Практические задания

Вариант 1 2
Вариант 2 18
Вариант 3 21
Вариант 4 25
Вариант 5 34

Вариант 1

Модель обработки запросов сервером

1.1. 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. На них мы остановимся позже. Приступим к созданию диаграммы процесса.

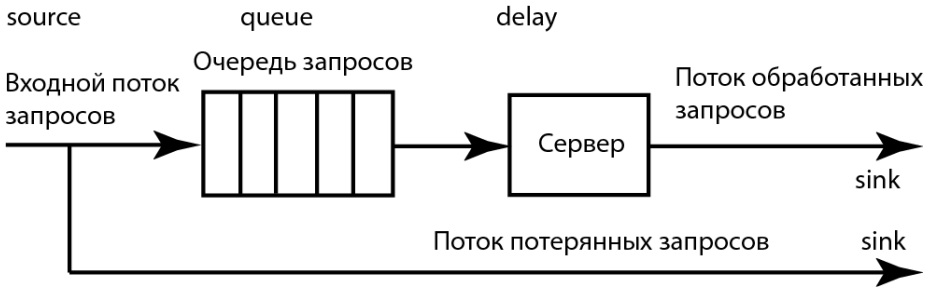


Рис. 1.1. Сервер как система массового обслуживания

1.2. Создание диаграммы процесса

1. Выполните **Файл/Создать/Модель** на панели инструментов. Появится диалоговое окно **Новая модель** (рис. 1.2).
2. Задайте имя новой модели. В поле **Имя модели** введите Server.
3. Выберите каталог, в котором будут сохранены файлы модели. Если хотите сменить предложенный по умолчанию каталог на какой-то другой, то можете ввести путь к нему в поле **Местоположение** или выбрать этот каталог с помощью диалога навигации по файловой системе, открывающегося нажатием кнопки **Выбрать**.

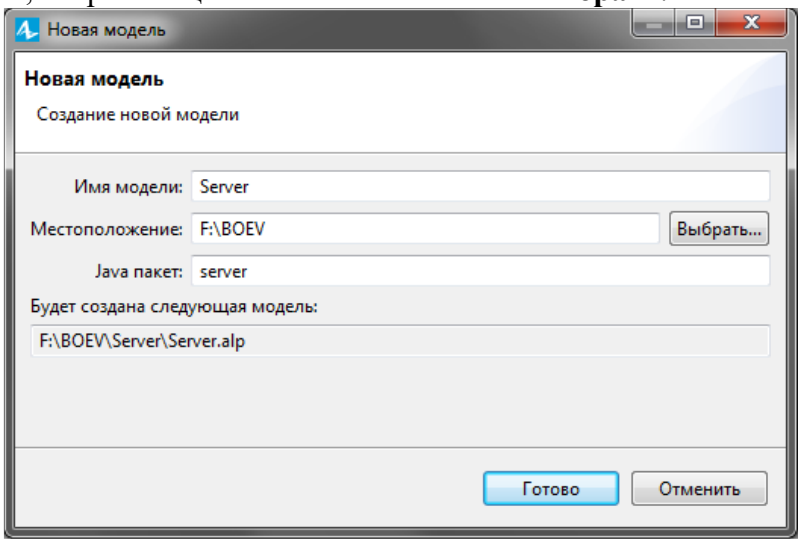


Рис. 1.2. Диалоговое окно **Новая модель**

4. Щёлкните кнопку **Готово**. Откроется пользовательский интерфейс (рис. 1.3). Остановимся на нём.

5.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. Для добавления объекта на диаграмму, надо щёлкнуть его мышью и, не отпуская её, перетащить в графический редактор.

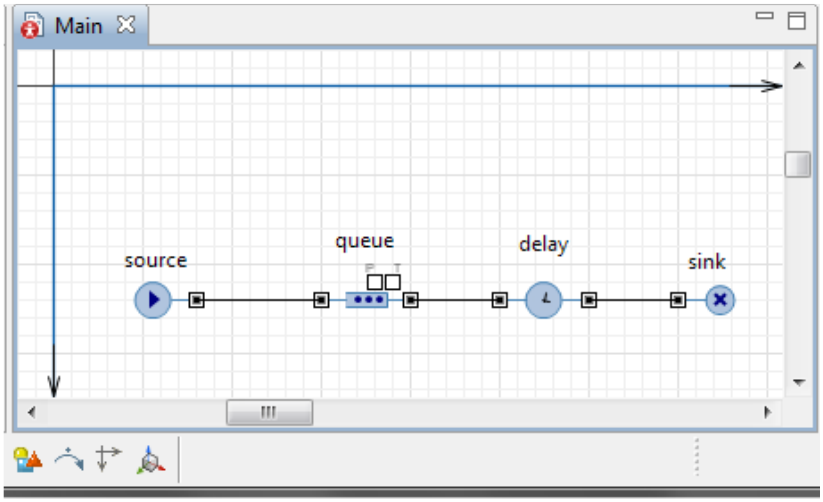


Рис. 1.5. Диаграмма системы массового обслуживания

1.3. Изменение свойств блоков модели, её настройка и запуск

Помним, что мы хотим сначала создать простейшую модель, в которой будем рассматривать только обработку запросов сервером.
В основе каждой дискретно-событийной модели лежит диаграмма процесса— последовательность соединенных между собой объектов (**Библиотеки моделирование процессов**), задающих последовательность операций, которые будут производиться над проходящими по диаграмме процесса заявками.

1.3.1. Изменение свойств блоков диаграммы процесса

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. Однако имеется возможность установить время в минутах, часах, днях, в чем вы убедитесь несколько позднее, когда будете устанавливать модельное время.
1. Выделите объект source. В выпадающем списке **Прибывают согласно:** укажите, что запросы поступают согласно **Времени между прибытиями:** (рис. 1.6).
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.

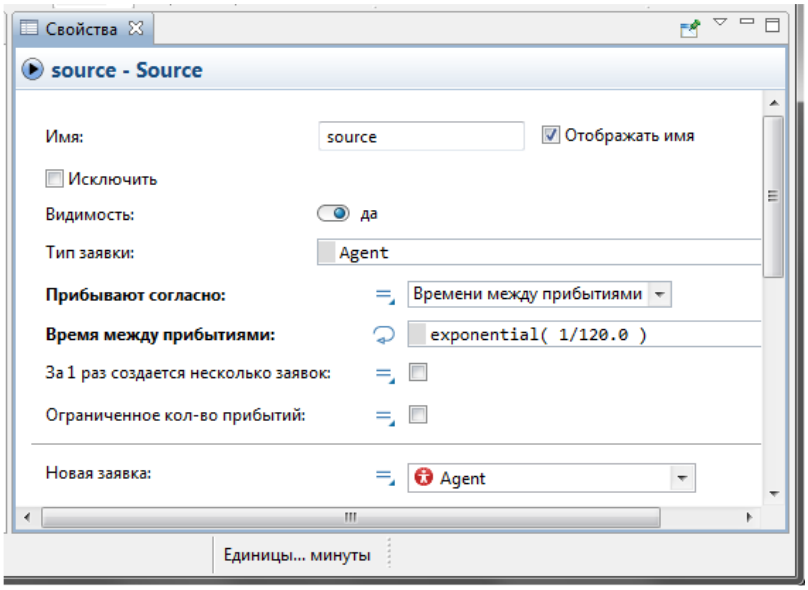


Рис. 1.6. Свойства объекта **source**

Измените свойства объекта **queue** (рис. 1.7).

1. Задайте длину очереди. Введите в поле **Вместимость:** 5. В очереди будут находиться не более 5 запросов.
2. Установите флажок **Включить сбор статистики**, чтобы включить сбор статистики для этого объекта. В этом случае по ходу моделирования будет собираться статистика по количеству запросов в очереди. Если же вы не установите этот флажок, то данная функциональность будет недоступна, поскольку по умолчанию она отключена для повышения скорости выполнения модели. Для вывода, например, средней длины очереди, нужно в модели предусмотреть Java код.

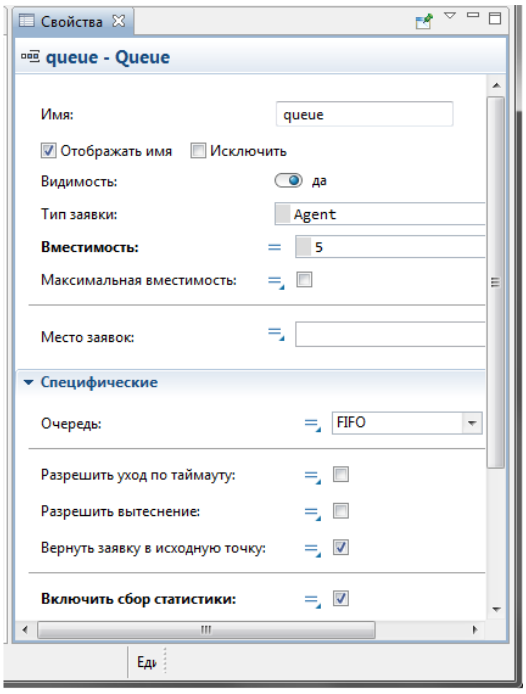


Рис. 1.7. Свойства объекта **queue**

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. 1.8).
1. Обработка одного запроса занимает примерно 3 мин. Задай- те время обслуживания, распределенное по экспоненциальному за- кону со средним значением 3 мин. Для этого введите в поле **Время задержки:** exponential(1/180.0). Функция exponential() является стандартной функцией генератора случайных чисел AnyLogic. AnyLogic предоставляет функции и других случайных распределений, таких как нормальное, треугольное, и т. д.
2. Установите флажок **Включить сбор статистики**.

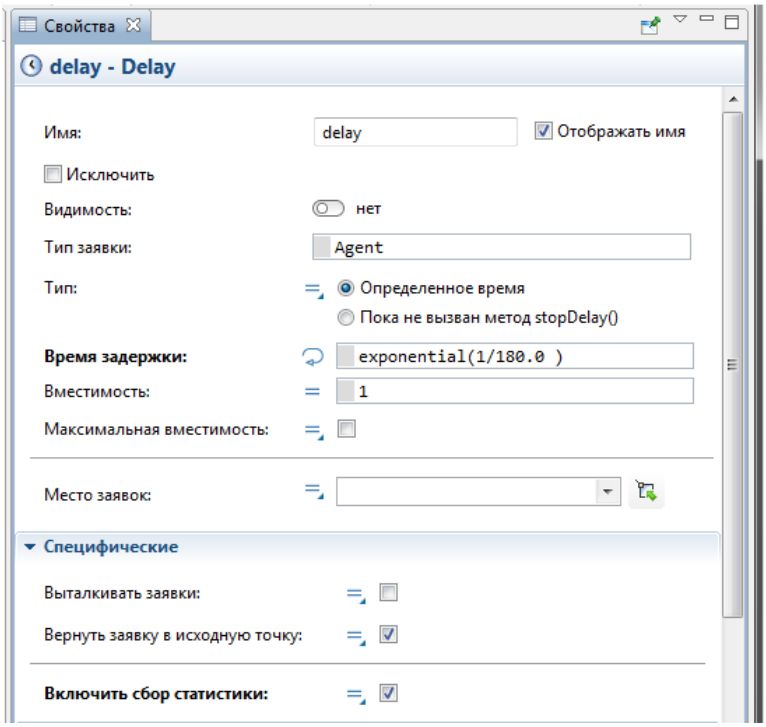


Рис. 1.8. Свойства объекта **delay**

Для вывода коэффициента использования объекта **delay** в модели также следует предусмотреть соответствующий Java код.
Последним в диаграмме нашей дискретно-событийной модели находится объект **sink**. Этот объект уничтожает поступившие заявки. Обычно он используется в качестве конечной точки потока заявок (и диаграммы процесса соответственно). В нашем случае он выводит из модели обработанные сервером запросы.

1.3.2. Настройка запуска модели

Обработку запросов сервером мы планируем исследовать в течение одного часа, т.е. 3600 с.
1. В панели **Проект** выделите эксперимент **Simulation: Main**.
2. Щелчком раскройте вкладку **Модельное время**.
3. Установите **Виртуальное время (максимальная скорость)** (рис. 1.9).
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. 1.10).
10. Из выпадающего списка **Единицы модельного времени:** выберите секунды.

1.3.3. Запуск модели

Постройте вашу модель с помощью кнопки панели инструментов  (F7) **Построить модель** (при этом в рабочей области AnyLogic должен быть выбран какой-то элемент именно этой модели). Если в модели есть какие-нибудь ошибки, то построение не будет завершено, и в панель **Ошибки** будет выведена информация об ошибках, обнаруженных в модели. Двойным щелчком мыши по ошибке в этом списке вы можете перейти к предполагаемому месту ошибки, чтобы исправить её. При этом откроется соответствующее место ошибки.

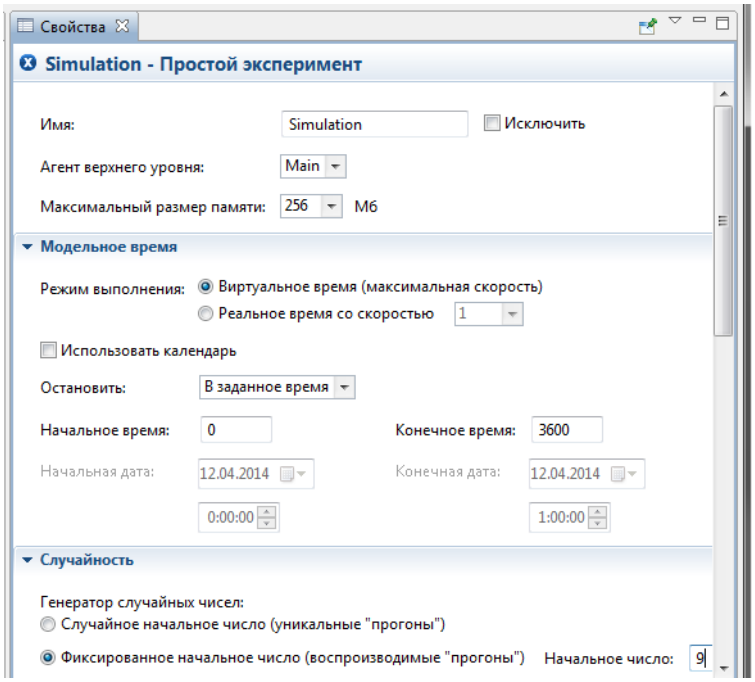


Рис. 1.9. Установка свойств эксперимента

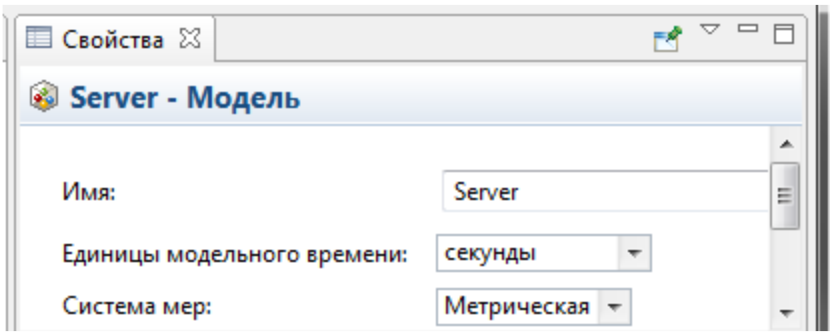


Рис. 1.10. Установка модельного времени

После исправления ошибок и построения модели, запустите её:

1. Щёлкните мышью кнопку панели инструментов  **Запустить** (или нажмите F5) и выберите из открывшегося списка эксперимент, который вы хотите запустить. Эксперимент этой модели будет называться **Server/Simulation**.

1.4. Создание анимации модели

Анимация модели рисуется в той же диаграмме (в графическом редакторе), в которой задается и диаграмма моделируемого процесса.
Нарисуйте прямоугольный узел, который будет обозначать на анимации сервер.

1. Откройте палитру **Разметка пространства** (рис. 1.15). Чтобы открыть какую-либо палитру, нужно щелчком из **Проекты** перейти в **Палитра** и щёлкнуть по иконке этой палитры.
2. Палитра **Разметка пространства** (рис. 1.15) содержит в качестве элементов различные примитивные фигуры, используемые для рисования презентаций моделей. Это путь, прямоугольный узел, многоугольный узел, точечный узел, аттрактор, стеллаж, масштаб.

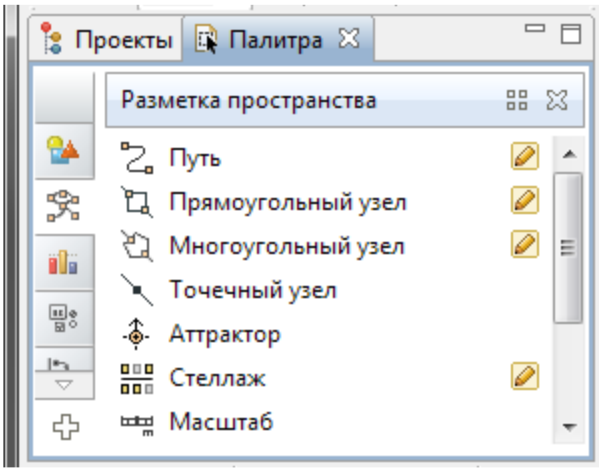


Рис. 1.15. Палитра **Разметка пространства**

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

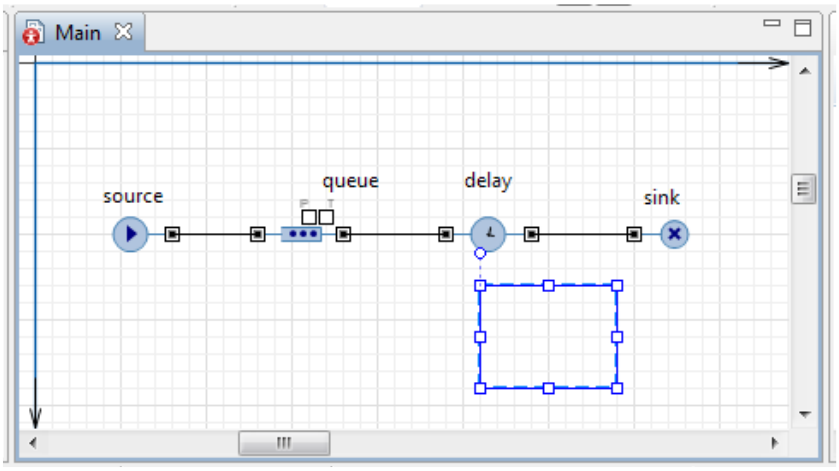


Рис. 1.16. Элемент **Прямоугольный узел** на диаграмме

4. Давайте сделаем так, чтобы цвет этого прямоугольного узла будет меняться в зависимости от того, обрабатывает ли сервер в данный момент времени запрос или нет.
5. Для этого выделите нарисованную нами фигуру на диаграмме. Перейдите на страницу **Внешний вид** панели свойств (рис. 1.17).

Если нужно, чтобы по ходу моделирования то или иное свойство фигуры меняло своё значение в зависимости от каких-то условий, то можете ввести в поле соответствующего динамического свойства выражение, которое будет постоянно вычисляться заново при выполнении модели.
Возвращаемый результат вычисления будет присваиваться текущему значению этого свойства. 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

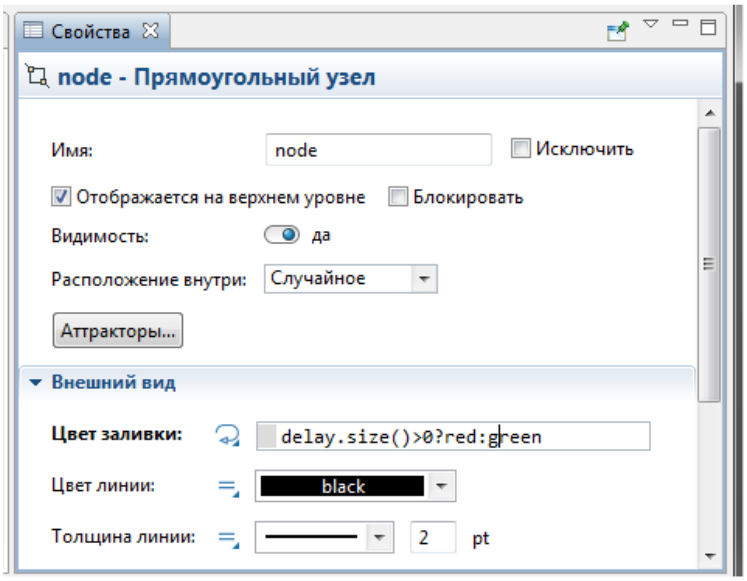


Рис. 1.17. Установлено динамическое значение цвета заливки

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.

6. Нарисуйте путь, который будет обозначать на анимации очередь к серверу (рис. 1.18).

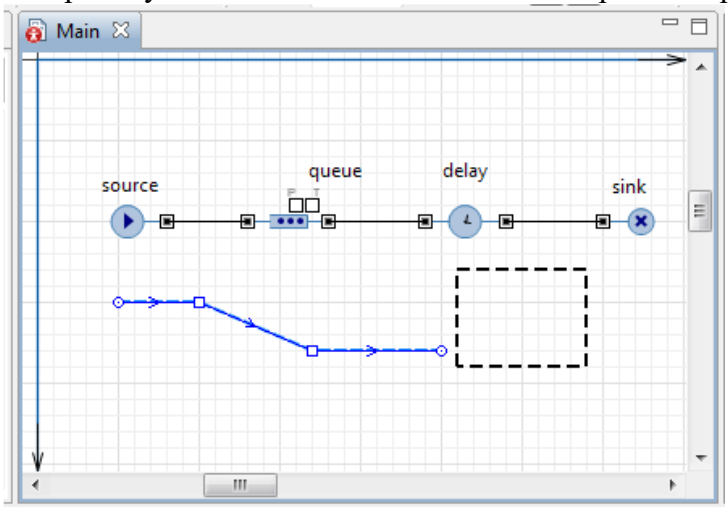


Рис. 1.18. Путь на диаграмме процесса

7. Теперь мы должны задать созданные анимационные объекты в качестве анимационных фигур объектов диаграммы нашего процесса. Задайте путь в качестве фигуры анимации очереди. Вы- делите объект queue. На странице свойств объекта queue в поле **Место заявок:** выберите из выпадающего списка path (рис. 1.19).

8. Задайте прямоугольный узел в качестве фигуры анимации сервера. Выделите объект delay. Введите в поле **Место заявок:** из выпадающего списка имя нашего прямоугольного узла: node (рис. 1.20).

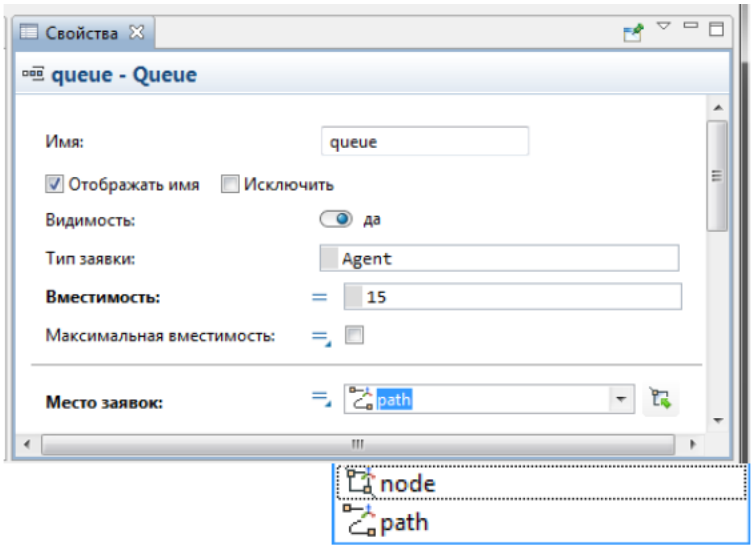


Рис. 1.19. Задание пути в качестве фигуры анимации очереди

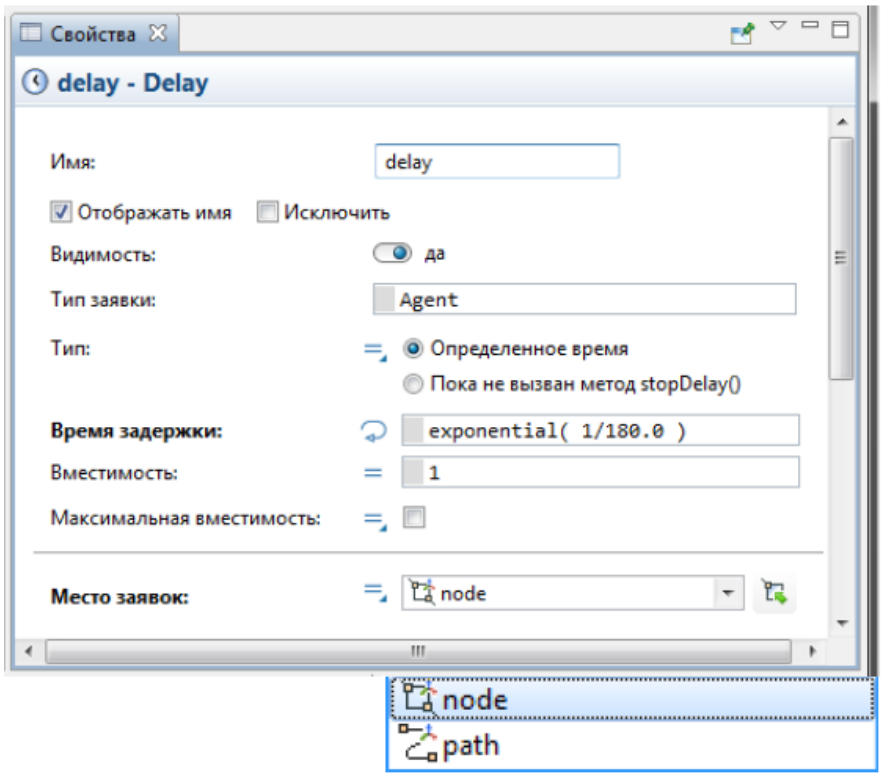


Рис. 1.20. Задание прямоугольного узла в качестве фигуры анимации сервера

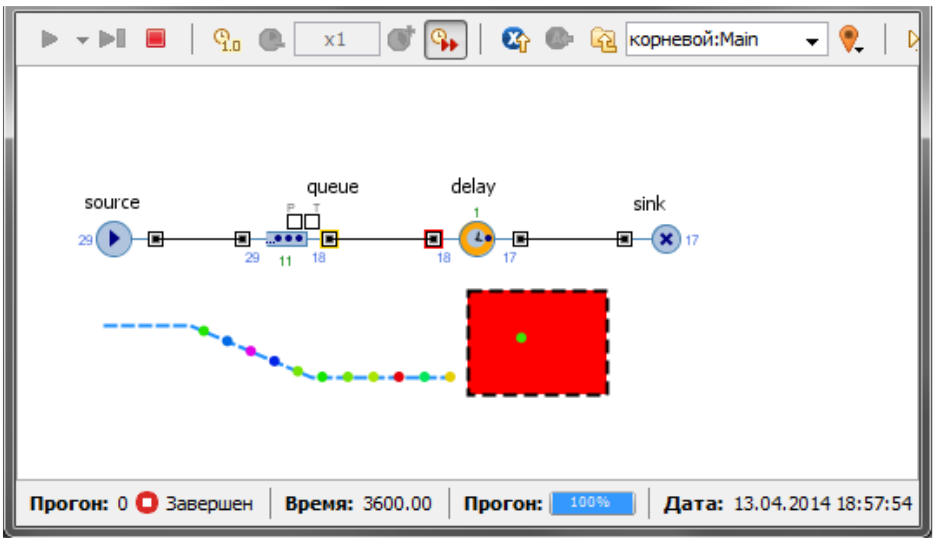


Рис. 1.21. Анимация модели

9. Запустите модель. Вы увидите, что у модели теперь есть простейшая анимация— сервер и очередь запросов к нему (рис. 1.21). Цвет фигуры сервера будет меняться в зависимости от того, обрабатывается ли запрос в данный момент времени или нет.

1.5. 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**. Эта палитра содержит элементы сбора данных и статистики, а также диаграммы для визуализации данных и результатов моделирования.
2. Перетащите элемент **Столбиковая диаграмма** из палитры **Статистика** на диаграмму класса и измените ее размер, как показано на рис. 1.32.

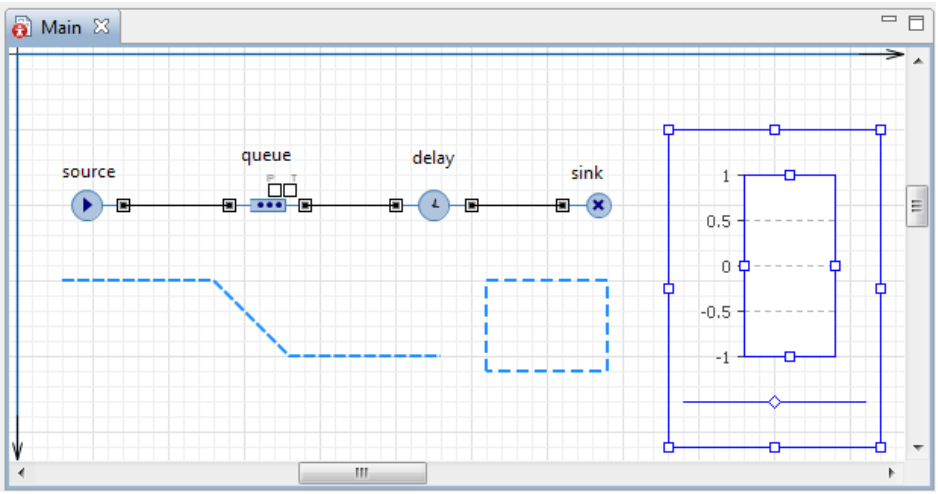


Рис. 1.22. Элемент **Столбиковая диаграмма** на диаграмме класса

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. 1.23).

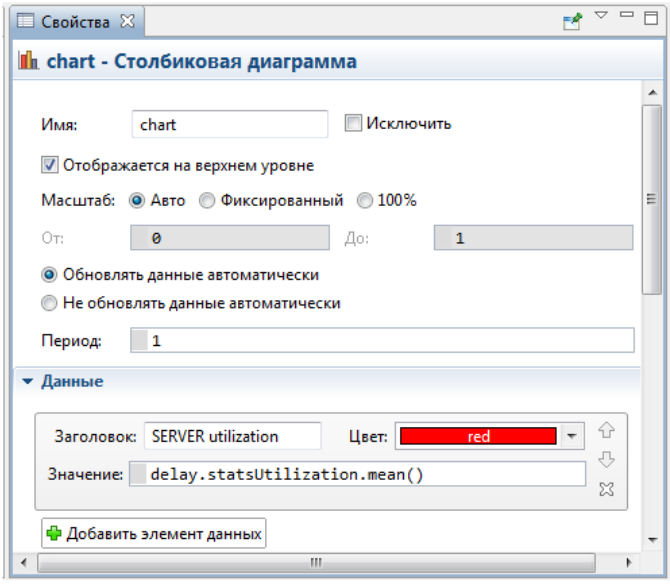


Рис. 1.23. Страница **Свойства**

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(). Полный список методов можно найти на странице документации этого класса набора данных: **StatisticsContinuous** (на английском языке).
6. Щёлкните **Внешний вид** (рис. 1.24). Установите свойства: направление столбцов, цвета фона, границ, меток, сетки, положение подписей у столбцов.

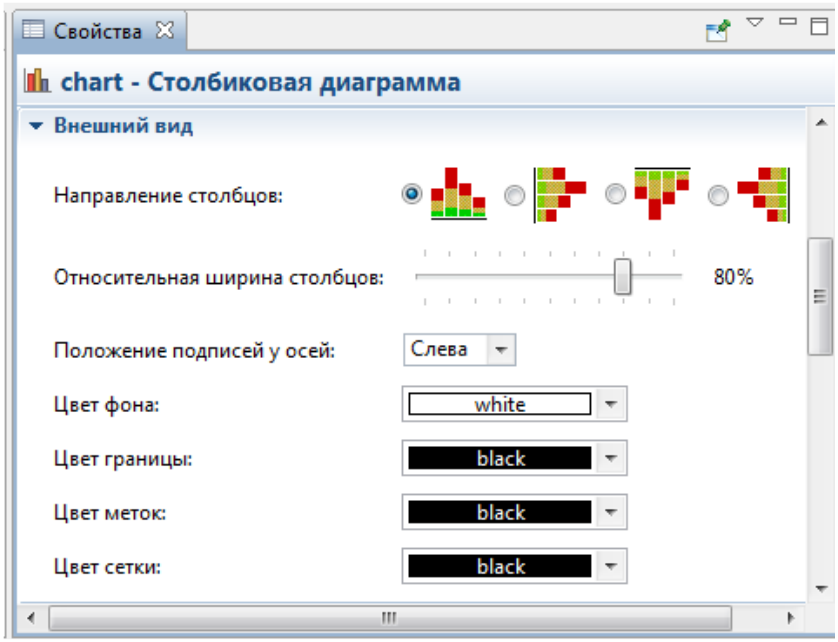


Рис. 1.24. Вкладка **Внешний вид**

7. Раскройте щелчками страницы (вкладки) **Местоположение и размер**, **Легенда**, **Область диаграммы** (рис. 1.25). 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.

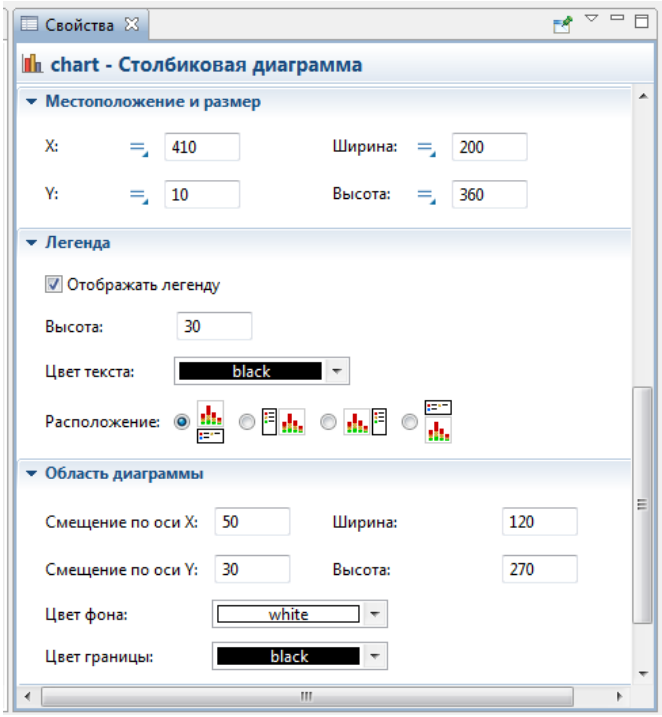


Рис. 1.25. Вкладки **Местоположение и размер**, **Легенда**, **Область диаграммы**

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. 1.26).
10. **Заголовок:** и **Значение:** измените так, как показано на рис. 1.26. 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.

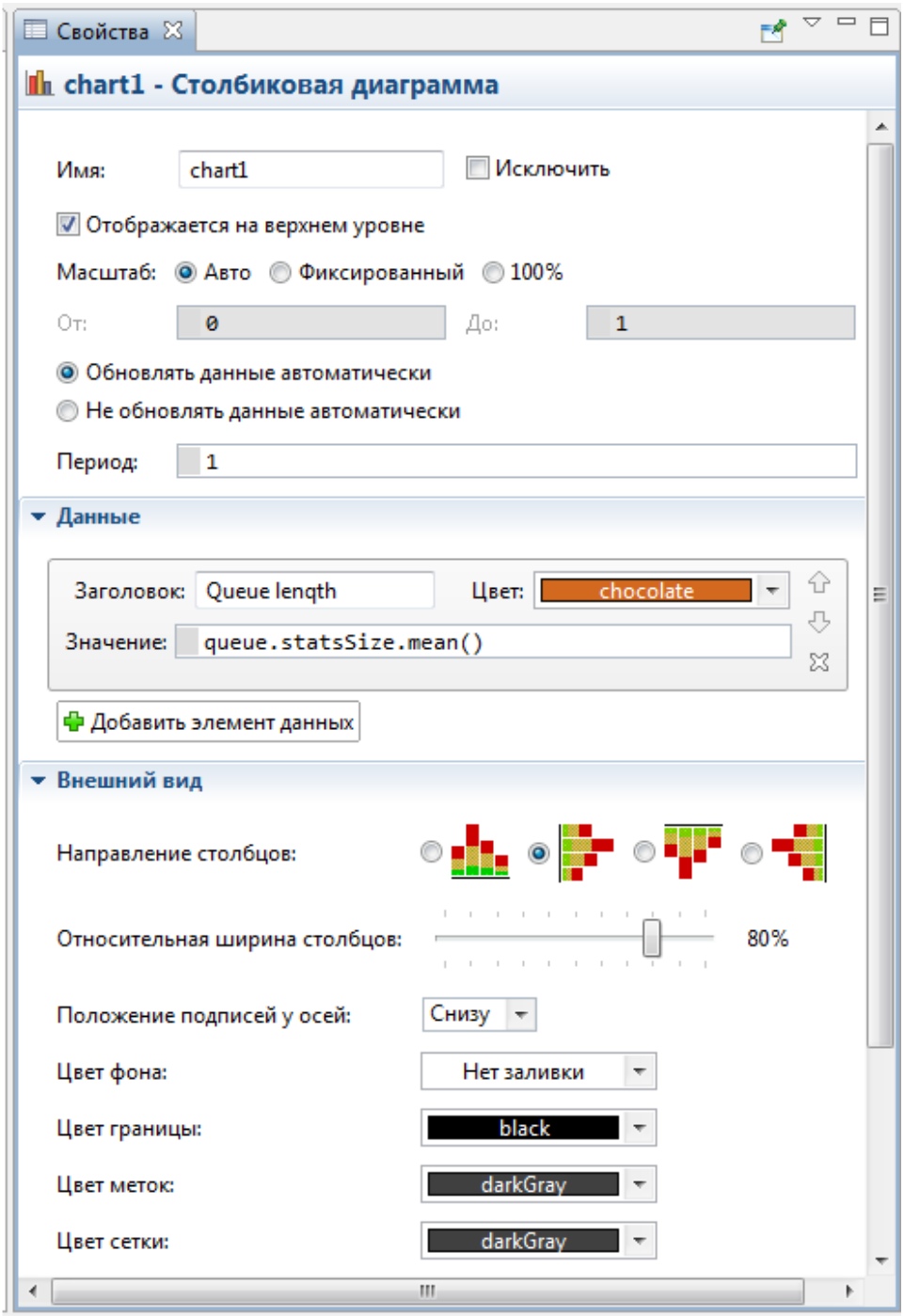


Рис. 1.26. Страницы **Данные**, **Внешний вид** панели **Свойства**

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.

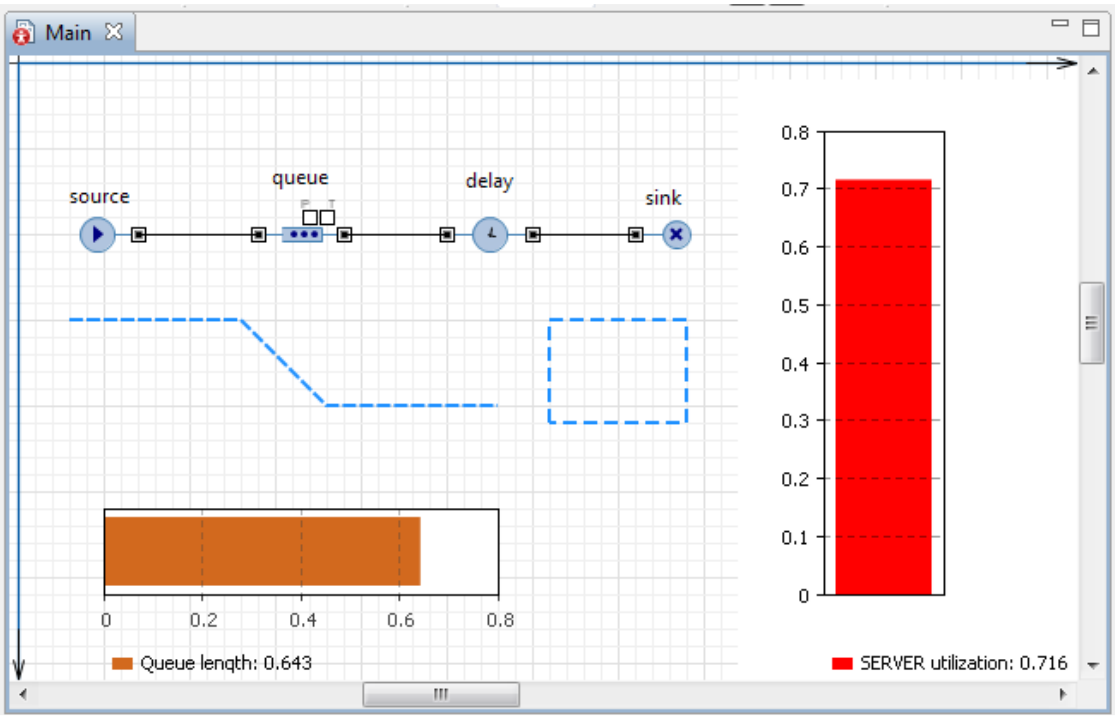


Рис. 1.27. Добавлены две столбиковые диаграммы

13. На странице **Внешний вид** панели **Свойства** выберите в секции свойств **Направление столбцов** вторую опцию (рис. 1.26), чтобы столбцы во второй столбиковой диаграмме, расположенной горизонтально, росли вправо (рис. 1.27).
14. Запустите модель с двумя столбиковыми диаграммами, установив модельное время 3600 единиц, и понаблюдайте за её работой (рис. 1.28).

1.6. Уточнение модели согласно ёмкости входного буфера

На рис. 1.28 (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. 1.22) — 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.
А возможно ли выполнить данное условие постановки задачи средствами AnyLogic? Оказывается, что можно. Причем, различными способами. Уточним модель согласно постановке задачи одним из этих способов.

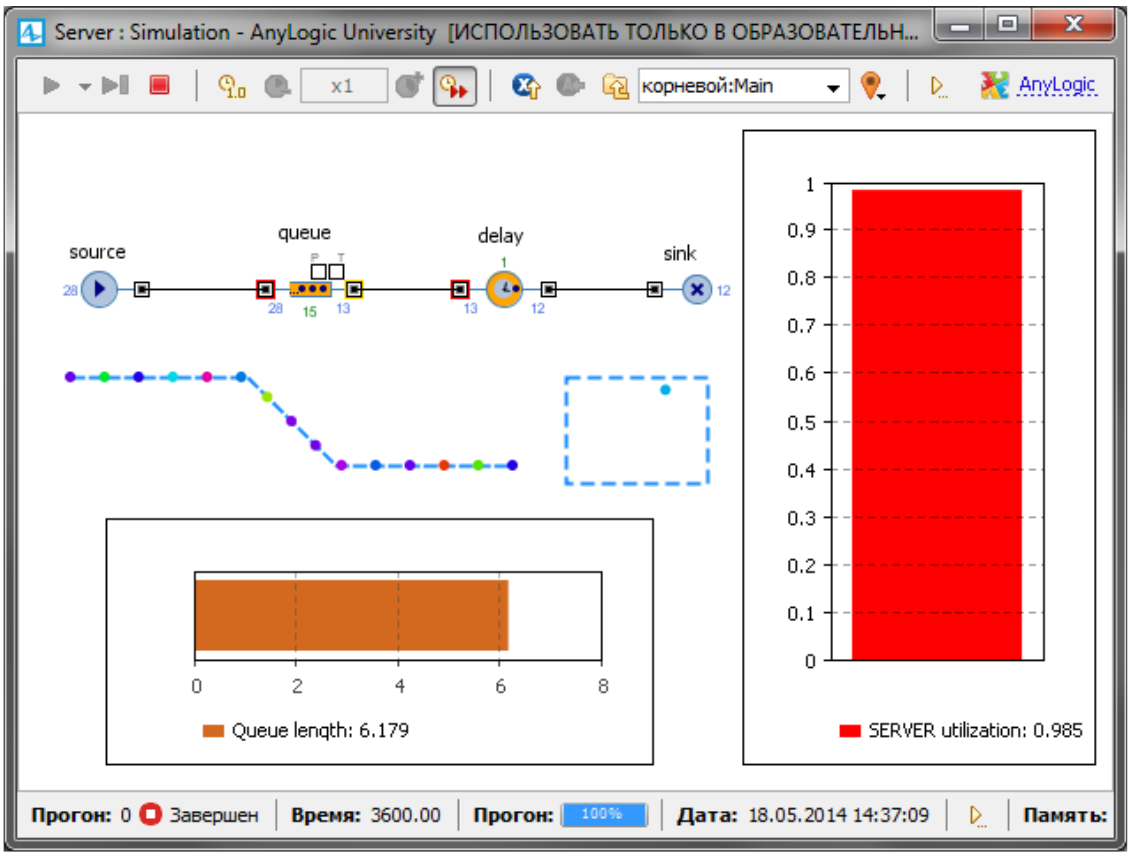


Рис. 1.28. Наблюдение за моделью с двумя столбиковыми диаграммами

Объект queue моделирует очередь заявок, ожидающих приёма объектами, следующими за ним в потоковой диаграмме, или же моделирует хранилище заявок общего назначения. Для выполнения условия постановки задачи воспользуемся последним способом вытеснения. 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. 1.29). 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.
5. После двойного щелчка по второму порту вы увидите, что появится соединитель. Если выделить его мышью, то при правильном соединении портов конечные точки соединителя должны подсветиться зелеными точками. Если нет, то точки не были помещены точно внутрь портов, и их нужно будет туда передвинуть.

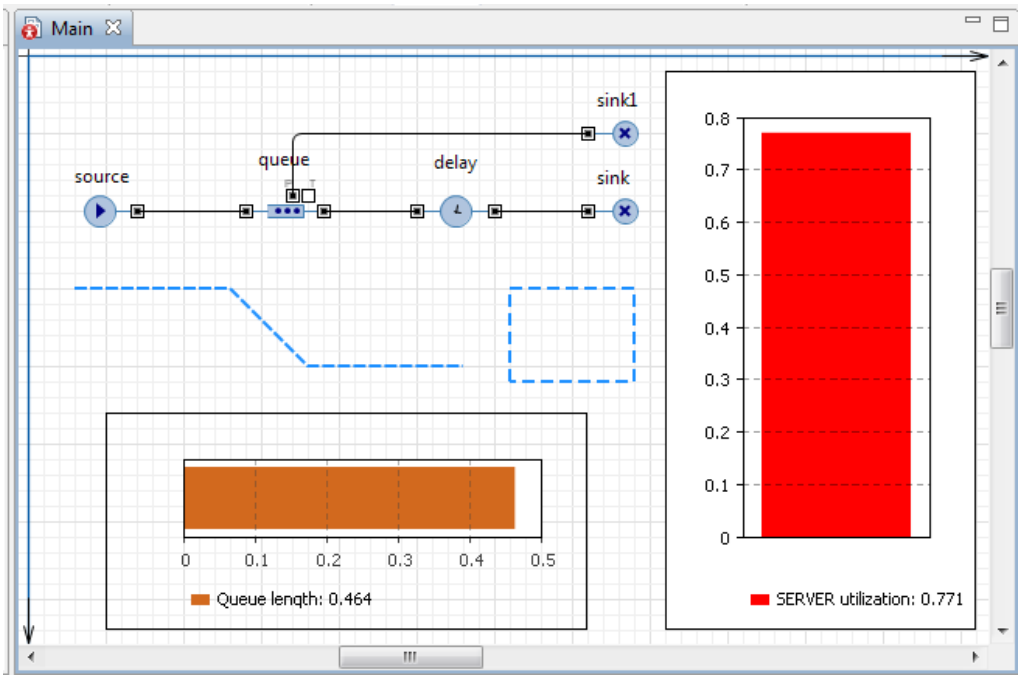


Рис. 1.29. Уточненная модель

6. Запустите уточненную модель и понаблюдайте за ее работой. Сравните рис. 1.30 с рис. 1.22. На рис. 1.30 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.

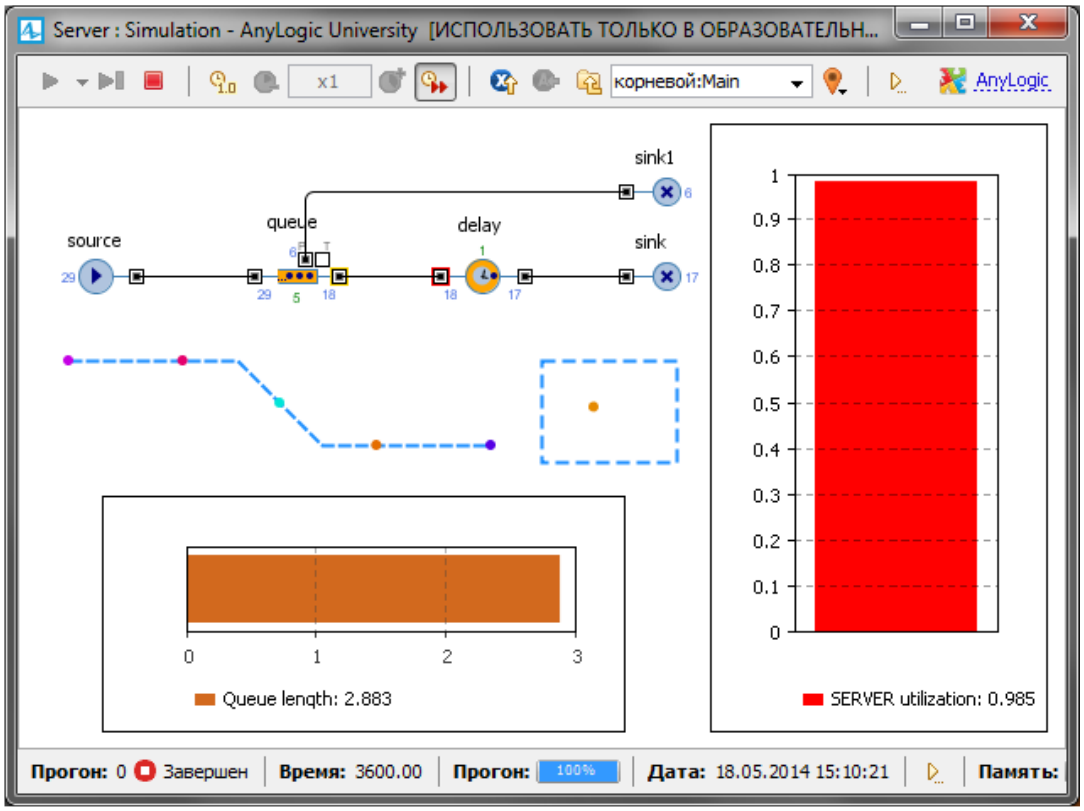


Рис. 1.30. Работа модели согласно ёмкости входного буфера

1.7. 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**. Entity по существу является обычным Java классом с теми функциональными возможностями, которые необходимы и достаточны для обработки и отображения анимации заявки объектами **Библиотеки моделирования процессов**. Эти функциональные возможности можно расширить добавлением дополнительных полей и методов и работой с ними из объектов диаграммы, описывающей моделируемый процесс.
Согласно постановке задачи нужно определять математическое ожидание времени и вероятности обработки запросов сервером.

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

Замечание. Ничего необычного во введённых дополнительных полях нет. Это параметры реальных элементов потоков, в данном случае запросов. AnyLogic предоставляет возможность создавать запросы с теми параметрами, которые необходимы в модели.

Дополнительное задание (*): добавьте анимацию для заявок

Вариант 2

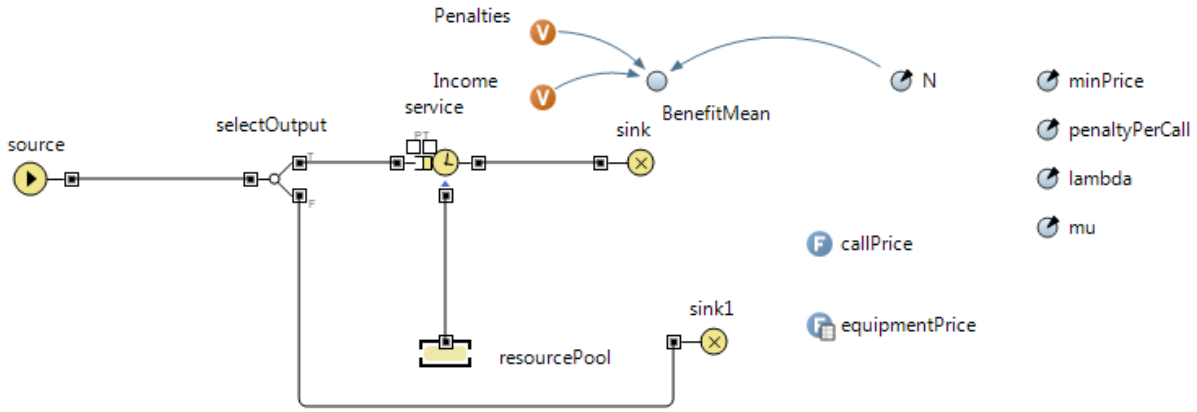
Задача Эрланга

Задача Эрланга связана с разработкой системы массового обслуживания для телефонной сети.

Постановка задачи.

В систему поступают заявки – телефонные вызовы с интервалом времени между вызовами, который соответствует экспоненциальному закону распределения с интенсивностью λ=1.5. Заявки обслуживаются в процессоре. Обслуживание заявки выполняется с интенсивностью λ =0.5. При обслуживании вызова используется N каналов телефонной сети. Число каналов телефонной станции изменяется от 1 до 100.

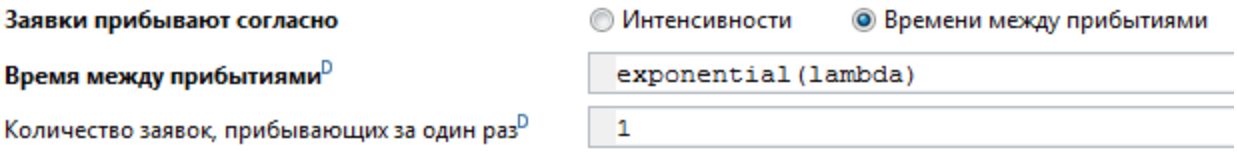
Задание: Постройте модель обслуживания телефонных вызовов в соответствии с описанием. Вид модели в AnyLogic 6 показан на рисунке:



Параметры модели: N- число каналов – линий связи, lambda (λ) – интенсивность прихода заявок, mu (μ) – интенсивность обслуживания заявок в процессоре – service. Параметр N должен быть целого типа int .

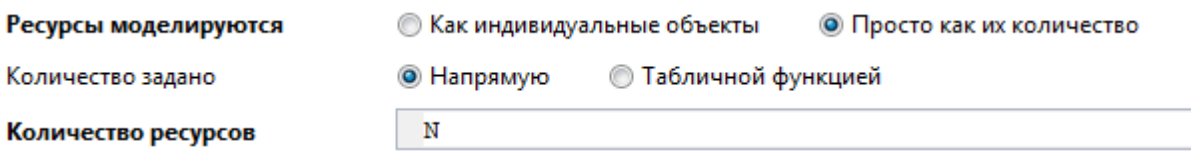
Рекомендации:

1. Элемент source – источник заявок. Его настройки показаны на рисунке:

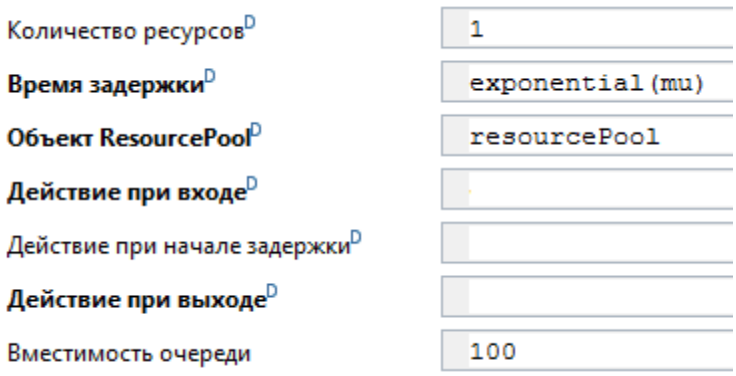


Для моделирования интервала времени используется встроенная функция AnyLogic exponential.

2. Процессор обслуживания использует ресурсы – каналы телефонной сети, которые моделируются элементом resourcePool. Настройки элемента должны соответствовать рисунку:



3. Процессор обслуживания моделируется элементом service. Его настройки показаны на рисунке:



4. Время обслуживания заявки подчиняется экспоненциальному закону. Система обслуживания телефонных вызовов представляет собой двух канальную систему. Первый основной канал, в котором обслуживаются вызовы, второй канал служит для приема отвергнутых системой вызовов.
5. Распределение заявок-вызовов выполняет элемент selectOutput. Его настройки показаны на рисунке:

6. Выполните настройку эксперимента модели:

- Моделирование останавливается в заданное время.
- Модельное время измеряется в минутах.
- Конечное время моделирования равно 500 минут.
- Презентация прорисовывается со скоростью 8 единиц.

Определение расходов на обслуживание телефонных вызовов

7. Требуется определить такое количество каналов связи N, чтобы при заданной интенсивности поступления звонков λ и интенсивности их обслуживания μ прибыль от использования системы была бы максимальной. Для вычисления прибыли используйте формулу:

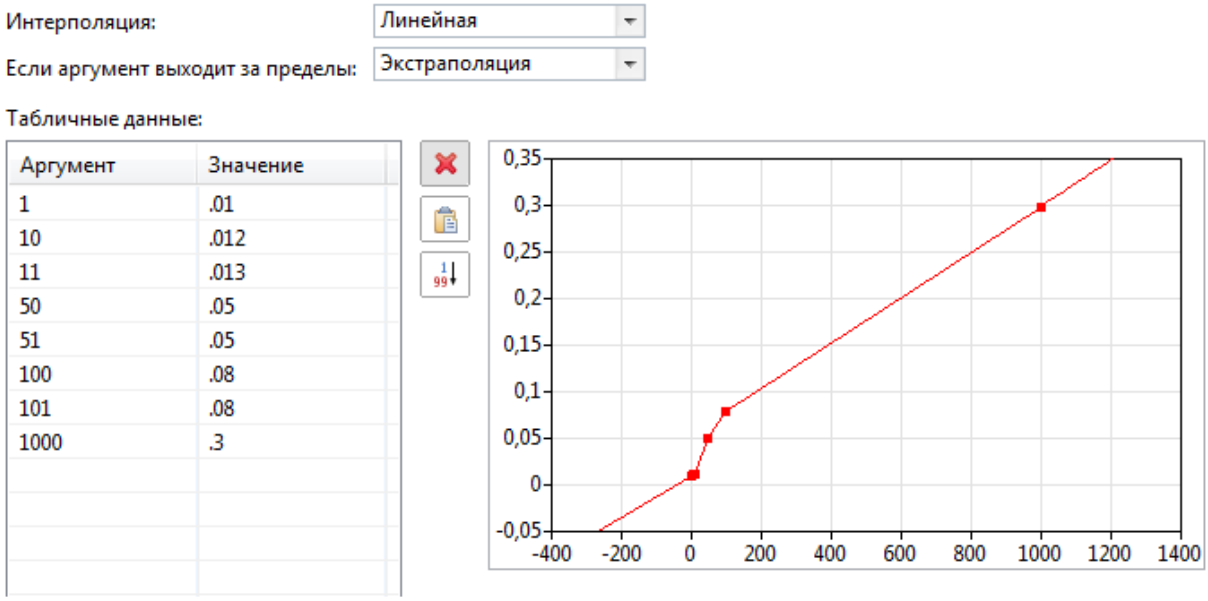
$$BenefitMea\,n = \frac{Income - Penalties}{t} - equipmentPrice(N)$$

Здесь: t – текущее время, Income – стоимость обслуженных звонков, Penalties – начисленный штраф за не обслуженный вызов, equipmentPrice – функция, которая определяет стоимость облуживания задействованных каналов.

8. Введем в модель системы еще два параметра: стоимость обслуживания звонка клиента и стоимость штрафа, за не обслуженный вызов.

9. Создайте функцию для вычисления стоимость обслуженного звонка.

10. Для получения стоимости обслуживания каналов связи нужно в модель ввести табличную функцию, со следующими свойствами:



11. Для использования функции callPrice нужно знать время, затраченное на обслуживание заявки в процессоре service. Что бы вычислить это время введите в модель тип агента заявки Call. Он содержит два атрибута: атрибут для фиксирования времени начала обслуживания вызова в процессоре service и атрибут для фиксирования момента времени завершения обслуживания заявки в процессоре.

12. Настройте модель таким образом, что бы она учитывала прохождение по каналам системы заявки типа агента Call. Определите, используя атрибуты, время начала обслуживания вызова и время завершения обслуживания вызова в процессоре service.

13. При уничтожении заявки в элементе sink должен выполняться оператор, для подсчета стоимости обслуженного вызова.

14. При уничтожении заявки отклоненного вызова должен выполняться код, для подсчета стоимости штрафов.

15. Создайте динамическую переменную для вычисления прибыли по формуле:

$$BenefitMea\,n = \frac{Income - Penalties}{t} - equipmentPrice(N)$$

16. Измените эксперимент модели. Создайте интерфейс для изменения значения числа каналов N с помощью «бегунка».

17. Проведите несколько экспериментов с разными значениями N. Проследите, как меняется прибыль от эксплуатации системы с разным числом каналов.

Дополнительное задание(*): определите оптимальное число каналов

Подсказка: Что бы определить это оптимальное число каналов, при котором прибыль от использования системы будет максимальная, нужно использовать оптимизационную подсистему. Введите в модель новый – оптимизационный эксперимент.

Вариант 3

Модель пешеходного перехода

1.1. Постановка задачи

Построить модель регулируемого пешеходного перехода со светофором, разрешающим или запрещающим движение транспорта

Светофор, регулирующий движение автомобилей на пешеходном переходе, может находиться в следующих состояниях: движение транспорта разрешено (зеленый), приготовиться к запрещающему сигналу (мигающий зеленый), приготовиться к остановке (желтый), движение запрещено (красный) и приготовиться к движению (красный и желтый). Светофор работает в автоматическом режиме. В каждом состоянии светофор находится определенный постоянный период времени.

1.2. Построение модели

1) Создайте новый проект под названием Svetofor
2) Постройте диаграмму состояний *Для_автомобилей,* как на рис.1

Рисунок 1

3) Задайте условия срабатывания переходов:
   - В состоянии *движение* светофор находится 10 секунд,
   - Затем 7 секунд зеленый сигнал мигает,
   - В состоянии *медленно* 4 секунды горит желтый

- В течение 10 секунд движение запрещено и 5. 4 секунды светофор находится в состоянии *приготовиться*.
- В модели единица модельного времени соответствует 1 секунде реального времени.

Для задания условий срабатывания переходов, выделите переход *t1*, и в поле *Происходит* оставьте без изменения вариант **По таймауту**, а в поле *По таймауту* введите 10 (рис2)

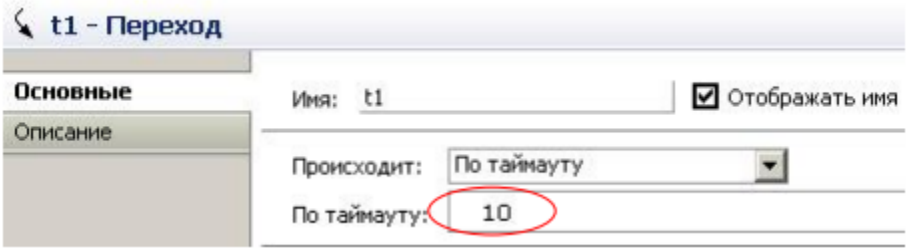


Рисунок 2

4) Создайте три параметра логического типа: *красный*, *желтый* и *зеленый*, которые будут принимать истинное значение тогда, когда у светофора горит соответствующий сигнал: красный, желтый или зеленый. Начальные значения этих булевых параметров можно не задавать: по умолчанию они будут равны *false*.
5) В свойствах состояния *движение* в поле **Действие при входе** запишите: *зеленый=true;* в поле **Действие при выходе** запишите: *зеленый=false;*
6) В остальных свойствах пропишите аналогично, чтобы выполнялось следующее: в состоянии *медленно* должен гореть желтый, в состоянии *stop* должен загореться красный свет, а в состоянии *приготовиться* должны гореть красный и желтый одновременно, в состоянии *Внимание* желтый должен моргать.
7) Нарисуйте светофор, используя графические элементы (простейший вариант – нарисовать три овала).
8) У каждого овала цвет сделайте динамическим и запишите следующие условия соответственно:

красный? red: gray
желтый? yellow: gray
зеленый? green: gray

9) Запустите модель и проверьте ее работу.
10) Добавьте второй светофор для пешеходов: он будет иметь два сигнала (зеленый и красный) и три состояния: идите, внимание и стойте (рис.3)

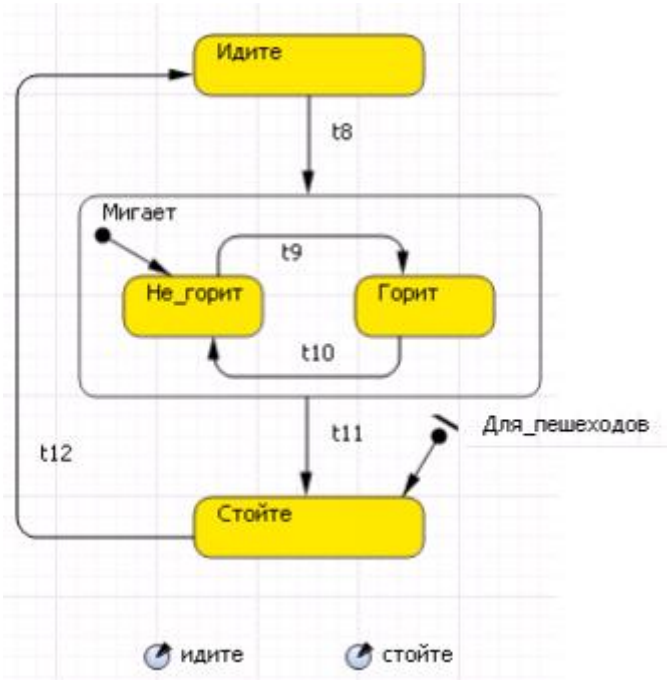


Рисунок 3

11) Настроим условия срабатывания переходов стейтчартов между состояниями. Необходимо синхронизировать два светофора для того, чтобы когда светофор для пешеходов находился в состоянии «Идите» или «Мигает», светофор для автомобилей был в значении «Stop».

В нашей модели светофоры будут обмениваться сообщениями «Автомобили» и «Пешеходы». Мы будем использовать метод fireEvent (), который должен вызываться в том стейтчарте, которому предназначено сообщение.

В t5 в поле «Действие» вставьте команду Для_пешеходов.fireEvent("ПЕШЕХОДЫ"), в поле t11 вставьте Для_автомобилей.fireEvent("АВТОМОБИЛИ") – рис.4

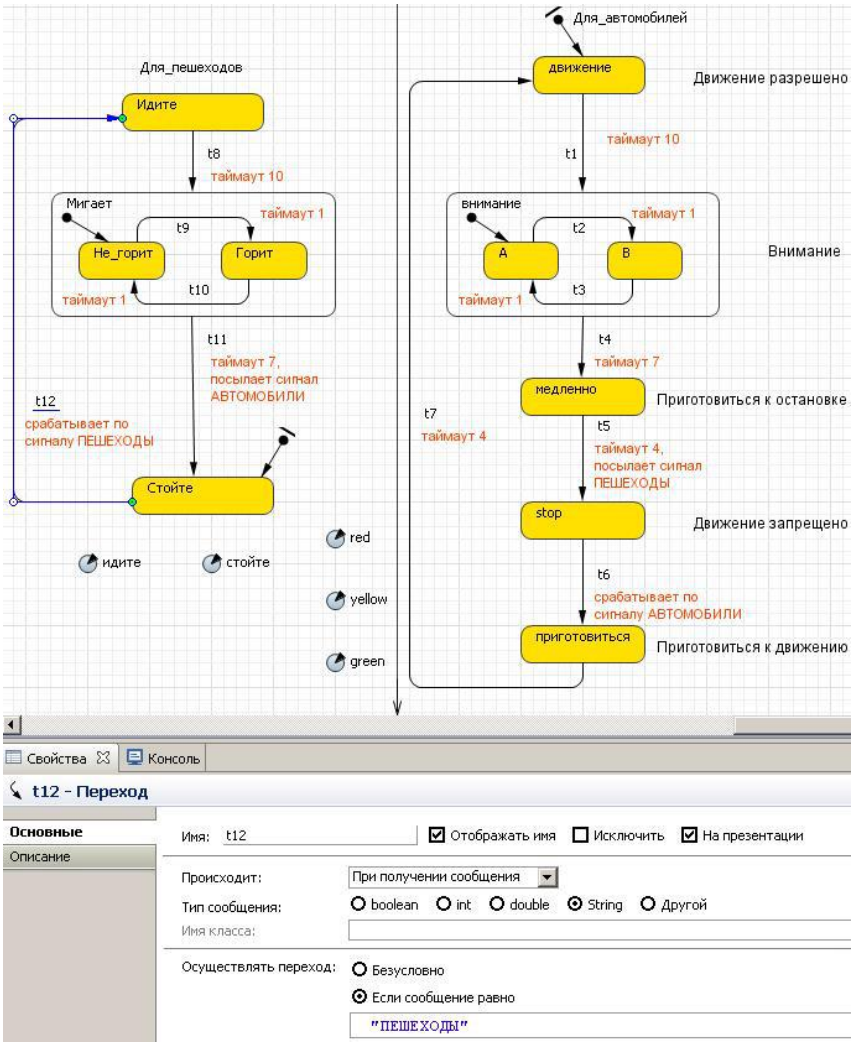


Рисунок 4

12) Для срабатывания перехода в t12 в поле «Происходит»: При получении сообщения, «Тип сообщения»:String, «Осуществлять переход»: **Если сообщение равно** введите "ПЕШЕХОДЫ" (как на рис.4).
13) Аналогично в t6 введите "АВТОМОБИЛИ".
14) На изображении светофора автомобиля дорисуйте светофор для пешеходов с двумя сигналами: красной надписью СТОЙТЕ и зеленой ИДИТЕ. Аналогично цветам создайте два параметра СТОЙТЕ и ИДИТЕ.
15) Запустите модель, проверьте работу светофоров.

Дополнительное задание (*):

16) Измените модель автоматического светофора таким образом, чтобы по кнопке **НОЧЬ** включался режим мигания желтого света, а по кнопке **ДЕНЬ** светофор переходил в нормальный режим работы.

Вариант 4

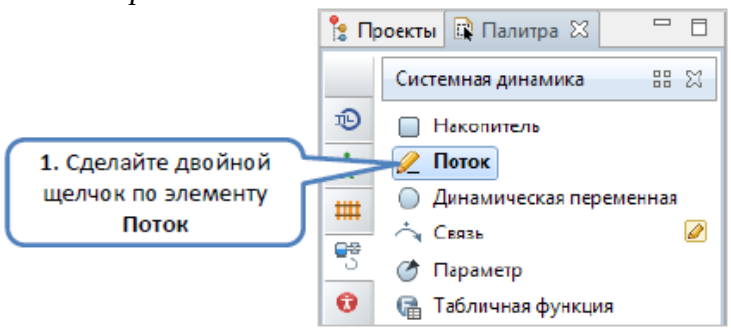
Модель распространения нового продукта по Бассу.

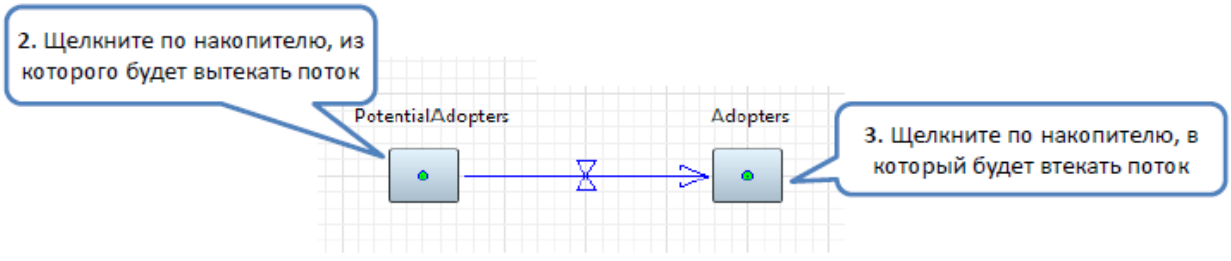
1.1. Постановка задачи

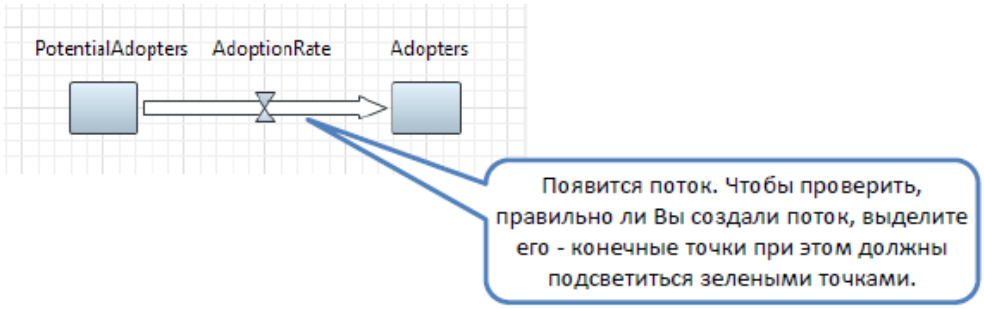
Необходимо создать модель распространения нового продукта по Бассу. Модель Басса описывает процесс распространения продукта. Изначально продукт никому не известен, и для того, чтобы люди начали его приобретать, он рекламируется. В итоге определенная доля людей приобретает продукт под воздействием рекламы. Также люди приобретают продукт в результате общения с теми, кто этот продукт уже приобрел

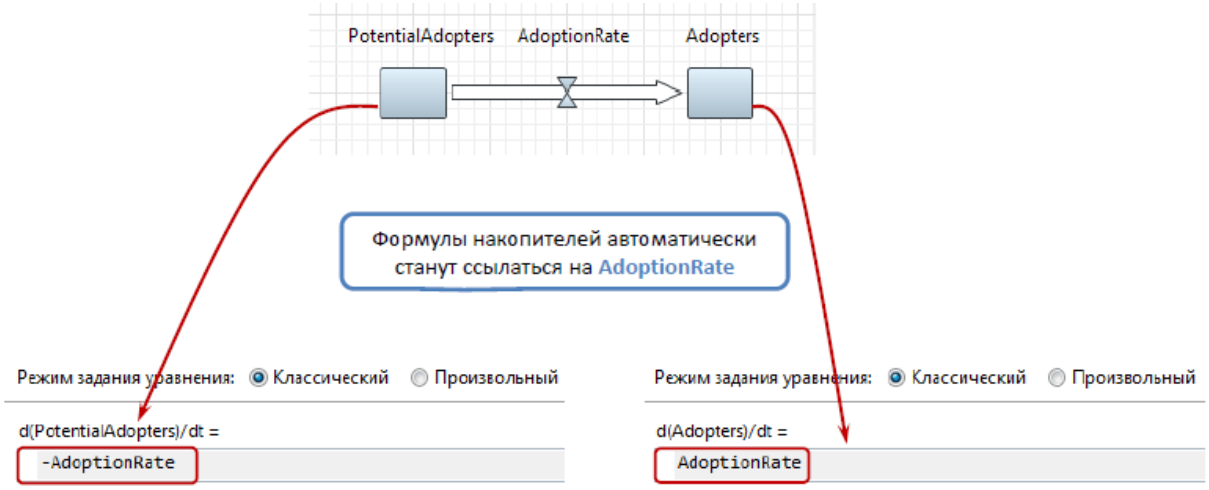
1.2. Построение модели

1. Создайте новую модель и назовите ее *Bass Diffusion*
2. Откройте свойства модели и выберите *минуты* в качестве единиц модельного времени
3. Добавьте на диаграмму два *Накопителя* (палитра **Системная динамика**): *PotentialAdopters* и *Adopters*.
4. Создайте поток, ведущий из *PotentialAdopters* в *Adopters*. Назовите поток *AdoptionRate*.

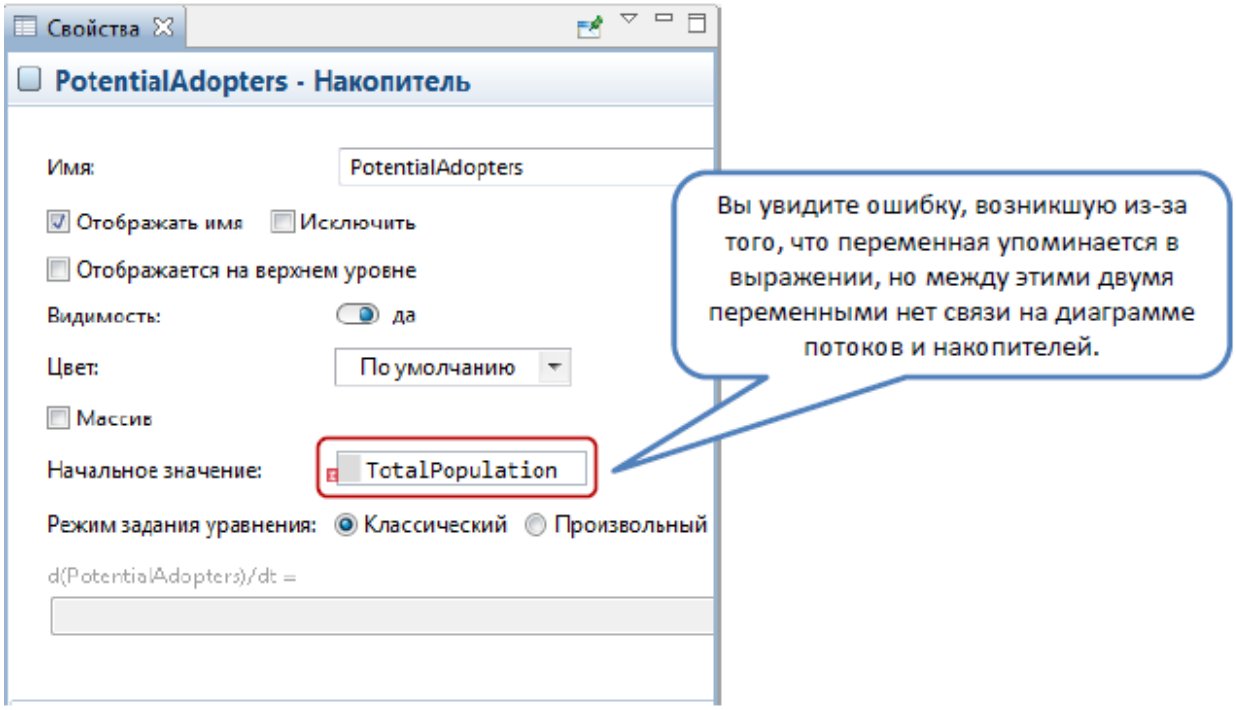




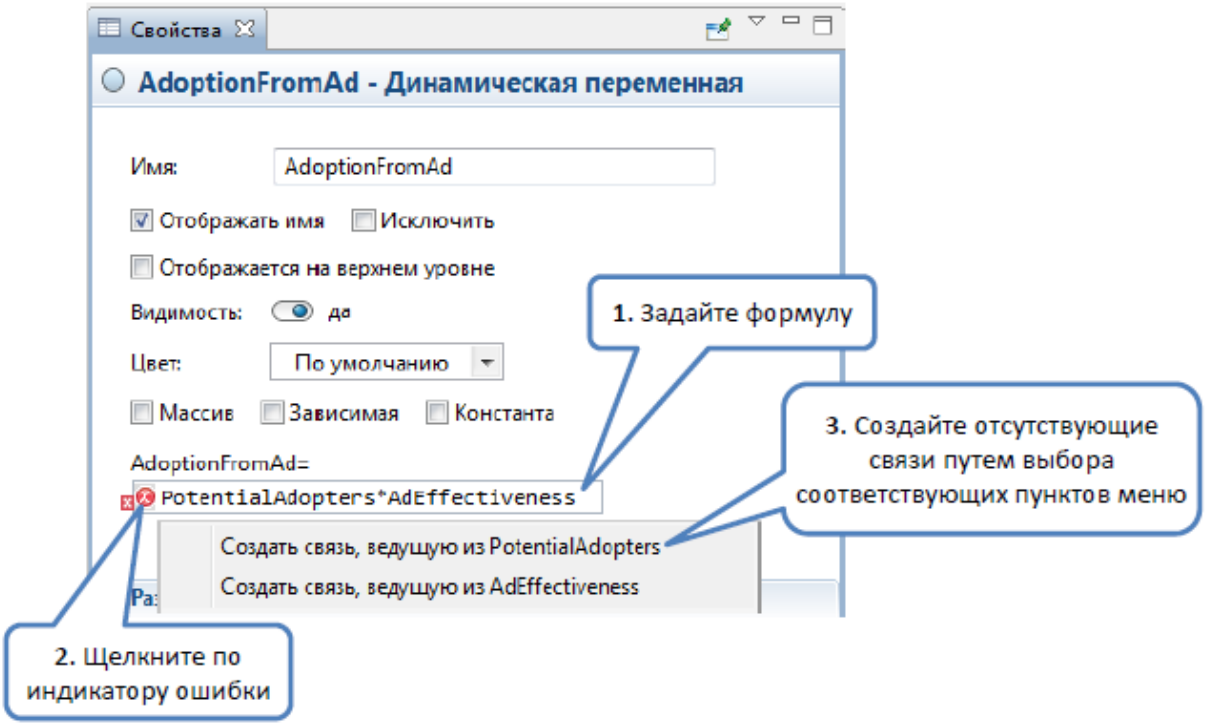


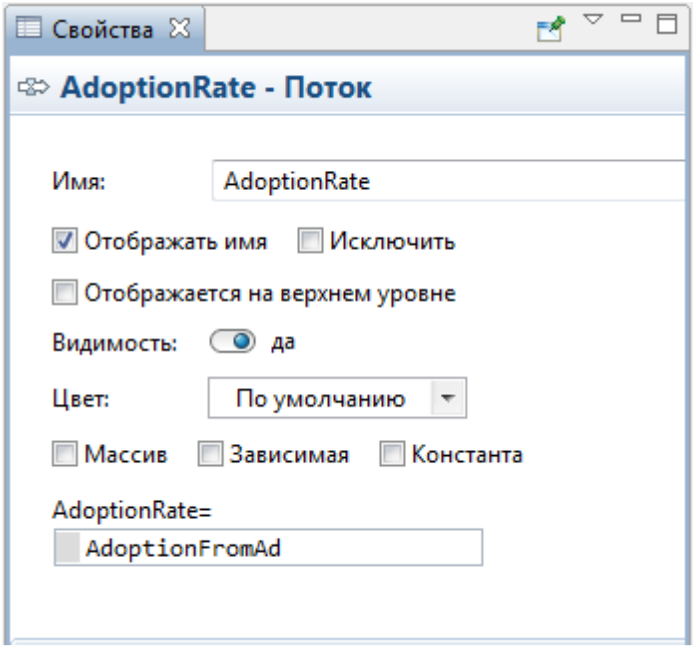


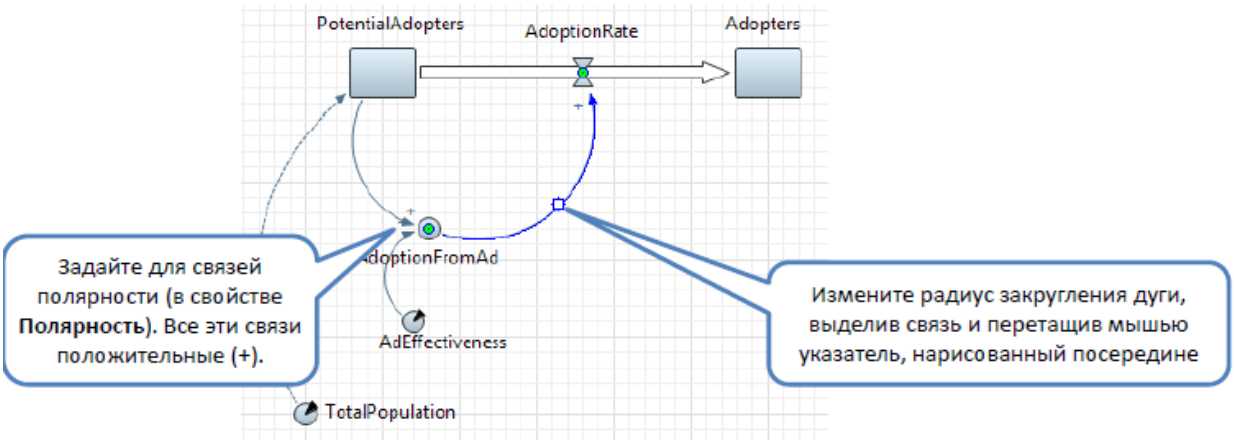
5. Создайте параметры: *TotalPopulation* (значение по умолчанию: 10000, тип: double) и *AdEffectiveness* (значение по умолчанию: 0.011, тип: double).
6. Задайте начальное значение накопителя *PotentialAdopters*.



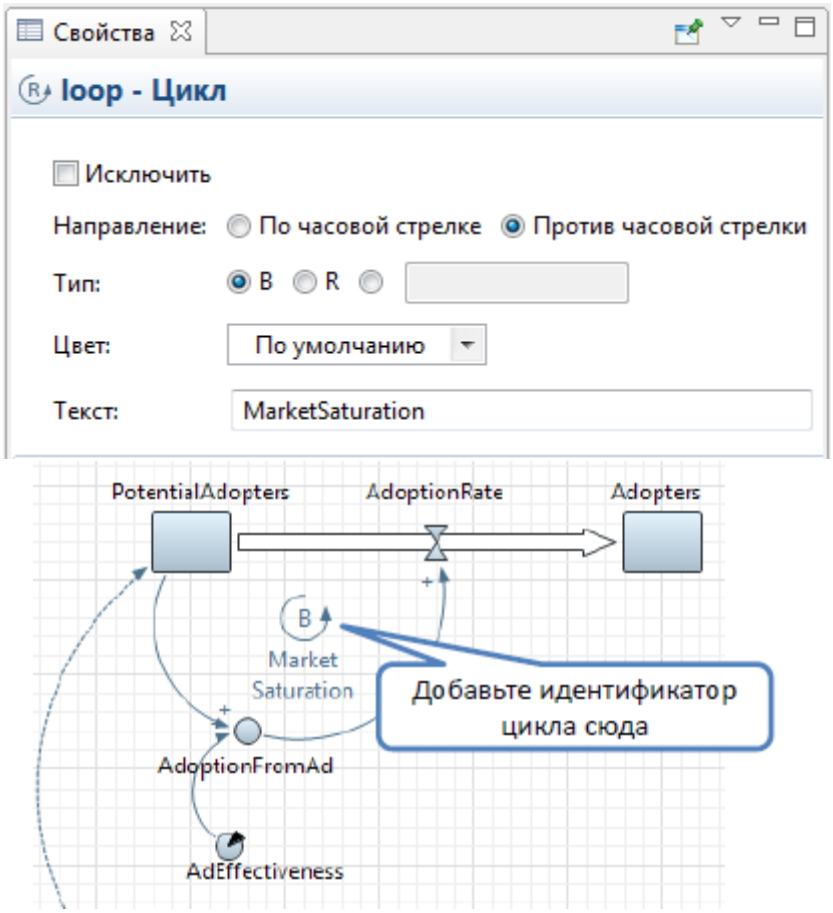
7. Нарисуйте связь, задающую причинную зависимость между*TotalPopulation* и *PotentialAdopters*.
8. Добавьте Динамическую переменную *AdoptionFromAd*. Нарисуйте связь, ведущую от *AdoptionFromAd* к *AdoptionRate*. Задайте формулу для *AdoptionRate*.



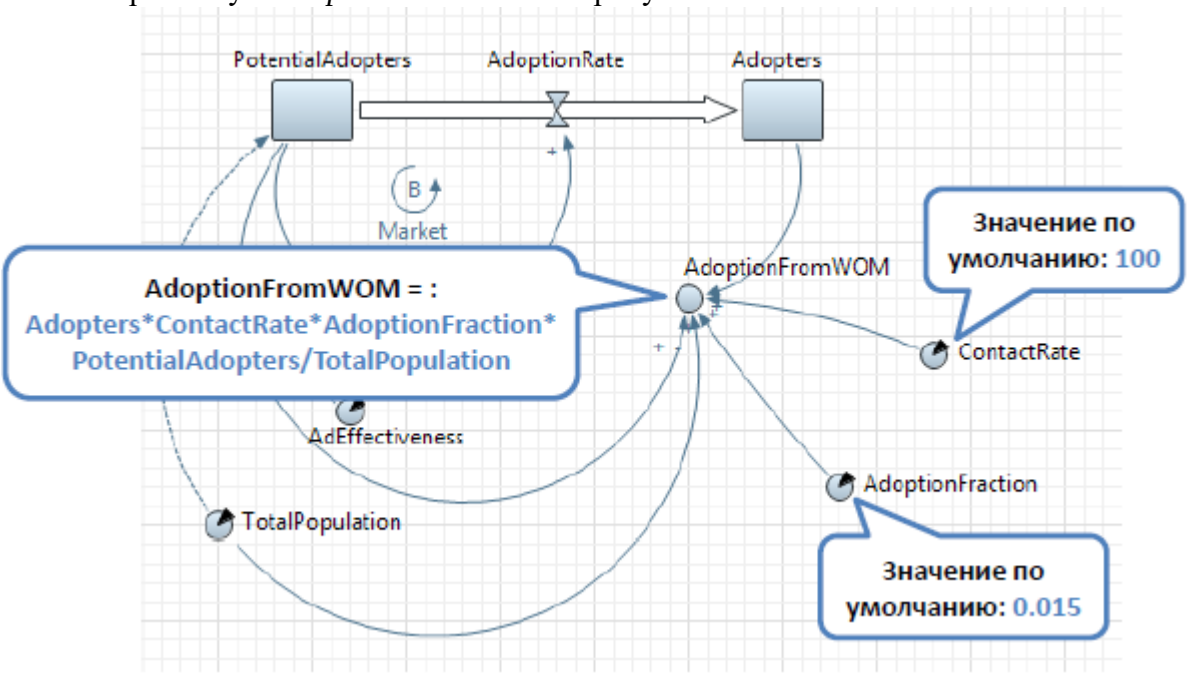




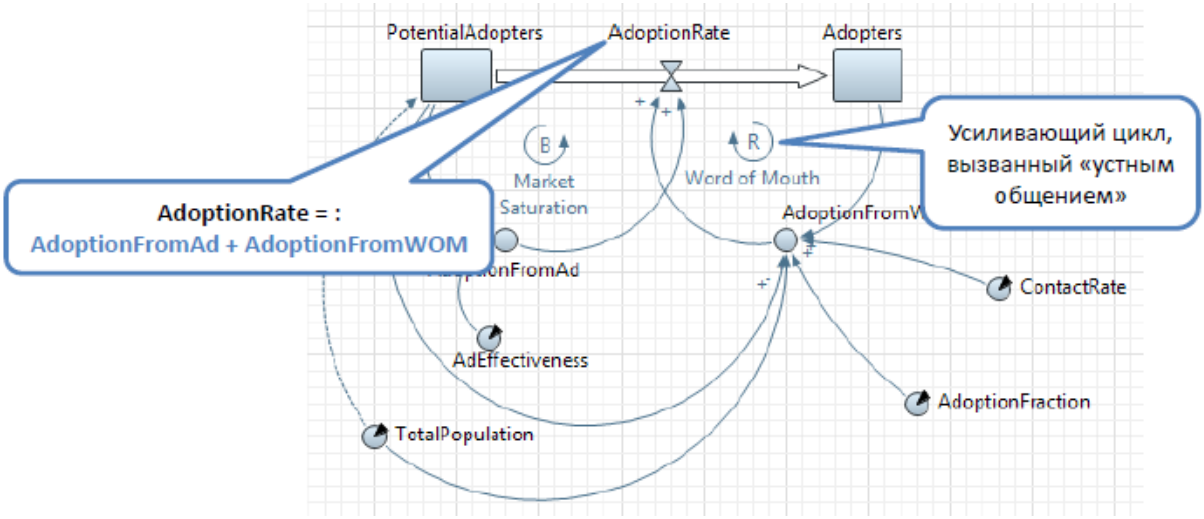
9. Добавьте *Цикл*, обозначающий уравновешивающий цикл зависимостей, вызванный насыщением рынка. Задайте направление, тип и краткое описание цикла



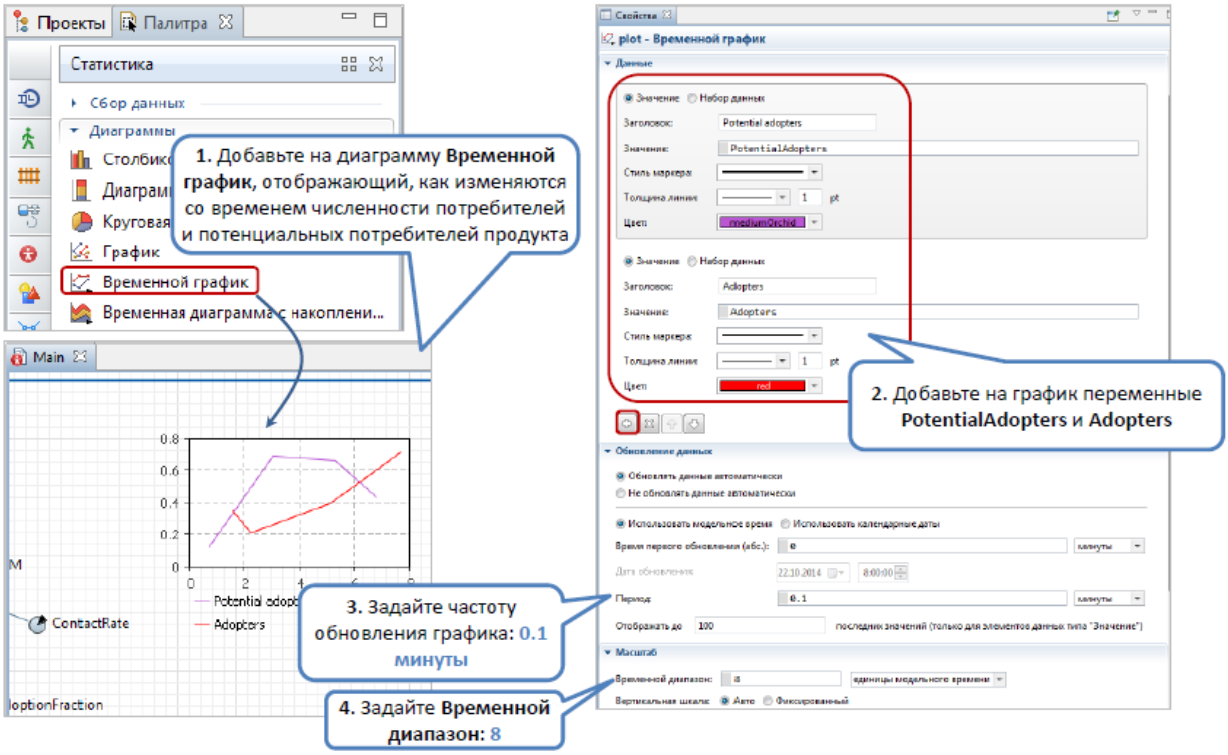
10. Создайте параметры *ContactRate* и *AdoptionFraction* . Создайте динамическую переменную *AdoptionFromWOM*. Нарисуйте связи зависимостей.



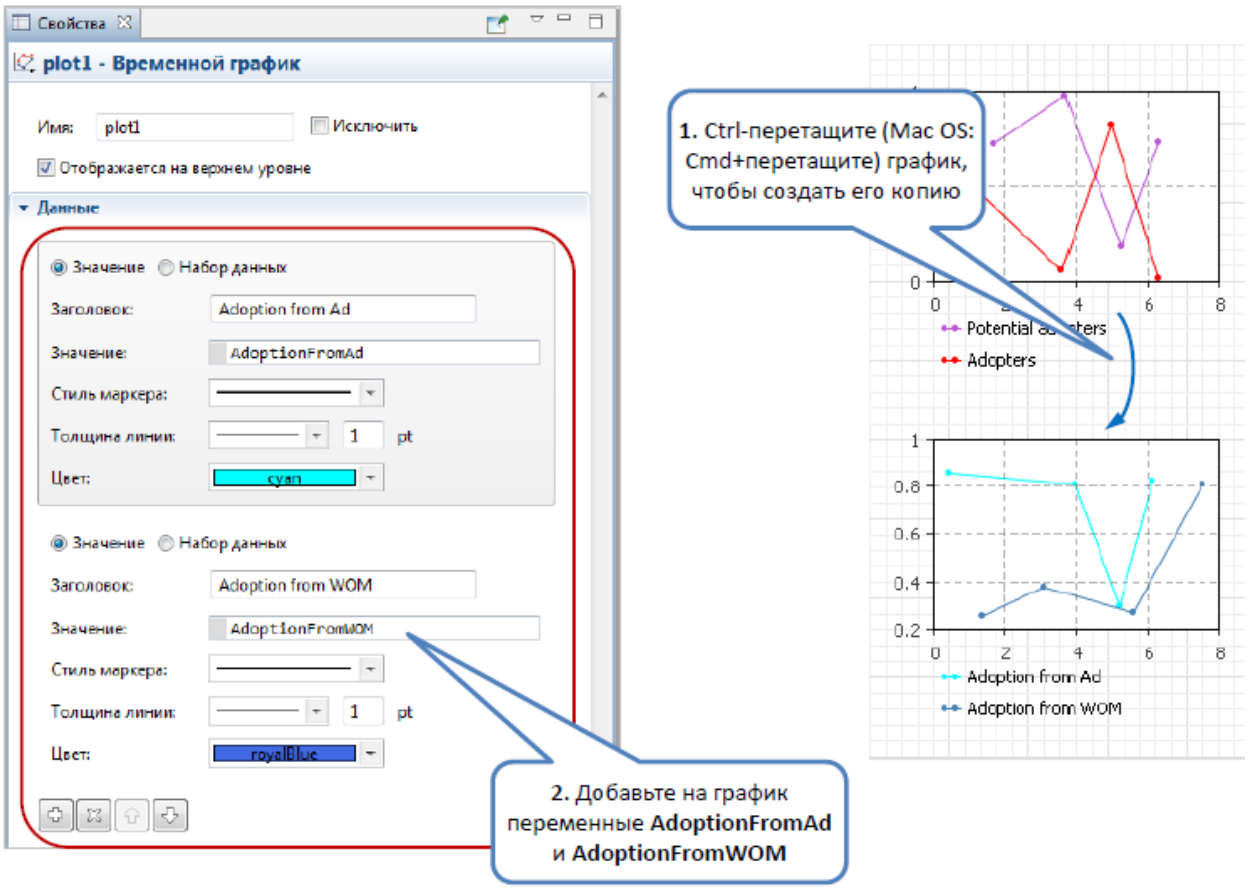
11. Нарисуйте связь, ведущую из *AdoptionFromWOM* в *AdoptionRate*. Измените формулу *AdoptionRate*. Добавьте еще один идентификатор цикла.


PotentialAdopters
AdoptionRate
Adopters
B
Market Saturation
R
Word of Mouth
AdoptionFromW
AdoptionRate = :
AdoptionFromAd + AdoptionFromWOM
Усиливающий цикл, вызванный «устным общением»
ContactRate
AdEffectiveness
AdoptionFraction
TotalPopulation

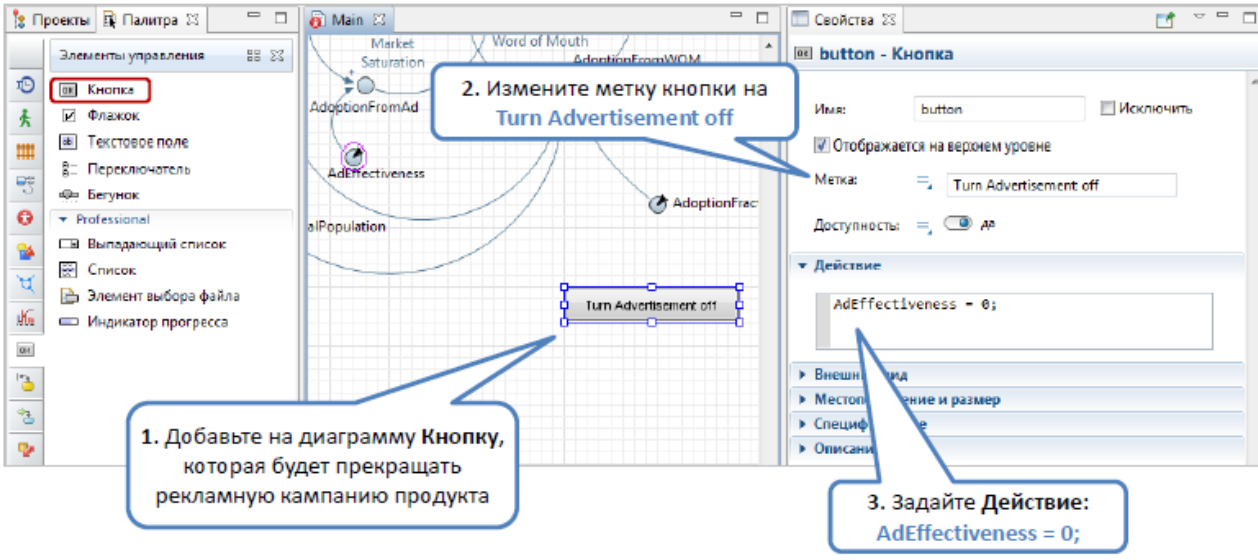
12.


Проекты
Палитра
Статистика
Сбор данных
Диаграммы
Временной график
1. Добавьте на диаграмму Временной график, отображающий, как изменяются со временем численности потребителей и потенциальных потребителей продукта
Main
ContactRate
Potential adopt
Adopters
plot - Временной график
2. Добавьте на график переменные PotentialAdopters и Adopters
3. Задайте частоту обновления графика: 0.1 минуты
4. Задайте Временной диапазон: 8

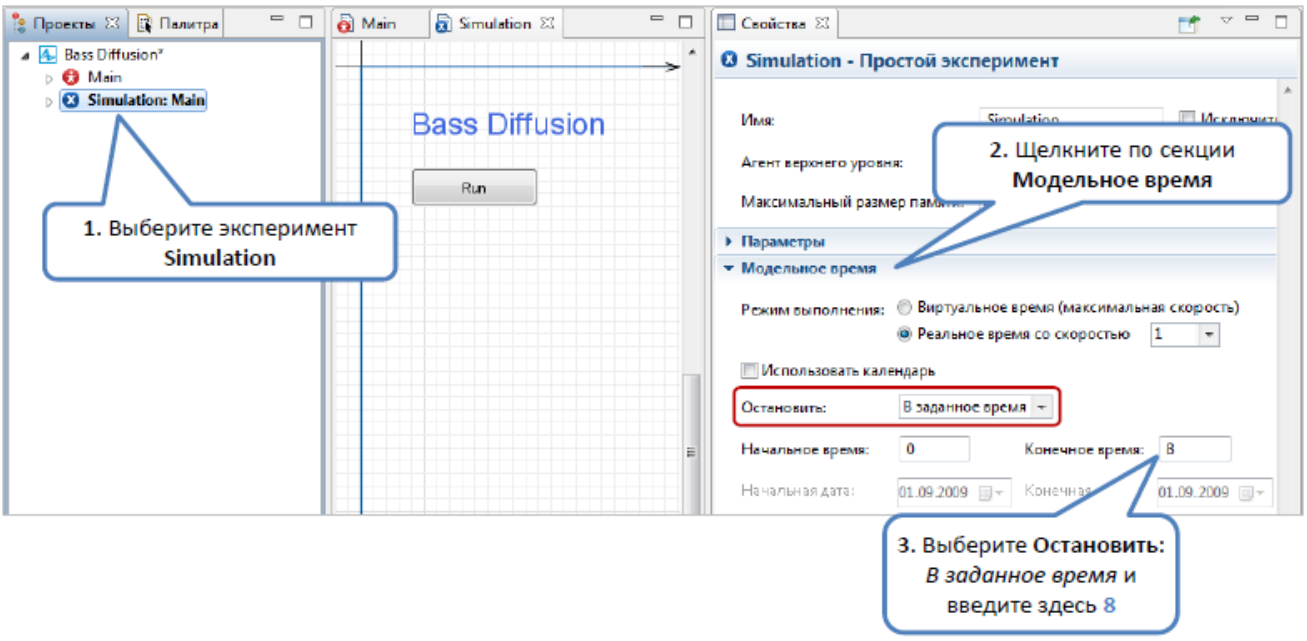
13.



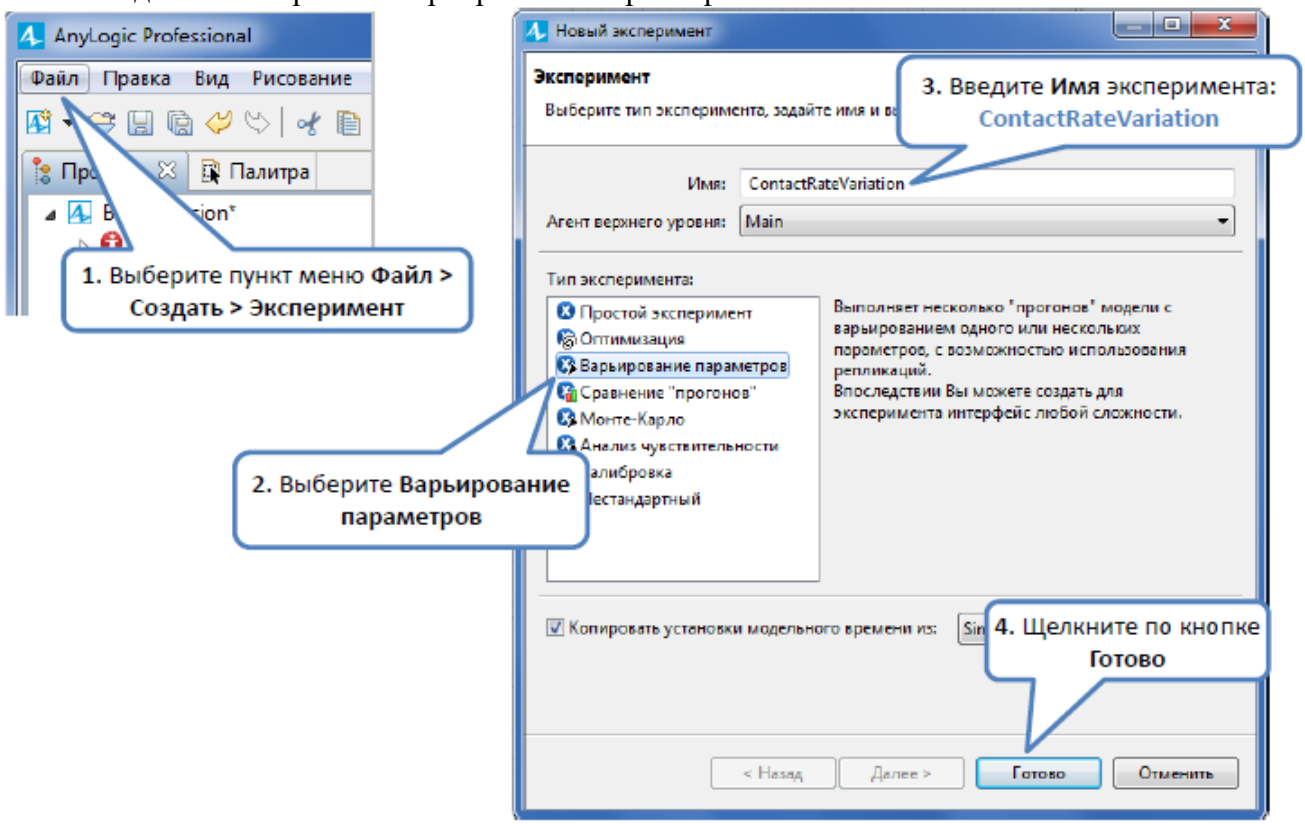
14.



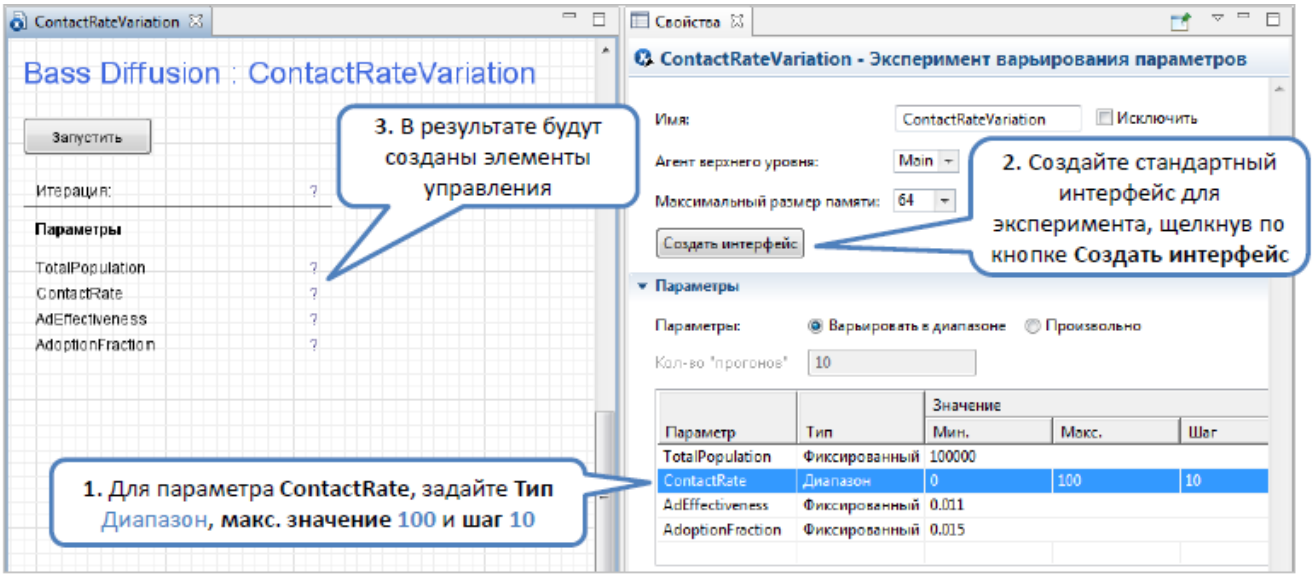
15. Задайте время остановки модели



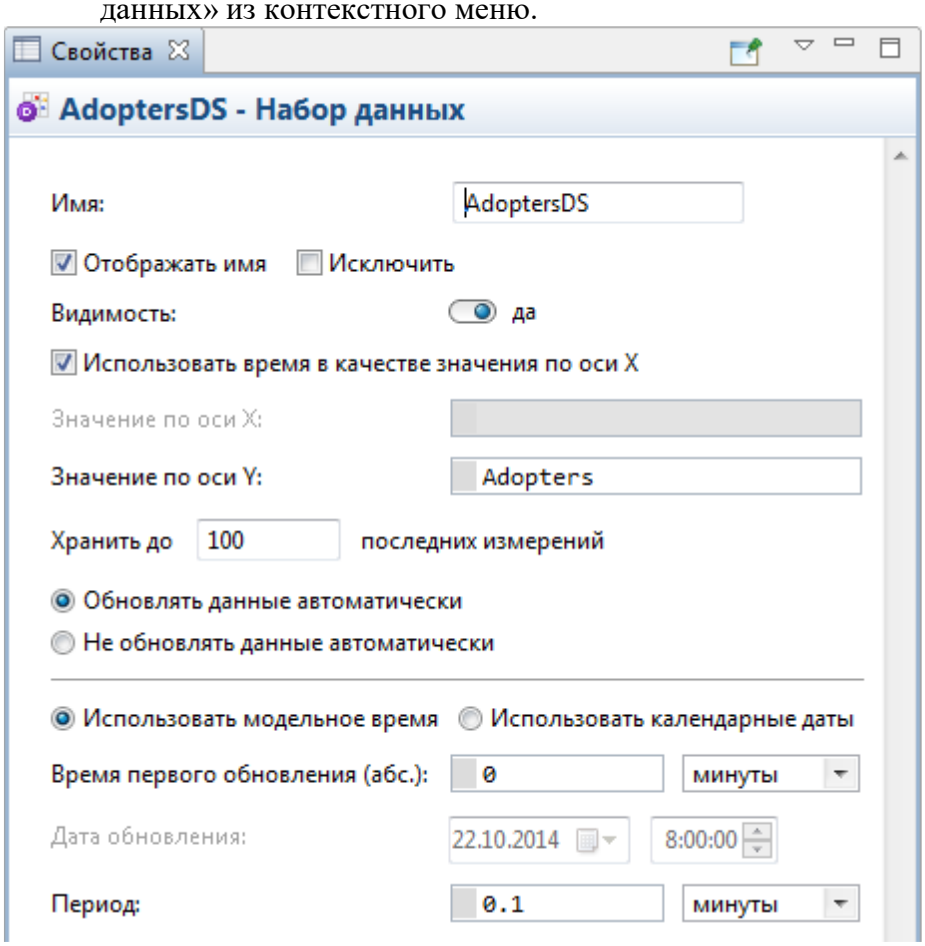
16. Запустите модель и проследите за динамикой с помощью графиков. Прекратите рекламу и проследите, как это отразится на процессе.
17. Создайте эксперимент варьирования параметров.



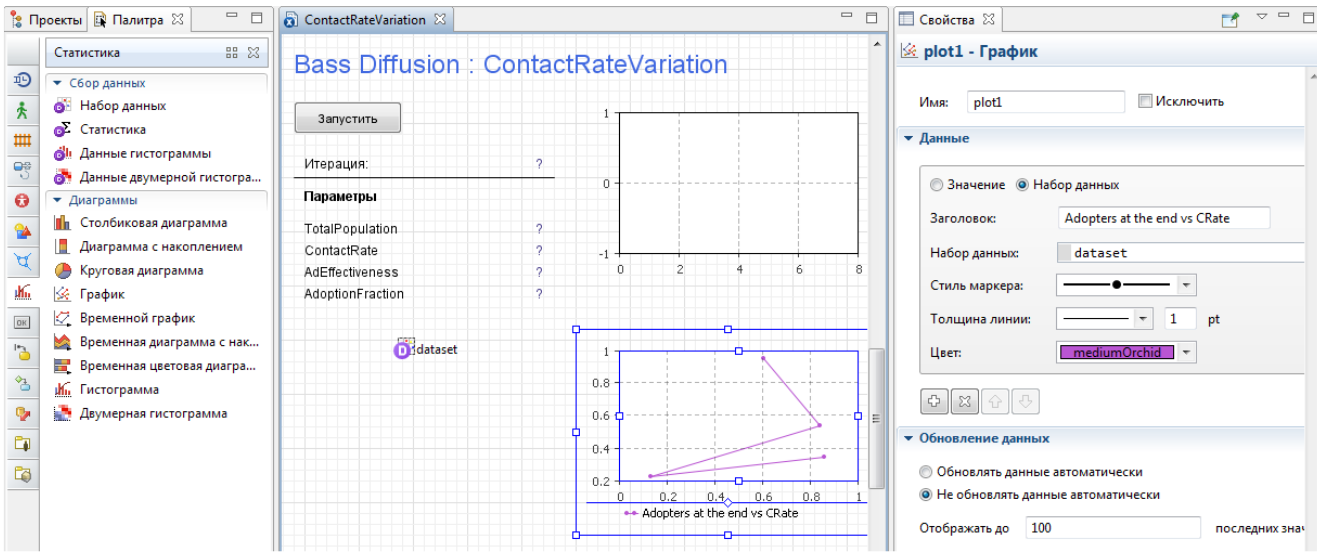
18. Задайте настройки эксперимента:пусть эксперимент варьирует параметр *ContactRate* в диапазоне от 0 до100 с шагом10.



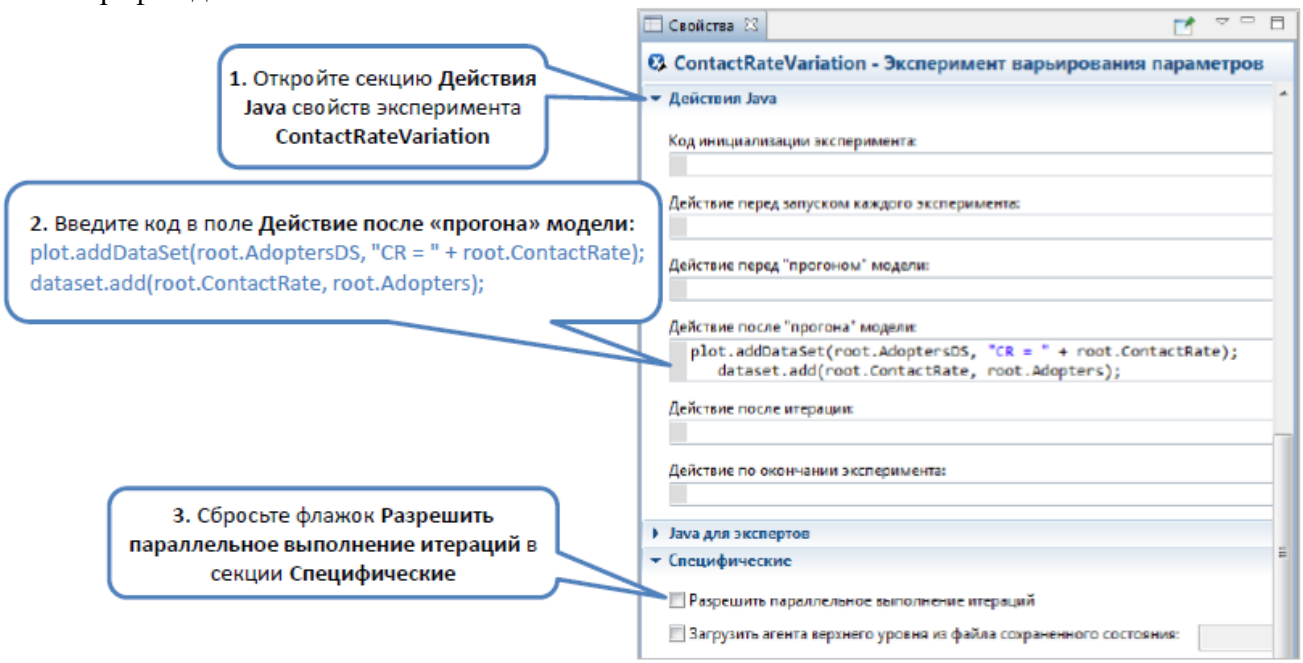
19. Поместите временной график на диаграмму эксперимента *ContactRateVariation*: выберите **Не обновлять данные автоматически** и задайте **Временной диапазон: 8.**
20. Создайте набор данных для хранения истории переменных *Adopters*: щелкните правой кнопкой мыши по переменной *Adopters* и выберите «Создать набор данных» из контекстного меню.



21. Добавьте Набор данных на диаграмму эксперимента *ContactRateVariation*. Добавьте *График* для отображения зависимости числа клиентов на момент окончания моделирования от интенсивности контактов.



22. Измените свойства эксперимента, чтобы в конце каждого «прогона» модели на график добавлялись новые значения



23. Запустите эксперимент *ContactRateVariation*. Пронаблюдайте, как процесс зависит от интенсивности контактов.

Дополнительное задание(*):
Смоделируйте повторные покупки, полагая, что потребители продукта снова становятся потенциальными потребителями, когда продукт, который они приобрели, становится непригоден.

Вариант 5

Физический маятник

1.1. Постановка задачи

Построить модель маятника. Тело прикреплено к кронштейну нитью длиной L. Исследовать колебательные движения тела.
Тело – материальная точка с массой 1. Движение тела по второму закону Ньютона.
На тело воздействует три силы:

- сила тяжести – mg,
- реакция натяжения нити,
- сила сопротивления воздуха (пропорциональна квадрату скорости движения).

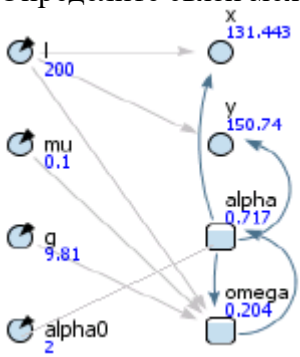
Тело совершает колебательные движения т.к. mg уравновешивается вертикальной составляющей реакции натяжения нити.
С математической точки зрения это задача Коши.

$$x = l \cdot \sin(\alpha); \quad y = l \cdot \cos(\alpha) \qquad \begin{matrix} \alpha(0) = \alpha 0; \\ \omega(0) = 0 \end{matrix} \qquad \frac{d\omega}{dt} = -\frac{g \cdot \sin(\alpha)}{l} - \mu \cdot \omega \cdot |\omega| \qquad \frac{d\alpha}{dt} = \omega;$$

Где α - текущий угол отклонения маятника от вертикали, ω - его угловая скорость, μ - коэффициент сопротивления среды.

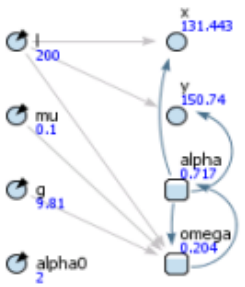
1.2. Построение модели

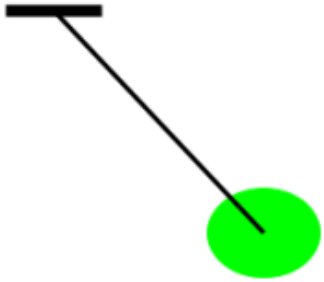
1. Создайте новую модель в AnyLogic.
2. Создайте две переменные состояния alpha и omega (интегралами в соответствии с формулами Коши, начальная угловая скорость omega - 0)
3. Создайте две переменные x и y (формулами Коши).
4. Создайте четыре параметра L, mu, g, alpha0(начальное значение alpha).
5. Определите связи между объектами, как показано на рисунке.

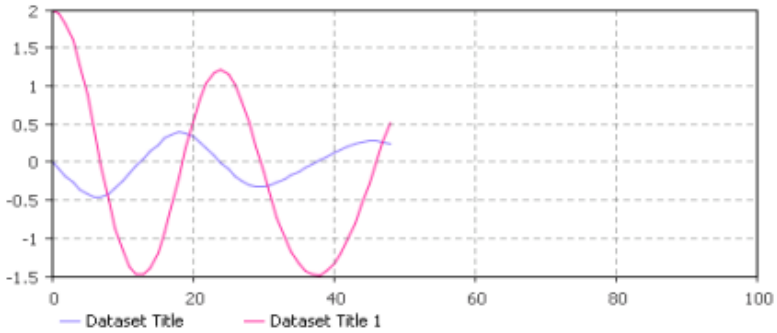


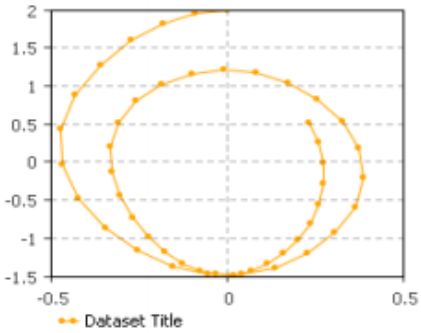
6. Нарисуйте маятник (две линии и овал). Линия это нить, на одном конце координаты (0,0), на другом координаты (x,y). Координаты вводятся в Свойствах линии во вкладке **Динамические.** У овала во вкладке Динамические координаты (x,y).
7. Создайте область с названием модели и пояснением. Ниже создайте два поля с вводом данных переменных L и mu.
8. Создайте график зависимости alpha и omega от времени.
9. Создайте фазовую диаграмму зависимости alpha от omega.
10. Запустите модель. Поэкспериментируйте с оформлением модели.

Пример отображения модели









Дополнительное задание (*):

11. Доработайте презентацию модели таким образом, чтобы при остановке маятника в верхней точке траектории он менял цвет, ровно на 2 секунды.