидания в очереди к первой линии i, $i = 1, \ldots, NA$; $t_startL2_i$ – время начала ожидания в очереди к второй линии i, $i = 1, \ldots, NA$; t_endL1_i – время окончания ожидания в очереди к первой линии i, $i = 1, \ldots, NA$; t_endL2_i – время окончания ожидания в очереди к второй линии i, $i = 1, \ldots, NA$; $Nq1$ – количество запросов в очереди к первой линии, прибывших к моменту времени t; $Nq2$ – количество запросов в очереди к второй линии, прибывших к моменту времени t; $B1_k$ – суммарное время занятости оператора k первой линии, $k = 1, \ldots, m1$; $B2_k$ – суммарное время занятости оператора k второй линии, $k = 1, \ldots, m2$; t_now – время окончания моделирования; C_{bot} – количество запросов, решенных

чат-ботом; C_{L1} – количество запросов, решенных первой линией; C_{L2} – количество запросов, решенных второй линией.

Приведем алгоритм, моделирующий работу системы обработки клиентских запросов со случайным распределением к первому из имеющихся свободных операторов, для второго и третьего уровня системы.

Алгоритм моделирования системы обработки запросов

Входные данные:

T — период моделирования (в минутах);
m1 — количество операторов первой линии;
m2 — количество операторов второй линии;
p_bot — вероятность решения запроса ботом;
p_L1 — вероятность решения запроса первой линией (при переводе);
μ_bot — среднее время обработки ботом, мин;
μ_L1 — среднее время обработки первой линией, мин;
μ_L2 — среднее время обработки второй линией, мин;
K — количество экспериментов;

Переменные:

t — текущее время в минутах.

Состояния системы:

$SS = ((q1; q2), (n1_1; n1_2; ...; n1_{m1}), (n2_1; n2_2; ...; n2_{m2}))$, где:
q1 — количество запросов в очереди к первой линии;
q2 — количество запросов в очереди ко второй линии;
$n1_k$, $n2_k$ — идентификаторы запроса, обслуживаемого *k*-м оператором первой и второй линии соответственно (0 — свободен).

Информация о запросах:

Для каждого запроса *i* хранятся:
A_i — время поступления;
$t_startL1_i$, t_endL1_i — время начала и окончания ожидания в очереди первой линии;
$t_startL2_i$, t_endL2_i — время начала и окончания ожидания в очереди второй линии;
t_depart_i — время ухода из системы;
$resolved_i$ — уровень решения (0 — бот, 1 — первая линия, 2 — вторая линия).

Количественные переменные:

NA — общее число поступивших запросов;
ND — общее число обслуженных запросов;
C_bot, C_L1, C_L2 — количество запросов, решённых на каждом уровне;
N_q1, N_q2 — количество запросов, побывавших в очереди к первой и второй линии;
$B1_k$, $B2_k$ — суммарное время занятости операторов (для каждого оператора);
Q1, Q2 — суммарное время ожидания в очередях (для расчета средних).

Список событий:

EL = {(*t_event, event_type, k, i*)}
t_event = {*t_ARRIVA, t_BOT, t_L1, t_L2*}
event_type = {*"ARRIVAL", "BOT_END", "L1_END", "L2_END"*}
EL={(*t_ARRIVAL,"ARRIVAL"*),(*t_BOT, "BOT_END", i*), (*t_L1, "L1_END", k, i*), (*t_L2, "L2_END", k, i*) }
queue_l1 = {(*i, t_startL1*)}; *// Список запросов в очереди к 1-й линии* (*i-й запрос, с временем поступления в очередь t_startL1*)
queue_l2 = {(*i, t_startL2*)}*; // Список запросов в очереди ко 2-й линии* (*i-й запрос, с временем поступления в очередь t_startL2*)

События хранятся в приоритетной очереди (куче) и извлекаются в порядке возрастания времени.

Выходные данные:

S_t — среднее время нахождения запроса в системе;
S_tq1 — среднее время задержки запроса в очереди первой линии;
S_tq2 — среднее время задержки запроса в очереди второй линии;
$R1_k$ — коэффициент занятости k-го оператора первой линии;
$R1$ — средний коэффициент занятости операторов первой линии;
$R2_k$ — коэффициент занятости k-го оператора второй линии;
$R2$ — средний коэффициент занятости операторов второй линии;
R_bot — доля запросов, решенных ботом;
R_L1 — доля запросов, решенных операторами первой линии;
R_L2 — доля запросов, решенных операторами второй линии;
$E1$ — доля запросов, переведенных операторам первой линии;
$E2$ — доля запросов, переведенных операторам второй линии.

Шаг 1. Инициализация

1. $t = 0$, $q1 = 0$, $q2 = 0$, $NA = 0$, $ND = 0$, $C_bot = 0$, $C_L1 = 0$, $C_L2 = 0$, $Q1 = 0$, $Q2 = 0$, $N_q1 = 0$, $N_q2 = 0$, $i = 0$
2. Для всех k от 1 до $m1$: $n1_k = 0$, $B1_k = 0$
3. Для всех k от 1 до $m2$: $n2_k = 0$, $B2_k = 0$
4. Сгенерировать время первого поступления: $t_ARRIVAL = GetArrivalTime(0)$
5. Добавить событие поступление запроса в список событий: $EL = \{(t_ARRIVAL, "ARRIVAL")\}$

Шаг 2. Основной цикл обработки событий

Пока $EL \neq \emptyset$ и $K > 0$:

1. Извлечь событие с минимальным временем: $t = min\{t(t_event) \mid t_event \in EL\}$
2. Обработать событие согласно его типу:
   Если $(t, "ARRIVAL") \in$ EL, то Случай 1
   Если $(t, "BOT_END", i) \in$ EL, то Случай 2
   Если $(t, "L1_END", k, i) \in$ EL, то Случай 3
   Если $(t, "L2_END", k, i) \in EL$, то Случай 4
3. Если $t \geq T$ и система пуста ($q1 = 0$, $q2 = 0 \mid q1, q2 \in SS$), то $K = K - 1$
4. Если $K = 0$, то переход к Шагу 3

Шаг 3. Расчет итоговых показателей

Рассчитать:

$$S_t = \frac{\sum_{i=1}^{NA}(t_depart_i - A_i)}{NA} \quad \text{по (1)}$$

$S_tq1 = Q1 / C_L1$, *если* $C_L1 > 0$ по (2)
$S_tq2 = Q2 / C_L2$, *если* $C_L2 > 0$ по (3)
$R1_k = B1_k / T$, *для* $\forall k$ по (4)
$R1 = \left(\sum_{k=1}^{m1} R1_k\right) / m1$ по (5)
$R2_k = B2_k / T$, *для* $\forall k$ по (6)
$R2 = \left(\sum_{k=1}^{m2} R2_k\right) / m2$ по (7)
$R_bot = C_bot / ND$ по (8)
$R_L1 = C_L1 / ND$ по (9)
$R_L2 = C_L2 / ND$ по (10)
$E1 = N_q1 / ND$ по (11)
$E2 = N_q2 / ND$ по (12)

Шаг 4. Конец алгоритма

Рассмотрим функции и процедуры, используемые в алгоритме моделирования системы обработки запросов.

Функция GetArrivalTime(last_time)

// Генерация времени следующего поступления методом прореживания для неоднородного пуассоновского процесса, last_time — время поступления крайнего запроса

1. $t = last_time$
2. Повторять:
   Сгенерировать $U_1 \sim Uniform(0, 1)$
   $t = t - (1 / \lambda_max) * ln(U_1)$
   Если $t \geq T$, вернуть ∞ (прибытий больше нет)
   Сгенерировать $U_2 \sim Uniform(0, 1)$, независимое от U_1
   Если $U_2 \leq \lambda(t) / \lambda_max$, то вернуть t
   Иначе повторить цикл

Функция GetServiceTime(mean)

// Генерация экспоненциально распределенного времени обслуживания, mean — среднее время обработки запросов

1. Сгенерировать $U \sim Uniform(0, 1)$
2. Вернуть $-mean * ln(U)$

Функция lambda(t)

// Возвращает значение интенсивности (рис. 2) для заданного времени t, start — начало интервала, end — конец интервала, count_per_15min — среднее количество запросов, поступивших в интервал от start до end

1. Для каждого интервала ($start, end, count_per_15min$):
   Если $start \leq t < end$, вернуть $count_per_15min$ / 15.0
2. Если интервал не найден, вернуть 0

Приведем описания каждого из случаев, рассмотренных в алгоритме моделирования системы обработки запросов.

Случай 1. Обработчик поступления запроса в систему (t, ARRIVAL):

1. $NA = NA + 1$, $i = NA$
2. $A_i = t$
3. Инициализировать поля запроса i:
   $t_startL1_i = 0$, $t_endL1_i = 0$,
   $t_startL2_i = 0$, $t_endL2_i = 0$,
   $t_depart_i = 0$, $resolved_i = 0$
4. Сгенерировать время следующего поступления:
   $t_ARRIVAL = GetArrivalTime(t)$
   Если $t_ARRIVAL < \infty$, добавить событие ($t_ARRIVAL$, *"ARRIVAL"*)
5. Начать обработку ботом:
   $serv_time = GetServiceTime(\mu_bot)$
   Добавить событие ($t + serv_time$, *"BOT_END"*, i)

Случай 2. Обработчик завершения обработки чат-ботом (t, BOT_END, i):

1. Сгенерировать $U \sim Uniform(0, 1)$
2. Если $U < p_bot$: *// Запрос решен ботом*
   $C_bot = C_bot + 1$
   $resolved_i = 0$
   $t_depart_i = t$
   $ND = ND + 1$
3. Иначе (перевод на первую линию):
   $N_q1 = N_q1 + 1$

$t_startL1_i = t$
Поместить запрос i в очередь $L1$: $(i, t_startL1_i) \in queue_l1$
$q1 = q1 + 1$
Вызвать процедуру $Assign_L1()$

Случай 3. Завершение работы оператора первой линии (t, L1_END, k, i):

1. Освободить оператора: $n1_k = 0$
2. Сгенерировать $U \sim Uniform(0, 1)$
3. Если $U < p_L1$: *// Запрос решен первой линией*
   $C_L1 = C_L1 + 1$
   $resolved_i = 1$
   $t_depart_i = t$
   $ND = ND + 1$
4. Иначе (перевод на вторую линию):
   $N_q2 = N_q2 + 1$
   $t_startL2_i = t$
   Поместить запрос i в очередь $L2$: $(i, t_startL2_i) \in queue_l2$
   $q2 = q2 + 1$
   Вызвать процедуру $Assign_L2()$
5. Вызвать процедуру $Assign_L1()$ (для назначения следующих запросов из очереди $L1$)

Случай 4. Завершение работы оператора второй линии (t, L2_END, k, i):

1. Освободить оператора: $n2_k = 0$
2. Запрос гарантированно решен:
   $C_L2 = C_L2 + 1$
   $resolved_i = 2$
   $t_depart_i = t$
   $ND = ND + 1$
3. Вызвать процедуру $Assign_L2()$

Приведем описания процедур, используемых в алгоритме моделирования системы обработки запросов.

Процедура Assign_L1():

// Обработка запроса операторами 1-й линей поддержки
Пока $q1 > 0$ и существует k такой, что $n1_k = 0$:

1. Извлечь первый запрос i из очереди $L1$: $i = min\{i \mid (i, t_startL1_i) \in queue_l1\}$
2. $q1 = q1 - 1$
3. $t_endL1_i = t$
4. $Q1 = Q1 + (t - t_startL1_i)$
5. $n1_k = i$
6. $serv_time = GetServiceTime(\mu_L1)$
7. Добавить событие $(t + serv_time, "L1_END", k, i)$
8. $B1_k = B1_k + serv_time$

Процедура Assign_L2():

// Обработка запроса операторами 2-й линей поддержки
Пока $q2 > 0$ и существует k такой, что $n2_k = 0$:

1. Извлечь первый запрос i из очереди $L2$: $i = \min\{i \mid (i, t_startL2_i) \in queue_l2\}$
2. $q2 = q2 - 1$
3. $t_endL2_i = t$
4. $Q2 = Q2 + (t - t_startL2_i)$
5. $n2_k = i$
6. $serv_time = GetServiceTime(\mu_L2)$

7. Добавить событие (t + *serv_time, "L2_END", k, i*)
8. $B2_k = B2_k$ + *serv_time*

Численные эксперименты

Для реализации предложенного алгоритма было разработано программное обеспечение на языке программирования Python версии 3.12.3[1] в среде программирования Visual Studio Code версии 1.109.5 и проведены численные эксперименты. В реализации используются следующие встроенные модули языка программирования Python: модуль heapq для организации списка событий в виде приоритетной очереди (кучи), что обеспечивает эффективное извлечение события с минимальным временем; модуль random для генерации случайных чисел, необходимых для моделирования вероятностных процессов (поступление запросов, время обслуживания, решение проблем); модуль math для выполнения математических операций, включая вычисление логарифмов при генерации экспоненциально распределенных величин.

Для проверки правильности работы программы была проведена ее верификация. Предположим, что моделированию подлежат N запросов в системе, состоящей из k устройств 1-й линии и m устройств 2-й линии: $N = 10$, $k = 3$, $m = 2$. Соответствующие времена событий приведены в табл. 1 (время в минутах).

Таблица 1

Данные моделирования

№ запроса	1	2	3	4	5	6	7	8	9	10
Время прибытия на 1-й уровень	10,34	14,86	15,75	19	19,32	20	20,88	24,8	28,32	37,95
Время прибытия на 2-й уровень	15,85	18,74	17,12	27,43	0	0	0	24,94	34,72	38,57
Время прибытия на 3-й уровень	0	0	0	0	0	0	0	29,35	0	0
Время ухода из системы	25,48	32,77	22,04	27,61	22,35	22,38	28,37	56,38	37,08	41,44
На каком уровне решен запрос	2	2	2	2	1	1	1	2	1	1
№ оператора 1-й линии	1	3	2	1	0	0	0	2	1	1
№ оператора 2-й линии	0	0	0	0	0	0	0	1	0	0

Среднее время нахождения запроса в системе, полученное в программе, равно 10,468 минут. Из таблицы 1 следует, что среднее время запроса в системе S_T равно:

$$S_T = \frac{15{,}14 + 17{,}91 + 6{,}29 + 8{,}61 + 3{,}03 + 2{,}38 + 7{,}49 + 31{,}58 + 8{,}76 + 3{,}49}{10} = 10{,}468(мин).$$

Следовательно, программа работает корректно.

Были проведены численные эксперименты на данных, предоставленных экспертом, для системы, используемой контакт-центром банка:

$T = 720$— период моделирования (в минутах);
$m1 = 8$ — количество операторов первой линии;
$m2 = 3$ — количество операторов второй линии;
$p_bot = 0.35$ — вероятность решения запроса ботом;
$p_L1 = 0.70$ — вероятность решения запроса первой линией (при переводе);
$\mu_bot = 2.5$ — среднее время обработки ботом, мин;
$\mu_L1 = 4.5$ — среднее время обработки первой линией, мин;
$\mu_L2 = 12.0$ — среднее время обработки второй линией, мин.

Приведем результаты тридцати экспериментов из 180, цель которых – оценить эффективность действующей системы, результаты представлены в листинге 1.

[1] Python Software Foundation. Python Language Reference, version 3.12. URL: https://docs.python.org/3.12/ (дата обращения: 15.02.2026).

Листинг 1

Результаты моделирования действующей системы контакт-центра банка

```
================================================
СРЕДНИЕ РЕЗУЛЬТАТЫ ПО 30 ЭКСПЕРИМЕНТАМ
================================================
Всего поступило запросов (ср.): 791.0
Всего обслужено запросов (ср.): 791.0
Среднее время пребывания в системе: 14.974 мин
Среднее время ожидания в очереди L1: 0.052 мин
Среднее время ожидания в очереди L2: 36.758 мин
Средний коэффициент занятости L1: 0.403
Средний коэффициент занятости L2: 0.814

Доли решённых запросов (ср.):
  Чат-бот: 0.351
  Первая линия: 0.457
  Вторая линия: 0.192

Доли переведенных запросов (ср.):
  На первую линию: 0.649
  На вторую линию: 0.192

========== СРЕДНЯЯ ПОЧАСОВАЯ СТАТИСТИКА ==========
Час      | L1 | L2 | Поступило | Обслужено | Ср.ожид.L1 | Ср.ожид.L2 | Ср.время в системе | Загрузка L1 | Загрузка L2
-------------------------------------------------------------------------------------------------------------------
08:00-09:00 |  8 |  3 |     20.7 |      17.7 |     0.000 |     0.096 |              5.993 |     11.20% |     18.94%
09:00-10:00 |  8 |  3 |     45.3 |      42.8 |     0.001 |     0.797 |              7.265 |     26.81% |     46.98%
10:00-11:00 |  8 |  3 |     84.8 |      78.2 |     0.062 |     5.238 |              7.534 |     49.70% |     80.14%
11:00-12:00 |  8 |  3 |     94.0 |      87.8 |     0.072 |    14.178 |              9.032 |     57.00% |     90.23%
12:00-13:00 |  8 |  3 |     67.1 |      69.6 |     0.030 |    27.662 |             12.805 |     45.21% |     95.91%
13:00-14:00 |  8 |  3 |     75.9 |      76.2 |     0.042 |    29.172 |             13.402 |     44.63% |     89.15%
14:00-15:00 |  8 |  3 |     88.5 |      82.8 |     0.059 |    31.003 |             12.265 |     53.15% |     96.96%
15:00-16:00 |  8 |  3 |     99.0 |      96.2 |     0.091 |    39.777 |             13.058 |     58.20% |     98.51%
16:00-17:00 |  8 |  3 |     80.1 |      79.0 |     0.045 |    46.887 |             14.541 |     47.88% |     99.17%
17:00-18:00 |  8 |  3 |     61.7 |      64.0 |     0.064 |    58.596 |             21.093 |     39.97% |     95.75%
18:00-19:00 |  8 |  3 |     51.5 |      56.7 |     0.004 |    60.482 |             22.897 |     34.40% |     91.31%
19:00-20:00 |  8 |  3 |     22.4 |      31.3 |     0.002 |    56.514 |             35.025 |     15.80% |     74.13%
```

Результаты моделирования показали неэффективность действующий системы, поскольку время ожидания в очереди к операторам второй линии поддержки составляет в среднем более 35 минут, загрузка операторов первой линии поддержки мала и составляет 40% от рабочего времени. Во избежание такой ситуации был предложен вариант сменного графика в зависимости от рабочего времени при условии загрузки операторов двух линий не более 85%, а среднее почасовое ожидание в очереди – не более 5 минут. Один из вариантов распределения числа операторов по часам, представлен в табл. 2, остальные входные данные остались прежними.

Таблица 2

Распределение числа операторов по часам

Время	Количество операторов 1-й линии	Количество операторов 2-й линии
8:00–9:00	2	2
9:00–10:00	3	2
10:00–11:00	5	4
11:00–12:00	6	5
12:00–13:00	5	5
13:00–14:00	5	4
14:00–15:00	5	5
15:00–16:00	6	5
16:00–17:00	5	5
17:00–18:00	4	4
18:00–19:00	4	4
19:00 –0:00	2	2

Результаты 30 экспериментов с предложенным сменным графиком операторов представлены в листинге 2.

Листинг 2

Результаты моделирования с предложенным сменным графиком операторов

```
================================================
СРЕДНИЕ РЕЗУЛЬТАТЫ ПО 30 ЭКСПЕРИМЕНТАМ
================================================
Всего поступило запросов (ср.): 794.4
Всего обслужено запросов (ср.): 794.4
Среднее время пребывания в системе: 9.698 мин
Среднее время ожидания в очереди L1: 2.124 мин
Среднее время ожидания в очереди L2: 2.649 мин
Средний коэффициент занятости L1: 0.716
Средний коэффициент занятости L2: 0.625

Доли решённых запросов (ср.):
  Чат-бот: 0.346
  Первая линия: 0.458
  Вторая линия: 0.196

Доли переведенных запросов (ср.):
  На первую линию: 0.654
  На вторую линию: 0.196

========== СРЕДНЯЯ ПОЧАСОВАЯ СТАТИСТИКА ==========
Час      | L1 | L2 | Поступило | Обслужено | Ср.ожид.L1 | Ср.ожид.L2 | Ср.время в системе | Загрузка L1 | Загрузка L2
-------------------------------------------------------------------------------------------------------------------
08:00-09:00 |  2 |  2 |     20.7 |      16.6 |     0.970 |     0.819 |              6.879 |     45.80% |     25.26%
09:00-10:00 |  3 |  2 |     45.8 |      42.3 |     1.849 |     2.802 |              8.059 |     65.38% |     61.95%
10:00-11:00 |  5 |  4 |     84.0 |      78.7 |     1.490 |     2.608 |              8.433 |     77.37% |     69.51%
11:00-12:00 |  6 |  5 |     96.8 |      91.8 |     1.608 |     3.061 |              9.403 |     76.15% |     68.83%
12:00-13:00 |  5 |  5 |     64.9 |      73.3 |     2.021 |     3.052 |             10.820 |     70.46% |     63.39%
13:00-14:00 |  5 |  4 |     75.7 |      73.7 |     1.740 |     2.070 |              9.355 |     74.95% |     72.07%
14:00-15:00 |  5 |  5 |     93.7 |      89.2 |     3.017 |     1.471 |              9.579 |     85.68% |     65.70%
15:00-16:00 |  6 |  5 |     98.6 |     100.0 |     2.116 |     2.306 |              9.693 |     81.00% |     76.36%
16:00-17:00 |  5 |  5 |     80.3 |      81.3 |     1.560 |     2.979 |              9.832 |     77.27% |     72.34%
17:00-18:00 |  4 |  4 |     60.0 |      63.6 |     3.161 |     2.774 |             10.972 |     79.63% |     67.96%
18:00-19:00 |  4 |  4 |     51.6 |      55.8 |     1.562 |     1.297 |              9.847 |     66.55% |     56.43%
19:00-20:00 |  2 |  2 |     22.4 |      25.7 |     1.528 |     1.420 |             10.205 |     58.43% |     50.47%
```

По результатам моделирования можно сделать вывод, что модификация имеющейся системы соответствует требованиям, а именно: загрузка операторов не более 85% и среднее почасовое ожидание в очереди не более 5 минут. Кроме того, уменьшилось среднее время ожидания в очереди к операторам второй линии поддержки с 37 минут до 2,7 минут и соответственно общее время в системе уменьшилось с 15 минут до 9,7 минут, что положительно сказалось на обслуживании клиентов.

Анализ финансовых издержек гибридного контакт-центра показал, что снижаются затраты на заработную плату операторов в среднем на 18,3%. Предположим, что часовая зарплата оператора первой линии будет равна x, а зарплата оператора второй линии $1{,}5x$. Тогда при существующем варианте распределения финансовые затраты на оплату труда будут составлять $12\times 8x + 12\times 4{,}5x = 150x$. При предложенном варианте распределения финансовые затраты на оплату труда будут составлять

$$(2+3+5+6+5+5+5+6+5+4+4+2)x + +(2+2+4+5+5+4+5+5+5+4+4+2)\cdot 1{,}5x = 122{,}5x.$$

Таким образом, в день затраты уменьшатся на величину $150x - 122{,}5x = 27{,}5x$.

Заключение

В статье была представлен алгоритм, моделирующий работу трехуровневого гибридного контакт-центра банка, включающего чат-ботов, операторов первой и второй линий поддержки

и учитывающий неоднородный характер входящего потока запросов, вероятностный перевод запросов между уровнями и экспоненциальное распределение времени обслуживания. На основе проведенных численных экспериментов были получены следующие основные результаты:

- анализ текущей системы показал ее неэффективность: при фиксированном количестве операторов (8 первой линии и 3 второй линии) среднее время ожидания в очереди ко второй линии превышало 35 минут, а загрузка операторов первой линии составляла всего около 40%. Это свидетельствует о несбалансированности распределения ресурсов и необходимости оптимизации;
- предложен сменный график работы операторов, при котором количество операторов изменяется по часам в зависимости от интенсивности входящего потока. Целевыми показателями выбраны: загрузка операторов не более 85% и среднее почасовое ожидание в очереди не более 5 минут. Разработанное расписание представлено в табл. 1;
- анализ результатов моделирования гибридного контакт-центра показал, что предложенные модификации имеющейся системы позволяют адаптировать ресурсы контакт-центра под реальную динамику нагрузки.
- переход на сменный график позволяет сократить фонд оплаты труда в среднем на 18,3% ежедневно. Экономия достигается за счет привлечения операторов только в часы пиковой нагрузки и снижения избыточности в периоды низкой активности.

Разработанный алгоритм и программное обеспечение находятся в стадии тестового режима у специалистов контакт-центра для: планирования количества операторов, оценки влияния изменения вероятностей решения проблем (например, при дообучении чат-ботов) на загрузку системы, прогнозирования времени ожидания клиентов при различных сценариях нагрузки, оптимизации графиков работы персонала с учетом экономических показателей.

Дальнейшие направления исследований могут включать: внедрение приоритетов для различных типов запросов, интеграцию модели с системами мониторинга реального времени для создания цифрового двойника контакт-центра, а также исследование влияния других законов распределения времени обслуживания на характеристики системы.

СПИСОК ЛИТЕРАТУРЫ | REFERENCES

[Ani21] Anisimov V., Limnios N. (eds.). Queuing Theory 2. Advanced Trends. Hoboken: Wiley, 2021. 324 p.

[Bei19] Beisbart C., Saam N. J. (eds.). Computer Simulation Validation. Fundamental Concepts, Methodological Frameworks, and Philosophical Perspectives. Cham: Springer, 2019. 382 p. 10.1007/978-3-319-70766-2.

[Bor21] Borshchev A. The Big Book of Simulation Modeling: Multimethod Modeling with AnyLogic 8. Oakbrook Terrace: AnyLogic North America, 2021. 648 p. 10.13140/RG.2.2.29143.09125.

[Cha22a] Chakravarthy S. R. Introduction to Matrix-Analytic Methods in Queues 1. Analytical and Simulation Approach – Basics. Hoboken: Wiley, 2022. 366 p. 10.1002/9781394165421.

[Cha22b] Chakravarthy S. R. Introduction to Matrix-Analytic Methods in Queues 2. Analytical and Simulation Approach – Advanced Models. Hoboken: Wiley, 2022. 440 p. 10.1002/9781394174201.

[Fak24] Fakhimi M., Mustafee N. (eds.). Hybrid Modeling and Simulation. Conceptualizations, Methods and Applications. Cham: Springer, 2024. 352 p. 10.1007/978-3-031-59999-6.

[Kel23] Kelton W. D., Sadowski R. P., Zupick N. B. Simulation with Arena. 7th ed. New York: McGraw-Hill, 2023. 688 p.

[Law19] Law A. M. Simulation Modeling and Analysis. 6th ed. New York: McGrawHill, 2024. 688 p.

[Ros23] Ross S. M. Simulation. 6th ed. London: Elsevier Science, 2023. 336 p. 10.1016/B978-0-12-824160-3.00001-2.

[Sim26] SimPy. Discrete event simtion for Python. URL: https://simpy.readthedocs.io (дата обращения: 15.02.2026).

[Smi24] Smith J. S., Sturrock D. T. Simio and Simulation: Modeling, Analysis, Applications. 7th ed. Sewickley: Simio LLC, 2024. 505 p.

[Van18] Van der Ham R. Salabim: Open Source Discrete Event Simulation and Animation in Python // Proc. of the 2018 Winter Simulation Conference (WSC). Gothenburg, Sweden, 2018. P. 395–406. 10.21105/joss.00767.

[Van23] Van der Ham R. Salabim: Discrete Event Simulation in Python: User and Reference Manual, 2023. 330 p.

ОБ АВТОРАХ | ABOUT THE AUTHORS



БУРХАНОВ Ирик Альбертович
Уфимский университет науки и технологий, Россия.
IrikBurkhanov1@yandex.ru.
Магистрант по напр. «Мат. обеспечение и администрирование информационных систем».

BURKHANOV Irik Albertovich
Ufa University of Science and Technology, Russia.
IrikBurkhanov1@yandex.ru.
Master's student in Mathematical Support and Administration of Information Systems.



ВАЛЕЕВА Аида Фаритовна
Уфимский университет науки и технологий, Россия.
aida_val2004@mail.ru ORCID: 0000-0003-3687-4925.
Проф. каф. вычислительной математики и кибернетики. Д-р техн. наук по мат. моделированию, числ. методам и комплексам программ, профессор.

VALEEVA Aida Faritovna
Ufa University of Science and Technology, Russia.
aida_val2004@mail.ru ORCID: 0000-0003-3687-4925.
Prof., Computational Mathematics and Cybernetics Dept. Dr. of Tech. Sc. in Mathematical Modeling, Numerical Methods, and Software Systems.



ВАЛЕЕВ Руслан Сагитович
Уфимский университет науки и технологий, Россия.
ruslan__valeev@inbox.ru
ORCID: 0000-0002-0550-4676.
Доц. каф. вычислительной математики и кибернетики. Вед. консультант ООО «Вентра Ай-Ти». Канд. техн. наук.

VALEEV Ruslan Sagitovich
Ufa University of Science and Technology, Russia.
ruslan__valeev@inbox.ru
ORCID: 0000-0002-0550-4676.
Assoc. Prof., Computational Mathematics and Cybernetics Dept. Cand. of Tech. Sc. in Mathematical Modeling, Numerical Methods, and Software Systems.

МЕТАДАННЫЕ | METADATA

Заглавие: Моделирование системы обработки запросов в гибридном контакт-центре банка.

Авторы: Бурханов И. А., Валеева А. Ф., Валеев Р. С.

Аннотация: Статья посвящена моделированию процесса обработки входящих запросов в гибридном контакт-центре банка, включающем трехуровневую систему обслуживания клиентов, состоящую из чат-ботов, операторов первой и второй линий поддержки. Поступление запросов описывается неоднородным пуассоновским процессом. Разработан алгоритм, моделирующий приход запросов на различные уровни с учетом нерешенных клиентских запросов. При этом вычисляются показатели, позволяющие оценить эффективность работы контакт-центра, а именно: среднее время пребывания запроса в системе, время ожидания в очередях, коэффициенты загрузки операторов и доли решенных запросов на каждом уровне. Предложенный алгоритм позволяет руководству банка оптимизировать работу контакт-центра без проведения экспериментов в реальном времени.

Ключевые слова: Трехуровневая система обслуживания клиентов; система массового обслуживания с устройствами; соединенными параллельно; неоднородный пуассоновский процесс; дискретно-событийная имитационная модель.

Язык: Русский.

Статья поступила в редакцию 22 июля 2026 г.

Title: The modeling of the query processing system in a bank's hybrid contact center.

Authors: Burhanov I. A., Valeeva A. F., Valeev R.S.

Abstract: The article is devoted to the modeling of incoming requests processing in a bank's hybrid contact center that includes a three-level customer's service system consisting of chat-bots, the first and the second line of the operational support. The acceptance of requests is described by a non-homogeneous Poisson process. The algorithm that simulates an incoming of requests at various levels with account of the unresolved client requests has been developed. At the same time, the indicators that enable to assess the efficiency of a contact center are calculated, namely: the average time of a request's life in the system, a queue waiting time, the operator's load factors, and the percentage of resolved requests at each level. The proposed algorithm allows the bank's management to optimize the work of a contact center without conducting experiments in the real time.

Key words: A three-level customer's service system; a queuing system with devices connected in parallel; a non-homogeneous Poisson process; a discrete event simulation model.

Language: Russian.

The article was received by the editors on 22 July 2026.