электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 004.942

Анализ вариантов архитектуры сервис ориентированной АСУ

Глинка М.В., *студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Системы обработки информации и управления»

Научный руководитель: Черненький В.М., д.т.н.
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана
chernen@bmstu.ru

Описание предметной области. Перед авиакомпаниями по всему миру остро стоит вопрос о расширении возможностей по продаже билетов. Чем больше у компании возможностей продать билеты на свои рейсы, тем большую прибыль она получит. Вместе с тем, до 60-х годов 20 века возможности по бронированию билетов у населения были ограничены. Такое положение дел сохранялось вплоть до изобретения первых GDS.

GDS - глобальная система бронирования, содержащая тарифы различных авиакомпаний. GDS предоставляет возможности для бронирования авиабилетов по всему миру. Кроме авиабилетов различные GDS позволяют бронировать все что только возможно забронировать: отели, яхты, машины и даже велосипеды. На сегодняшний день, в мире существует несколько GDS, наиболее крупными GDS являются Amadeus, Sabre, Sirena.

Однако, доступ к системе могли иметь только крупные компании из-за солидной платы за использование. К примеру, в любой достаточно большой туристической фирме есть терминал для работы с той или иной GDS.

Владельцы GDS, так же как и владельцы авиакомпаний, были заинтересованы в дальнейшем расширении рынков сбыта. С развитием сетевых технологий некоторые GDS стали предоставлять технические решения, для посредников. Для того, чтобы воспользоваться сервисами GDS, достаточно создать сервис-ориентированную АСУ, поддерживающую протокол взаимодействия с конкретной GDS. Теперь не нужно идти в турфирму, чтобы спланировать свое путешествие. Доступ к системам бронирования стал общим. GDS предоставляет свои сервисы по поиску и бронированию тарифов компаниям-посредникам OneTwoTrip, Ozon.travel, AnyWayAnyDay, Agent.ru.

Для того, чтобы забронировать билет на самолет, достаточно зайти на сайт нужной вам компании посредника и выбрать даты перелета. Остальное выполнит за вас технологическое решение, реализованное в данной компании-посреднике.

Постановка задачи исследования. Компании-посредники предоставляют сервисы различных GDS для широкого пользователя. При построении АСУ компании-посредника возникает вопрос о наиболее эффективной архитектуре. Система должна обрабатывать множество запросов от пользователей. Время выполнения запроса ограничено, и по истечении таймаута запрос снимается с обслуживания. Критерием эффективности работы системы будет отношение количества невыполненных запросов бронирования к количеству выполненных запросов.

Требуется проанализировать несколько вариантов структуры АСУ, определить узкие места, и сделать вывод о целесообразности внедрения того или иного архитектурного решения. Требуется учесть сезонное повышение интенсивности бронирований на сайте. В качестве исходной архитектуры выбираем базовый вариант, предложенный на Рис. 1.

Выбор метода моделирования. Для моделирования системы можно использовать как аналитические методы моделирования, так и имитационные. Аналитические методы позволяют получить показатели эффективности работы системы в виде функций от ее параметров. Как правило, аналитическая модель представляет собой систему уравнений. Решив ее, можно получить параметры, нужные для вычисления выходных характеристик системы (среднее время обработки запроса, длину очереди и прочие).

Аналитические методы дают компактное решения. однако применить их для решения широкого класса задач затруднительно. Это связано с тем, что реальная система зачастую не поддается математическому описанию из-за повышенной сложности. Кроме того, в процессе выведения формул допускаются ряд предположений, не всегда верно отражающих свойства реальной системы.

Имитационное моделирование свободно от указанных недостатков. Поэтому при выборе метода исследования было отдано предпочтение методу имитационного моделирования.

В статье предлагается описать функционирование АСУ с помощью задания совокупности связанных процессов. Для этого был использован аппарат описания, изложенной в [4]. В соответствии с этим все описания моделируемых процессов приведены в виде подпроцессных графов состояний (ПГС).

Выбор среды моделирования. Язык GPSS – это язык декларативного типа, построенный по принципу объектно-ориентированного языка. Основными элементами этого

языка являются транзакты и блоки, которые отображают соответственно динамические и статические объекты моделируемой системы.[5]. Язык отличают простота изучения, широкий диапазон использования, простой и удобный интерфейс.

Структура рассматриваемой АСУ отображена на Рис. 1.

Рис. 1. Базовая схема АСУ компании-посредника

Описание базовой схемы. Базовая схема АСУ состоит из следующих компонентов:

1. Сайт. Принимает запросы пользователей на поиск и бронирование билетов в различных GDS. На сайте пользователь задает временной интервал и вид перелета, а так же вводит свои личные данные.

Сайт обрабатывает множество запросов параллельно. Количество параллельно выполняемых запросов может варьироваться в зависимости от производительности сервера. Сайт выполняет запросы к сервису бронирования.

У запросов пользователя к страницам сайта существует таймаут. По истечении таймаута пользователю придется повторить попытку зайти на сайт.

2. Сервис бронирования выполняет запросы к GDS. Сервис обрабатывает множество запросов параллельно. Количество параллельно выполняемых запросов так же зависит от производительности сервера. Для нашей конкретной задачи, мы примем число одновременно выполняемых запросов равным 15. Выбор обусловлен техническими возможностями существующей АСУ.

Сервисы запрашивают тарифы компаний, которые попадают в заданный временной интервал. Сервисы так же выполняют операции бронирования перелета и его аннуляции.

Обращение сервиса к GDS происходит за 2-6 секунд.

3. GDS - глобальная система бронирования. Является черным ящиком.

GDS обрабатывает запрос на бронирование в течение 4-15 секунд.

Запрос может выполняться не более 30 секунд. По истечении 30 секунд запрос считается невыполненным.

Для решения задачи необходимо:

- Провести моделирование выбранного архитектурного решения
- Учесть таймаут запросов.
- Учесть сезонный характер нагрузки на АСУ.
- Предусмотреть возможность изменения архитектуры системы для достижения лучших результатов.
- Найти оптимальное архитектурное решение для АСУ компании-посредника
- Реализовать таймаут у запросов. По истечении таймаута запрос должен быть уничтожен.
- Определить как влияет изменение количества сервисов, выполняющих запросы к GDS на выбранный показатель качества функционирования системы.

По результатам исследования необходимо сделать выводы о целесообразности проводимых преобразований.

Анализируемые варианты архитектуры. Вариант архитектуры с одним сервисом представлен на Рис. 2.

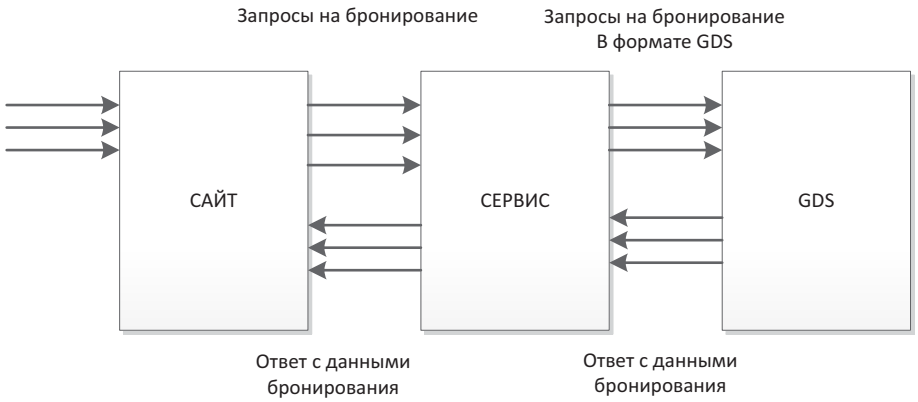

Рис. 2. Вариант архитектуры с одним сервисом

Промоделируем первый вариант. Обращения к GDS здесь выполняет один сервис. На сайт обращаются примерно 2 раза в секунду, сайт может обрабатывать до 100 запросов бронирования одновременно. Сервис может обрабатывать 15 бронирований

одновременно. Ограничение обусловлено техническими возможностями сервера, на котором располагается сервис. Процессный граф состояний для варианта архитектуры с одним сервисом представлен на Рис. 2.

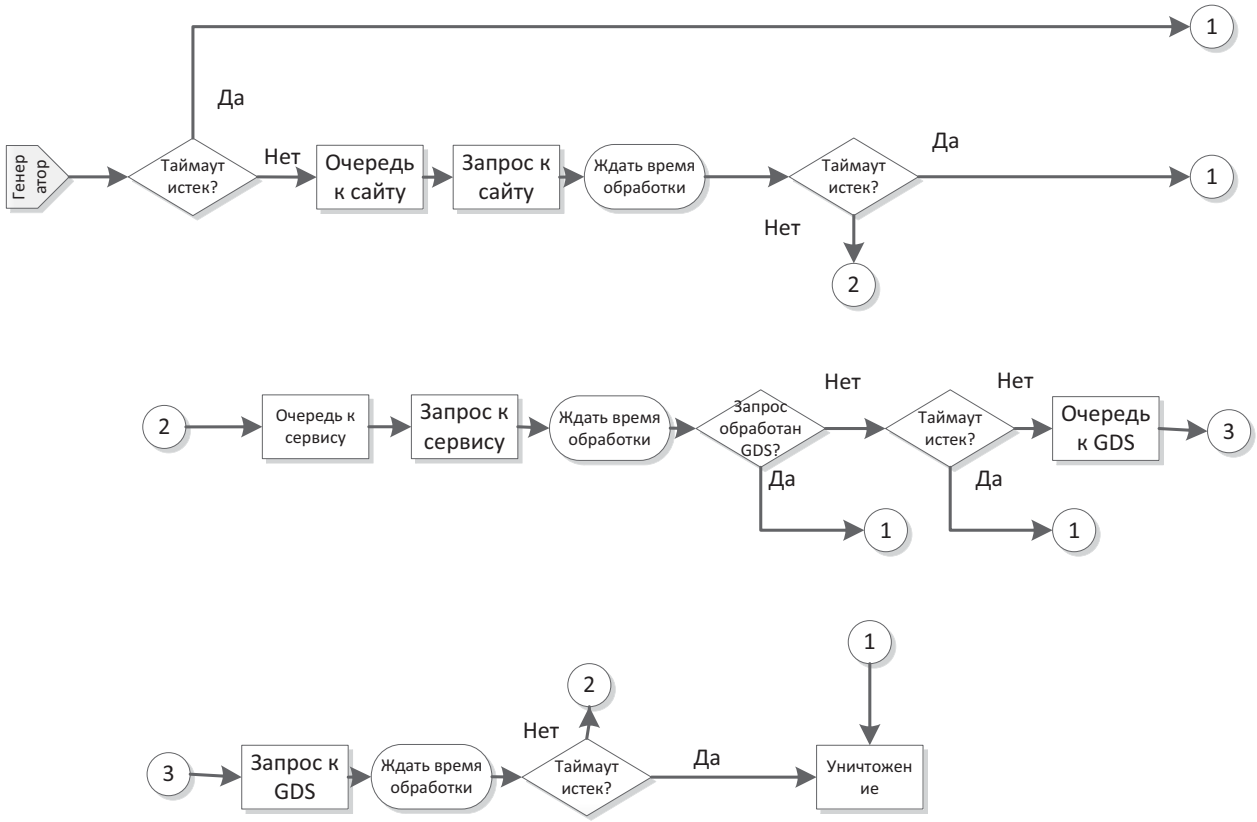

Рис. 2. Процессный граф состояний для варианта архитектуры с одним сервисом

Листинг 1. Моделирование АСУ компании-посредника с одним сервисом.

```
SITE      STORAGE   100
SERVICE1  STORAGE   15
GDS STORAGE   100
GENERATE .5
ASSIGN 1,0

;САЙТ

QUEUE QUEUE1         ;Занять очередь к сайту
ENTER SITE           ;Занять сайт
DEPART QUEUE1
ADVANCE 4,2          ;Ждем, пока сайт обработает запрос
LEAVE SITE



;СЕРВИСЫ БРОНИРОВАНИЯ

BOOKING QUEUE SERVICE1_QUEUE ;Занять очередь к сервису
ENTER SERVICE1       ;Занять сервис
DEPART SERVICE1_QUEUE
ADVANCE 4,2          ;Обработка запроса
LEAVE  SERVICE1      ;Освободить сервис
TEST LE M1,30,TERMINATOR;
```

http://sntbul.bmstu.ru/doc/593460.html

```
TEST NE P1,1,OK_TERMINATOR
TRANSFER ,GDSMARK

;GDS - Глобальная система бронирования

GDSMARK QUEUE GDS_QUEUE
ENTER GDS           ;Занять GDS
DEPART GDS_QUEUE
ADVANCE 9,5         ; Обработка запроса
LEAVE  GDS          ; Освободить сервис 3
ASSIGN 1,1          ;Пометить запрос как обработанный

TEST LE M1,25,TERMINATOR;
TRANSFER ,BOOKING   ;Отправить ответ на один из сервисов



OK_TERMINATOR  TERMINATE
TERMINATOR TERMINATE
GENERATE 600
TERMINATE 1
```

Листинг 2. Результаты моделирования

```
OK_TERMINATOR      24         TERMINATE          789             0       0
FAIL_TERMINATOR    25         TERMINATE          355             0       0


QUEUE              MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0)
RETRY
 QUEUE1              1    0   1199   1199     0.000      0.000      0.000   0
 SERVICE1_QUEUE     24   13   2227     53    12.850      3.462      3.547   0
 GDS_QUEUE           1    0   1177   1177     0.000      0.000      0.000   0


STORAGE            CAP. REM. MIN. MAX.  ENTRIES AVL.  AVE.C. UTIL. RETRY
DELAY
 SITE              100   91   0    11     1199   1    7.885  0.079    0    0
 SERVICE1           15    0   0    15     2214   1   14.687  0.979    0   13
 GDS               100   82   0    26     1177   1   17.508  0.175    0    0
```

Как видно из результатов моделирования, наибольшая очередь, 2227 транзактов, выстраивается к сервису (SERVICE1_QUEUE). Кроме того, на метку OK_TERMINATOR, означающую успешное окончание процесса, пришло в 2 раза меньше транзактов, чем на FAIL_TERMINATOR, означающую неудачное окончание.

Вывод. Вариант архитектуры с одним сервисом не подходит, поскольку около половины всех запросов отбрасываются до истечения таймаута. Бутылочным горлышком в данном случае является сервис бронирования, поскольку к нему выстраивается наибольшая очередь.

Вариант архитектуры с двумя сервисами представлен на Рис. 3.

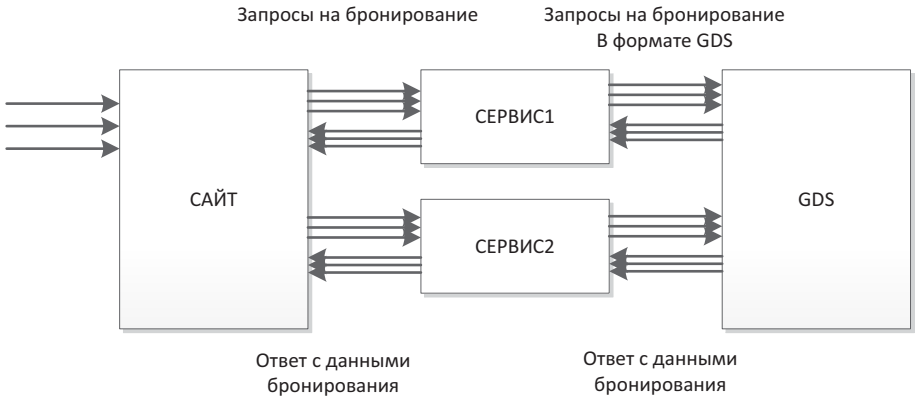

Рис. 3. Вариант архитектуры с двумя сервисами

Промоделируем второй вариант архитектуры. Обращения к GDS здесь выполняют два сервиса. На сайт обращаются примерно 2 раза в секунду, сайт может обрабатывать до 100 запросов бронирования одновременно. Один сервис может обрабатывать 15 бронирований одновременно. Ограничение обусловлено техническими возможностями сервера, на котором располагается сервис. . Процессный граф состояний для варианта архитектуры с двумя сервисами представлен на Рис. 4.

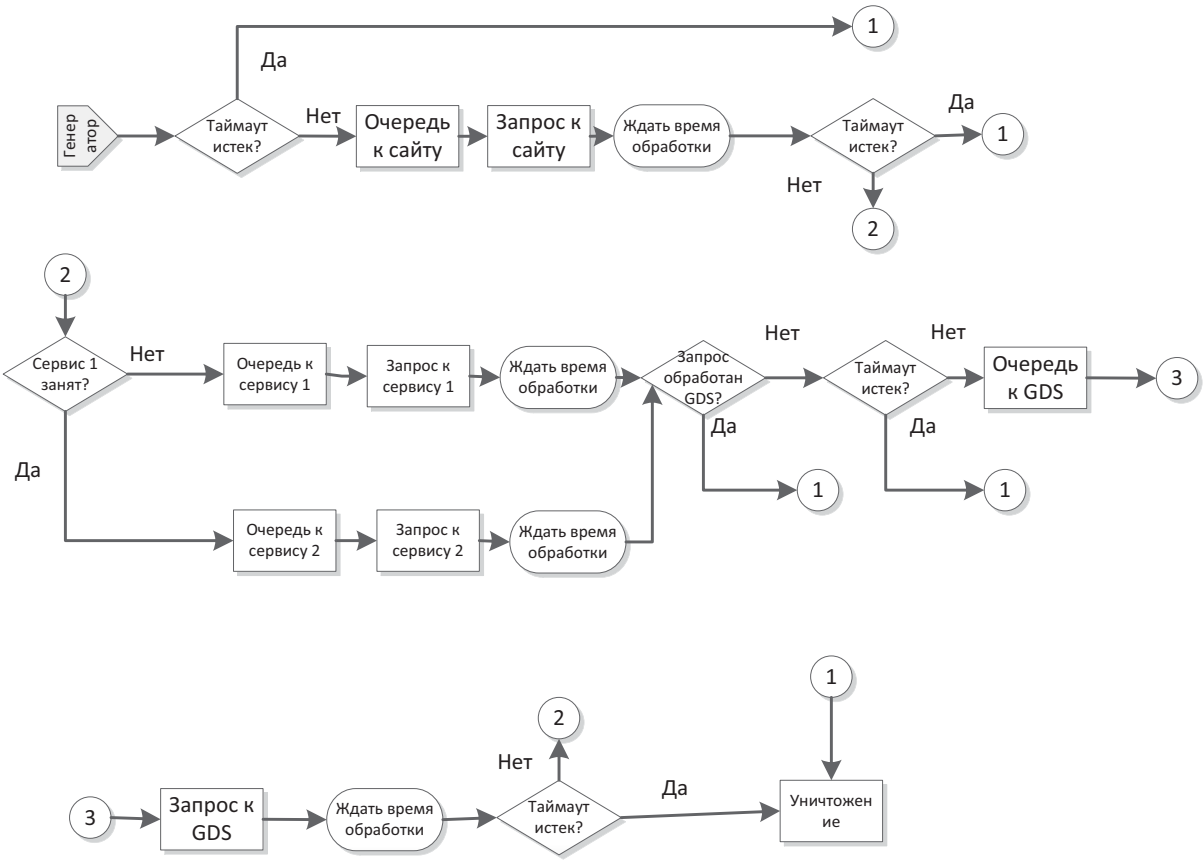

Рис. 4. ПГС для варианта архитектуры с двумя сервисами

Листинг 3. Моделирование АСУ компании-посредника с двумя сервисами.

```
SITE      STORAGE   100
SERVICE1  STORAGE   15
SERVICE2  STORAGE   15
GDS STORAGE   100
GENERATE .5
ASSIGN 1,0

;САЙТ

QUEUE QUEUE1        ;Занять очередь к сайту
ENTER SITE          ;Занять сайт
DEPART QUEUE1
ADVANCE 4,2         ;Ждем, пока сайт обработает запрос
LEAVE SITE



;СЕРВИСЫ БРОНИРОВАНИЯ
;РАСПРЕДЕЛЕНИЕ

BOOKING_FORK GATE SNF SERVICE1,BOOKING_SERVICE2 ;TRANSFER
.500,BOOKING_SERVICE1,BOOKING_SERVICE2

;СЕРВИС БРОНИРОВАНИЯ 1

BOOKING_SERVICE1 QUEUE SERVICE1_QUEUE ;Занять очередь к сервису
ENTER SERVICE1       ;Занять сервис
DEPART SERVICE1_QUEUE
ADVANCE 4,2          ;Обработка запроса
LEAVE  SERVICE1      ;Освободить сервис
TEST LE M1,30,TERMINATOR;
TEST NE P1,1,OK_TERMINATOR
TRANSFER ,GDSMARK



;СЕРВИС БРОНИРОВАНИЯ 2

BOOKING_SERVICE2 QUEUE SERVICE2_QUEUE ;Занять очередь к сервису
ENTER SERVICE2       ;Занять сервис
DEPART SERVICE2_QUEUE
ADVANCE 4,2          ;Обработка запроса
LEAVE  SERVICE2      ;Освободить сервис
TEST LE M1,30,TERMINATOR;
TEST NE P1,1,OK_TERMINATOR
TRANSFER ,GDSMARK


;GDS - Глобальная система бронирования

GDSMARK QUEUE GDS_QUEUE
ENTER GDS           ;Занять GDS
DEPART GDS_QUEUE
ADVANCE 9,5         ; Обработка запроса
LEAVE  GDS          ; Освободить сервис 3
ASSIGN 1,1          ;Пометить запрос как обработанный

TEST LE M1,25,TERMINATOR;
TRANSFER ,BOOKING_FORK              ;Отправить ответ на один из сервисов
```

```
OK_TERMINATOR  TERMINATE
TERMINATOR TERMINATE
GENERATE 600
TERMINATE 1
```

Листинг 4. Результаты моделирования

```
OK_TERMINATOR      33    TERMINATE         1157             0       0
FAIL_TERMINATOR    34    TERMINATE            0             0       0

QUEUE              MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0)
RETRY
 QUEUE1              1    0   1200   1200     0.000      0.000      0.000   0
 SERVICE1_QUEUE      1    0   1907   1907     0.000      0.000      0.000   0
 GDS_QUEUE           1    0   1184   1184     0.000      0.000      0.000   0
 SERVICE2_QUEUE      1    0    449    449     0.000      0.000      0.000   0


STORAGE            CAP. REM. MIN. MAX.  ENTRIES AVL.  AVE.C. UTIL. RETRY
DELAY
 SITE              100   92   0    12     1200   1    8.045  0.080    0    0
 SERVICE1           15    0   0    15     1907   1   12.590  0.839    0    0
 SERVICE2           15   15   0     9      449   1    3.000  0.200    0    0
 GDS               100   80   0    25     1184   1   17.344  0.173    0    0
```

Учтем, что количество обращений к сайту и сервисам может увеличиваться за счет сезонности. Летом клиенты бронируют намного больше билетов, нежели зимой. Увеличим интенсивность обращения к АСУ в два раза.

Листинг 5. Результаты работы моделирования при увеличении интенсивности запросов.

```
OK_TERMINATOR      33    TERMINATE          728             0       0
TERMINATOR         34    TERMINATE         2102             0       0


QUEUE              MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0)
RETRY
 QUEUE1              1    0   2999   2999     0.000      0.000      0.000   0
 SERVICE1_QUEUE      1    0   2138   2138     0.000      0.000      0.000   0
 GDS_QUEUE           1    0   2295   2295     0.000      0.000      0.000   0
 SERVICE2_QUEUE    109   99   2310     30    77.002     20.001     20.264   0


STORAGE            CAP. REM. MIN. MAX.  ENTRIES AVL.  AVE.C. UTIL. RETRY
DELAY
 SITE              100   81   0    27     2999   1   19.992  0.200    0    0
 SERVICE1           15    0   0    15     2138   1   14.101  0.940    0    0
 SERVICE2           15    0   0    15     2211   1   14.652  0.977    0   99
 GDS               100   79   0    57     2295   1   34.296  0.343    0    0
```

Как видно из листинга 4, транзакты теперь распределяются между сервисами. Очереди на каждом из сервисов (SERVICE1_QUEUE, (SERVICE2_QUEUE) меньше, чем

в предыдущем варианте архитектуры. На метку FAIL_TERMINATOR, означающую неудачное окончание процесса не пришло ни одного транзакта.

Однако, при увеличении интенсивности обращений в два раза, согласно листингу 5, удачными являются только 35% всех запросов на бронирование. Таким образом, можно говорить о том, что для того чтобы предупредить сезонное увеличение интенсивности следует добавить еще один сервис.

Вывод. Вариант архитектуры с двумя сервисам лучше варианта с одним сервисом, поскольку теперь запросы не отбрасываются из-за истечения таймаута. Бутылочным горлышком в данном случае являются сервисы бронирования, что видно из отчета STORAGE, эти ресурсы используются почти полностью - максимальное количество одновременно обрабатываемых запросов у первого сервиса 15, у второго 9. Это видно при увеличении интенсивности запросов в два раза. Таким образом, этот вариант архитектуры не отвечает всем выдвинутым требованиям к качеству функционирования АСУ.

Вариант архитектуры с тремя сервисами представлен на Рис. 5.

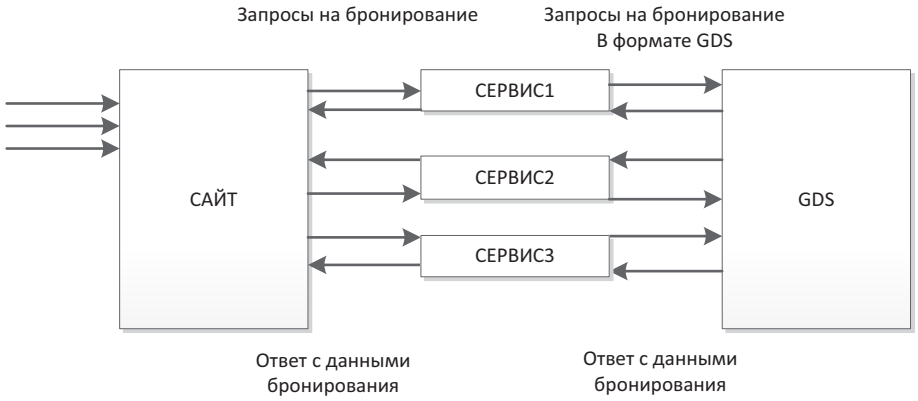

Рис. 5. Вариант архитектуры с тремя сервисами .

Промоделируем третий вариант архитектуры. Предыдущий вариант обеспечивал работу АСУ без потерянных запросов. Обращения к GDS теперь будут выполнять три сервиса. На сайт обращаются примерно 2 раза в секунду, сайт может обрабатывать до 100 запросов бронирования одновременно. Один сервис может обрабатывать 15 бронирований одновременно. Ограничение обусловлено техническими возможностями сервера, на котором располагается сервис. ПГС для варианта архитектуры с тремя сервисами представлен на Рис. 6.

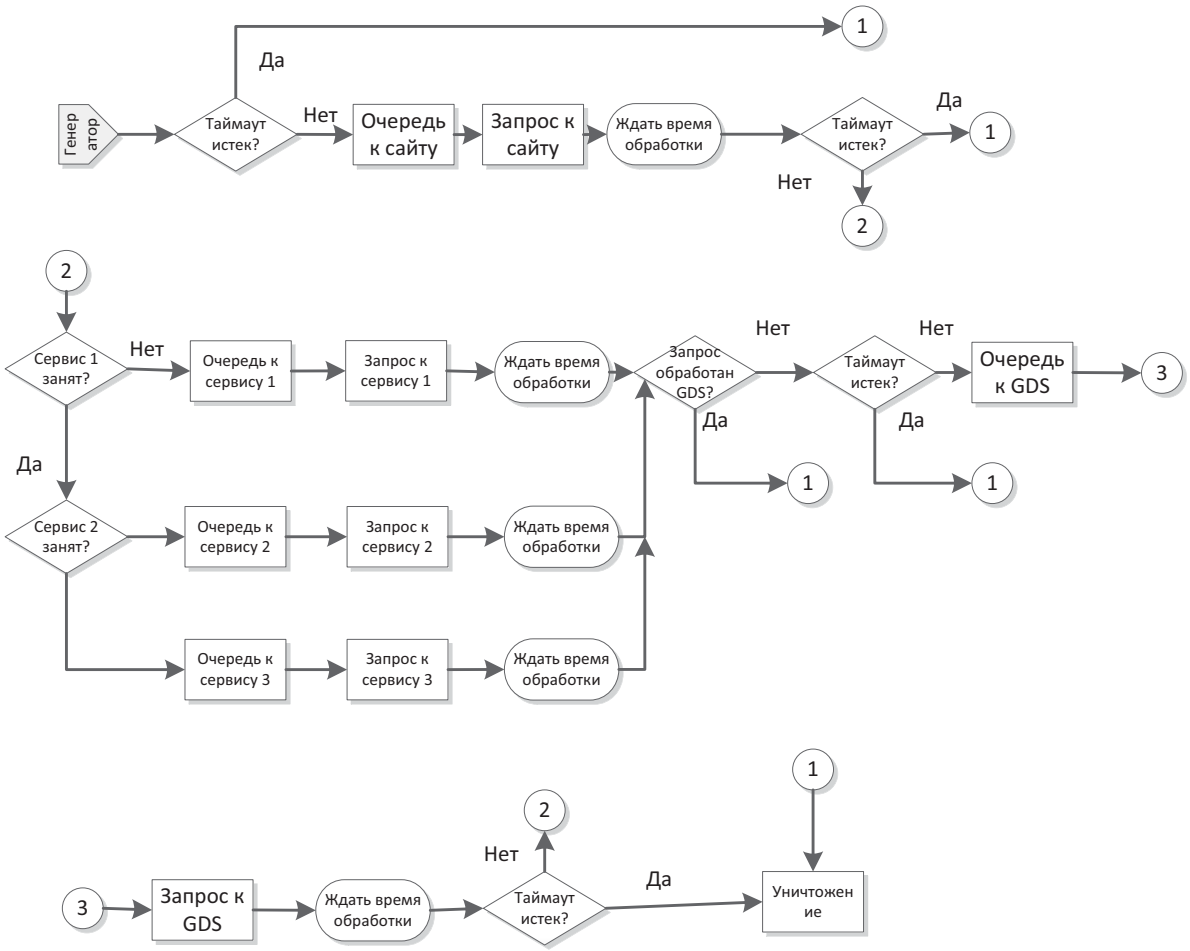

Рис. 6. ПГС для варианта архитектуры с тремя сервисами

Листинг 4. Моделирование АСУ компании-посредника с тремя сервисами.

```
SITE      STORAGE   100
SERVICE1  STORAGE   15
SERVICE2  STORAGE   15
SERVICE3  STORAGE   15
GDS STORAGE   100
GENERATE .2
ASSIGN 1,0

;САЙТ

QUEUE QUEUE1         ;Занять очередь к сайту
ENTER SITE           ;Занять сайт
DEPART QUEUE1
ADVANCE 4,2          ;Ждем, пока сайт обработает запрос
LEAVE SITE



;СЕРВИСЫ БРОНИРОВАНИЯ
;РАСПРЕДЕЛЕНИЕ



;СЕРВИС БРОНИРОВАНИЯ 1
BOOKING_FORK GATE SNF SERVICE1,NEXTFORK
```

http://sntbul.bmstu.ru/doc/593460.html

```
BOOKING_SERVICE1 QUEUE SERVICE1_QUEUE ;Занять очередь к сервису
ENTER SERVICE1       ;Занять сервис
DEPART SERVICE1_QUEUE
ADVANCE 4,2          ;Обработка запроса
LEAVE  SERVICE1      ;Освободить сервис
TEST LE M1,30,TERMINATOR;
TEST NE P1,1,OK_TERMINATOR
TRANSFER ,GDSMARK

;NEXTFORK TRANSFER .500,BOOKING_SERVICE2,BOOKING_SERVICE3
;NEXTFORK  GATE SNF SERVICE2,BOOKING_SERVICE3



;СЕРВИС БРОНИРОВАНИЯ 2
NEXTFORK  GATE SNF SERVICE2,BOOKING_SERVICE3

BOOKING_SERVICE2 QUEUE SERVICE2_QUEUE ;Занять очередь к сервису
ENTER SERVICE2       ;Занять сервис
DEPART SERVICE2_QUEUE
ADVANCE 4,2          ;Обработка запроса
LEAVE  SERVICE2      ;Освободить сервис
TEST LE M1,30,TERMINATOR;
TEST NE P1,1,OK_TERMINATOR
TRANSFER ,GDSMARK

;СЕРВИС БРОНИРОВАНИЯ 3

BOOKING_SERVICE3 QUEUE SERVICE3_QUEUE ;Занять очередь к сервису
ENTER SERVICE3       ;Занять сервис
DEPART SERVICE3_QUEUE
ADVANCE 4,2          ;Обработка запроса
LEAVE  SERVICE3      ;Освободить сервис
TEST LE M1,30,TERMINATOR;
TEST NE P1,1,OK_TERMINATOR
TRANSFER ,GDSMARK



;GDS - Глобальная система бронирования

GDSMARK QUEUE GDS_QUEUE
ENTER GDS           ;Занять GDS
DEPART GDS_QUEUE
ADVANCE 9,5         ; Обработка запроса
LEAVE  GDS          ; Освободить сервис 3
ASSIGN 1,1          ;Пометить запрос как обработанный

TEST LE M1,25,TERMINATOR;
TRANSFER ,BOOKING_FORK              ;Отправить ответ на один из сервисов



OK_TERMINATOR  TERMINATE
TERMINATOR TERMINATE
GENERATE 600
TERMINATE 1
```

Листинг 6. Результаты работы моделирования

```
OK_TERMINATOR      42    TERMINATE         2883             0       0
TERMINATOR         43    TERMINATE           11             0       0
```

```
QUEUE                MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0)
RETRY
 QUEUE1                1    0   3000   3000     0.000      0.000      0.000   0
 SERVICE1_QUEUE        1    0   2159   2159     0.000      0.000      0.000   0
 GDS_QUEUE             1    0   2963   2963     0.000      0.000      0.000   0
 SERVICE2_QUEUE        1    0   2029   2029     0.000      0.000      0.000   0
 SERVICE3_QUEUE       13    0   1696   1220     0.719      0.254      0.907   0

STORAGE              CAP. REM. MIN. MAX.  ENTRIES AVL.  AVE.C. UTIL. RETRY
DELAY
 SITE                100   79   0    27     3000   1   19.849  0.198    0    0
 SERVICE1             15    0   0    15     2159   1   14.349  0.957    0    0
 SERVICE2             15    4   0    15     2029   1   13.384  0.892    0    0
 SERVICE3             15    6   0    15     1696   1   11.175  0.745    0    0
 GDS                 100   50   0    57     2963   1   44.339  0.443    0    0
```

Как видно из листинга 6, на метку FAIL_TERMINATOR, означающую неудачное окончание процесса, приходит всего 11 транзактов.

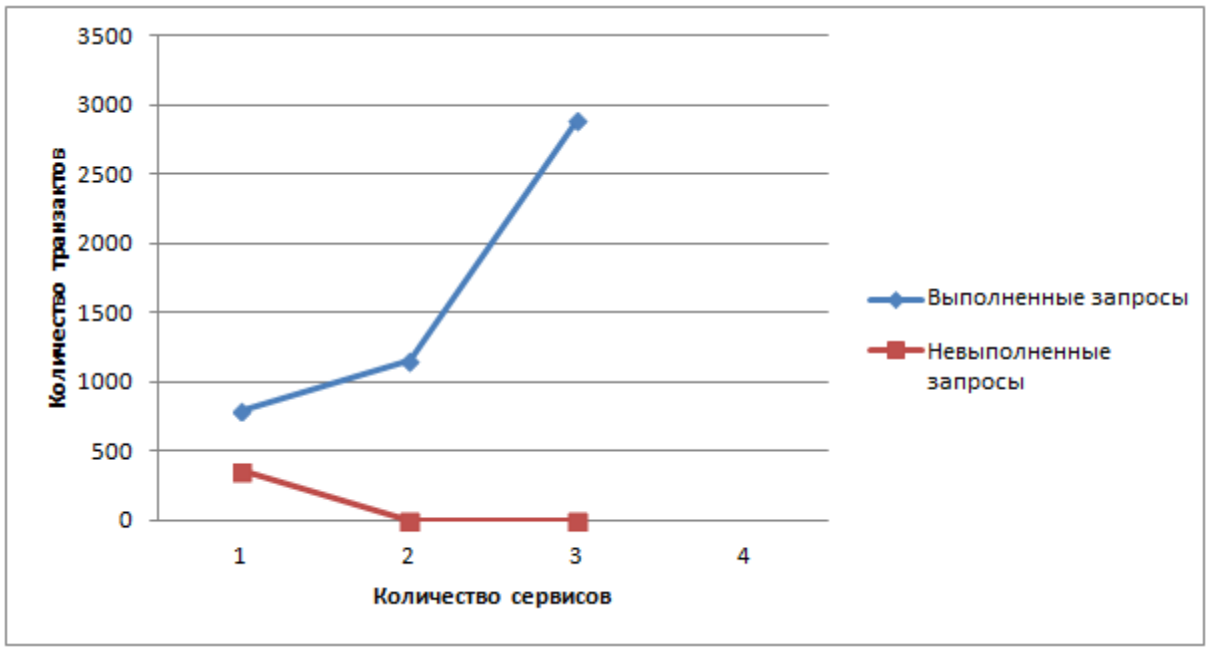

Рис. 7. Зависимость количества удачных и неудачных запросов от количества сервисов при интенсивности 2 запроса в секунду

Вывод. Теперь доля потерянных запросов к удачным составляет всего 0.3%.Третий вариант архитектуры является наиболее подходящим из всех рассмотренных, поскольку учитывает также двукратное увеличение интенсивности запросов. Графики, иллюстрирующие зависимость количества бронирований от количества сервисов представлены на Рис. 7

Заключение. В статье представлена имитационная модель, выполненная по методической схеме предварительного описания всех процессов функционирования АСУ на некотором псевдоязыке. Это позволило учесть особенности бизнес-процессов в

рассматриваемой системе. В результате моделирования найден наиболее подходящий вариант архитектуры для АСУ компании-посредника, который не только минимизирует количество не прошедших запросов, но и учитывает сезонные нагрузки на АСУ. Таким образом, задача решена.

Список литературы

1. В. Г. Олифер, Н. А. Олифер Компьютерные сети принципы, технологии, протоколы. Издательский дом Питер, 2005, 863 с.
2. Э. Танненбаум, М.ван Стеен. Распределенные системы принципы и парадигмы. Издательский дом Питер, 2003,876с.
3. Дж. Смит. Основы Windows Communication Foundation. Перевод на русский язык, БХВ-Петербург 2008, 375с.
4. Черненький В.М. Теоретические основы описания процессов функционирования дискретных систем // http://www.inforeg.ru ФГУП «Информрегистр». 2011 URL http: // iu5.bmstu/nir.php (Электронное учебное издание № 0321100676) , (дата обращения 28.12.11).
5. Черненький В.М. Адаптированное описание системы имитационного моделирования GPSS // http://www.inforeg.ru ФГУП «Информрегистр». 2011 URL http: // iu5.bmstu/nir.php (Электронное учебное издание № 0321100673) , (дата обращения 18.12.11).