МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Ухтинский государственный технический университет»
(УГТУ)

Имитационное моделирование на основе GPSS и AnyLogic

Контрольные работы

Методические указания

Ухта
УГТУ
2017

УДК 519.876.5 (075.8)
ББК 32.81 я7
С 30

Семериков, А. В.

С 30 Имитационное моделирование на основе GPSS и AnyLogic. Контрольные работы [Текст] : метод. указания / А. В. Семериков. – Ухта : УГТУ, 2017. – 44 с.

Настоящие методические указания предназначены для выполнения контрольных работ по дисциплине «Имитационное моделирование». Они включают в себя пояснения для выполнения заданий и варианты заданий для контрольных работ. Методические указания предназначены для студентов дневной и заочной формы обучения по специальности 09.03.02 Информационные системы и технологии.

Содержание указаний соответствует рабочей учебной программе.

УДК 519.876.5 (075.8)
ББК 32.81 я7

Методические указания рассмотрены и одобрены кафедрой ВТИСиТ протокол № 6 от 21.11.16 г. и рекомендованы к изданию.

Рецензент: Рочев К. В., зав. кафедрой ВТИСиТ, к.э.н.

Технический редактор: К. В. Зелепукина.

В методических указаниях учтены замечания рецензента.
Методические указания изданы в авторской редакции с минимальными правками.

План 2017 г., позиция 127.
Подписано в печать 31.03.2017 г. Компьютерный набор.
Объём 44 с. Тираж 100 экз. Заказ № 315.

169300, г. Ухта, ул. Первомайская, 13.
Типография УГТУ
169300, г. Ухта, ул. Октябрьская, 13.

Оглавление

Введение

Моделирование является одним из эффективных способом изучения реального мира. Моделирование позволяет проводить исследования существующих в окружающем мире объектов не с самим объектом, а с его моделью. Анализ, проведённых экспериментов, позволяет принять взвешенное и логически оправданное решение по функциональным возможностям системы. Для принятия решения можно использовать и наблюдения за реально работающей системой, а также использовать интуитивное мышление, накопленный опыт. Однако, такой подход не позволяет всесторонне оценить наблюдаемый объект и может принести большие экономические потери. Кроме того во многих случаях проведение экспериментов над реальной системой просто недопустимо.

Особенно важно проводить моделирование вновь создаваемых систем, так как появляется возможность ещё на стадии проектирования оценить основные параметры и функции объекта. При этом можно сравнить эффективность различных управляющих решений в поисках наиболее подходящего варианта.

Различают два вида моделирования: математическое и имитационное моделирование. Математическое моделирование применяется для создания моделей физических процессов с использованием численных методов. Имитационное моделирование отражает структуру и поведение объекта. Компьютерный эксперимент позволяет установить характеристики, как всей системы, так и её составных частей.

С помощью имитационного моделирования можно решать различные задачи. Существует различная классификация этих задач. Известна [1] следующая классификация: моделирование динамических систем, дискретно-событийное моделирование, системная динамика и агентное моделирование.

Имитационное моделирование представляет собой статистический эксперимент. Его результаты должны основываться на статистических проверках. Имитационный эксперимент должен удовлетворять следующим требованиям:

1. Наблюдения должны иметь стационарные распределения.
2. Наблюдения подчиняются нормальному распределению.
3. Наблюдения независимы.

Соблюдение этих требований гарантирует корректный сбор наблюдений при использовании имитационной модели. На практике выполнение этих требований вызывает большие трудности. Пути выполнения требований к имитационным моделям хорошо известны, хотя не всегда выполнимы [2].

Для разработки имитационных моделей имеются программные средства: Simulink, GPSS, AnyLogic [3, 4]. Эффективное использование этих

инструментальных средств предполагает глубокие знания предметной области моделирования, знания в программировании, статистике, теории вероятности. В настоящем методическом указании используется язык моделирования GPSS (General Purpose System Simulation), который был создан для дискретно-событийного моделирования, и инструментальное средство AnyLogic7 имеющее более широкий спектр применимости.

В рамках курса имитационного моделирования предлагается выполнить две контрольные работы.

Первая из них содержит задание, выполнение которого предполагает написание программы на каком-либо алгоритмическом языке не ориентированном на имитационное моделирование. Для создания программы необходимы знания, как предметной области моделирования, так и какого-либо языка программирования.

Во второй контрольной работе рассматривается дискретно-событийное моделирование. Выполнение этой работы предполагает знания теории системы массового обслуживания (СМО) и специализированного языка имитационного моделирования GPSS или AnyLogic.

1. Определение площади геометрической фигуры

Дана сеточная функция

Точки	1	2	3
x	0	3.6	4,0
y	2,0	0,8	0

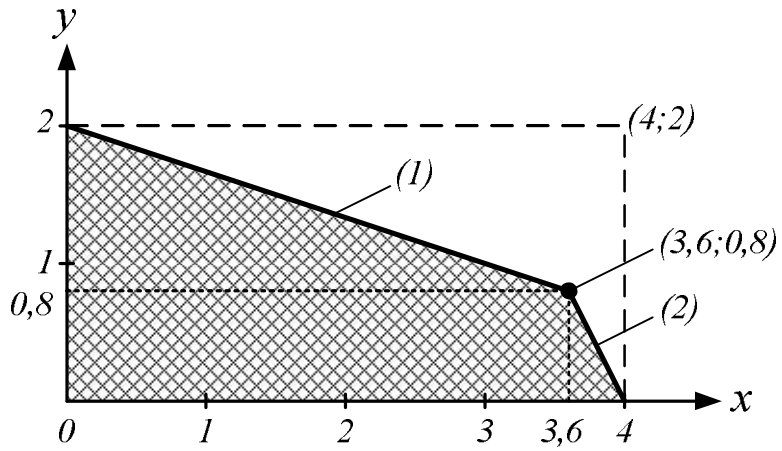
Необходимо найти площадь под графиком этой функцией, полагая, что координаты промежуточных точек определяются на основе линейной зависимости.

Использую, полином Лагранжа первой степени построим линейную функцию, между первой и второй точками.

$$y_1 = 2\frac{x-3,6}{0-3,6} + 0,8\frac{x-0}{3,6-0} = -0,33x + 2; \tag{1.1}$$

$$y_2 = 0,8\frac{x-4}{3,6-4} + 0\frac{x-3,6}{4-3,6} = -2x + 8. \tag{1.2}$$

Представим графики функций y_1 и y_2.



Из графика и из сеточной функции видно, что значения y и x изменяются соответственно от 0 до 2 и от 0 до 4. Поэтому для оценки площади под рассматриваемой ломанной кривой заключим эту площадь в прямоугольник со сторонами 4 и 2. Определение площади обозначенной фигуры основано на предположении, что все точки входящие в прямоугольник равновероятны. Предположим, была произведена случайная выборка, которая показала, что из общего количества точек n в площадь фигуры попало m точек. Таким образом, с некоторой погрешностью площадь фигуры можно вычислить по формуле

$$\textit{площадь Фигуры} = m/n \cdot (\textit{площадь Прямоугольника}) = m/n \cdot 8.$$

При увеличении числа испытаний n эта площадь будет стремиться к истинному значению.

Координаты y и x точек прямоугольника можно представить как равномерно распределённые случайные величины с плотностью вероятностей

$$f(x) = 1/4 \quad 4 \le x \ge 0; \tag{1.3}$$

$$f(y) = 1/2 \quad 2 \le y \ge 0. \tag{1.4}$$

Обе функции равны нулю вне указанных интервалов. Для получения пары случайных чисел y и x запишем выражения $y = 2R_1$ и $x = 4R_2$, где R_1 и R_2 случайные числа из интервала (0,1). Истинные случайные числа могут быть сгенерированы с использованием электронных приборов. Однако, так как имитационная модель реализуется на компьютере, такая генерация замедлит работу программы. Поэтому в имитационных моделях используются генераторы случайных чисел основанных на арифметических вычислениях.

Такие числа не являются случайными. Их называют псевдослучайными, так как они могут быть заранее определены. Так, например, для получения псевдослучайных чисел может быть использована формула из теории целых чисел $u_n = (b \cdot u_{n+1} + c) \bmod(m); R_n = \dfrac{u_n}{m}$, где u_0, b, c, m наперёд заданные параметры. Так, например, при $u_0 = 11, b = 9, c = 5, m = 12$, псевдослучайные числа R_1, R_2, R_3, R_4, R_5 будут соответственно равны

$$u_1 = (9 \cdot 11 + 5) \bmod(12) = 8; R_1 = \frac{8}{12} = 0,6667,$$

$$u_2 = (9 \cdot 8 + 5) \bmod(12) = 5; R_2 = \frac{5}{12} = 0,4167,$$

$$u_3 = (9 \cdot 5 + 5) \bmod(12) = 2; R_3 = \frac{2}{12} = 0,1666,$$

$$u_4 = (9 \cdot 2 + 5) \bmod(12) = 11; R_4 = \frac{11}{12} = 0,9166,$$

$$u_5 = (9 \cdot 11 + 5) \bmod(12) = 8; R_5 = \frac{8}{12} = 0,6667.$$

На основе анализа результатов расчёта можно сделать вывод, что эта формула не может быть использована для генерации координат точек в прямоугольнике, так как числа начинают повторяться с периодом четыре. Такой период не позволяет обеспечить необходимую случайную выборку координат точек прямоугольника. Поэтому перед использованием выбранного генератора необходимо проводить статистические проверки по выявлению качества генератора, которое, в данном случае, зависит от значения параметров u_0, b, c.

Между тем следует заметить, в языках высокого уровня присутствуют функции для получения псевдослучайных чисел с большим периодом повторения. При

этом разработчик программы должен обеспечить автоматический контроль за началом повторения псевдослучайных чисел, чтобы избежать получения неправильной имитации.

После выбора подходящего генератора псевдослучайных чисел выполняется имитация определения площади фигуры. Суть имитации заключается в следующем.

1. С помощью генератора определяется два числа R_1 и R_2. Допустим они соответственно равны 0,6675 и 0,2345. Тогда координаты точки будут равны:

$$y = 2 \cdot 0,6675 = 1,335 \text{ и } x = 4 \cdot 0,2345 = 0,938.$$

2. Запишем (1) и (2) уравнение в таком виде

$$y + 0,33x = 2, \quad (1.5)$$

$$y + 2x = 8. \quad (1.6)$$

и подставим туда вычисленные значения x и y:

$$1,335 + 0,333 \cdot 0,938 = 1,647,$$

$$1,335 + 2 \cdot 0,938 = 3,211.$$

Таким образом, так как обе левые части выражений (1.5) и (1.6) оказались меньше правых, делаем заключение, точка попала под ломанную кривую. В этом случае $n = 1$ и $m = 1$. Если хотя бы в одном из выражений (1.5) и (1.6) левая часть окажется больше правой, делаем заключение, что точка не попала под ломанную кривую и присваиваем $n = 2$ и $m = 1$. В противном случае присваиваем $n = 2$ и $m = 2$.

Описанный процесс вычислений повторялся $n = 5000$ раз при 10 прогонах. Он показал, что погрешность вычислений площади уменьшается с увеличением числа генерируемых точек и числа прогонов.

Результаты эксперимента представлены в таблице 1.

Таблица 1

Номер прогона	1	2	3	4	5	6	7	8	9	10
m	3248	3253	3256	3246	3249	3255	3247	3255	3239	3259
F	5,69	5,52	5,50	5,55	6,00	5,52	5,90	5,57	5,58	5,53
S^2	1,87 E-05	1,35 E-05	7,19 E-05	5,66 E-05	7,4 E-06	4,73 E-05	3,5 E-04	4,73 E-05	3,5 E-04	1,76 E-04

Ввиду того, что оценки площади имеют разброс необходимо результаты экспериментов выразить в виде доверительного интервала точного значения.

В данном примере точная площадь

$$F = (2 + 0,8) \cdot 3,6/2 + (0 + 0,8) \cdot 0,4/2 = 5,2.$$

Доверительный интервал для F можно записать в виде

$$\overline{F} - \frac{s}{\sqrt{N}} t_{\alpha/2,N-1} \geq F \leq \overline{F} + \frac{s}{\sqrt{N}} t_{\alpha/2,N-1}, \qquad (1.7)$$

где $t_{\alpha/2,\,N-1}$ – коэффициент Стьюдента;

$\overline{F}$ – среднее значение площади;

N – количество прогонов;

α – коэффициент значимости;

s^2 – дисперсия.

Согласно данным таблицы 1 определяем $\overline{F} = 5,201$, $s^2 = 0,00009$. При десяти прогонах и $\alpha = 0,05$, $t_{\alpha/2,\,N-1} = 2,23$ доверительный интервал $5,19 \geq F \leq 5,208$.

На точность вычислений большое влияние оказывает величина площади прямоугольника, в который вписана геометрическая фигура, площадь которой надо определить. В представленном примере площадь прямоугольника равна $4 \times 2 = 8$. Если принять стороны прямоугольника 6 и 8, то при объёме вычислений представленных выше результаты вычислений имеют вид.

Таблица 2

Номер прогона	1	2	3	4	5	6	7	8	9	10
m	506	553	503	527	558	534	494	551	657	649
F	4,86	5,31	4,83	5,01	5.36	5,13	4,74	5,29	5,34	5,27
S^2	0,066	0,038	0,081	0,01	0,059	0,0002	0,138	0,03	0,054	0,024

Согласно данным таблицы 2 определяем $\overline{F} = 5,11$, $s^2 = 0,056$. При десяти прогонах и $\alpha = 0,05$, $t_{\alpha/2,\,N-1} = 2,23$ доверительный интервал $5,19 \geq F \leq 5,208$.

Сопоставление результатов расчёта позволяет сказать, что при одинаковом объёме имитации во втором случае доверительный интервал в 19 раз больше, чем в первом случае. Это означает, что в первом случае расчёта результат расчёта ближе к истинному значению площади геометрической фигуры.

Оценивая с практической точки зрения полученный доверительный интервал, можно принять решение о необходимости увеличении или уменьшении объёма выборки и количестве прогонов.

2. Модель системы массового обслуживания

В этом разделе рассматривается дискретно-событийная имитационная модель. Эта модель описывает поведение системы, изменяющееся в заданные моменты времени. Типичным примером являются системы массового обслуживания (СМО), в результате функционирования которых образуются очереди из клиентов. Эти системы представляют собой совокупность двух объектов: сервис и клиент. Клиент поступает на обслуживание в сервис, а сервис его обслуживает.

В рассматриваемой модели происходит только два события: приход и уход клиента, так все статистические характеристики системы изменяются только в эти моменты времени. В другие моменты времени ничего в системе не происходит.

Таким образом, логику имитационной модели можно представить, рассматривая два события: приход и уход клиента.

В момент прихода клиента в модели необходимо выполнить следующие операции:

1. Сгенерировать и сохранить время прибытия клиента. Оно равняется текущее время плюс промежуток времени между приходами двух последних клиентов.

2. Если сервис свободен:

2.1. Начать обслуживание, поступившего клиента, изменить состояние сервиса на рабочее.

2.2. Сгенерировать и сохранить время ухода клиента.

3. Если сервис занят, поставить его в очередь и увеличить её длину на 1.

В момент ухода клиента в модели необходимо выполнить следующие операции:

1. Объявить сервис свободным, если очередь пуста. Пересчитать статистические характеристики системы.

2. Если очередь не является пустой:

2.1. Начать обслуживание клиента из очереди. Уменьшить её на 1 и пересчитать статистические характеристики системы.

2.2. Сгенерировать и сохранить время ухода клиента. Оно равно текущее время моделирования плюс время обслуживания клиента.

2.1 Механика моделирования СМО

Для иллюстрации процесса дискретно-событийного имитационного моделирования рассмотрим следующий пример.

В обслуживающую организацию поступают клиенты по экспоненциальному закону в среднем 6 человек в час. Обслуживание клиентов осуществляет один человек, который выполняет два вида работ. Время обслуживания клиента

является дискретным. Первый вид работ занимает в среднем 15 минут, второй вид работ выполняется в среднем за 10 минут. Один из четырёх клиентов заказывает первый вид работы.

Необходимо построить имитационную модель при времени её работы 50 минут и определить:

1. Среднюю занятость сервиса.
2. Среднюю длину очереди.
3. Среднее время ожидания клиентов в очереди.

Обозначим p и q как случайные величины времени прибытия и обслуживания клиентов, которые можно определить так

$$p = -10Ln(R) \text{ мин} \qquad 0 \le R \le 1 \qquad (2.1)$$

$$q = 15 \text{ мин} \qquad 0 \le R \le 0,25 \qquad \text{первая работа} \qquad (2.2)$$

$$q = 10 \text{ мин} \qquad 0 \le R \le 0,75 \qquad \text{вторая работа,} \qquad (2.3)$$

где R – равномерно распределённая величина в диапазоне 0…1.

Обозначим T – время наблюдения за системой, которое в начале положим равным 0, сервис объявляем свободным. С помощью датчика случайных чисел определяем значение R. Затем согласно (2.1), (2.2), (2.3) определяем случайное время прихода и обслуживания клиентов.

Для иллюстрации динамики изменения этих величин и образного восприятия сути дискретно-событийного моделирования время прихода и обслуживания клиентов изобразим на временной оси T на рисунке 2.1, рисунке 2.2. В этой модели (рис. 2.1) первый клиент приходит в систему через 0 минут, второй клиент через 28,3 минуты, третий клиент через 7,34 минуты, четвёртый клиент через 4,88 минуты и пятый клиент через 17,25 минуты. На рисунке 2.2 представлено время обслуживания клиентов, первый клиент обслуживается с 0 до 15 минут, второй клиент обслуживается с 28,3 до 43,3 минут, третий клиент обслуживается с 43,3 до 58,3 минуты. После их обслуживания они покидают систему, а их место занимают клиенты из очереди, если она не пуста. Так на рисунке 2,2 видно, что система находится в простое 15 до 28,3 минуты, так как первый клиент покидает систему в 15 минут, второй клиент поступает на обслуживание в 28,3 минуты.

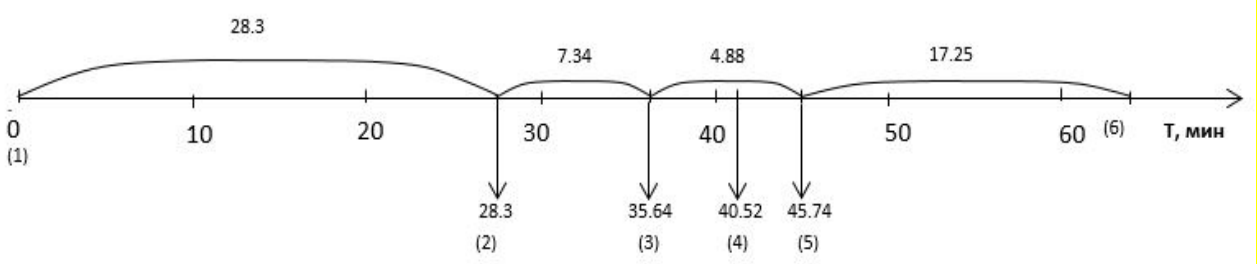


Рисунок 2.1. Приход клиентов
(1), (2), (3), (4), (5) – номер клиента

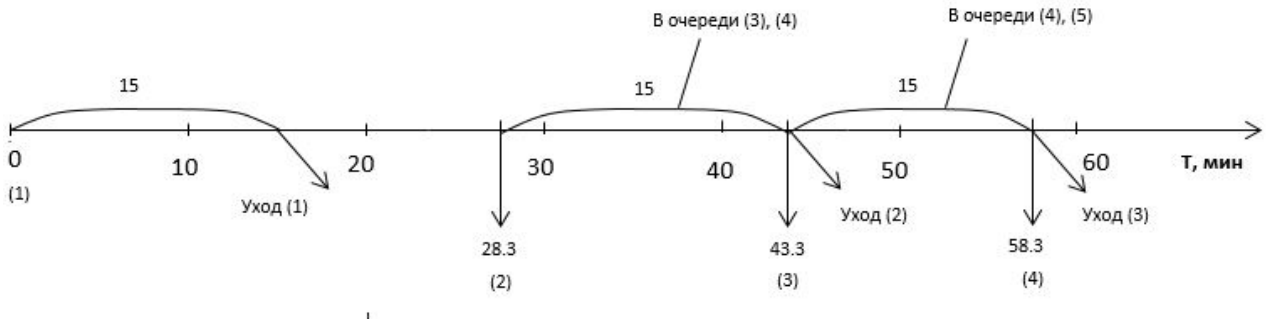


Рисунок 2.2. Обслуживание клиентов
(1), (2), (3), (4), (5) – номер клиента

Рассматривая последовательно приход и уход клиентов можно построить (рис. 2.3–2.4) графики количества клиентов в очереди и занятость системы в зависимости от модельного времени.

На основе сопоставления этих рисунков построим зависимость (рис. 2.3) и (рис. 2.4) длины очереди и занятости сервиса от времени имитации.

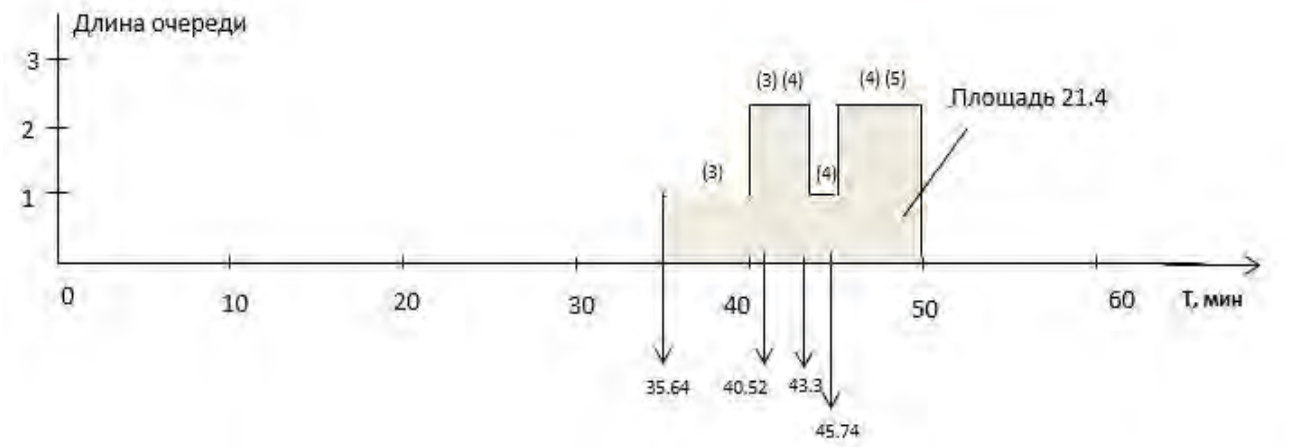


Рисунок 2.3. Длина очереди
(1), (2), (3), (4), (5) – номер клиента

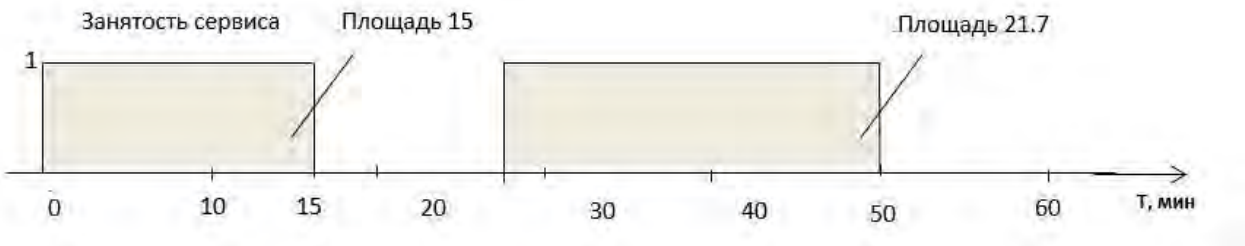


Рисунок 2.4. Занятость сервиса

И в заключении с помощью (рис. 2.3) и (рис. 2.4) определяем искомые показатели работы моделируемой системы:

– средняя длина очереди 21,4/50 = 0,43 чел.;

– средняя занятость сервиса (15 + 21,7)/50 = 0,73;

– среднее время ожидания клиентов, которые были в очереди 21,4/3 = 7,3 мин;

– среднее время ожидания всех клиентов, которые были в системе обслуживания в очереди 21,4/5 = 4,28 мин.

Как видно из представленного примера, моделирование системы СМО связано с выполнением большого количества вычислений и поэтому предполагает написание приложения для реализации на компьютере. При этом очевидно, что написание и сопровождение такого рода программных продуктов требует значительных затрат.

В настоящее время имеются специальные программные продукты решающие проблему по автоматизации создания приложений для имитационного моделирования систем. В данном методическом указании представлены примеры использования языка программирования GPSS.

2.2 Краткое описание языка GPSS

Язык GPSS используется при решении задач, в которых рассматриваются процессы с дискретными событиями. Такие процессы наблюдаются, например, при функционировании СМО.

В основе языка заложено понятие транзакта, который представляет собой формальный объект, перемещающийся по системе. Каждый транзакт обладает совокупностью параметров, принимающие различные значения в зависимости от места нахождения его в системе.

Язык GPSS – язык интерпретируемого типа. При выполнении программы происходит пошаговое выполнение операторов, называемые блоками. Комплекс программ реализующих функционирование блоков называется симулятором.

В GPSS присутствуют следующие объекты: устройство, память, очередь, ячейки, таблица.

Устройство имитирует единицу оборудования, который обрабатывает один транзакт.

Память имитирует единицу оборудования, в котором может обрабатываться несколько транзактов.

Очередь имитирует задержку транзакта перед устройством или памятью.

Таблица обеспечивает накопление статистики о параметре модели.

Ячейки используются для накопления и хранения входных и выходных параметров.

Для написания программы на GPSS необходимо знание формализмов языка и предметной логики моделируемой системы. Для иллюстрации работы

блоков языка рассмотрим три примера моделирования с пошаговым комментированием программного кода и результатов решения.

2.3 Работа кассы с двумя кассирами

Необходимо построить имитационную модель работы кассы по продаже билетов при следующих параметрах работы. Посетители приходят в кассу через 10…30 секунд. Знакомятся с помещением 0…15 секунд и занимают очередь. В кассе работают два кассира. Каждый кассир затрачивает на обслуживание посетителя одинаковое время около 15…25 секунд. Касса работает 5 часов.

Код программы имеет следующий вид:

Код программы	Комментарий
`10 GENERATE 20,10`	Приход клиентов, генерирование транзактов (клиентов) в интервале
`20 TRANSFER .5,,PROM`	Выбор кассира клиентом с вероятностью 50 %
`30 ADVANCE 15,15`	Знакомство клиента с помещением в перёд
`40 QUEUE OCH`	Транзакт (клиент) задерживается в очереди OCH. Поставить клиента в очередь
`50 SEIZE KASS`	Занятие устройства (кассы) KASS, если оно свободно
`60 DEPART OCH`	Выход транзакта(клиента) из очереди OCH.
`70 ADVANCE 20,5`	Задержка транзакта (клиента) в устройстве (кассе) KASS в интервал времени 15…25 секунд. Покупка билета
`80 RELEASE KASS`	Освобождение устройства (кассы) KASS. Обслуживание клиента закончено
`90 TERMINATE`	Уничтожение транзакта. Клиент покидает устройство (кассу)
`100 PROM QUEUE OCH1`	Поставить транзакта (клиента) в очередь OCH1
`110 SEIZE KASS1`	Занятие второго устройства (кассы) KASS1, если оно свободно. Обращение ко второму кассиру
`120 DEPART OCH1`	Выход из очереди OCH1
`130 ADVANCE 20,5`	Задержка транзакта (клиента) в устройстве (кассе) KASS1 в интервал времени 15…25 секунд. Покупка билета
`140 RELEASE KASS`	Освобождение устройства (кассы) KASS1. Обслуживание клиента закончено
`150 TERMINATE`	Уничтожение транзакта. Клиент покидает устройство (кассу)
`160 GENERATE 18000`	Время работы устройства (кассы)в секундах
`170 TERMINATE 1`	Закрытие кассы. 1 – это уничтожение одного цикла работы кассы
`START 1`	Старт программы 1 – один цикл работы устройства (кассы)

Если вместо 1 поставить, например 3, то это означает моделирование работы кассы на протяжении 54000 секунд.

После запуска программы на выполнение получаем следующий результат.

```
START TIME          END TIME   BLOCKS  FACILITIES  STORAGES
   0.000           18000.000    17        2          0
```

START TIME – начало моделирования.

END TIME – конец моделирования.

BLOCKS – количество блоков в коде.

FACILITIES – количество устройств.

STORAGES – количество оборудований, обрабатывающих количество транзактов больше одного

NAME	VALUE
KASS	10003.000
KASS1	10001.000
OCH	10002.000
OCH1	10000.000
PROM	10.000

NAME – Названия устройств, очередей и начала следующей очереди.

VALUE – Значение имён.

LABEL	LOC	BLOCK TYPE	ENTRY COUNT	CURRENT COUNT	RETRY
	1	GENERATE	909	0	0
	2	TRANSFER	909	0	0
	3	ADVANCE	443	1	0
	4	QUEUE	442	0	0
	5	SEIZE	442	0	0
	6	DEPART	442	0	0
	7	ADVANCE	442	1	0
	8	RELEASE	441	0	0
	9	TERMINATE	441	0	0
PROM	10	QUEUE	466	0	0
	11	SEIZE	466	0	0
	12	DEPART	466	0	0
	13	ADVANCE	466	0	0
	14	RELEASE	466	0	0
	15	TERMINATE	466	0	0
	16	GENERATE	1	0	0
	17	TERMINATE	1	0	0

LABEL – присвоенное имя блока.

LOC – номер блока.

BLOCK TYPE – названия блока.

ENTRY COUNT – количество транзактов, побывавших в блоке.

CURRENT COUNT – количество транзактов на момент окончания моделирования.

RETRY – количество транзактов ожидающих выполнения специальных условий.

```
FACILITY        ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 KASS1            466    0.517      19.952  1        0    0     0     0      0
 KASS             442    0.492      20.053  1      909    0     0     0      0
```

FACILITY – имя устройства, присвоенное в программе.

ENTRIES – количество раз было занято устройство.

UTIL. – средняя загрузка устройства.

AVE. TIME – среднее время транзакта в устройстве.

AVAIL. – доступность устройства на момент окончания моделирования.

OWNER – номер транзакта в устройстве.

PEND – количество транзактов в очереди при блоке PREEMPT.

INTER – количество транзактов в очереди после прерывания.

RETRY – количество транзактов в очереди в зависимости от состояния устройства.

DELAY – количество транзактов в очереди при блоке SEIZE и PREEMPT.

```
QUEUE              MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 OCH1                2    0    466    339     0.048      1.855      6.808   0
 OCH                 3    0    442    287     0.109      4.433     12.642   0
```

QUEUE – название очереди.

MAX – максимальная длина очереди.

CONT – длина очереди на момент окончания моделирования.

ENTRY – общее количество входов в очередь.

ENTRY(0) – количество нулевых входов.

AVE.CONT – средняя длина очереди.

AVE.TIME – среднее время нахождения в очереди.

Изменяя входные данные, можно получить различные статистические характеристики моделируемой системы и определить нужный режим работы предприятия. На практике присутствуют предприятия с различными режимами и графиками работы. Вместе с тем логика функционирования предприятий одинакова. При этом при моделировании системы необходимо учитывать уникальные особенности конкретного предприятия. С помощью языка GPSS эти особенности можно смоделировать, используя различные сочетания блоков. В следующем примере представим модель предприятия с двумя видами работ и одним работником (в предыдущем примере был один вид работ и два работника).

2.4 Работа парикмахерской с одним мастером

Необходимо построить имитационную модель работы парикмахерской по обслуживанию клиентов при следующих параметрах работы. В обслуживающую

организацию поступают клиенты по экспоненциальному закону в среднем 2 человека в час (математическое ожидание 30 минут). Обслуживание клиентов осуществляет один человек, который выполняет два вида работ. Время обслуживания клиента является дискретным. Первый вид работы занимает в среднем 20 минут, второй вид работы выполняется в среднем за 30 минут. Один из четырёх клиентов заказывает первый вид работы.

Необходимо построить имитационную модель при времени её работы 60 минут и определить статистические характеристики моделируемой системы.

Код программы имеет следующий вид:

Код программы	Комментарий
`10 GENERATE (EXPONENTIAL(1,0,30))`	Приход клиентов. Генерирование времени прихода по экспоненциальному закону.
`20 ADVANCE 5,0`	Знакомство с парикмахерской на интервале времени 0…5 минут
`30 QUEUE OCH`	Встать в очередь
`40 SEIZE MAS`	Обращение к мастеру. Занятие устройства MAS если оно свободно
`50 DEPART OCH`	Выход из очереди
`60 TRANSFER .75,,prom`	Выбор с вероятностью 0,75 второй вид работ
`70 ADVANCE 20,0`	Выполнение первого вида работ интервал времени 0…20 минут
`80 RELEASE MAS`	Освобождение устройства (мастера) MAS
`90 TERMINATE`	Уничтожение транзакта
`100 PROM ADVANCE 30,0`	Выполнение второго вида работ в интервал времени 0…30 минут
`110 RELEASE MAS`	Освобождение устройства (мастера) MAS
`120 TERMINATE`	Уничтожение транзакта
`130 GENERATE 480`	Время работы парикмахерской в минутах
`140 TERMINATE 1`	Закрытие парикмахерской
`START 1`	Старт программы

После запуска программы на выполнение получаем следующий результат.

```
     START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
           0.000            480.000   14        1          0

     NAME            VALUE
     KASS          10001.000
     OCH           10000.000
     PROM             10.000

LABEL          LOC  BLOCK TYPE     ENTRY COUNT CURRENT  COUNT   RETRY
                    1    GENERATE            15            0       0
                    2    ADVANCE             15            0       0
                    3    QUEUE               15            0       0
                    4    SEIZE               15            0       0
                    5    DEPART              15            0       0
```

```
                    6    TRANSFER            15             0       0
                    7    ADVANCE              3             0       0
                    8    RELEASE              3             0       0
                    9    TERMINATE            3             0       0
PROM                10    ADVANCE             12             1       0
                    11    RELEASE             11             0       0
                    12    TERMINATE           11             0       0
                    13    GENERATE             1             0       0
                    14    TERMINATE            1             0       0

FACILITY   ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
KASS         15    0.821      26.257   1      16    0    0     0      0

QUEUE        MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
OCH           3    0     15     4      0.781     25.003     34.095    0
```

В рассмотренном примере для моделирования нескольких видов работ используется TRANSFER, который моделирует выполнение двух работ на сервисе.

В следующем примере рассмотрим работу предприятия, в котором возможен отказ клиента от обслуживания.

2.5 Работа парикмахерской с несколькими мастерами при отказе от обслуживания

Необходимо построить имитационную модель работы парикмахерской по обслуживанию клиентов при следующих параметрах работы. В обслуживающую организацию поступают клиенты по экспоненциальному закону в среднем 2 человека в час (математическое ожидание 30 минут). Обслуживание клиентов осуществляют четыре мастера. Время обслуживания клиента является дискретным и составляет от 14 до 19 минут. Клиент отказывается от обслуживания, если в очереди находится более двух человек.

Необходимо построить имитационную модель при времени её работы 480 минут и определить статистические характеристики системы.

Код программы имеет следующий вид:

```
10 KRES   STORAGE       4
20        GENERATETYT
30        TEST L        Q1,3,OTKAZ
40        QUEUE         1
50        ENTER         KRES
60        DEPART        1
70        ADVANCE       19,5
80        LEAVE         KRES
90        TERMINATE
```

```
      100 OTKAZ TERMINATE
      110       GENERATE      480
      120       TERMiNATE     1
      START 1
```

Представленная программа отличается от предыдущих следующим:

– используется объект память KRES, которая описывается картой STORAGE. Карта не является объектом, так как транзакты в неё не входят. Она просто описывает объекты. В данном примере в памяти находятся 4 мастера по обслуживанию клиентов;

– используется блок ENTER, в котором проверяется достаточно ли места для вхождения транзакта в память KRES;

– используется блок LEAVE, в котором освобождается место памяти KRES;

– используется блок TEST, в котором проверяется условие отказа от обслуживания. При этом символом L обозначается условие меньше, символом Q1 обозначается текущая длина очереди номер 1, цифра 3 обозначает критическую длину очереди, символ OTKAZ предназначен для удаления транзакта из системы.

После запуска программы на выполнение получаем следующий результат.

```
START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
            0.000            480.000   11        0          1

      NAME      VALUE
      KRES    10000.000
      OTKAZ     9.000

LABEL          LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                1    GENERATE            18             0       0
                2    TEST                18             0       0
                3    QUEUE               18             0       0
                4    ENTER               18             0       0
                5    DEPART              18             0       0
                6    ADVANCE             18             0       0
                7    LEAVE               18             0       0
                8    TERMINATE           18             0       0
OTKAZ           9    TERMINATE            0             0       0
               10    GENERATE             1             0       0
               11    TERMINATE            1             0       0

QUEUE     MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 1         1    0     18     18     0.000      0.000      0.000   0

STORAGE      CAP. REM. MIN. MAX.  ENTRIES AVL.  AVE.C. UTIL. RETRY DELAY
 KRES         4    4   0     3       18    1    0.720  0.180    0    0
```

STORAGE – имя объекта память.

CAP. – ёмкость памяти.

REM. – количество свободных ячеек памяти на момент окончания моделирования.

MIN. MAX. – минимальное и максимальное количество занятых ячеек в течении моделирования.

ENTRIES – общее количество входов.

AVL. – Занятость памяти на момент окончания моделирования.

AVE.C – среднее количество занятых ячеек памяти.

UTIL. – средняя загрузка памяти в течении моделирования.

Представленные примеры иллюстрируют принципиальные моменты решения задач по имитационному моделированию. Для более глубокого изучения и решения конкретных задач предлагается использовать уже имеющиеся решения [1–4].

2.6 Имитационная модель с применением инструментальных средств AnyLogic7

При имитации дискретных процессов в качестве инструментального средства получила широкое распространение система общецелевого назначения GPSS World, использование которой было продемонстрировано на примерах, представленных выше.

Наряду с ней применяется система моделирования AnyLogic7. Она разработана компанией «The AnyLogic Company» на основе современных концепций в области информационных технологий и результатов исследований в теории гибридных систем и объектно-ориентированного моделирования. Это комплексный инструмент, охватывающий в одной модели основные в настоящее время направления моделирования: дискретно-событийное, системной динамики, агентное. Многоподходность AnyLogic7 позволяет решать задачи моделирования характерные для GPSS World.

Последнее обстоятельство является основанием для представления в настоящем методическом указании выше рассмотренных моделей на основе AnyLogic7. Кроме того обучаемые получат более широкие знания по моделированию процессов в разнородных системах, и сможет сравнивать эффективность их для решения одинаковых задач.

2.6.1 Имитационная модель парикмахерской с одним мастером при отказе от обслуживания, построенная с применением инструментальных средств AnyLogic7

2.6.1.1 Постановка задачи

Построить имитационную модель работы парикмахерской по обслуживанию клиентов при следующих параметрах работы. В обслуживающую организацию поступают клиенты по экспоненциальному закону в среднем 2 человека в час (математическое ожидание 30 минут). Обслуживание клиентов осуществляет один мастер. Время обслуживания клиента является дискретным и составляет от 14 до 19 минут. Клиент отказывается от обслуживания, если в очереди находится более двух человек.

Парикмахерская представляет собой однофазную систему массового обслуживания разомкнутого типа с ограниченной входной ёмкостью, то есть с отказами, и абсолютной надёжностью (рис. 2.6.1).

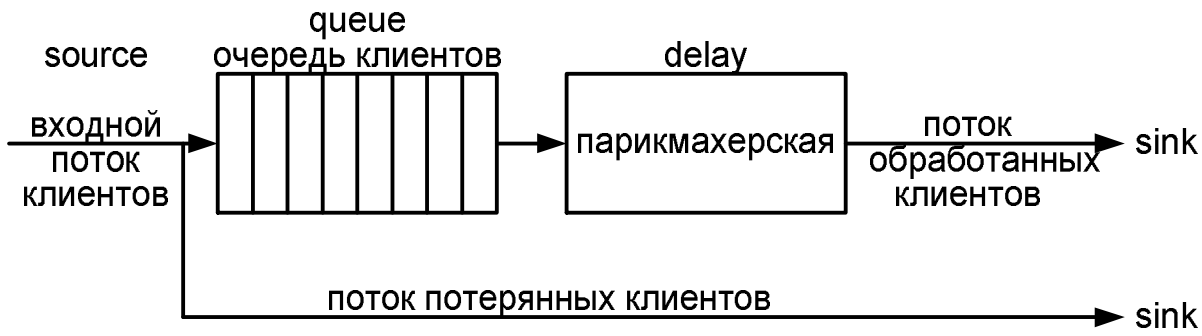


Рисунок 2.6.1. Парикмахерская как система массового обслуживания

На рисунке 2.6.1 приведены объекты AnyLogic, которые будут использованы для создания диаграммы процесса. Приступим к созданию диаграммы процесса.

Необходимо построить имитационную модель при времени её работы 480 минут и определить статистические характеристики системы. На основе модели определить математического ожидания времени и вероятности обслуживания клиента.

2.6.1.2 Создание диаграммы процесса

1. Запустите AnyLogic7.

2. Выполните **Файл/Создать/Модель** на панели инструментов. Появится диалоговое окно **Новая модель** (рис. 2.6.2).

3. Задайте имя новой модели. В поле **Имя модели** введите Парикмахер.

4. Выберите каталог в поле **Местоположение**, в котором будут сохранены файлы модели.

5. Щёлкните кнопку **Готово**. Откроется пользовательский интерфейс (рис. 2.6.3).

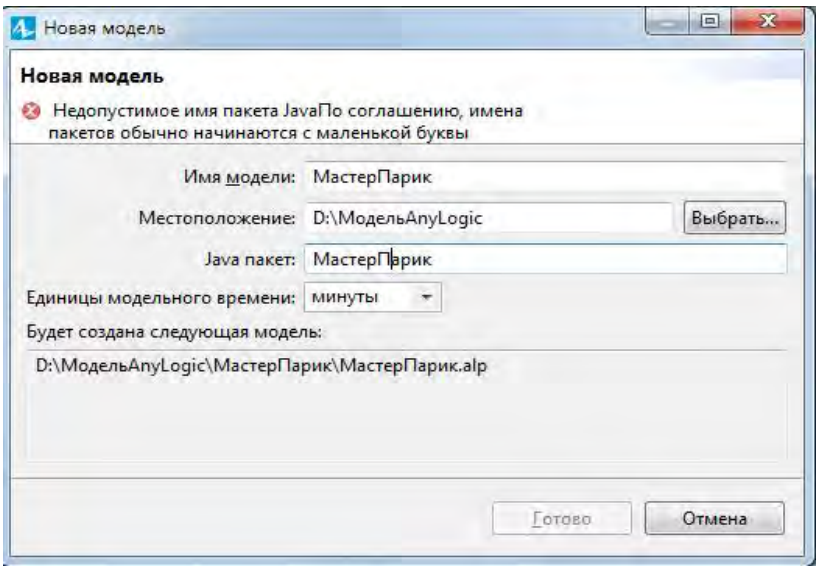


Рисунок 2.6.2. Диалоговое окно **Новая модель**

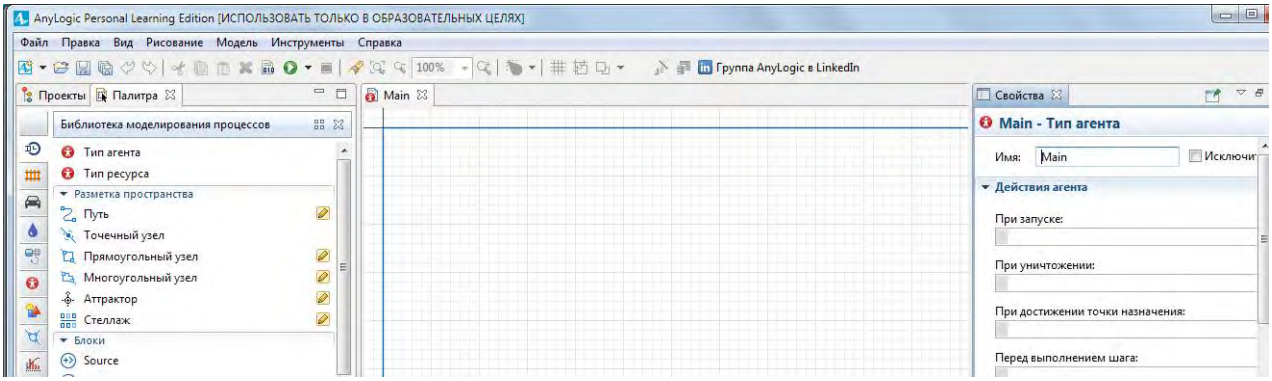


Рисунок 2.6.3. Пользовательский интерфейс

6. В левой части рабочей области находятся панель **Проекты** и панель **Палитра**. Панель **Проект** обеспечивает навигацию по элементам моделей, открытых в текущий момент времени. Модель организована иерархически. Она отображается в виде дерева.

7. Панель **Палитра** содержит разделённые по категориям элементы, которые могут быть добавлены на графическую диаграмму типа агентов или эксперимента. На рисунке 2.6.3 раскрыта палитра Библиотека моделирования процесса, на основе которой будем создавать модель работы парикмахерской.

8. В правой части рабочей области отображается панель **Свойства**. Панель **Свойства** используется для просмотра и изменения свойств выбранного в данный момент элемента модели.

9. В центре рабочей области размещён **графический редактор** диаграммы агента **Main.** В рабочей области располагаются объекты создаваемой модели.

10. Создадим диаграмму процесса. Для этого в Палитре выделите **Библиотеку моделирования процессов**. Из неё перетащите с помощью мышки объекты на диаграмму в графический редактор и соедините, как показано на рисунке 2.6.4.

Рисунок 2.6.4. Диаграмма системы массового обслуживания

11. Объект **source** генерирует заявки определённого типа. Обычно он используется в качестве начальной точки диаграммы процесса, формализующей поток заявок. В данном примере заявками называют клиентов парикмахерской, а объект **source** будет моделировать их поступление.

12. Объект **queue** моделирует очередь заявок ожидающих приёма к мастеру.

13. Объект **delay** задерживает заявки на заданный период времени. Он представляет в нашей модели место работы мастера.

14. Объект **sink** уничтожает поступившие заявки.

2.6.1.3 Изменение свойств блоков и запуск модели

В основе каждой дискретно-событийной модели лежит диаграмма процесса; последовательность соединённых между собой объектов (**Библиотеки моделирование процессов**), задающих последовательность операций, которые будут производиться над проходящими по диаграмме процесса клиентами.

Диаграмма процесса создаётся путём добавления объектов библиотеки из палитры на диаграмму класса активного объекта, соединения их портов и изменения значений свойств блоков в соответствии с требованиями модели. Чтобы сделать созданную диаграмму модели (рис. 2.6.4) адекватной постановке задачи, необходимо изменить некоторые свойства объектов.

2.6.1.4 Изменение свойств блоков диаграммы процесса

Свойства объекта можно изменить в панели **Свойства,** которая является контекстно-зависимой. Она отображает свойства выделенного в текущий момент элемента. Поэтому для изменения свойств элемента нужно будет предварительно щелчком мыши выделить его в графическом редакторе или в панели **Проекты**.

Первым объектом в диаграмме процесса является объект класса **source**. Объект **source** генерирует заявки определённого типа. В рассматриваем примере объект **source** будет моделировать поступление клиентов в парикмахерскую.

В нашем случае объект создаёт заявки через временной интервал, распределённый по показательному (экспоненциальному) закону со средним значением 30 мин.

Установим среднее время поступления запросов и среднее время в минутах.

1. Выделим объект **source**. В выпадающем списке **Прибывают согласно:** укажем, что запросы поступают согласно **Времени между прибытиями:** (рис. 2.6.5).

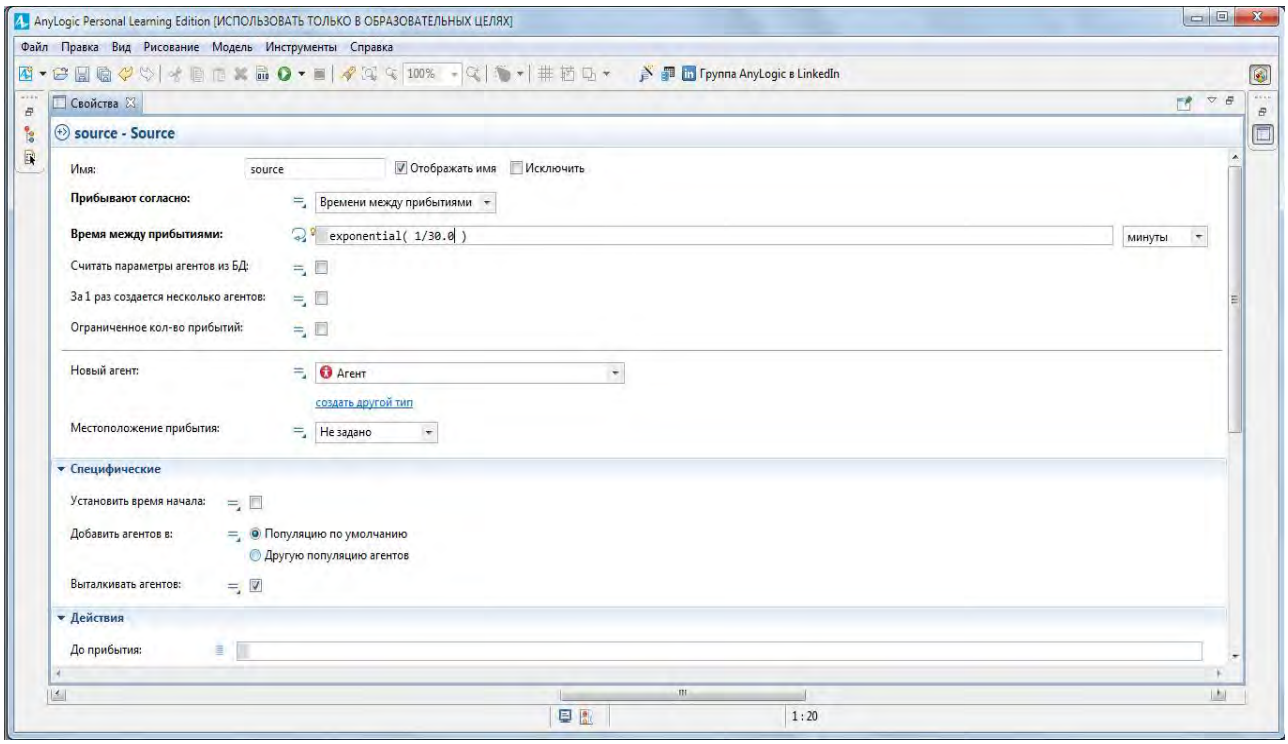


Рисунок 2.6.5. Свойства объекта **source**

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 введём 1/30.0

В языке программирования Java символ / означает целочисленное деление, т. е. если оба числа целые, то и результат будет целым. Для получения вещественного результата, необходимо, чтобы хотя бы одно из чисел было вещественным (double). Поэтому в качестве характеристики экспоненциального распределения (интенсивности поступления запросов) необходимо указать 1/30.0 или 1.0/30.

Следующий объект – **queue**. Выделим его. Он моделирует очередь заявок, ожидающих приёма объектами, следующими за данным объектом в диаграмме процесса. В нашем случае он будет, как уже отмечалось, моделировать очередь клиентов, ждущих освобождения мастера.

Изменим свойства объекта **queue** (рис. 2.6.6).

1. Зададим длину очереди. Введём в поле **Вместимость: 2**. В очереди будут находиться не более 2 клиентов. Если появляется третий клиент, то он сразу покидает парикмахерскую без обслуживания.

2. Установим флажок **Включить сбор статистики**, чтобы включить сбор статистики для этого объекта. В этом случае по ходу моделирования будет собираться статистика по очереди.

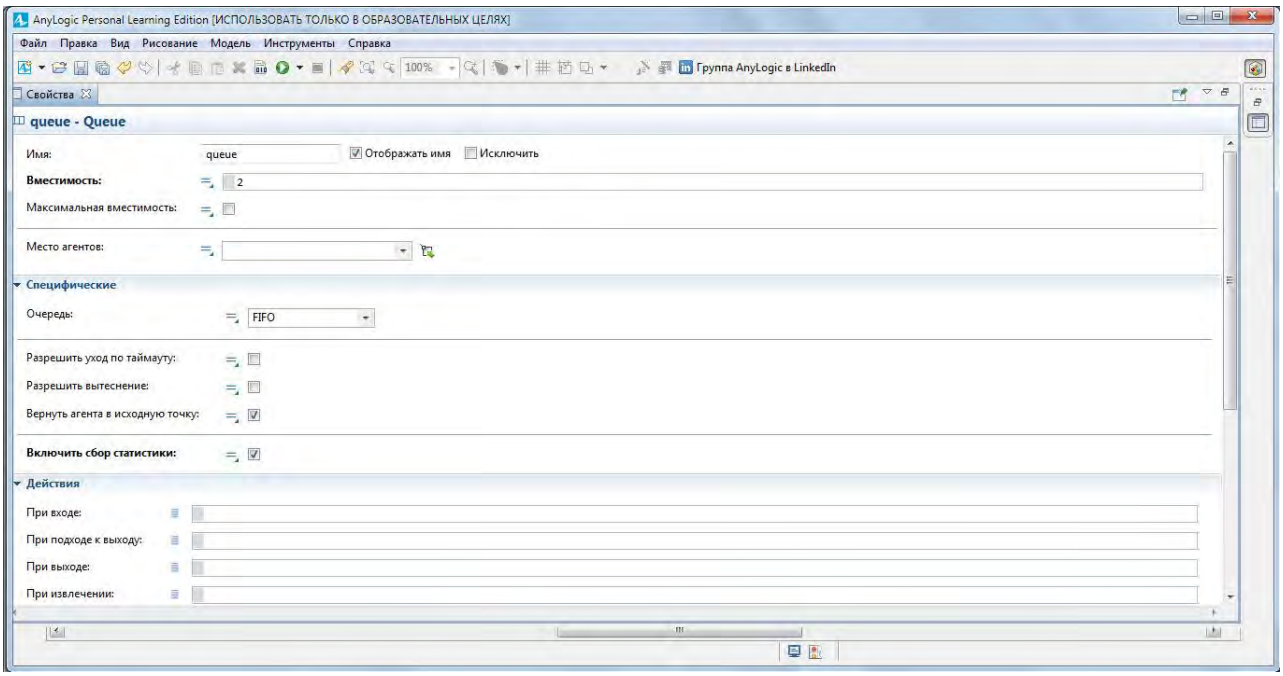


Рисунок 2.6.6. Свойства объекта **queue**

Следующим в диаграмме процесса расположен объект **delay**. Он задерживает клиентов на заданный период времени, представляя в нашей модели непосредственно парикмахерскую, в которой обслуживаются клиенты.

Изменим свойства объекта **delay** (рис. 2.6.7).

1. Обработка одного клиента занимает примерно от 14 до 19 мин. Зададим время обслуживания, распределённое по равномерному закону. Для этого введём в поле **Время задержки:** uniform_discr(14,19).

2. Установите флажок **Включить сбор статистики**.

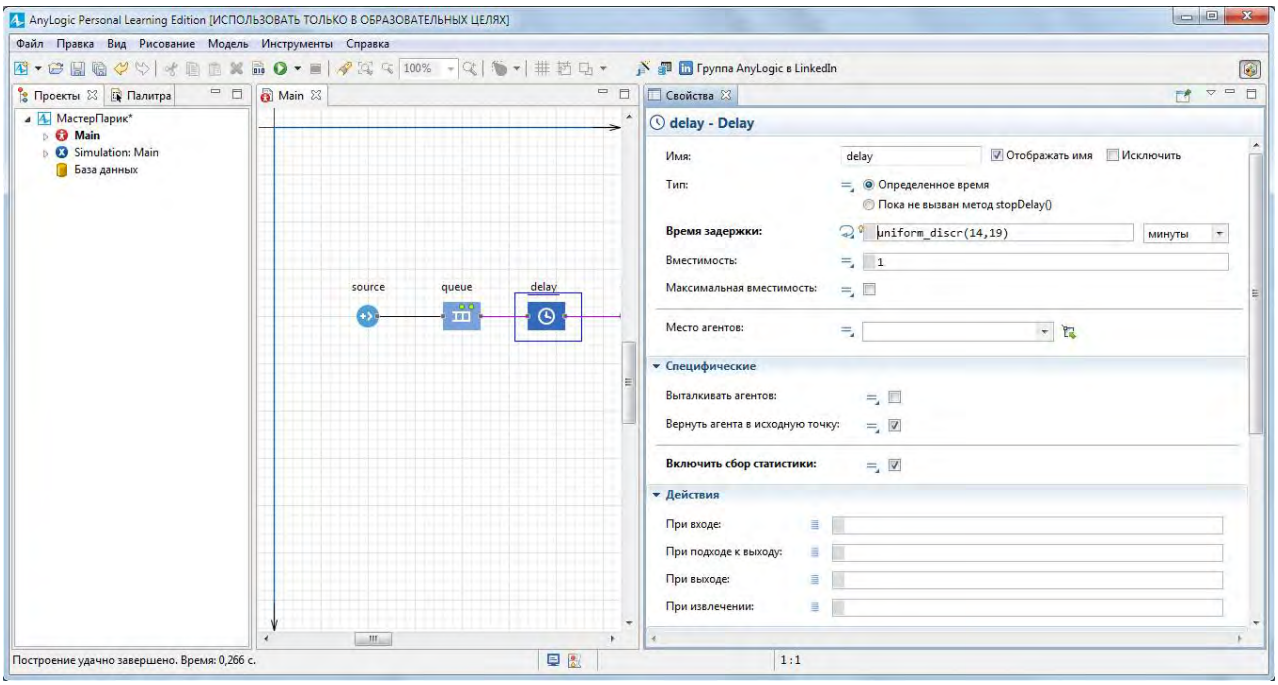


Рисунок 2.6.7 Свойства объекта **Delay**

Последним в диаграмме дискретно-событийной модели находится объект **sink**. Этот объект уничтожает поступивших клиентов. Обычно он используется в качестве конечной точки потока заявок (и диаграммы процесса соответственно). В данном случае он выводит из модели обслуженных клиентов.

2.6.1.5 Настройка запуска модели

Модель выполняется в соответствии с набором установок, заданных в элементе модели эксперимент. Можно создать несколько экспериментов с различными установками.

В панели **Проект** эксперименты отображаются в нижней части дерева модели. Один эксперимент, названный **Simulation**, создаётся по умолчанию. Это простой эксперимент, позволяющий запускать модель с заданными значениями параметров, поддерживающий режимы виртуального и реального времени, анимацию и отладку модели.

1. В панели **Проект** выделим эксперимент **Simulation:Main**.
2. Щелчком раскроем вкладку **Модельное время**.
3. Установим **Виртуальное время (максимальная скорость)** (рис. 2.6.8).

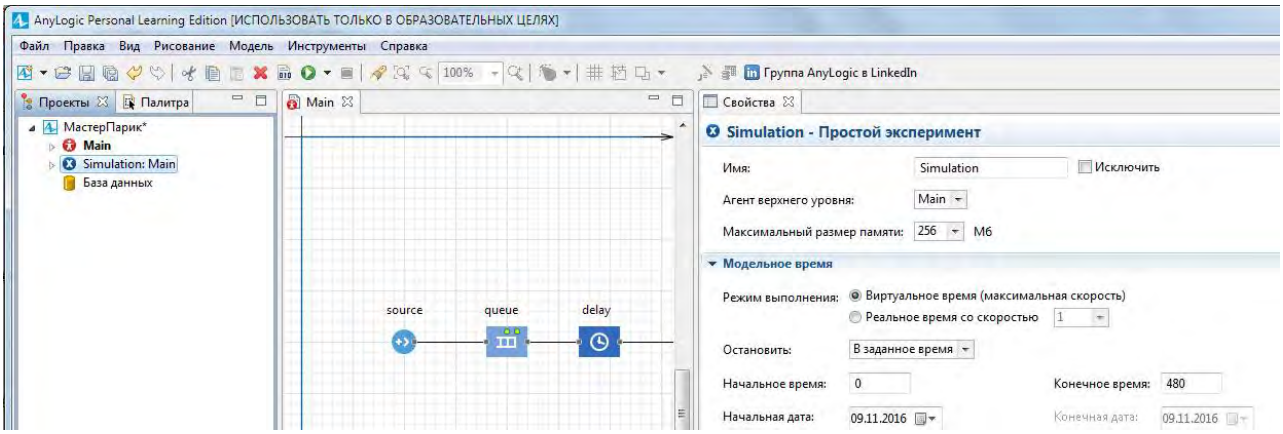


Рисунок 2.6.8. Установка свойств эксперимента

4. В поле **Остановить:** выберем из списка В заданное время.
5. В поле **Конечное время:** установим 480.
6. Раскроем вкладку **Случайность**.
7. Выберем опцию **Фиксированное начальное число (воспроизводимые прогоны)**.
8. В поле **Начальное число:** установите 9.
9. В панели **Проект**, выделим **МастерПарик** (рис. 2.6.9).
10. Из выпадающего списка **Единицы модельного времени:** выберите минуты.

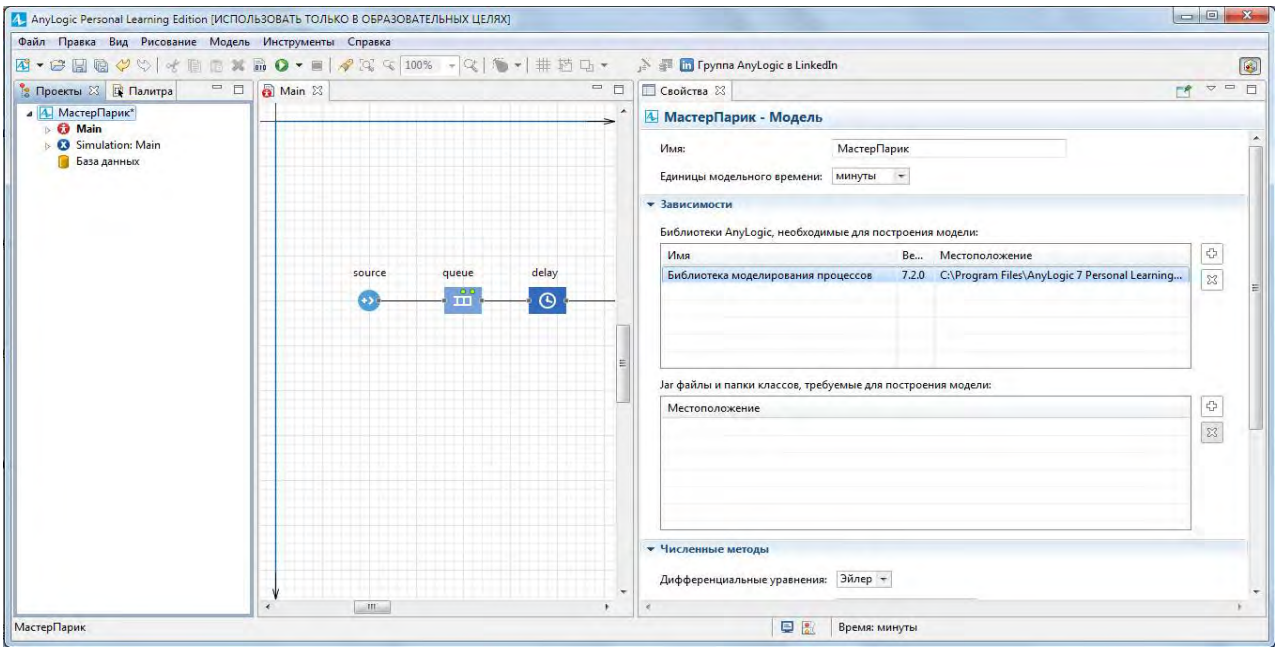


Рисунок 2.6.9. Установка модельного времени

После построения модели, запустим 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 **МастерПарик/Simulation.**

2. После запуска модели можно увидеть окно презентации этой модели (рис. 2.6.10). В нем будет отображена презентация запущенного эксперимента. AnyLogic автоматически помещает на презентацию каждого простого эксперимента заголовок и кнопку, позволяющую запустить модель и перейти на презентацию.

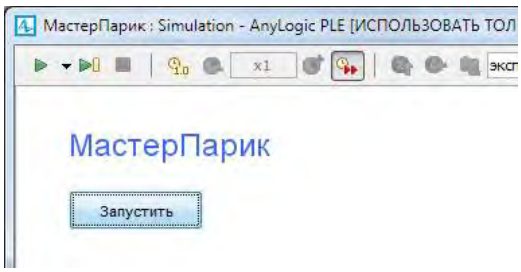


Рисунок 2.6.10. Окно презентации модели

3. Щёлкните данную кнопку. Этим щелчком вы запустите модель и перейдёте к презентации корневого класса активного объекта запущенного эксперимента. Для каждой модели, созданной в **Библиотеке моделирования процессов**, автоматически создаётся блок-схема с наглядной визуализацией процесса, с помощью которой вы можете изучать текущее состояние модели, например, длину очереди, количество обработанных запросов и так далее (рис. 2.6.11).

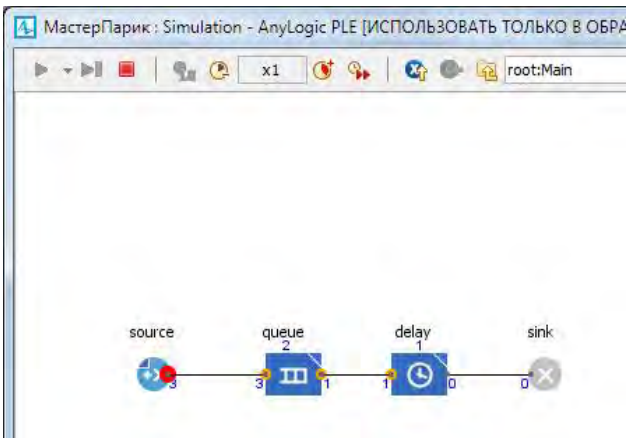


Рисунок 2.6.11. Модель остановилась с ошибкой

4. 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. 2.6.11). Ошибка означает, что клиент не может покинуть объект **source** и войти в блок **queue**, так как его ёмкость, равная 2, заполнена. Также выдаётся сообщение о логической ошибке в модели (рис. 2.6.12)

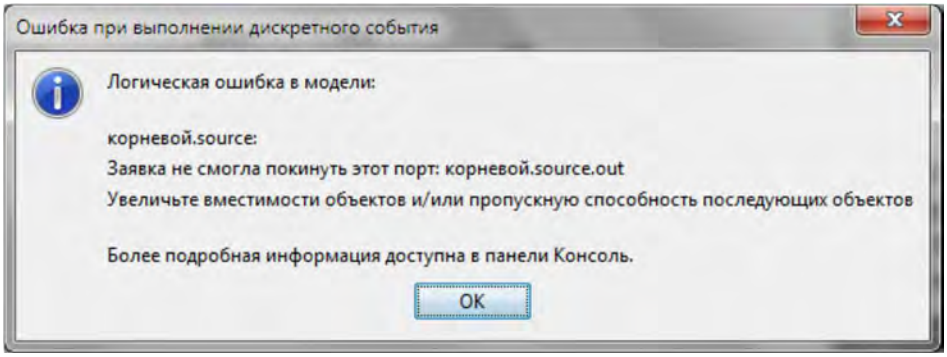


Рисунок 2.6.12. Сообщение о логической ошибке в модели

5. За состоянием любого объекта диаграммы процесса можно следить во время выполнения модели с помощью окна инспекта этого объекта. Чтобы открыть окно инспекта, нужно щёлкнуть мышью по значку нужного блока.

6. 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**.

7. для остановки выполнения модели, Щёлкните мышью кнопку **Прекратить выполнение** панели управления окна презентации.

Для предотвращения остановок модели по ранее указанной ошибке – недостаточной ёмкости объекта **queue –** необходимо доработать модель. Для этого в модели необходимо предусмотреть ветку ухода при критическом значении числа клиентов в очереди. Тем самым мы выполним условия, указанные в постановке задачи.

2.6.1.6 Уточнение модели согласно ёмкости входного буфера

Все клиенты, вырабатываемые объектом **source**, имеют один и тот же приоритет. Поэтому при полном заполнении накопителя (очереди) теряться будет последний входящий клиент. Внесём изменения в модель.

1. Выделите объект **queue**. На панели **Свойства установим Вместимость 2** клиента.

2. Здесь же установите **Разрешить вытеснение**.

3. Для уничтожения потерянных клиентов вследствие полного заполнения накопителя нужно добавить второй объект **sink**. Откроем в **Палитре Библиотеку моделирования процессов** и перетащите блок **sink** на диаграмму (рис. 2.6.13).

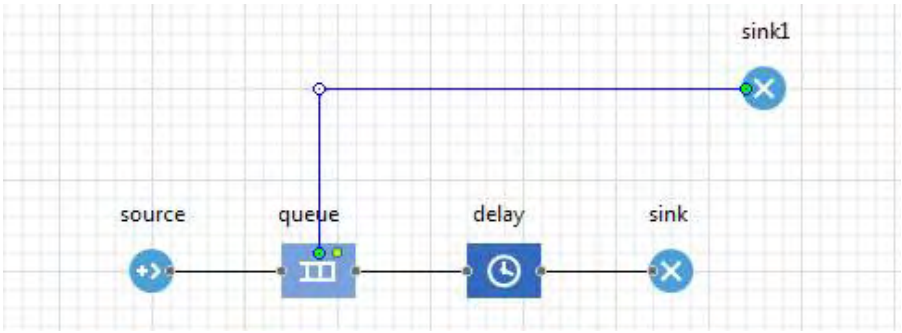


Рисунок 2.6.13. Работа модели согласно ёмкости входного буфера

4. Соединим порт outPreempted объекта queue с входным портом InPort блока **sink1**.

5. Запустим модель и убедимся в отсутствии ошибки исполнения кода. Кроме того на диаграмме процесса можно заметить уход не обслуженных клиентов.

2.6.1.7 Создание нестандартного Java класса

Для выполнения расчётов в модели необходимо создать нестандартный тип агента. Создадим тип агента **Inquiry**.

1. Откроем палитру **Библиотека моделирования процессов**.

2. Перетащим элемент **Тип агента** в графический редактор.

3. В диалоговом окне **Шаг 1. Создание нового типа агента** (рис. 2.6.14) введите **Inquire**, нажмите кнопку **Далее,** выберите анимацию агента 2D, выберите из выпадающего списка, например, **Человек** (рис. 2.6.15)**,** нажмите кнопку **Далее.**

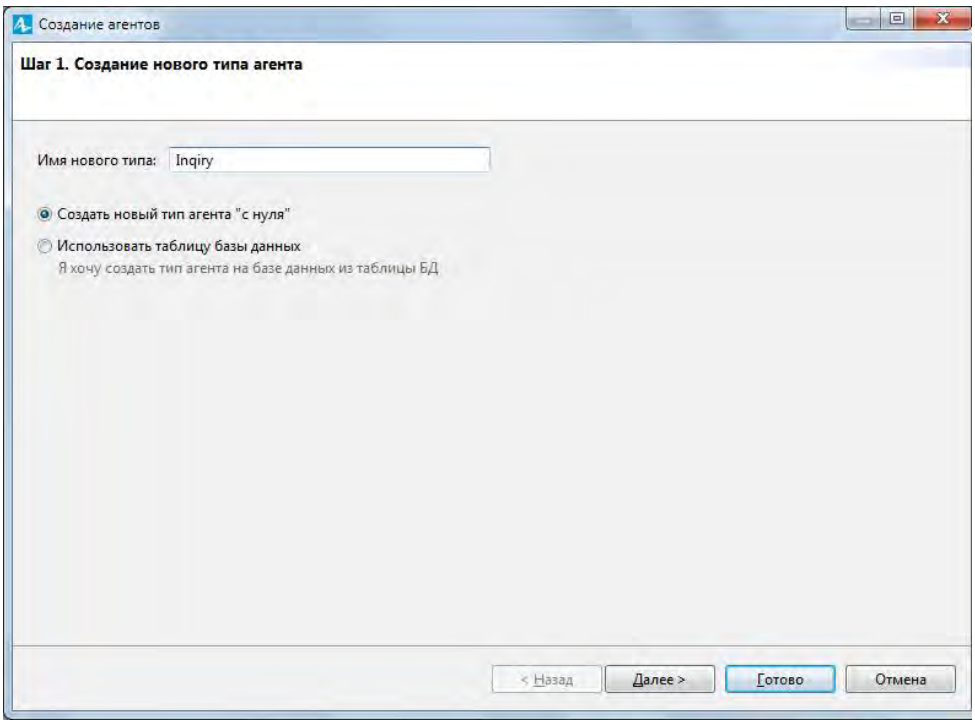


Рисунок 2.6.14. **Окно** создание агента **Inquiry**

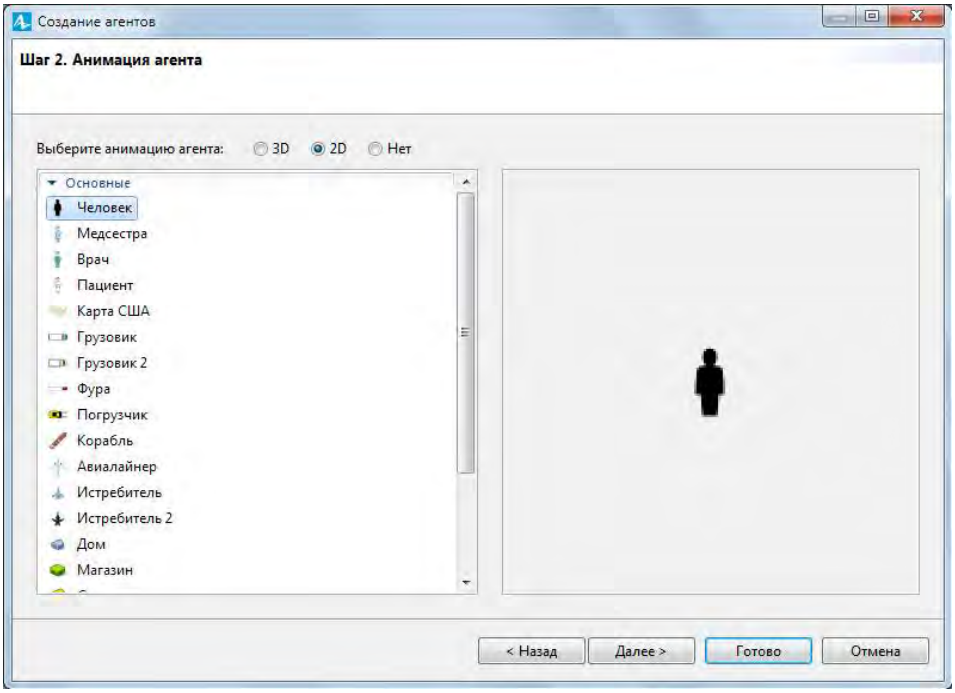


Рисунок 2.6.15. **Диалоговое окно** Создание агента.
Шаг 1. Анимация агента

4. Появится диалоговое окно **Создание агента. Шаг 3. Параметры агента** (рис. 2.6.16).

5. Нажмите **<добавить…>**. В поле **Параметр:** введите t_vxod (рис. 2.6.16).

6. Из выпадающего списка **Тип:** выберите double.

7. Нажмите **<добавить…>**. В поле **Параметр:** введите t_vixod.

8. Из выпадающего списка **Тип:** выберите double.

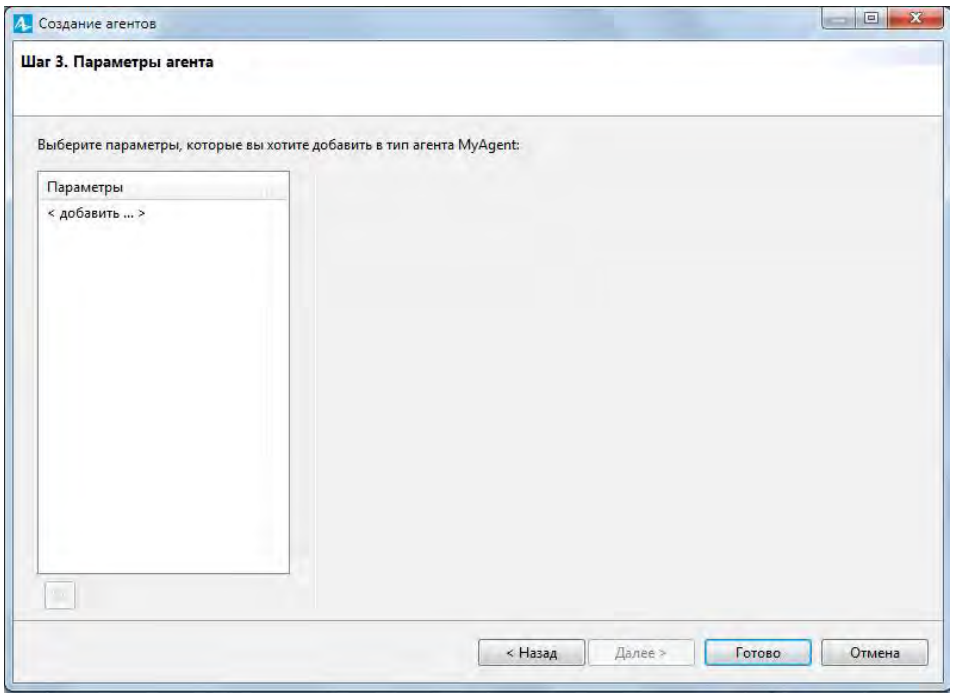


Рисунок 2.6.16. **Диалоговое окно** Создание агента.
Шаг 3. Параметры агента

9. Нажмите **<добавить…>**. В поле **Параметр:** введите c_vxod.
10. Из выпадающего списка **Тип:** оставьте int.
11. Нажмите **<добавить…>**. В поле **Параметр:** введите c_vixod.
12. Из выпадающего списка **Тип:** оставьте int.

Получим следующие параметры (рис. 2.6.17).

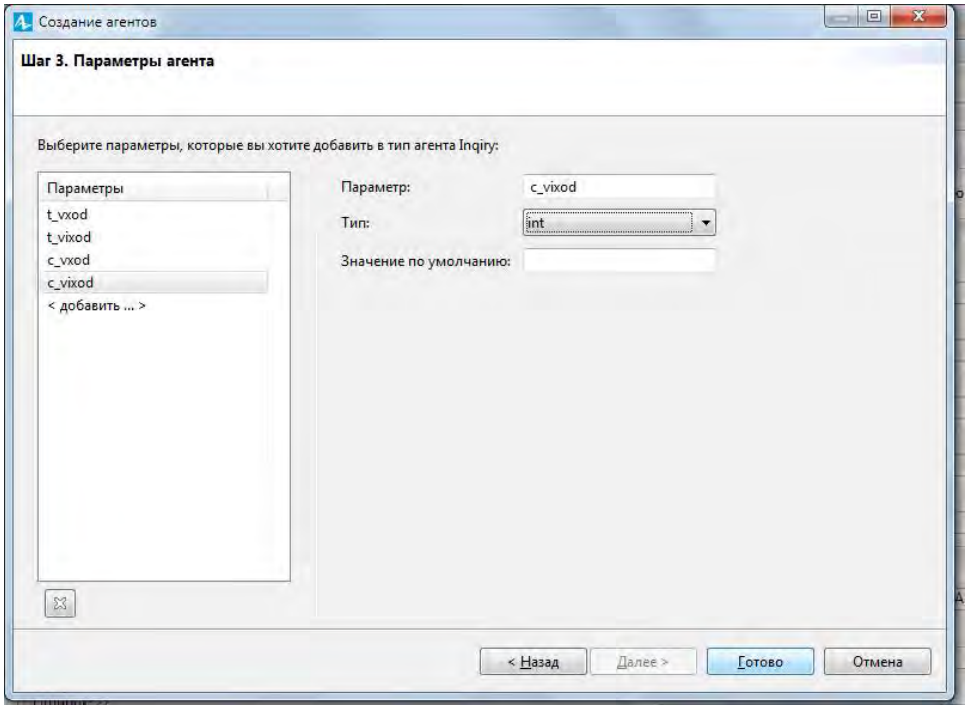


Рисунок 2.6.17. **Диалоговое окно** Создание агента.
Шаг 3. Ввод параметров агента

13. Нажмите кнопку **Готово**. Мы увидим окно, в котором будут показаны автоматически созданные параметры нестандартного типа агента **Inquiry**. (рис. 2.6.18). Закроем окно.

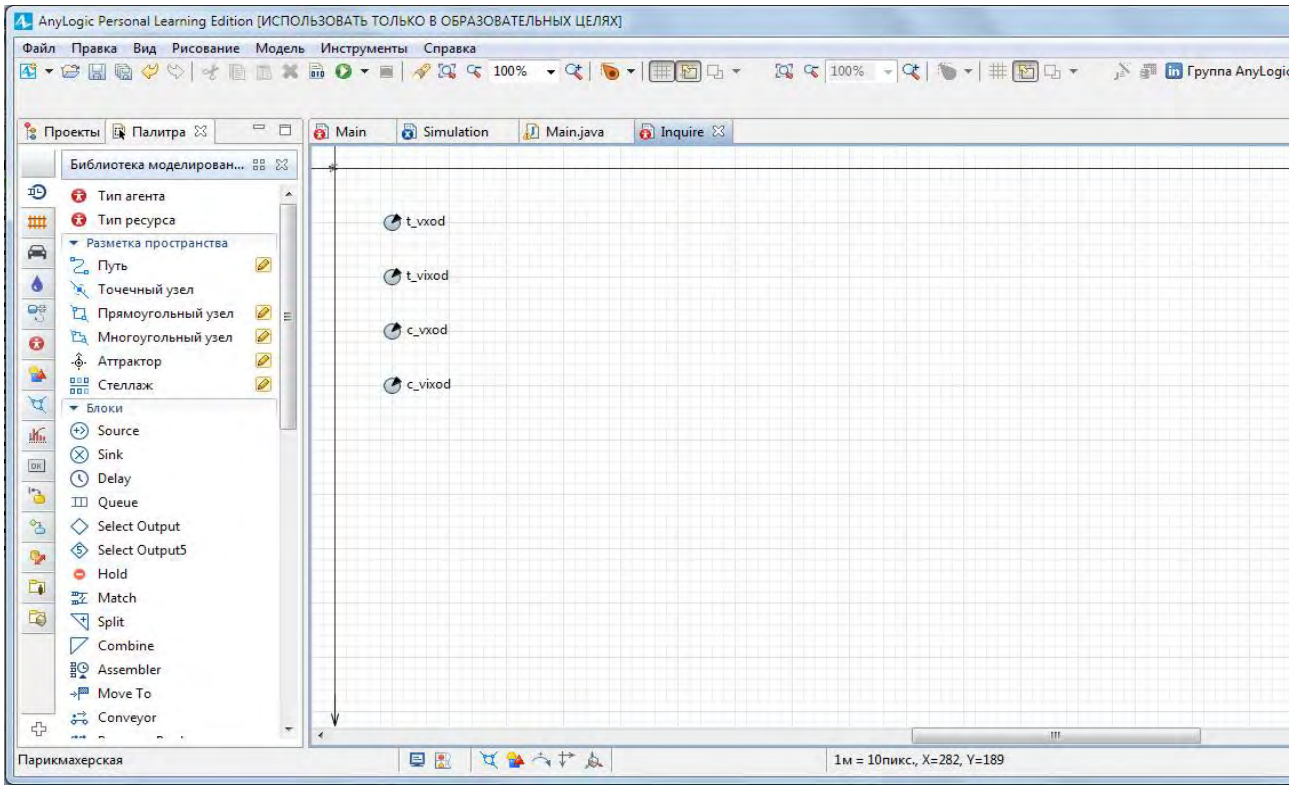


Рисунок 2.6.18. Окно с параметрами нестандартного типа заявки **Inquiry**

2.6.1.8 Добавление элементов статистики

Для сбора статистических данных о времени обслуживания клиентов парикмахерской необходимо добавить элемент статистики. Этот элемент будет запоминать значения времён для каждого клиента.

1. Перетащите элемент **Данные гистограммы** с палитры **Статистика** на диаграмму активного класса **Main**.

2. Задайте свойства элемента (рис. 2.6.19).

Измените **Имя:** на t_obrabotki, сделайте **Кол-во интервалов** равным 50, задайте **Нач. размер интервала** равным **0.01**.

Добавьте ещё элемент сбора статистики для определения вероятности обслуживания клиента.

1. Перетащите элемент **Данные гистограммы** с палитры **Статистика** на диаграмму активного класса **Main**.

2. Задайте свойства элемента (рис. 2.6.20).

Измените **Имя:** на ver_obrabotki, сделайте **Кол-во интервалов:** равным 50, задайте **Нач. размер интервала:** 0.01.

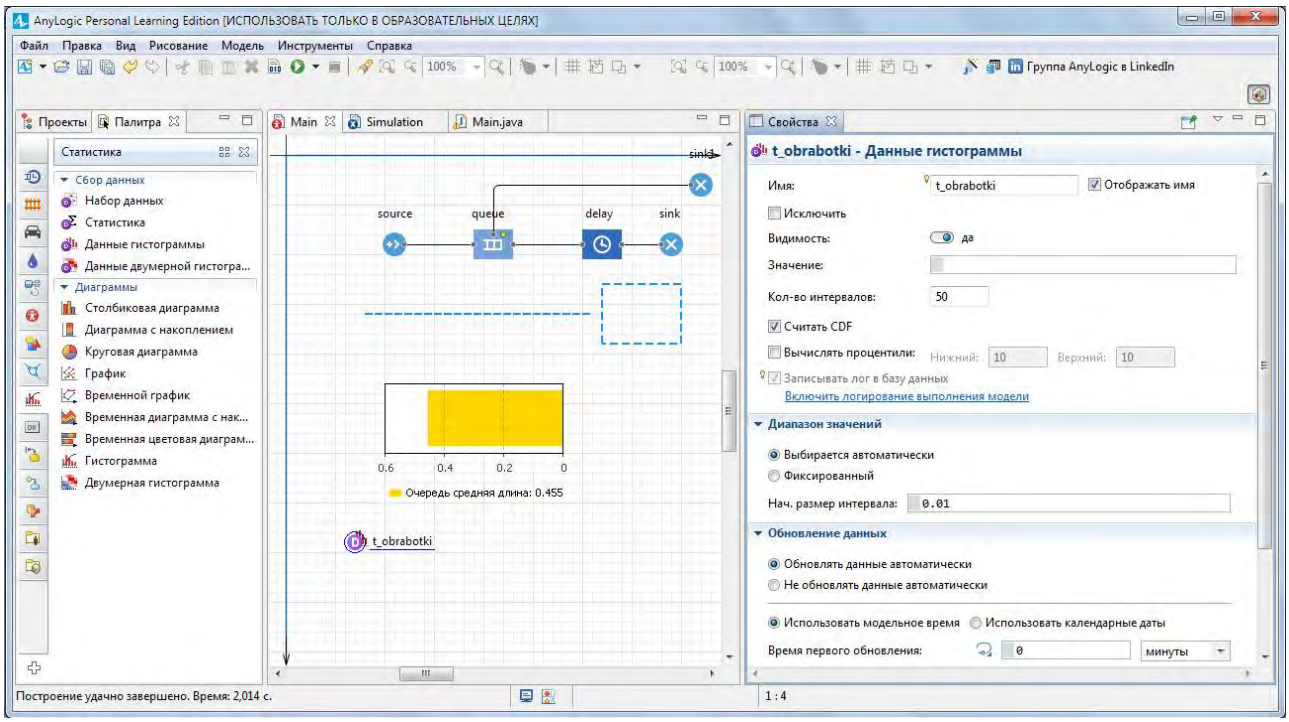


Рисунок 2.6.19. Элемент сбора статистики о времени обработки клиента

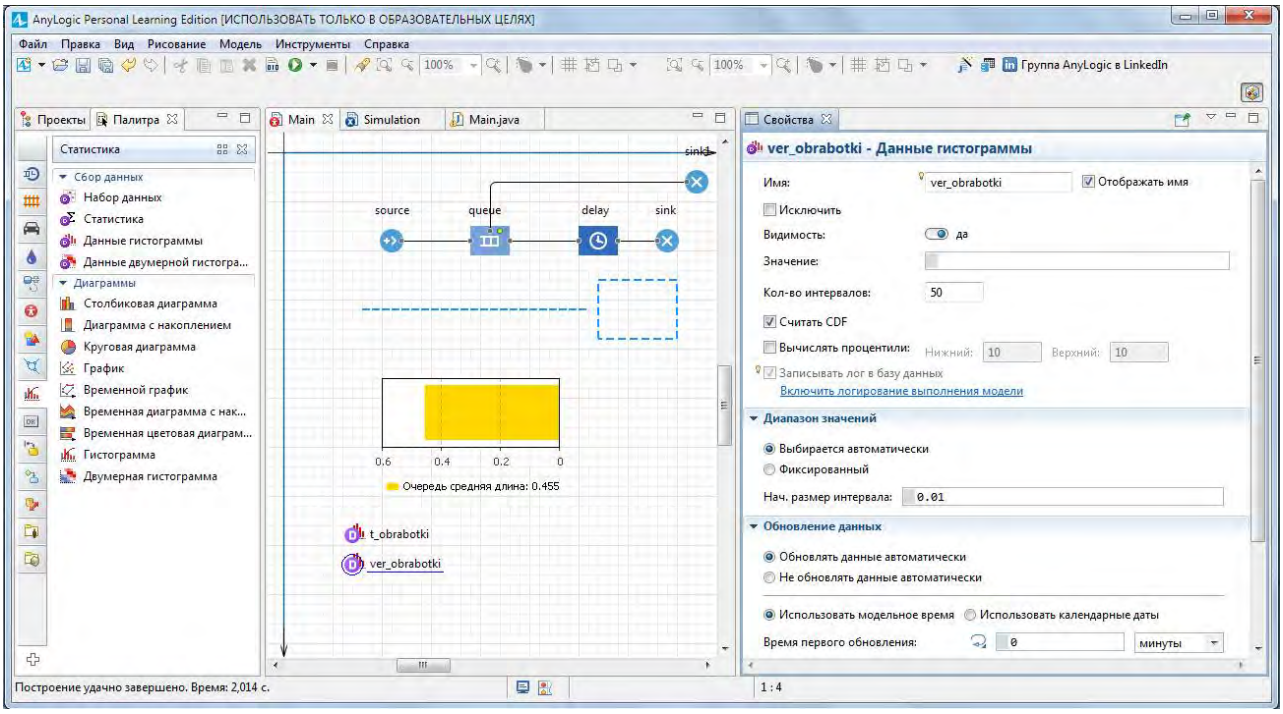


Рисунок 2.6.20. Элемент сбора статистики о вероятности обработки клиентов

2.6.1.9 Изменение свойств объектов диаграммы

Чтобы создавать заявки нестандартного типа **Inquiry**, 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 **Agent**. Поэтому они не позволят явно обращаться к дополнительным полям класса **Inquiry**.

Чтобы разрешить доступ к полям нестандартного типа агента в коде динамических параметров объектов потоковой диаграммы, нужно указать имя нестандартного типа агента качестве **Типа агента** этого объекта. В рассматриваемой потоковой диаграмме с учётом блока **Source** всего пять объектов.

Измените их свойства.

1. Измените свойства объекта **source** (рис. 2.6.21).

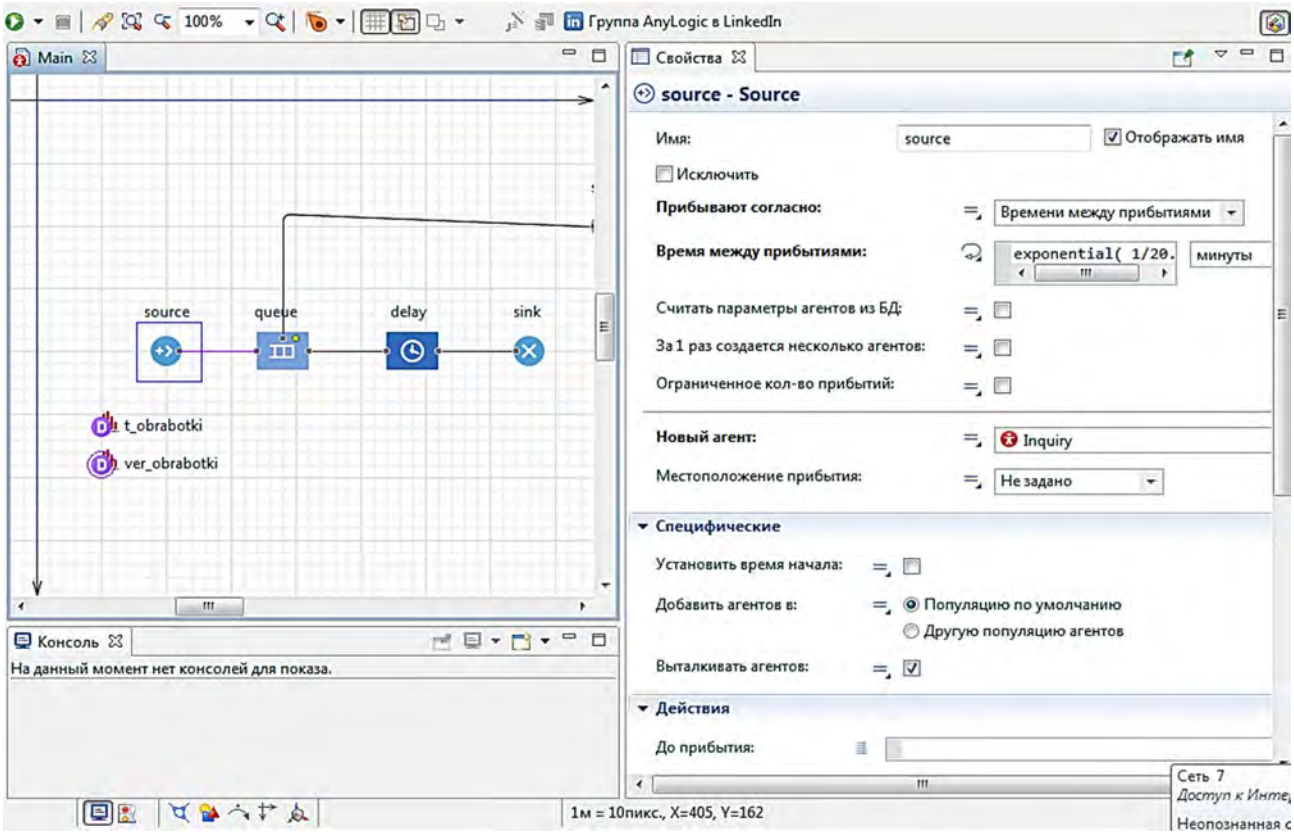


Рисунок 2.6.21. Объект **Source** с изменёнными свойствами

Введите **Inquiry** в поле **Тип агента.** Это позволит напрямую обращаться к полям типа заявки **Inquiry** в коде динамических параметров этого объекта. Теперь этот объект будет создавать заявки типа **Inquiry**.

Введите **agent.t_vxod=time()**; в поле **Действия При выходе**. Код будет сохранять время создания клиента – заявки в параметре t_vxod заявки типа **Inquiry**.

Функция time() возвращает текущее значение модельного времени.

2. Измените свойства объекта **queue**:

Введите **Inquiry** в поле **Тип агента:**.

3. Измените свойства объекта **delay**:

Введите **Inquiry** в поле **Тип агента**.

4. Измените свойства объекта **sink1**:

введите **Inquiry** в поле **Тип агента:**.

5. Измените свойства объекта **sink**:

введите Inquiry в поле **Тип агента:**;

введите в поле **Действие при входе** следующие коды:

t_obrabotki.add(time()-agent.t_vxod)

Этот код добавляет время обработки одного запроса в объект сбора данных гистограммы t_obrabotki. Данное время определяется как разность между текущим модельным временем time() и временем входа клиента в модель. add – встроенная функция добавления элемента в массив.

agent.c_vixod=sink.count();

agent.c_vxod=source.count();

Эти коды заносят количество клиентов, 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** клиентов.

ver_obrabotki.add(agent.c_vixod/agent.c_vxod);

Этот код добавляет относительную долю обработанных клиентов 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 обслуживания клиентов парикмахерской.

И так, модель работы парикмахерской создана с учётом требований задания. Теперь можно провести эксперименты с использованием модели, изменяя входные параметры.

Для образного восприятия работы модели создадим анимацию

2.6.1.10 Создание анимации модели

Анимация модели создаётся в той же диаграмме (в графическом редакторе), в которой задаётся и диаграмма моделируемого процесса.

1. Откройте палитру **Разметка пространства**.

2. Палитра **Разметка пространства** содержит в качестве элементов различные примитивные фигуры, используемые для рисования презентаций моделей.

3. Выделите элемент **Прямоугольный узел** и перетащите его на диаграмму класса активного объекта. Поместите элемент **Прямоугольный узел** так, как показано на рисунке 2.6.22.

4. Чтобы цвет этого прямоугольного узла менялся в зависимости от того, обслуживается клиент или нет, выделите нарисованную фигуру на диаграмме. Перейдите на страницу **Внешний вид** панели свойств (рис. 2.6.23) и в поле **Цвет заливки:** по стрелке, выберите **Динамическое значение** и введите там следующую строку: delay.size()>0?red:green. Здесь **delay** – это имя объекта **delay**. Функция size()

возвращает число запросов, обслуживаемых в данный момент времени. Если сервер занят, то цвет прямоугольника будет красным, в противном случае – зелёным.

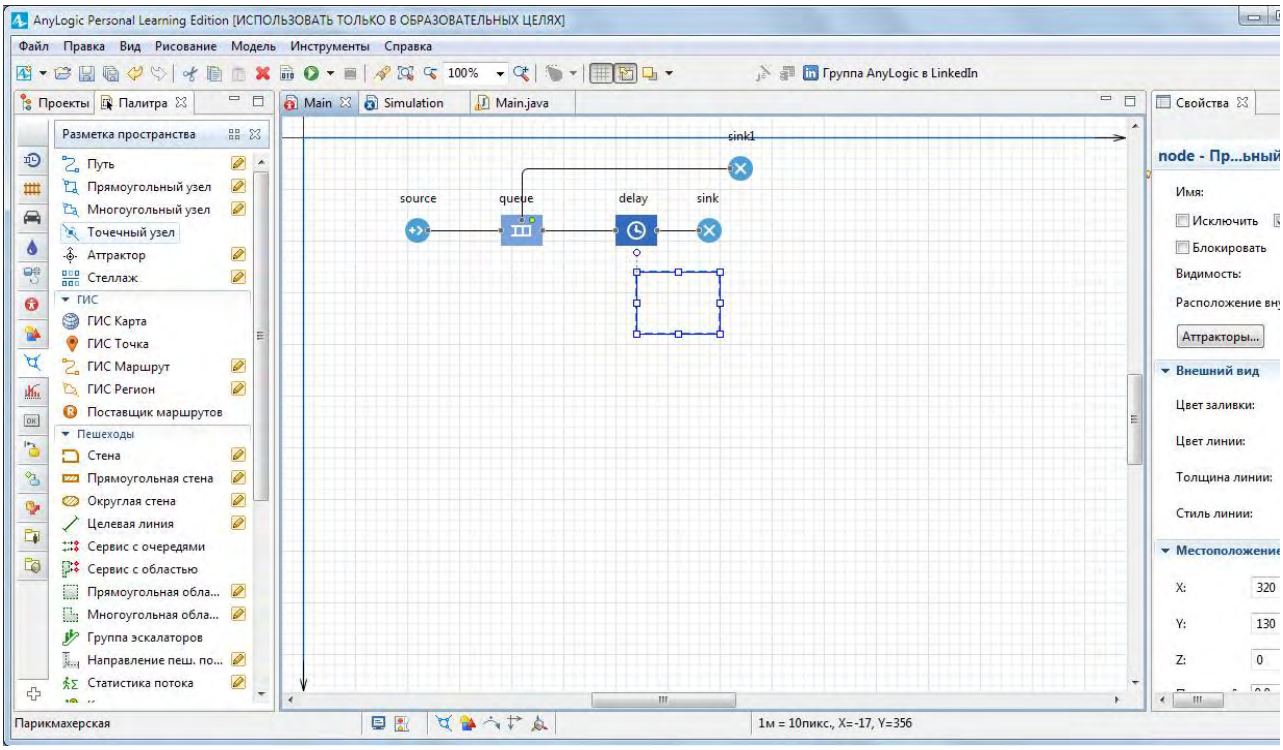


Рисунок 2.6.22. Элемент **Прямоугольный узел** на диаграмме

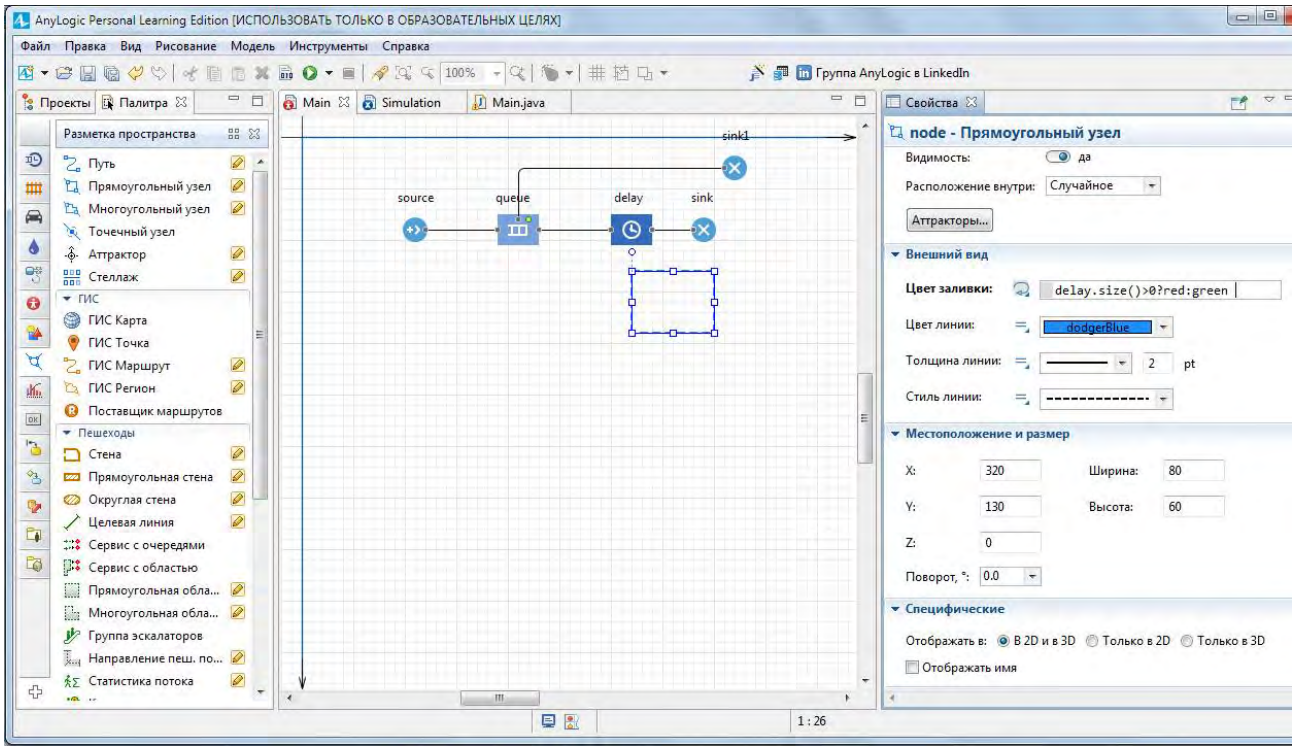


Рисунок 2.6.23. Динамическое значение цвета заливки

5. Нарисуйте путь, который будет обозначать на анимации очередь к парикмахеру (рис. 2.6.24). Чтобы нарисовать путь, сделайте двойной щелчок

мышью по элементу **Путь** палитры **Разметка пространства**, чтобы перейти в режим рисования. Теперь вы можете рисовать путь точка за точкой, последовательно щелкая мышью в тех точках диаграммы, куда вы хотите поместить вершины пути.

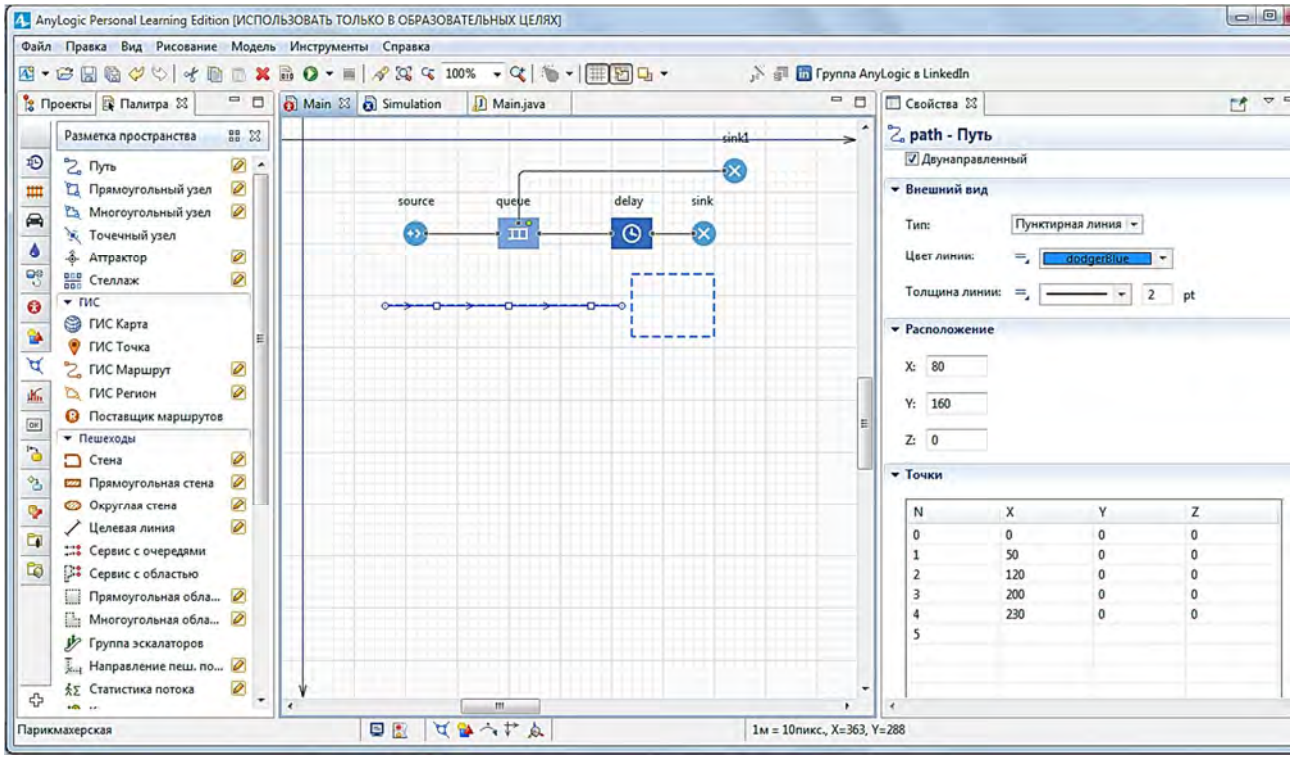


Рисунок 2.6.24. Путь на диаграмме процесса

6. Выделите объект **queue**. На странице свойств объекта queue в поле **Место агентов:** выберите из выпадающего списка path.

7. Задайте прямоугольный узел в качестве фигуры анимации парикмахерской. Выделите объект **delay**. Введите в поле **Место агентов:** из выпадающего списка имя нашего прямоугольного узла: node.

8. Запустите модель. Цвет фигуры парикмахерской будет меняться в зависимости от того, обслуживается клиент в данный момент времени или нет.

2.6.1.11 Сбор статистики использования ресурсов

В процессе работы модели в её объектах накапливаются статистические данные. Для их визуализации можно использовать диаграммы, с помощью которых можно изобразить, например, статистику занятости парикмахерской и длины очереди

Добавим диаграмму для отображения среднего коэффициента использования парикмахерской:

1. Откройте палитру **Статистика**.

2. Перетащите элемент **Столбиковая диаграмма** из палитры **Статистика** на диаграмму класса и измените её размер, как показано на рисунке 2.6.25.

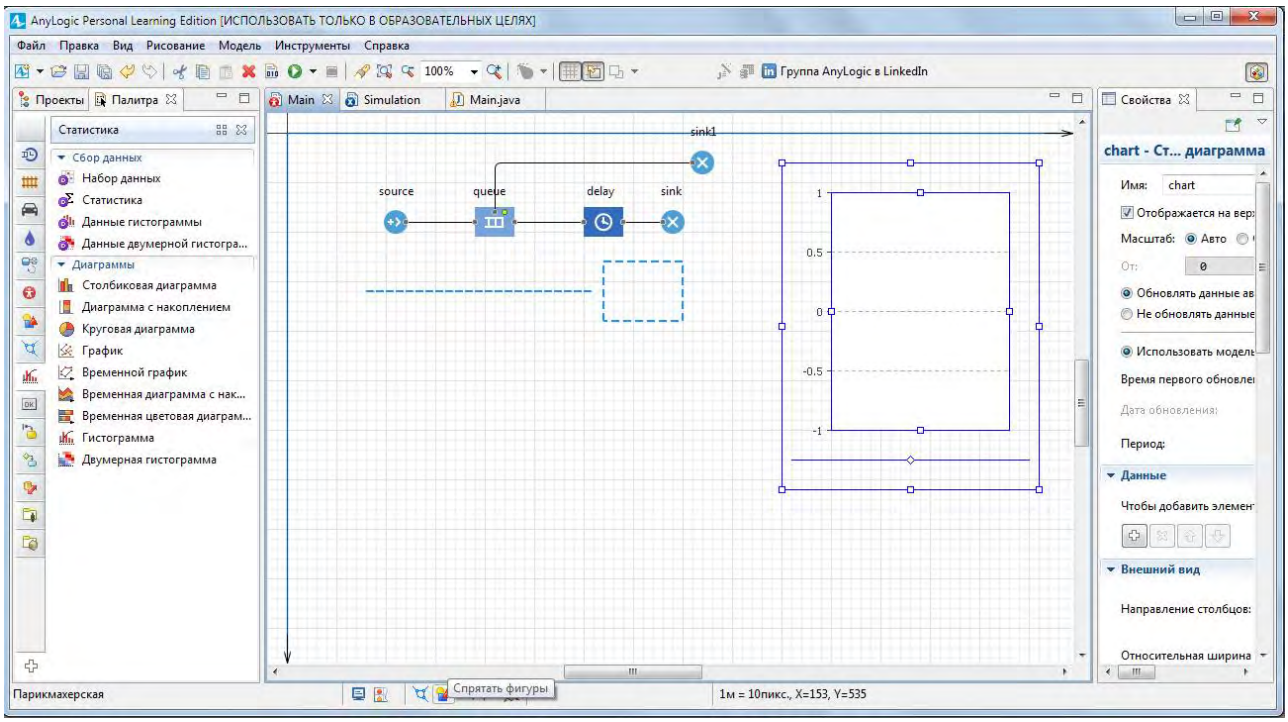


Рисунок 2.6.25. Столбиковая диаграмма

3. Перейдите на панель **Свойства**. Щёлкните кнопку + в секции **Данные**. В результате появится секция свойств того элемента данных (**chart – Столбиковая диаграмма**), представленная на рисунке 2.6.26.

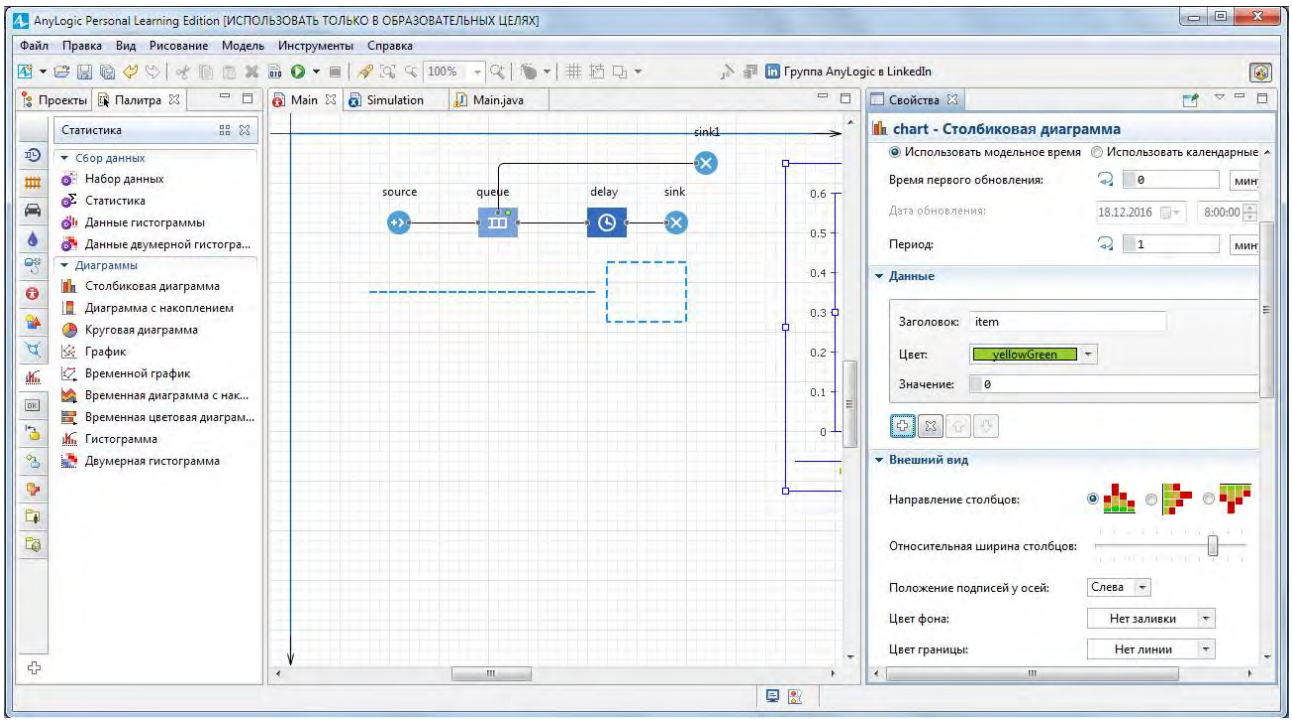


Рисунок 2.6.26. Страница **Свойства**

4. Измените **Заголовок** на Парикмахерская.

5. Введите delay.statsUtilization.mean() в поле **Значение**. Здесь delay – это имя объекта **delay**. statsUtilization, используется для сбора статистики объекта. Функция mean() возвращает среднее из всех измеренных этим набором данных значений. Можно использовать и другие методы сбора статистики, такие, как min() или max().

6. На закладке **Внешний вид можно установить** направление столбцов, цвета фона.

7. На закладке **Местоположение и размер**, **Легенда**, **Область диаграммы** (рис. 2.6.27) можно изменить расположение легенды относительно диаграммы, размер диаграммы, высоту, ширину, координаты размещения на диаграмме, цвета текста, границы.

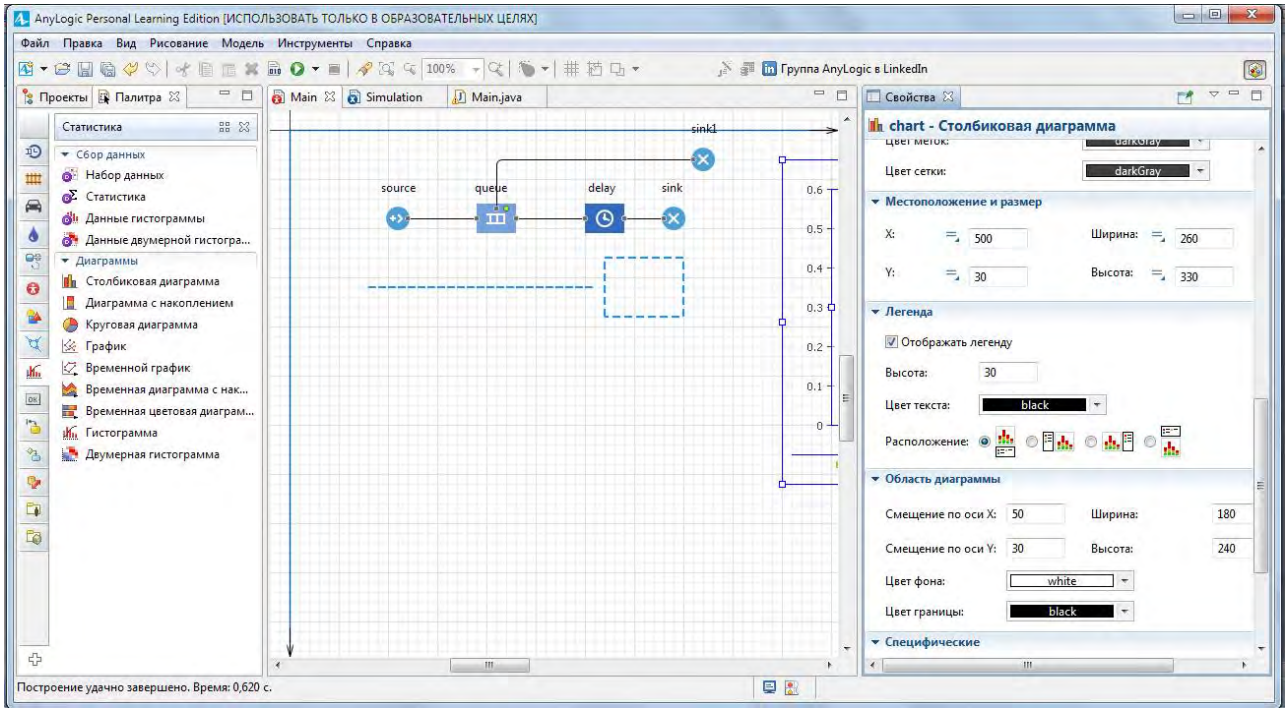


Рисунок 2.6.27. Местоположение и размер диаграммы

8. Аналогичным образом создадим столбиковую диаграмму для отображения средней длины очереди.

9. **Заголовок:** и **Значение:** изменим так, как показано на рисунке 2.6.27. В поле **Заголовок:** введите Очередь средняя длина, а в поле **Значение:** введите queue.statsSize.mean().

2.6.1.12 Расчёт дохода парикмахерской

На основе построенной модели процесса работы парикмахерской можно рассчитать прибыль предприятия. В данном примере она складывается из доходов

от клиентов за вычетом затрат на обустройство очереди и затрат на содержание парикмахеров. Для проведения расчёта необходимо ввести изменения в модель.

1. Из палитры **Агент** перетащим в **Main** параметр и назовём его **Прибыль.** В нем будет записываться рассчитанная прибыль. Сюда же перетащим переменные колПарикмахеров, затратыПарикмахеров, размерОчереди, в которые будут вносится количество парикмахеров, затраты на одного парикмахера, размер обустроенной очереди (рис. 2.6.28).

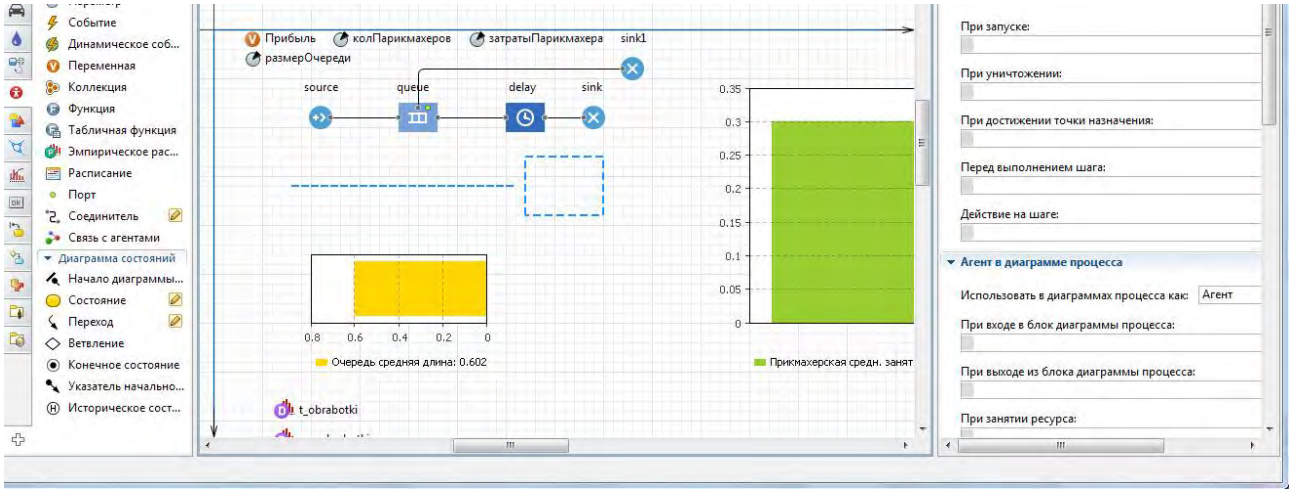


Рисунок 2.6.28. Ввод параметра и переменных

2. Из палитры **Агент** перетащим в Inquire переменные доходОтКлиента, затратыНаОчередь, в которые будут вносится доходы от одного клиента и затраты на обустройство очереди в расчёте на одного клиента (рис. 2.6.29).

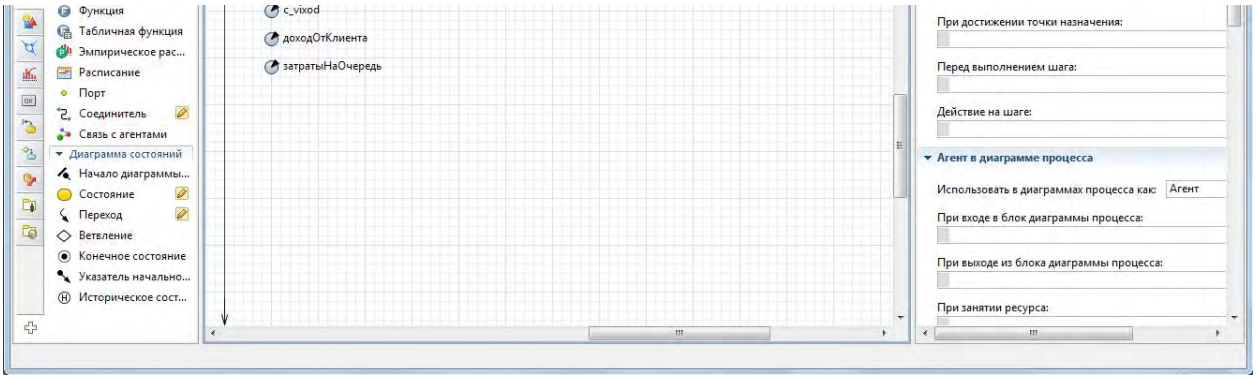


Рисунок 2.6.29. Ввод параметра и переменных

3. В свойстве объекта **queue** в поле **Вместимость** введите размерОчереди эк.

4. В свойстве объекта **sink** в поле **при входе** введите:

Прбыль=(agent.c_vixod*agent.доходОтКлиента)

– (размерОчереди*agent.затратыНаОчередь)

– (колПарикмахеров*затратыПарикмахера); (рис. 2.6.30).

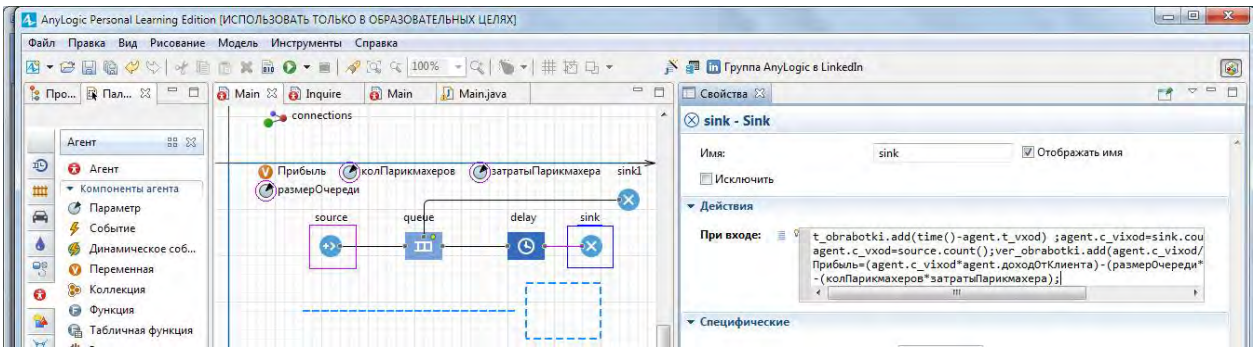


Рисунок 2.6.30. Ввод формулы прибыли

Теперь всё готово для проведения экспериментов для определения прибыли.

При переменных колПарикмахеров=1, затратыПарикмахеров=1, размерОчереди=2, доходОтКлиента=1, затратыНаОчередь=10 прибыль составит 7); (рис. 2.6.31)

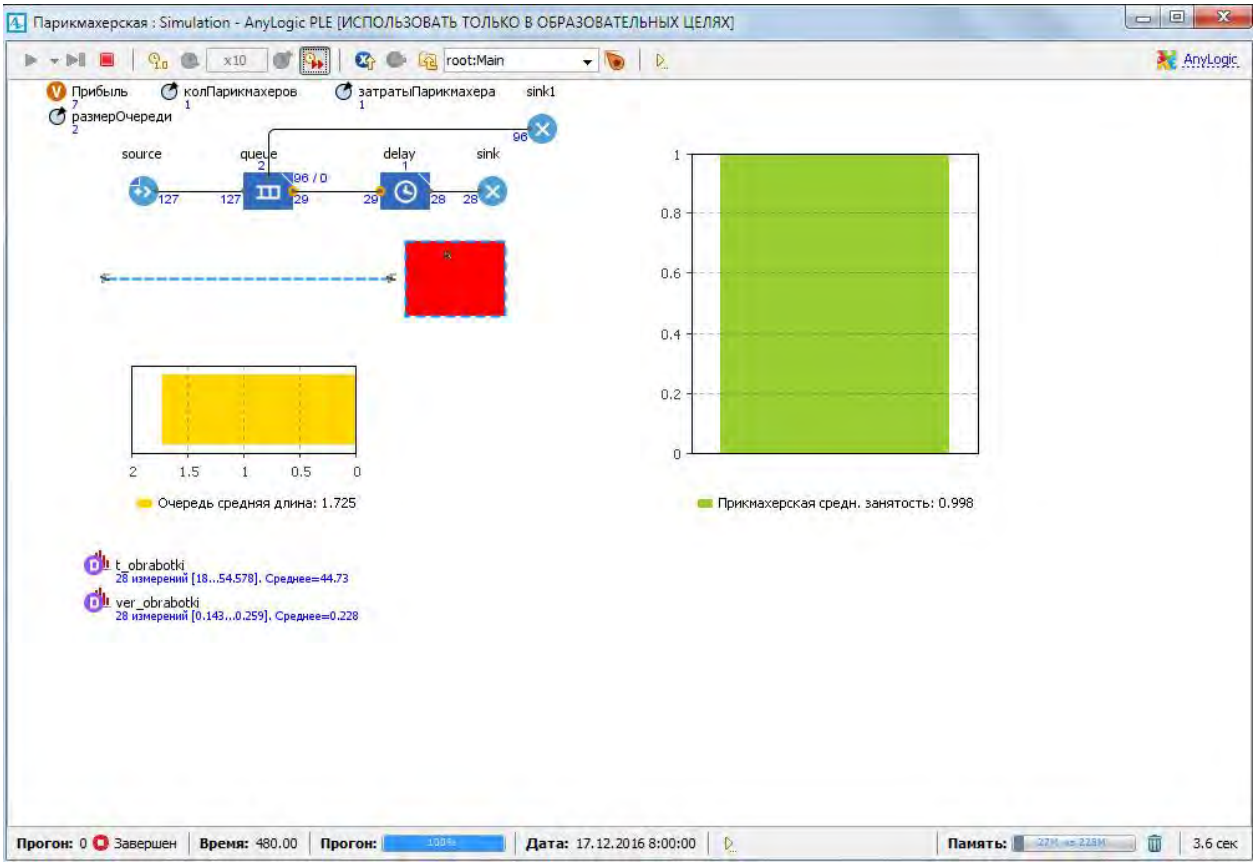


Рисунок 2.6.31. Результаты расчёта прибыли

3. Варианты контрольных заданий

3.1 Задание 1

В соответствии с вариантом задания предлагается:

1. выбрать из таблицы 5 сетевую функцию;

2. построить полином первой степени;

3. определить с помощью имитационного моделирования площадь геометрической фигуры, ограниченной полученным полиномом и осью *x*.

Таблица 5

№ задания	Сетевая функция						
1	x	1	2	3	4	5	6
	y	3	8	11	18	27	38
2	x	1	2	3	4	5	6
	y	2	8	18	32	50	72
3	x	1	2	3	4	5	6
	y	4	10	20	34	52	74
4	x	1	2	3	4	5	6
	y	6	9	14	21	30	41
5	x	1	2	3	4	5	6
	y	9	12	17	24	33	44
6	x	1	2	3	4	5	6
	y	11	14	19	26	35	46
7	x	1	2	3	4	5	6
	y	13	16	21	28	37	48
8	x	1	2	3	4	5	6
	y	10	16	26	40	58	80
9	x	1	2	3	4	5	6
	y	12	18	28	42	60	82
10	x	1	2	3	4	5	6
	y	15	21	31	45	63	85
11	x	1	2	3	4	5	6
	y	2	16	54	128	250	432
12	x	1	2	3	4	5	6
	y	3	17	55	129	251	433
13	x	1	2	3	4	5	6
	y	7	21	59	133	255	437
14	x	1	2	3	4	5	6
	y	20	10	6,6	5	4	3,3
15	x	1	2	3	4	5	6
	y	21	11	7,6	6	5	4.3
16	x	1	2	3	4	5	6
	y	24	14	10,6	9	8	7,3
17	x	1	2	3	4	5	6
	y	40	20	13,3	10	8	6,6
18	x	1	2	3	4	5	6
	y	42	22	15,2	12	10	8,6
19	x	1	2	3	4	5	6
	y	47	27	20,2	17	15	13,6
20	x	1	2	3	4	5	6
	y	100	25	11,1	6,2	4	2,8

3.2 Задание 2

В супермаркете имеется *M* касс. Посетители занимают места в очереди кассы. Как правило, очереди к различным кассам примерно одинаковые. Было замечено, что при длине очереди больше *N* покупатель уходит из супермаркете без покупки. Время прихода покупателей и время их обслуживания имеет экспоненциальную зависимость. Среднее время прихода равно *T*, а среднее время обслуживания равно *Z*. Для привлечения покупателей перед супермаркетом построена парковка на *X* машин. В течение суток супермаркет работает *S* часов.

В соответствии с вариантом задания предлагается построить имитационную модель работы супермаркета и определить статистические характеристики системы согласно исходным данным, представленным в таблице 6.

Таблица 6

№ задания	*M*, число касс (штук)	*N*, длина очереди (человек)	*T*, время прихода покупателя (минута)	*Z*, время обслуживания покупателя (минута)	*X*, Мест на парковке (штук)	*S*, время работы супермаркета (минута)
1	3	5	20	10	4	480
2	6	7	25	15	7	960
3	5	10	16	26	40	580
4	7	12	18	8	25	60
5	6	15	21	31	45	180
6	1	2	16	5	28	250
7	2	3	17	15	12	240
8	3	7	21	9	13	300
9	4	2	10	6	5	960
10	5	11	11	7	6	120
11	1	24	14	10	9	720
12	3	4	20	13.	10	660
13	5	2	22	15	12	580
14	8	7	27	20	17	120
15	3	7	21	9	13	300
16	4	2	10	6	8	960
17	5	9	11	7	6	120
18	1	6	14	10	5	720
19	3	4	20	13.	10	660
20	5	7	23	14	3	580
21	8	5	20	10	16	180

Библиографический список

1. Карпов, Ю. Имитационное моделирование систем. Введение в моделирование с AnyLogic / Ю. Карпов. – СПб. : БХВ-Петербург, 2005.

2. Таха, Х. А. Введение в исследование операций : В 2-х кн. / Х. А. Таха. – М. : Изд. Дом «Вильямс», 2005. – 912 с.

3. Имитационное моделирование. Теория и практика [Электронный ресурс]. – Режим доступа: http://www.gpss.ru

4. AnyLogic. Многоподходное имитационное моделирование [Электронный ресурс]. – Режим доступа: http://www.anylogic.com