Лабораторная работа

МОДЕЛЬ ОБРАБОТКИ ЗАПРОСОВ СЕРВЕРОМ

Постановка задачи

Сервер обрабатывает запросы, поступающие с автоматизированных рабочих мест с интервалами, распределенными по показательному закону со средним значением 2 мин. Время обработки сервером одного запроса распределено по экспоненциальному закону со средним значением 3 мин. Сервер имеет входной буфер ёмкостью 5 запросов.

Построить имитационную модель для определения математического ожидания времени и вероятности обработки запросов.

Сервер представляет собой однофазную систему массового обслуживания разомкнутого типа с ограниченной входной емкостью, то есть с отказами, и абсолютной надёжностью (рис. 1.1).

На рис 1.1 приведены также объекты AnyLogic, которые будут использованы для создания диаграммы процесса.

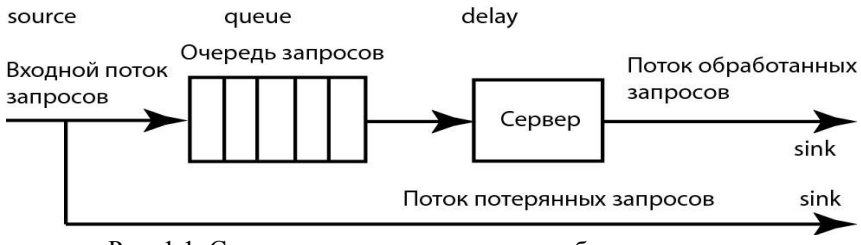

Рис. 1.1. Сервер как система массового обслуживания

Создание диаграммы процесса

1. Выполните **Файл/Создать/Модель** на панели инструментов. Появится диалоговое окно **Новая модель**
2. Задайте имя новой модели. В поле **Имя модели** введите `Server`.
3. Выберите каталог, в котором будут сохранены файлы модели.
4. Создайте диаграмму процесса. Для этого в Палитре выделите **Библиотеку моделирования процессов.** Из неё перетащите объекты на диаграмму и соедините, как показано на рис. 1.
5. При перетаскивании объектов вы можете видеть, как появляются соединительные линии между объектами.

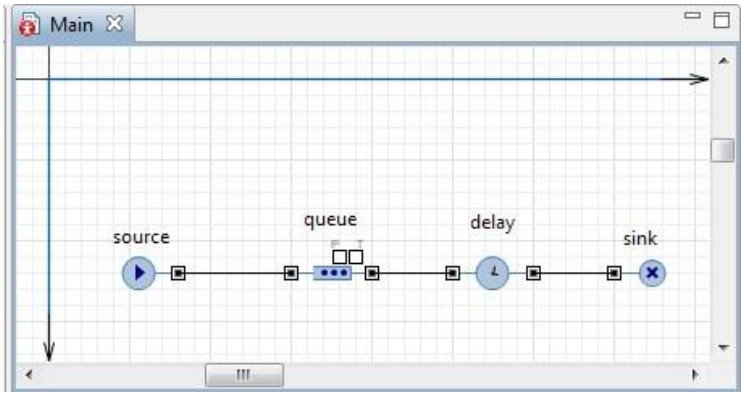

Рис. 1. Диаграмма системы массового обслуживания

Дадим краткую характеристику объектов диаграммы.

- Объект **Source** генерирует заявки определенного типа. Обычно он используется в качестве начальной точки диаграммы процесса, формализующей поток заявок. В нашем примере заявками будут запросы на обработку сервером, а объект Source будет моделировать их поступление.
- Объект **Queue** моделирует очередь заявок, ожидающих приема объектами, следующими за данным в диаграмме процесса. В нашем случае он будет моделировать очередь запросов, ожидающих освобождения сервера.
- Объект **Delay** задерживает заявки на заданный период времени. Он представляет в нашей модели сервер, обрабатывающий запросы.
- Объект **Sink** уничтожает поступившие заявки. Обычно он используется в качестве конечной точки потока заявок (и диаграммы процесса соответственно).

Изменение свойств блоков модели, её настройка и запуск

Помним, что мы хотим сначала создать простейшую модель, в которой будем рассматривать только обработку запросов сервером.

Всё, что нам нужно, чтобы сделать созданную диаграмму модели (см. рис. 1.) адекватной постановке задачи — это изменить некоторые свойства объектов.

Изменение свойств блоков диаграммы процесса

Первым объектом в диаграмме процесса является объект класса **Source.** Объект **source** генерирует заявки определенного типа. В нашем примере заявками будут *запросы на обработку данных*, а объект source будет моделировать поступление запросов на сервер.

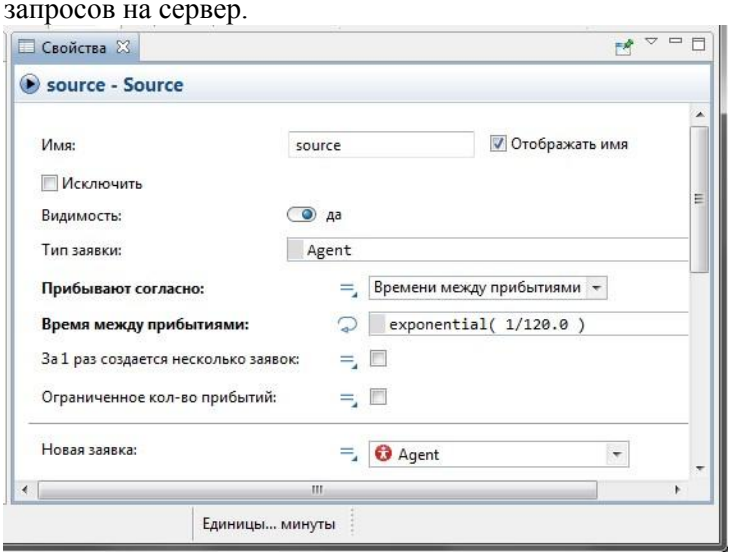

Рис. 2. Свойства объекта **source**

В нашем случае объект создает заявки через временной интервал, распределенный по показательному (экспоненциальному) закону со средним значением 2 мин.

Установим среднее время поступления запросов и среднее время их обработки в секундах.

1. Выделите объект source. В выпадающем списке **Прибывают согласно:** укажите, что запросы поступают согласно **Времени между прибытиями:** (рис. 2).
2. В поле **Время между прибытиями** появится запись exponential(1). Установите согласно постановке задачи среднее значение интервалов времени поступления запросов на сервер, изменив свойства объекта **source**. Для этого вместо характеристики распределения 1 введите `1/120.0`.

Следующий объект — **queue**. Выделите его. Он моделирует очередь заявок, ожидающих приема объектами, следующими за данным объектом в диаграмме процесса. В нашем случае он будет моделировать очередь запросов, ждущих освобождения сервера. Измените свойства объекта **queue** (рис. 3).

1. Задайте длину очереди. Введите в поле **Вместимость:** `5`. В очереди будут находиться не более 5 запросов.
2. Установите флажок **Включить сбор статистики (**Вкладка **Специфические)**, чтобы включить сбор статистики для этого объекта. В этом случае по ходу моделирования будет собираться статистика по количеству запросов в очереди.

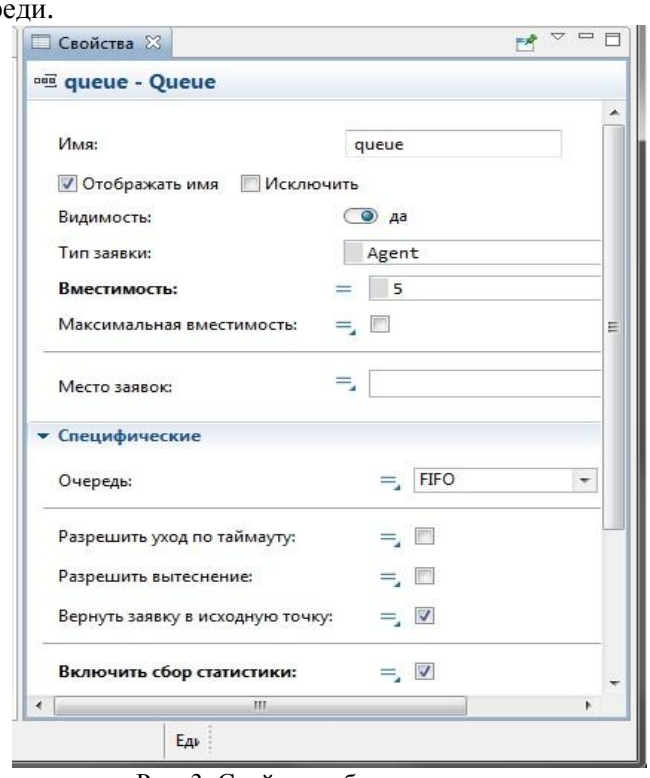

Рис. 3. Свойства объекта **queue**

Следующим в нашей диаграмме процесса расположен объект **delay**. Он задерживает заявки на заданный период времени, представляя в нашей модели непосредственно сервер, на котором обрабатываются запросы. Измените свойства объекта **delay** (рис. 4).

1. Обработка одного запроса занимает примерно 3 мин. Задайте время обслуживания, распределенное по экспоненциальному закону со средним значением 3 мин. Для этого введите в поле **Время задержки**: exponential(1/180.0).
2. Установите флажок **Включить сбор статистики**.

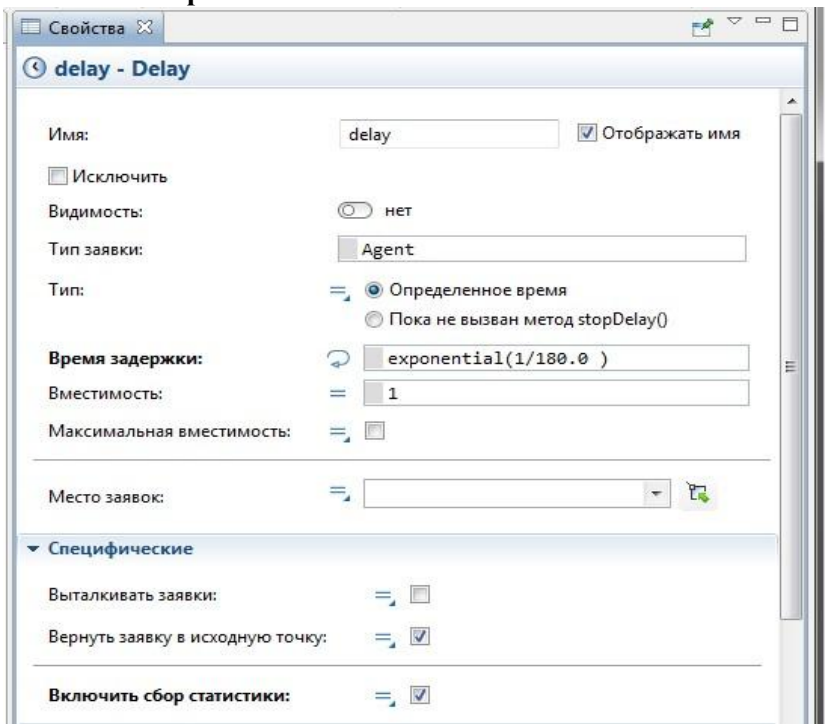

Рис. 4. Свойства объекта **delay**

Последним в диаграмме нашей дискретно-событийной модели находится объект **sink**. Этот объект уничтожает поступившие заявки. В нашем случае он выводит из модели обработанные сервером запросы.

Настройка запуска модели

1. В панели **Проект** выделите эксперимент **Simulation: Main**.
2. Щелчком раскройте вкладку **Модельное время**.
3. Установите **Виртуальное время (максимальная скорость)** (рис. 5).

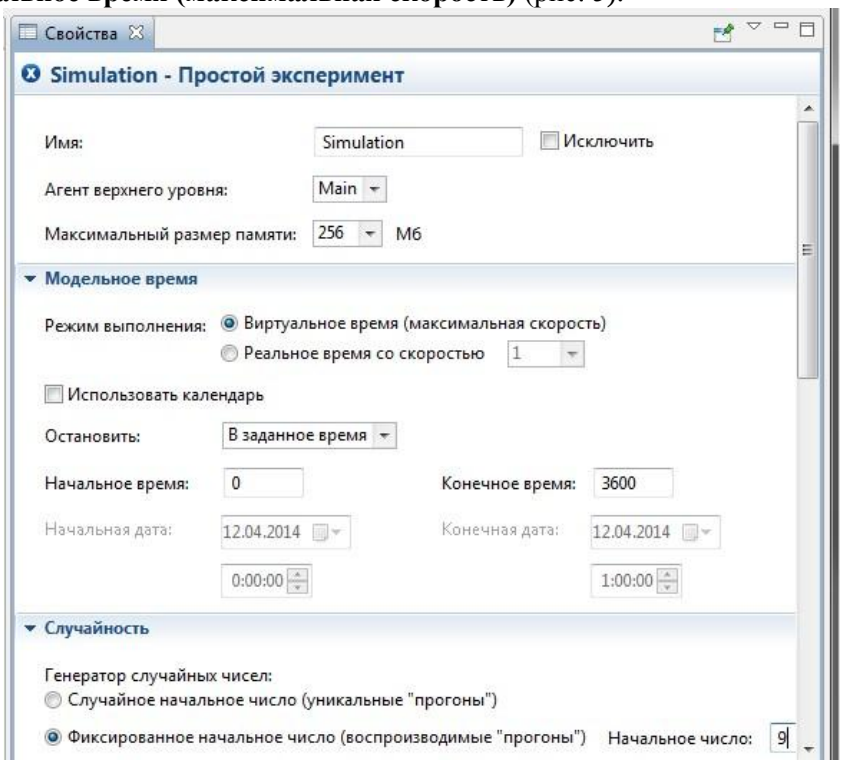

Рис. 5. Установка свойств эксперимента

4. В поле **Остановить:** выберите из списка `В заданное время`.
5. В поле **Конечное время:** установите `3600`.
6. Раскройте вкладку **Случайность**.
7. Выберите опцию **Фиксированное начальное число (воспроизводимые прогоны)**.
8. В поле **Начальное число:** установите `9`.
9. В панели **Проект**, выделите **Server** (рис. 6).

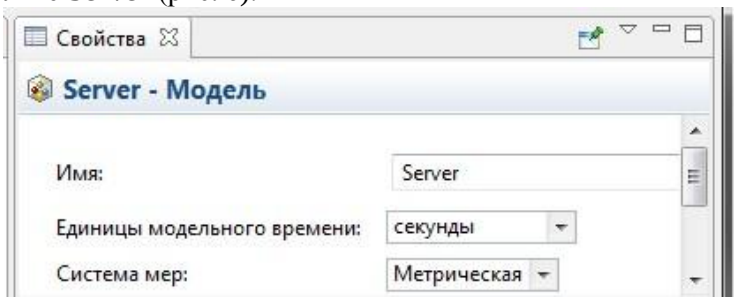

Рис. 6. Установка модельного времени

10. Из выпадающего списка **Единицы модельного времени:** выберите `секунды`.
11. Постройте вашу модель с помощью кнопки панели инструментов (F7). После построения модели, если обнаружены ошибки, исправьте их и запустите модель.
12. Для каждого объекта определены правила, при каких условиях принимать заявки. Некоторые объекты задерживают заявки внутри себя, некоторые — нет. Для объектов также определены правила: может ли заявка, которая должна покинуть объект, ожидать на выходе, если следующий объект не готов её принять. Если заявка должна покинуть объект, а следующий объект не готов её принять, и заявка не может ждать, то модель останавливается с ошибкой (рис. 7).

Рис. 7. Модель остановилась с ошибкой

13. Ошибка означает, что запрос не может покинуть объект source и войти в блок queue, так как его ёмкость, равная 5, заполнена. Также выдаётся сообщение о логической ошибке в модели (рис. 8)

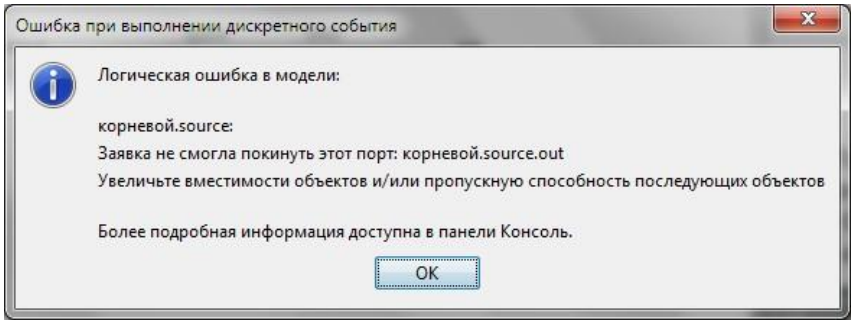

Рис. 8. Сообщение о логической ошибке в модели.

14. Нажмите OK. Далее измените свойства объекта **queue**, т. е. увеличьте длину очереди (см. рис. 4). Для этого введите в поле **Вместимость** `15`. Можете убедиться, что при увеличении ёмкости в пределах 6 … 14 модель по-прежнему останавливается с этой же ошибкой. Момент появления ошибки зависит от длительности времени моделирования.
15. Снова запустите модель.

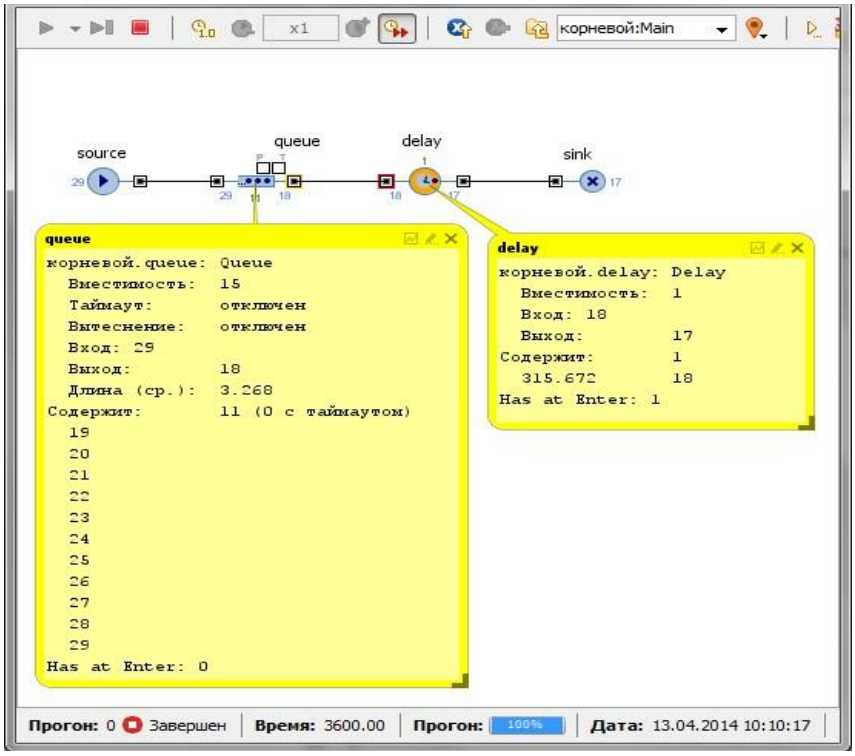

Рис. 9. Окно инспекта 8.

16. Вы можете следить за состоянием любого объекта диаграммы процесса во время выполнения модели с помощью окна инспекта этого объекта.
17. В окне инспекта будет отображена базовая информация по выделенному объекту: например, для объекта queue будут отображены вместимость очереди, количество заявок, прошедшее через каждый порт объекта и т. д. Такая же информация содержится в инспекте и для объекта delay (рис. 9).

Создание анимации модели

1. Нарисуйте прямоугольный узел, который будет обозначать на анимации сервер.
2. Откройте палитру **Разметка пространства** (рис. 10). Чтобы открыть какую-либо палитру, нужно щелчком из **Проекты** перейти в **Палитра** и щёлкнуть по иконке этой палитры.
3. Палитра **Разметка пространства** (рис. 10) содержит в качестве элементов различные примитивные фигуры, используемые для рисования презентаций моделей. Это путь, прямоугольный узел, многоугольный узел, точечный узел, аттрактор, стеллаж, масштаб.

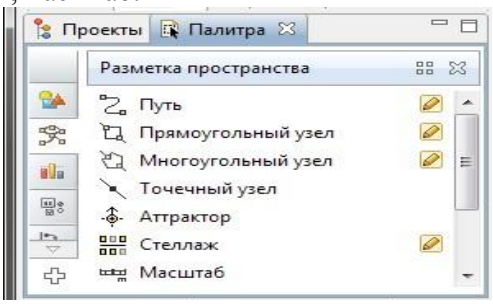

Рис. 10. Палитра **Разметка пространства**

4. Выделите элемент **Прямоугольный узел** и перетащите его на диаграмму класса активного объекта. Поместите элемент **Прямоугольный узел** так, как показано на рис. 11.

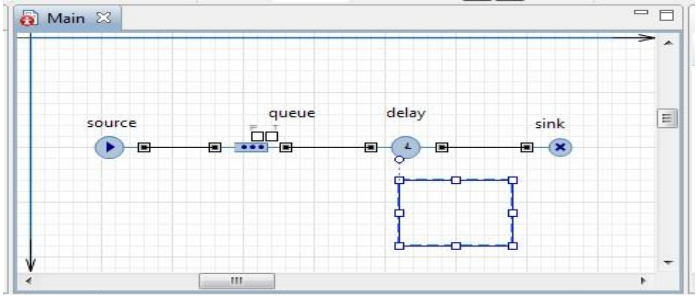

Рис. 11. Элемент **Прямоугольный узел** на диаграмме

5. Давайте сделаем так, чтобы цвет этого прямоугольного узла будет меняться в зависимости от того, обрабатывает ли сервер в данный момент времени запрос или нет.

6. Для этого выделите нарисованную нами фигуру на диаграмме. Перейдите на страницу **Внешний вид** панели свойств (рис. 12).

Мы хотим, чтобы во время моделирования менялся цвет нашей фигуры, поэтому щёлкните в поле **Цвет заливки:** по стрелке, выберите **Динамическое значение** и введите там следующую строку: `delay.size()>0?red:green`:

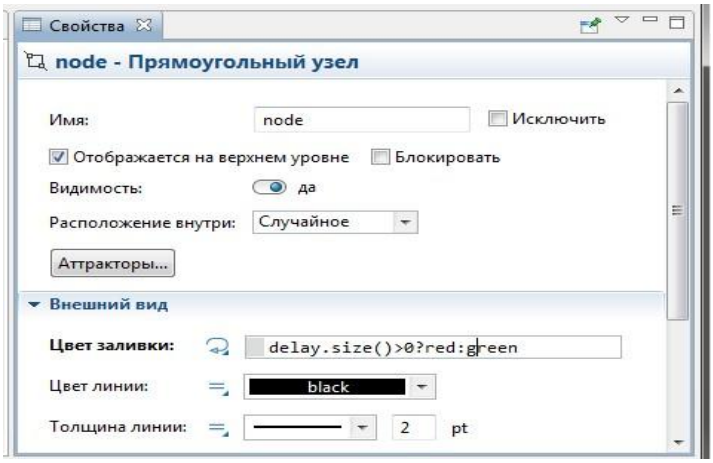

Рис. 12. Установлено динамическое значение цвета заливки

Здесь delay — это имя нашего объекта **delay.** Функция size() возвращает число запросов, обслуживаемых в данный момент времени. Если сервер занят, то цвет кружка будет красным, в противном случае — зелёным.

7. Нарисуйте путь, который будет обозначать на анимации очередь к серверу (рис. 13). Чтобы нарисовать путь, сделайте двойной щелчок мышью по элементу **Путь** палитры **Разметка пространства**, чтобы перейти в *режим рисования*. Очень важно, какую точку пути вы создаете первой. Заявки будут располагаться вдоль нарисованного вами пути в направлении от конечной точки к начальной точке. Поэтому обязательно начните рисование пути слева и поместите рядом с сервером конечную точку пути, которая будет соответствовать в этом случае началу очереди.

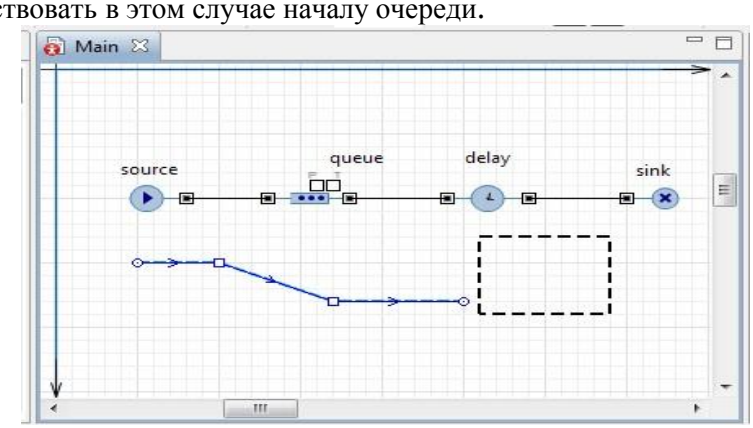

Рис. 13. Путь на диаграмме процесса

Теперь мы должны задать созданные анимационные объекты в качестве анимационных фигур объектов диаграммы нашего процесса. Задайте путь в качестве фигуры анимации очереди. Выделите объект queue. На странице свойств объекта queue в поле **Место заявок (агентов):** выберите из выпадающего списка `path` (рис. 14).

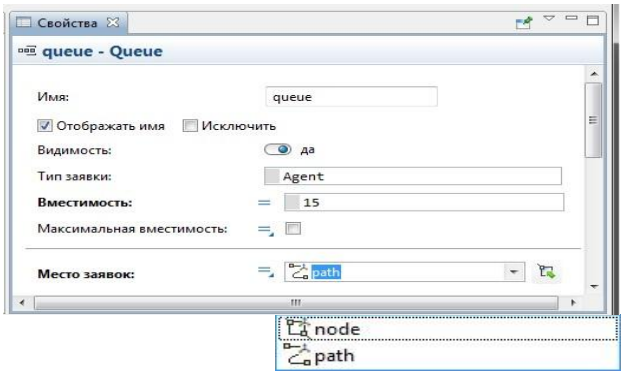

Рис. 14. Задание пути в качестве фигуры анимации очереди

8. Задайте прямоугольный узел в качестве фигуры анимации сервера. Выделите объект delay. Введите в поле **Место заявок (агентов):** из выпадающего списка имя нашего прямоугольного узла: `node` (рис. 15).

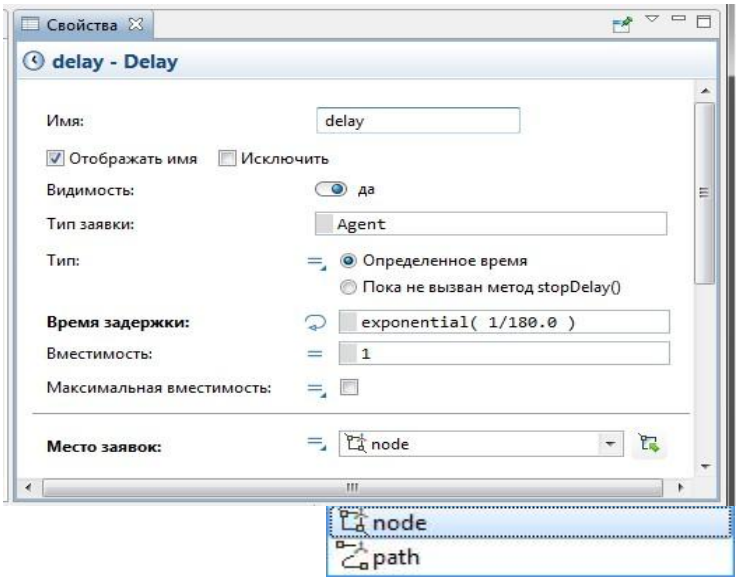

Рис. 15. Задание прямоугольного узла в качестве фигуры анимации сервера

Вы увидите, что у модели теперь есть простейшая анимация — сервер и очередь запросов к нему (рис. 16).

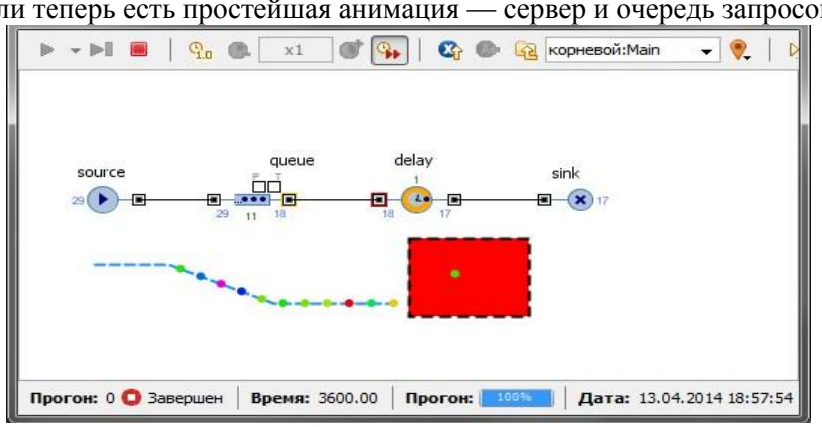

Рис. 16. Анимация модели 9. Запустите модель.

Цвет фигуры сервера будет меняться в зависимости от того, обрабатывается ли запрос в данный момент времени или нет.

Сбор статистики использования ресурсов

AnyLogic предоставляет пользователю удобные средства для сбора статистики по работе блоков диаграммы процесса. Объекты Enterprise Library самостоятельно производят сбор основной статистики. Все, что вам нужно сделать — это включить сбор статистики для объекта.

Поскольку мы уже сделали это для объектов **delay** и **queue**, то теперь мы можем, например, просмотреть интересующую нас статистику (скажем, статистику занятости сервера и длины очереди) с помощью диаграмм.

Добавьте диаграмму для отображения среднего коэффициента использования сервера:

1. Откройте палитру **Статистика**.
2. Перетащите элемент **Столбиковая диаграмма** из палитры **Статистика** на диаграмму класса и измените ее размер, как показано на рис. 17.

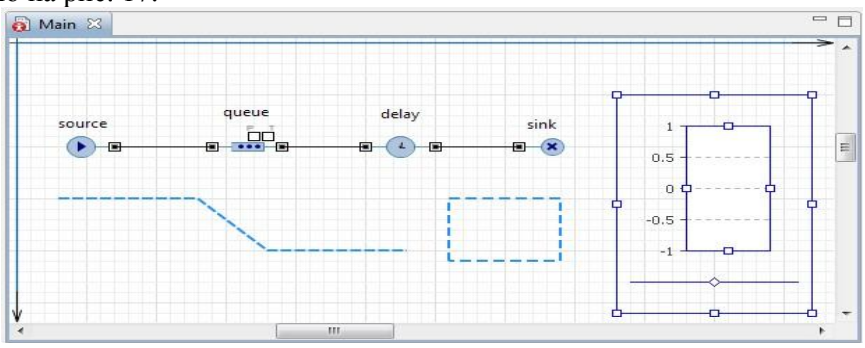

Рис. 17. Элемент **Столбиковая диаграмма** на диаграмме класса

3. Перейдите на панель **Свойства**. Щёлкните кнопку **Добавить элемент данных**. После щелчка появится секция свойств того элемента данных (**chart – Столбиковая диаграмма**), который будет отображаться на этой диаграмме (рис. 18).

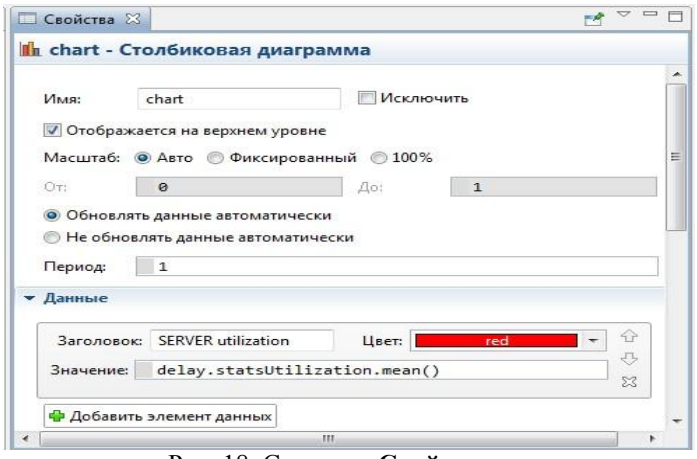

Рис. 18. Страница **Свойства**

4. Измените **Заголовок** на SERVER utilization.
5. Введите delay.statsUtilization.mean() в поле **Значение**. Здесь delay — это имя нашего объекта **delay**. У каждого объекта **delay** есть встроенный набор данных statsUtilization, занимающийся сбором статистики использования этого объекта. Функция mean() возвращает среднее из всех измеренных этим набором данных значений. Вы можете использовать и другие методы сбора статистики, такие, как min() или max().
6. Щёлкните **Внешний вид** (рис. 19). Установите свойства: направление столбцов, цвета фона, границ, меток, сетки, положение подписей у столбцов.

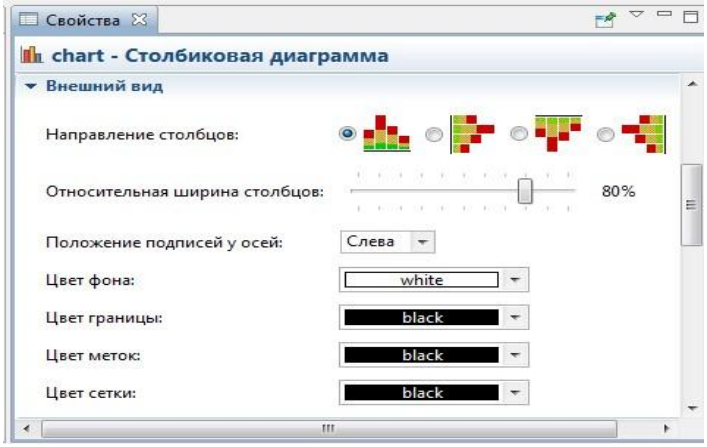

Рис. 19. Вкладка **Внешний вид**

7. Раскройте щелчками страницы (вкладки) **Местоположение и размер**, **Легенда**, **Область диаграммы** (рис. 20). Установите свойства, чтобы изменить расположение легенды относительно диаграммы (мы хотим, чтобы она отображалась внизу), размер диаграммы, высоту, ширину, координаты размещения на диаграмме, цвета текста, границы.

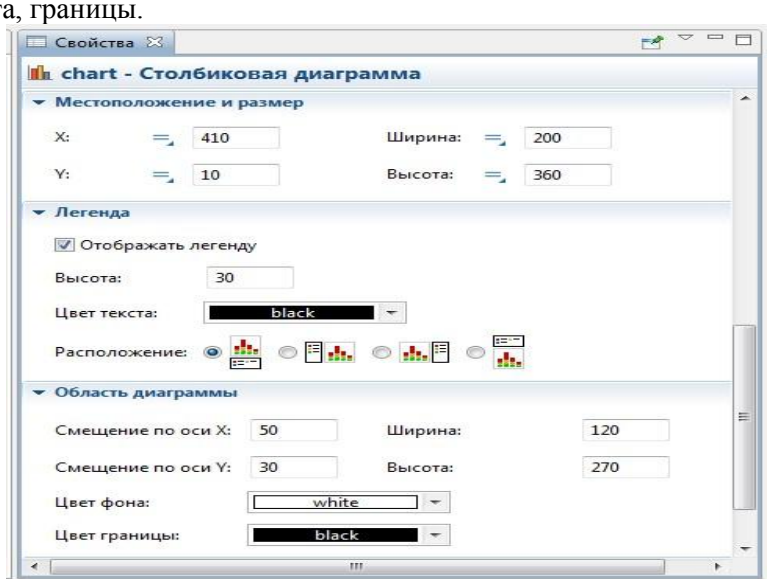

Рис. 20. Вкладки **Местоположение и размер**, **Легенда**, **Область диаграммы**

8. Аналогичным образом добавьте еще одну столбиковую диаграмму для отображения средней длины очереди.

9. На панели **Свойства** щёлкните **Добавить элемент данных**. После щелчка появится страница **Данные** свойств элемента данных (**chart1 – Столбиковая диаграмма**), который также будет отображаться на этой диаграмме (рис. 21).
10. **Заголовок:** и **Значение:** измените так, как показано на рис. 21. В поле **Заголовок:** введите `Queue lenqth`, а в поле **Значение:** введите `queue.statsSize.mean()`.

11. На страницах **Внешний вид**, **Местоположение и размер**, **Легенда**, **Область диаграммы** установите свойства самостоятельно. Столбцы диаграммы должны размещаться горизонтально.

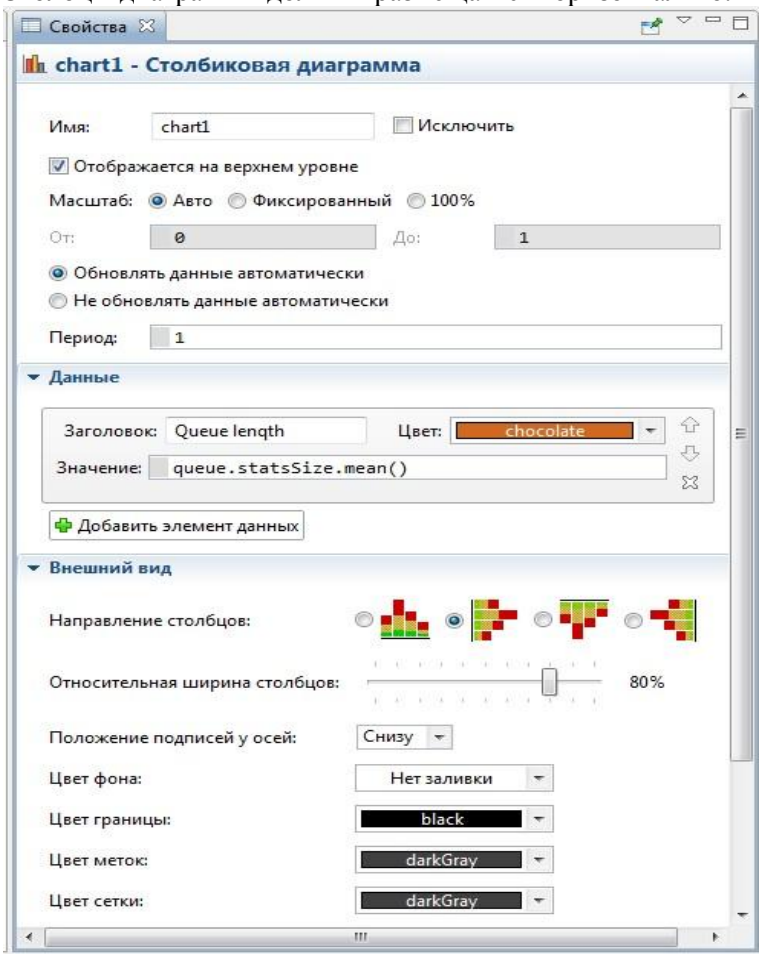

Рис. 21. Страницы **Данные**, **Внешний вид** панели **Свойства**

12. В поле **Значение:** queue — это имя нашего объекта **queue**. У каждого объекта **queue**, как и объекта **delay**, также есть встроенный набор данных statsSize, занимающийся сбором статистики использования этого объекта. Функция mean() также возвращает среднее из всех измеренных этим набором данных значений.

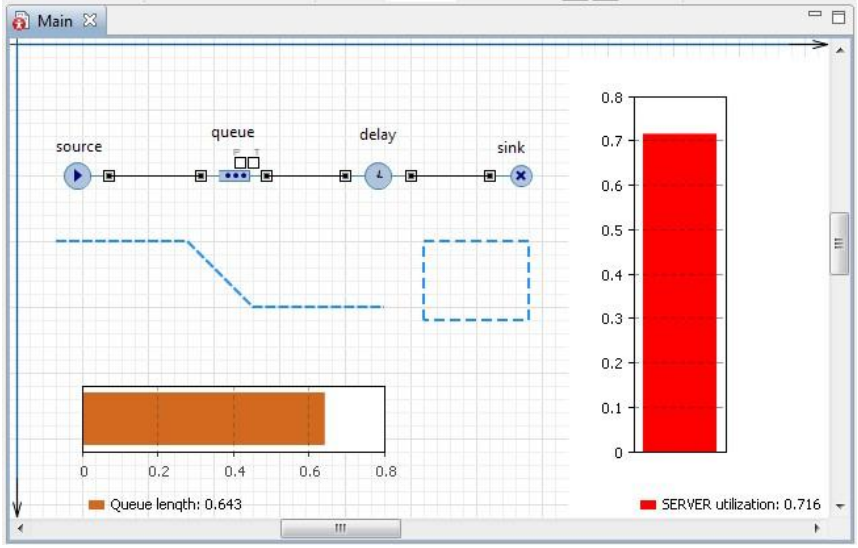

Рис. 22. Добавлены две столбиковые диаграммы

13. На странице **Внешний вид** панели **Свойства** выберите в секции свойств **Направление столбцов** вторую опцию (рис. 21), чтобы столбцы во второй столбиковой диаграмме, расположенной горизонтально, росли вправо (рис. 22).

14. Запустите модель с двумя столбиковыми диаграммами, установив модельное время 3600 единиц, и понаблюдайте за её работой (рис. 23).

Уточнение модели согласно ёмкости входного буфера

На рис. 23 (снимок сделан по окончании времени моделирования) видно, что длина очереди равна 14 запросам при установленной максимальной длине 15. Но ведь в постановке задачи ёмкость буфера была определена в 5 запросов. Нам не удалось до этого построить модель с такой ёмкостью из-за ошибки (см. рис. 7) — невозможности очередного запроса покинуть блок source, так как длина очереди уже была равна 5 запросам. Нам пришлось во избежание этой ошибки увеличить ёмкость буфера до 15 запросов.

Уточним модель согласно постановке задачи.

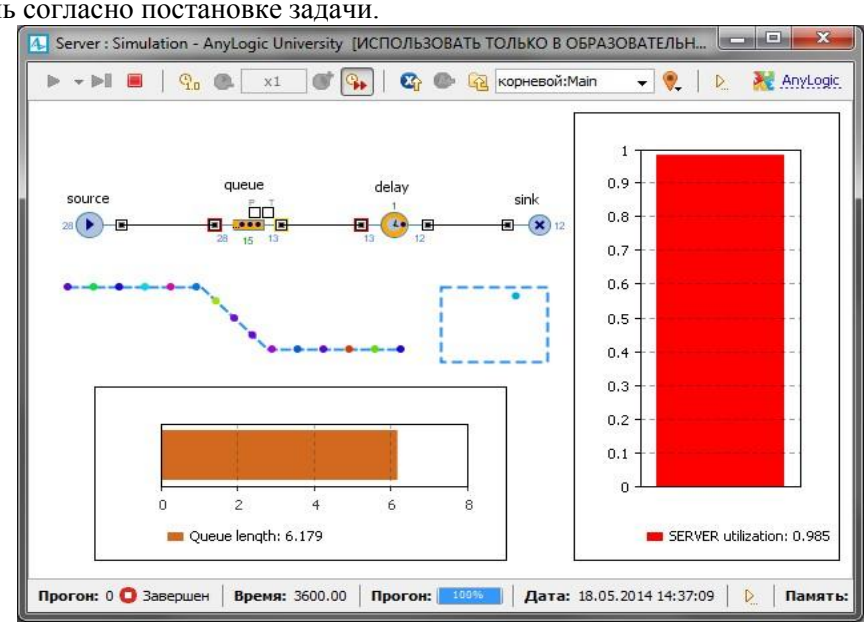

Рис. 23. Наблюдение за моделью с двумя столбиковыми диаграммами

Для выполнения условия постановки задачи воспользуемся следующим способом. Все запросы, вырабатываемые объектом source, имеют один и тот же приоритет. Поэтому при полном заполнении накопителя (5 запросов) теряться будет последний запрос. Уточните модель.

1. Выделите объект queue. На панели **Свойства** измените **Вместимость** с 15 на 5 запросов.
2. Здесь же установите **Разрешить вытеснение**.
3. Для уничтожения потерянных запросов вследствие полного заполнения накопителя нужно добавить второй объект sink. Откройте в **Палитре Библиотеку моделирования процессов** и перетащите блок sink на диаграмму (рис. 24). При перетаскивании объект пытается автоматически соединиться с входами имеющимися на диаграмме объектами. Но это может вас не устраивать.
4. Тогда соедините порт outPreempted объекта queue с входным портом InPort блока sink1. Чтобы соединить порты, сделайте двойной щелчок мышью по одному порту, например, outPreempted, затем последовательно Щёлкните в тех местах диаграммы, где вы хотите поместить точки изгиба соединителя.

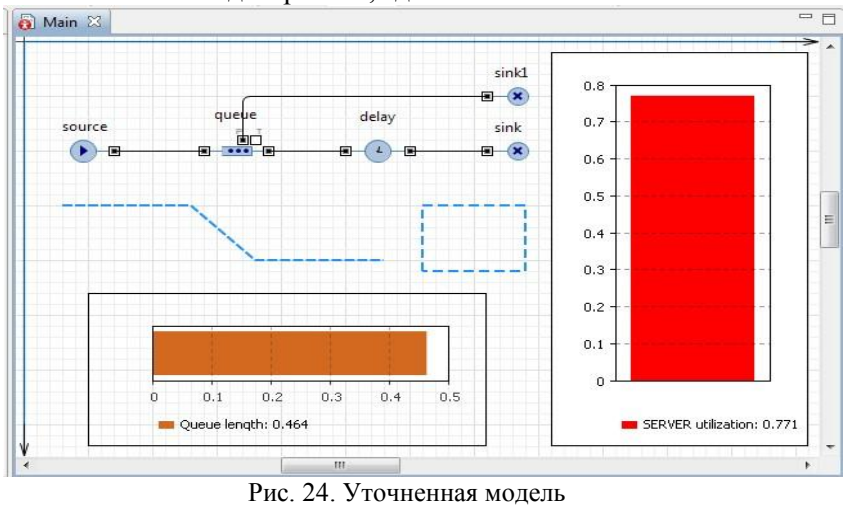

Рис. 24. Уточненная модель

5. Запустите уточненную модель и понаблюдайте за ее работой. Сравните рис. 25 с рис. 17. На рис. 25 видно, что запросы при длине очереди в 5 запросов теряются, и ошибки при этом не возникает. Модель по ограничению ёмкости входного буфера и значениям других параметров соответствует постановке задачи.

Однако согласно постановке задачи требуется определить математическое ожидание времени обработки одного запроса и математическое ожидание вероятности обработки запросов.

Сбор статистики по показателям обработки запросов

Entity (заявка) являются базовым классом для всех заявок, которые создаются и работают с ресурсами в процессе, описанном вами с помощью диаграммы из объектов **Библиотеки моделирования процессов**. Согласно постановке задачи нужно определять математическое ожидание времени и вероятности обработки запросов сервером.

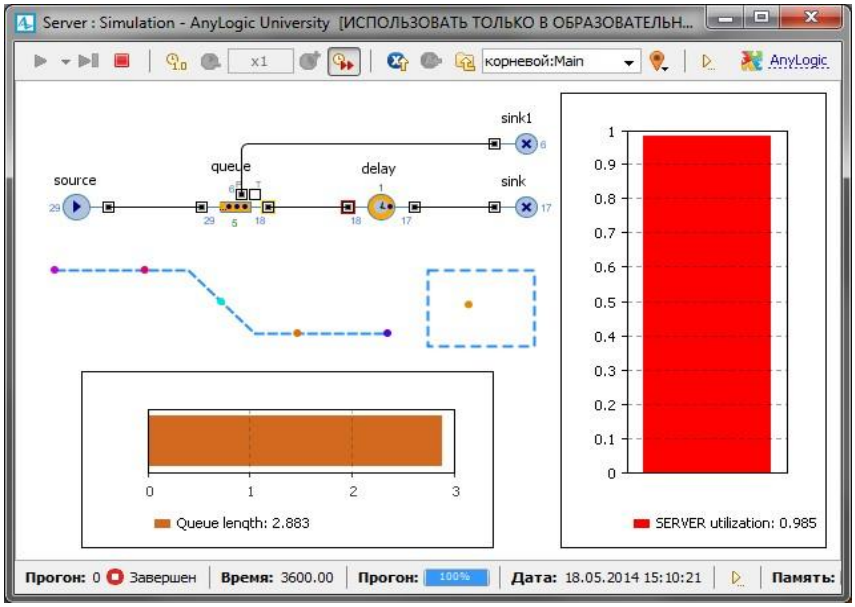

Рис. 25. Работа модели согласно ёмкости входного буфера

Математическое ожидание или среднее время обработки одного запроса определяется как отношение суммарного времени обработки *n* запросов к их количеству, т. е. к *n*. Для определения суммарного времени нужно знать время обработки *i*-го запроса. Для этого введем дополнительные поля:

`time_vxod` – время входа запроса в буфер сервера,

`time_vixod` – время выхода запроса с сервера (входа в блок sink).

Тогда `time_obrabotki=time_vixod-time_vxod`

Вероятность обработки запросов сервером определяется как отношение количества обработанных запросов к количеству всех поступивших запросов. Значит, нужно вести счет запросов на выходе источника запросов и на выходе с сервера (входе в блок sink). Для этого также введем дополнительные поля: `col_vxod` — количество поступивших всего запросов, `col_vixod` — количество обработанных сервером запросов.

Тогда `ver_obrabotki=col_vixod/col_vxod`

Создание нестандартного Java класса

Для включения в запросы дополнительных полей необходимо создать нестандартный тип заявки. Это возможно двумя способами. Создадим первым способом тип заявок `Inquiry`.

1. В панели **Проект** щёлкните правой кнопкой мыши элемент модели верхнего уровня дерева и выберите из контекстного меню **Создать/Java класс**.
2. Появится диалоговое окно **Новый Java класс** (рис. 26). В поле **Имя:** введите имя нового класса `Inquiry`.
3. В поле **Базовый класс:** выберите из выпадающего списка `Entity` в качестве базового класса. Щёлкните кнопку **Далее**.

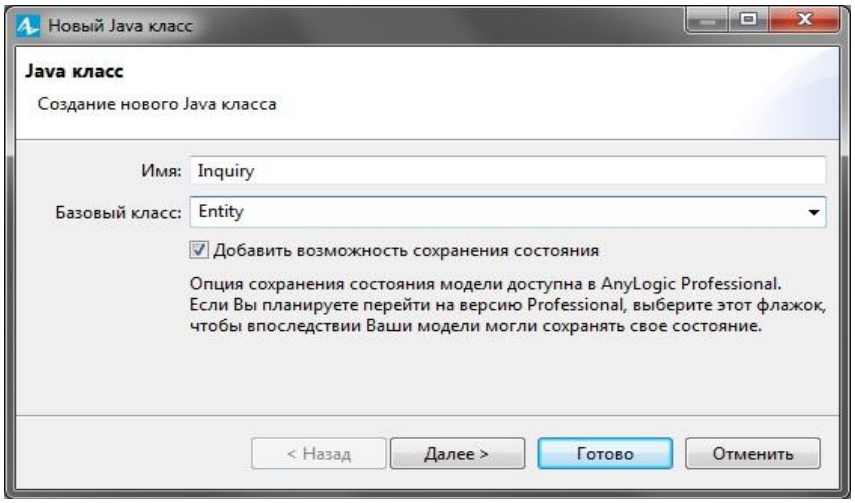

Рис. 26. Диалоговое окно создания нового Java класса 4.

Появится вторая страница **Мастера создания Java класса.** (рис. 27).

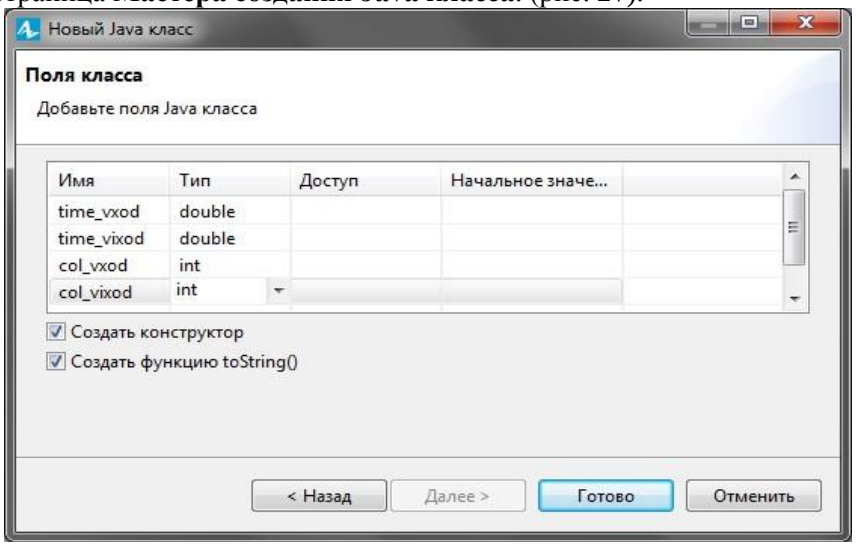

Рис. 27. Вторая страница **Мастера создания Новый Java класс**

4. Добавьте поля Java класса: `time_vxod` типа `double`, `time_vixod` типа `double`, `col_vxod` типа `int`, `col_vixod` типа `int`. Типы полей выбираются из выпадающего списка.

Начальные значения всех параметров, поскольку не указаны, по умолчанию будут установлены равными нулю.

5. Оставьте выбранными флажки **Создать конструктор** и **Создать метод toString ()**. Тогда у класса будут созданы сразу два конструктора: один, по умолчанию, без параметров, и второй, с параметрами, инициализирующими поля класса. Эти конструкторы используются объектами, создающими новые заявки, такие, как Source.
6. Щёлкните кнопку **Готово**. Вы увидите редактор кода, в котором будет показан автоматически созданный код вашего Java класса (рис. 28). Закройте редактор.
7. Теперь нужно преобразовать Java класс в тип агента. Для этого щёлкните правой кнопкой мыши в панели **Проект** только что созданный Java класс и в контекстном меню выберите **Преобразовать Java класс в тип агента**.
8. Появится окно с автоматически созданными параметрами нестандартного типа заявок Inquiry (см. рис. 33).

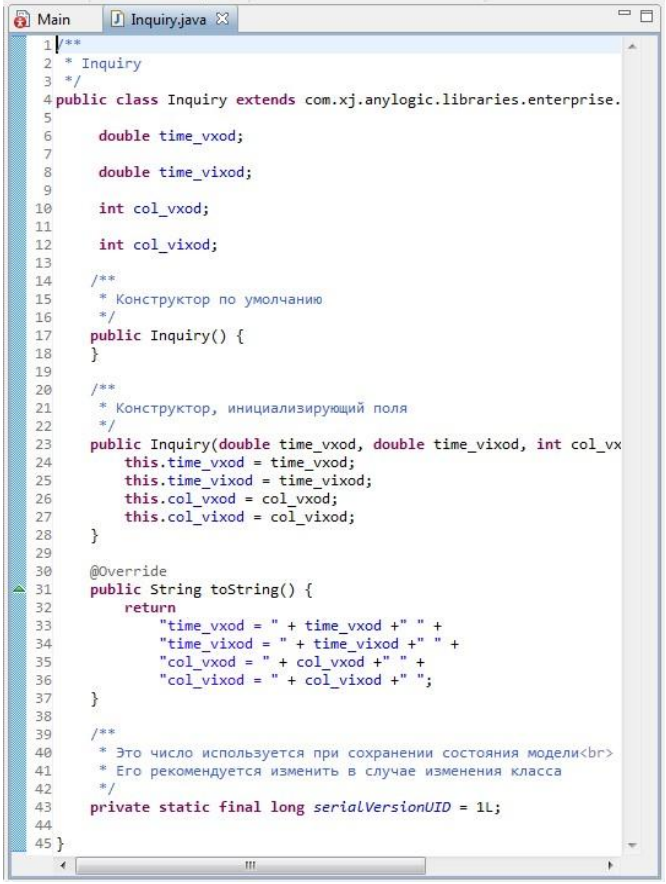

Рис. 28. Окно редактора кода созданного нестандартного класса

9. Закройте окно, щелкнув крестик в закладке рядом с его названием. Перейдём к созданию непосредственно нестандартного типа заявки.

Создайте тип заявок `Inquiry`.

1. Откройте палитру **Библиотека моделирования процессов**.
2. Перетащите элемент **Тип заявки** в графический редактор. Появится тип заявки `Entity` (рис. 29).

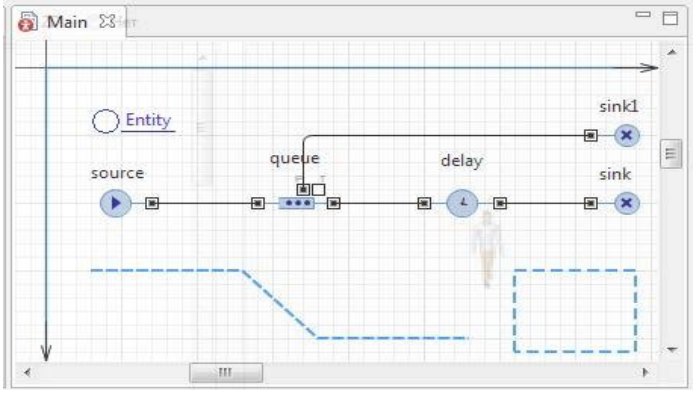

Рис.29. Появился тип заявки `Entity`

3. Появится диалоговое окно **Создание агента** (рис. 30).

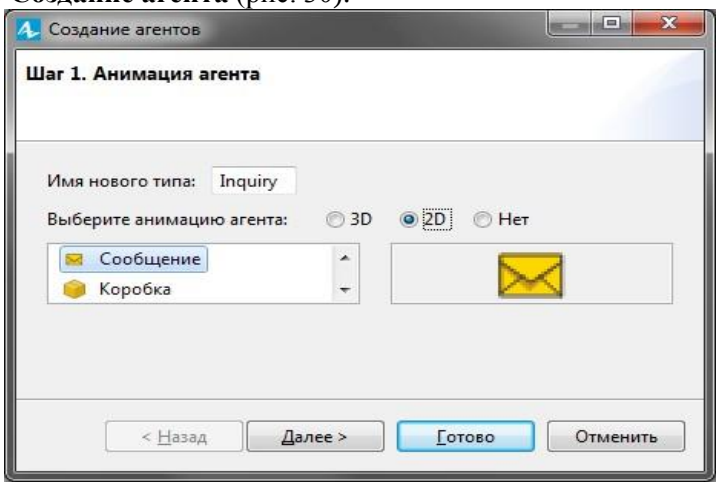

Рис. 30. Диалоговое окно **Создание агента.**

Шаг 1. Анимация агента.

В поле **Имя нового агента:** введите `Inquiry`. Выберите анимацию агента: установите 2D и выберите из выпадающего списка, например, **Сообщение**. Щёлкните **Далее.**

4. Появится диалоговое окно **Создание агента.**

Шаг 2. Параметры агента (рис. 31).

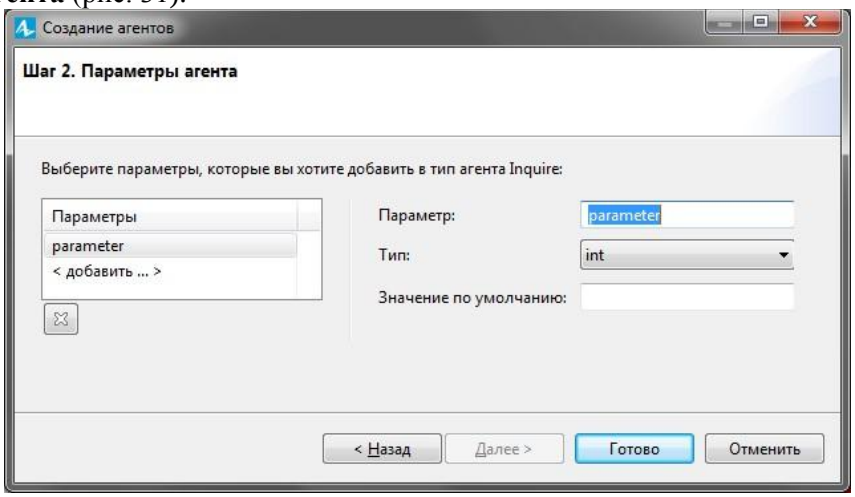

Рис. 31. Диалоговое окно **Создание агента. Шаг 2. Параметры агента**

5. Щёлкните **<добавить…>**. В поле **Параметр:** введите `time_vxod` (рис. 32).

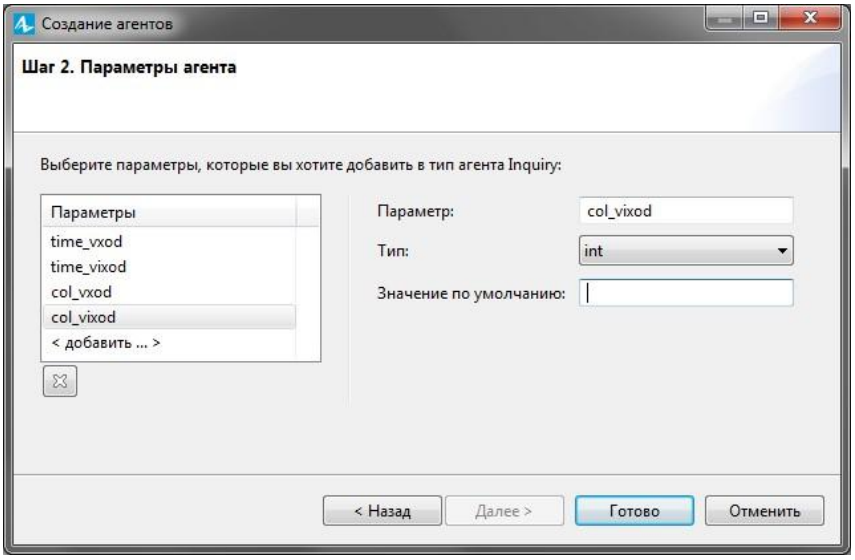

Рис. 32. Диалоговое окно **Создание агента.**

6. Из выпадающего списка **Тип:** выберите `double`.
7. Щёлкните второй раз **<добавить…>**. В поле **Параметр:** введите `time_vixod.`
8. Из выпадающего списка **Тип:** выберите `double`.
9. Щёлкните третий раз **<добавить…>**. В поле **Параметр:** введите `col_vxod.` Из выпадающего списка **Тип:** оставьте `int`.

Шаг 1. Анимация агента

1. Щёлкните третий раз **<добавить…>**. В поле **Параметр:** введите `col_vixod.`
2. Из выпадающего списка **Тип:** оставьте `int`.

Шаг 2. Параметры агента с установленными параметрами нестандартного типа заявок Inquiry

3. Так как в поле **Значение по умолчанию** мы не устанавливали никаких значений, то всем параметрам будет установлен 0.

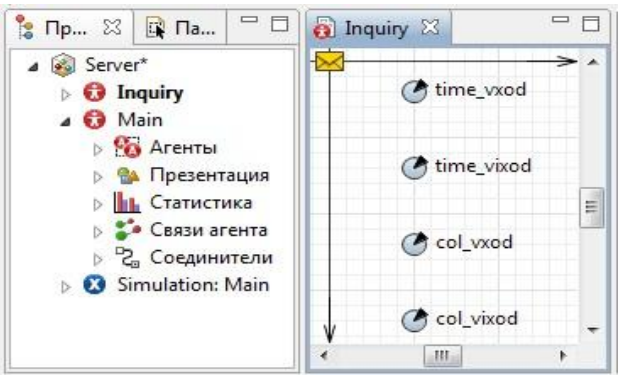

Рис. 33. Окно с параметрами нестандартного типа заявки Inquiry

4. Щёлкните кнопку **Готово**. Вы увидите окно, в котором будут показаны автоматически созданные параметры нестандартного типа заявок Inquiry (рис. 33). Закройте оно, щелкнув крестик в закладке рядом с его названием.

Добавление элементов статистики

Для сбора статистических данных о времени обработки запросов сервером необходимо добавить элемент статистики. Этот элемент будет запоминать значения времен для каждого запроса. На основе этого он предоставит пользователю стандартную статистическую информацию (среднее, минимальное, максимальное из измеренных значений, среднеквадратичное отклонение и т.д.).

1. Чтобы добавить элемент сбора данных гистограммы на диаграмму, перетащите элемент **Данные гистограммы** с палитры **Статистика** на диаграмму активного класса.
2. Задайте свойства элемента (рис. 34): измените **Имя:** на `time_obrabotki`; сделайте **Кол-во интервалов:** равным `50`; задайте **Нач. размер интервала:** `0.01`.

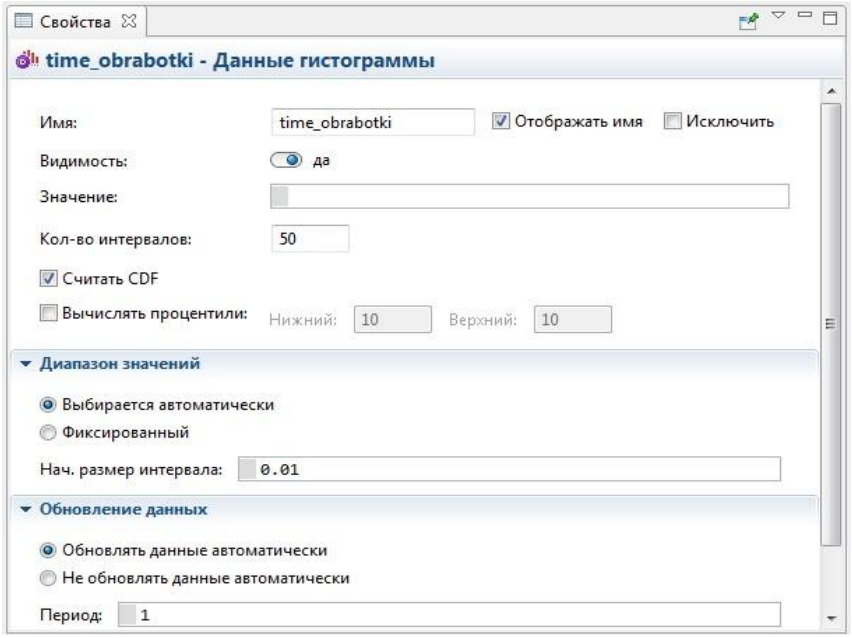

Рис. 34. Элемент сбора статистики о времени обработки запросов

Добавьте еще элемент сбора статистики для определения вероятности обработки запросов.

1. Перетащите элемент **Данные гистограммы** с палитры **Статистика** на диаграмму активного класса.
2. Задайте свойства элемента (рис. 35): измените **Имя:** на `ver_obrabotki`; сделайте **Кол-во интервалов:** равным `50`; задайте **Нач. размер интервала:** `0.01`.

Диаграмма после добавления элементов сбора статистики представлена на рис. 36.

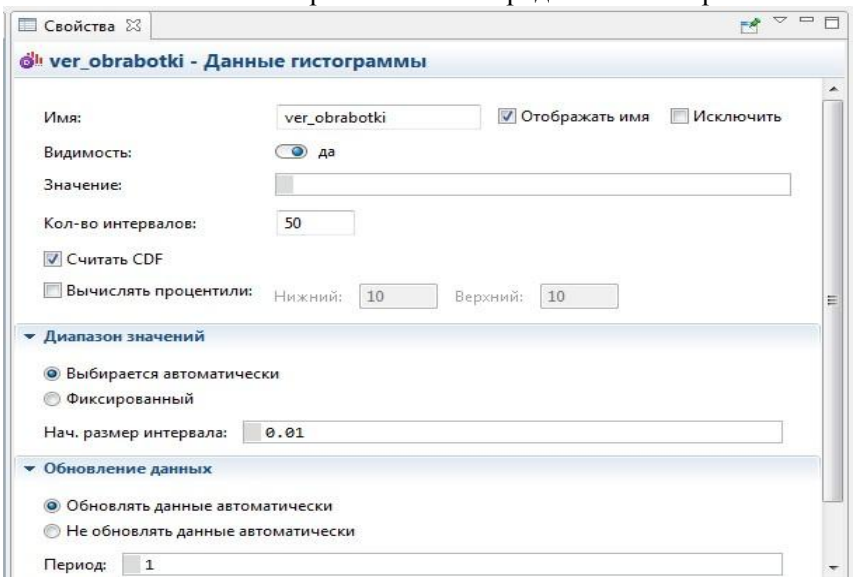

Рис. 35. Элемент сбора статистики о вероятности обработки запросов

Изменение свойств объектов диаграммы

Чтобы создавать заявки нестандартного типа, как в нашем случае `Inquiry`, вам нужно поместить вызов конструктора этого типа в поле **Новая заявка** объекта source. Но, несмотря на то, что заявки в потоке теперь и будут типа `Inquiry`, остальные объекты диаграммы будут продолжать их считать заявками типа `Entity`.

Поэтому они не позволят явно обращаться к дополнительным полям класса `Inquiry`. Чтобы разрешить доступ к полям вашего нестандартного типа заявки в коде динамических параметров объектов потоковой диаграммы, вам нужно указать имя нестандартного типа заявки в качестве **Типа заявки** этого объекта. В нашей потоковой диаграмме с учётом блока **source** всего пять объектов.

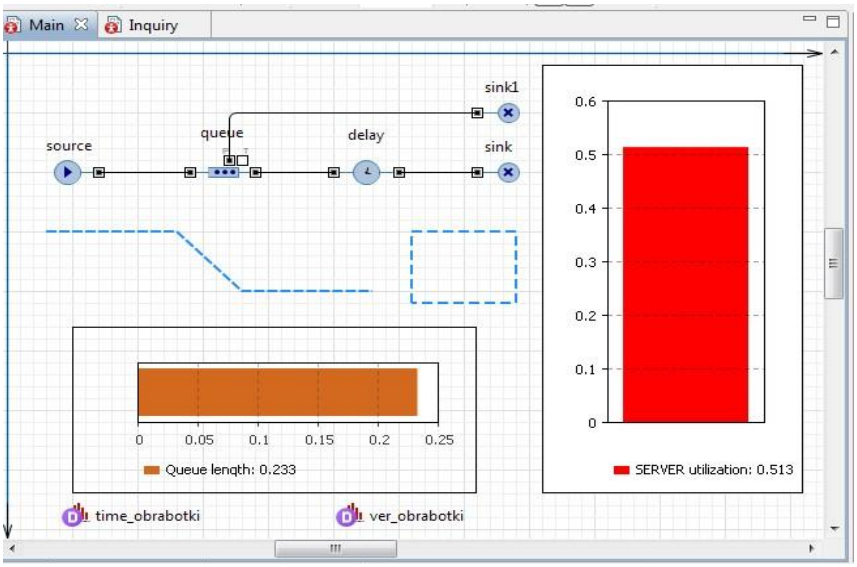

Рис. 36. Диаграмма после добавления элементов сбора статистики

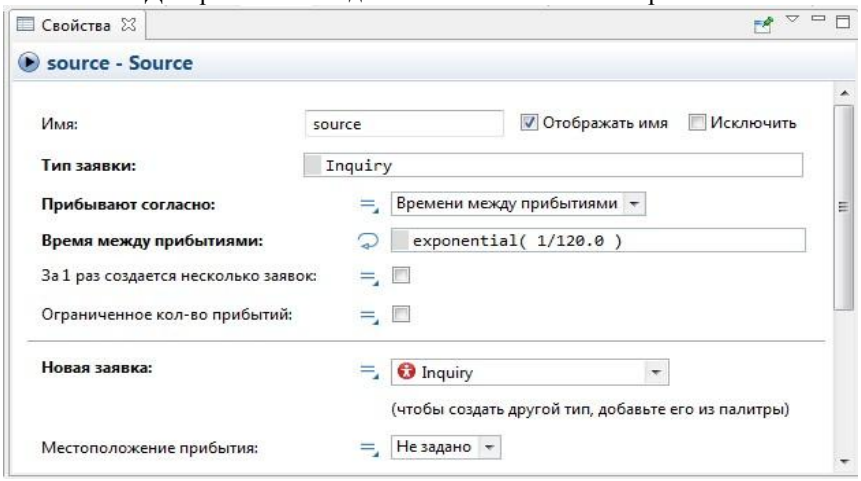

Рис. 37. Объект source с изменёнными свойствами Измените их свойства.

1. Измените свойства объекта **source** (рис. 37): введите `Inquiry` в поле **Тип заявки:**. Это позволит напрямую обращаться к полям типа заявки `Inquiry` в коде динамических параметров этого объекта; выберите из выпадающего списка `Inquiry()` в поле **Новая заявка:**. Теперь этот объект будет создавать заявки нашего типа `Inquiry`; введите `entity.time_vxod=time();` в поле **Действия При выходе:**. Код будет сохранять время создания заявки-запроса в параметре `time_vxod` нашего типа заявки `Inquiry`. Функция `time()` возвращает текущее значение модельного времени.
2. Измените свойства объекта queue: введите `Inquiry` в поле **Тип заявки:**.
3. Измените свойства объекта delay: введите `Inquiry` в поле **Тип заявки:**.
4. Измените свойства объекта sink1: введите `Inquiry` в поле **Тип заявки:**.
5. Измените свойства объекта sink: введите `Inquiry` в поле **Тип заявки:**; введите в поле **Действие при входе** следующие коды: `time_obrabotki.add(time()-entity.time_vxod);`

Этот код добавляет время обработки одного запроса в объект сбора данных гистограммы `time_obrabotki`. Данное время определяется как разность между текущим модельным временем `time()` и временем входа запроса в модель. `add` — встроенная функция добавления элемента в массив. `entity.col_vixod=sink.count(); entity.col_vxod=source.count();`

Эти коды заносят количество запросов, вошедших в блок sink и вышедших из блока source соответственно. `count()` — встроенная функция этих блоков, возвращает количество вошедших в блок sink и количество вышедших из блока source заявок.

`ver_obrabotki.add(entity.col_vixod/entity.col_vxod);`

Этот код добавляет относительную долю обработанных запросов в объект сбора данных гистограммы `ver_obrabotki` при поступлении каждого обработанного запроса в блок sink. На основе множества таких относительных долей определяется математическое ожидание вероятности обработки запросов сервером.

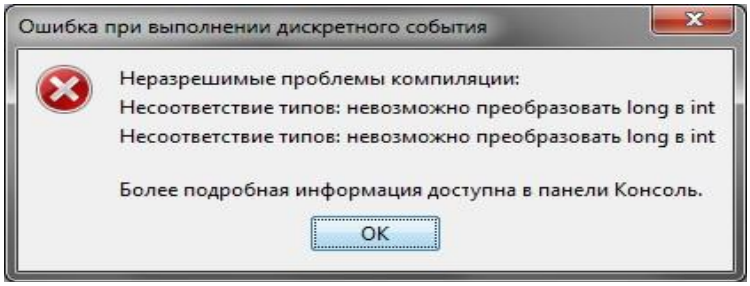

Рис. 38. Второе сообщение об ошибке

6. Запустите модель. Появится сообщение об ошибке. Щёлкните **Продолжить**. Появится второе сообщение (рис. 38).

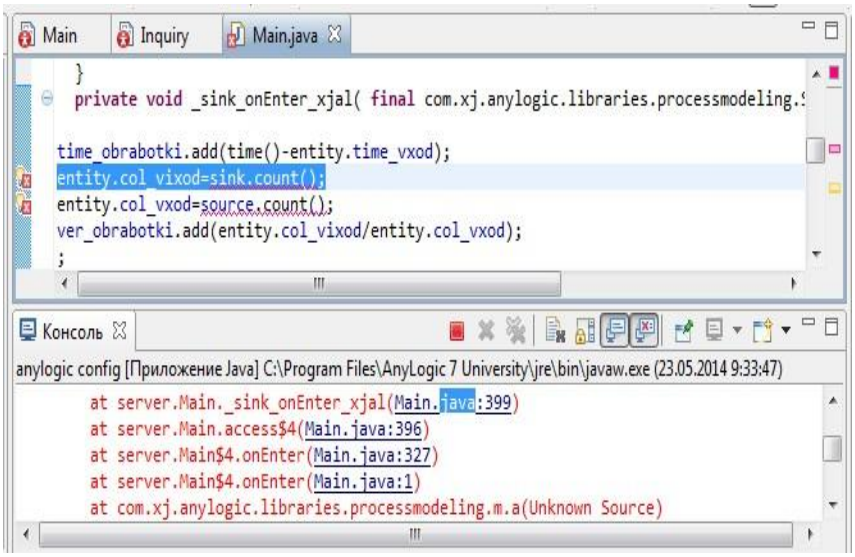

Рис. 39. Информация об ошибке в панели **Консоль**

7. Щёлкните выделенный Java код в панели **Консоль** `Main.java.399` (рис. 39). Появится код с выделенными ошибками.

Мы установили тип `int` для `col_vxod` и `col_vixod` (см. рис. 27). Изменим этот тип на `double`.

Удаление и добавление новых полей типа заявок

Обратите внимание, что вам не пришлось использовать поле `time_vixod`, так как вместо него была использована функция `time()`, возвращающая, как вам уже известно, текущее значение модельного времени. Удалите поле `time_vixod`.

1. В поле **Проект** дважды Щёлкните кнопку `Inquiry`. Откроется окно Inquiry (см. рис. 33).
2. Удалите `time_vixod`, выделив его и нажав Delete.
3. Выделите `col_vxod`. Из выпадающего списка **Тип:** вместо `int` выберите `double` (рис. 40).
4. Выделите `col_vixod`. Из выпадающего списка **Тип:** вместо `int` выберите `double`.

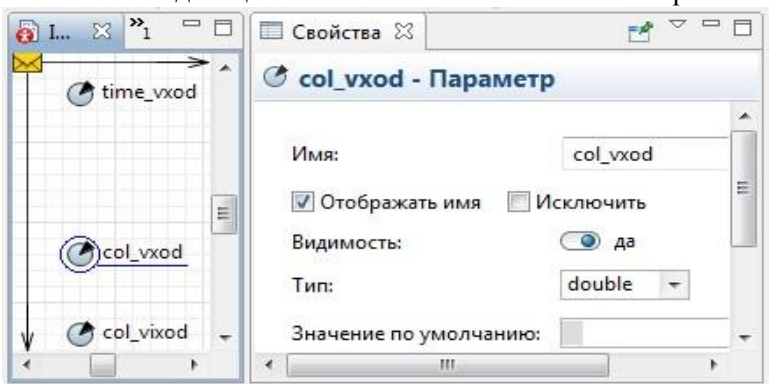

Рис. 40. Окно после удаления поля `time_vixod`

5. Из процедуры удаления поля следует, что так же можно вводить новые дополнительные поля нестандартного типа заявок. Например, перетащите из библиотеки **Основная** элемент **Параметр**. Дайте ему любое имя и установите **Тип:** из выпадающего списка. В дальнейшем это поле нестандартного типа заявки вы можете использовать в кодах модели.

Итак, все условия постановки задачи выполнены. Чтобы наблюдать за работой модели, установите, что время остановки модели не задано. Запустите модель. (рис. 41).

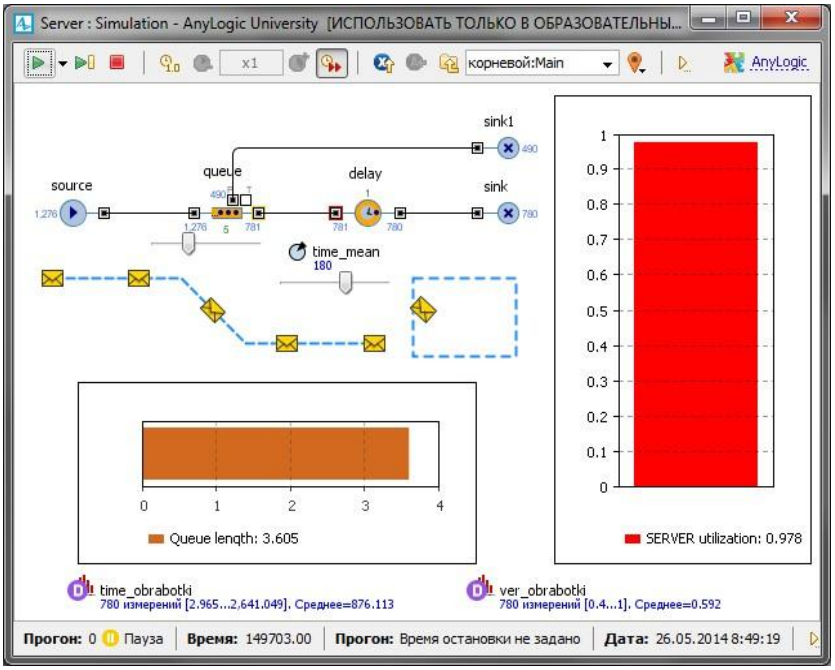

Рис. 41.Фрагмент работы модели

Добавление параметров и элементов управления

Активный объект может иметь параметры. Параметры обычно используются для задания статических характеристик объекта. Но значения параметров при необходимости можно изменять во время работы модели. Для этого нужно написать код обработчика события, то есть действий, которые должны выполняться при изменении значения параметра.

Создайте параметр `time_mean` объекта `delay`.

1. В **Палитре** выделите **Основная**.
2. Перетащите элемент **Параметр** на диаграмму класса `Main` и разместите ниже объекта `delay`, чтобы было видно, к какому объекту относится параметр.
3. Перейдите на панель **Свойства** (рис. 42).
4. В поле **Имя** введите имя параметра `time_mean` (среднее время). По этому имени параметр будет доступен из кода.
5. Задайте тип параметра `double`.

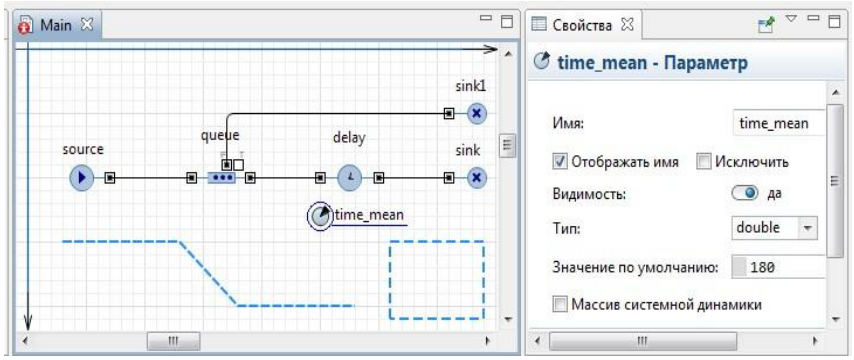

Рис. 42. Окно установки свойств элемента **Параметр**

6. В поле **Значение по умолчанию** установите 180. Если значение не задано явно, по правилам Java оно будет равно нулю.
7. Выделите объект `delay`.
8. На панели **Свойства** в поле **Время задержки** вместо выражения `exponential(1/180.0)` введите выражение `exponential(1/time_mean)`.

Пусть вы хотите изменять среднее время обработки запросов `time_mean` в ходе моделирования. Используйте для этого элемент управления — бегунок.

1. Откройте палитру **Элементы управления** и перетащите элемент **Бегунок** из палитры на диаграмму класса `Main` (рис. 43).

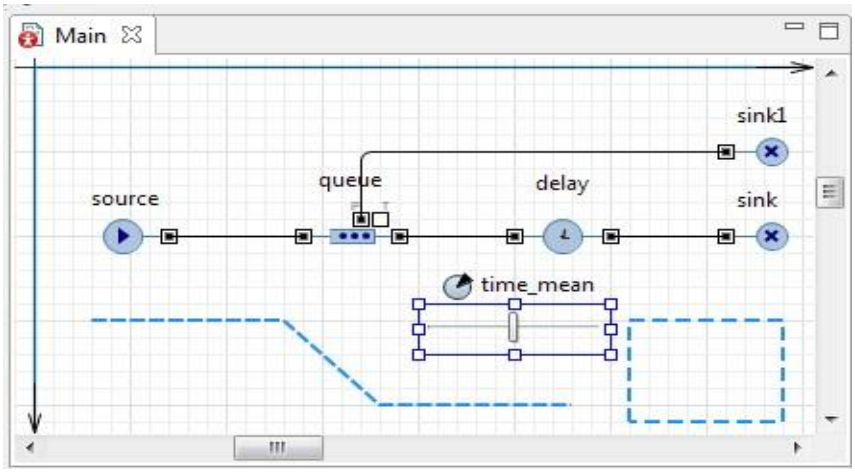

Рис. 43. Установка элемента управления **Бегунок**

2. Поместите бегунок под параметром `time_mean`, чтобы было понятно, что с помощью этого бегунка будет меняться среднее время обработки запросов объектом delay.

3. Пусть вы хотите варьировать среднее время от 1 до 300. Поэтому введите 1 в поле **Минимальное значение:**, а 300 — в поле **Максимальное значение:** (рис. 44).

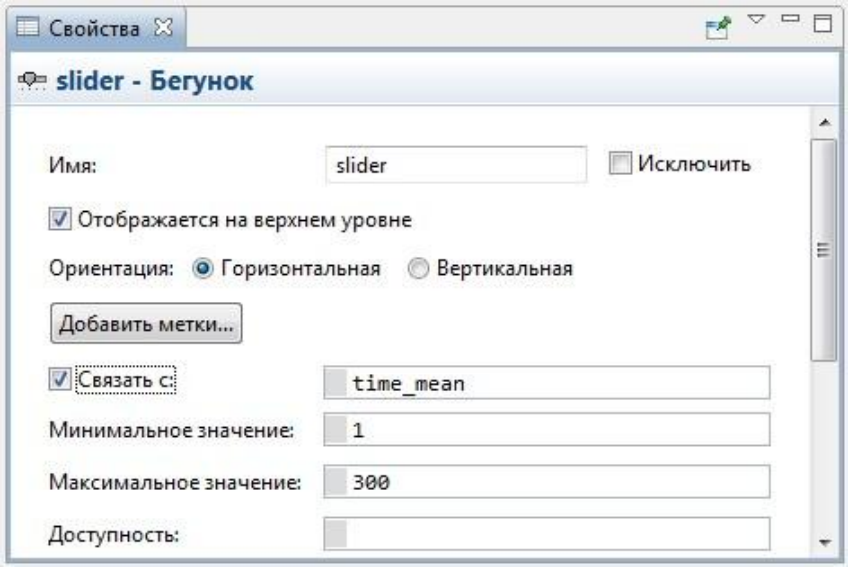

Рис. 44. Окно установки свойств элемента управления **Бегунок**

4. Установите флажок **Связать с:** и в активизированное поле введите `time_mean`.

Пусть теперь вы хотите также изменять ёмкость буфера в ходе моделирования. Используйте для этого также бегунок.

1. Откройте палитру **Элементы управления** и перетащите элемент **Бегунок** из палитры на диаграмму класса `Main` (рис. 44).
2. Поместите бегунок под объектом queue, чтобы было понятно, что с помощью этого бегунка будет меняться вместимость данного объекта, имитирующего входной буфер.
3. Пусть вы хотите варьировать ёмкость буфера от `0` до `15` запросов. Поэтому введите `15` в поле **Максимальное значение**.
4. Установите флажок **Связать с:** и в активизированное поле введите `queue.capacity`.
5. Запустите модель. Теперь вы можете изменять в процессе моделирования ёмкость входного буфера и среднее время обработки запросов с помощью бегунков. Можете также командой **Приостановить** приостановить работу модели, изменить значения параметров, а затем продолжить моделирование (рис. 45).
6. Остановите модель и перейдите на диаграмму класса Main.

Мы научились добавлять элементы **Параметр** и **Бегунок**. Но согласитесь, что какие параметры модели нужно будет менять, и в каких интервалах, заранее определить затруднительно. Также при использовании элемента **Бегунок** существуют трудности точного установления значения характеристики, так как невозможно предусмотреть нужный масштаб или цену деления **Бегунка**.

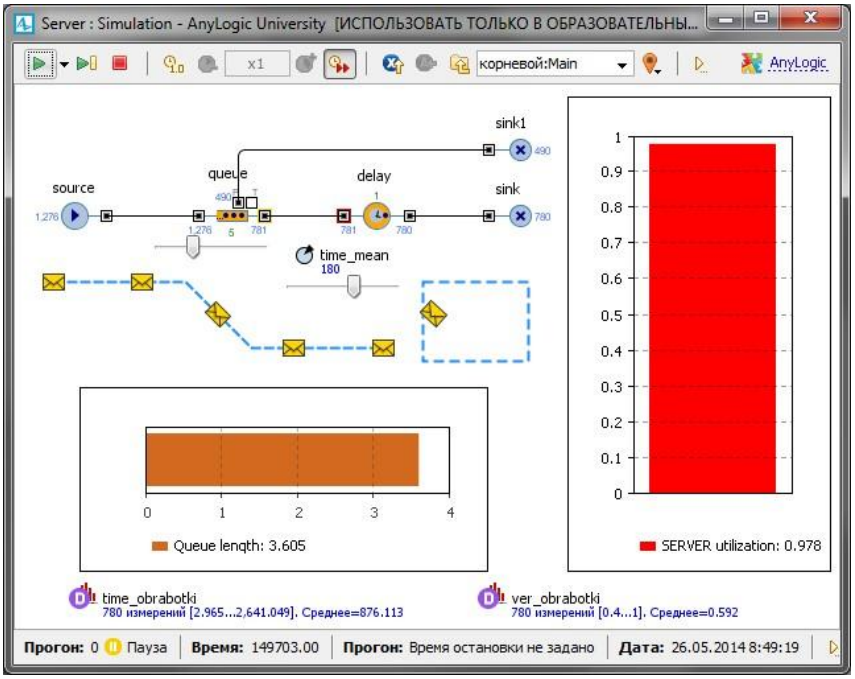

Рис. 45. Фрагмент работы модели с добавленным элементом **Параметр** и элементами управления

Существует и другой способ изменения свойств объектов во время выполнения модели: нужно щёлкнуть по элементу, войти в режим редактирования и ввести новое значение в одной из закладок всплывающего окна инспекта. Поэтому заранее не нужно продумывать, значения каких параметров планируется изменять, и не добавлять специальные элементы управления (например, бегунки).

Изменение значения в окне инспекта поддерживается для следующих элементов: простая переменная; параметр; накопитель.

И для следующих типов: численные; логический (boolean); текстовый (String).

1. Удалите элемент **Бегунок** для **Параметра** time_mean.
2. Запустите модель и приостановите её.
3. Щёлкните по значку **Параметра** time_mean.
4. Перейдите в режим редактирования (рис. 46).
5. Введите новое значение: `240.0`.
6. Закройте инспект. Рядом с элементом **Параметр** вы увидите введённое вами значение 240.0.
7. Запустите модель с новым свойством объекта delay.

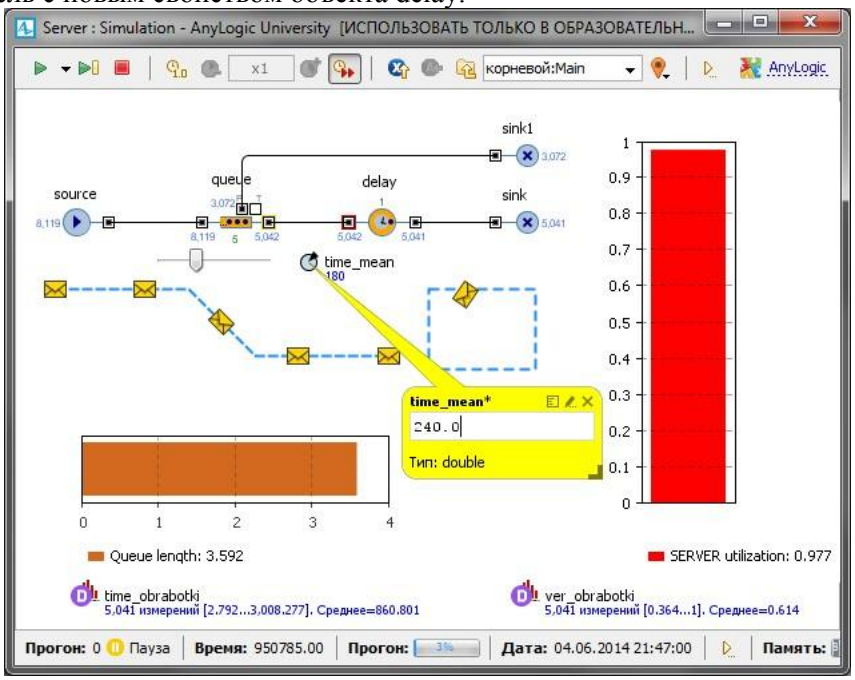

Рис. 46. Ввод нового значения параметра в окно инспекта

Добавление гистограмм

Теперь добавим на диаграмму нашего потока гистограмму, которая будет отображать собранную временную статистику.

1. Перетащите элемент **Гистограмма** из палитры **Статистика** в то место графического редактора, куда хотите ее поместить.
2. Укажите, какой элемент сбора данных хранит данные, которые вы хотите отображать на гистограмме: щёлкните кнопку **Добавить данные** и введите в поле **Данные** имя соответствующего элемента: `time_obrabotki` (рис. 47). Установите **Отображать среднее**.
3. В поле **Заголовок:** введите `Histogram Time obrabotki`.

Добавим на диаграмму нашего потока гистограмму, которая будет отображать собранную вероятностную статистику.

1. Перетащите элемент **Гистограмма** из палитры **Статистика**.
2. Щёлкните кнопку **Добавить данные** и введите в поле **Данные** имя элемента: `ver_obrabotki`. Установите **Отображать среднее**.
3. В поле **Заголовок:** введите `Histogram Ver obrabotki`. 4. Запустите модель. Фрагмент работы показан на рис. 48.

Замечание. Обратите внимание, что после нового запуска модели time_mean=180, хотя ранее мы изменили его значение на 240.

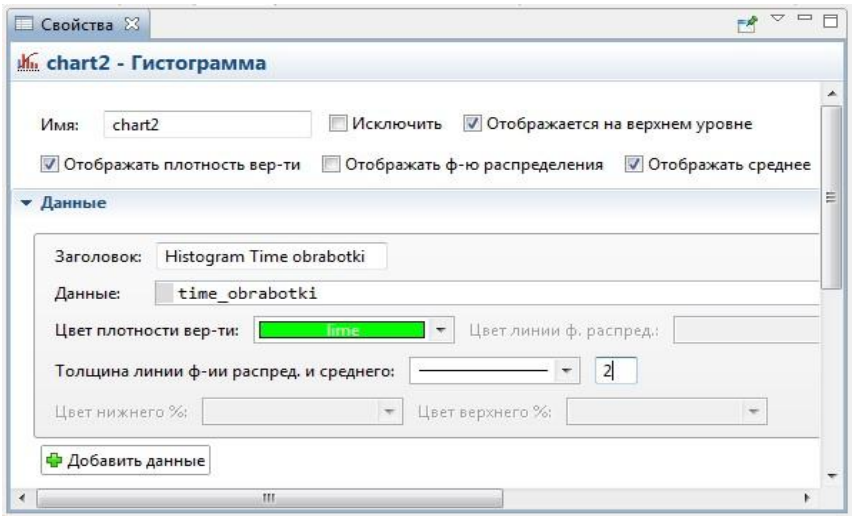

Рис. 47. Окно установки свойств элемента **Гистограмма**

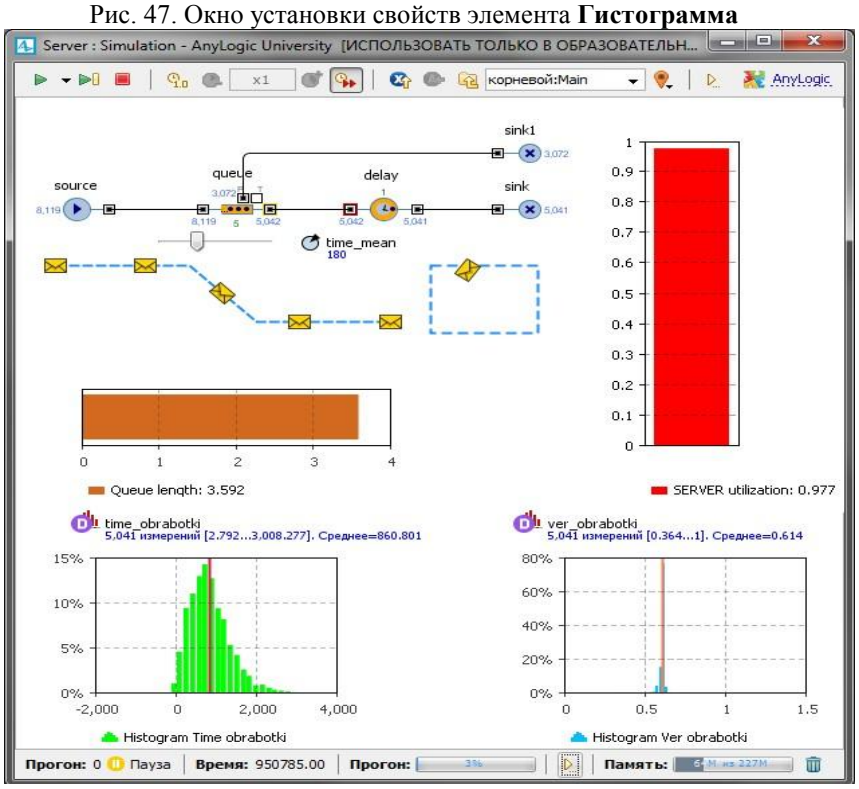

Рис. 48. Фрагмент работы модели с элементом управления и гистограммами

Изменение времени обработки запросов сервером

Построенная модель соответствует постановке задачи. В ней, с целью упрощения процесса построения первой модели, время обработки запросов сервером было принято распределённым по показательному (экспоненциальному) закону со средним значением $T2$ = 3мин.

Однако в модели время обработки поступающих запросов зависит от производительности сервера $Q = 6\ 10^5$ оп/с и вычислительной сложности запросов, распределенной по нормальному закону с математическим ожиданием $S1 = 6\ 10^7$ оп и среднеквадратическим отклонением $S2 = 2\ 10^5$ оп.

Кроме того, в модели определяется среднее количество запросов, обработанных за время моделирования 3600 с.

Внесите в модель изменения для аналогичного расчёта времени обработки запросов.

1. Удалите элемент **Параметр** с именем time_mean элемент **Бегунок** для элемента queue.
2. Из палитры **Основная** перетащите три элемента **Параметр** на диаграмму класса Main (рис. 49).

Рис. 49. Элементы AnyLogic-модели, соответствующие постановке

3. В поле **Имя** каждого из элементов введите `S1_`, `S2_` и `Q_` соответственно. Выберите **Тип** double.
4. В поле **Значение по умолчанию** каждого из элементов введите `60000000`,`200000` и `600000` соответственно.
5. Перетащите элемент **Переменная**. В поле **Имя** укажите `KolZap`. Выделите объект delay.
6. В поле **Время задержки** вместо exponential(1/time_mean) введите: `(normal(S2_,S1_))/Q_`
7. Выделите объект sink. В поле **Действие при входе** к имеющемуся там коду добавьте код:

```
KolZap=sink.in.count()/9604.0;
```

Для получения результатов моделирования с доверительной вероятностью $\alpha = 0{,}95$ и точностью $\varepsilon = 0{,}01$ нужно выполнить 9604 прогонов модели:

$$N = t_\alpha^2 \frac{p(1-p)}{\varepsilon^2} \approx 1{,}96^2 \frac{0{,}5^2}{0{,}01^2} \approx 9604,$$

t_α =1,96 где — табулированный аргумент функции Лапласа, $p-$ ожидаемая вероятность исхода события, в данном случае вероятность обаботки запросов сервером.

Расчёт проведен для так называемого «худшего» случая, то есть в предположении, что ожидаемая вероятность обработки запросов $p = 0{,}5$. Увеличим время моделирования в AnyLogic-модели в 9604 раз. А так как статистические данные о количестве обработанных запросов собираются за всё время моделирования, увеличенное в 9604 раз, то для получения среднего значения это количество нужно разделить на 9604, что и предусмотрено в коде.

10. Показатели моделируемой системы нужно определить в течение 3600 с, поэтому время моделирования в AnyLogic составит 3600*9604 = 34574400 единиц модельного времени.
11. В панели **Проект** выделите Simulation. На странице **Модельное время** в поле **Установить** выберите **В заданное время**.
12. В поле **Конечное время** установите `34574400`.
13. Запустите модель и дождитесь окончания моделирования. Результаты моделирования приведены на рис. 50.

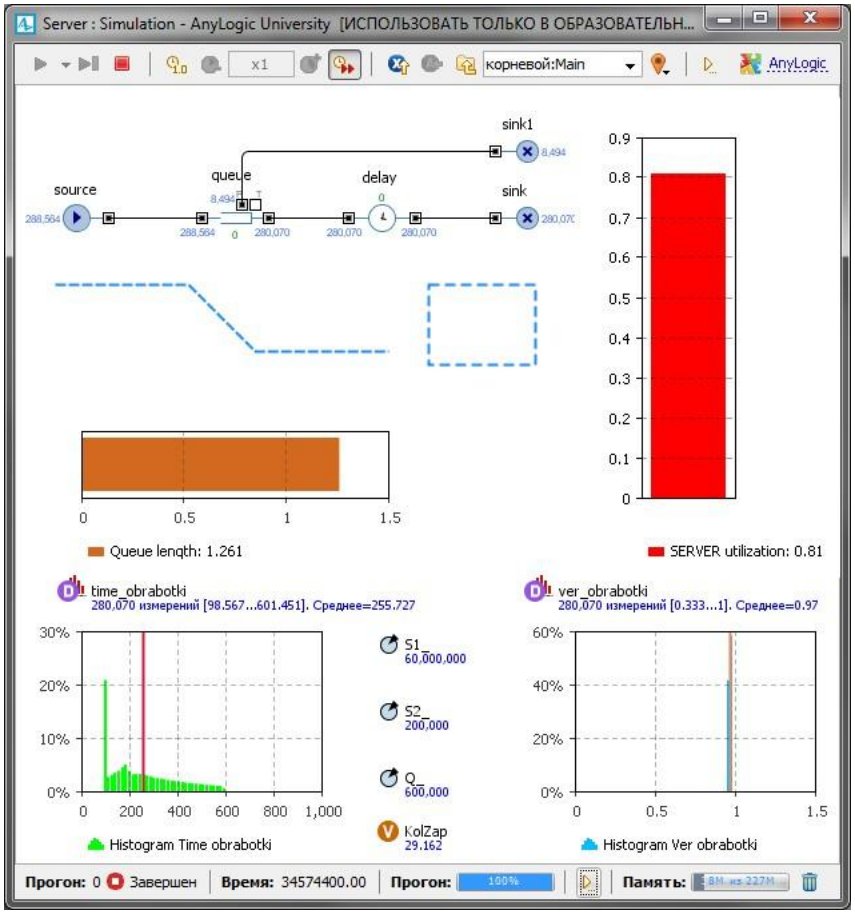

Рис. 50. Результаты моделирования обработки запросов сервером

Интерпретация результатов моделирования

Для проведения исследований на модели сделайте ещё несколько дополнений и изменений. Можно было бы обойтись и без них, но они улучшат эксплуатацию модели.

1. Из палитры **Основная** перетащите три элемента **Параметр** на диаграмму агента Main. Разместите их выше параметров S1_, S2_, Q_. Можно перетащить один параметр, скопировать его и вставить остальные два. Если установить свойства первого элемента **Параметр**, то они будут и в скопированных элементах. Не будем их устанавливать. Установим после копирования и вставки.
2. Выделите первый элемент **Параметр**. В поле **Имя:** первого параметра введите `timeMean` — среднее время поступления запросов для обработки на сервере. Оставьте тип double.
3. В поле **Значение по умолчанию** введите `120`.
4. Выделите объект source. В поле **Время между прибытиями** вместо 120.0 введите `timeMean`. Теперь вам при изменении среднего времени поступления запросов не придётся искать нужный код в свойствах объекта модели.
5. Выделите второй элемент **Параметр**. В поле **Имя:** введите `kolProg` — количество прогонов модели. Как и в предыдущем случае, при корректировке числа прогонов код в свойствах искать будет не нужно. Оставьте тип double.
6. В поле **Значение по умолчанию** введите `9604`. Выделите объект sink.
7. В поле **Действие при входе** в имеющемся там коде замените последнюю строку следующей:

```
KolZap=round(sink.in.count()/kolProg);
```

На рис. 1. количество обработанных запросов KolZap сервером выдаётся с дробной частью. Теперь, вследствие применения процедуры round, количество запросов будет целым.

Как уже отмечалось, при изменении количества прогонов модели не надо будет искать код для требуемой корректировки. Однако всё-таки потребуется изменить модельное время. Например, если нужно выполнять с моделью 1000 прогонов, то модельное время следует установить равным 3600*1000=3600000.

8. Выделите третий элемент **Параметр**. В поле **Имя:** введите `emkBuf` — ёмкость в сообщениях входного буфера сервера. Установите тип `int`.
9. В поле **Значение по умолчанию** введите `5`.
10. Выделите объект queue. В поле **Вместимость** введите

`emkBuf`. 12. Запустите модель. Результаты моделирования на рис. 51.

Проведём несколько экспериментов. Будем изменять среднее время поступления запросов timeMean в предположении, что с течением времени количество источников запросов будет расти, то есть timeMean будет

уменьшаться. В сторону увеличения будем изменять ёмкость входного буфера emkBuf и производительность Q_ сервера.

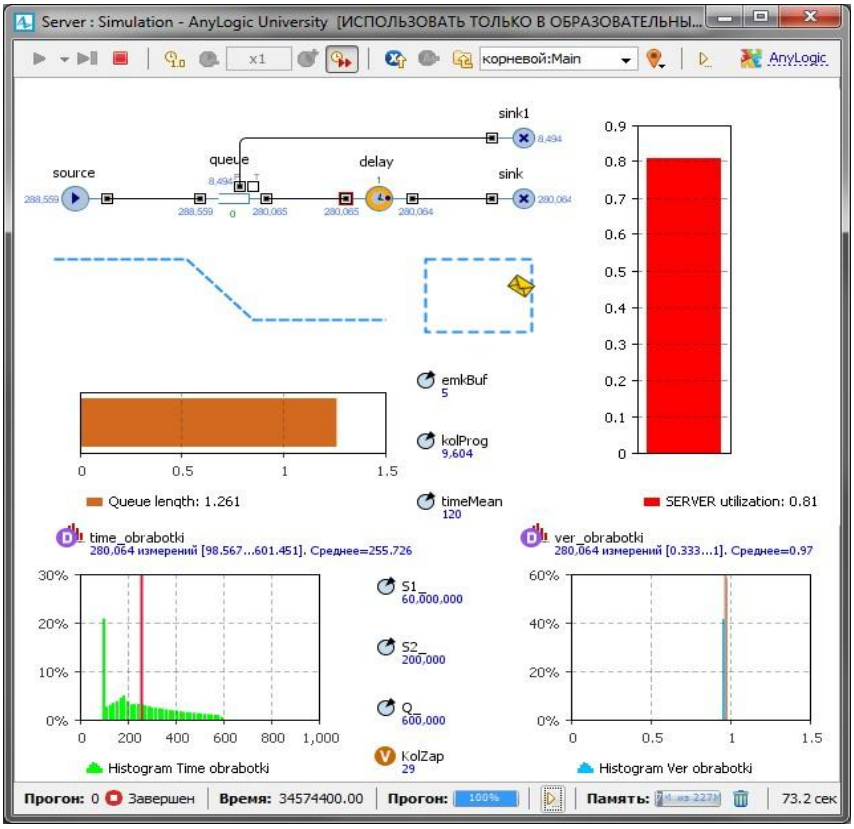

Рис. 51. Результаты моделирования при условиях постановки задачи

Результаты экспериментов приведены в табл. 1.1.

Из экспериментов 1 и 2 следует, что при увеличении ёмкости входного буфера в два раза вероятность обработки запросов увеличивается незначительно на 0,025, то есть количество обработанных запросов практически одно и тоже. Среднее время обработки одного запроса возрастает в 1,27 раза вследствие увеличения длины очереди в 1,48 раза.

Увеличение интенсивности поступления запросов в три раза при увеличении ёмкости входного буфера в два раза (эксперименты 3 и 4) также не даёт существенного увеличения количества обработанных запросов: 36 вместо 30.

Таблица 1.1 Показатели обработки запросов сервером

Показатели	AnyLogic6	AnyLogic7
1) timeMean = 120, emkBuf = 5		
Количество обработанных запросов	29	29
Вероятность обработки запросов	0,971	0,970
Среднее время обработки одного запроса	254,942	255,727
Средняя длина очереди запросов к серверу	1,254	1,261
Коэффициент использования сервера	0,810	0,810
2) timeMean = 120, emkBuf = 10		
Количество обработанных запросов	30	30
Вероятность обработки запросов	0,995	0,995
Среднее время обработки одного запроса	321,540	325,017
Средняя длина очереди запросов к серверу	1,841	1,870
Коэффициент использования сервера	0,831	0,831
3) timeMean = 40, emkBuf = 5		
Количество обработанных запросов	36	36
Вероятность обработки запросов	0,399	0,400
Среднее время обработки одного запроса	1055,108	554,983
Средняя длина очереди запросов к серверу	4,552	4,550
Коэффициент использования сервера	1,00	1,00
4) timeMean = 40, emkBuf = 10		
Количество обработанных запросов	36	36
Вероятность обработки запросов	0,399	0,400

Среднее время обработки одного запроса	591,304	1054,907
Средняя длина очереди запросов к серверу	9,551	9,549
Коэффициент использования сервера	1,00	1,00
5) timeMean = 40, emkBuf = 10, Q_ = 1000000		
Количество обработанных запросов	60	60
Вероятность обработки запросов	0,665	0,667
Среднее время обработки одного запроса	591,304	591,129
Средняя длина очереди запросов к серверу	8,855	8,852
Коэффициент использования сервера	1,00	1,00
6) timeMean = 40, emkBuf = 15, Q_ = 2000000		
Количество обработанных запросов	90	90
Вероятность обработки запросов	1,00	1,00
Среднее время обработки одного запроса	75,049	75,096
Средняя длина очереди запросов к серверу	1,096	1,128
Коэффициент использования сервера	0,750	0,751

При этом уменьшается вероятность обработки запросов в 2,5 раза, а время обработки одного запроса возрастает в 3,25 раза. Возросла и средняя длина очереди запросов к серверу

Коэффициент использования сервера равен 1. Из этого следует, что добиться увеличения вероятности и количества обработанных запросов можно только увеличением производительности сервера.

В экспериментах 5 и 6 увеличена производительность сервера до 1000000 и 2000000 оп/с соответственно.

По сравнению с экспериментом 1 количество обработанных запросов в эксперименте 5 увеличилось в 2 раза, а в эксперименте 6 — в 3 раза (вероятность обработки запросов равна 1). Среднее время обработки одного запроса всё равно примерно в 2 раза больше в эксперименте 5, а в эксперименте 6 — в 3,4 раза меньше. Можно полагать, что такая разница в среднем времени обработки одного запроса вызвана тем, что в эксперименте 5 средняя длина очереди запросов к серверу 8,852, а в эксперименте 6 — 1,128, то есть в 7,85 раза меньше.

Коэффициент использования сервера в эксперименте 6 равен 0,751. Из этого следует, что дальнейшее увеличение производительности сервера приведёт к уменьшению длины очереди и среднего времени обработки одного запроса.

Машинное время выполнения модели в AnyLogic примерно 70…90 с.