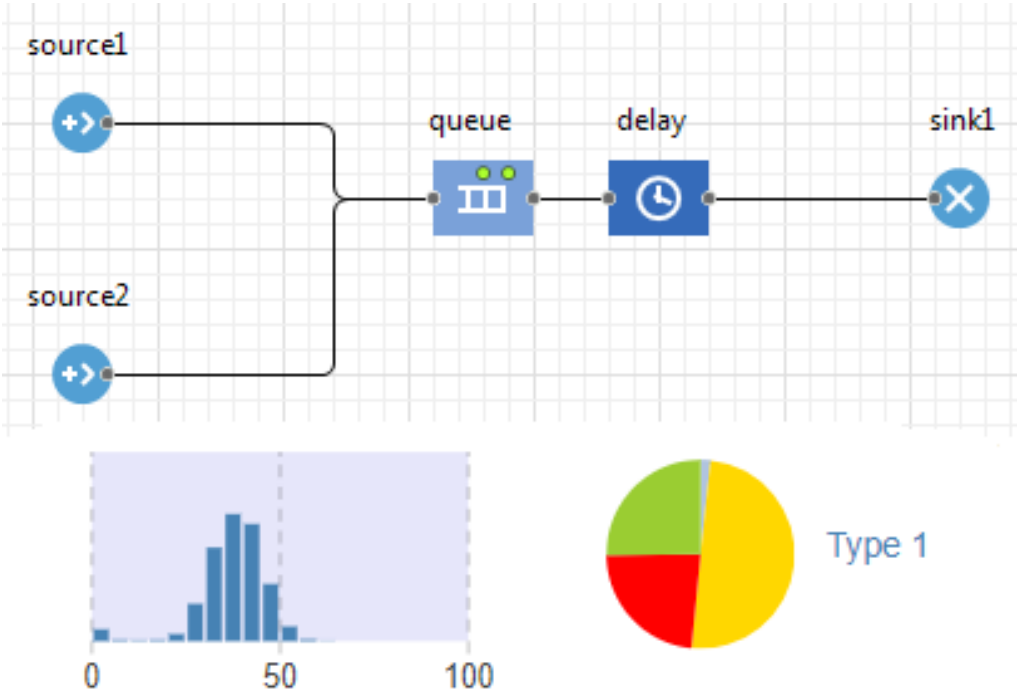
С.А. Олейникова

МОДЕЛИРОВАНИЕ

Учебное пособие



Воронеж 2020

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Воронежский государственный технический университет»

С.А. Олейникова

Моделирование

Учебное пособие

Воронеж 2020

УДК
ББК
О

Рецензенты:

кафедра информационной безопасности и систем связи Международного института компьютерных технологий

Барабанов В.Ф., д.т.н., проф, г. Воронеж

Олейникова С.А.

О??? **Моделирование:** учебное пособие / С.А. Олейникова; ФГБОУ «Воронежский государственный технический университет». – Воронеж: Изд-во ВГТУ, 2020. – 127 с.

ISBN

Рассматриваются современные подходы, используемые для моделирования сложных вычислительных и других систем. Основное внимание уделено разнообразным приемам и возможностям имитационного моделирования, используемым в среде AnyLogic.

Предназначено для студентов направления 09.03.01 «Информатика и вычислительная техника» (профиль «Вычислительные машины, комплексы, системы и сети») очной и заочной форм обучения.

Ил. 122. Табл. 1. Библиогр.: 6 назв.

УДК

ББК

Печатается по решению учебно-методического совета Воронежского государственного технического университета

ISBN

ВВЕДЕНИЕ

Для эффективного функционирования сложных обслуживающих и производственных систем необходимо обеспечить возможность получения разнообразных стохастических характеристик. К таким характеристикам могут относиться средняя длина очереди, среднее время отклика на запрос, загрузка устройств и т.д.

Наличие сложной структуры исследуемой системы с большим количеством связей между своими элементами, а также интенсивное развитие информационных технологий обуславливает применение имитационных методов моделирования для решения данной задачи. Современные среды моделирования отличаются не только возможностью получения необходимой для пользователя информации о характеристиках системы, но развитым графическим интерфейсом, позволяющим, при необходимости, отобразить процесс функционирования систем и процесс их обслуживания в динамике. Это делает аппарат имитационного моделирования неотъемлемой частью системы управления предприятием или организацией, позволяющей получать необходимую информацию в оперативном режиме.

Данное учебное пособие, в основном, посвящено изучению особенностей одной из таких сред – Anylogic. Данный программный продукт выгодно отличается от своих аналогов не только развитым графическим интерфейсом, позволяющим моделировать различные виды перевозок (железнодорожные, автомобильные и т.д.), но и имеющем в своем арсенале новейшие библиотеки, позволяющие использовать агентный подход для моделирования сложных распределенных систем.

Очевидно, что освоение аппарата имитационного моделирования невозможно без теоретических основ, позволяющих усвоить такие понятия, как модель, ее свойства и функции, виды моделирования, классификация моделей в соответствии с теми или иными признаками и т.д. В связи с этим, первая глава пособия посвящена теоретическим аспектам моделирования вообще и имитационного моделирования в частности.

Вторая глава содержит общие сведения о среде Anylogic как наиболее популярной среде имитационного моделирования. Описана специфика работы в данной среде, ее возможности, представлена простейшая дискретно-событийная модель и статистика о ее функционировании. В этой же главе представлен обзор основных библиотек Anylogic. Описаны такие современные подходы к моделированию, как системная динамика, а также использование диаграммы состояний. Здесь также представлены возможности пешеходной библиотеки и библиотеки дорожного движения.

Специфика использования ресурсов описана в третьей части пособия. Данная часть снабжена примерами, иллюстрирующими последовательность действий для моделирования ресурсов, а также их хранение, перемещение и т.д.

1. ТЕОРЕТИЧЕСКИЕ АСПЕКТЫ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ

1.1. Модель. Причины использования моделей

Основными понятиями при моделировании являются понятия объекта (или системы) и модели.

Объект – тот реальный предмет, процесс, который необходимо изучить или описать.

Система – объект, процесс, в котором участвующие элементы связаны некоторыми связями и отношениями.

Модель – объект-заместитель объекта-оригинала, обеспечивающий возможность изучения необходимых свойств объекта-оригинала.

Моделирование – это процесс замещения объекта-оригинала другим объектом с целью получения информации о свойствах исследуемого объекта. При этом модель воспроизводит только те свойства оригинала, которые потребуются при эго исследовании.

Если результаты моделирования подтверждаются, то говорят, что модель *адекватна* объекту. Обобщенно моделирование можно определить как метод опосредованного познания, при котором исследуемый объект-оригинал находится в некотором соответствии с другим объектом-моделью, причем модель способна замещать оригинал на некоторых стадиях исследования.

Моделирование является основным методом исследований во всех областях знаний и научно-обоснованным методом оценок характеристик сложных систем, используемым для принятия решений в различных сферах инженерной деятельности. Современный этап развития вычислительной техники позволяет эффективно исследовать системы с помощью моделей любой сложности.

В основе моделирования лежит теория подобия, которая утверждает, что абсолютное сходство может иметь место лишь при замене объекта точно таким же. При моделировании абсолютное подобие не имеет места. При этом стремятся к тому, чтобы модель достаточно хорошо отображала исследуемую сторону функционирования объекта.

Причины использования моделей

Очевидно, что использование моделирование обусловлено тем, что исследование реального объекта по каким-либо причинам невозможно. Основные причины использования моделей вместо исследуемых систем следующие:

1. Сложность реальных объектов. Число факторов, которые относятся к решаемой проблеме, выходит за пределы человеческих возможностей. Поэтому одним из выходов (а часто единственным) в сложившейся ситуации является упрощение ситуации с помощью моделей, в результате чего

уменьшается разнообразие этих факторов до уровня восприимчивости специалиста.

2. Необходимость проведения экспериментов. На практике встречается много ситуаций, когда экспериментальное исследование объектов ограничено высокой стоимостью или вовсе невозможно (опасно, вредно, ограниченность науки и техники на современном этапе).

3. Необходимость прогнозирования. Важное достоинство моделей состоит в том, что они позволяют «заглянуть в будущее», дать прогноз развития ситуации и определить возможные последствия принимаемых решений.

4. Среди других причин можно назвать следующие:
• исследуемый объект либо очень велик (модель Солнечной системы), либо очень мал (модель атома);

• процесс протекает очень быстро (модель двигателя внутреннего сгорания) или очень медленно (геологические модели);

• исследование объекта может привести к его разрушению (модель самолета, автомобиля).

1.2. Требования, предъявляемые к модели. Функции модели

1.2.1 Требования

Модель должна быть:

1) *простой и понятной* пользователю;

2) *удобной* в управлении и «общении» с ней;

3) *универсальной* (то есть применимость модели к анализу ряда однотипных систем в одном или нескольких режимах функционирования. Это позволяет расширить область применимости модели для решения большего круга задач);

4) *точной* (т.е. позволять получать результаты близкие к заранее установленным)

5) *полной*, с точки зрения решения главных задач;

6) *адаптивной*, т.е. позволять без существенных усилий переходить к другим модификациям или обновлять данные, т.е. совершенствоваться во взаимодействии с пользователем;

7) *адекватной*, т.е. в полной мере в соответствии с постановкой задачи воспроизводить оригинал;

8) *экономичной*, то есть точность получаемых результатов и общность решения задачи должны увязываться с затратами на моделирование. И удачный выбор модели, как показывает практика, — результат компромисса между отпущенными ресурсами и особенностями используемой модели.

1.2.2 Функции модели

В силу того, что идея представления системы с помощью модели носит столь общий характер, четкого определения функций модели не приводят. Тем не менее, можно определить следующие функции:

1) *Познавательная* (модель как средство осмысления действительности). Моделирование является одним из основных методов познания, формой отражения действительности и заключается в выяснении или воспроизведении тех или иных свойств реальных объектов, предметов и явлений с помощью других объектов (моделей) с помощью абстрактного описания в виде изображения, плана, карты, совокупности уравнений, алгоритмов и программ.

2) *Коммуникативная* (модель как средство общения). Как средство общения правильно построенные модели помогают исследователю устранить неточности человеческого языка, предоставляя более действенные и более успешные способы общения. Преимущество модели перед словесными описаниями – в сжатости и точности представления заданной ситуации. Модель делает более понятной общую структуру исследуемого объекта и вскрывает важные причинно-следственные связи.

3) *Тренировочная* (модель как средство обучения и тренажа). Моделирование получило широкое применение в качестве средства профессиональной подготовки и обучения. В частности, они распространены для обучения лиц, которые должны справляться со случайными факторами до возникновения реальной критической ситуации (например, модели космических кораблей, используемые для тренировки космонавтов, тренажеры для обучения машинистов поездов, деловые игры для обучения административного персонала фирм).

4) *Прогностическая* (модель как средство предсказания развития событий). Моделирование позволяет осуществлять прогнозирование поведения реальных объектов. Например, исследование летных характеристик реактивного самолета или космического корабля возможно без построения данного объекта. Моделирование может быть использовано задолго до того, как реальный объект будет построен. В процессе такого моделирования выявляются возможные характеристики объекта, определяются рациональные способы их построения, оптимальные приемы управления и прогнозируются критические ситуации, которые не исключены при функционировании конструированных объектов.

5) *Экспериментальная* (модель как средство постановки экспериментов). Применение моделей позволяет проводить контролируемые эксперименты в ситуациях, где экспериментирование на реальных объектах было бы практически невозможным, экологически опасным или экономически нецелесообразным.

6) *Управленческая* (модель как средство принятия решений с целью планирования процессов и управления ими). Основными направлениями моделирования управления производственно-экономическими системами

является создание моделей управления производством. В настоящее время находят применение следующие модели следующих функций управления производством:

- планирования производственно-экономической деятельности предприятия;
- управления материально-техническим снабжением производства;
- управления сбытом готовой продукции

И т.д.

7) *Метрологическая* (модель как средство совершенствования измерений). Моделирование в метрологии используется для обоснованного планирования измерений и правильной интерпретации результатов и погрешностей измерений необходимо на начальном этапе решения задачи измерений (например, при разработке методики выполнения измерений).

1.3. Классификация моделей

В процессе моделирования необходимо определить отличительные особенности исследуемого объекта (системы). Они помогут выбрать такой тап модели, которая будет наилучшим образом отражать данные особенности. Приведем классификацию моделей. Она будет зависеть, в первую очередь, от признака, согласно которому будет проводиться данная классификация. В настоящее время существует целая серия, признаков, согласно которому можно выделить разные группы моделей.

Классификация в зависимости от характера изучаемых объектов

В зависимости **от характера изучаемых процессов** в системе S все виды моделирования могут быть разделены на:

- детерминированные и стохастические;
- статические и динамические;
- дискретные, непрерывные и дискретно-непрерывные.

Детерминированное моделирование отображает процессы, в которых предполагается отсутствие всяких случайных воздействий; стохастическое моделирование отображает вероятностные процессы и события.

Статическое моделирование служит для описания поведения объекта в какой-либо момент времени, в динамическое моделирование отражает поведение объекта во времени.

Дискретное моделирование служит для описания дискретных процессов (т.е. процессов, область возможных значений реализаций которого есть конечное или счетное множество). Непрерывное моделирование позволяет отразить непрерывные процессы (т.е. случайные процессы, область возможных значений реализаций которого есть несчетное множество). Дискретно-непрерывное моделирование используется для случаев, когда в системе необходимо выделить наличие как дискретных, так и непрерывных процессов.

Данный вид классификации приведен на *рис.1.1*.

Классификация в зависимости от форм представления объекта

В зависимости от **формы представления объекта** можно выделить мысленное и реальное моделирование. Основу мысленного моделирования составляет имеющаяся информация о реальном объекте. Это теоретический метод познания окружающей среды. Мысленное моделирование может быть реализовано в виде наглядного, символического и математического.

Рис. 1.1. Классификация моделей в зависимости от характера изучаемых объектов

При наглядном моделировании на базе представления человека о реальных объектах создаются наглядные модели, отображающие явления и процессы, протекающие в объекте. Наглядное моделирование, в свою очередь, можно разделить на гипотетическое, аналоговое моделирование и макетирование. В основе гипотетического моделирования исследователем закладывается некоторая гипотеза о закономерностях протекания процесса в реальном объекте, которая отражает уровень знаний исследователя об объекте и базируется на причинно-следственных связях между входом и выходом

изучаемого объекта. Гипотетическое моделирование используется, когда знаний об объекте недостаточно.

Аналоговое моделирование основывается на применении аналогий различных уровней. Наивысшим уровнем является полная аналогия, имеющая место только для достаточно простых объектов.

Мысленный макет может применяться в случаях, когда протекающие в реальном объекте процессы не поддаются физическому моделированию, либо может предшествовать проведению других видов моделирования.

Символическое моделирование представляет собой искусственный процесс создания логического объекта, который замещает реальный и выражает основные его свойства с помощью определенной системы знаков или символов.

Если ввести условные обозначения отдельных понятий, т.е. знаки, а также определенные операции между этими знаками, то можно реализовать знаковое моделирование.

В основе языкового моделирование лежит некоторый тезаурус (т.е. специальная терминология, полномерно охватывающий понятия, определения и термины специальной области знаний или сферы деятельности). Он формируется из набора входящих понятий причем этот набор должен быть фиксированным.

При реальном моделировании используется возможность исследования различных характеристик либо на реальном объекте целиком, либо на его части. Такие исследования могут проводиться как на объектах, работающих в обычном режиме, так и при организации специальных режимов для оценки интересующих исследователя характеристик.

Натурным моделированием называют проведение исследования на реальном объекте с последующей обработкой результатов на основе теории подобия. При функционировании объекта в соответствии с поставленной целью удается выявить закономерности протекания реального процесса. Такие разновидности натурного эксперимента, как производственный эксперимент и комплексные испытания обладают высокой степенью достоверности.

Научный эксперимент позволяет исследовать функционирование объекта в различных критических ситуациях. В ходе эксперимента вводятся новые факторы, возмущающие воздействия и т.д. Одна из разновидностей эксперимента – комплексные испытания – позволяют в следствие повторения испытания изделий выявить общие закономерности об их надежности, характеристиках качества и т.д. Наряду со специально организованными испытаниями возможна реализация натурного моделирования путем обобщения опыта, накопленного в ходе производственного процесса. В производственном эксперименте на базе теории подобия обрабатывают статистический материал по производственному процессу.

Физическое моделирование отличается от натурного тем, что исследование проводится на установках, которые сохраняют природу явлений и обладают физическим подобием. В процессе физического моделирования

исследуется поведение модели при заданных или искусственно создаваемых воздействиях внешней среды. Пример: Изучение устойчивости сложных конструкций, под воздействием сложных силовых нагрузок и т.д.

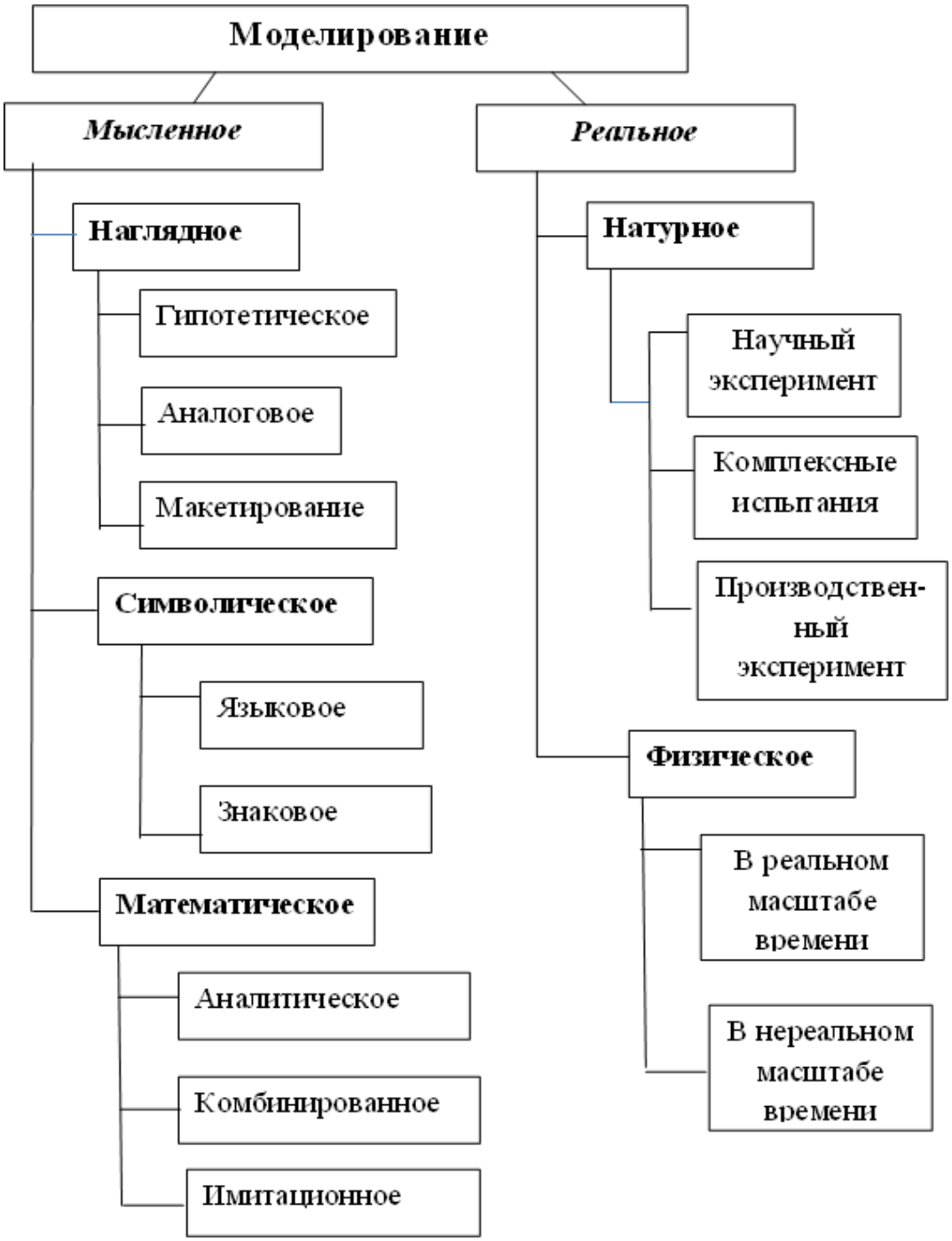


Рис. 1.2. Классификация моделей в зависимости от характера изучаемых объектов

1.4. Аналитическое моделирование

Аналитические методы позволяют получить характеристики системы как некоторые функции параметров ее функционирования. Таким образом,

аналитическая модель представляет собой систему уравнений и неравенств, при решении которой получают параметры, необходимые для оценки системы.

Достоинства аналитических методов – высокая точность вычислений;

Недостатки – при выводе формул необходимо принять некоторые допущения, упрощения.

Один из важных моментов построения математической модели — это целевое предназначение модели, т. е. перед началом моделирования необходимо определить решаемые задачи, для которых создается данная модель. Создание модели системы, описывающей все аспекты существования и развития системы, является малопродуктивной с точки зрения возможности ее использования, т. к. вряд ли возможно адекватно оценить все стороны состояния системы при ее построении. В этом случае модель получается чрезвычайно громоздкой и непригодной для каких-либо серьезных исследований. Модель всегда должна быть конкретной и нацеленной на решение поставленной задачи. Если ставится задача исследования ряда аспектов, то необходимо создавать несколько моделей, а не пытаться разрабатывать одну всеобъемлющую модель.

Для **аналитического** моделирования характерно то, что в основном моделируется только функциональный аспект системы. При этом глобальные уравнения системы, описывающие закон (алгоритм) ее функционирования, записываются в виде некоторых аналитических соотношений (алгебраических, интегродифференциальных, конечноразностных и т.д.) или логических условий. Аналитическая модель исследуется несколькими методами:

- аналитическим, когда стремятся получить в общем виде явные зависимости, связывающие искомые характеристики с начальными условиями, параметрами и переменными состояния системы;

- численным, когда, не умея решать уравнения в общем виде, стремятся получить числовые результаты при конкретных начальных данных;

- качественным, когда, не имея решения в явном виде, можно найти некоторые свойства решения (например, оценить устойчивость решения).

Помимо общих требований, предъявляемых к модели, для аналитических моделей характерны следующие принципы:

- Упрощение при сохранении существенных свойств системы. Модель должна быть в некоторых отношениях проще прототипа — в этом смысл моделирования. Чем сложнее рассматриваемая система, тем по возможности более упрощенным должно быть ее описание, умышленно утрирующее типичные и игнорирующее менее существенные свойства. Этот принцип может быть назван принципом абстрагирования от второстепенных деталей.

- Соответствие между требуемой точностью результатов моделирования и сложностью модели. Модели по своей природе всегда носят приближенный характер. Возникает вопрос, каким должно быть это приближение. С одной стороны, чтобы отразить все сколько-нибудь существенные свойства, модель необходимо детализировать. С другой стороны, строить модель,

приближающуюся по сложности к реальной системе, очевидно, не имеет смысла. Она не должна быть настолько сложной, чтобы нахождение решения оказалось слишком затруднительным. Компромисс между этими двумя требованиями достигается нередко путем проб и ошибок. Практическими рекомендациями по уменьшению сложности моделей являются:

- изменение числа переменных, достигаемое либо исключением несущественных переменных, либо их объединением. Процесс преобразования модели в модель с меньшим числом переменных и ограничений называют агрегированием. Например, все типы ЭВМ в модели гетерогенных сетей можно объединить в четыре типа — ПЭВМ, рабочие станции, большие ЭВМ (мейнфреймы), кластерные ЭВМ;
- изменение природы переменных параметров. Переменные параметры рассматриваются в качестве постоянных, дискретные — в качестве непрерывных и т.д. Так, условия распространения радиоволн в модели радиоканала для простоты можно принять постоянными;
- изменение функциональной зависимости между переменными. Нелинейная зависимость заменяется обычно линейной, дискретная функция распределения вероятностей — непрерывной;
- изменение ограничений (добавление, исключение или модификация). При снятии ограничений получается оптимистичное решение, при введении — пессимистичное. Варьируя ограничениями можно найти возможные граничные значения эффективности. Такой прием часто используется для нахождения предварительных оценок эффективности решений на этапе постановки задач;
- ограничение точности модели. Точность результатов модели не может быть выше точности исходных данных.

Пример. Моделирование одноканальной обслуживающей системы с отказами. На вход системы, состоящей из одного канала обслуживания, поступает поток заявок с интенсивностью λ. Если система свободна, то заявка принимается к обслуживанию, иначе – получает отказ. Интенсивность обслуживания - μ. Определить вероятности обслуживания и отказа.

1. **Упрощения**. Предположим, что поток поступающих заявок является простейшим. В этом случае он удовлетворяет трем свойствам:

- стационарность (интенсивность потока не меняется во времени);
- отсутствие последствия (независимость отдельных элементов потока друг от друга);
- ординарность (означает невозможность появления двух и более событий в бесконечно малый интервал времени; вероятность появления одного события прямопропорциональна величине временного интервала).

Выделим состояния системы:

0 – система свободна; 1 – система занята.

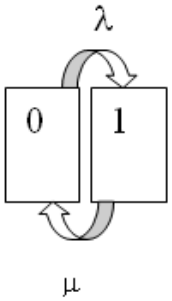


Рис. 1.3. Граф состояний одноканальной СМО с отказами

2. Составление математической модели

Вероятности пребывания системы в каждом из состояний могут быть найдёны с помощью следующей системы дифференциальных уравнений:

$$\begin{cases} \frac{dP_0(t)}{dt} = \mu P_1(t) - \lambda P_0(t) \\ P_0(t) + P_1(t) = 1 \end{cases} \quad (1.1)$$

В данном случае система (1.1) – эта математическая модель исследуемой системы. Решим ее аналитическим способом. Получим:

$$P_0(t) = \frac{\lambda}{\lambda+\mu} e^{-(\lambda+\mu)t} + \frac{\mu}{\lambda+\mu}; \quad (1.2)$$

$$P_1(t) = \frac{\lambda}{\lambda+\mu}\left(1 - e^{-(\lambda+\mu)t}\right). \quad (1.3)$$

Таким образом, получим следующие характеристики системы:

а) вероятность обслуживания

$$P_{обсл} = P_0^* = \frac{\lambda}{\lambda+\mu} e^{-(\lambda+\mu)t} + \frac{\mu}{\lambda+\mu}; \quad (1.4)$$

б) вероятность отказа

$$P_{отк} = P_1(t) = \frac{\lambda}{\lambda+\mu}\left(1 - e^{-(\lambda+\mu)t}\right) \quad (1.5)$$

3. Исследование сложности модели

В данном случае математическая модель достаточно проста. Однако, очевидно, что при добавлении числа каналов обслуживания, очереди и т.д. ее сложность будет существенно возрастать. Для того, чтобы уменьшить сложность модели, проанализируем полученное выражение. Очевидно, что при возрастании времени слагаемое, содержащее экспоненту, будет близко нулю (график экспоненты приведен ниже).

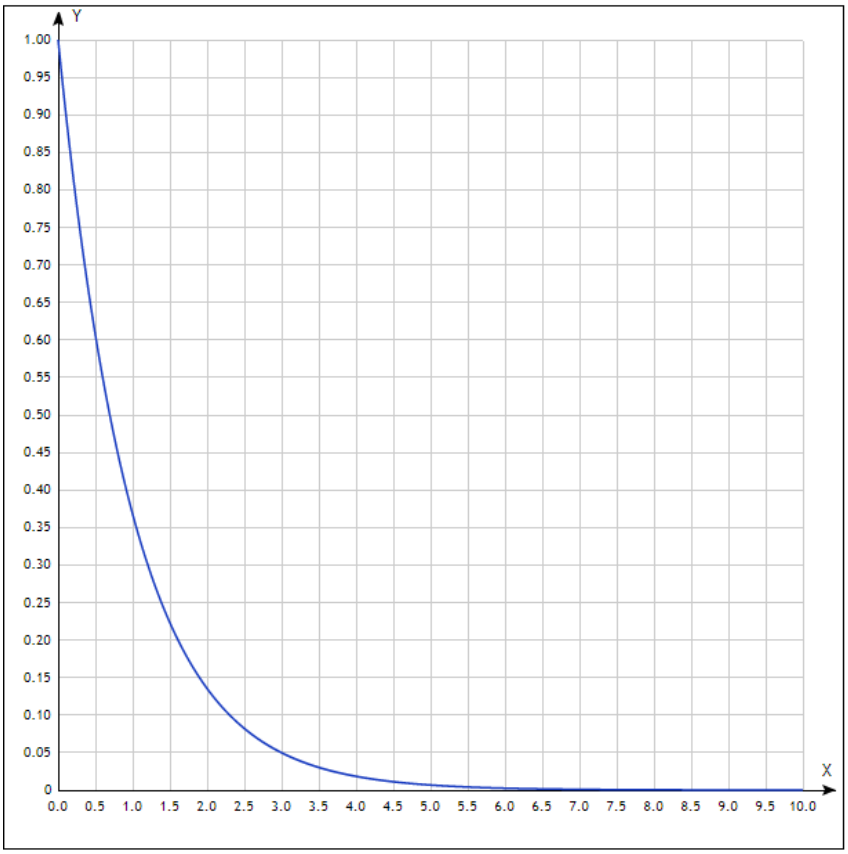


Рис. 1.4. График экспоненты

В связи с этим, можно упростить модель, рассмотрев ее функционирование в так называемом стационарном или устоявшемся режиме (т.е. по прошествии некоторого времени). В этом режиме вероятности пребывания системы в состоянии 0 и 1 уже не будут зависеть от времени. Таким образом, воспользуемся приемом «изменение природы переменных параметров», заменяя переменные параметры (вероятности) постоянными. В этом случае получим более простой вариант математической модели:

$$\begin{cases} \mu P_1^* - \lambda P_0^* = 0 \\ P_1^* + P_0^* = 1 \end{cases} \quad (1.6)$$

Очевидно, что при неограниченном возрастании времени система будем иметь стационарный режим, причём вероятности пребывания в каждом из состояний будут определяться формулами:

$$P_0^* = \frac{\mu}{\lambda + \mu}; \quad (1.7)$$

$$P_1^* = \frac{\lambda}{\lambda + \mu}. \quad (1.8)$$

Вероятность обслуживания такой системы будет определяться формулой

$$P_{обсл} = 1 - P_1^* = P_0^* = \frac{\mu}{\lambda + \mu}. \quad (1.9)$$

Исследования показали, что формула (1.4), которая более точно описывает вероятность отказа, достаточно быстро стремится к значению,

описываемому с помощью формулы (1.9). Следовательно, в данном случае целесообразным представляется принять допущения о стационарном режиме системы и исследовать более простой вариант модели.

Этапы построения аналитической модели

Сущность построения аналитической модели состоит в том, что реальная система упрощается, схематизируется и описывается с помощью того или иного математического аппарата. Можно выделить следующие основные этапы построения моделей.

1. **Содержательное описание моделируемого объекта.** Объекты моделирования описываются с позиций системного подхода. Исходя из цели исследования устанавливаются совокупность элементов, взаимосвязи между элементами, возможные состояния каждого элемента, существенные характеристики состояний и отношения между ними. Такое предварительное, приближенное представление системы называют концептуальной моделью. На этом этапе моделирования широко применяются качественные методы описания систем, знаковые и языковые модели. В дальнейшем полученная схема уточняется и дополняется в соответствии с тем уровнем детализации, который определяется (стратифицируется) постановкой задачи. Действия, которые позволяют представить модель в виде совокупности частей (подсистем, элементов), называют *декомпозицией* системы. Составные части модели должны обеспечивать сохранение целостности системы, с одной стороны, а с другой – достижение поставленных целей моделирования. Процесс построения концептуальной схемы системы завершается структуризацией (указанием и общим описанием связей между выделенными элементами системы), а также укрупненным описанием динамики функционирования системы и ее возможных состояний. От того, как будет построена концептуальная схема имитационной модели, зависит результат исследования.

2. **Формализация операций.** Формализация сводится в общих чертах к следующему. На основе содержательного описания определяется исходное множество характеристик системы. Для выделения существенных характеристик необходим хотя бы приближенный анализ каждой из них. При проведении анализа опираются на постановку задачи и понимание природы исследуемой системы. После исключения несущественных характеристик выделяют управляемые и неуправляемые параметры и производят символизацию. Затем определяется система ограничений на значения управляемых параметров. Если ограничения не носят принципиальный характер, то ими пренебрегают.

Дальнейшие действия связаны с формированием целевой функции модели. В соответствии с известными положениями выбираются показатели исхода операции и определяется примерный вид функции полезности на исходах. Если функция полезности близка к пороговой (или монотонной), то оценка эффективности решений возможна непосредственно по показателям исхода операции. В этом случае необходимо выбрать способ свертки

показателей (способ перехода от множества показателей к одному обобщенному показателю) и произвести саму свертку. По свертке показателей формируются критерий эффективности и целевая функция.

Если при качественном анализе вида функции полезности окажется, что ее нельзя считать пороговой (монотонной), прямая оценка эффективности решений через показатели исхода операции неправомочна. Необходимо определять функцию полезности и уже на ее основе вести формирование критерия эффективности и целевой функции.

В целом замена содержательного описания формальным — это итеративный процесс.

3. **Проверка адекватности модели.** Требование адекватности находится в противоречии с требованием простоты, и это нужно учитывать при проверке модели на адекватность. Исходный вариант модели предварительно проверяется по следующим основным аспектам:

- Все ли существенные параметры включены в модель?
- Нет ли в модели несущественных параметров?
- Правильно ли отражены функциональные связи между параметрами?
- Правильно ли определены ограничения на значения параметров?

Для проверки рекомендуется привлекать специалистов, которые не принимали участия в разработке модели. Они могут более объективно рассмотреть модель и заметить ее слабые стороны, чем ее разработчики. Такая предварительная проверка модели позволяет выявить грубые ошибки. После этого приступают к реализации модели и проведению исследований. Полученные результаты моделирования подвергаются анализу на соответствие известным свойствам исследуемого объекта. Для установления соответствия создаваемой модели оригиналу используются следующие пути:

- сравнение результатов моделирования с отдельными экспериментальными результатами, полученными при одинаковых условиях;
- использование других близких моделей;
- сопоставление структуры и функционирования модели с прототипом.

Главным путем проверки адекватности модели исследуемому объекту выступает практика. Однако она требует накопления статистики, которая далеко не всегда бывает достаточной для получения надежных данных. Для многих моделей первые два приемлемы в меньшей степени. В этом случае остается один путь: заключение о подобии модели и прототипа делать на основе сопоставления их структур и реализуемых функций. Такие заключения не носят формального характера, поскольку основываются на опыте и интуиции исследователя.

По результатам проверки модели на адекватность принимается решение о возможности ее практического использования или о проведении корректировки.

4. **Корректировка модели.** При корректировке модели могут уточняться существенные параметры, ограничения на значения управляемых параметров, показатели исхода операции, связи показателей исхода операции с существенными параметрами, критерий эффективности. После внесения изменений в модель вновь выполняется оценка адекватности.

5. **Оптимизация модели.** Сущность оптимизации моделей состоит в их упрощении при заданном уровне адекватности. Основными показателями, по которым возможна оптимизация модели, выступают время и затраты средств для проведения исследований на ней. В основе оптимизации лежит возможность преобразования моделей из одной формы в другую. Преобразование может выполняться либо с использованием математических методов, либо эвристическим путем.

1.5.Особенности имитационного моделирования

1.5.1. Понятие имитационного моделирования

Когда говорят об имитационном моделировании, то предполагают использование некоторой структурной схемы, математического обеспечения, а также вычислительного устройства для создания соответствующей программы. Отсюда следует определение имитационного моделирования.

Определение 2.1. Имитационной моделью называют абстрактную динамическую модель, реализованную, как правило, на ЭВМ, и воспроизводящую в рамках установленных ограничений поведение оригинала в хронологическом порядке.

Имитационные модели не способны формировать решение в таком виде, как в аналитических моделях, а служат лишь средством для анализа поведения системы (оригинала) в условиях, которые определяются экспериментатором. По сути, имитационное моделирование является экспериментальной и прикладной методологией, которая имеет следующие цели:

1) описать поведение системы;
2) построить теории и гипотезы, которые могут объяснить наблюдаемое поведение;
3) использовать данные теории для предсказания будущего поведения системы.

1.5.2. Преимущества и недостатки имитационного моделирования

Имитационное моделирование целесообразно применять при наличии условий:

1) Не существует законченной математической постановки данной задачи (например, модель многофазных, многоканальных систем массового обслуживания).
2) Аналитические методы имеются, но очень сложны и трудоемки, а имитационное моделирование дает более простой способ решения.
3) Аналитические решения имеются, но их реализация невозможна из-за недостаточной подготовки имеющегося персонала. В этом случае

сопоставляются затраты на работу с имитационным моделированием и затраты на приглашение специалистов со стороны.

4) Кроме оценки определенных параметров необходимо осуществлять наблюдение за ходом процесса в течение определенного периода.

5) Имитационное моделирование может быть единственно возможным вследствие трудности постановки эксперимента и наблюдения явлений в реальных условиях (наблюдение за поведением космических кораблей).

6) Может понадобиться сжатие шкалы времени (как замедление, так и ускорение; например, исследование проблемы развития городов).

Вышеперечисленные условия можно определять как значимые преимущества применения имитационных моделей. Также можно заметить, что имитационное моделирование являются непревзойденным средством создания средств обучения в виде тренажеров, симуляторов и т.д. С помощью имитационного моделирования можно разыграть реальные процессы и ситуации, которые помогут исследователю понять и прочувствовать проблему, что стимулирует процесс поиска нововведений. Благодаря этому, порядка 30% из всех используемых на практике моделей являются имитационными моделями.

Вместе с тем имитационное моделирование обладает рядом недостатков.

Во-первых, имитационное моделирование представляет собой весьма дорогостоящий процесс, требующий существенных затрат временных ресурсов и привлечения высококвалифицированных специалистов.

Во-вторых, в процессе моделирования не представляется возможным получить точный результат. При этом оценка точности может быть выполнена путем анализа чувствительности модели к изменению определенных параметров.

В-третьих, имитационное моделирование в действительности не отражает полного положения вещей. Данный факт необходимо учитывать при анализе исследуемого объекта (процесса).

1.5.3. Процесс имитационного моделирования

Построение имитационной модели, так же, как и любое исследование, требует проведения работ по следующим этапам.

1) Определение границ модели.
2) Разработка концептуальной модели.
3) Подготовка исходных данных.
4) Создание концептуальной модели в виде диаграммы.
5) Трансляция модели.
6) Оценка адекватности модели.
7) Планирование машинных экспериментов.
   a) Стратегическое планирование.
   b) Тактическое планирование.
8) Моделирование – проведение эксперимента.
9) Анализ (интерпретация) результатов.

10) Документирование и реализация.

Границы системы определяются таким образом, чтобы охватить те компоненты, взаимодействие которых определяет важные стороны поведения системы. При этом система должна быть способна сама генерировать любую ситуацию, любые затруднения, которые, возможно потребуется проанализировать.

Разработка концептуальной схемы объекта (системы) является одним из самых важных этапов исследования. На этом этапе осуществляется *формализация системы*, т.е. переход от реального объекта к некоторой логической схеме (абстракции). Такая формализация начинается со словесного описания реальности в системе принятых терминов и формальных понятий. Здесь «приводятся сведения о природе и параметрах (характеристиках) элементарных явлений исследуемой системы, о виде и степени взаимодействия между ними, о месте и значении каждого элементарного явления в общем процессе функционирования системы». Завершается формализация построением общей схемы процессов, подлежащих исследованию.

В дальнейшем полученная схема уточняется и дополняется в соответствии с тем уровнем детализации, который определяется (стратифицируется) постановкой задачи. Действия, которые позволяют представить модель в виде совокупности частей (подсистем, элементов), называют *декомпозицией* системы. Составные части модели должны обеспечивать сохранение целостности системы, с одной стороны, а с другой – достижение поставленных целей моделирования.

Процесс построения концептуальной схемы системы завершается структуризацией (указанием и общим описанием связей между выделенными элементами системы), а также укрупненным описанием динамики функционирования системы и ее возможных состояний. От того, как будет построена концептуальная схема имитационной модели, зависит результат исследования.

Следующий, не менее важный этап имитационного моделирования – **подготовка исходных данных**. В некоторых случаях он проходит параллельно с построением концептуальной схемы. Фактически на данном этапе формируется информационное пространство системы. Здесь выявляются количественные характеристики (параметры) функционирования системы и ее элементов, численные значения которых составят исходные данные для моделирования.

Когда подготовлены исходные данные и концептуальная схема модели, последняя **оформляется в виде диаграммы**, состоящей из стандартных блоков. Это технический этап, благодаря которому схема модели становится доступной для понимания широкому кругу специалистов, владеющих соответствующей методикой. Как правило, диаграмма оформляется с помощью специализированных прикладных программных средств, таких как BP Win, Microsoft Project, Visio.

На этапе **трансляции** модели осуществляется преобразование диаграммы модели в отдельную компьютерную программу или сценарий специализированной системы моделирования. В современных версиях таких систем такой этап выполняется автоматически, благодаря наличию визуальных средств построения моделей.

Оценка адекватности полученной модели осуществляется путем ее экспертизы и проигрывания на тестовых данных. На данном этапе модель проверяется на корректность, т.е. на соответствие реальному объекту в рамках поставленной задачи (границ системы). Когда модель не адекватна, то она подвергается исправлениям и корректировкам до приемлемого уровня степени уверенности, с которой можно судить о корректности выводов, касающихся реальной системы.

Добившись адекватности модели, исследователи **осуществляют стратегическое и тактическое планирование эксперимента.** Когда говорят о стратегическом планировании, то предусматривают схему получения желаемых результатов с помощью имитационной модели. На тактическом уровне планируют способ проведения каждой серии испытаний, предусмотренных планом эксперимента.

На этапе экспериментирования осуществляется проигрывание запланированных сценариев с целью получения желаемого результата.

После того, как получены результаты моделирования, наступает важный этап исследования – **интерпретация результатов**. По полученным выходным данным эксперимента строятся выводы о поведении исследуемой системы. При этом очень важно не пропустить эффект двоякого прочтения одних и тех же результатов. В этом случае следует дорабатывать модель.

На этапе интерпретации результатов также дается заключение о полезности или бесполезности модели. В случае полезности модели осуществляется ее **реализация**, т.е. практическое использование. На данном этапе выполняются вспомогательные действия, такие как регистрация хода осуществления исследования и его результатов, документирование процесса создания и использования модели.

1.6. Дискретно-событийное моделирование

1.6.1. Особенности дискретно-событийного моделирования

Дискретно-событийное моделирование используется для построения модели, отражающей развитие системы во времени, когда состояния переменных меняются мгновенно в конкретные моменты времени. В такие моменты времени происходят события, при этом событие определяется как мгновенное возникновение, которое может изменить состояние системы. Хотя теоретически дискретно-событийное моделирование можно осуществлять с помощью вычислений вручную, количество данных, которые должны

сохраняться и обрабатываться при моделировании большинства реальных систем, диктует необходимость применения вычислительных машин.

Пример 1. Рассмотрим систему с одним устройством обслуживания. Необходимо подсчитать ожидаемую среднюю задержку требований в очереди. При этом задержка требования в очереди равна времени, прошедшему с момента его появления в системе до начала его обслуживания.

Для того чтобы рассчитать среднюю задержку в дискретно-событийной имитационной модели, определяются такие переменные состояния, как состояние устройства обслуживания (занято или свободно), число требований в очереди (если таковые имеются) и время поступления каждого требования, ожидающего своей очереди. При поступлении требования должно быть определено состояние устройства обслуживания: может ли требование быть обслужено немедленно или его необходимо поместить в конец очереди.

После завершения обслуживания требования, исходя из числа требований в очереди, определяется, будет устройство обслуживания свободно или же начнет обслуживание первого требования в очереди. Чтобы вычислить задержку требования в очереди, необходимо установить время его поступления, так как задержка равна времени начала обслуживания требования (которое будет известно) минус время его поступления. В этой системе есть два типа событий: поступление требования и завершение обслуживания требования, приводящее к его уходу. Поступление требования является событием, поскольку оно вызывает изменение состояния устройства обслуживания (переменной состояния) со свободного на занятое или увеличение числа требований в очереди (переменной состояния) на единицу. Соответственно уход требования также является событием, так как вызывает изменение состояния устройства обслуживания с занятого на свободное или уменьшение числа требований в очереди на единицу.

В приведенном выше примере оба типа событий действительно меняют состояние системы, тогда как в некоторых дискретно-событийных моделях события применяются для задач, не вызывающих таких изменений. Так, событие может использоваться, чтобы задать окончание имитационного прогона на определенное время или вычислить результаты работы системы в определенный момент времени, при этом оно не будет вызывать действительного изменения состояния системы.

1.6.2. Механизмы продвижения времени

Динамическая природа дискретно-событийных имитационных моделей требует, чтобы мы следили за текущим значением имитационного времени по мере функционирования имитационной модели. Нам необходим также механизм для продвижения имитационного времени от одного значения к другому. В имитационной модели переменная, обеспечивающая текущее значение модельного времени, называется часами модельного времени. При создании модели на таких универсальных языках, как FORTRAN или C, единица времени для часов модельного времени никогда не устанавливается

явно. Подразумевается, что оно будет указываться в тех же единицах, что и входные параметры. К тому же модельное время и время, необходимое для прогона имитационной модели на компьютере, как правило, невозможно соотнести.

Существует два основных подхода к продвижению модельного времени: продвижение времени с постоянным шагом и продвижение времени от события к событию.

Принцип Δt

Принцип состоит в том, что алгоритмом моделирования имитируется движение, то есть изменение состояния системы, в фиксированные моменты времени: $t, t + \Delta t, t + 2\Delta t, t + 3\Delta t, \ldots$

Для этого заводится счетчик времени (часы), который на каждом цикле увеличивает свое значение t на величину шага во времени Δt, начиная с нуля (начало моделирования). Таким образом, изменения системы отслеживаются такт за тактом в заданные моменты: $t, t + \Delta t, t + 2\Delta t, t + 3\Delta t, \ldots$

Особенности реализации принципа Δt

Это наиболее универсальный из рассматриваемых принципов, так как применяется для очень широкого класса систем. Он же является наиболее простым в реализации, поскольку принцип Δt совпадает с пониманием человека о времени, как о последовательном явлении, текущем с постоянным темпом.

Однако это самый неэкономичный принцип, поскольку вся система анализируется моделирующим алгоритмом на каждом такте, даже если в ней не происходит никаких изменений.

Другой недостаток состоит в том, что времена событий округляются до величины Δt, что ведет к погрешностям в определении переменных, характеризующих систему.

Принцип особых состояний (принцип δz)

При рассмотрении процессов функционирования некоторых систем можно обнаружить, что для них характерны два типа состояний:

1) Особые, присущие процессу в некоторые моменты времени;
2) Неособые, в которых процесс находится все остальное время.

Особые состояния — это такие состояния в изолированные моменты времени, в которых характеристики системы изменяются скачкообразно. Для изменения состояния системы нужна определенная причина, (например, приход очередного входного сигнала). С точки зрения моделирования интерес представляет именно изменение характеристик системы, то есть принцип требует отслеживать моменты перехода системы из одного особого состояния в другое.

При использовании продвижения времени от события к событию часы модельного времени в исходном состоянии устанавливаются в 0 и определяется время возникновения будущих событий. После этого часы модельного времени переходят на время возникновения ближайшего события, и в этот момент обновляется состояние системы с учетом произошедшего события, а также

сведения о времени возникновения будущих событий. Затем часы модельного времени продвигаются ко времени возникновения следующего (нового) ближайшего события, обновляется состояние системы и определяется время будущих событий, и т. д.

Процесс продвижения модельного времени от времени возникновения одного события ко времени возникновения другого продолжается до тех пор, пока не будет выполнено какое-либо условие останова, указанное заранее. Поскольку в дискретно-событийной имитационной модели все изменения происходят только во время возникновения событий, периоды бездействия системы просто пропускаются, и часы переводятся со времени возникновения одного события на время возникновения другого. Следует отметить, что длительность интервала продвижения модельного времени от одного события к другому может быть различной.

2. ОБЗОР ОСНОВНЫХ БИБЛИОТЕК Anylogic

2.1. Назначение и возможности среды Anylogic

Пакет Anylogic – это профессиональный инструмент нового поколения, предназначенный для разработки и исследования имитационных моделей. При разработке модели в среде Anylogic можно использовать концепции и средства из нескольких классических областей имитационного моделирования: дискретно-событийного моделирования, системной динамики и агентного моделирования. Кроме того, Anylogic обеспечивает возможность интеграции различных подходов для моделирования сложных обслуживающих систем. Для реализации данного приложения был использован язык Java. Данный язык можно использовать обычным пользователям Anylogic при описании действий, возникающих при входе или выходе из некоторых блоков, при выполнении или невыполнении условий и т.п.

Среда моделирования Anylogic имеет следующий вид (*рис. 2.1*).

Визуально приложение состоит из трех составляющих. Левая часть представляет собой палитру, включающую множество специализированных библиотек, каждая из которых предназначена решения какой-либо специфической задачи моделирования (например, моделирование дорожного движения, моделирование потоков и т.д.). Каждая библиотека содержит ряд блоков, каждый из которых выполняет Центральная часть – это рабочая область, на которую помещаются блоки из библиотеки. Блоки необходимо помещать в прямоугольник, ограниченный сверху и слева осями координат, снизу и справа – прямыми линиями. Именно данный прямоугольник будет отображаться во время моделирования. Если какой-либо из блоков помещен в модель вне этого пространства, то в процессе моделирования он виден не будет. Правая часть окна содержит информацию о свойствах активного блока (выделенного или того, который в данный момент был помещен в рабочую область).

Рассмотрим процесс построения модели в anylogic. На первом этапе на рабочую область необходимо последовательно поместить все блоки, совокупность которых будет образовывать логику модели. Каждый блок (кроме Source (источник заявок) и Sink (блок для уничтожения заявок)) имеет входные и выходные порты. Блок Source имеет лишь входной порт; Sink – только выходной. При этом выходной порт предыдущего блока автоматически соединяется с входным портом текущего блока. Если требуется изменить соединение блоков, то можно это сделать самостоятельно. Для этого сначала необходимо дважды щелкнуть на выходной порт первого из соединяемых блоков, а затем провести линию к входному порту, после чего щелкнуть мышью.

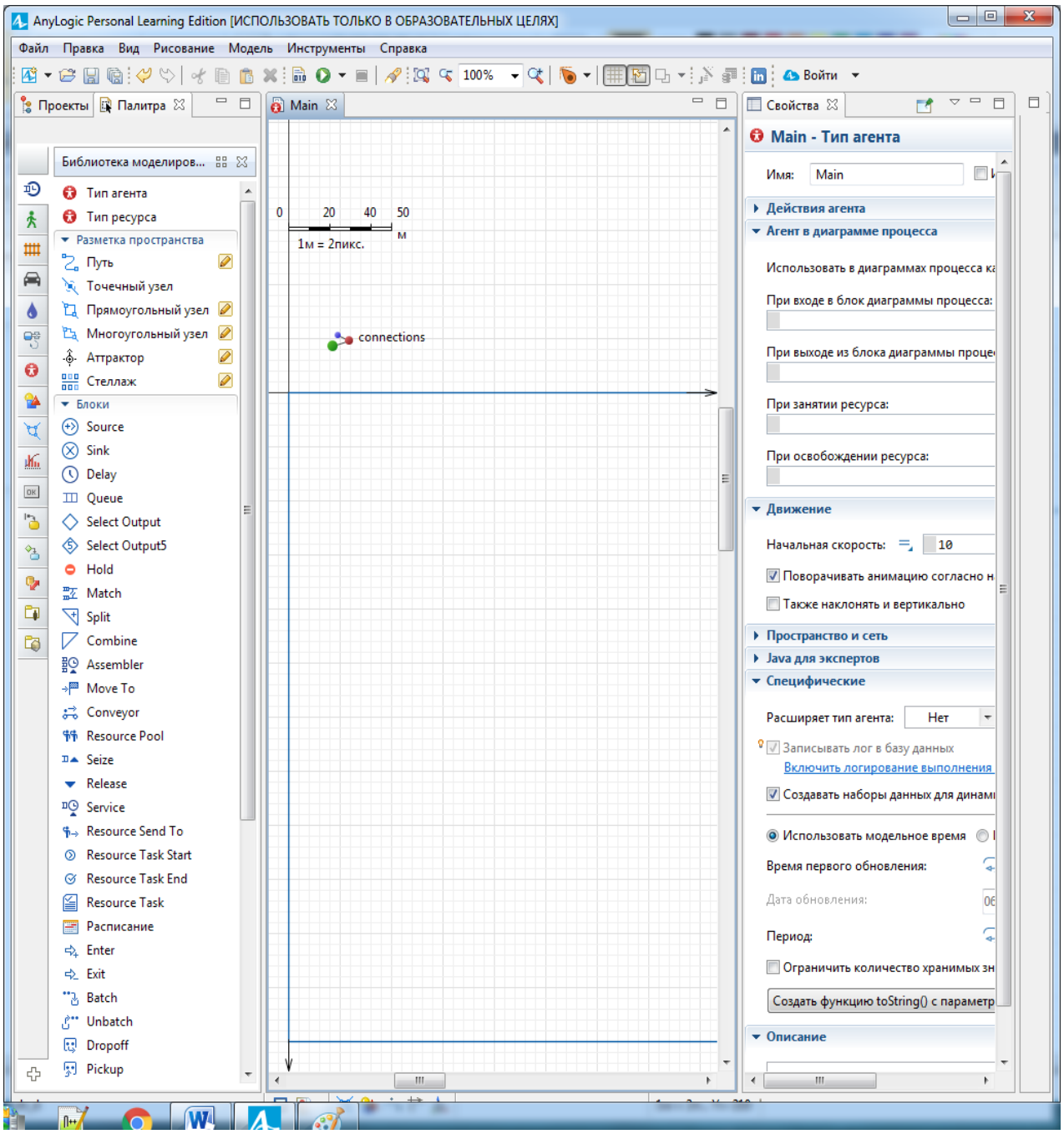


Рис. 2.1. Основное окно среды Anylogic

Можно также сделать связь между объектами в виде ломаной любого вида. Для этого сначала необходимо дважды щелкнуть по выходному порту первого блока, а далее везде, где требуется изменить направление линии, также осуществлять щелчок мыши. Пример с выделенными областями щелчка мышью представлен на *рис. 2.2*.

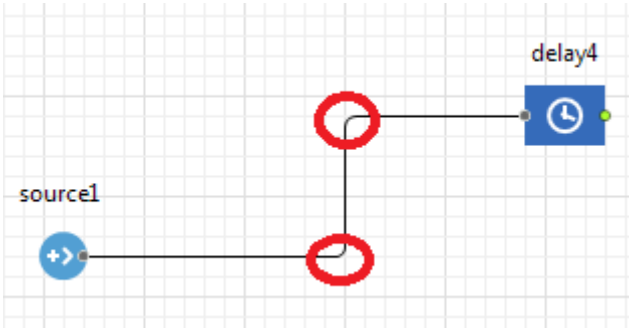


Рис. 2.2. Прорисовка связей между объектами

После соединения блоков необходимо настроить их свойства. При нанесении любого блока на рабочую область автоматически открывается окно, содержащее все свойства этого блока.

После задания свойств модели, она запускается соответствующей кнопкой быстрого доступа или нажатием клавиши F5.

2.2 Библиотека моделирования процессов и простейшая дискретно-событийная модель

Для исследования возможностей среды AnyLogic проанализируем основные библиотеки этого программного продукта. Как было отмечено ранее, все эти библиотеки располагаются в левой части рабочей области приложения. Первой из библиотек представлена библиотека моделирования процессов. Данная библиотека представлена на *рис. 2.3*.

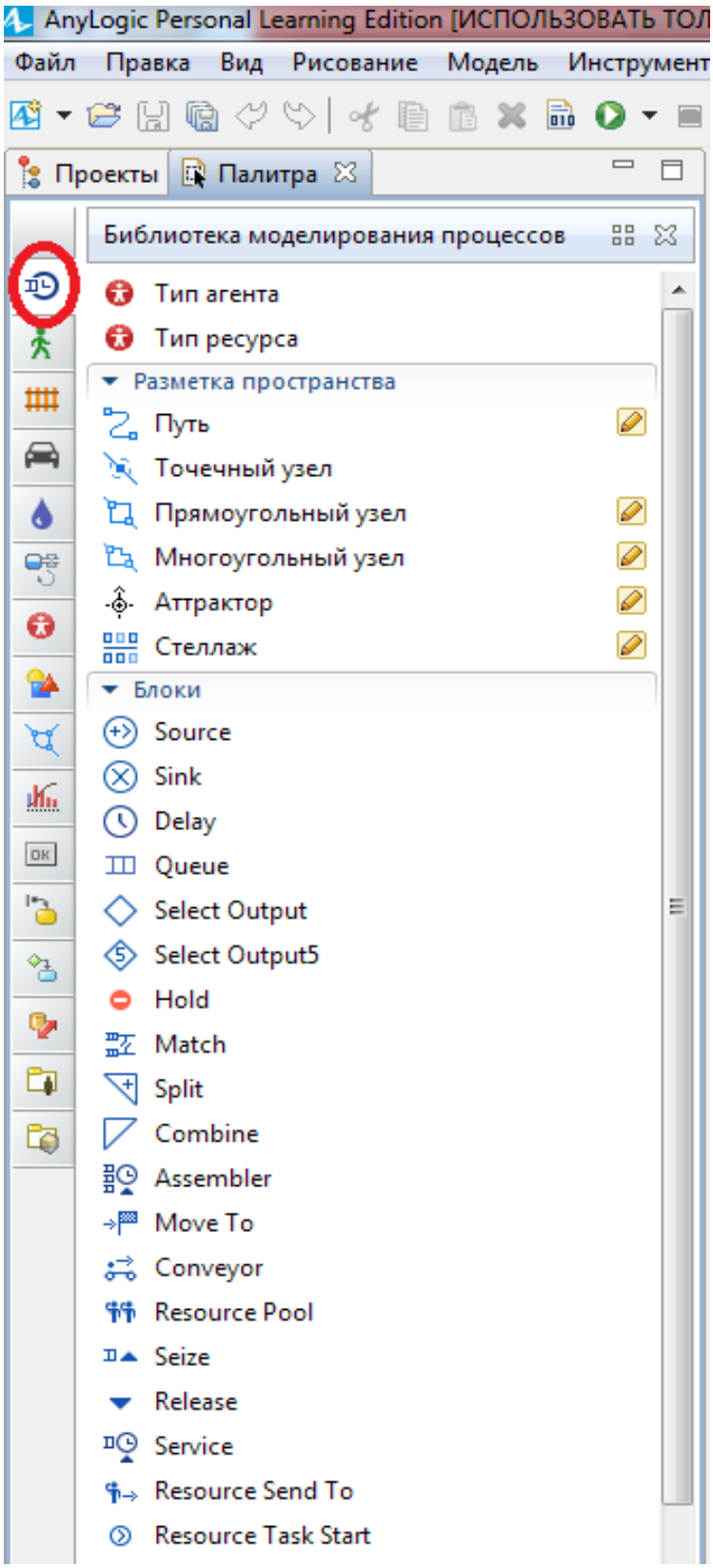


Рис. 2.3. Библиотека моделирования процессов

Данная библиотека содержит множество объектов, позволяющих смоделировать дискретно-событийную модель.

Дискретно-событийное моделирование используется для построения модели, отражающей развитие системы во времени, когда состояния

переменных меняются мгновенно в конкретные моменты времени. Такой вид моделирования предполагает представление исследуемой модели в виде некоторого процесса, т.е. последовательности операций, выполняемых с заявками.

Простейшая обслуживающая система, для которой целесообразно использовать дискретно-событийный подход, включает в себя:

- входящий поток заявок;
- обслуживающее устройство (одно или несколько);
- поток обслуженных заявок (и/или поток заявок, получивших отказ).

К основным блокам библиотеки моделирования процессов следует отнести:

- основные блоки для работы с заявками (Source – источник заявок; Sink – позволяет удалить заявку из системы; Delay – задержка – имитация обслуживания);
- блоки, изменяющие маршруты заявок (SelectOutput и SelectOutput5);
- блоки для работы с группами заявок (Match – синхронизация двух потоков; Split – создание копий заявок; Assemble – объединение заданного количества заявок (возможно, при определенных условиях) и т.д.);
- блоки для работы с ресурсами (ResourcePool – задание ресурсов; Seize – занять ресурсы; Release – освободить ресурсы; ResourceTask – задание задач для ресурсов).

Рассмотрим более подробно первую категорию блоков, связанные с имитацией простейших обслуживающих систем..

Поток заявок создается с помощью блока **Source** (источник). Данный блок предоставляет возможность создавать заявки через определенные моменты времени. Данные моменты могут быть постоянными или случайными, могут задаваться пользователем вручную или считываться из базы данных. После помещения компонента на форму, становятся доступными следующие его свойства (*рис. 2.4*).

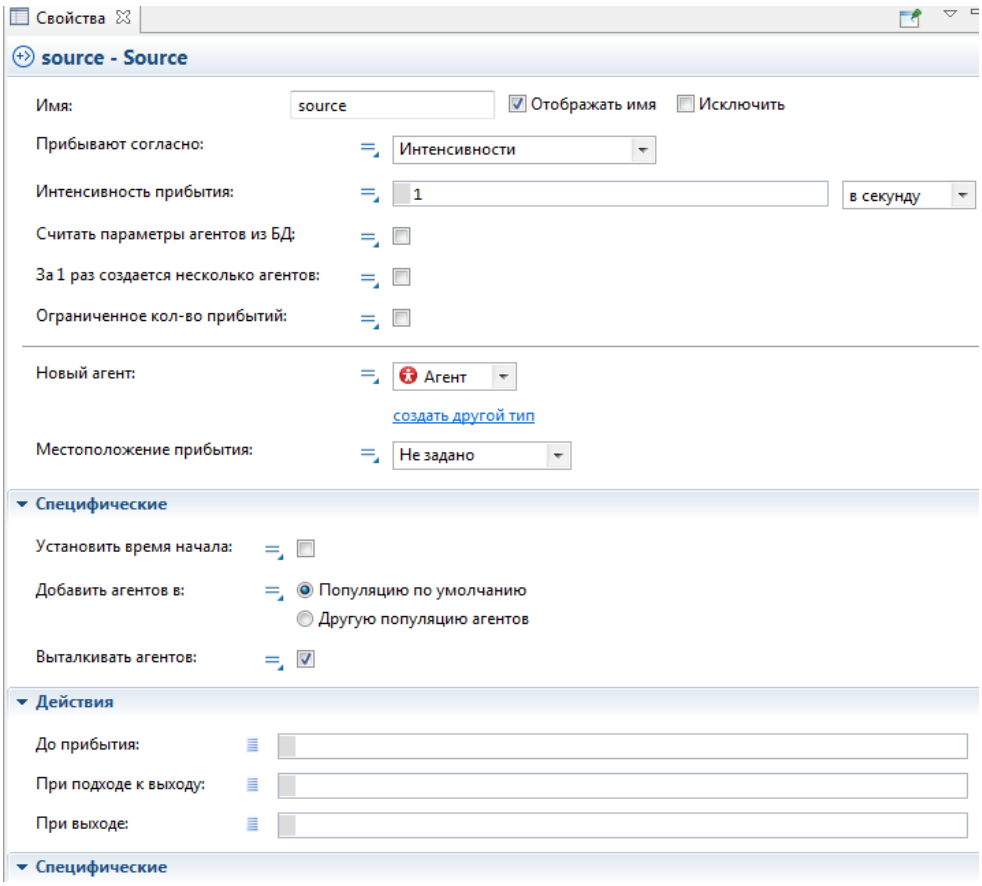


Рис. 2.4. Свойства блока Source

Основным настраиваемым свойством является свойство «Прибывают согласно:». В данном случае предусматривается несколько вариантов:

- интенсивности (в этом случае следующее поле будет представлять собой интенсивность, которая обязательно должна содержать некоторое числовое значение или случайную функцию, как показано на рис. 2.1);
- времени между прибытиями (в этом случае в следующем поле задается время (константа или функция) между двумя последовательно поступающими заявками);
- расписанию прибытий из БД (в этом случае следующим полем идет таблица БД, из которой необходимо считать значения);
- расписание интенсивностей (предполагается, что интенсивность поступления заявок меняется, и данные изменения отражены с помощью блока «Расписание»);
- расписанию прибытий (отличается от предыдущего случая тем, что в качестве исходных данных берется не интенсивность, а время между двумя последовательно поступившими в систему заявками);
- вызовам функции inject (данный вид поступления заявок используется в том случае, когда источник необходимо «запустить» вручную, например, при возникновении какого-либо события).

Как видно из *рис. 2.4*, за один раз можно создать как одну, так и несколько заявок (в последнем случае необходимо поставить галочку в поле «за

1 раз создается несколько агентов» и в появившемся поле ввести значение агентов).

По умолчанию заявки из данного блока будут поступать до окончания моделирования. Однако, можно ограничить количество поступлений с помощью галочки в поле «Ограниченное количество прибытий». В этом случае в появившемся поле необходимо ввести число заявок, которое должен создать этот блок.

Блок Delay предназначен для имитации процесса обслуживания путем задержки заявок. Для настройки доступны следующие его свойства (*рис. 2.5*).

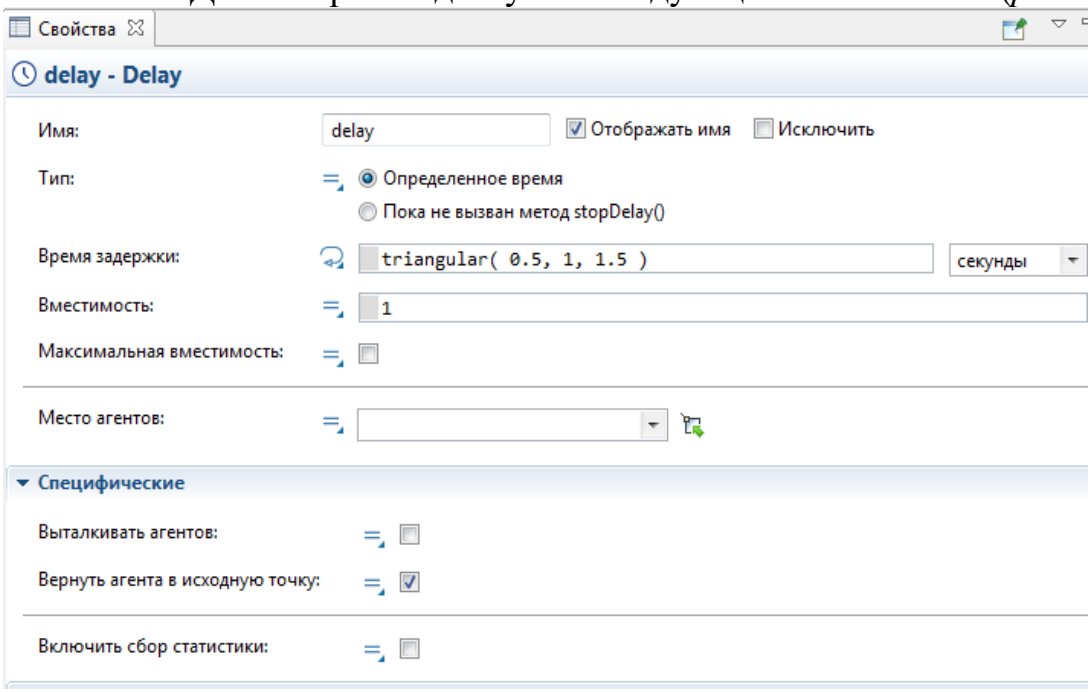


Рис. 2.5. Свойства блока Delay

Основным свойством блока является время задержки. Это может быть определенное время (как показано на рис. 2.5) или пока не будет вызван метод StopDelay, который останавливает процесс задержки.

Обслуживающее устройство состоит из заданного количества идентичных каналов обслуживания (например, 3 кассы магазина и т.д.). Режим функционирования любого устройства следующий: если в момент поступления заявки хотя бы один канал обслуживания свободен, то заявка принимается к обслуживанию, в противном случае - заявка ждет перед устройством момента, когда оно освободится, а потом занимает освободившийся канал обслуживания. Таким образом, важной характеристикой является количество каналов обслуживания. В блоке Delay это свойство задается с помощью поля «Вместимость». С помощью свойства «место агентов» можно задать фигуру разметки (например, некоторый узел или путь), где будут располагаться заявки в момент задержки блоком Delay.

Булевское свойство «Выталкивать агентов» позволяет реализовать некоторый сценарий, согласно которому заявки будут покидать данный блок

(если опция выбрана, то заявки по окончании задержки будут вытолкнуты в следующий блок).

Если выбрана опция «Вернуть агента в исходную точку», то после окончания обслуживания заявки возвращаются в то место (узел или путь), где она находились до задержки.

В случае, если количество заявок превышает вместимость, указанную в блоке Delay, возникает ошибка. Для того, чтобы ее избежать, необходимо использовать очередь. Она моделируется с помощью блока **queue**.

Объект Queue моделирует очередь заявок, ожидающих приема объектами, следующими за ним (например, блоком Delay). Очередь можно ограничить количеством или временем ожидания.

Свойства данного блока представлены на *рис. 2.6*.

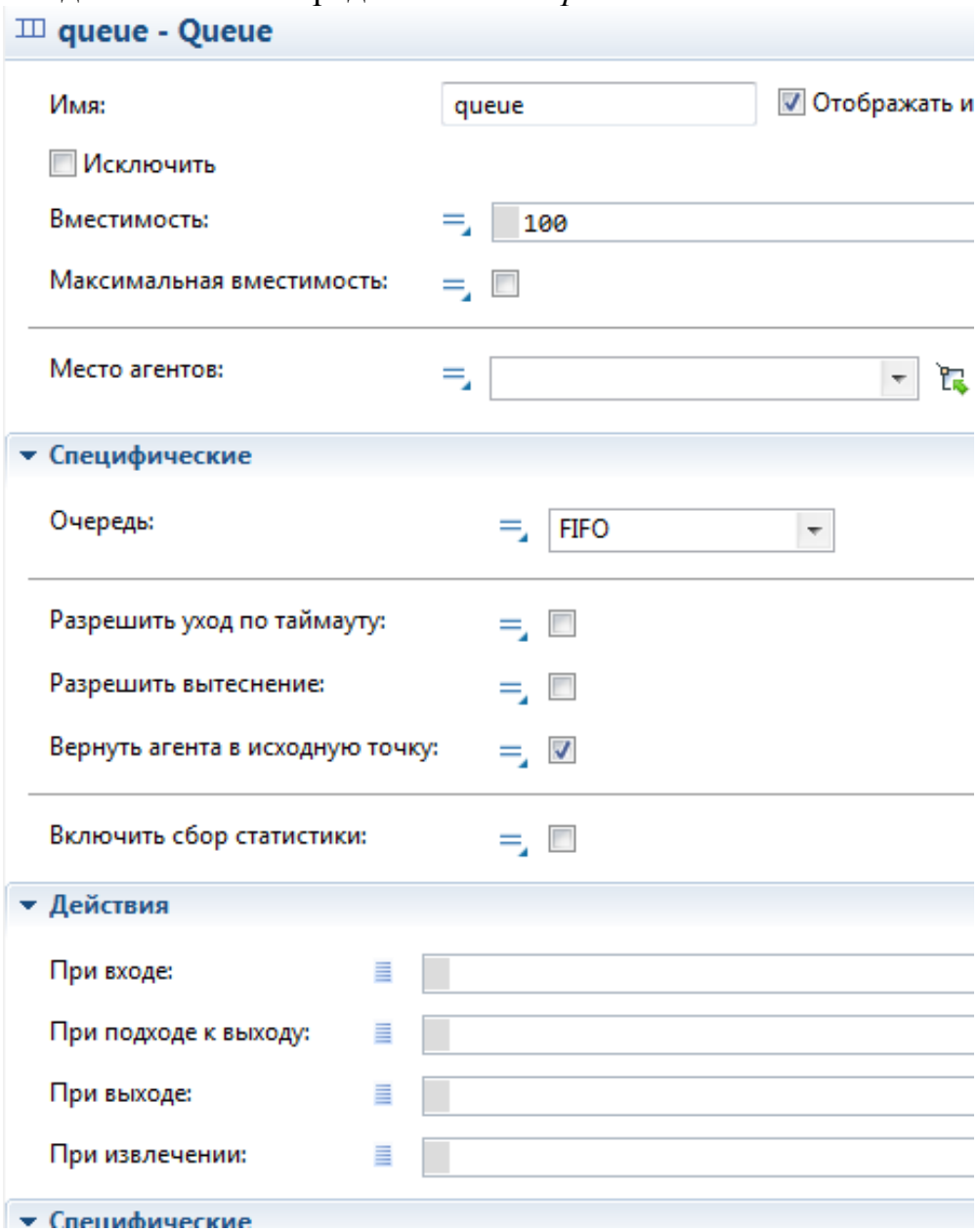


Рис. 2.6. Свойства блока queue

Заявка может покинуть объект Queue четырьмя различными способами:

1. Обычным способом через порт Out, когда объект, следующий за объектом Queue, готов принять заявку.
2. Через порт TimeOut, если заявка проведет в очереди заданное количество времени. Для моделирования очередей с ограничением на время ожидания в свойствах объекта Queue в категории «Специфические» необходимо включить флаг «Разрешить уход по таймауту» и задать время ожидания.
3. Через порт OutPreempted, будучи вытесненной другой заявкой при заполненной очереди. Таким образом моделируются очереди ограниченной длины. При этом необходимо включить флаг «Разрешить вытеснение».
4. «Вручную», путем написания программного кода (функции remove() или removeFirst()).

Порты объекта Queue представлены на *рис. 2.7*.

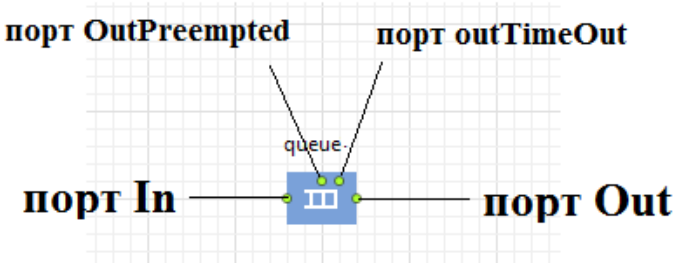


Рис. 2.7. Порты объекта Queue

Существует возможность организации различных режимов ожидания. По умолчанию заявки помещаются и извлекаются из очереди согласно правилу FIFO (первый пришел – первый обслужен). Возможна также организация очередей согласно правилу LIFO (последний пришел – первый обслужен), либо по приоритету заявки. В этом случае в случае занятости очереди заявка может быть вытеснена лишь в том случае, когда приоритет новой поступившей заявки ВЫШЕ приоритета находящейся в очереди заявки.

Выбрать дисциплину ожидания можно в категории «Специфические» свойств объекта Queue (*рис. 2.8*).

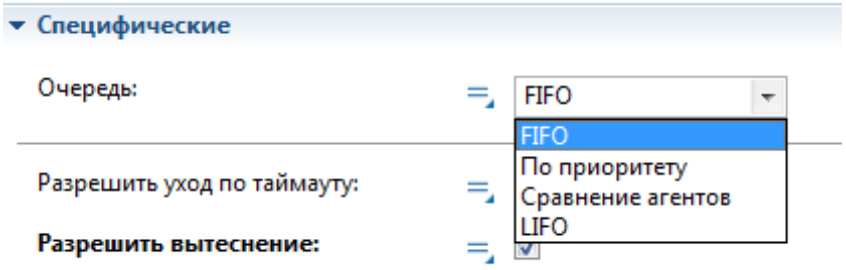


Рис. 2.8. Выбор дисциплины ожидания

Для уничтожения заявок, завершивших моделирование, используется блок Sink. Данный объект не имеет каких-либо специфических свойств. Для него можно задать лишь действия, которые необходимо выполнить при входе в этот блок. Sink имеет лишь один входной порт; выходных портов у него нет.

В некоторых случаях возникает необходимость изменения маршрутов заявок в зависимости от выполнения или невыполнения условия, а также в вероятностном режиме. Для этого используется объект **Select Output**. Этот объект представляет собой логический блок с одним входным и двумя выходными портами (true и false). Выход true можно выбрать с заданной вероятностью (в этом случае в появившейся строке указывается вероятность данного события p, а с вероятностью 1-p заявка пойдет в выходной порт false) или при выполнении условия (в этом случае заявка будет направлена на порт true, если условие истинно, на порт false в противном случае).

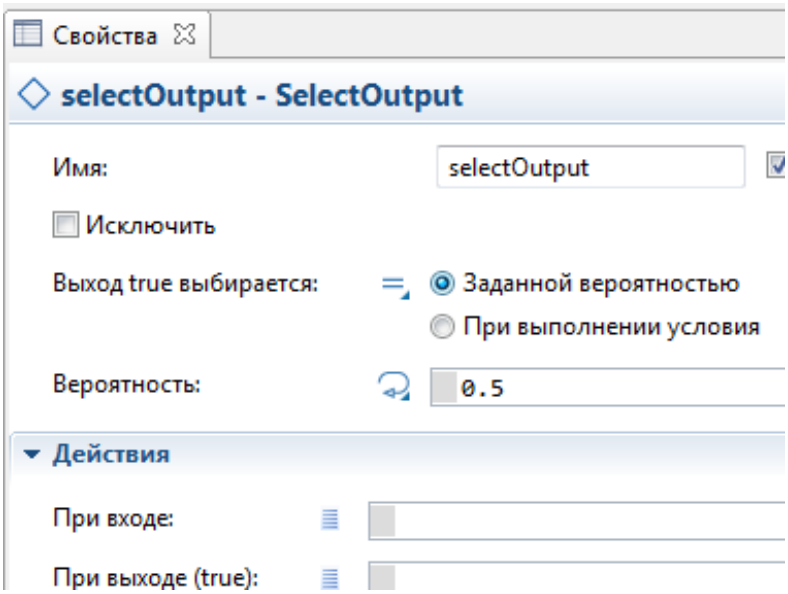


Рис.2.9. Свойства объекта Select Output

Рассмотрим процесс создания элементарной модели.

Пример. Пусть процесс обслуживания заключается в обработке заявки на одном из трех идентичных устройств. Заявки поступают с интенсивностью 2 в минуту. Время обслуживания распределено экспоненциально со средним значением 90 с.

Смоделируем работу системы. Для этого предварительно составим схему, представленную на *рис. 2.10*.

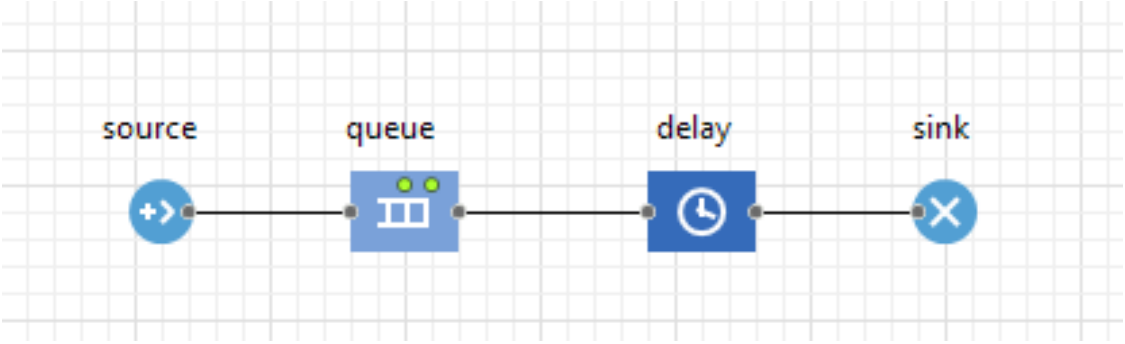


Рис. 2.10. Пример простейшей модели

Рассмотрим свойства каждого из блоков. У блока Source в свойствах «Прибывают согласно» выберем из списка «Интенсивности» и в следующем поле зададим саму интенсивность.

Далее перейдем к блоку Delay. В поле «Время задержки» зададим значение exponential(1.0/90.0). Подробнее о времени обслуживания описано в приложении 1 (стр. ???). В данном случае 0 после запятой ставить необходимо, поскольку, в противном случае, деление будет целочисленным, в результате чего будет получен 0. Можно предварительно выполнить деление, и в параметре функции exponential написать готовый результат. В следующем свойстве «вместимость» зададим значение 3: три идентичных устройства (*рис. 2.11*).

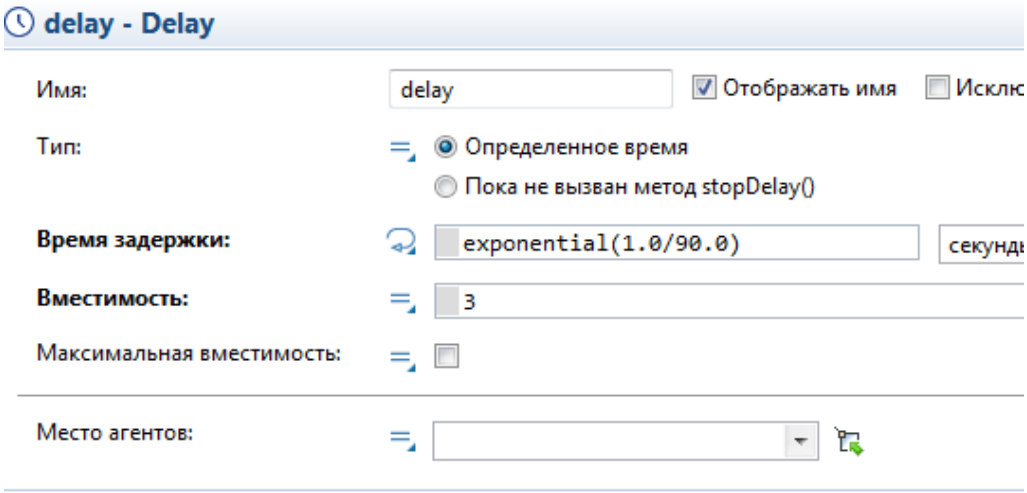


Рис. 2.11. Свойства блока delay

На этом первый этап моделирования завершен. После него необходимо приступить к так называемому прогону модели. Эта имитация процесса прохождения заявок по всем заданным блокам в соответствии с имеющимися свойствами этих блоков. Нажав кнопку F5, переходим к прогону. В процессе прогона пользователю отображается движение заявок от одного блока к другому с указанием числа заявок, вошедших в каждый блок (*рис. 2.12*).

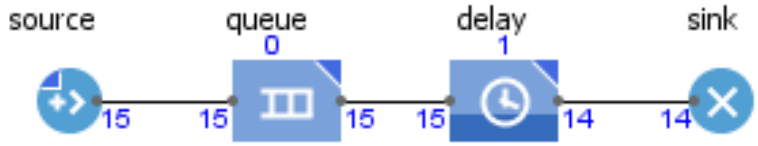


Рис.2.12. Прогон модели

Из данного рисунка видно, что блоком Source было создано 15 заявок, 15 заявок входили в очередь и 15 заявок вышли из нее (т.е. в данный момент очередь пуста); 15 заявок посетили блок delay, имитирующий обслуживающее устройство, а вышли из него 14 заявок; 1 заявка в данный момент находятся в этом устройстве. Все 14 заявок, покинувших устройство delay, были уничтожены блоком sink и покинули систему.

Щелкнув по любому объекту модели, можно увидеть статистику о его функционировании. На *рис. 2.13* представлена статистика о работе трехканального устройства.

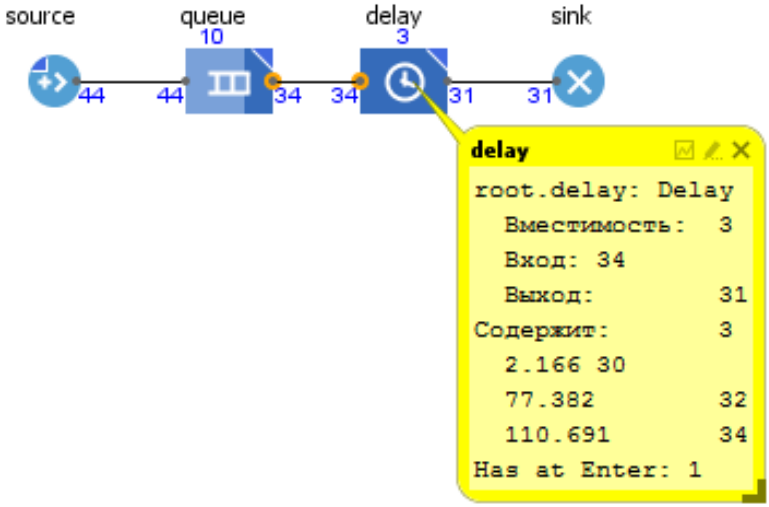


Рис.2.13. Статистика о работе устройства

В частности, здесь показано, что в данный момент в устройстве находится 3 заявки. 34 заявки входили в устройство за все время до данного момента. Сейчас в устройстве заявки под номерами 30, 32 и 34. Перед ними указано оставшееся время обслуживания.

По умолчанию после запуска процесс имитации будет происходить до тех пор, пока не будет нажата кнопка «Стоп». Однако, можно задать остановку процесса по прошествии некоторого времени. Для этого необходимо перейти на вкладку «Проекты» и выбрать Simulation (Main). Далее задать необходимое модельное время. Есть возможность выбора режима виртуального или реального времени. В режиме виртуального времени будет осуществляться прогон настолько быстро, насколько это возможно. В режиме реального

времени задается связь модельного и реального времени. Задав конечное время и из выпадающего списка остановить выбрав «В заданное время», получим модель, которая проработает заданное время, после чего остановится. После этого уже имеет смысл анализировать статистику.

2.3. Статистика

Система anylogic имеет ряд объектов, позволяющих осуществлять сбор статистики. В частности, на палитре есть вкладка «Статистика», которая позволяет собрать разнообразные статистические сведения и представить в наиболее удобном для пользователя виде. Она содержит следующие элементы (*рис. 2.14*).

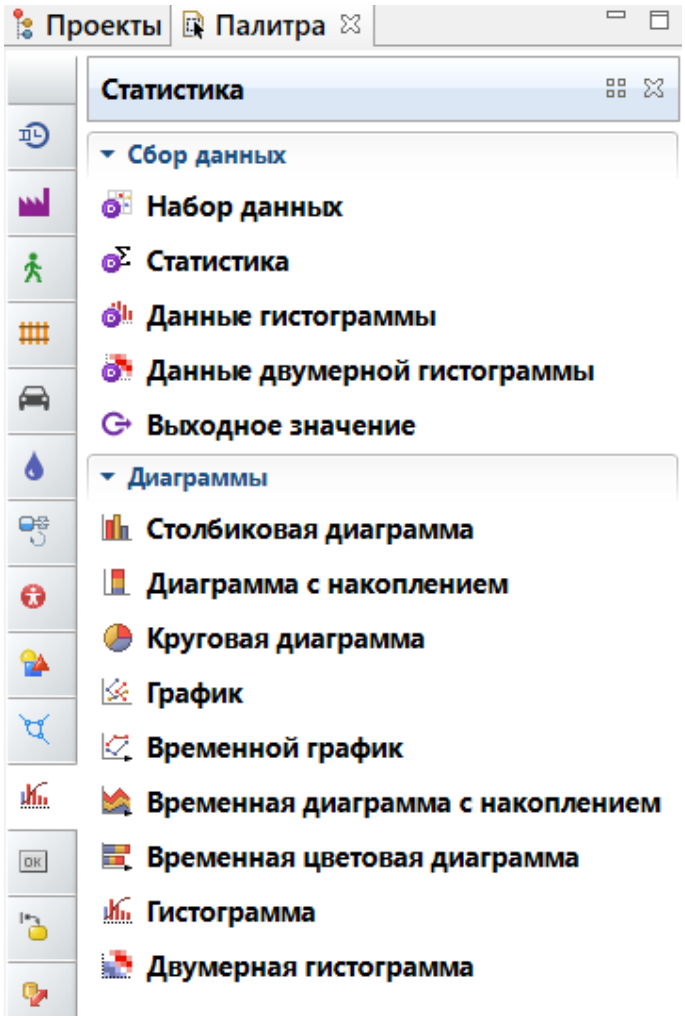


Рис. 2.14. Палитра «Статистика»

Рассмотрим пример использования данной палитры при моделировании трехканальной обслуживающей системы с неограниченной очередью. К обычной модели, включающей блоки Source (создать), Queue (встать в очередь), Delay (имитировать обслуживание), Sink (удалить из модели), добавим два объекта «Статистика». Один из них свяжем с текущей загрузкой устройства, а другой – с текущей загрузкой очереди (на *рис. 2.15* приведен пример «связывания» объекта с устройством Delay).

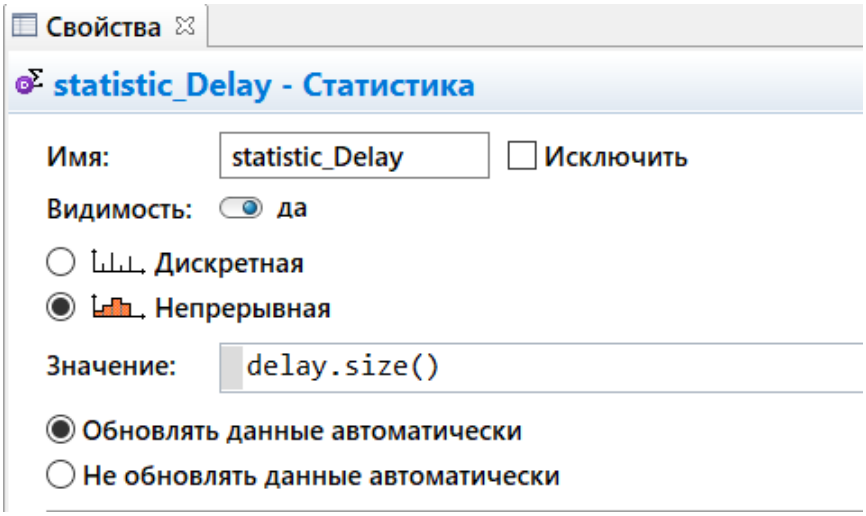


Рис. 2.15. Настройка свойств элемента «статистика»

В этом случае после прогона модели можно увидеть более подробную информацию об интересующих величинах (в частности, о загрузке устройства и очереди). Результат представлен на *рис. 2.16*.

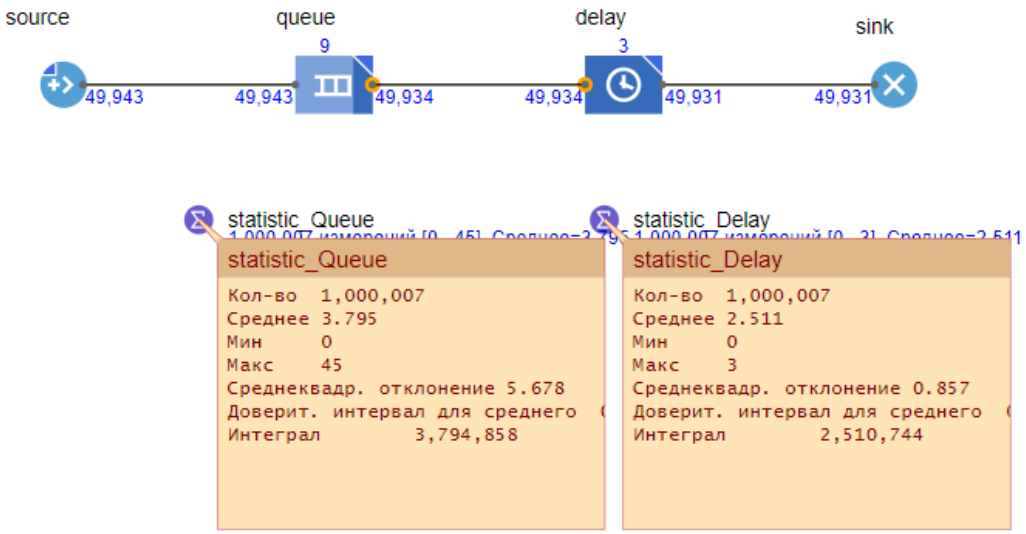


Рис. 2.16. Результаты моделирования

Из результатов видно, что средняя длина очереди составляет 3.795 ед., максимальная длина – 45; среднеквадратическое отклонение (характеризует разброс значений относительно среднего) – 5.78 и т.д. Аналогичную информацию можно получить по устройству Delay: среднее число занятых устройств – 2.511, максимальное – 3 (это ограничение, поставленное на Delay), среднеквадратическое отклонение – 0.857 и т.д.

При моделировании также достаточно часто требуется сбор временных статистических данных: время ожидания, время, проведенное в системе в

целом или на некотором ее участке и т.д. Для этого могут использоваться временные графики и гистограммы.

Сбор временных данных можно выполнить как с помощью объектов anylogic, так и программным способом. В первом случае необходимо включить в модель два объекта: «Time Mesure Start» в точке начала временного отсчета и «Time Mesure End» в точке окончания сбора временной статистики. Для программного сбора статистики необходимо, чтобы поступающие в систему заявки содержали два параметра: «начало отсчета» и «конец отсчета». Смысл этих параметров идентичен объектам «Time Mesure Start» и «Time Mesure End». Разница между этими параметрами даст необходимое время. В свойстве «Объекты TimeMesureStart» для объекта TimeMesureEnd необходимо указать объект TimeMesureStart.

Статистика собирается в два объекта. Один из них предназначен для построения гистограмм (distribution). Второй объект необходим для построения временных графиков (dataset).

Пример.

Пусть для обычной системы обслуживания с очередью необходимо определить гистограмму случайной величины, описывающей время ее пребывания в системе и временной график пребывания. Модель системы будет иметь следующий вид (*рис. 2.17*).

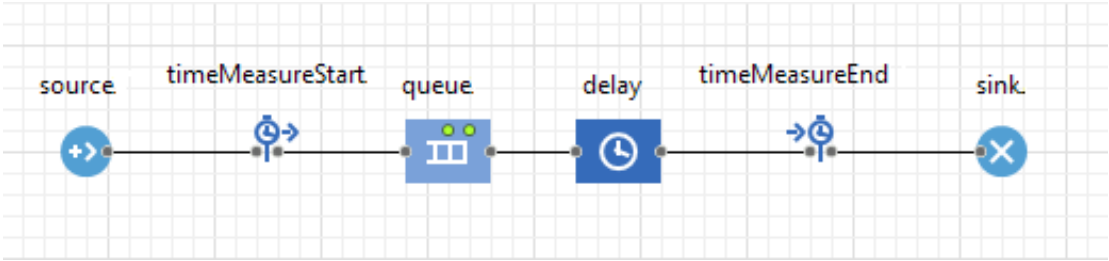


Рис. 2.17. Модель исследуемой системы

Для визуализации собранной статистики в anylogic есть элемент «Гистограмма». Основное ее свойство «Данные» содержит объект, который использовался для сбора данных. Это может быть, например, объект «Time Mesure End». Поместим два объекта из палитры «Статистика»: гистограмму и временной график. В свойстве Данные гистограммы укажем: timeMesureEnd.distribution (*рис.2.18*).

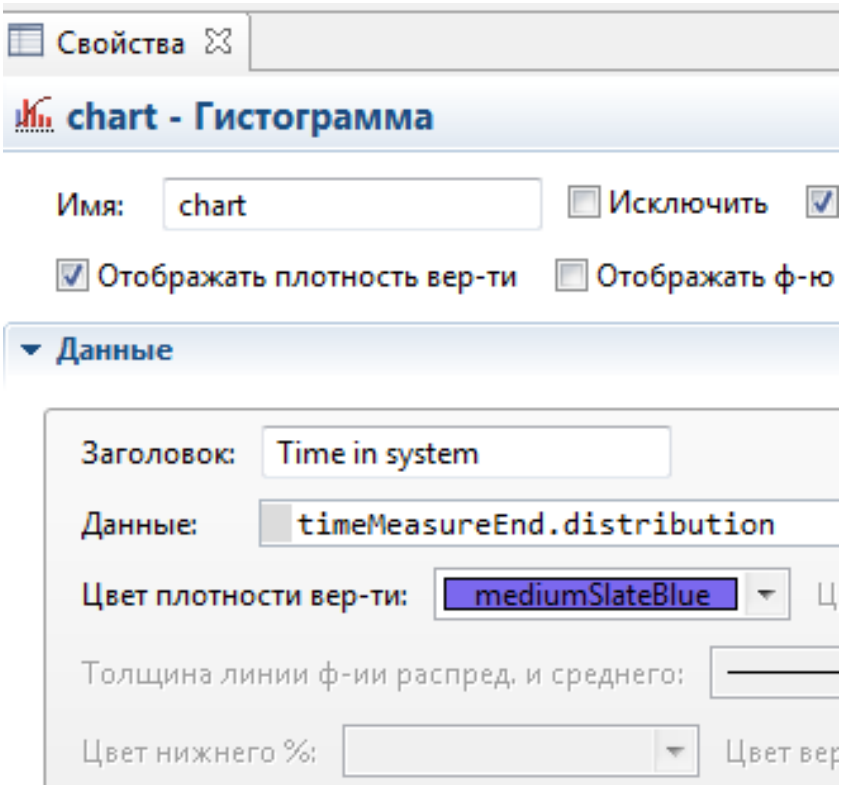


Рис. 2.18. Настройка свойств гистограммы

В аналогичном свойстве временного графика укажем timeMesureEnd.dataset (*рис.2.19*).

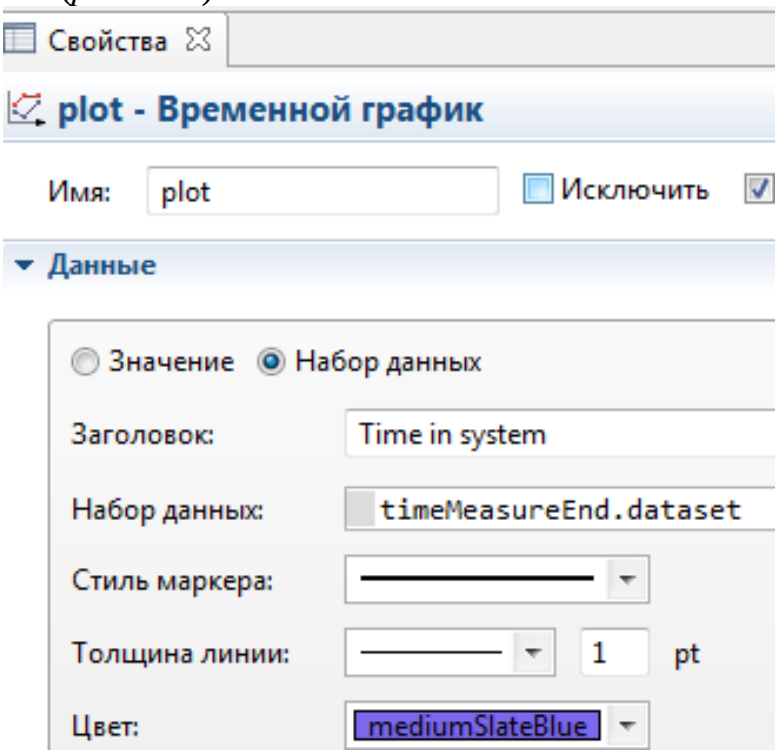


Рис. 2.19. Настройка свойств временного графика

Получим следующие результаты (*рис. 2.20*).

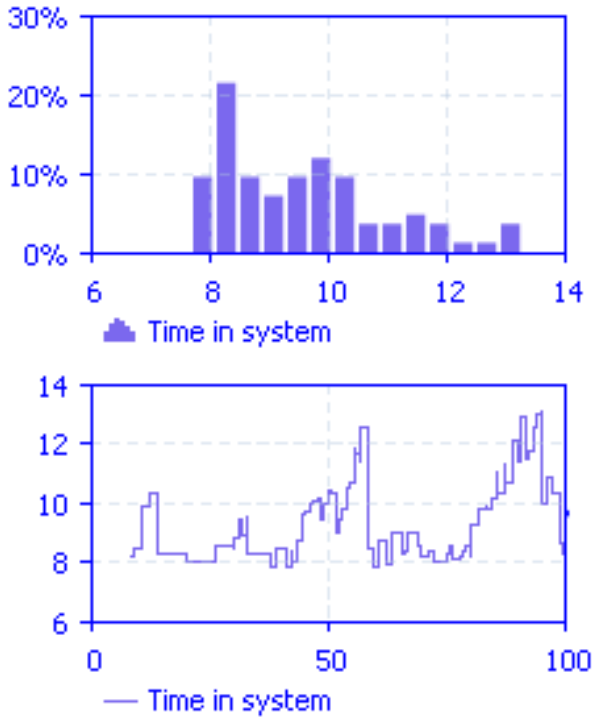


Рис. 2.20. Графики

Другой вариант набора данных для гистограммы – это объект «Данные гистограммы». Рассмотрим другой пример сбора данных об ожидании перед одноканальным устройством. Схема модели представлена на *рис. 2.21*.

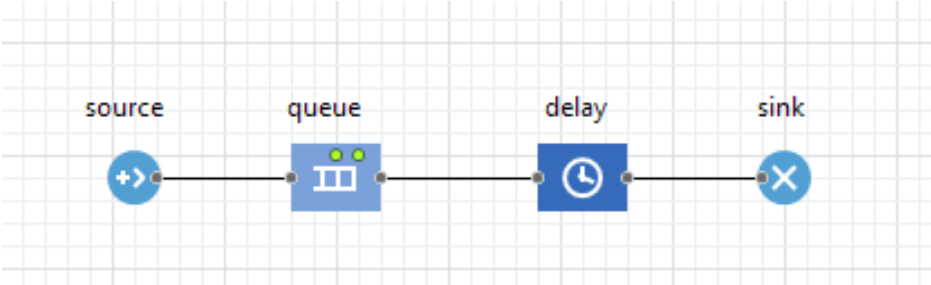


Рис. 2.21. Схема модели обслуживания с очередью

Добавим элемент «Данные гистограммы», который назовем TimeWaitDistr.

Чтобы собрать статистику, создадим новый тип агента «Clients», у которого будет свойство StartWaiting типа «время». При входе в очередь будем фиксировать это время, а при выходе получим время ожидания как разницу между текущим на тот момент временем и временем StartWaiting. Это временную разницу будем добавлять в элемент «TimeWaitDistr». Поскольку параметр StartWaiting является параметром агента Clients, то обратиться к параметру необходимо с указанием агента (*рис. 2.22*).

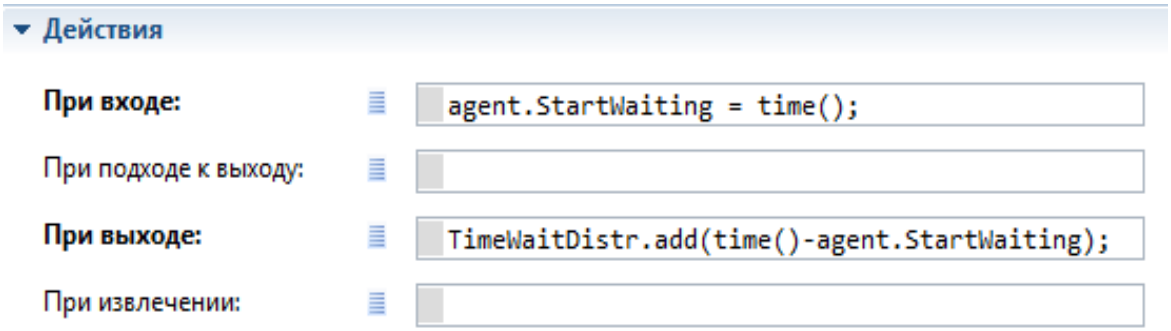


Рис. 2.22. Действия при входе и выходе из очереди

При этом у объекта source необходимо в поле «Новый агент» выбрать «Clients», поскольку необходимо, чтобы в модель поступали заявки, имеющие свойство StartWaiting, что есть именно у агентов созданного типа Clients.

2.4. Параметры

Любой активный объект может иметь параметры. Параметры обычно используются для задания характеристик объекта или расчета каких-либо величин (счетчики и т.д.). Значения параметров можно при необходимости менять во время работы модели.

Элемент параметры находится на вкладке «Агент» Палитры. Перетащив его на диаграмму класса Main, можно задать его значение как сразу (в свойствах поле «Значение по умолчанию»), так и изменять его во время моделирования.

Пример. Рассмотрим возможность изменения интенсивности поступления заявок в процессе моделирования. Создадим параметр intens и зададим ему значение 10 (*рис. 2.23*).

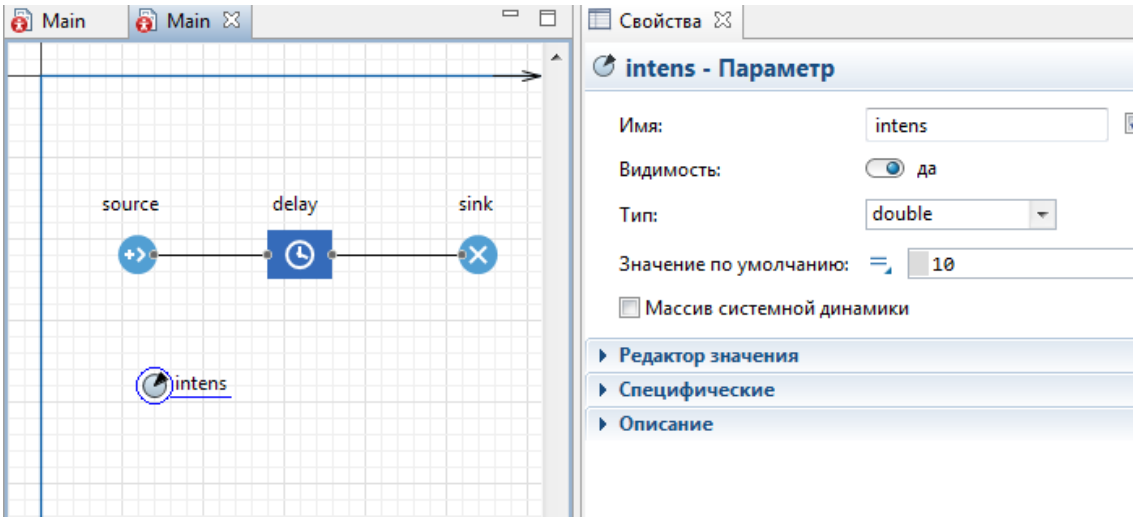


Рис. 2.23. Работа с параметром

Объект «Delay» сделаем 6-канальным с временем обслуживания – 1 с (время распределено экспоненциально).

В свойстве «Интенсивность прибытия» объекта source впишем параметр intens.

Из вкладки Палитры «Элементы управления» перетащим в рабочую область объект «Бегунок» и свяжем его с параметром intens (*рис. 2.24*).

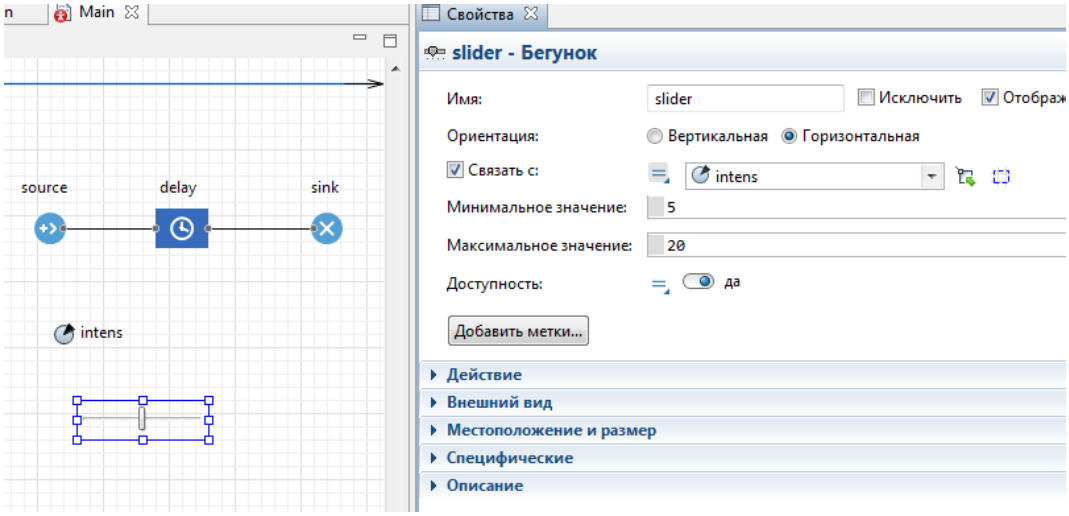


Рис. 2.24. Бегунок

Добавим объект Select Output для отказа в случае занятости устройства delay. Условие будет иметь вид:

Delay.size()<6

В результате получим следующую модель (*рис. 2.25*).

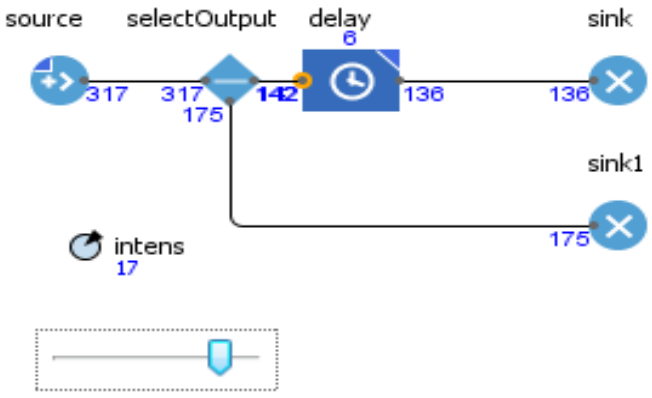


Рис.2.25. Моделирование с переменной интенсивностью

Варьируя значения бегунка, будем изменять интенсивность входного потока. Возможна также корректировка значений параметров в самой модели. Предположим, что если заявка получила отказ, то интенсивность потока снижается до 5 заявок в с, а в случае успешного обслуживания – увеличивается до 15 заявок в с.

В свойствах объекта Sink необходимо прописать действия, совершаемые при входе. Выбрав соответствующую строку, запишем (*рис. 2.26*).

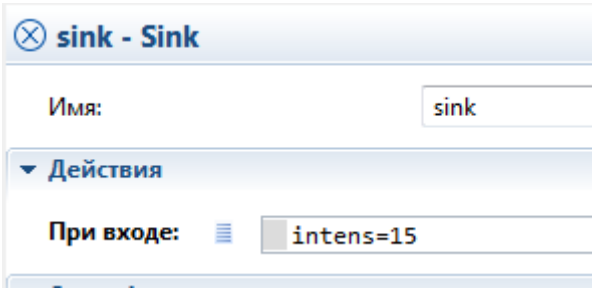


Рис.2.26. Изменение значений параметра при входе в блок Sink

Аналогичным образом для случая попадания заявки в объект Sink1 значению параметра intens присваивается значение 5.

2.5. Диаграмма состояний

Если у активного объекта (или у системы в целом) можно выделить несколько состояний, возникающих из-за наступления определенных событий и требующих выполнения определенных действий, то поведение такого объекта или системы может быть описано в виде диаграммы состояний. Диаграмма состояний позволяет графически задать пространство состояний алгоритма поведения объекта, а также события, которые являются причинами срабатывания переходов из одних состояний в другие и действия, происходящие при смене состояний.

Объекты диаграммы состояний представлены на *рис.2.27*. Рассмотрим отдельные блоки этой палитры.

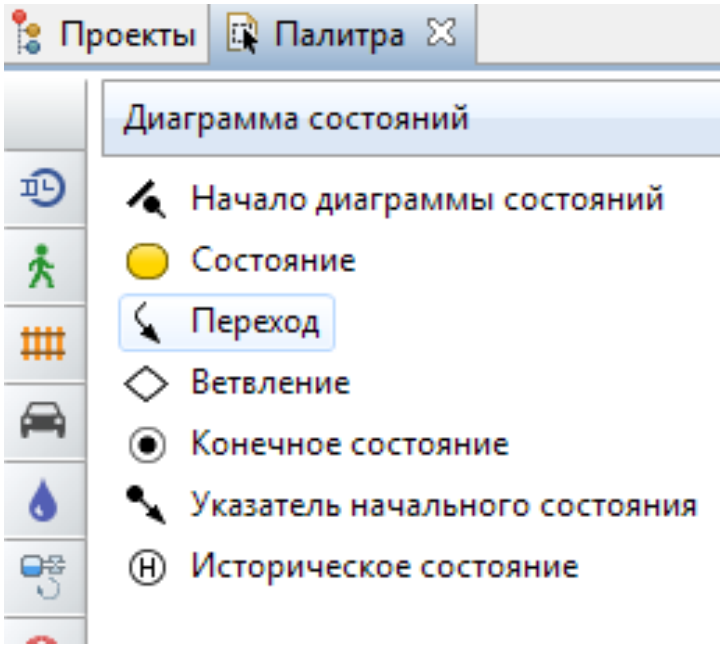


Рис. 2.27. Диаграмма состояний

Состояние

Состояние представляет собой местонахождение управления диаграммы состояний. При необходимости, можно задать действия, которые должны быть выполнены при наступлении определенных событий и/или выполнении некоторых условий. Состояние может быть простым или сложным (составным). Сложное состояние включает в себя множество простых состояний.

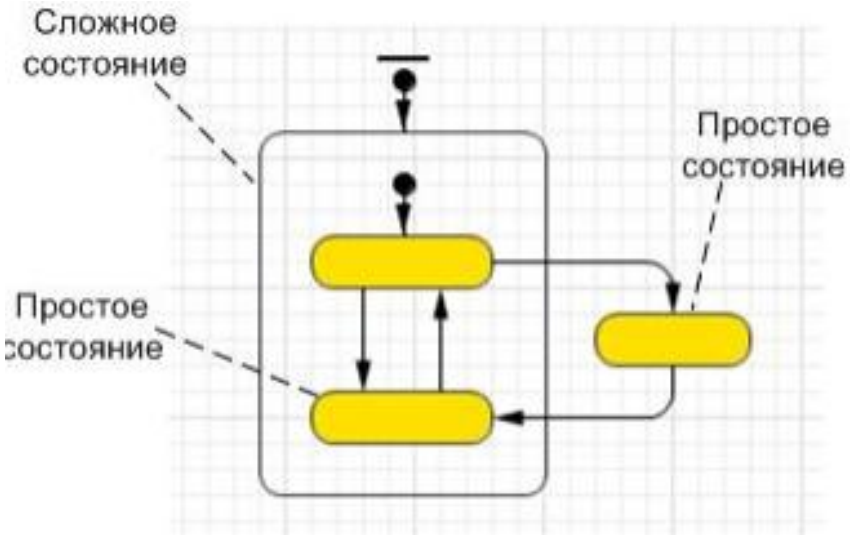


Рис. 2.28. Простые и сложные состояния

Управление всегда принадлежит простому состоянию. В связи с этим, в сложном состоянии обязательно необходимо указать, с какого простого состояния внутри сложного состояния будет начат моделируемый процесс.

Основными свойствами состояний являются:

- действие при входе - код, выполняемый, когда управление переходит в это состояние;

- действие при выходе – код, выполняемый, когда управление покидает это состояние (состояние перестает быть активным).

Переход

Переход означает переключение управление диаграммы состояний, ее переход из одного состояния в другое. В Anylocig можно задать тип события, при наступлении которого срабатывает переход. Для этого необходимо заполнить свойство «происходит», которое может принимать следующие значения.

1. По таймауту – переход будет активирован по истечении заданного времени (в этом случае необходимо ввести временное значение в появившееся поле «Таймаут», по истечении которого должен сработать переход).

2. С заданной интенсивностью (в данном случае необходимо значение интенсивности в появившееся поле «Интенсивность»). Переход сработает через экспоненциально распределенное время с заданной интенсивностью.

3. При выполнении условия (появляется дополнительное полу «Условие», в которое необходимо написать логическое условие, при котором должен сработать переход).

4. При получении сообщения (в данном случае необходимо выбрать тип сообщения, при получении которого должен сработать переход). Переход будет активирован при прибытии сообщения в соединенный с диаграммой состояний порт. В этом случае возможно также задать одно из следующих условий опции «Осуществлять переход»:

- безусловно (если проверка самого сообщения не требуется);
- если сообщение равно (переход сработает в случае, если сообщение будет совпадать с заданным сообщением);
- если выполняется условие (в данном случае можно описать код проверки содержимого сообщения, при выполнении которой будет осуществлен переход).

5. По прибытию агента – переход будет активирован, когда агент, чье поведение задается диаграммой состояний, достигнет точки назначения.

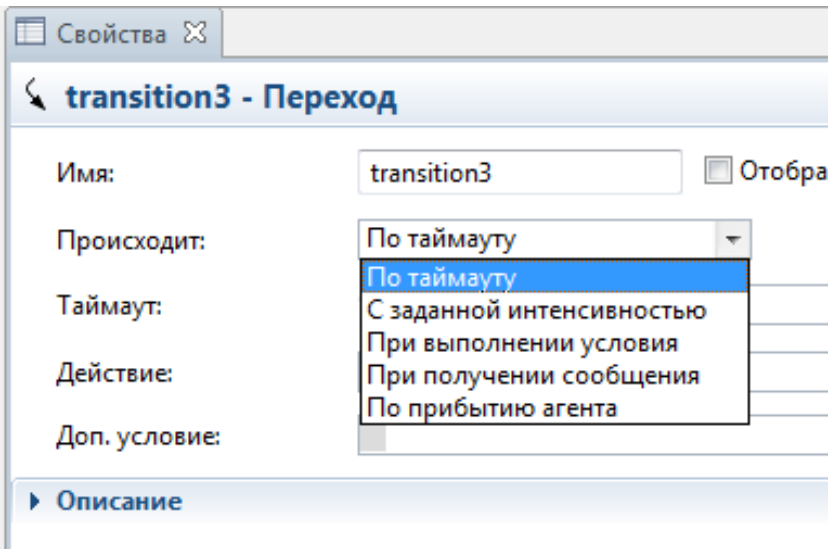


Рис. 2.29. Свойства перехода

Свойство «Действие» содержит последовательность операторов Java, выполняемых при срабатывании перехода.

Дополнительное условие – это логическое выражение, разрешающее (true) или запрещающее (false) срабатывание перехода. Если условие не задано, то оно подразумевается true.

Конечное состояние

Конечное состояние является конечной точкой диаграммы состояний. Когда управление передается в конечное состояние, выполняется действие этого состояния и диаграмма завершает свою работу. Из конечного состояния не могут выходить никакие переходы.

Ветвление

Ветвление представляет собой точку разветвления или соединения переходов. С помощью ветвления можно создать переход, имеющий более одного пункта назначения или соединить несколько переходов, выполняющих некоторое общее действие.

Когда управление проходит через состояние-ветвление, выполняется действие этого состояния, и вычисляются дополнительные условия переходов, исходящих из этого состояния. Срабатывает первый же найденный разрешенный переход.

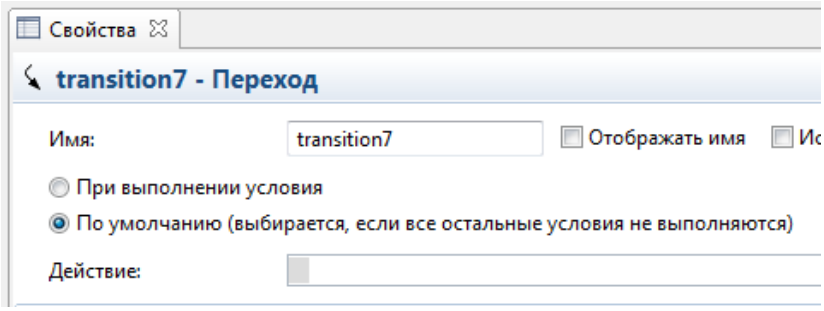


Рис. 2.30. Переход из состояния – ветвления

Переходы, ведущие из состояний-ветвлений, имеют следующие дополнительные свойства:

- при выполнении условия (при выборе данного типа перехода он будет срабатывать, если заданное логическое условие истинно);
- по умолчанию при выборе данного типа перехода он будет выполняться в том случае, если условия всех остальных переходов, ведущих из состояния ветвления, не выполняются.

Пример. Рассмотрим работу диаграммы состояний на примере иллюстрации работы светофора. Поскольку в данном случае невозможно выделить начало и конец, а функционирование светофора фактически представляет собой смену состояний (определенных цветов) через заданные промежутки времени, для решения данной задачи целесообразно использовать диаграмму состояний. Выделим следующие состояния:

- зеленый (движение);
- зеленый мигающий (внимание);
- желтый (замедление).
- красный (остановка);
- красный с желтым (приготовиться);

Все состояния, кроме «зеленый мигающий» являются простыми. Состояние «внимание» - это сложное состояние, состоящее из переключения между двумя простыми:

- зеленый свет включен;
- зеленый свет выключен.

Таким образом, схема диаграммы состояний будет следующей (*рис. 2.31*).

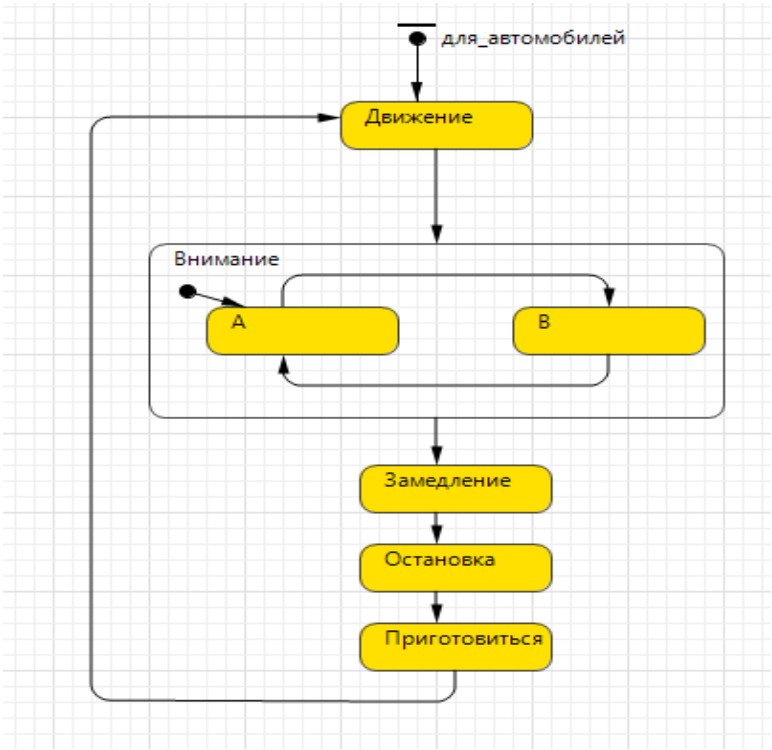


Рис. 2.31. Диаграмма состояний для задачи «Светофор»

Далее целью является попеременное включение и выключение определенных цветов светофора. Для этого создадим три булевых параметра cl_red, cl_yellow и cl_green, соответствующие включению и выключению красного, желтого и зеленого цветов. Эти параметры будут принимать значение true, если необходимо включить соответствующий цвет, и false – в противном случае. Тогда в случае попадания в состояние присваиваем соответствующему параметру значение true, если выходим из состояния – false. Пример для параметра, отвечающего за зеленый свет при входе и выходе в состояние «Движение» представлен на *рис. 2.32*.

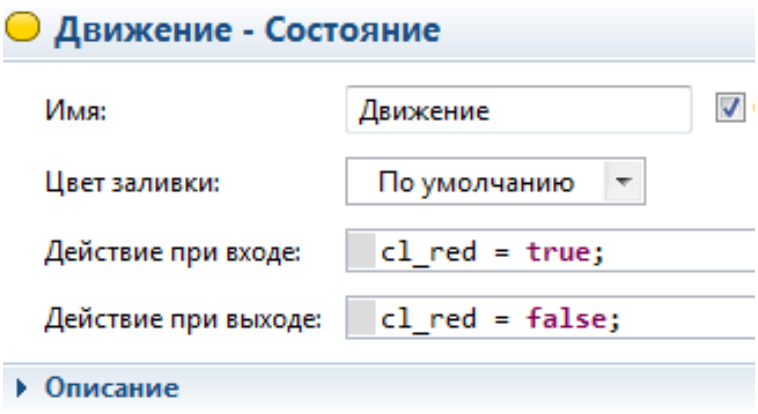


Рис. 2.32. Задание параметрам значений

Далее необходимо выполнить визуализацию работы светофора. Для этого воспользуемся библиотекой «Презентация», у которой есть объекты типа «прямоугольник» и «овал». С помощью данных объектов нарисуем светофор. Для связи параметров с цветами овалов, создадим динамическое изменение цвета в зависимости от значений параметров. Для красного цвета настройка будет иметь следующий вид (*рис. 2.33*).

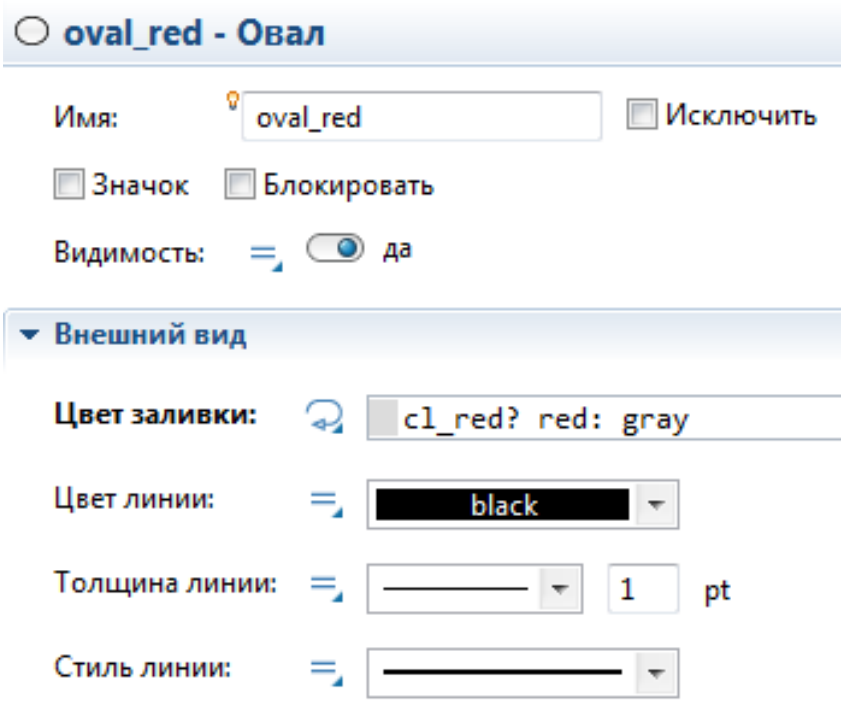


Рис. 2.33. Настройка красного цвета для светофора

Остальные цвета будут настраиваться аналогичным образом.

2.6. Пешеходная библиотека

В условиях растущего населения крупных городов и увеличения темпов строительства зданий все большую актуальность приобретает моделирование пешеходных потоков. В связи с этим, в среде Anylogic предусмотрены блоки, которые обеспечат возможность моделирования и наглядной визуализации движения пешеходов.

Целесообразность построения переходных моделей продиктована целым рядом причин. Во-первых, такая модель весьма эффективна на стадии проектирования нового проекта. Ее наличие поможет осуществить поиск наилучших решений, сравнение различных вариантов и быстрой оценки вносимых изменений. Во-вторых, она целесообразна на стадии предварительной оценки проекта. В этом случае имитационное моделирование поможет оценить планируемую нагрузку проекта и, при необходимости, внести корректировки на ранних стадиях. Пешеходные модели могут использоваться и на работающих объектах для достижения следующих целей:

- поиск наилучших точек для размещения рекламы, торговых палаток и т.д.;

- оптимизация работы торговых точек (количество персонала, часы работы и т.д.);

- оценка пропускной способности исследуемого объекта при увеличении нагрузки;

- решение различных вопросов по безопасности (планы эвакуации и т.д.);

- поиск наиболее оптимальных временных маршрутов при ремонтных работах;

и т.д.

Исходя из данных целей, при моделировании движений пешеходов решаются следующие задачи:

- оценка пропускной способности зданий и объектов внутри них;

- оптимизация бизнес-процессов в пунктах обслуживания;

- оценка плотности потока посетителей торговых зон;

- нахождение «узких мест» пешеходных потоков;

- создание и обоснование планов эвакуации при ЧС;

- оценка доступности парковок, дорожной сети и общественного транспорта.

Пешеходная библиотека имеет вид, представленный на *рис. 2.34*.

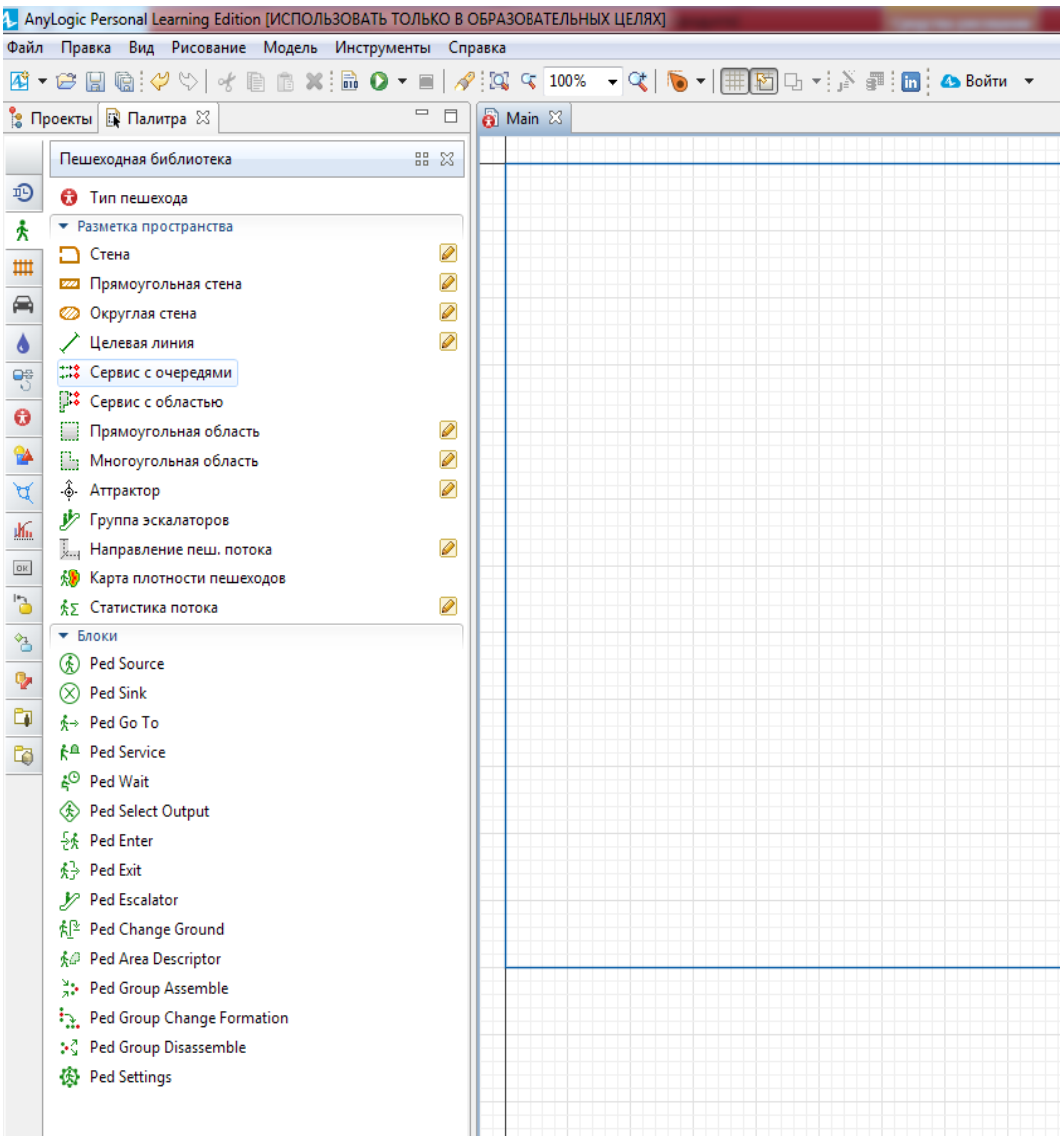


Рис. 2.34. Пешеходная билиотека

Моделирование процесса пешеходного движения осуществляется в три этапа. На первом этапе на рабочую область наносится чертеж моделируемого объекта. С точки зрения Anylogic он будет выступать в качестве рисунка. На втором этапе осуществляется так называемая разметка пространства. Она заключается в нанесении поверх данного чертежа пешеходных маршрутов, направлений пешеходных потоков, стен, торговых точек и т.д. Этот этап осуществляется с помощью библиотека «Разметка пространства», однако все основные блоки разметки пространства, необходимые для решения данной задачи, вынесены также в пешеходную библиотеку. На последнем этапе производится непосредственно моделирование. На основании статистических данных определяются источники потоков и их интенсивности, направления потоков и т.д.

Рассмотрим основные блоки пешеходной библиотеки, доступные для решения описанной задачи моделирования.

PedSource. Данный блок предназначен для создания в модели пешеходов. Поток пешеходов задается:

- местом своего появления;
- интенсивностью или временем появления;
- скоростью.

Свойства пешеходов представлены на *рис. 2.35*. Если возникает необходимость в задании более качественной анимации, можно в свойствах «Новый пешеход» выбрать «Создать другой тип». При этом откроется диалоговое окно, в котором можно выбрать иллюстрацию для пешеходов (например, рабочие).

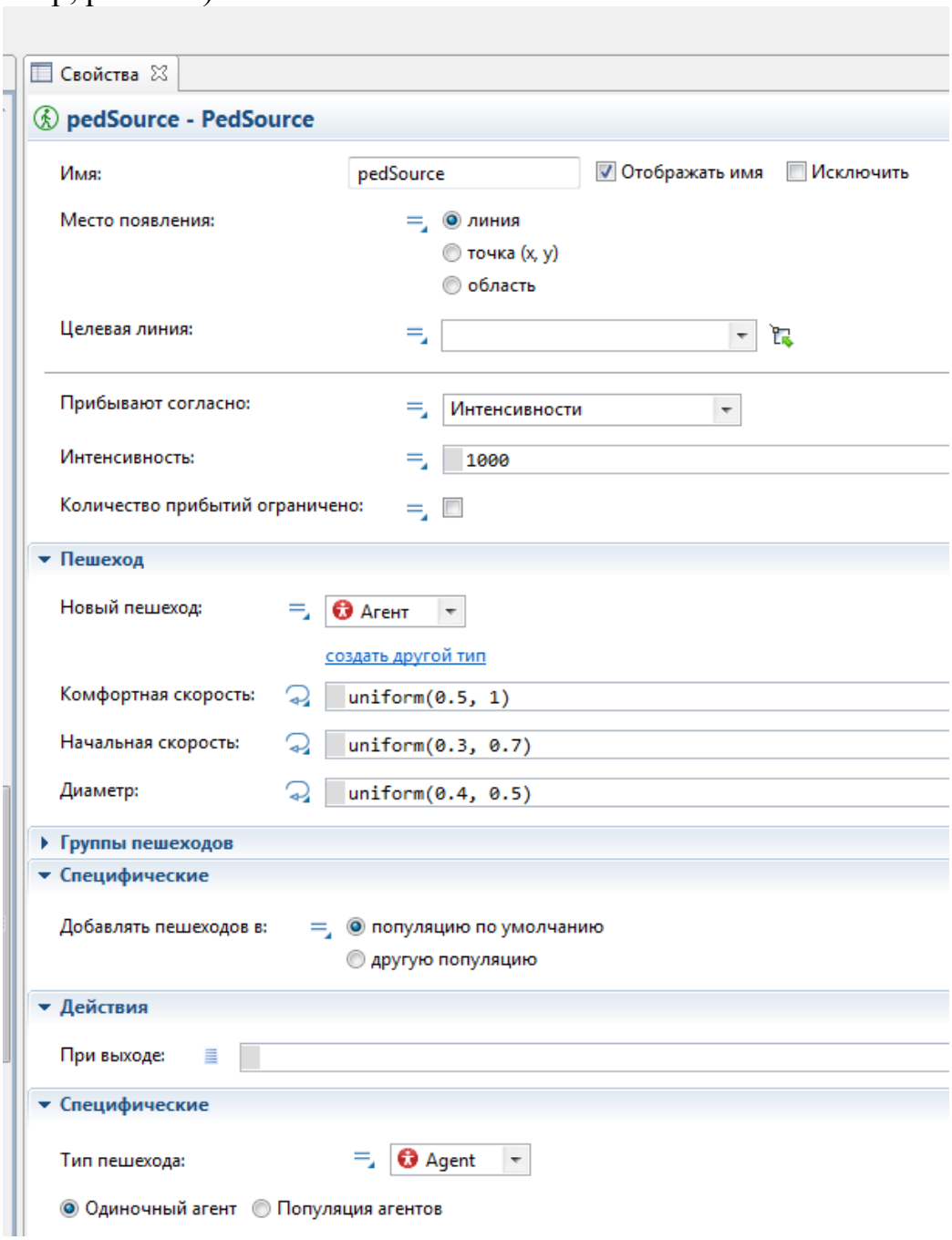


Рис. 2.35. Свойства пешеходов

PedGoTo. Заставляет пешеходов перейти в заданное место моделируемого пространства, которое может быть задано линией или точкой. Свойства блока представлены на *рис. 2.36*.

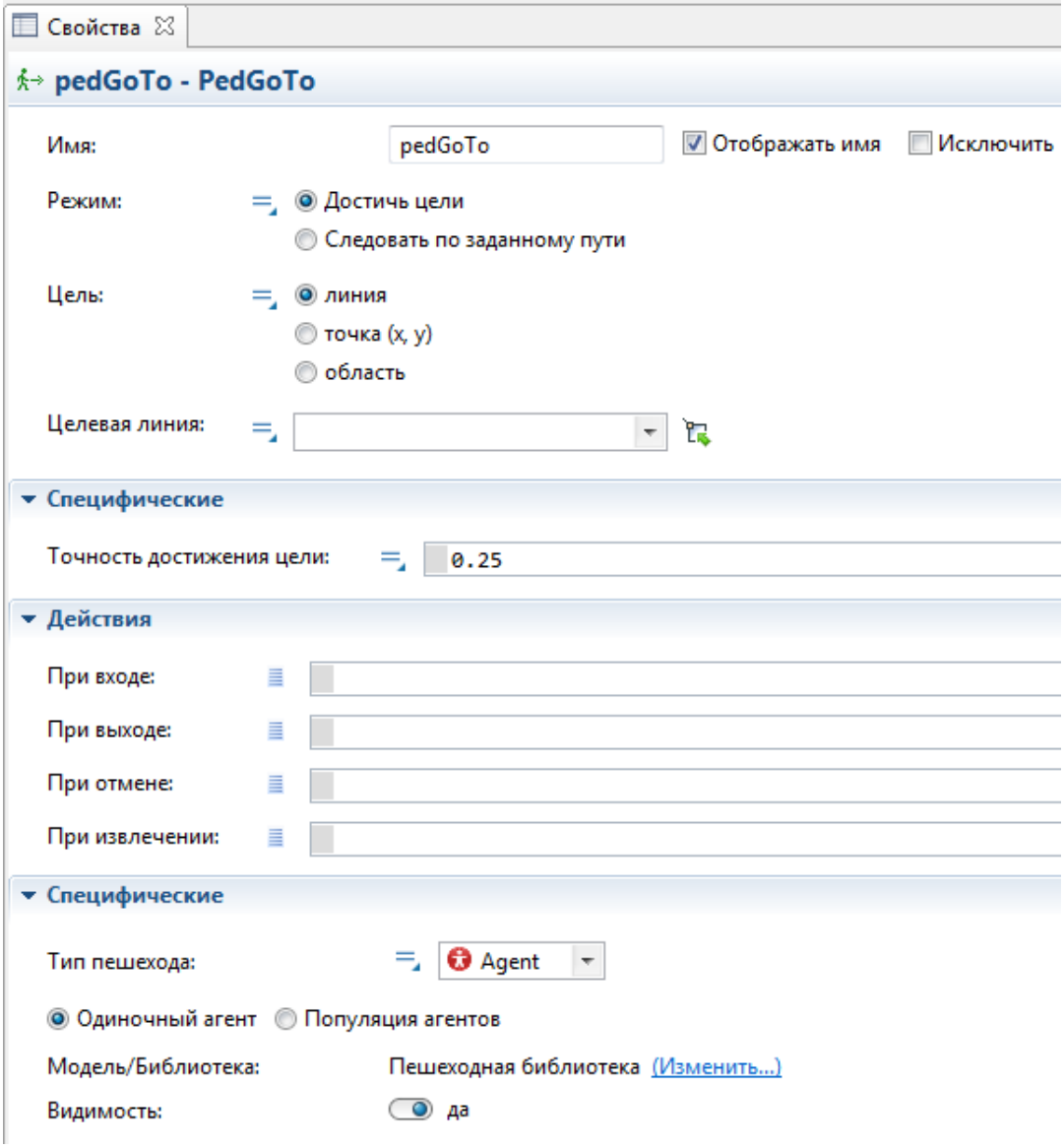


Рис. 2.36. Свойства PedGoTo

Как следует из данного рисунка, пешеход может осуществлять движение одним из двух способов:

- идти к заданной цели;
- двигаться по заданному маршруту.

Цель может быть задана линией, точкой или областью. Маршрут задается блоком «Направление пеш. Потока». При помещении на рабочую область этот блок фактически представляет собой дорогу, которую можно нарисовать с необходимой для пользователя точностью (в виде ломаной, с поворотами и т.д.).

PedSink. Удаляет поступивших в объект пешеходов из моделируемой среды. Обычно объект используется в качестве конечной точки блок-схемы,

формализующей поток пешеходов. PedSink автоматически ведет подсчет пешеходов.

PedWait. Заставляет пешеходов перейти в заданное место и ожидать там в течение определенного периода времени. Место может быть выбрано случайно внутри заданной области (с аттракторами или без них) или указано вручную. Аттракторы - это места внутри области, которые притягивают пешеходов во время их ожидания (например, реклама, информационные стенды и т.д.).

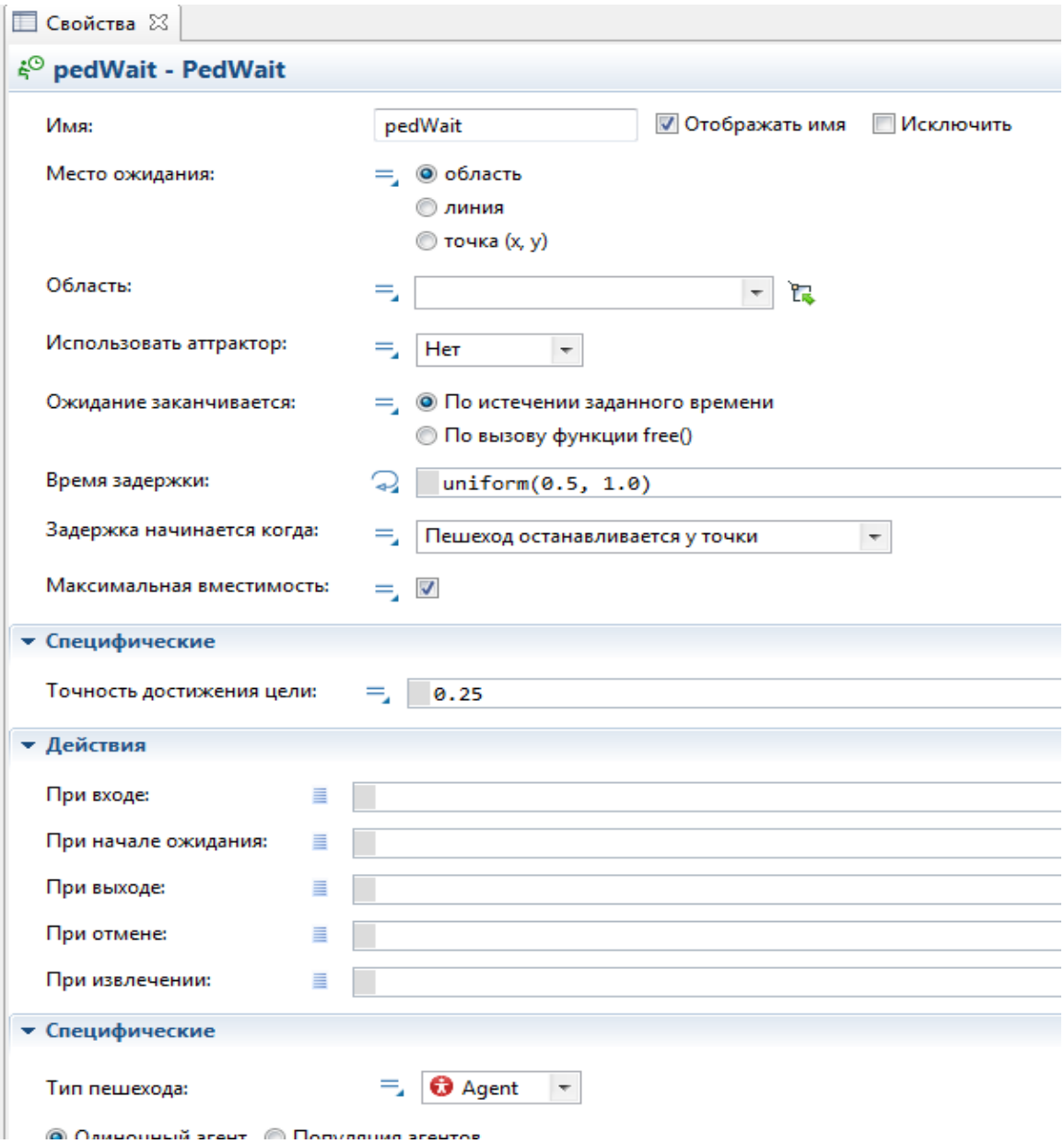


Рис. 2.37. Свойства блока PedWait

Блок PedWait предусматривает два вида ожиданий:

- ожидание в течение заданного времени;
- ожидание до тех пор, пока не произойдет некоторое событие.

Эта особенность регулируется свойством «ожидание заканчивается». В первом случае появляется поле, в которое необходимо ввести время, по истечении которого ожидание будет завершено. Во втором случае, в каком-либо из блоков необходимо вызвать функцию:

pedWait.freeAll();

PedService. Направляет поток пешеходов на обслуживание в некоторый сервис и задерживает там пешехода в течение заданного времени. Данный сервис создается с помощью одного из двух блоков разметки пространства:

- сервис с очередями (используется для того, чтобы задавать сервисы, в которых пешеходы ждут в очереди, пока сервис не будет доступен);
- сервис с областью – используется для того, чтобы задавать сервисы с электронной очередью. В таком случае пешеходы не стоят в очереди, а ждут в расположенной рядом области.

Сервис с областью фактически не имеет существенных свойств, за исключением единственного свойства – количество сервисов. Сервис с очередями, кроме указанного выше свойства содержит также множество свойств, описывающих очереди. Свойства представлены на *рис. 2.38*.

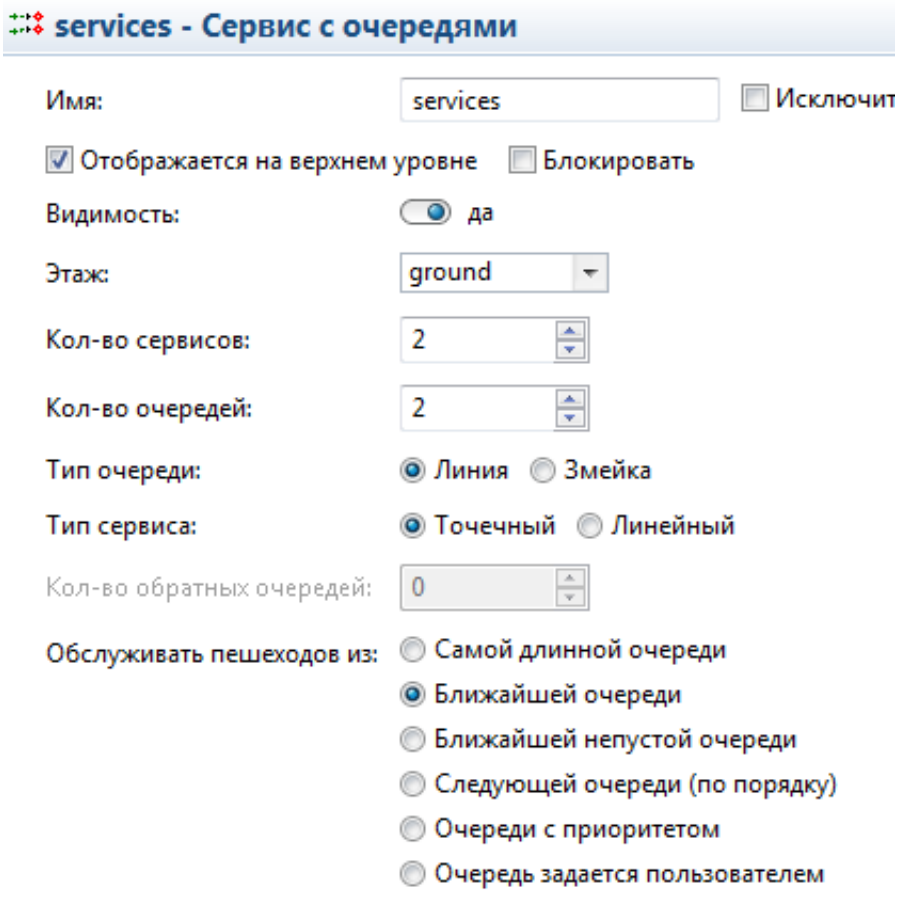


Рис. 2.38. Сервис с очередями

Как видно из данного рисунка, можно выбрать, этаж, тип очереди, а также способ обслуживания (самой длинной очереди, ближайшей очереди и т.д.).

Когда пешеход направляется к конкретному сервису, этот сервис назначается этому пешеходу и становится недоступным (занятым) для остальных пешеходов. Пешеход перемещается в начальную точку линии и начинается ожидать там в течение заданного времени обслуживания, имитируя обслуживание. Затем пешеход перемещается в конечную точку линии.

2.3. Моделирование дорожного движения

Современный этап развития общества характеризуется интенсивным увеличением количества автомобилей. В связи с этим, важной задачей является повышение эффективности транспортной работы автомобильных дорог в проекте организации движения. Ее решение возможно только при широком внедрении в практику вариантного проектирования автомобильных дорог и имитационного моделирования, которое практически невозможно без широкой автоматизации проектирования на основе ЭВМ.

В настоящее время в ведущих дорожных проектных организациях объем автоматизированного проектирования и моделирования составляет от 85% общего объема работ проектирования [].

Высокие требования, которые предъявляет современный автомобильный транспорт к качеству автомобильных дорог, могут быть реализованы лишь при системном подходе, как к самому процессу проектирования, так и к последующим этапам реализации результатов этого проектирования путем моделирования: строительству и эксплуатации [].

Среда Anylogic располагает специальной библиотекой дорожного движения, которая позволяет реализовать следующий функционал:

- моделирование улиц, дорог, перекрестков и транспортных развязок;
- работа общественного транспорта;
- действия автомобилей на парковках;
- анализ плотности траффика в дорожной сети;
- реакция транспорта на светофоры и правила дорожного движения.

Данная библиотека имеет следующий вид (рис. 2.39).

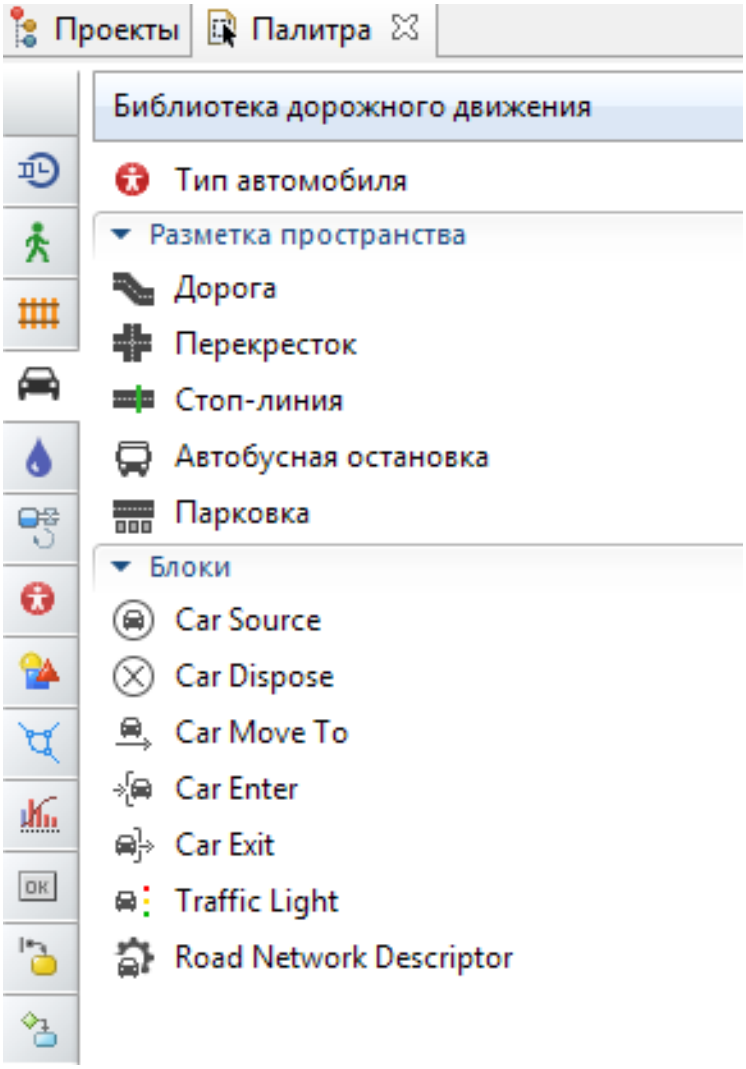


Рис. 2.39. Библиотека дорожного движения

Как и в случае пешеходных потоков, моделирование дорожного движения включает в себя три основные стадии:

- нанесение на рабочую область карты с исследуемыми дорогами (которая будет восприниматься просто как изображение, необходимое пользователю для повышения эффективности визуального восприятия результатов моделирования);

- нанесение поверх данной карты разметки пространства (дороги, стоп-линии, светофоры и т.д.), элементы которой необходимы для работы основных блоков библиотеки;

- описание непосредственно модели с помощью блоков библиотеки дорожного движения.

Рассмотрим основные элементы разметки пространства и библиотеки дорожного движения, необходимые для создания дорожной модели.

Элементы разметки пространства

1. **Дорога** – элементарный компонент разметки пространства, позволяющий создавать дорожную сеть. Дорога может содержать произвольное

количество полос. Дорога может быть как односторонняя, так и двусторонняя. Одна дорога содержит неизменное количество полос на всем своем протяжении. Можно создавать участки слияния дорог, путем соединения двух дорог с разным количеством полос.

2. **Перекресток** – необходим для слияния двух и больше дорог. Перекресток автоматически получается при соединении дорог. При формировании перекрестка автоматически появляются белые точечные линии – соединители полос, которые задают разрешенные направления движения транспорта на перекрестке.

3. **Стоп-линия** – ия является графическим элементом разметки пространства, задающим точку на дороге, у которой транспорт должен останавливаться. Стоп-линия может быть использована блоком Traffic Light (Светофор) для регулирования движения на сложно контролируемых участках дороги.

Блоки дорожного движения

- CarSource – Создает автомобили и пытается поместить их в указанное место дорожной сети. Автомобиль можно поместить на указанную дорогу или парковку. Основные свойства:
  - Интенсивность появления;
  - Место появления;
  - Скорость

И т.д.

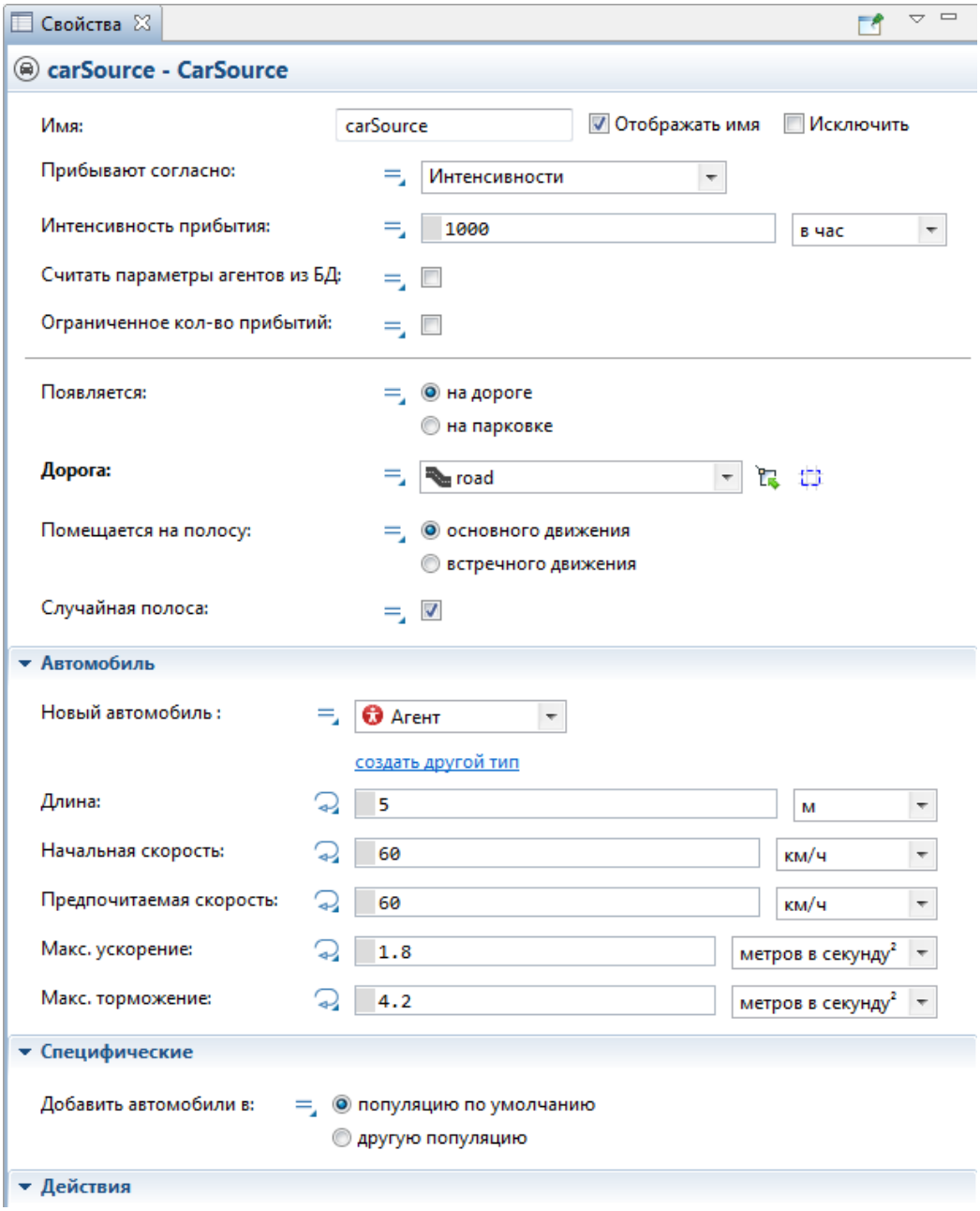


Рис. 2.40. Свойства компонента CarSource

- CarMoveTo – Управление движением машин. Автомобиль может ехать, только когда он находится в блоке CarMoveTo. Автомобиль пытается рассчитать путь от своего текущего места до указанного места назначения, когда поступает в блок CarMoveTo. Одним из важнейших свойств элемента является цель движения. В качестве цели движения могут выступать: дорога, парковка, автобусная остановка или стоп-линия. Указанное место назначения должно находиться в той же дорожной сети, что и автомобиль. Если от текущего местоположения автомобиля к указанному месту нет пути, автомобиль покидает блок через порт outWayNotFound.

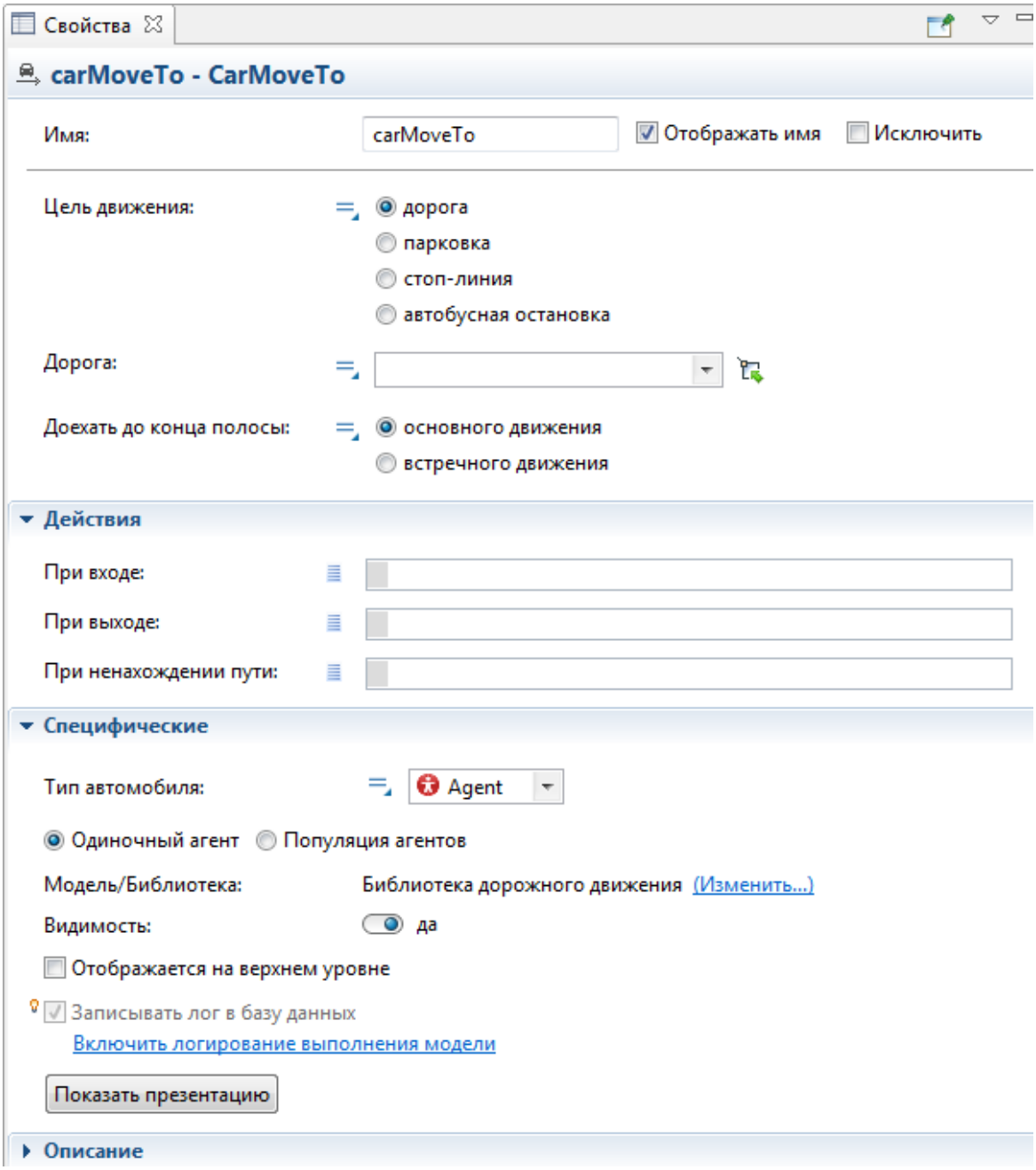


Рис. 2.41. Свойства блока CarMoveTo

Удаление машины из модели осуществляется элементом Car Dispose.

TrafficLight – моделирует светофор (оптическое устройство, предназначенное для регулирования движения на автомобильных, пешеходных перекрестках, а также других сложно контролируемых участках дорожного движения). В свойствах необходимо указать, где данный светофор будет задавать режим работы. Существуют следующие варианты:

- режим работы для стоп-линий перекрестка (подразумевается, что в модели есть перекресток и далее происходит настройка его стоп-линий);
- соединителей полос перекрестка;
- заданных стоп-линий.

Выбрав необходимый вариант, далее производится настройка фаз светофора (выбираются цвета фаз для каждой стоп-линии и указывается их длительность).

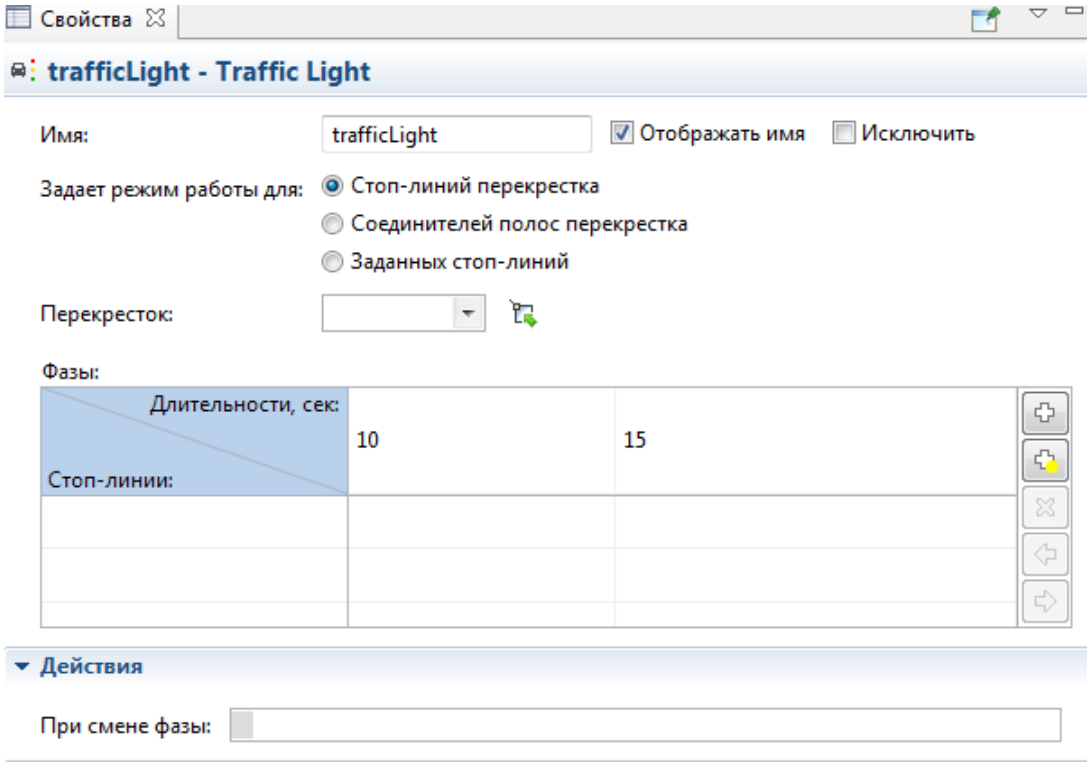


Рис. 2.42. Свойства TrafficLight

Предположим, что, к примеру, представленному на рис. 2.31, необходимо добавить дорогу и светофор, который работал бы синхронно со светофором, описанным с помощью диаграммы состояний. Поместим на рабочую поверхность дорогу и две стоп-линии, как это показано на *рис. 2.43*.

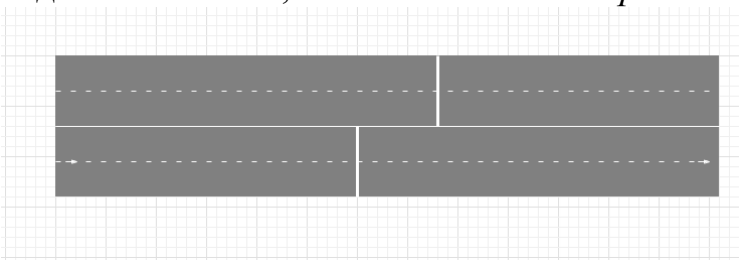


Рис. 2.43. Дорога со стоп-линиями

Стоп-линии должны располагаться на достаточном расстоянии от начала (и, соответственно, от конца) дороги, иначе машины не успеют затормозить, и светофор, заданный объектом TrafficLight, работать не будет.

Поместим данный объект на рабочую область. В свойстве «Задает режим работы для» выберем «заданных стоп-линий» (рис. 2.44).

После этого имеется возможность добавлять стоп-линии (метка (1) на рис. 2.30) и фазы светофора (метка(2) на рис. 2.44).

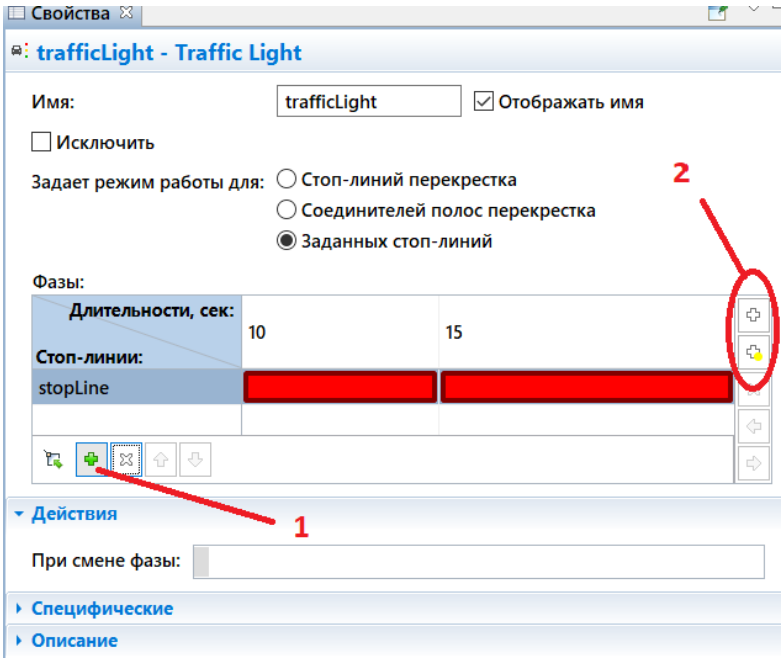


Рис. 2.44. Настройка свойств TrafficLight

Чтобы синхронизировать работу светофора со светофором, смоделированным ранее, перейдем в диаграмму состояний, и в действиях при входе в очередное состояние вызовем метод SwitchToNextPhase() для объекта trafficLight (*рис. 2.45*). При этом длительности фаз, которые задаются в свойствах TrafficLight, должны быть установлены в бесконечность (infinity).

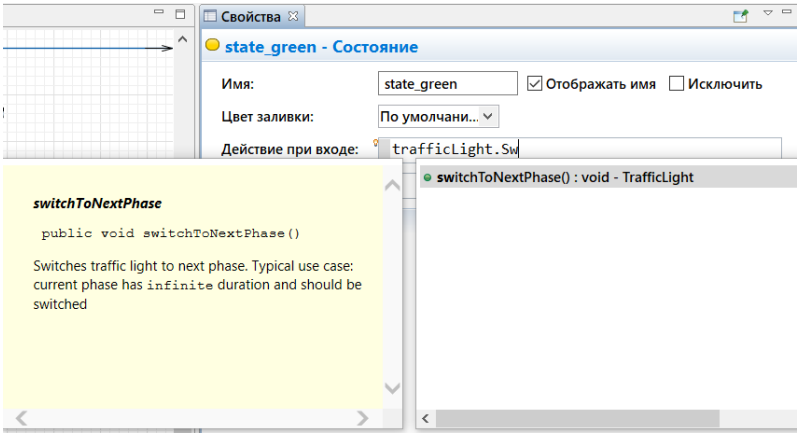


Рис. 2.45. Синхронизация объекта TrafficLight со светофором, смоделированным ранее

2.4. Системная динамика

Системная динамика представляет собой совокупность принципов и методов анализа динамических управляемых систем с обратной связью и их применения для решения производственных, организационных и социально-экономических задач. В системах поддержки принятия решений применение системной динамики позволяет объединить несколько функциональных пространств организации в одно целое и обеспечить организационный и

количественный базис для выработки более эффективной управленческой политики. Три достижения, приобретенные в основном благодаря разработкам в области вооружений, сделали возможным применение системной динамики:

1) Успехи в проектировании и анализе систем управления с обратной связью.

2) Прогресс в методах компьютерного моделирования и развитие вычислительной техники.

3) Накопленный опыт в моделировании процесса принятия решений.

Философия системной динамики заключается в предположении, что организация более эффективно представляется в терминах, лежащих в ее основе потоков, нежели в терминах отдельных функций. Потоки людей, денег, материалов, заявок и оборудования, а также интегрированных потоков информации могут быть выявлены во всех организациях.

Как потоковые диаграммы, так и системы уравнений выражают управленческие связи с помощью двух категорий: накопителей (уровней) и потоков (темпов). Накопители представляют собой такие объекты реального мира, в которых сосредотачиваются некоторые ресурсы: знания (идеи), фонды, источники рабочей силы и т.п. Потоки – это все активные компоненты системы: потоки усилий (попыток), информационные потоки, расходные платежи и т.п.

Библиотека системной динамики имеет следующий вид (*рис. 2.46*).

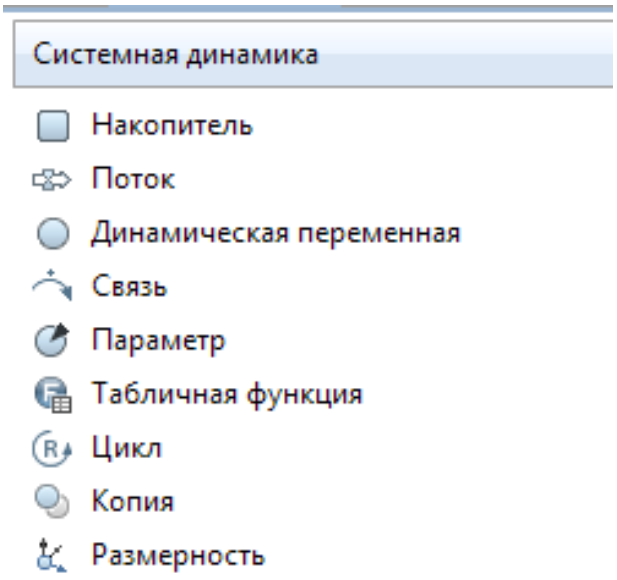


Рис. 2.46. Библиотека системной динамики

Рассмотрим функционирование данных блоков совместно с основными принципами системной динамики.

Основные принципы при создании модели с использованием системной динамики:

1. Дифференциальное уравнение в системной динамике описывается при помощи накопителя, потока, а также динамической переменной или параметра. Накопитель описывает непосредственно производную, динамическая переменная или параметр – переменные, составляющие правую часть уравнения.

2. Выбор динамической между динамической переменной и параметром зависит от того, можем или не можем управлять этим элементом. Если управление возможно, то необходимо использовать параметр, если нет, то это динамическая переменная.

3. Правая часть дифференциального уравнения моделируется потоком. Если правая часть уравнения имеет знак плюс, то она моделируется входящим потоком, если минус, то исходящим. В свойствах потока имеется уравнение, которое описывается с помощью переменных и параметров. При этом, необходимо создать связи от каждого такого параметра/переменной к потоку.

4. Если некоторая переменная или параметр ни от чего не зависит и не будет меняться, то в его свойствах следует пометить, что это константа.

5. Если параметр/динамическая переменная зависит от какого-то количества других параметров и/или динамических переменных, то от этих объектов к данному параметру/динамической переменной необходимо провести входящие связи. После этого в уравнении, заготовка для которого представлена в свойствах, необходимо описать вид зависимости от всех входящих элементов.

Пусть некоторый накопитель имеет один входящий и один исходящий поток. Поместим их на рабочую область, как показано на *рис. 2.47*.

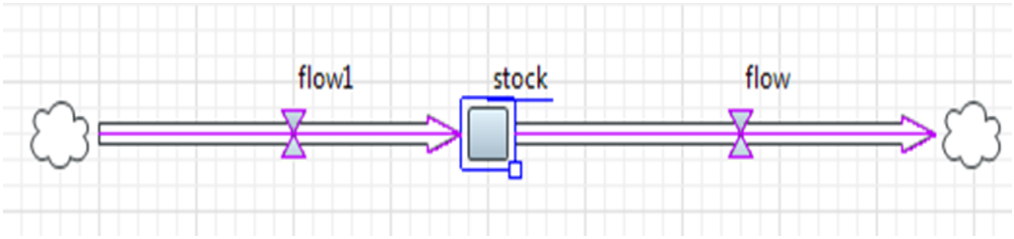


Рис. 2.47. Накопитель с одним входящим и одним исходящим потоком

При этом в свойствах накопителя автоматически сформируется соответствующее дифференциальное уравнение (*рис. 2.48*).

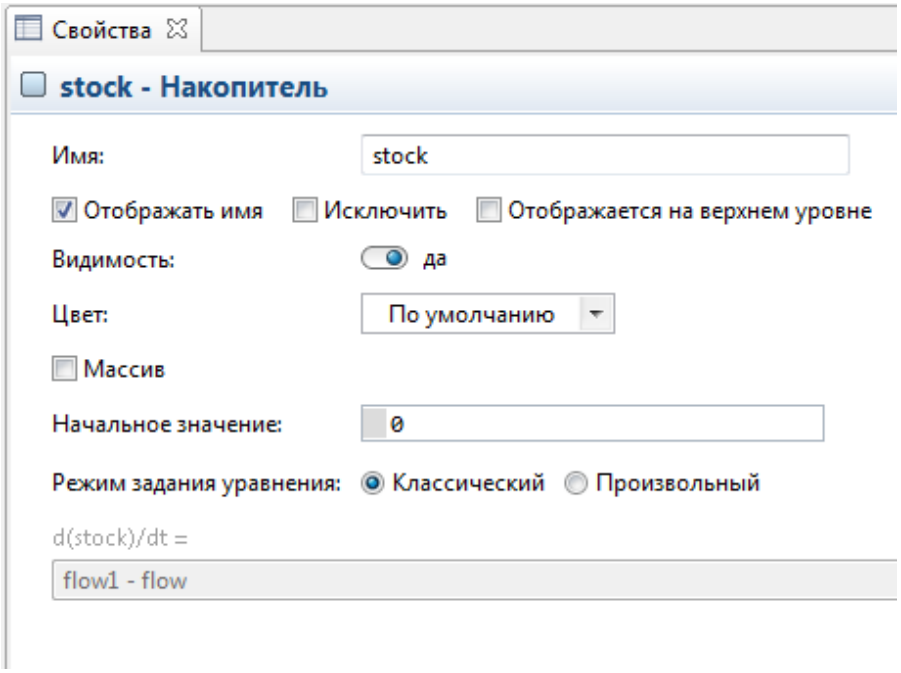


Рис. 2.48. Дифференциальное уравнение, описывающее накопитель

Согласно принципу 3, входящий поток flow1 был взят со знаком «+», а исходящий – со знаком «-».

Рассмотрим теперь уравнение, описывающее поток. Это обычное алгебраическое уравнение, зависящее от каких-либо переменных (параметров). Пусть поток flow зависит от двух параметров: parameter и parameter1. Эта взаимосвязь отражается следующим образом (*рис. 2.49*).

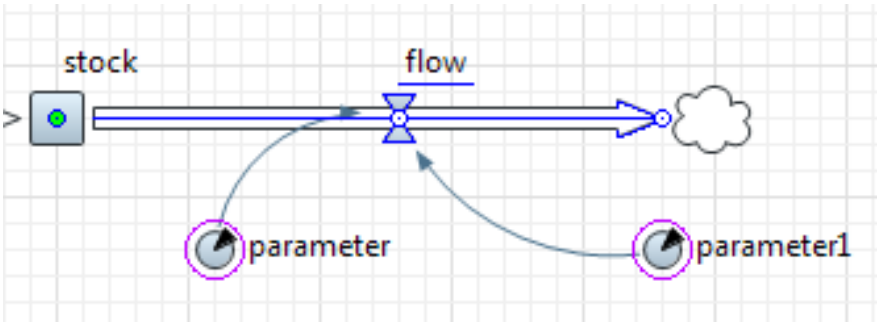


Рис. 2.49. Зависимость потока от параметров

При этом, уравнение зависимости описывается в свойствах потока (*рис. 2.50*).

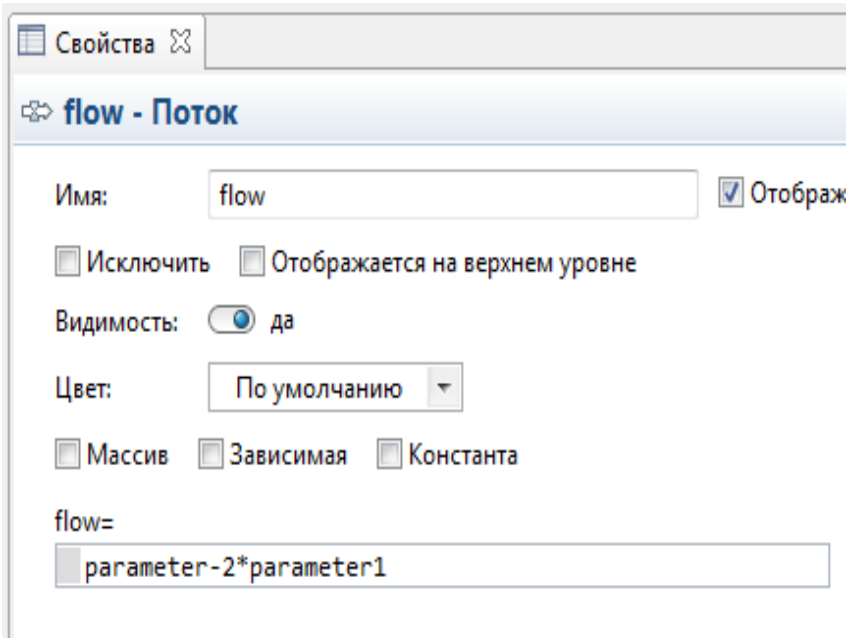


Рис. 2.50. Свойства потока

Пример. Рассмотрим пример реализации продукции по Бассу. Данная модель необходима для исследования динамики приобретения нового товара при предполагаемой эффективности рекламы, количестве контактов людей, численности населения и т.д. 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. Процесс приобретения нового продукта под влиянием убеждения его владельцев чем-то похож на распространение эпидемии.

Пусть имеется две категории людей:

- потенциальные покупатели;
- покупатели.

Интенсивность приобретения нового товара описана следующей системой дифференциальных уравнений:

$$\frac{dPA}{dt} = -AR;$$

PA(0)=TP;

$$\frac{dA}{dt} = AR;$$

AFA=PA·AE;

$$AFW = A \cdot CR \cdot AFr \cdot \frac{PA}{TP};$$

AR=AFA+AFW;

Здесь:

PA (*Potential Adopters*)– потенциальные потребители продукции;

A (*Adopters*)– потребители, которые уже купили продукт;

AR (*Adoption_Rate*) – поток, моделирующий процесс потребления (интенсивность процесса приобретения продукта);

В модели используются переменные:

AFA (Adoption_From_Adversiting)- число потребителей продукта, которые купили его благодаря рекламе;

AFW (Adoption_From_Word_of_Mouth) – число потребителей продукта, которые приобрели его благодаря общению с потребителями, которые уже купили этот продукт;

Константы-параметры модели:

- TP (Total_Population)– численность населения;
- CR (Contact_Rate) – число контактов;
- AE (Advertising_Efestiveness) - эффективность рекламы;
- AF (Adoption_Fraction) – эффективность убеждения.

Предположим, в результате анализа получены следующие значения введенных в модель параметров.

	Параметр	Значение
	TP	100000
	CR	100
	AE	0,011
	AF	0,015

В результате получим следующую модель (*рис. 2.51*).

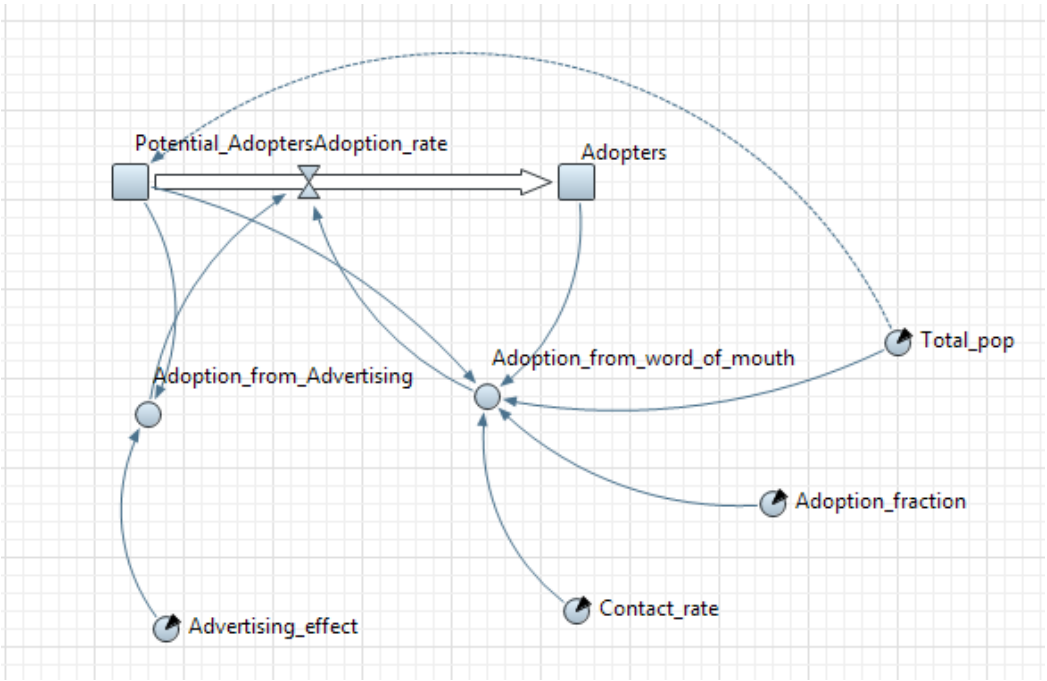


Рис. 2.51. Модель продукции по Бассу

Рассмотрим результаты моделирования. Кликнув по накопителю и раскрыв график, увидим динамику изменения данного накопителя во времени (*рис. 2.52*).

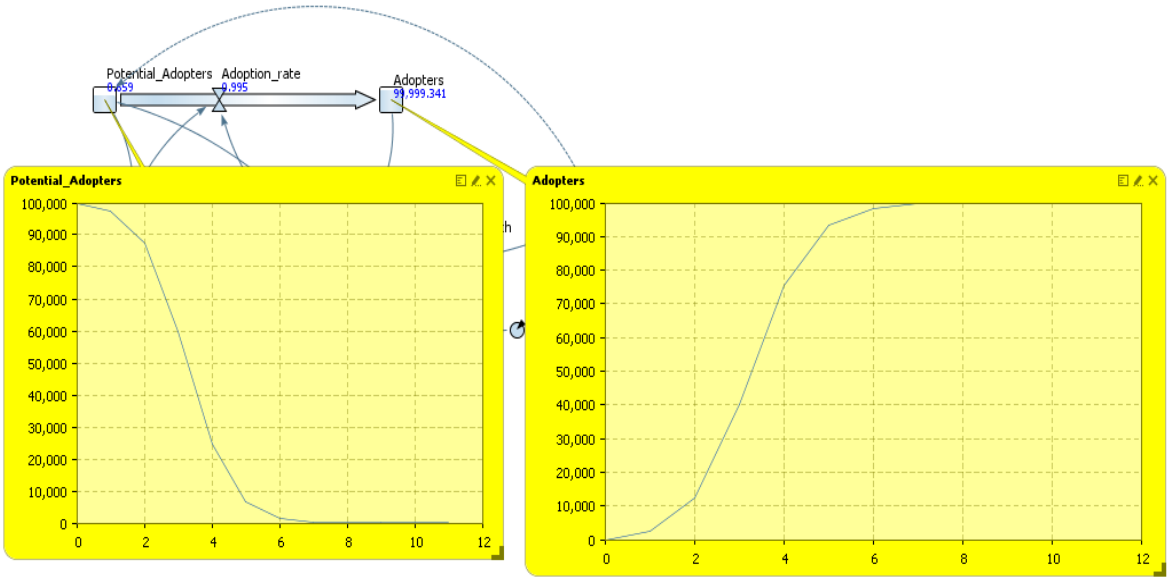


Рис. 2.52. Динамика изменения состояний

В частности, для данного примера видим, что количество потенциальных покупателей экспоненциально снижается, а число потребителей, наоборот, экспоненциально растет.

2.5. Блоки для группового обслуживания заявок

В некоторых случаях возникает необходимость выполнения работ с группой заявок. Так, например, процесс сборки некоторого изделия требует наличия нескольких составляющих и осуществляется для них одновременно. В ряде случаев возникает необходимость организации синхронного продвижения заявок, их объединения или, наоборот, «размножения». Для этого в библиотеке моделирования процессов предусмотрен ряд блоков, предназначенных для имитации группового обслуживания. Рассмотрим эти объекты AnyLogic более подробно.

2.5.1. Assembler

Этот блок позволяет осуществить сборку одного нового агента из определенного числа агентов, пришедших из различных источников (до 5, см. *рис. 2.53*).

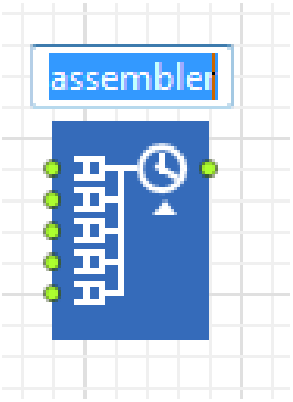


Рис. 2.53. Assembler

Тип нового агента, так же как и его инициализация и типы входных агентов, определяется пользователем. На *рис. 2.54* представлен фрагмент свойств данного блока.

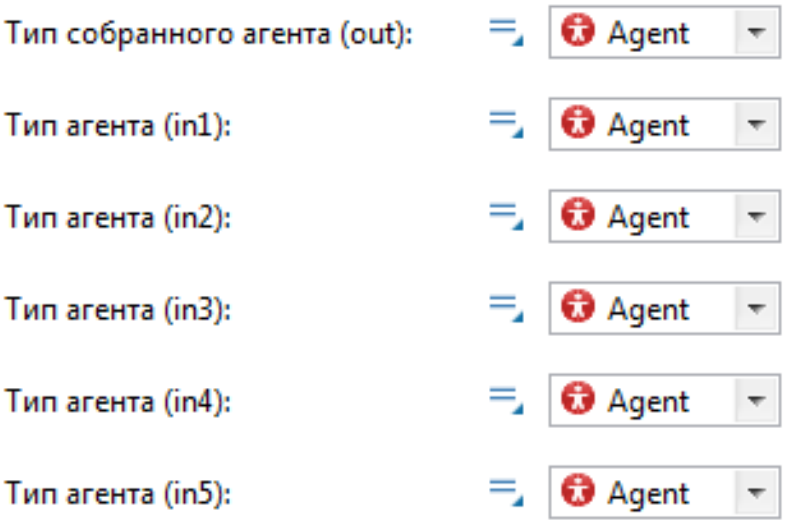


Рис. 2.54. Фрагмент свойств блока Assembler

Число агентов, которые должны поступить на каждый отдельно взятый порт для того, чтобы мог быть создан один новый агент, задается с помощью параметров объекта (Количество 1, Количество 2, и т.д.).

Поступившие агенты ожидают поступления всех необходимых для сборки агентов. Как только новый агент может быть собран, начинается операция сборки. Время сборки задается в параметре Время задержки. Операция сборки может потребовать участия ресурсов. Основные настраиваемые свойства приведены на *рис. 2.55*.

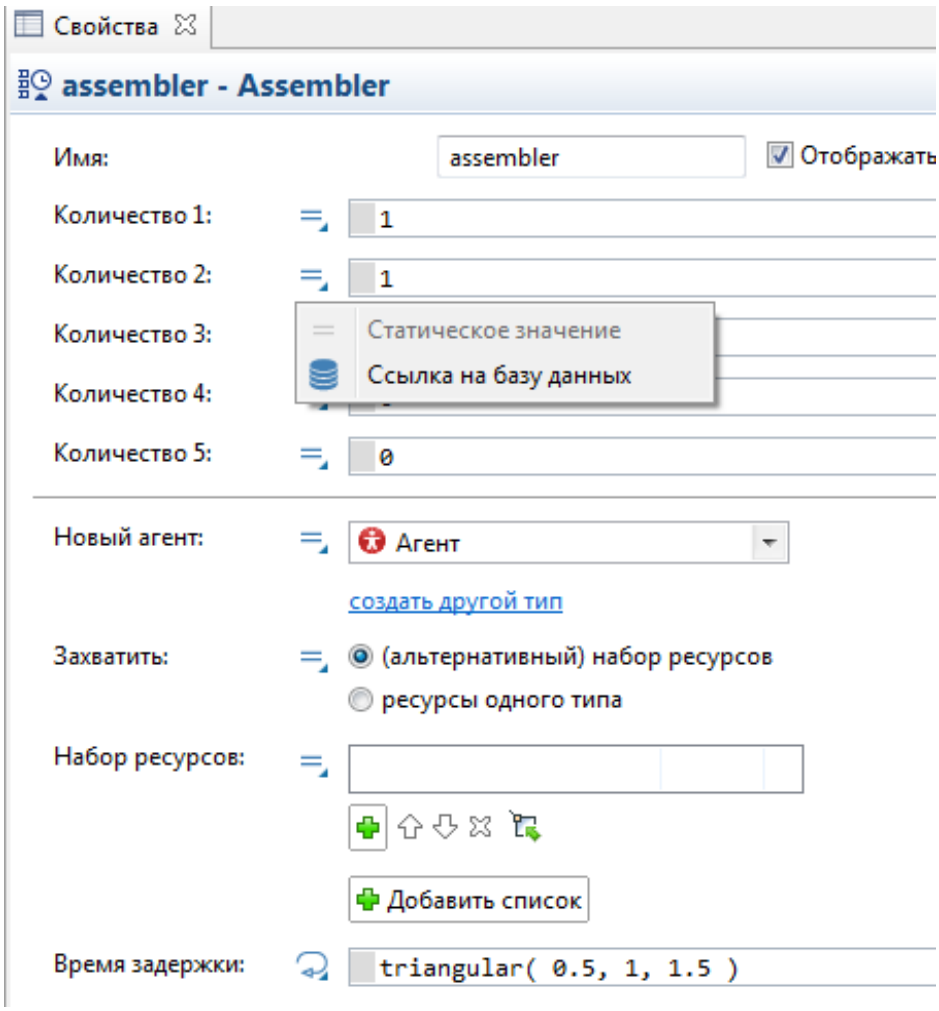


Рис. 2.55. Основные настраиваемые свойства Assembler

Рассмотрим простейший пример работы данного блока. Пусть имеются два потока заявок (заявки первого и второго типа). Процесс сборки требует наличия одной заявки первого типа и одной заявки второго. В этом случае модель будет иметь следующий вид (рис. 2.56).

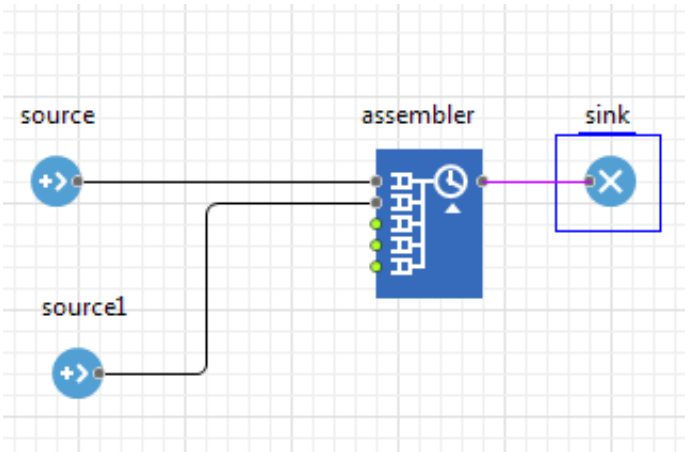


Рис. 2.56. Модель системы

В свойстве «Количество 1» и «Количество 2» у Assembler задаем 1, в поле «Время задержки» - время, затрачиваемое на процесс сборки.

2.7.2 Combine

В некотором смысле данный блок является частным случаем блока Assemble. Он дожидается поступления двух агентов в порты in1 и in2 (в произвольном порядке), а затем создает нового агента и направляет его на выходной порт. Агент, прибывающий первым, хранится внутри объекта, пока не придет другой. Как только прибывает другой агент, созданный агент сразу же покидает объект.

Объект Combine может использоваться для различных целей. Во-первых, он может служить точкой синхронизации, позволяющей одному агенту продолжать свое движение только после прихода другого. Во-вторых, можно использовать объект Combine для воссоединения агента с его копией (или родственным агентом), созданным объектом Split. В-третьих, Combine может служить альтернативой объекту Pickup, если требуется подобрать только одного агента: для этого нужно просто написать agent.addEntityToContents(agent2) в коде «действия При выходе». Свойства данного блока представлены на *рис. 2.57*.

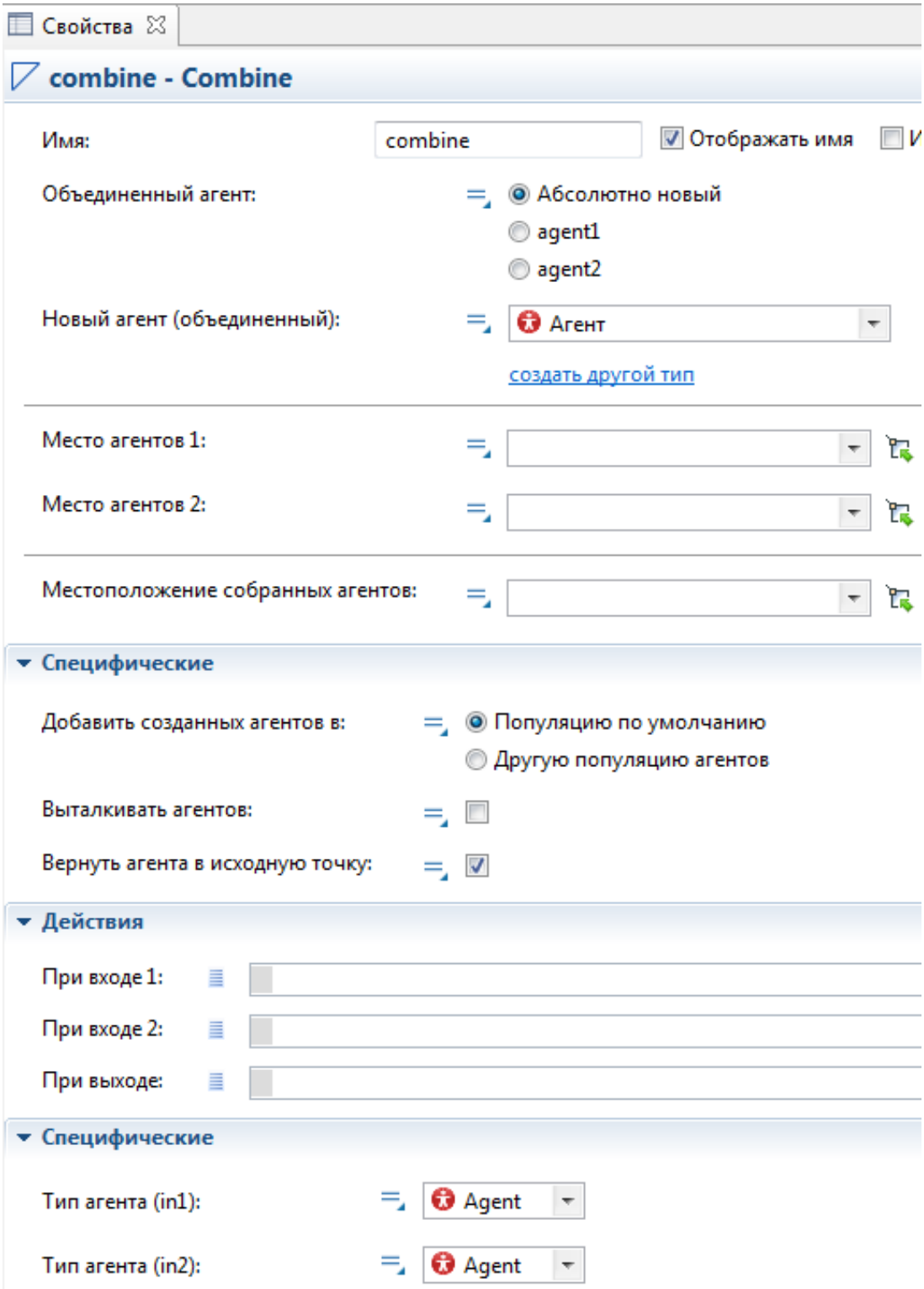


Рис. 2.57. Свойства Combine

2.7.3 Split

Позволяет воздавать копии заявок, поступивших на данный блок. Split имеет один входной порт, откуда поступают на него заявки, и два выходных – для заявки-родителя и для копий. С помощью свойств можно настроить тип заявки-родителя и тип копий (*рис. 2.58*).

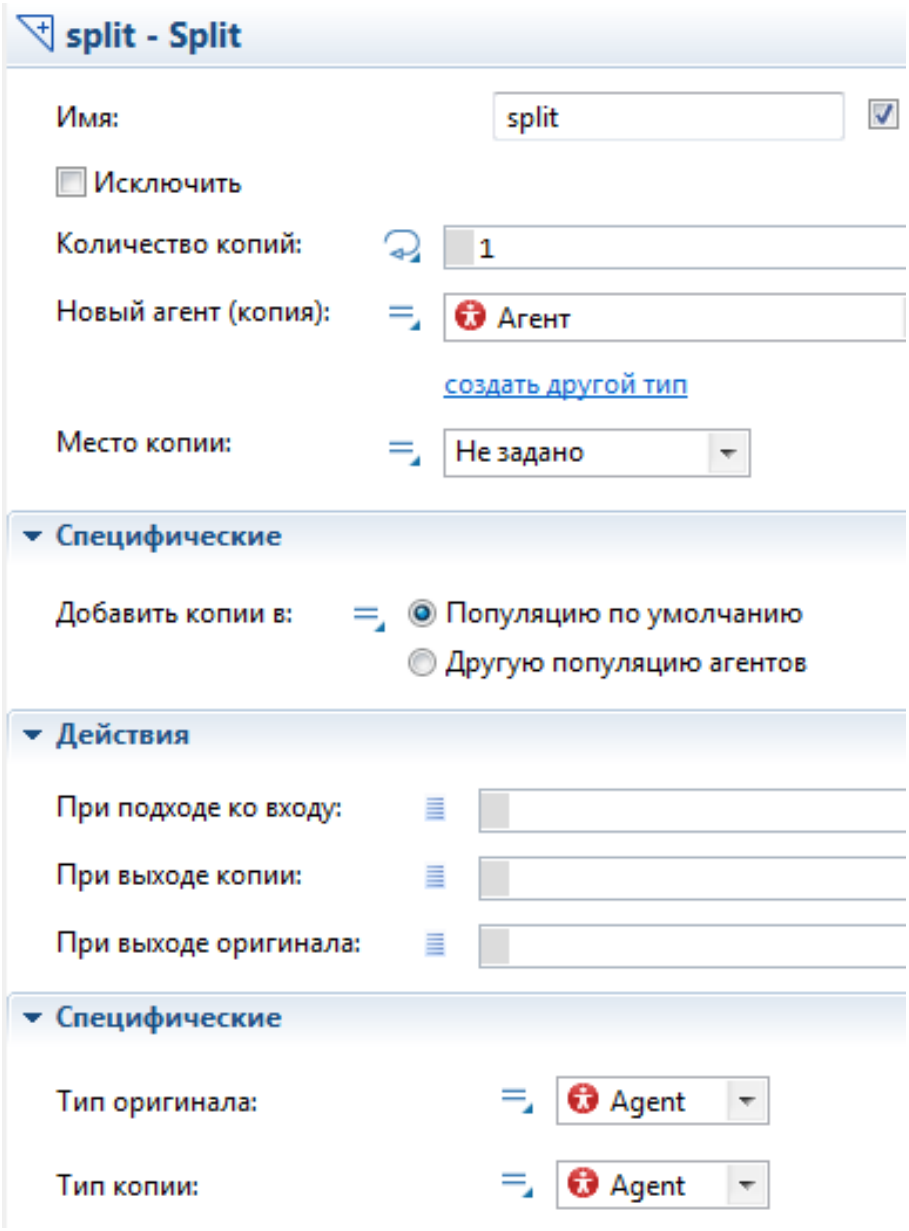


Рис. 2.58. Свойства Split

2.7.4 Match

Данный блок синхронизирует два потока агентов путем нахождения пар агентов, удовлетворяющих заданному критерию соответствия. Агенты, для которых не было найдено пары, для которой выполнялось бы заданное условие, хранятся в двух очередях (по одной на каждый входящий поток агентов). По прибытии нового агента в один из двух входных портов он проверяется на соответствие со всеми агентами, находящимися в очереди, хранящей агентов другого потока. Если соответствие будет найдено, будет выполнен метод onMatch, и оба агента сразу же покинут объект Match через два соответствующих выходных порта. При необходимости поведение очередей

может быть настроено на вашу конкретную задачу с помощью таймаутов, вытеснения, приоритетов и т.д.

За поиск соответствия отвечает свойство “Условие соответствия». По умолчанию задано простейшее условие true. Оно означает, что любые два агента будут соответствовать друг другу, и объект Match будет работать как простой синхронизатор потоков: он будет выдавать пары агентов.

Блок Match имеет следующую структуру (*рис. 2.59*).

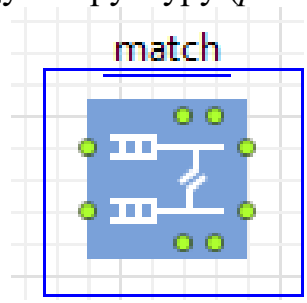


Рис. 2.59. Структура Match

Поскольку для каждого потока агентов (заявок) предусмотрено ожидание, блок снабжен портами выхода, аналогичными структуре Queue. В частности, для каждого потока можно выбрать дисциплину ожидания (FIFO, LIFO и т.д.), ограничить очередь по количеству или по времени, задать вместимость очереди, определить место ожидающих заявок в случае, если используется визуализация моделирования и т.д. Свойства блока Match приведены на *рис. 2.60*.

Пример использования блока Match приведен ниже в теме «Агентное моделирование» (стр. 137, пример 2).

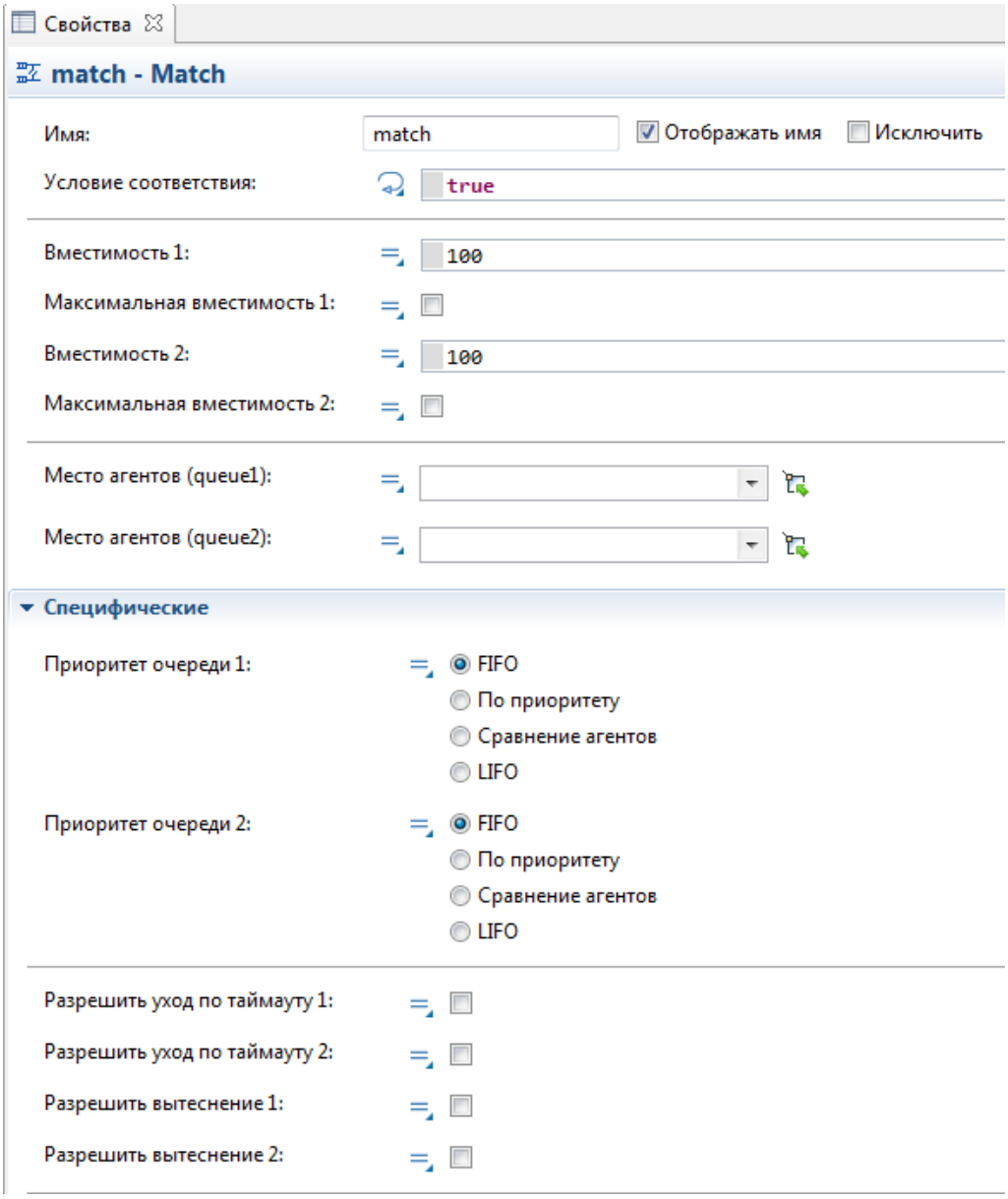


Рис. 2.60. Свойства блока Match

2.8. Разметка пространства

Одним из достоинств среды Anylogic является широкий арсенал средств для визуализации модели. К таким средствам можно отнести возможность задания изображениям заявкам, обслуживающим устройствам, схемам объектов (предприятий, складов, дорог и т.д.). Рассмотрим более подробно арсенал средств для разметки пространства. Соответствующая вкладка из палитры имеет следующий вид (*рис. 2.61*).

Первая часть вкладки позволяет задать место появления или скопления заявок. Это может быть точечный узел, аттрактор (узел, который «притягивает» заявки, например, рекламный шит и т.д.),прямоугольный или многоугольный узел или стеллаж. Стеллаж в первую очередь предназначен для хранения заявок.

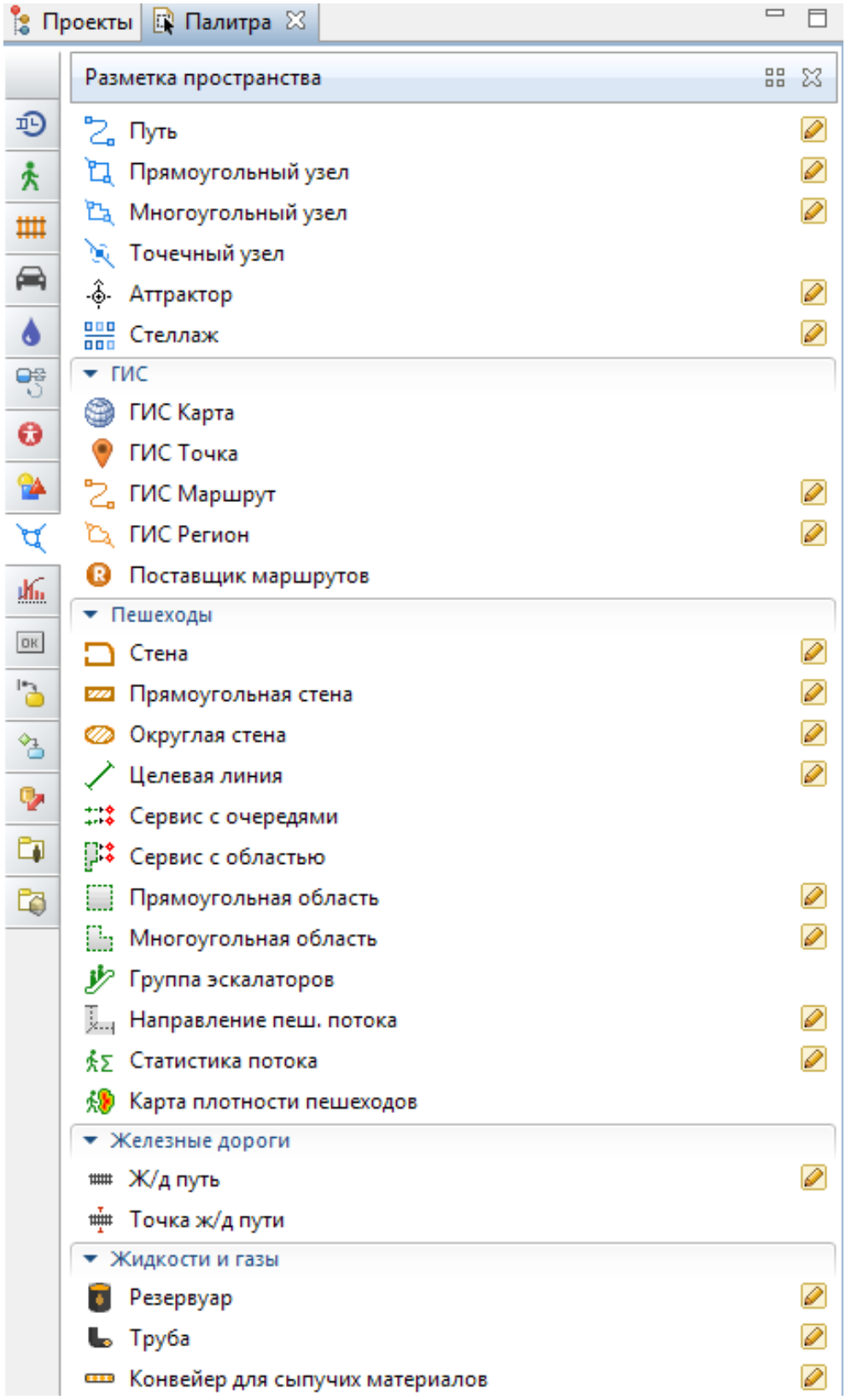


Рис. 2.61. Разметка пространства

Задание заявок с помощью таких объектов предполагает их перемещение в пространстве. Это осуществляется с помощью объекта «путь». Если заявку в процессе моделирования необходимо переместить из узла А в узел В, то узлы А

и В обязательно должны быть соединены путем. Аналогичные требования предъявляются к стеллажу (если некоторые объекты необходимо переместить из стеллажа в определенную точку, то стеллаж и данная точка должны быть соединены путем).

Вторая группа объектов разметки пространства представляет собой объекты для работы с ГИС-картами. Перед тем, как осуществлять работу с такими объектами, на рабочую область необходимо поместить элемент «ГИС карта». Работа с ним осуществляется аналогично работе с обычными картами. Постепенно укрупняя некоторую часть карты, из пространства выделяется определенный регион. Как и при работе с обычными картами, можно выполнить поиск (*рис. 2.62*).

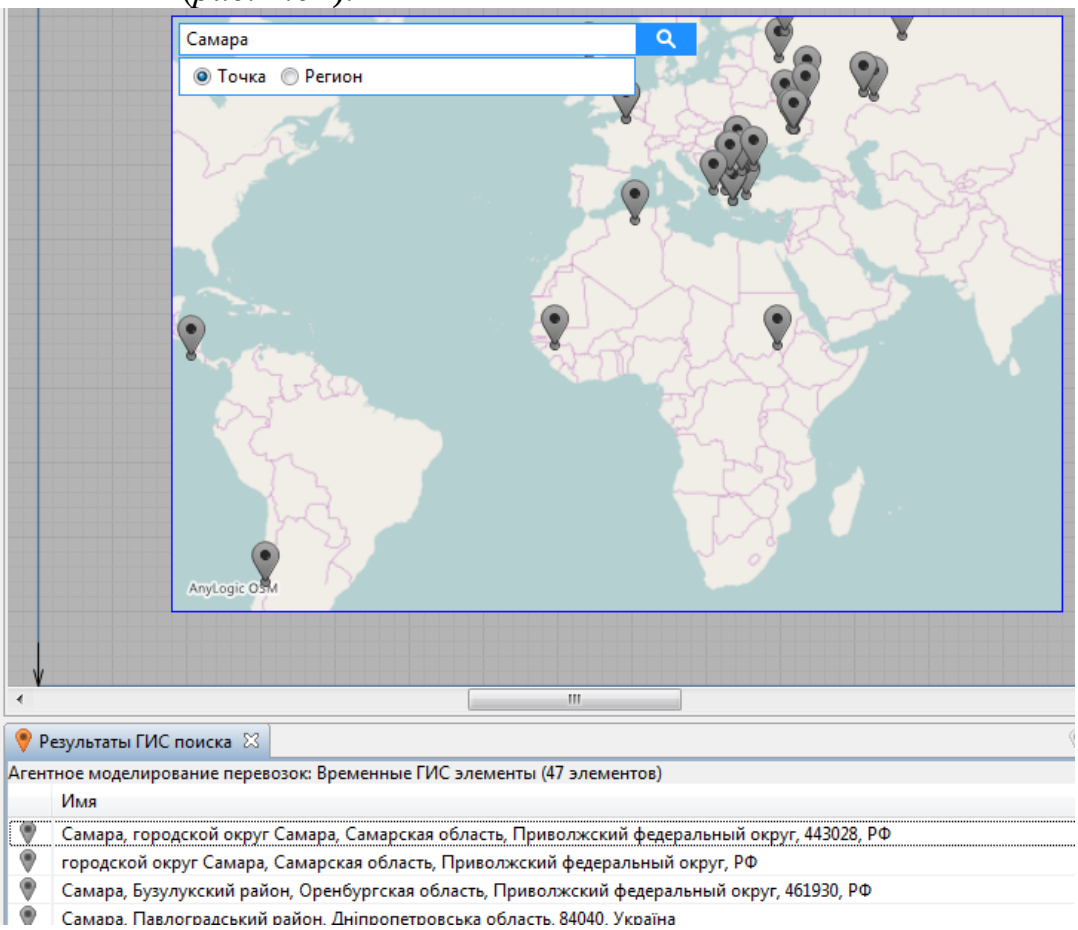


Рис. 2.62. Результаты ГИС-поиска

Внизу будут представлены все ГИС-точки, в которых встречалось интересующее название. В подавляющем большинстве интересующая точка указана первой. Правой щелчок мыши по результату поиска позволяет преобразовать ее в ГИС-точку на карте. После этого остальные элементы можно удалить с помощью соответствующей кнопки (*рис. 2.63*).

ГИС-точку можно поместить на ГИС-карту с помощью объекта ГИС-точка.

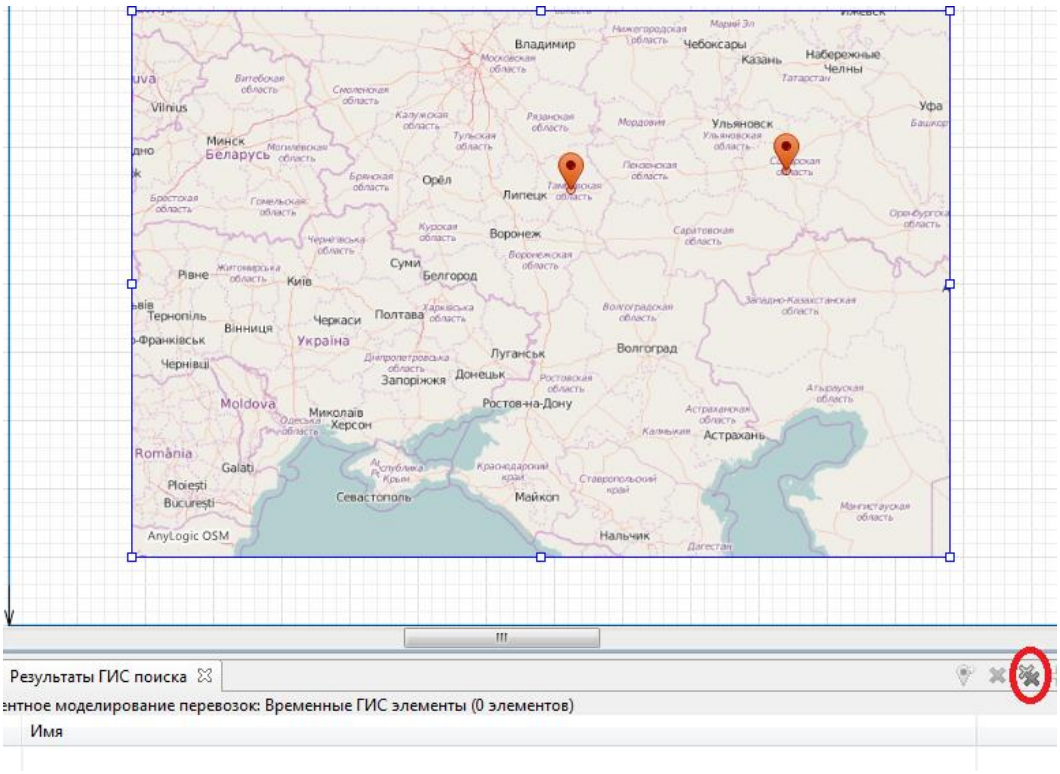


Рис. 2.63. Удаление лишних найденных объектов

В свойства ГИС-карты можно выбрать поставщика тайлов, а также вид дорог, по которым будет осуществляться передвижение (см. *рис.2.64*).

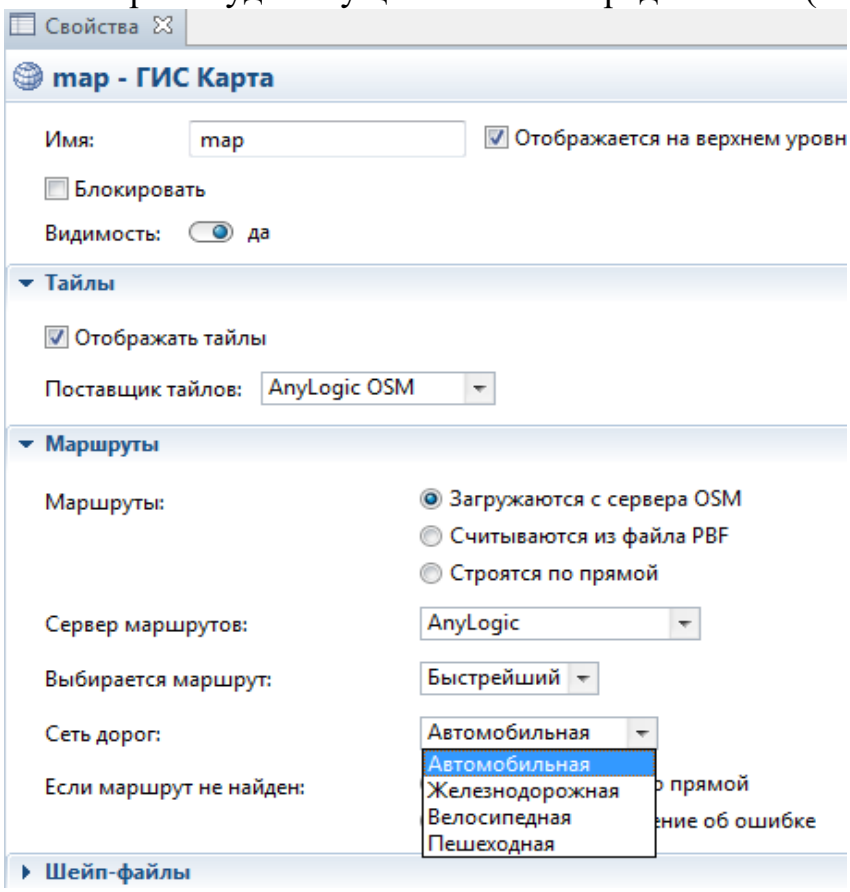


Рис. 2.64. Свойства ГИС-карты

Как правило, ГИС-карты используются для того, чтобы отобразить перемещение агентов или ресурсов. В этом случае при выборе соответствующего объекта или ресурса, необходимо указать его начальное положение с помощью ГИС-точки.

Создание коллекций

Коллекции используются для объединения нескольких однородных объектов. Такое объединение, как правило, упрощает работу с ними. В частности, можно создать коллекцию из ГИС-точек или узлов (или каких-либо других объектов). Для этого необходимо выделить все точки, щелкнуть правой кнопкой мыши, после чего выбрать команду «Создать коллекцию» (*рис. 2.65*).

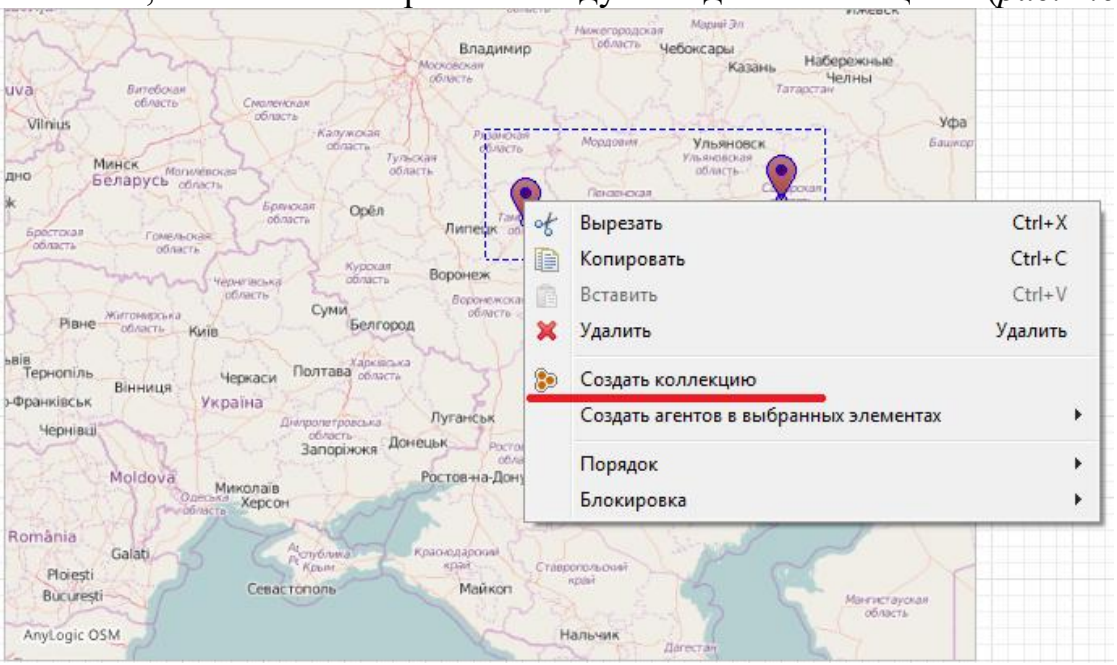


Рис. 2.65. Создание коллекции

Предположим, что была создана коллекция с названием «Потребители», каждый элемент которой содержит ГИС-узел, который будет определить потребителя определенного товара или продукции. Предположим, что необходимо создать агентов- потребителей продукции. В этом случае Создадим популяцию агентов, которую назовем «Узлы», зададим, при необходимости все свойства, а на последнем шаге отметим, что агентов будем создавать позже.

Далее, щелкнув по соответствующей иконке агента, зайдем в свойства и зададим значение:

Потребители.size()

Оно означает, что количество агентов будет определено количеством элементов данной коллекции. В местоположении агента выберем «в узле» и зададим узел (*рис. 2.66*):

Потребители.get(index)

Здесь get – это функция, которая позволяет получить соответствующий массив коллекции потребители; index – номер элемента данного массива.

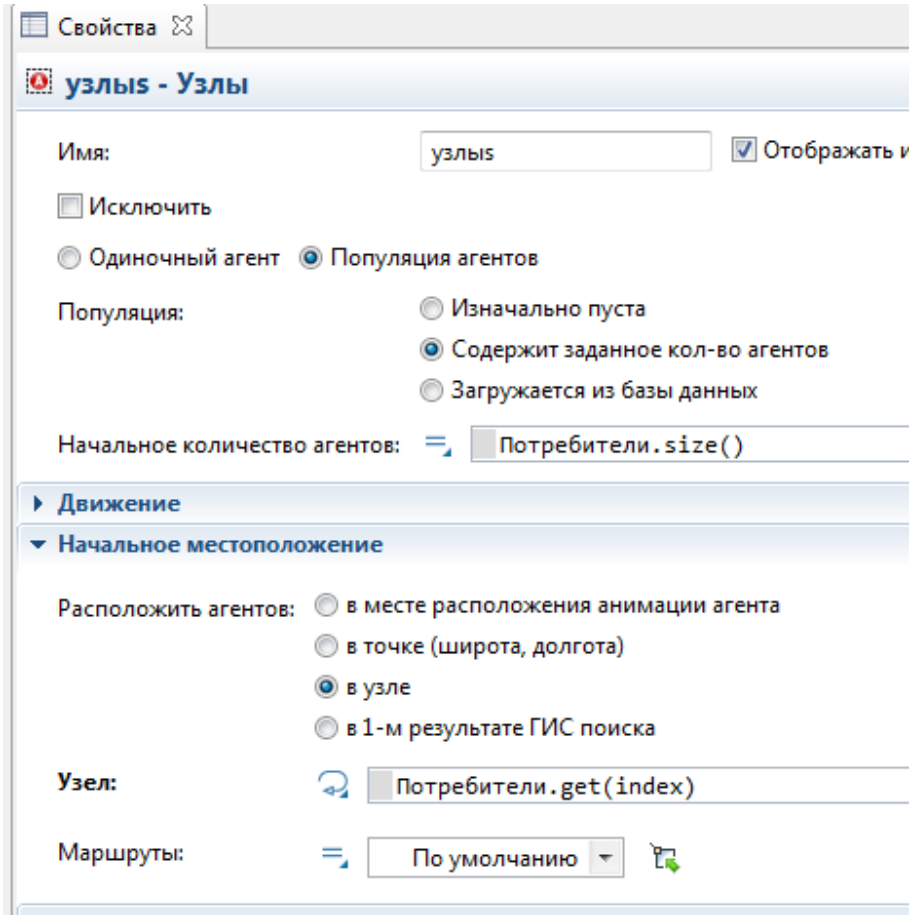


Рис. 2.66. Свойства агентов

После этого каждому агенту будет поставлен в соответствии свой ГИС-узел.

Такой способ задания достаточно удобен тем, что при добавлении или удалении ГИС-точек (в данном случае, потребителей), необходимо просто обновить коллекцию. После этого автоматически добавится или удалится соответствующий агент.

3. МОДЕЛИРОВАНИЕ ОБСЛУЖИВАНИЯ С РЕСУРСАМИ

3.1. Использование ресурсов

Большинство обслуживающих и производственных систем использует разнообразные ресурсы. К ним можно, в частности, отнести, специалистов, выполняющих работу, необходимое оборудование, материалы и т.д. В Anylogic ресурсы бывают трех типов:

- статические (привязаны к определенному местоположению и не могут перемещаться ни самостоятельно, ни принудительно);
- двигающиеся (могут перемещаться самостоятельно);
- перемещаемые (могут перемещаться агентами или движущимися ресурсами).

Каждый набор ресурсов задается блоком ResourcePool (*рис. 3.1*).

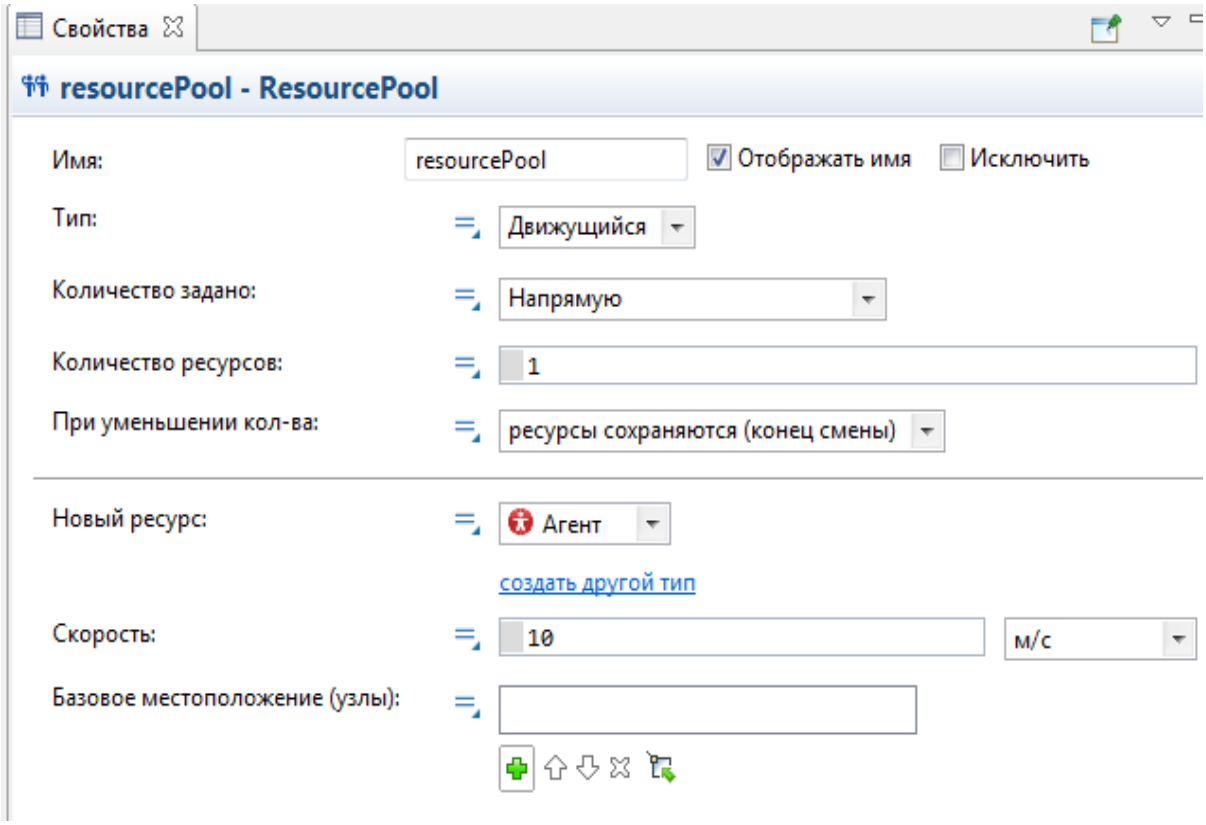


Рис. 3.1. Объект ResourcePool

Ресурсы могут быть заданы:
- напрямую;
- базовым местоположением;
- расписанием;
- расписанием доступности;
- сменами: расписаниями групп;
- планом смен.

Если ресурс задан напрямую, то ниже необходимо определить количество ресурсов и как будет вести себя ресурс в течение работы с ним. Ресурсы могут принадлежать к одному из двух видов:

- ресурсы, безвозвратно расходующиеся на стадии обслуживания;
- ресурсы, сохраняющиеся при завершении обслуживания.

К первому типу относятся, например, материалы, а ко второму – инструменты, оборудование. Кроме того, в свойствах можно задать принадлежность ресурса к некоторому объекту.

Задание ресурсов базовым местоположением целесообразно для статических ресурсов, которые не могут перемещаться.

Кроме того, в anylogic есть объекты типа «Расписание», на основании которых можно также задать ресурсы.

Для ресурса можно использовать стандартный агент или создать нового агента (например, если требуется сформировать изображение).

Базовое положение формируется с помощью разметки пространства (точки, прямоугольные узлы и т.д.).

Если в системе используются несколько разных видов ресурсов, то для каждого из них необходимо создать элемент «ResourcePool».

Если в модели используются ресурсы, то перед использованием они захватываются, а после использования – освобождаются. Для этого в Anylogic предусмотрены объекты SEIZE и RELEASE.

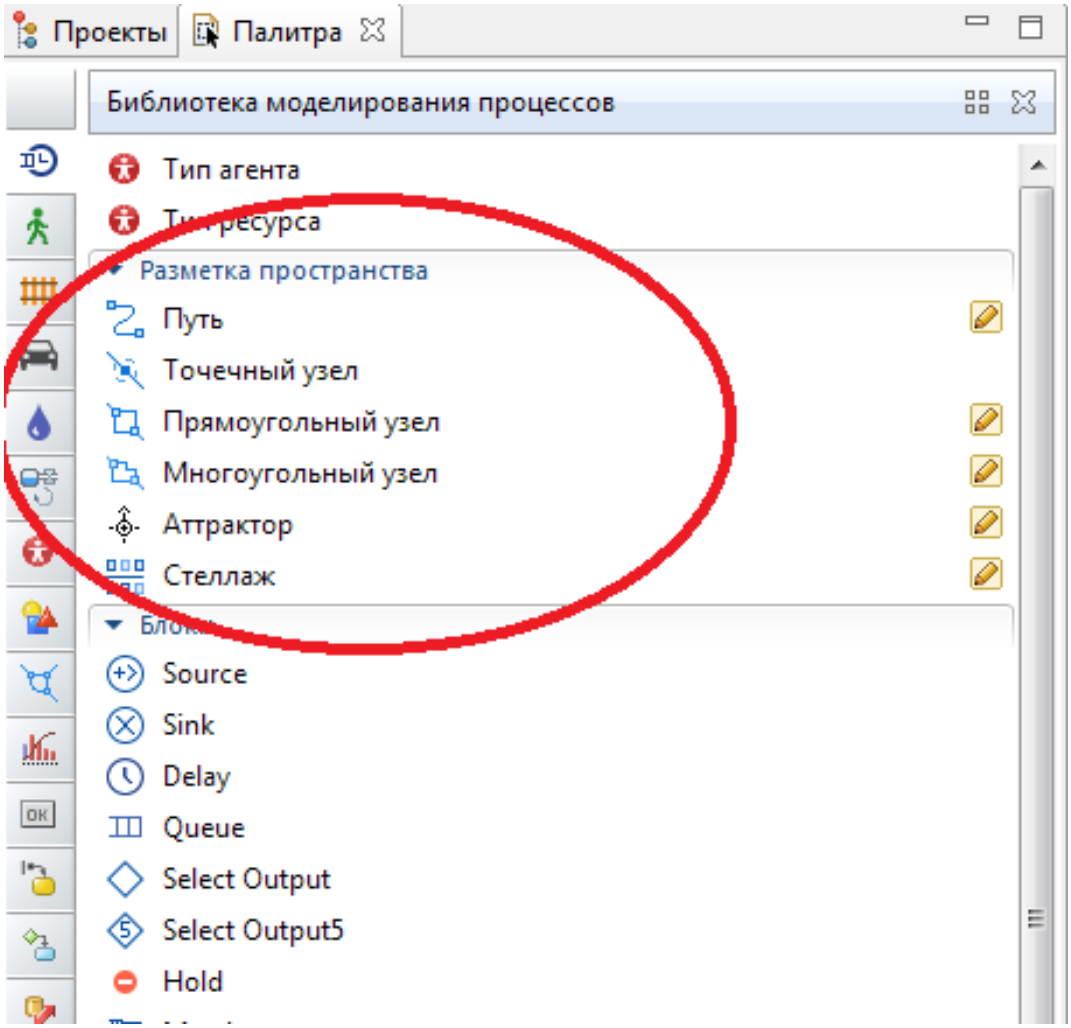


Рис.3.2. Базовое положение ресурсов

Объект SEIZE Захватывает заданное количество ресурсов для выполнения некоторой операции. Он (вместе со всеми портами) представлен на *рис. 3.3*.

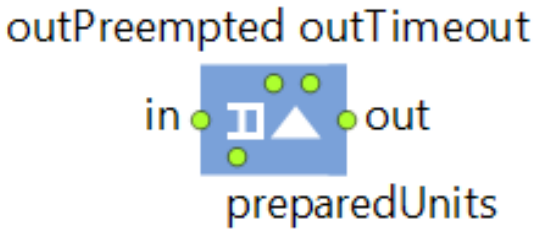


Рис. 3.3. Seize

Заявка может покинуть объект Seize различными способами:

- Обычным способом через порт Out, когда объект, следующий за объектом Queue, готов принять заявку.

- Через порт TimeOut, если заявка проведет в объекте заданное количество времени. В этом случае в свойствах объекта необходимо выбрать «Разрешить уход по таймауту» и в появившейся строке задать время;

- Через порт OutPreempted, будучи вытесненной другой заявкой. В этом случае в свойствах объекта необходимо выбрать «Разрешить вытеснение».

Захватить можно ресурсы одного типа (в этом случае выбирается тип ресурсов и указывается количество) или альтернативный набор ресурсов (в этом случае требуется последовательно добавлять тип захватываемых ресурсов и их количество).

Если движущиеся ресурсы в данный момент располагаются в другом месте, их можно переместить к заявке (или к некоторому узлу), выбрав «Пересылать захваченные ресурсы». В этом случае появляется новое поле «Место назначения», которое может принимать следующие значения:

- Агент (который захватил данный ресурс);
- узел сети;
- аттрактор;
- другой захваченный ресурс (появляется дополнительное поле, в котором можно выбрать ресурс);
- базовый узел захваченного ресурса;
- точка.

При необходимости можно присоединить захваченные ресурсы.

Release – освобождает заданное количество ресурсов после выполнения операции. Можно освобождать:

1. Все захваченные ресурсы любого типа. В этом случае, все ресурсы всех типов, которые были заняты, становятся свободными.
2. Все ресурсы, захваченные данным блоком seize. В этом случае необходимо выбрать блок seize, которые захватил ресурсы, и все захваченные ресурсы в количестве, заданном в блоке seize, станут свободными.
3. Все захваченные ресурсы данного типа. Появляется строка «Объекты ResourcePool», в которой необходимо указать тип ресурсов. После этого все ресурсы данного типа станут свободными.

4. Заданные ресурсы (список типов). Указывается тип освобождаемых ресурсов и количество по каждому типу.

5. Указанное количество ресурсов (выбирается конкретный объект типа ResourcePool и количество освобождаемых ресурсов этого типа).

При освобождении можно указать, что делать с движущимися ресурсами (они возвращаются в базовую точку или остаются на месте).

Свойства блока Release представлены на *рис. 3.4*.

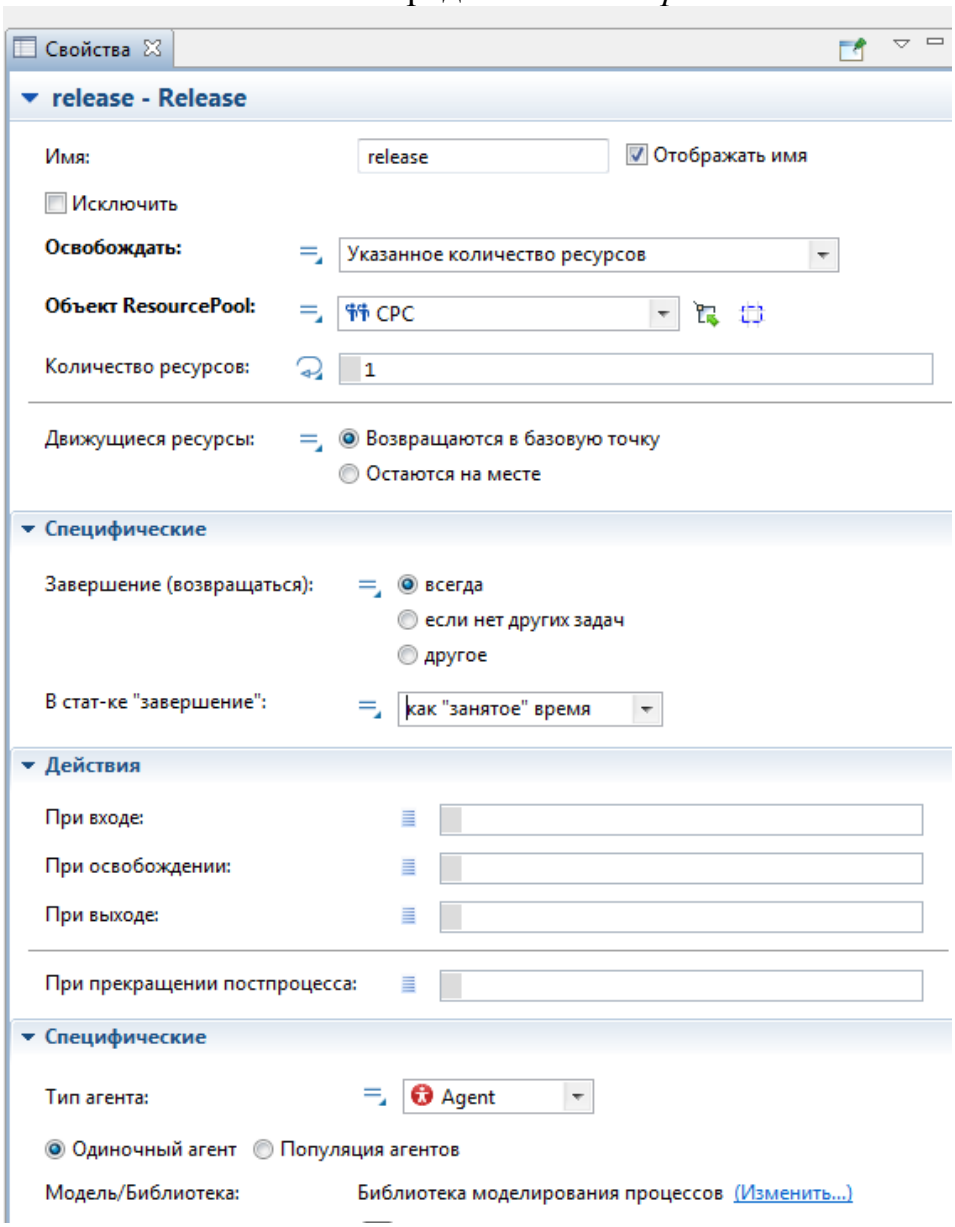


Рис. 3.4. Свойства блока Release

Пример. Имеется производственный участок, в котором работают 3 человека. Процесс изготовления изделия состоит из следующих этапов:

- сборки изделия (выполняется одним (любым) рабочим в течение 30±5 мин);

- его обработки на станке в течение 15±2 мин.

Изделия поступают в среднем через 50±5 мин. Смоделировать процесс обслуживания.

Решение. Создадим два элемента типа «ResourcePool»: один для рабочих(ResourcePool_W); другой – для станка (ResourcePool_M). Для данной постановки задачи пока не принципиально, к какому типу ресурсов будут относиться рабочие и станки. Тем не менее, укажем в типе ResourcePool_W – движущийся; в типе ResourcePool_M – статический.

После поступления ресурсов в модель, сначала необходимо занять ресурсы типа ResourcePool_W (в количестве одной единицы); после чего имитировать обслуживание с помощью блока «Delay». Поскольку после этого этот же рабочий будет обрабатывать изделие на станке, то, не освобождая ресурс ResourcePool_W, необходимо занять ресурс ResourcePool_M. Далее имитируем процесс обработки изделия на станке (блок Delay), после чего объектом Release освобождаем сначала машину, потом - рабочего.

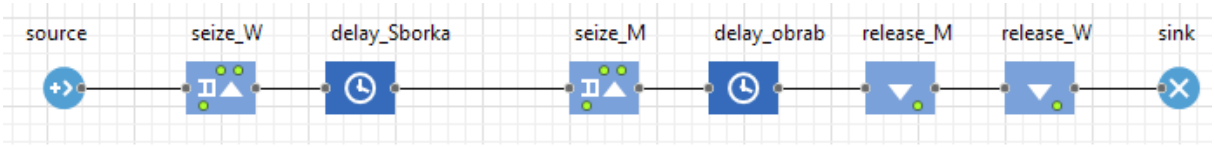


Рис. 3.5. Модель для рассматриваемого примера

Для многих обслуживающих систем тривиальные действия для работы с ресурсами включают в себя следующие команды:

- занять ресурс;
- обслужить в течение заданного времени;
- освободить ресурс.

Как можно увидеть из предыдущего примера, обработка изделия на станке моделировалась именно таким образом. Очевидно, что указанные выше действия можно смоделировать последовательностью команд: Seize – Delay – Release.

Для того, чтобы упросить моделирование таких элементарных систем, вместо трех блоков можно использовать один блок – Service, который сочетает в себе описанный выше функционал.

3.2. Хранение ресурсов

Для хранения ресурсов или агентов в anylogic используются стеллажи. Стеллаж представляет собой элемент разметки пространства. Существует возможность выбора конфигурации стеллажа (количество ячеек, их расположение, количество проходов).

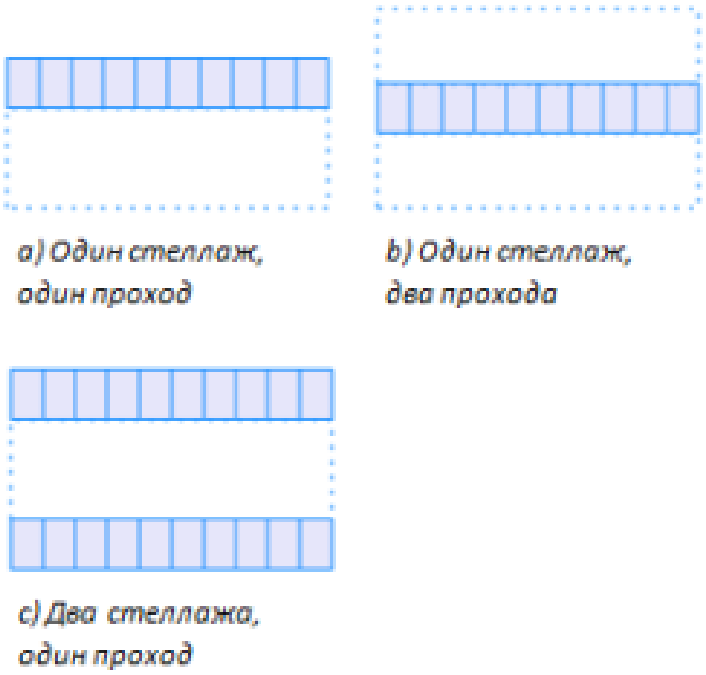


Рис. 3.6. Виды стеллажей

Стеллаж обязательно должен быть соединен путем с объектом, из которого заявки поступают в стеллаж, и объектом, куда будут перенаправлены заявки из стеллажа.

При работе со стеллажом используются два блока, позволяющие загружать элементы в стеллаж и извлекать их. Рассмотрим процесс их функционирования подробнее.

1) Rack Store

Данный объект помещает заявку (агента) в ячейку заданного стеллажа. Если к заявке присоединены какие-то ресурсы, то они перемещаются вместе с данной заявкой. Если заявка перемещается с помощью ресурсов, то RackStore сначала захватывает их, перемещает в местоположение заявки, присоединяет к заявке, а потом перемещает заявку к ячейке, после чего освобождает ресурсы.

2) Rack Pick

Извлекает агента из ячейки стеллажа и перемещает его в заданную точку пространства. При этом для перемещения агента могут использоваться движущие ресурсы. Если агент перемещается с помощью ресурсов, то Rack Pick захватывает их, переносит к ячейке, в которой хранится агент, присоединяет ресурсы к агенту, перемещает агента в заданный узел сети, а затем освобождает ресурсы.

3.3. Перемещение ресурсов

ResourceSendTo - Посылает (перемещает) сетевые ресурсы из их текущего местоположения в заданный узел сети. Могут перемещаться только

движущиеся ресурсы или переносные ресурсы в сопровождении движущихся. Ресурсы, пересылаемые этим объектом, могут находиться в различных местах, но в случае перемещения переносных ресурсов нужно, чтобы в месте нахождения каждого пересылаемого переносного ресурса находился и движущийся объект, который смог бы его переместить (такой движущийся ресурс должен быть также указан в списке перемещаемых объектом ресурсов). Агент покинет этот объект после прибытия последнего перемещаемого ресурса в заданный узел назначения. Каждая группа ресурсов, пересылаемых вместе, движется со скоростью самого медленного из этих ресурсов. Ресурс может быть перемещен:

- в заданный узел сети;
- в заданный аттрактор;
- в местоположение заданного агента;
- к захваченному ресурсу;
- к базовому узлу захваченного ресурса.

Ресурсы будут отображаться на анимации движущимися по кратчайшему из возможных путей от исходной точки до узла назначения. Агент при этом будет отображаться в ее текущем местоположении (в случайном месте внутри текущего узла сети).

MoveTo

Если есть необходимость перемещения агента, то можно использовать объект MoveTo. Если к агенту присоединены какие-то ресурсы, то они перемещаются вместе с агентом. Существуют следующие возможные точки перемещения агента:

- узел сети/ ГИС (при выборе появляется дополнительное поле с возможностью выбора узла);
- аттрактор (появляется поле с возможностью выбора аттрактора);
- захваченный ресурс;
- базовый узел захваченного агента;
- агент/ресурс (появляется поле с возможностью ввода агента);
- точка;
- узел +точка;
- широта, долгота;
- географическое место (появляется строка с возможностью ввода названия места);
- агент, который содержит меня;
- другой ресурс, захваченный моим агентом;
- базовый узел другого ресурса.

Пример [1]. Каждый час на завод приезжает грузовик с заготовками для деталей. На каждом ящике находится по четыре заготовки, готовые к обработке в данном цеху. Все находящиеся на грузовике ящики разгружаются в приемной зоне цеха. Далее эти ящики с помощью автопогрузчиков помещаются в подготовительную зону хранения. По прошествии определенного времени

ящики с заготовками доставляются автопогрузчиками к станку с ЧПУ. Здесь происходит обработка заготовок – производство конечных изделий.

Готовые изделия заново собираются в ящики и перевозятся в другую зону хранения, расположенную у зоны отгрузки.

Решение. Необходимо смоделировать следующие моменты:

- подача ящиков в приемную зону цеха;
- перемещение заготовок в зону хранения;
- перемещение заготовок к станкам и их обработка;
- отгрузка готовых деталей в зону отгрузки.

Поскольку характерной отличительной чертой среды AnyLogic является ориентация на визуализацию моделируемых процессов, рассмотрим данный пример с простейшей анимацией. Как было отмечено ранее, для этого необходимо:

- поместить чертеж объекта (в данном случае, цеха), который будет являться фоновой иллюстрацией, необходимой для дальнейшей анимации;
- провести поверх данного чертежа разметку пространства;
- создать собственно модель с использованием объектов AnyLogic, элементы которой будут перемещаться в моделируемом пространстве.

В качестве фонового рисунка возьмем изображение, представленное в [1].

С помощью разметки пространства, выделим на данном чертеже приемную зону цеха, зону для хранения (стеллаж), зону, где будут находиться автопогрузчики (их базовое местоположение), зону обслуживания (станки) и зону отгрузки. Все зоны необходимо соединить с помощью путей, как показано на *рис. 3.7*.

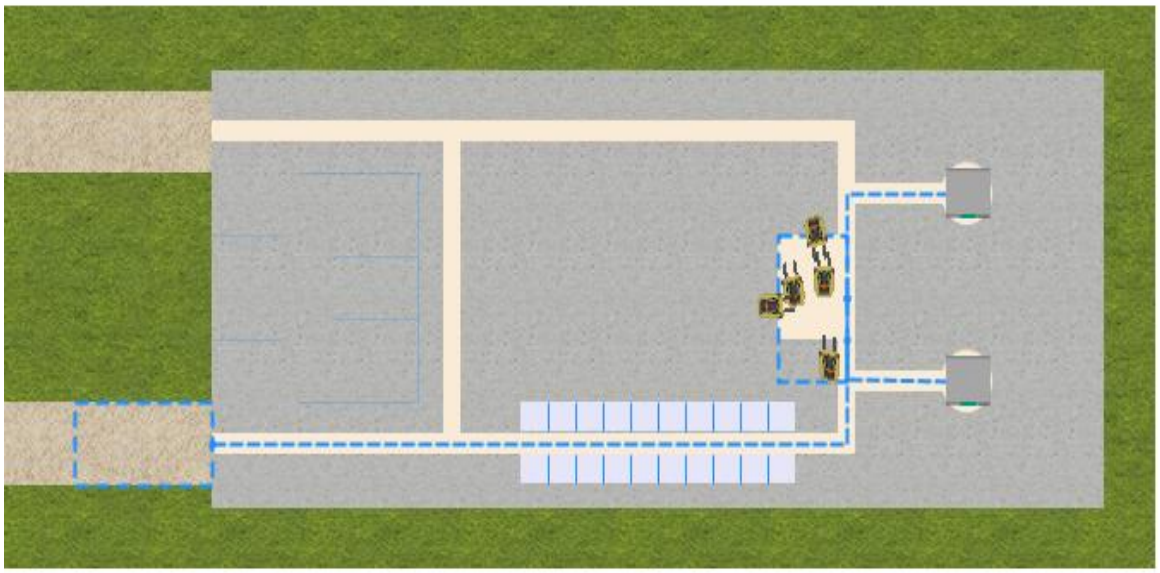


Рис. 3.7. Схема цеха

Моделирование процесса доставки заготовок осуществляется по следующему алгоритму:

- создать заявку (грузовик) через определенное время;

- переместить его к приемной зоне;
- задержать на определенное время, имитируя процесс разгрузки;
- переместить грузовик в точку, из которой он появился;
- удалить грузовик из модели.

Общая схема данного сегмента модели будет следующей (*рис. 3.8*).

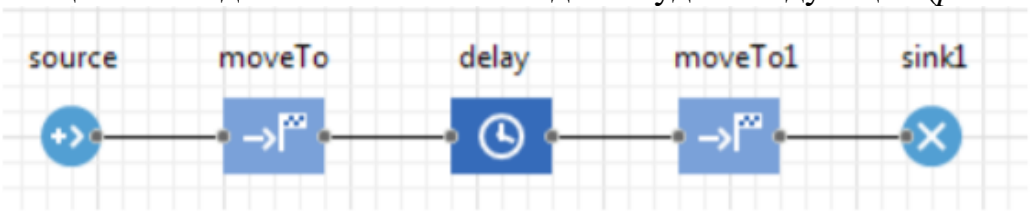


Рис. 3.8. Моделирование процесса поступления деталей

В блоке Source создается заявка (фура). Если необходимо отобразить данный процесс на рисунке, необходимо добавить точку возникновения фуры (на *рис. 3.9* Она указана номером 6) и путь от данной точки до приемной зоны цеха.

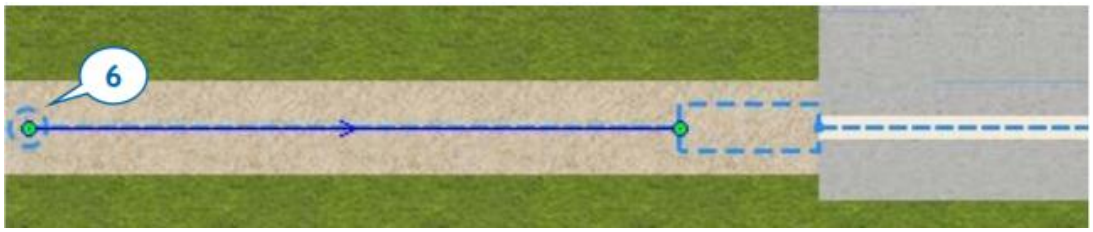


Рис. 3.9. Добавление пути для отображения поступления деталей

Для отображения изображения фуры в блоке Source в свойствах Агент необходимо выбрать «создать новый агент» и далее выбрать необходимую иконку.

Создадим процесс перемещения ящиков в зону хранения, а затем – к станкам для обслуживания. Необходимо создать два типа ресурсов: автопогрузчики (forklifts) и станки (CPC). Для задания станков зададим две точки и зададим станки базовым местоположением (*рис. 3.10*).

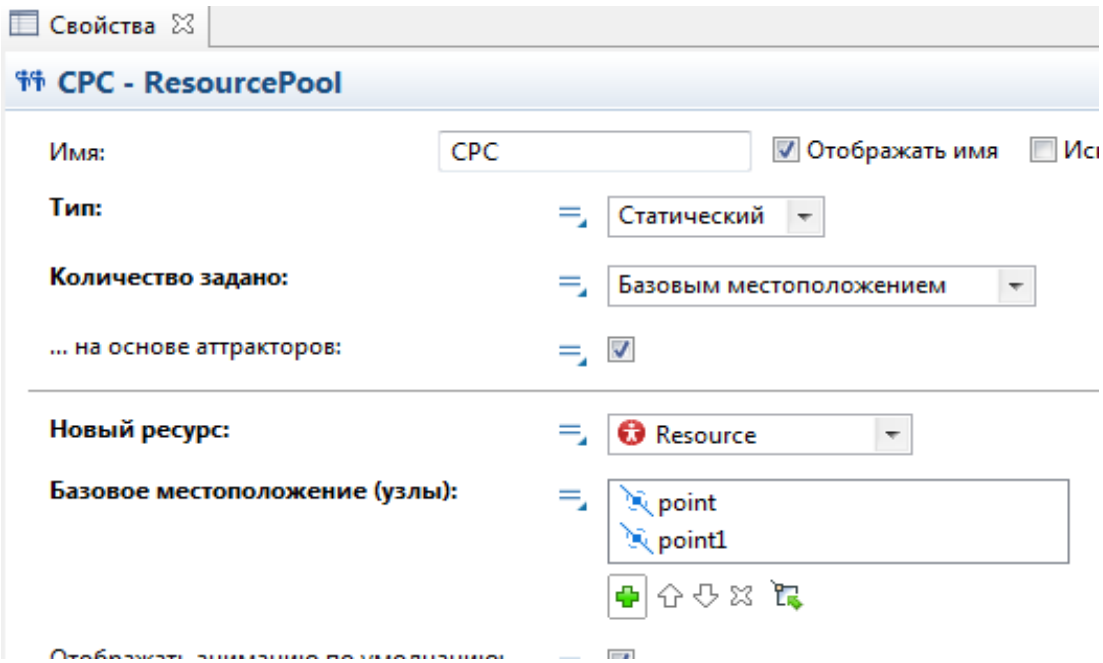


Рис. 3.10. Задание станков

Автопогрузчики будут являться динамическими ресурсами. Они могут быть заданы напрямую (*рис. 3.11*).

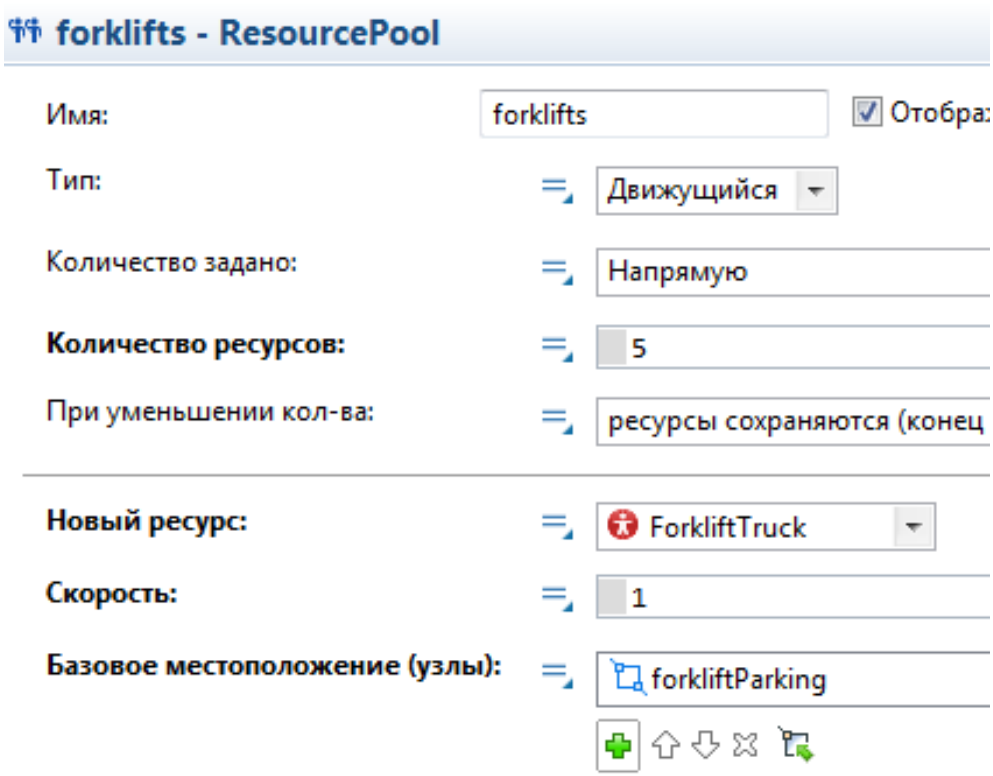


Рис. 3.11. Задание автопогрузчиков

Смоделируем следующие аспекты системы:
- доставка автопогрузчиков к заявкам;

- доставка их к зоне хранения (стеллаж);
- ожидание обслуживания;
- доставка деталей к станкам;
- обслуживание на станке;
- удаление детали из модели.

Схема модели представлена на следующем рисунке (*рис. 3.12*).

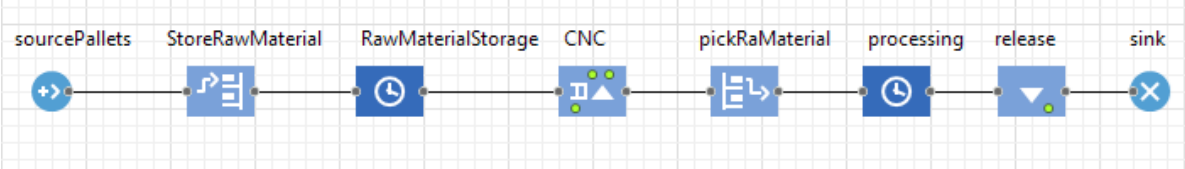


Рис. 3.12. Схема модели

Заметим, что в данной модели с помощью блока Seize занимается лишь ресурс-станок. Ресурс-автопогрузчик занимается с помощью объекта StoreRawMaterial (т.е. блоком RackStore), в свойствах которого указывается, что заявка будет перемещена на стеллаж с помощью ресурса «Автопогрузчик». Аналогично, освобождение автопогрузчика осуществляется с помощью объекта PickRaMaterial (т.е. блоком RackPick). Отметим также, что сначала осуществляется занятие ресурса-станка, а лишь потом – снятие заявки со стеллажа. Такая последовательность необходима из-за того, что заявка может быть извлечена из стеллажа лишь в случае, если станок свободен (стеллаж фактически является местом ожидания заявок своего обслуживания). В связи с этим, необходимо сначала занять станок, что возможно лишь в случае, если он свободен, и лишь потом – извлечение заявки со стеллажа.

Для перемещения готовых деталей к зоне отгрузки необходимо после блока Release, освобождающего станок, занять ресурс-автопогрузчик, и, присоединив его к заявке, переместиться с помощью объекта MoveTo к зоне отгрузки, после чего освободить ресурс-автопогрузчик.

4. АГЕНТНОЕ МОДЕЛИРОВАНИЕ

4.1. Основы агентного моделирования

Одним из новых подходов в моделировании основывается на использовании взаимодействующих агентов.

Под агентом понимается элемент модели, который может иметь поведение, память, историю, контакты и т.д. В качестве примеров агентов могут выступать люди, компании, товары, ресурсы и т.д.

Создание агента обычно начинается с определения его интерфейса для связи с внешним миром и/или с другими агентами.

Начальное состояние и поведение агента может быть реализовано различными способами. Состояние (накопленная история) может быть представлено с помощью переменных или состояния диаграммы состояний.

Агент может вести себя пассивно или активно. В случае пассивного поведения агент может только реагировать на прибытие сообщений или вызов методов. При этом он не имеет собственных событий. Если агент ведет себя активно, то у него существует внутренняя динамика, заданная в виде некоторой последовательности событий. В этом случае для такого агента необходимо разработать событие и/или диаграмму состояний.

Создание агентов

Для создания агентов необходимо перетащить в основное окно соответствующий элемент из библиотеки Агент. При этом можно создать:

- множество однотипных агентов;
- единственного агента;
- просто создать тип агента (*рис.4.1*).

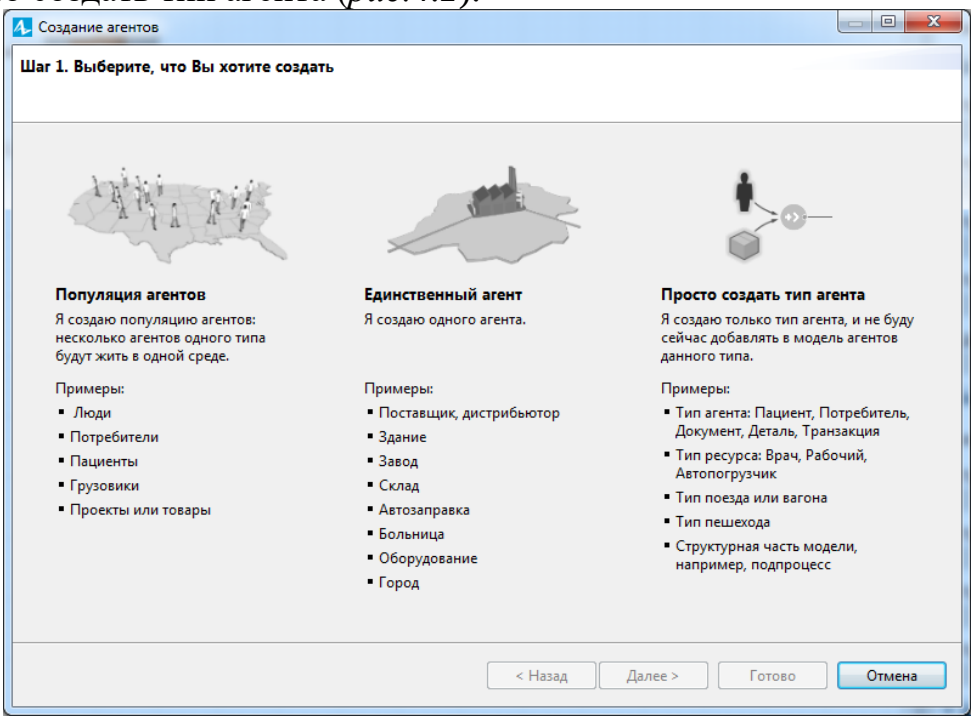


Рис. 4.1. Создание агентов

Агента можно создать «с нуля» или с использованием существующего типа агентов. После этого есть возможность выбора анимации для агента и задания для него множества параметров. Пример задания параметров приведен на *рис. 4.2*.

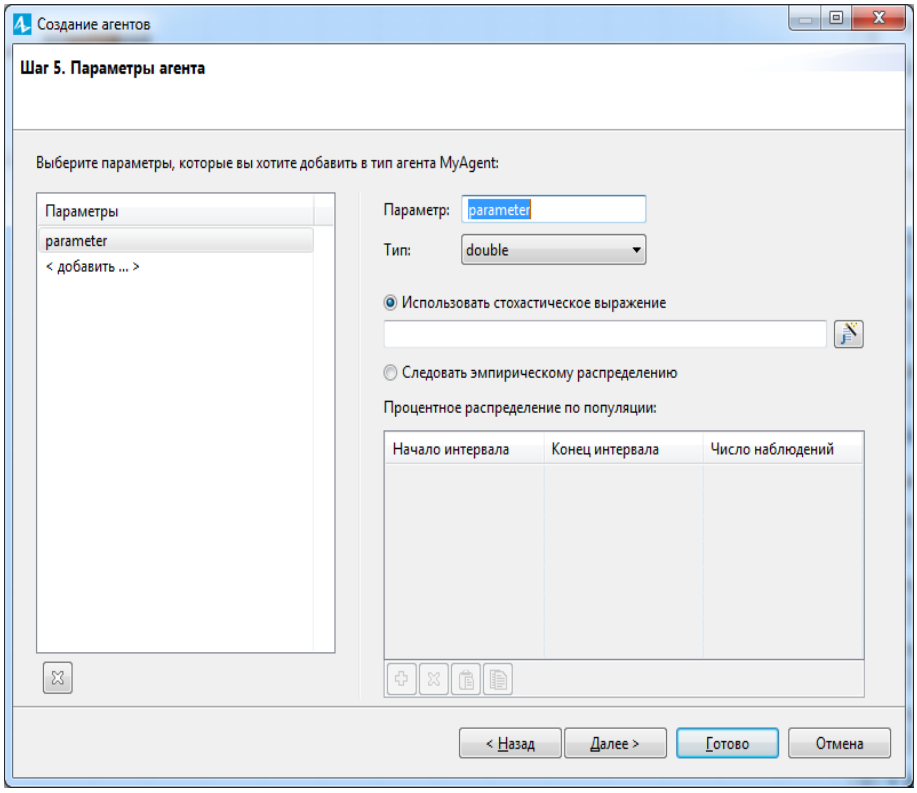


Рис. 4.2. Задание параметров для агента

После этого агент будет виден на основной форме. Кроме того, зайдя на вкладку «Проекты» можно выбрать данного агента двойным щелчком мыши. При этом откроется окно, в котором можно описывать любые действия для данного агента, строить для него диаграммы состояний и т.д.

Типичная архитектура агентной модели в Anylogic – агент Main, на диаграмме которого заданы все остальные агенты. В этом случае Main играет роль среды обитания для всех остальных агентов. Среда задает:

- тип пространства, в котором живут агенты;
- расположение агентов в пространстве;
- сеть контактов агентов, которая может использоваться при их общении друг с другом.

Остановимся более подробно на параметрах агента.

Изменить свойства агента в любой момент времени можно с помощью оператора присваивания вида:

<агент>.<параметр> = значение

В Anylogic доступна локальная переменная Agent, которая ссылается на того агента, который в настоящий момент активен (активизировал данное действие).

Пример.

На вход устройства поступают два потока заявок. Заявки первого типа поступают в среднем через 20с и обслуживаются 10±2 с. Заявки второго типа поступают в среднем через 15с и обслуживаются 8±2 с. Перед устройством имеется неограниченная очередь. Смоделировать работу системы.

Решение. Поскольку длительность обслуживания является индивидуальной для каждой заявки, целесообразно эту длительность отражать в свойствах заявки. Для этого требуется создать специальный тип агентов, имеющий одно дополнительное свойство «длительность». Это будет популяция агентов (поскольку это будет источник заявок).

Создадим модель согласно *рис. 4.3*.

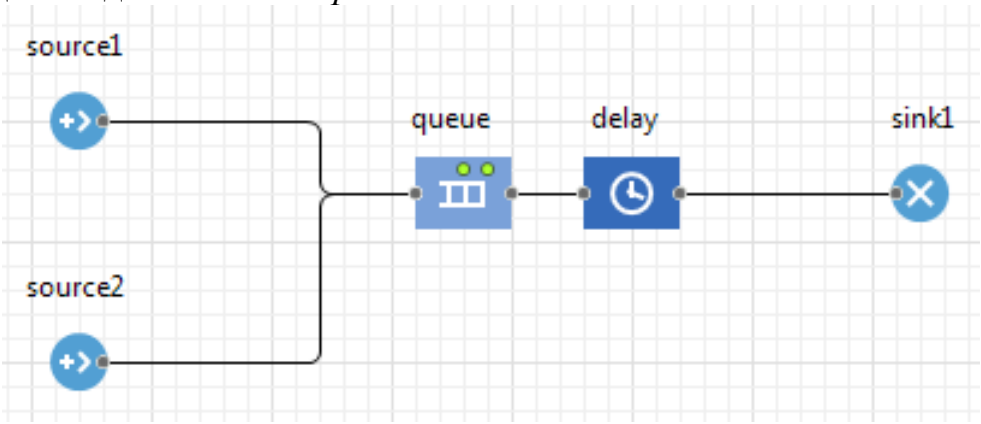


Рис. 4.3. Схема модели

Зайдем в свойства элемента source1 и в действиях выберем «действие при выходе» и напишем код:

Agent.длительность = 10;

Аналогичным образом опишем «действие при выходе» для элемента source2:

Agent.длительность = 8;

Далее необходимо эту длительность задать в блоке delay. Зададим следующее время задержки:

uniform(agent.длительность, 2)

Модель будет работать следующим образом. Источник source1 будет создавать агентов, и, как только они будут покидать данный блок, в свойство агентов «длительность» будет присваиваться значение 10. Аналогичным образом, все агенты, которые будут созданы источником source2, будут иметь в поле «длительность» значение 8. Блок delay будет анализировать свойство «длительность» поступившей заявки. Более точно, данный блок будет задерживать заявку на время, которое записано в ее свойстве «длительность».

4.2. Взаимодействие агентов

Специфика функционирования многоагентной системы заключается в том, что агенты, функционируя, постоянно взаимодействуют между собой и, в

зависимости от полученной информации, принимают те или иные решения. Исходя из этого, важнейшим аспектом реализации многоагентной системы является обеспечение их взаимодействия.

Основным способом организации взаимодействия агентов в anylogic являются сообщения. Агенты взаимодействуют посредством передачи друг другу сообщений. Получив некоторое сообщение, агент выполняет определенное действие, тем самым реагируя на это сообщение. Таким образом, для взаимодействия агентов необходимо обеспечить возможность посылать сообщения и выполнять действия при получении сообщения.

Пересылка сообщений осуществляется посредством функции send. Для нее необходимо указать два параметра:

- сообщение;
- кому это сообщение предназначается.

Сообщение может иметь разные типы. В простейшем случае оно может иметь тип String. Можно использовать или создавать более сложные типы сообщений (например, тип агента, в котором будут содержаться необходимые параметры).

Для формирования реакции на сообщение в anylogic разработан объект «Connections». содержит связи с контактами этого агента и задает настройки взаимодействия. Данный элемент располагается над осью X в графическом редакторе агента (*рис.4.4*).

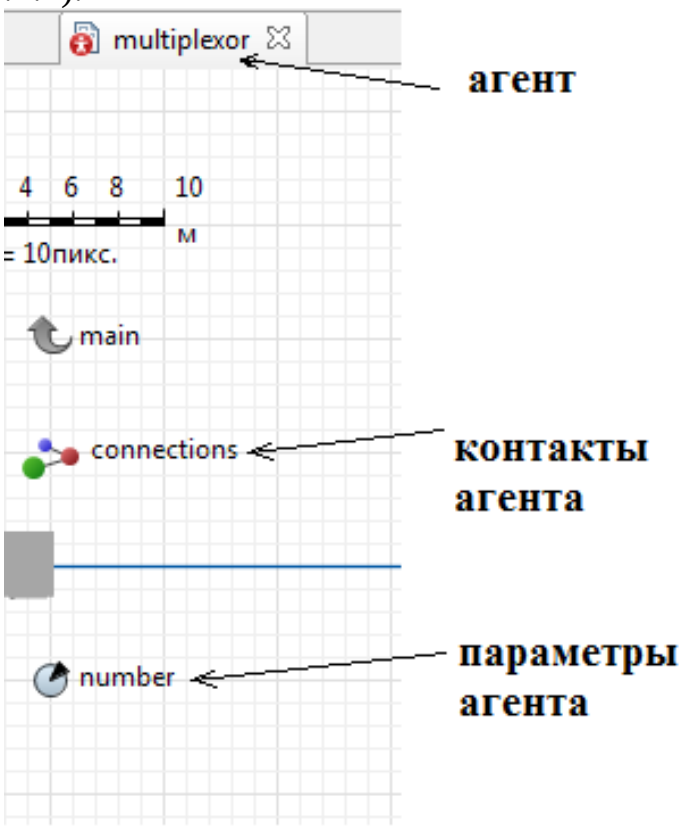


Рис. 4.4. Окно агента

Для задания реакции на получения сообщения необходимо:

1. Открыть диаграмму агента-получателя сообщения.

2. Щелкнуть мышью по его элементу connections.

3. Открыть секцию свойств «Взаимодействие»

4. Ввести соответствующий код в свойстве «Действие при получении сообщения».

5. Реакцию на получение сообщения можно организовать в виде диаграммы состояний или в виде цепочки действий дискретно-событийного моделирования. В первом случае необходимо, чтобы был выбран выбрать пункт «Перенаправлять сообщение в диаграммы состояний» и далее в таблице будут отображены все диаграммы состояний данного агента. Во втором случае действия можно начать с создания источника заявок source или вставить агента в уже существующую диаграмму. В первом случае в свойстве «Действие при получении сообщения» необходимо ввести код «source.inject(1);», а в свойствах объекта source задать прибытие заявок согласно функции inject. В случае вставки агента в свойстве «Действие при получении сообщения» необходимо ввести код enter.take(msg), где msg – данное сообщение.

Пример. Пусть имеется отдел реализации и отдел снабжения. Отдел реализации занимается изготовлением некоторых деталей. Потребность в комплектующих для деталей возникает в среднем через 60-80 часов. В этом случае оформляется заказ отделу снабжения и через 10-15 часов комплектующие будут доставлены.

Решение.

Сформируем два одиночных агента: Снабжение и Реализация. У отдела реализации можно выделить два состояния: нормальная работа и ожидание запчастей. В связи с этим, сформируем для данного агента следующую диаграмму (*рис. 4.5*).

Переход из состояния «Работа» в состояние «Ожидание» будет осуществляться «по таймауту», а именно через 60-60 часов: uniform(60, 80).

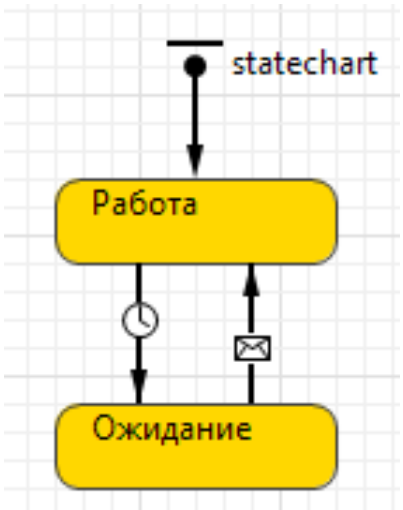


Рис. 4.5. Диаграмма состояний для отдела реализации

Переход из состояния «Ожидание» в состояние «Работа» будет осуществляться при получении сообщения. Пусть в данном случае это будет сообщение «Доставлено».

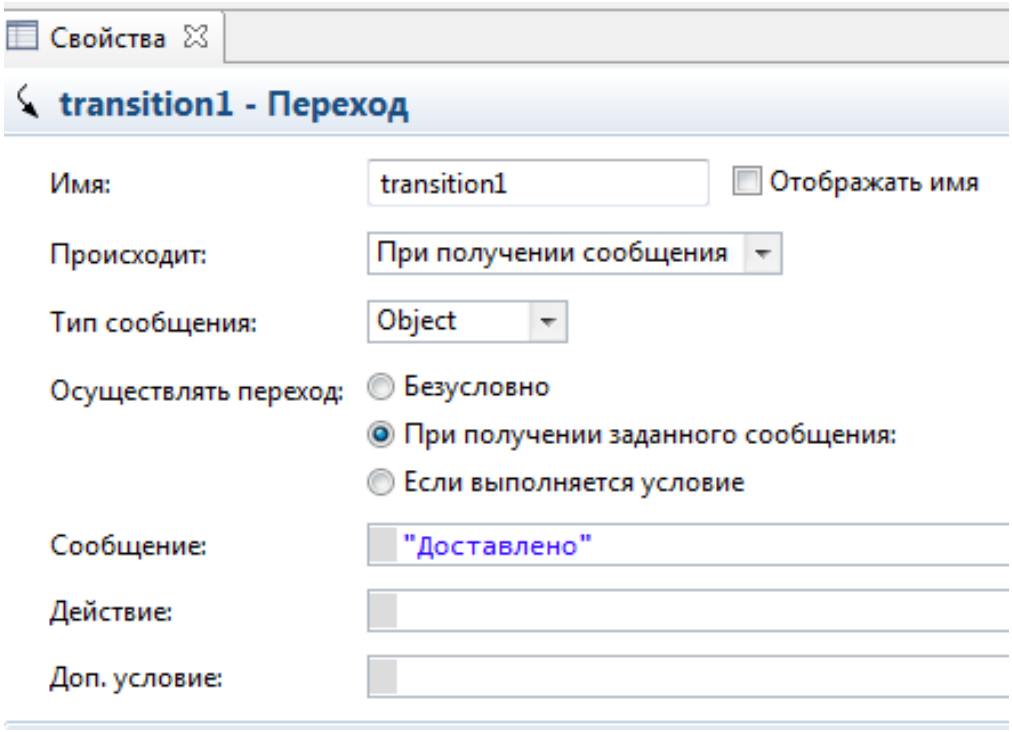


Рис. 4.6. Пример формирования сообщения для перехода

Чтобы сформировать заказ отделу снабжения, войдем в состояние «Ожидание» и в поле «действие при входе» напишем код:

send(«Заказ», main.снабжение);

В этом случае, как только агент «Реализация» перейдет в состояние «Ожидание», то сразу будет послано сообщение «Заказ».

Теперь необходимо сформировать реакцию агента «Снабжение» на получение данного сообщения. Как только он получит сообщение, он должен в течение заданного времени доставить заказ. Сделаем это с помощью блоков source, delay и sink соответственно. Сформировав соответствующую цепочку, укажем в блоке source в свойстве «Прибывают согласно» результат «вызовам функции inject».

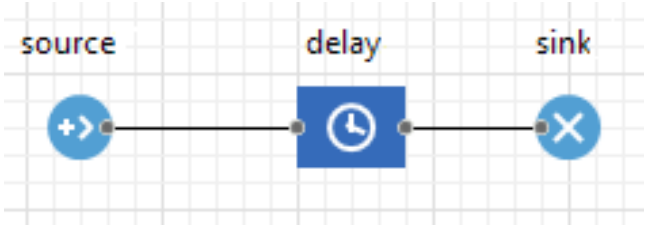


Рис. 4.7. Цепочка действий у агента «снабжение»

В объекте delay зададим время uniform (10,15), а в действиях выберем строку «при выходе» и напишем код:

send(«Доставлено», main.реализация);

Обратим внимание, что в обоих операторах send соответствующие агенты написаны строчными буквами, поскольку данные агенты названы именно так. Также отметим, что такой вызов возможен лишь при общении между единичными агентами. Если создан не один агент, а популяция агентов, то необходимо уточнять, кому именно будет передаваться сообщение. В данном случае сообщение будет передано отдельному агенту «реализация», благодаря чему его диаграмма состояний перейдет из состояния «Ожидание» в состояние «Работа».

Простейший случай отображение данного взаимодействия – с помощью параметра. У агента main сформируем параметр типа string, который будет описывать, что происходит в данное время. Перейдем в агент «реализация» и, выбрав состояние «Ожидание», допишем действия следующим образом (*рис. 4.8*).

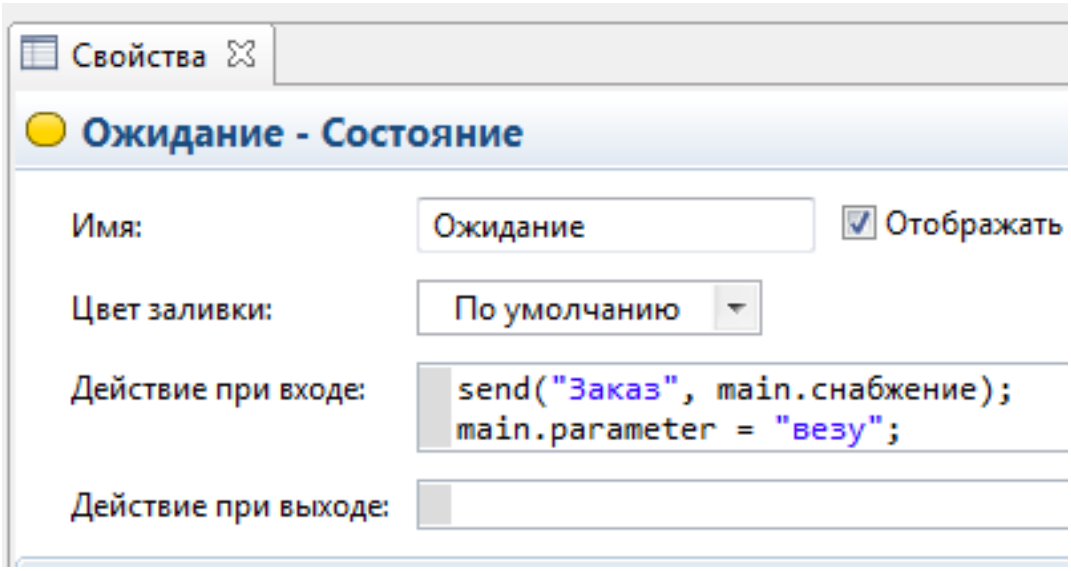


Рис. 4.8. Формирование заказа с отображением в параметре parameter.

У агента «реализация» в код добавится следующий оператор (*рис. 4.9*).

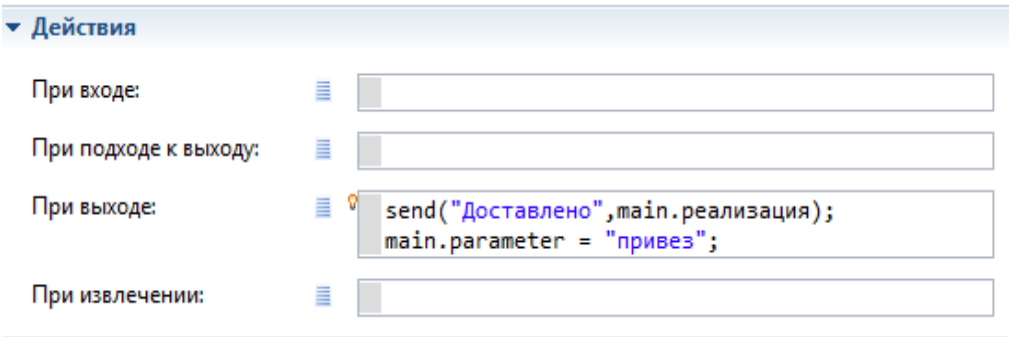


Рис. 4.9. Объект delay агента «снабжение»

4.3. Области видимости

Поскольку агенты имеет некоторую принадлежность к определенному местоположению, для обращения к ним необходимо соблюдать правила, продиктованные областью видимости тех или иных элементов. Рассмотрим пример области видимости, который приведен в справочной системе Anylogic.

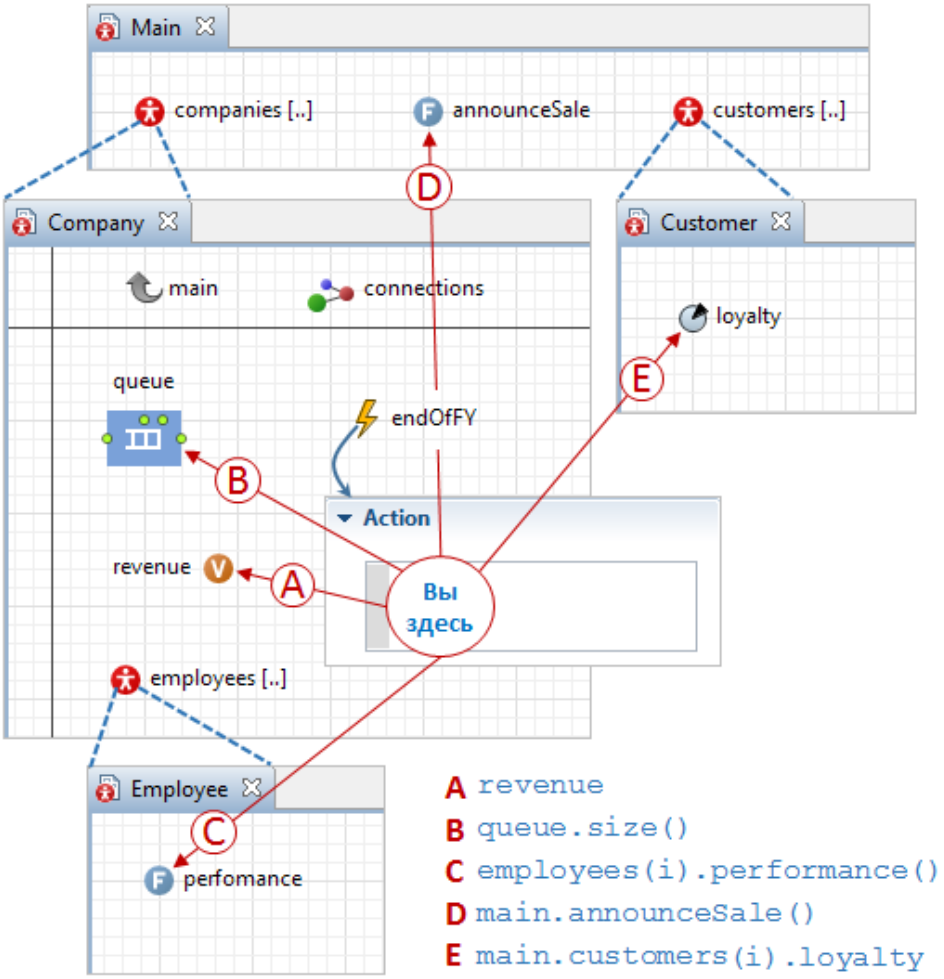


Рис. 4.10. Области видимости

Поясним данный рисунок. Если необходимо обратиться к некоторому объекту, находясь в агенте Company, который был создан из агента main, то функция revenue, располагающаяся в этом же агенте, вызывается напрямую, без дополнительных указаний (A). Аналогичным образом было определено обращение к размеру очереди queue, которая принадлежит агенту Company. Если необходимо обратиться к функции агента, который был непосредственно создан в данном агенте (на рис. 4.10 это агент Employee), то обращение производится через указание конкретного агента. Поскольку была создана популяция агентов (что обозначено символами […]), то требуется указать конкретного агента, а затем – функцию: employees(i).perfomance(). Если же требуется обратиться к функции, которая принадлежит главному агенту main, то необходимо сначала обратиться к агенту main, а затем указать эту функцию:

main.announceSale(). Аналогичным образом осуществляется обращение в случае, если требуется обратиться к другому агенту, располагающемуся на аналогичном уровне относительно агента main. В этом случае сначала указывается имя главного агента (main), затем – конкретный агент из созданной популяции агентов, а далее - его параметр: main.customers(i).loyalty.

4.4. Примеры

Пример 1. Модернизируем реализацию предыдущего примера следующим образом. Пусть в модели требуется визуально отобразить перемещение фуры, которая следует за заказом и обратно в отдел реализации.

Сформируем у агента main два прямоугольных узла node1 и node2:

- первый node будет описывать место, из которого необходимо забрать заказ;

- второй узел node1 указывает место, в которое надо будет доставить заказ.

Зададим агенту «снабжение» местоположение node, а агенту «реализация» - node1.

Создадим еще один агент «фура», которая будет перевозить заказ. Выберем для данного объекта изображение фуры. Начальное положение фуры, как и у производства, будет node. После этого войдем в агент «Снабжение» и добавим к существующей цепочке два блока moveto. Этот объект перемещает заданного агента в заданную точку. У блока source в качестве агента выберем агент «фура».

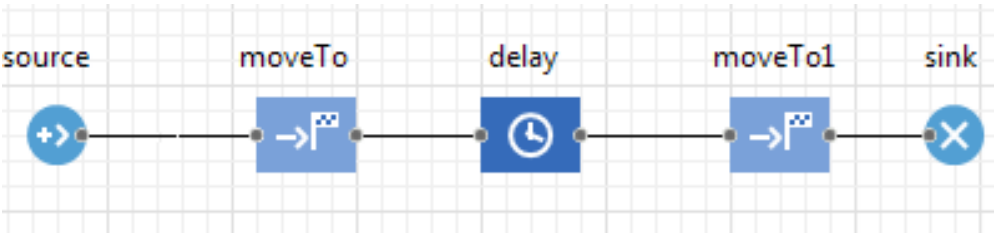


Рис. 4.11. Дополнение модели

У первого объекта moveto в качестве места назначения выберем «агент/ресурс» и укажем «main.снабжение». У второго объекта moveto аналогичным образом укажем «main.реализация». Запустив модель, увидим, что при возникновении заказа фура едет в узел node1, а затем возвращается в node. Однако, при этом такая же фура остается в узле node. Это произошло из-за того, что мы создали новую заявку, а не вставили агента в существующую цепочку. Очевидно, что в данном случае правильнее будет вставить агента «грузовик» в цепочку снабжения. Для этого вместо блоков source и sink воспользуемся блоками enter и exit соответственно:

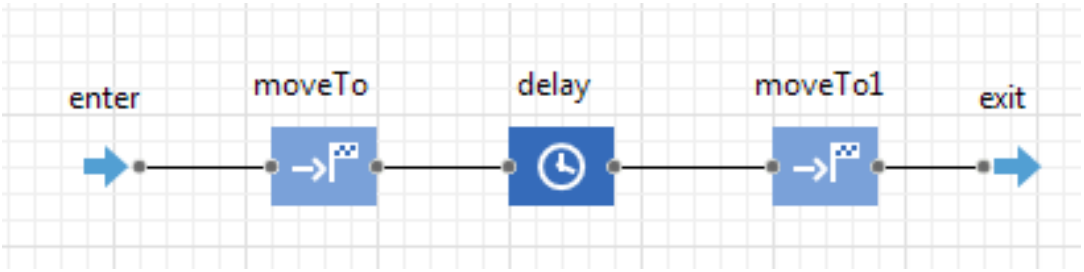


Рис. 4.12. Измененная цепочка для агента «снабжение»

Зайдем в connections для агента «снабжение» и в качестве действии при получении сообщения изменим существующий код на следующий:

enter.take(main.фура);

Запустив модель, увидим, что теперь автомобиль при перевозке покидает один узел и попадает в другой, а по истечении некоторого времени возвращается назад.

Рассмотрим специфику взаимодействия агентов в более сложном случае. Пусть имеется несколько агентов типа «реализация», у которых периодически возникает потребность в ресурсах и которые должна перевозить фура. Для этого щелкнем по агенту «реализация» на main и в открывшихся свойствах вместо «одиночный агент» выберем «популяция агентов». Пусть, без ограничения общности, их будет три (*рис. 4.13*).

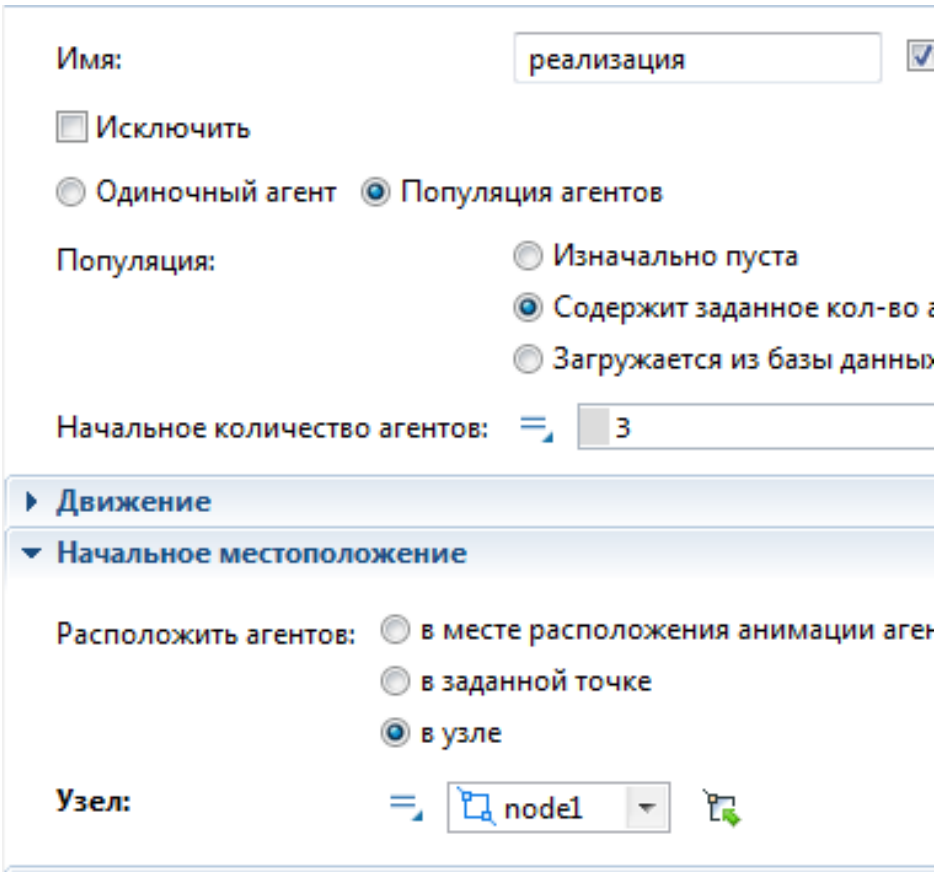


Рис. 4.13. Видоизмененная реализация

Создадим более сложный тип сообщения, который должен содержать информацию о том, у какого именно агента возникла в данный момент потребность в комплектующих. Эту информацию целесообразно хранить в параметре. В связи с этим создадим новый тип агента «Заказ», который содержит параметр «Заказчик». Тип у параметра «Заказчик» будет «Реализация», поскольку он должен хранить информацию о том, какой отдел реализации заказал комплектующие.

Перейдем в агент «реализация». Теперь диаграмма состояний, представленная на рис. 4.5, будет описывать поведение каждого из трех агентов. Перейдем в свойства состояния «Ожидание» (*рис. 4.14*). Теперь будем пересылать не просто сообщение типа string, а заказ, который должен содержать заказчика. Для этого сформируем новый экземпляр заказа следующим образом:

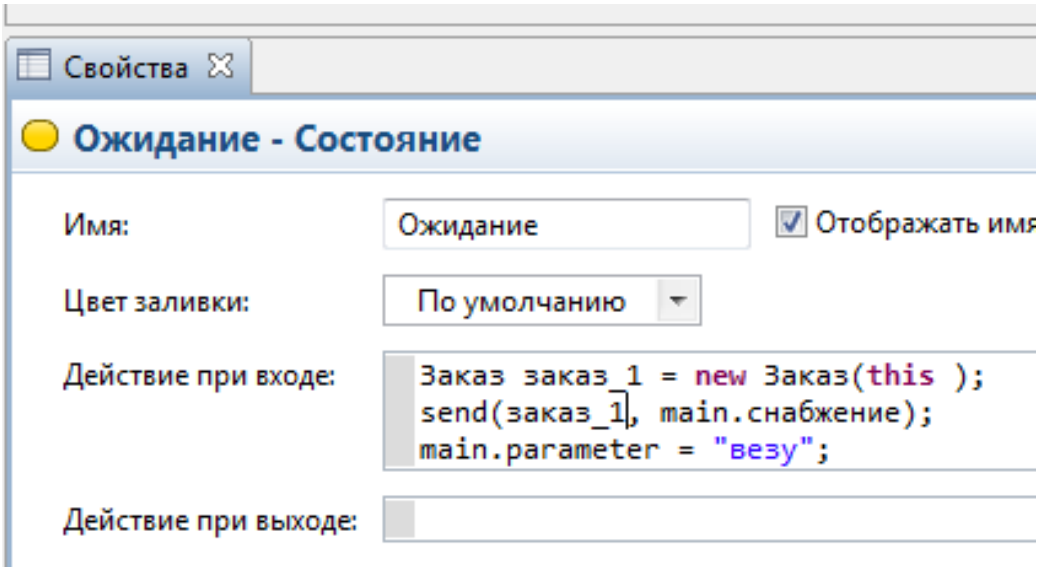


Рис. 4.14. Формирование заказа

Рассмотрим более подробно новый код. Первый оператор формирует новый заказ (он назван «заказ_1») типа Заказ. В качестве параметра передается параметр this, который является ссылкой на того агента, который в данный момент и осуществляет формирование нового заказа на комплектующие. Далее в отдел снабжения (main.снабжение) передается сообщение заказ_1.

Перейдем в отдел «Снабжение». На вход системы теперь поступает агент типа «Заказ». В связи с этим, у блока Enter в качестве типа агента необходимо заменить на «Заказ». После того, как фура дойдет до отдела снабжения, требуется ее возвратить в исходную точку. Поэтому в качестве точки возврата выбираем того агента, который являлся заказчиком. Поскольку у агента типа «Заказ», который поступил на вход объекта Enter, есть свойство «Заказчик», в которое был записан заказчик, необходимо обратиться к этому полю. Это можно сделать следующим образом (*рис. 4.15*).

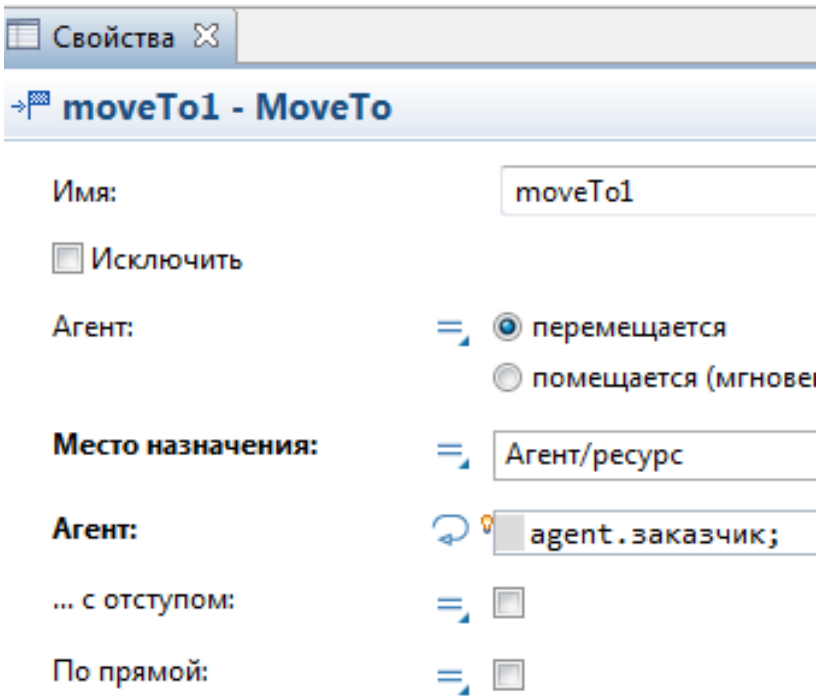


Рис. 4.15. Возвращение к заказчику

Необходимо также изменить тип входящего сообщения в разделе «Connections». Это будет являться типом «Заказ».

enter.take(msg);

Таким образом, получили модель, позволяющую организовать взаимодействие отдела снабжения с несколькими заказчиками.

Пример 2. Пусть имеется два потока заявок. Заявка каждого потока идентифицируется своим цветом, который может принимать одно из следующих значений:

- красный;
- синий;
- желтый;
- зеленый;
- оранжевый.

Заявка с первого потока должна дождаться заявки со второго потока с аналогичным цветом: в этом случае обе заявки проходят дальше.

Рассмотрим особенности данной задачи. Во-первых, в данном примере предлагается нестандартная характеристика заявок: цвет, который может содаржать 5 значений. Для формирования данной характеристики воспользуемся так называемым списком вариантов. По сути это перечислимый тип, позволяющий определить различные варианты. Он создается командой меню Файл/создать/список вариантов. В открывшемся окне необходимо определить наименование списка и перечислить варианты. Назовем данный тип Colors и перечислим заданные цвета.

Далее создадим популяцию агентов (назовем ее MyAgent), отличающихся параметром par, который имеет тип Colors. Данный параметр можно создать в процессе создания агентов, а можно позже добавить в окно агента (*рис. 4.16*).

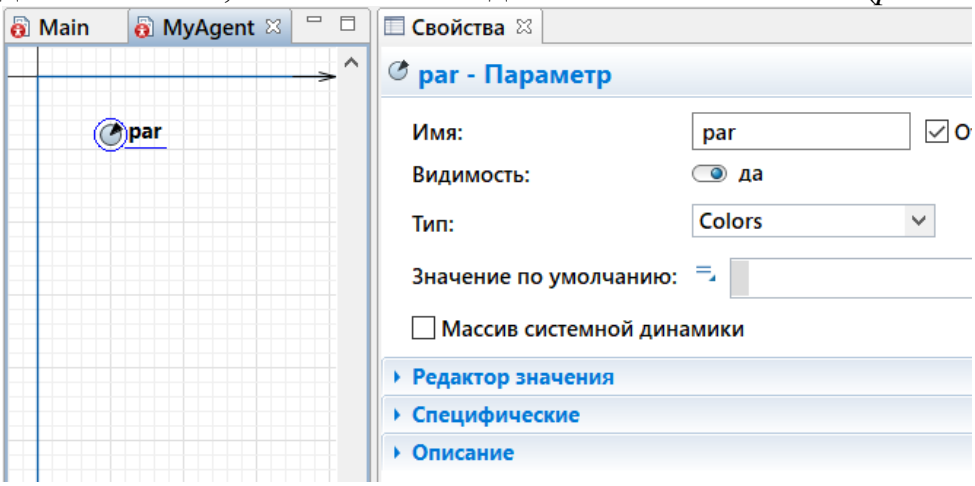


Рис. 4.16. Задание параметра для агента

Схема модели будет достаточно проста (*рис. 4.17*).

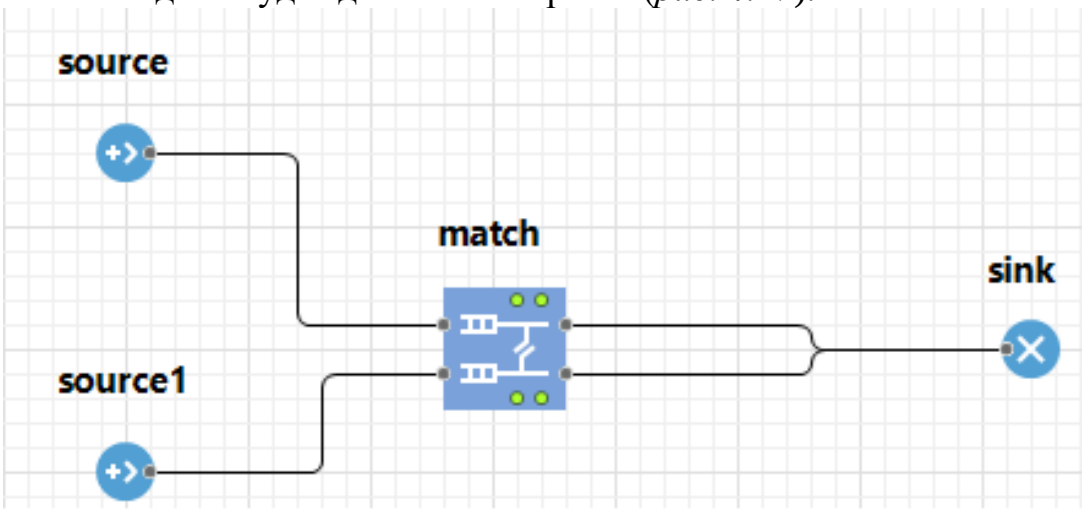


Рис. 4.17. Схема модели

Смоделируем возможность задания цвета для каждой заявки случайным образом. Для этого, в первую очередь, в каждом блоке Source отметим, что создаваемый агент будет являться агентом MyAgent, а, во-вторых, - обратимся к выбору одного из случайных вариантов типа Colors. Это реализуется следующим образом (*рис. 4.18*).

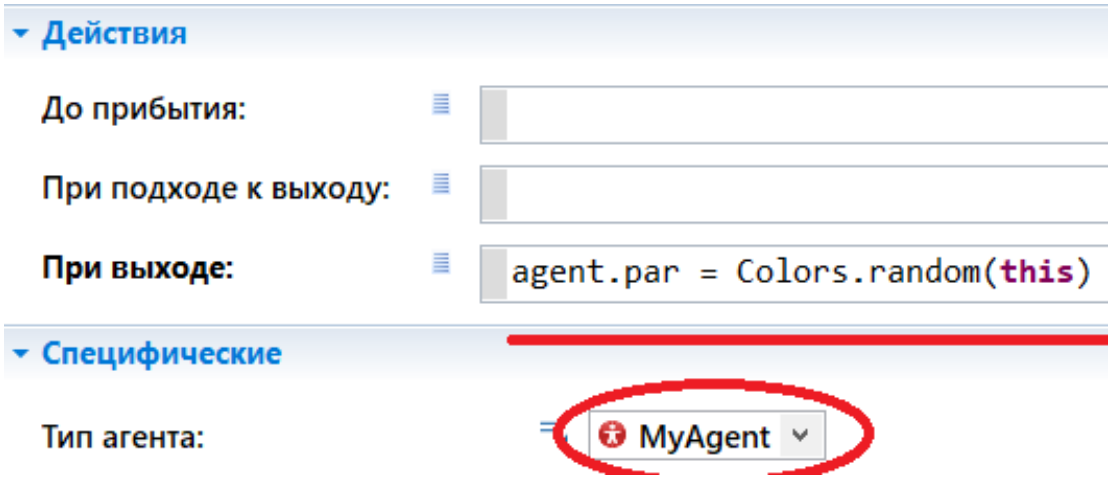


Рис. 4.18. Настройка свойств блоков Source

Далее, настроим свойство объекта Match следующим образом (*рис. 4.19*).

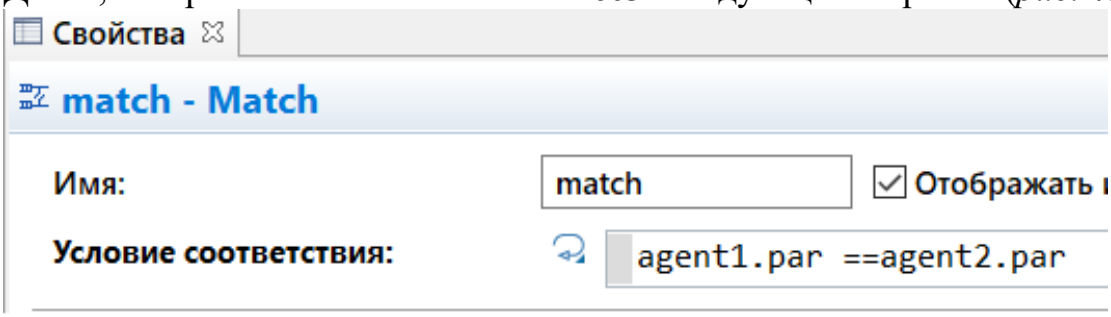


Рис. 4.19. Настройка свойств Match

Из данного рисунка видно, что к агенту первого потока обращаемся через agent1, а к агенту второго потока – через agent2. У каждого агента есть параметр par, который можно увидеть, если после точки (после написания agent1 или agent2) нажать комбинацию клавиш Ctrl+пробел.

5. ЭКСПЕРИМЕНТЫ В AnyLogic

Среда AnyLogic позволяет выполнять разнообразные эксперименты: от элементарных до сложных оптимизационных. В частности, свободно распространяемая версия данного продукта позволяет использовать в модели следующие эксперименты:

- простой эксперимент (Запускает модель с заданными значениями параметров, поддерживает режимы виртуального и реального времени, анимацию, отладку модели.);
- варьирование параметров;
- оптимизация.

5.1. Варьирование параметров

Производит повторный запуск модели с разными значениями параметров корневого объекта. Этот эксперимент позволяет сравнить поведение модели

при разных значениях параметров и оценить степень влияния отдельных параметров на поведение модели. Запуская несколько прогонов модели с фиксированными значениями параметров, можно оценить влияние случайных факторов в стохастических моделях.

Для создания такого эксперимента необходимо:

- с помощью меню Файл/Создать/ Эксперимент выбрать «Варьирование параметров»;

- выбрать агента, для которого осуществляется эксперимент («Агент верхнего уровня») – по умолчанию main;

- задать параметры, которые будут варьироваться. Чтобы разрешить варьирование параметра, необходимо в соответствующей строке таблица выбрать тип «Диапазон». После этого необходимо задать нижнюю и верхнюю границу изменения параметров и шаг.

Для того, чтобы результаты эксперимента были отображены, необходимо нажать на кнопку «Создать интерфейс».

Пример. Рассмотрим пример приобретения продукта, модель которого представлена на рис. ?. Будем исследовать изменения динамики приобретения продукта в зависимости от частоты контактов. Сделаем данный тип диапазоном, изменяющимся от 100 до 150 (*рис. 5.1*).

После этого нажмем на кнопку «Создать интерфейс».

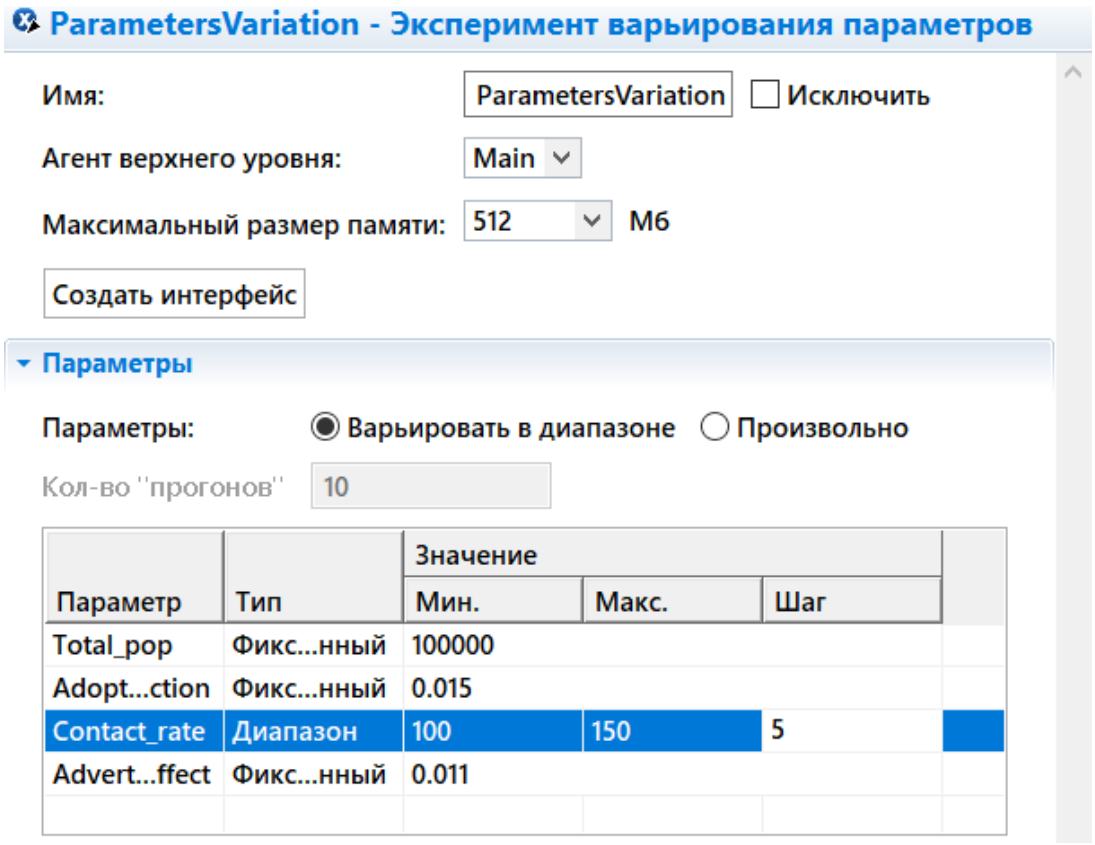


Рис. 5.1. Настройка свойств эксперимента

В результате появится стандартное окно интерфейса, представленное на *рис. 5.2*.

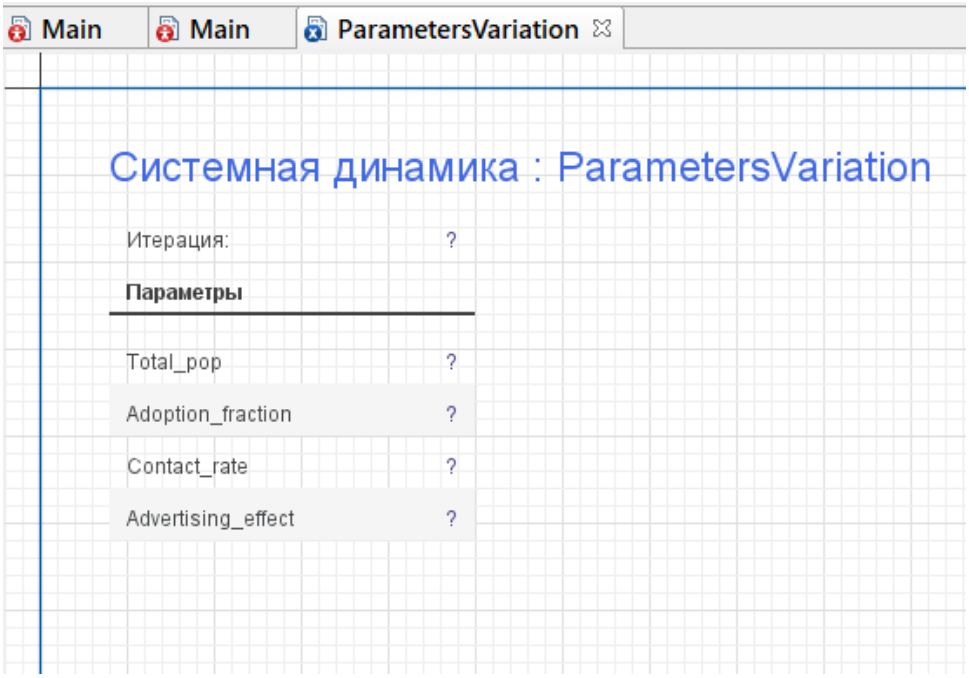


Рис. 5.2. Стандартный интерфейс для варьирования параметров

Добавим временной график для отображения результатов эксперимента. Для этого создадим набор данных для накопителя, динамику которого будем изучать (в данном случае, для накопителя «Покупатели» - Adopters, *рис. 5.3*).

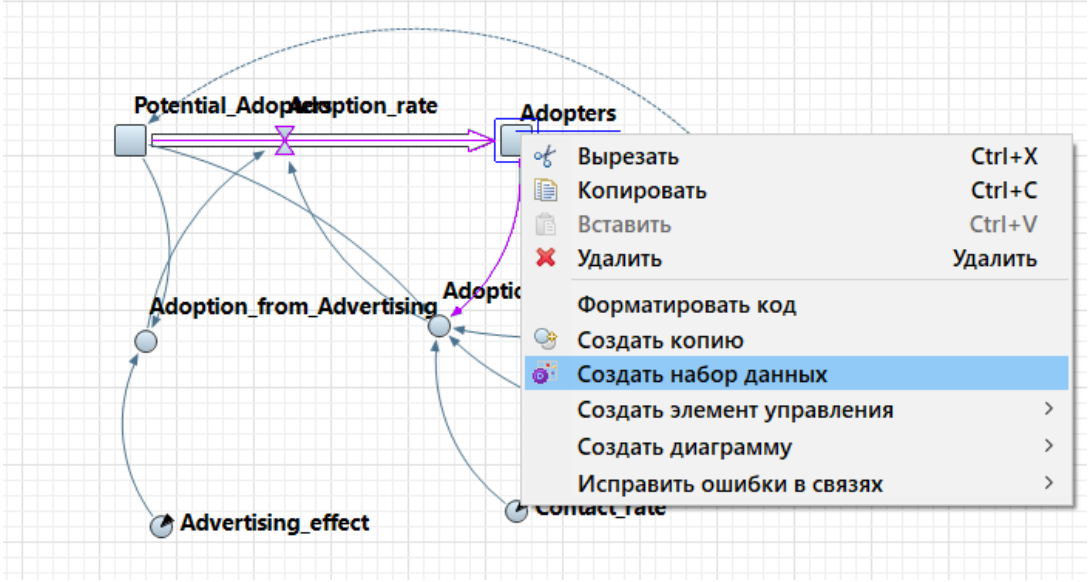


Рис. 5.3. Создание набора данных

В свойствах данного набора отобразим необходимость автоматического обновления данных (*рис. 5.4*). Добавим в окно интерфейса эксперимента временной график (*рис. 5.4*).

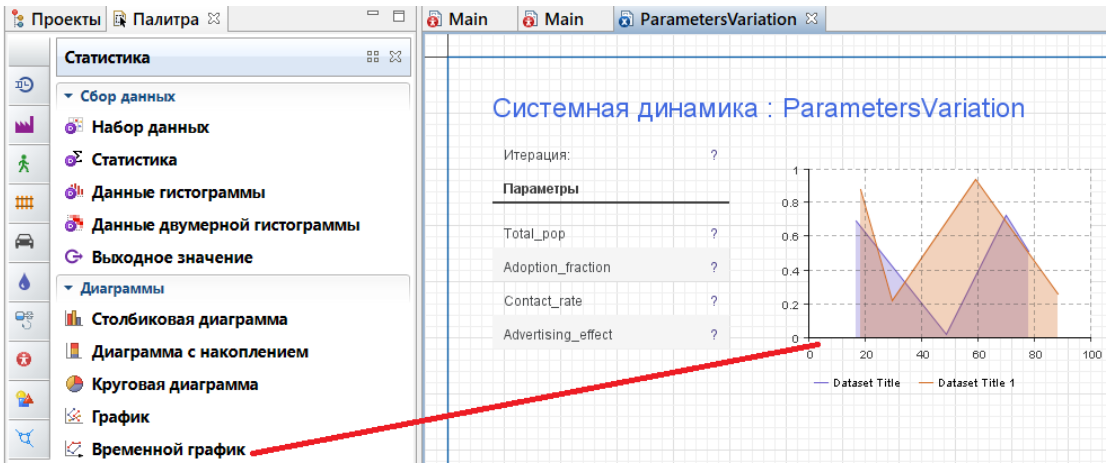


Рис. 5.4. Временной график

Для того, чтобы отображалась серия графиков, в свойствах эксперимента в Действиях Java выберем свойство «Действие после «прогона» модели» и напишем следующий код (*рис. 5.5*).

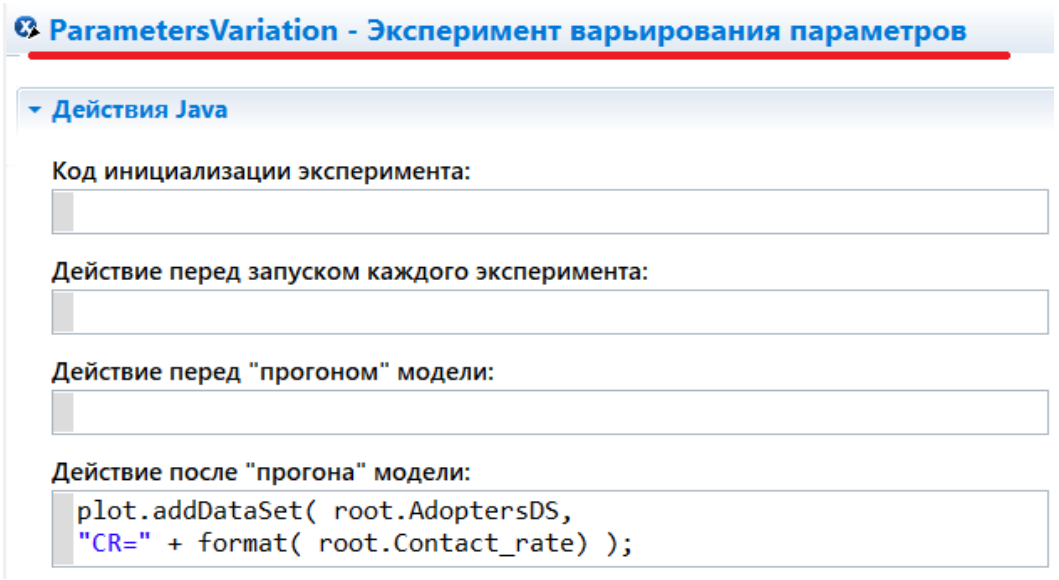


Рис. 5.5. Код для создания графика

Запустим эксперимент. Для этого выберем в раскрывающемся списке «Системная динамика/ParametersVariarion» (*рис. 5.6*).

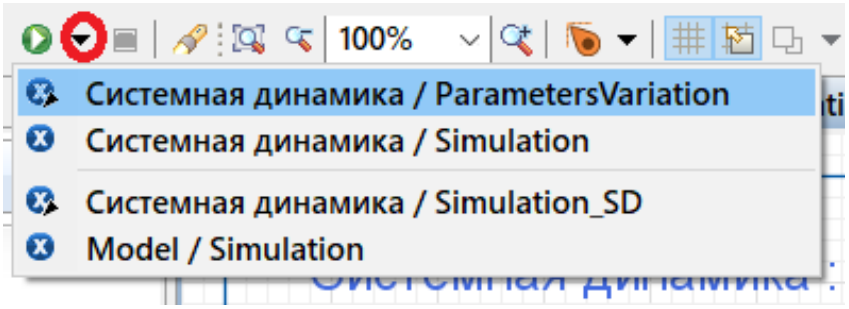


Рис. 5.6. Вызов эксперимента

Результаты эксперимента представлены на *рис. 5.7*.

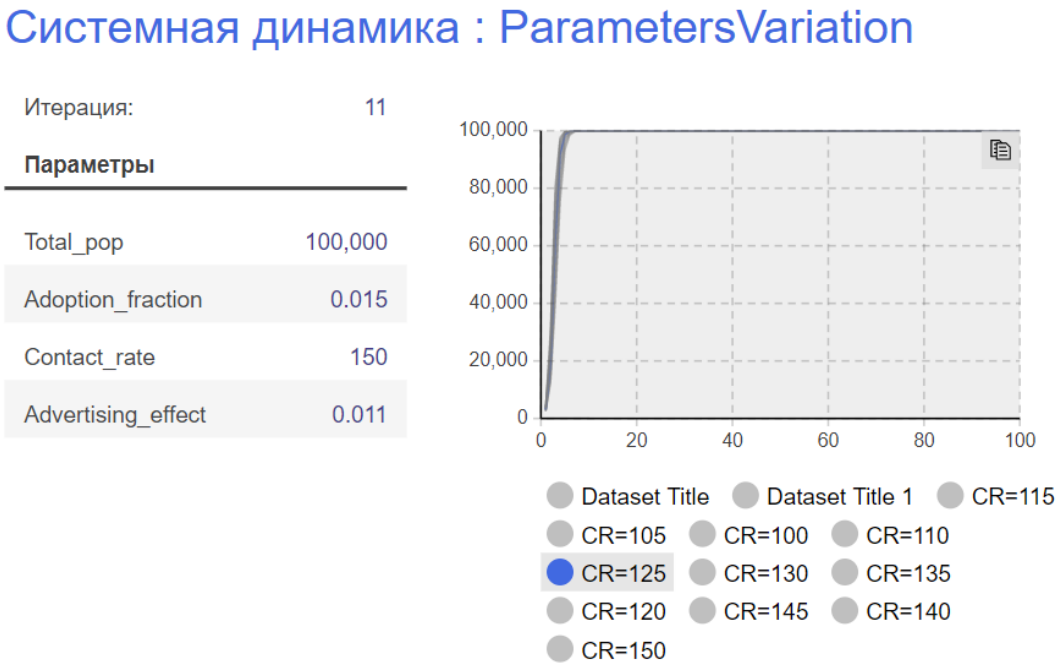


Рис. 5.7. Результаты эксперимента

На данном рисунке представлено множество графиков, каждый из которых представляет динамику изменения состояния «Покупатели» при заданном значении параметра «Число контактов». Как видно из результатов эксперимента, тенденция изменения данного состояния в зависимости от числа контактов незначительна.

5.2. Оптимизационный эксперимент

Оптимизационный эксперимент в среде AnyLogic заключается в многократном прогоне модели (аналогично случаю с варьированием параметров) для того, чтобы определить оптимальное значение некоторой целевой функции при заданных ограничениях. В общем случае, для того, чтобы реализовать оптимизационный эксперимент, необходимо:

- с помощью меню сформировать окно-заготовку для оптимизационного эксперимента;

- задать целевую функцию, зависящую от параметров;

- задать параметры, которые будут изменяться, диапазон и шаг изменения;

- задать ограничения задачи;

- запустить эксперимент и проанализировать результаты.

Рассмотрим данные этапы более подробно. Запуск оптимизационного эксперимента отличается от варьирования параметров лишь типом эксперимента – «Оптимизация». Поскольку цель эксперимента заключается в нахождении оптимальных значений параметров (одного или нескольких), необходимо задать цель – функцию, которая будет зависеть от этих параметров. Поскольку данные параметры будут, скорее всего, находиться в основном окне, к ним необходимо обращаться следующим образом:

<root>.<параметр>

Например:

root.par1-root.par2+5*root.par3

Эту функцию необходимо занести в поле «Целевая функция», представленное в верхней части свойств эксперимента (рис. 5.8). Далее следует определить, к какому значению (максимальному или минимальному) данная функция должна стремиться.

Ниже будут отмечены все параметры, которые есть в основной модели, на основании которой строится эксперимент. По умолчанию у них тип - «фиксированный». Поскольку эксперимент предполагает прогоны модели с разными значениями параметров, то для тех параметров, которые должны изменяться, необходимо внести поправки в тип (*рис. 5.8*).

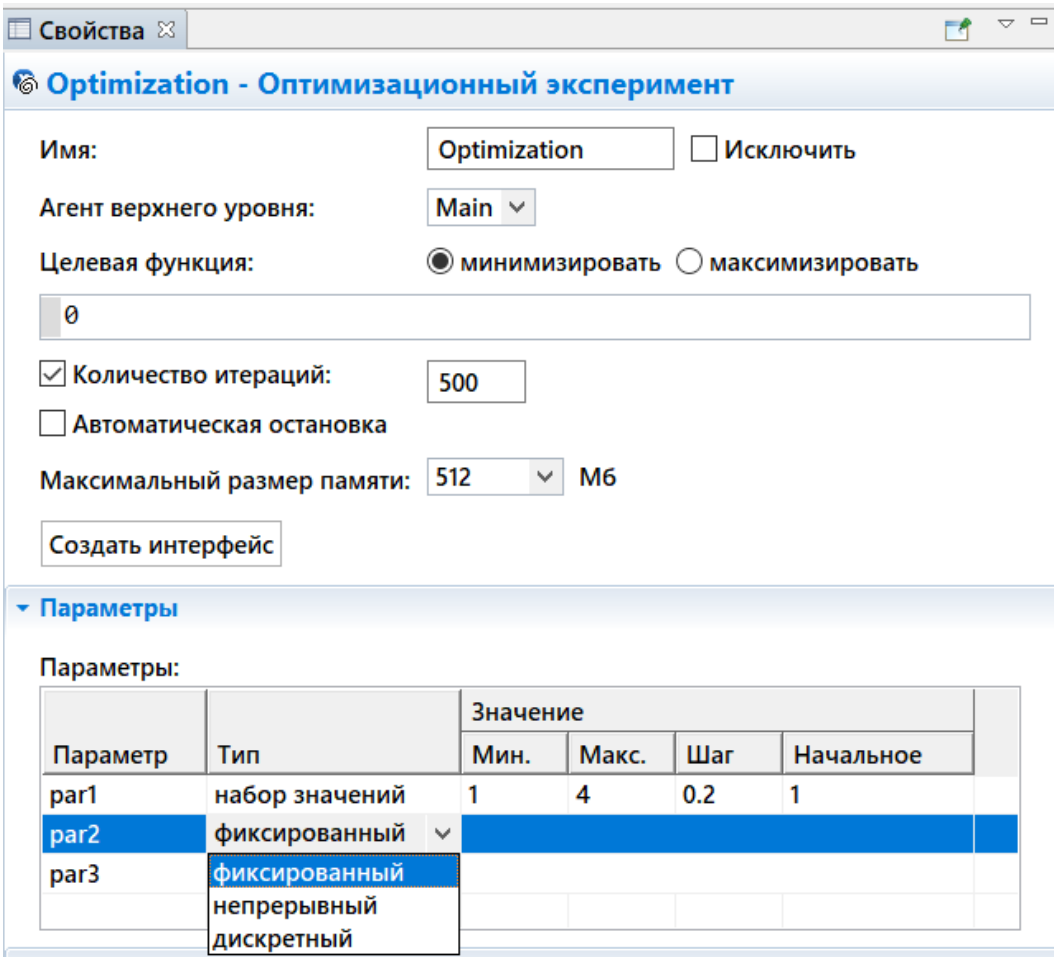


Рис. 5.8. Настройка параметров оптимизационного эксперимента

В частности, на рис. 5.8 представлен тип параметра par1, заданный набором значений от 1 до 4 с шагом 0.2. Для изменения типа необходимо щелкнуть мышью по типу напротив нужного параметра (в частности, приведено изменение типа для параметра par2) и в открывшемся списке выбрать нужный тип. Оптимизационные задачи решаются, как правило, при каких-либо ограничениях. В AnyLogic существует две категории ограничений: собственно ограничения и требования. Разница между этими думя категориями заключается в следующем. Ограничения, накладываемые в оптимизационном эксперименте, проверяются непосредственно перед прогоном модели, а требования – после. Рассмотрим разницу между этими двумя видами ограничений.

Пусть, например, требуется, чтобы вероятность отказа не превышала 0.1. Введем переменную ver_otk. Если ограничение на данную вероятность будет записано в разделе «Ограничения», то перед запуском модели будет проверено, чему равно данное значение. Поскольку вероятность рассчитывается, исходя из статистических данных, которые к началу прогона отсутствуют, то и вероятность посчитана быть не может. Следовательно, ее необходимо ввести именно в категории «Требования»:

root.ver_otk<=0.1

Пример 1. Объектом исследования является АТС, на вход которой поступает пуассоновский поток заявок с интенсивностью λ=1.5. Интенсивность обслуживания одним каналом в среднем равна μ=0.5 (время обслуживания распределено экспоненциально).

1. Построить модель в соответствии с описанием.
2. Определить оптимальное число каналов обслуживания, если известно, что:

- каждый разговор приносит прибыль, которая вычисляется по формуле:

Прибыль = Коэфф длительность разговора*;

- каждый необслуженный вызов приносит убытки (penalty), связанные с нарушением QoS;
- обслуживание каналов связи определяется некоторой функцией equiepment, зависящей от количества задействованных каналов.

Таким образом, целевая функция может быть описана следующим образом:

Ц.ф.= Прибыль – penalty – equipment (1)

Решение.

Спроектируем следующую модель системы (*рис. 5.9*).

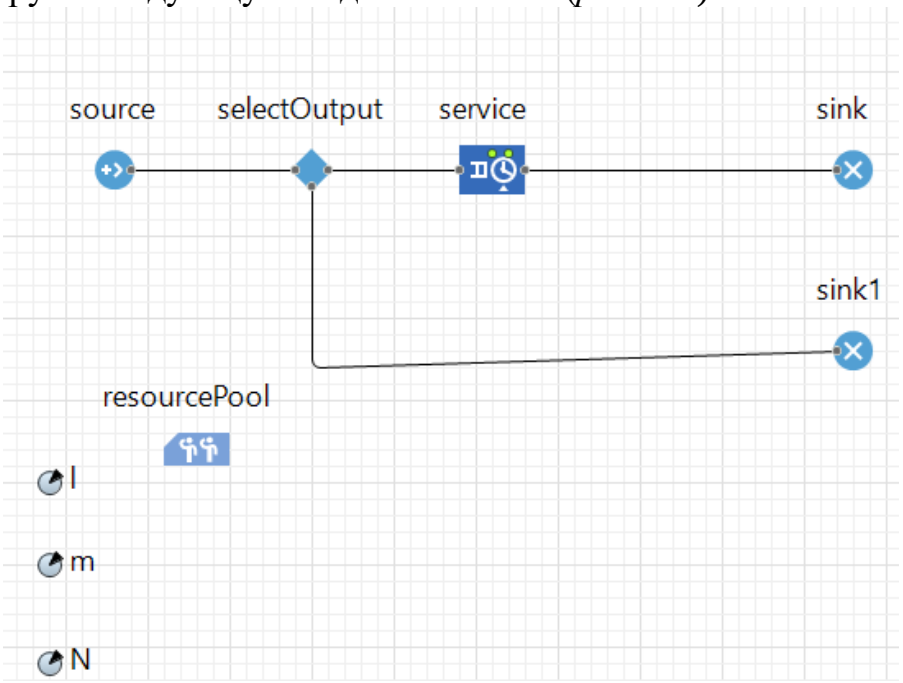


Рис. 5.9. Модель системы

В блоке SelectOutput выберем условие: количество занятых каналов меньше заданного текущего числа каналов:

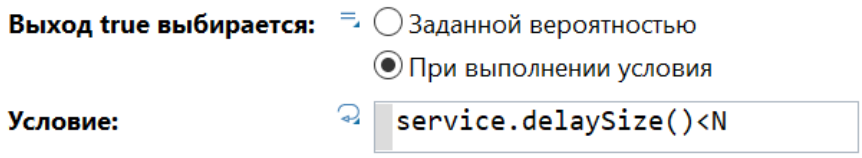


Рис. 5.10. Условие в блоке SelectOutput

В объекте ResourcePool зададим параметрически число ресурсов – N.

Для решения оптимизационной задачи введем динамическую переменную Itog, которая будет зависеть:

- от числа каналов N;
- от прибыли, которая будет описываться с помощью переменной income;
- от штрафа, который будет описываться переменной penal.

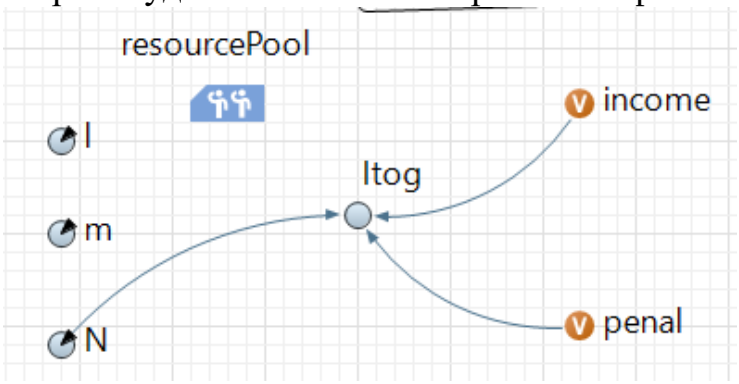


Рис 5.11. Определение итоговой прибыли

Определим переменные income и penal. Введем параметр penalty, который содержит количество единиц, которые определяют убытки за одно несостоявшееся соединение. В блоке Sink в действиях при выходе опишем следующий код:

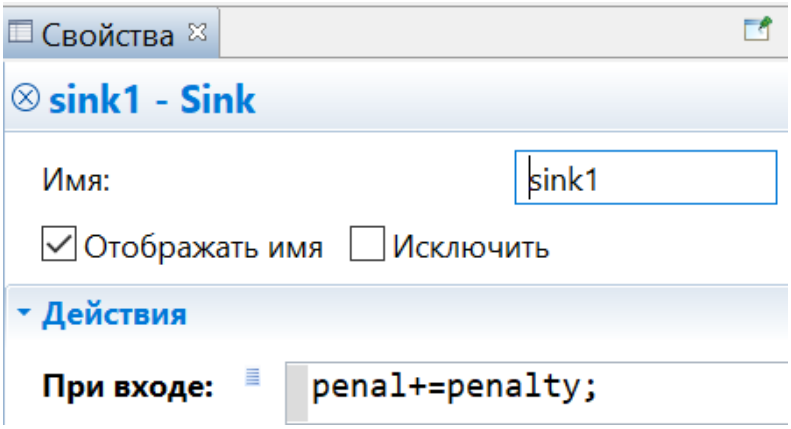


Рис 5.12. Настройка свойств объекта Sink

Введем параметр min_price, который будет содержать минимальную цену за соединение. Если соединение меньше одной минуты, то прибыль определяется данным коэффициентом. В противном случае он умножается на длительность соединения.

Для расчета данной прибыли введем функцию call_price, аргументом которой является время t. Тело функции будет следующим (*рис. 5.13*).

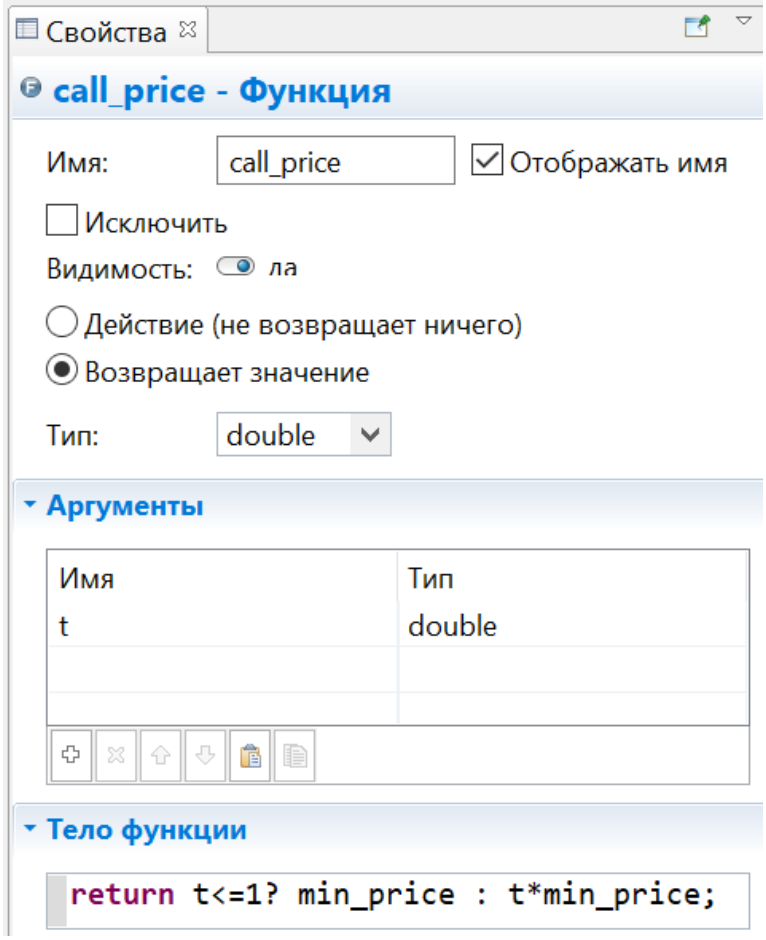


Рис 5.13. Функция call_price

Вызов данной функции будет осуществляться при успешном завершении соединения (в блоке Sink).

Определим длительность соединения. Для этого реализуем новый тип заявок, отличающихся от стандартных наличием двух параметров:

- ts – параметр, в который должно быть записано время начала соединения;
- tf – параметр, в который записывается время окончания соединения.

Значения этих параметров будут определяться в действиях при входе в блок Service и при выходе из него (*рис. 5.14*).

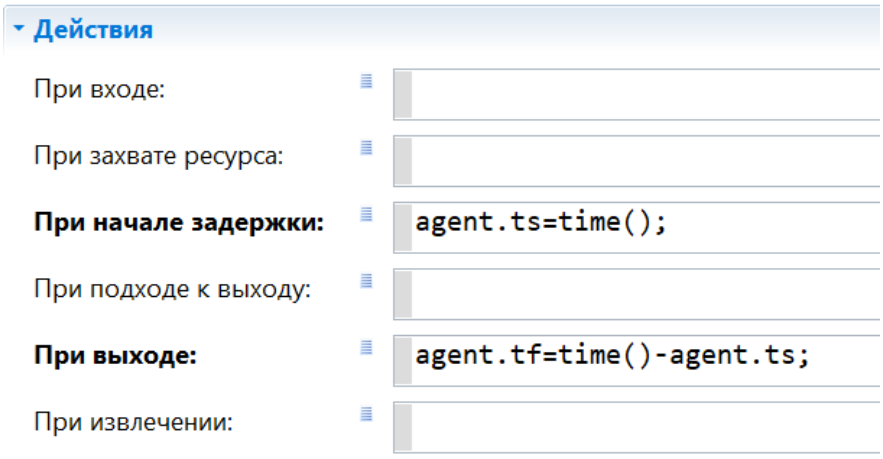


Рис. 5.14. Настройка свойств блока Service

Тогда при входе в блок Sink, в который попадают заявки после успешного соединения, необходимо описать следующие действия (*рис. 5.15*).

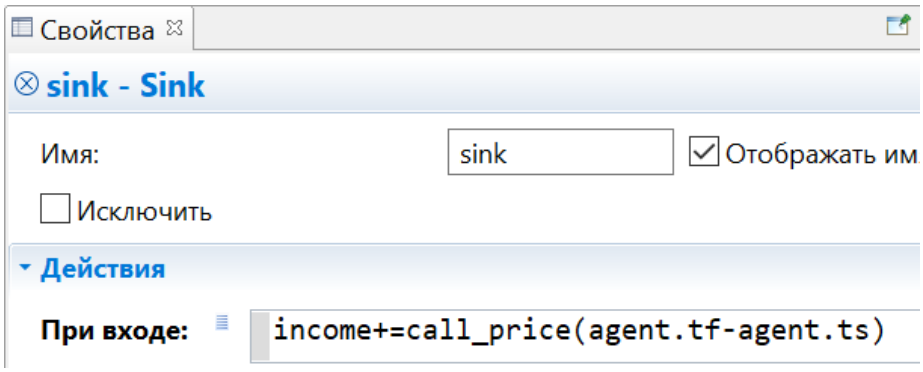


Рис 5.15. Настройка действий при входе в блок Sink

Как можно увидеть, переменная income увеличивается на значение функции call_price, аргументом которой является разница между завершением и началом соединения (т.е. фактически, длительность соединения).

В завершении, опишем зависимость, которая будет определять стоимость обслуживания заданного канала связи. Определим эту зависимость с помощью табличной функции equipment, которая может быть определена, например, следующим образом:

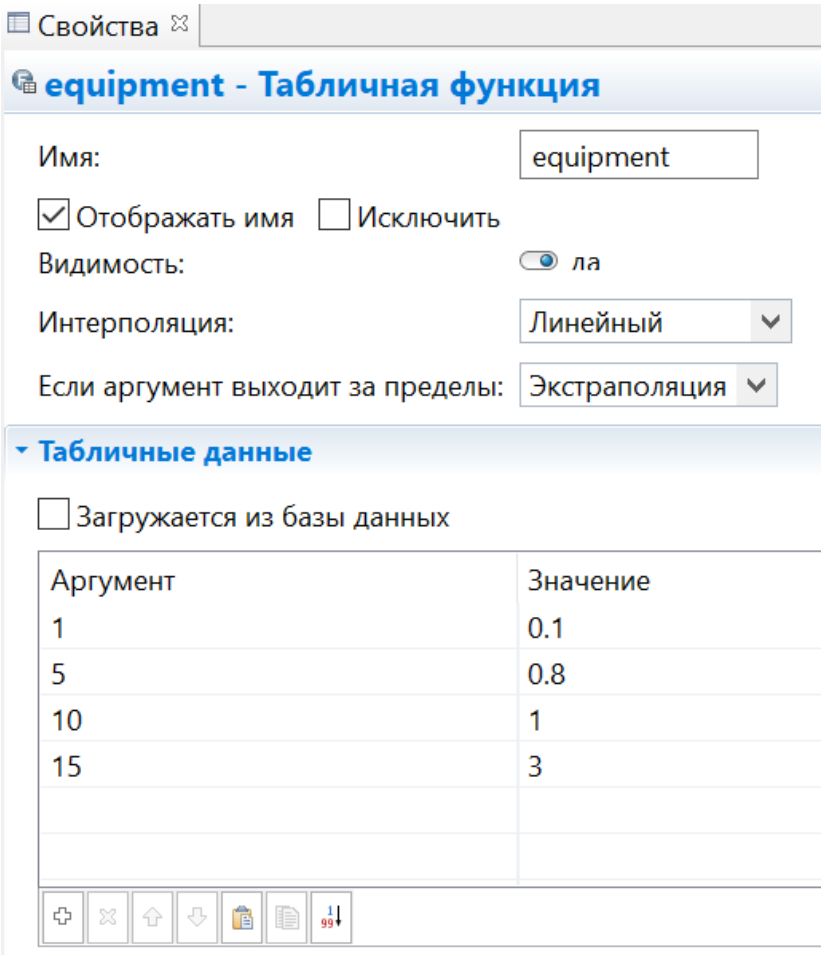


Рис 5.16. Определение функции equipment

В завершении, определим переменную Itog (*рис. 5.17*).

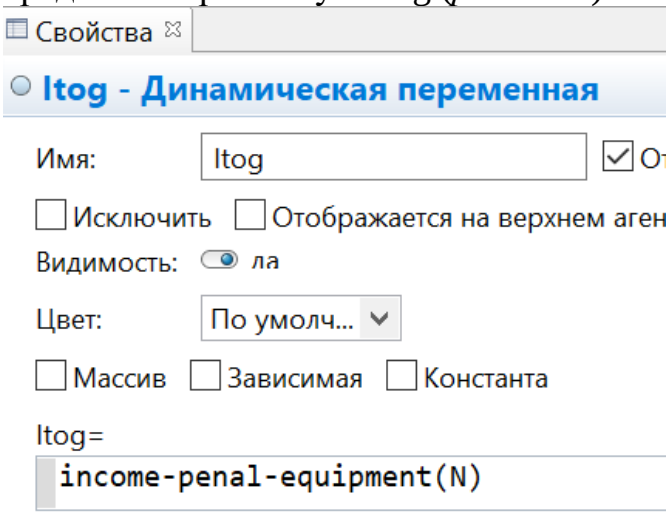


Рис.5.17. Определение переменной Itog

Все предварительные действия для формирования эксперимента завершены.

Создадим новый эксперимент. В качестве целевой функции выберем максимизацию переменной Itog, к которой обратимся через root.

В области «Параметры» будут отражены все параметры, участвующие в модели. Выберем параметр N и изменим его тип на Дискретный. Пусть значение данного параметра изменяется с 1 до 15 с шагом 1. После завершения описания эксперимента нажмем на кнопку «Создать интерфейс». Результат приведен на *рис. 5.18*.

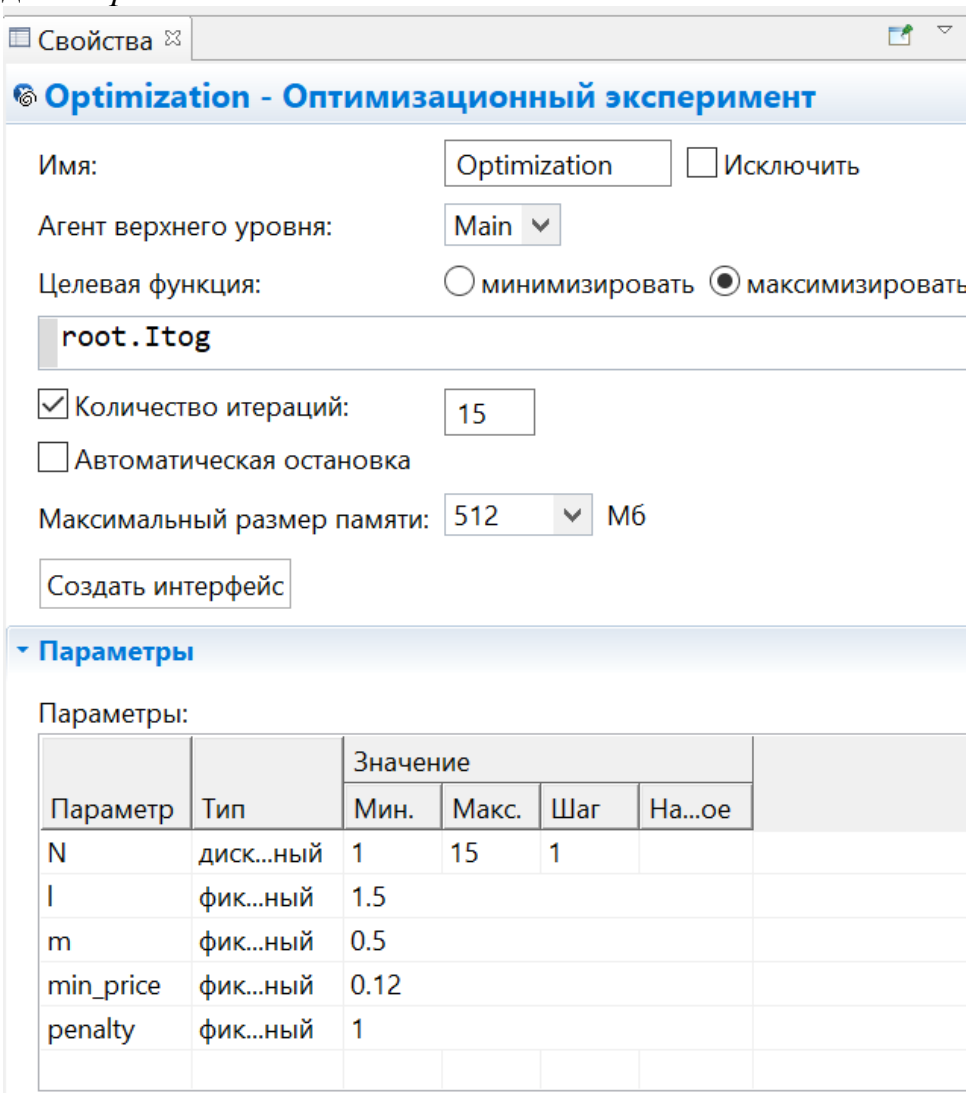


Рис. 5.18. Описание эксперимента

Запустим эксперимент. Получим следующий результат (*рис. 5.19*).

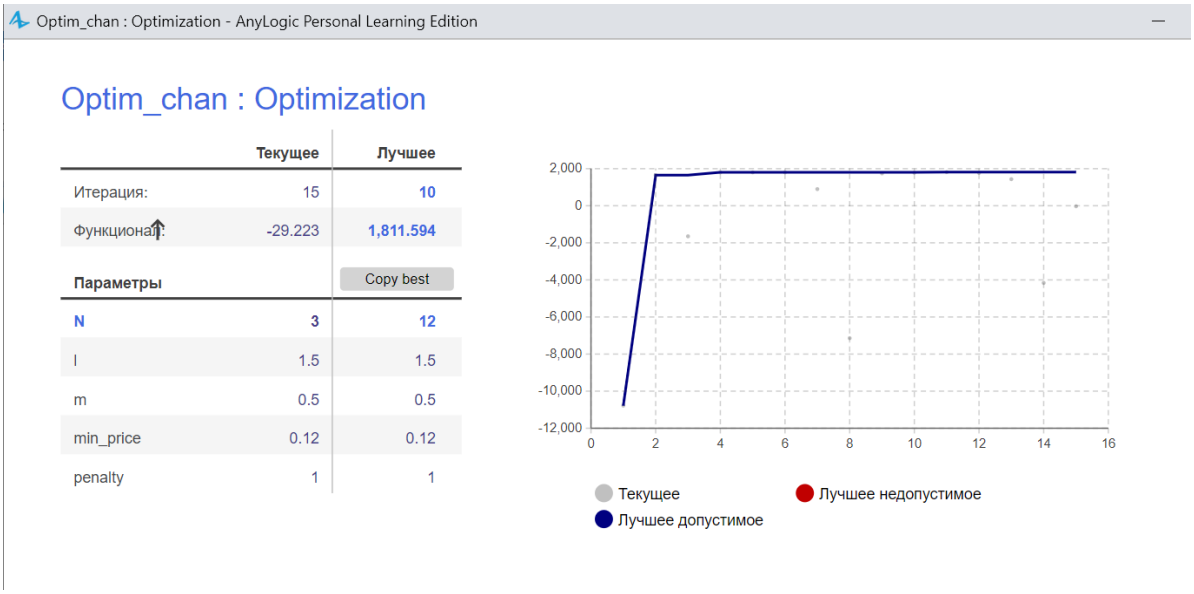


Рис. 5.19. Результаты эксперимента

Прокомментируем результаты эксперимента. Исходя из полученных данных, оптимальным значением будет являться 12 каналов обслуживания. Это, по всей видимости, связано с тем, что уже при 12 каналов получаем безотказную систему (т.е. количество переменной penalty перестает изменяться), а затраты на содержание оборудование возрастают.

Пример 2. Модифицируем предыдущую задачу следующим образом. Пусть, в условиях предыдущей задачи, требуется, чтобы мощности были задействованы не менее, чем на 30%.

Решение.

Добавим к сформированному эксперименту требование:

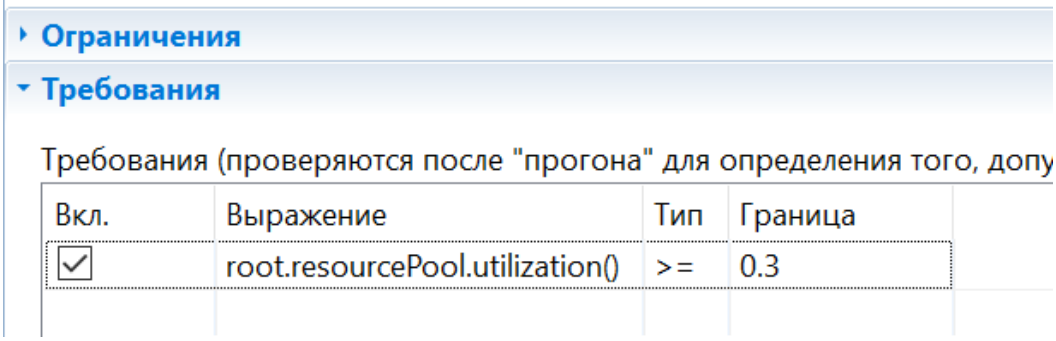


Рис. 5.20. Требование для эксперимента

Оно означает, что среднее число занятых каналов обслуживания (resourcePoll.utilization()) будет больше или равно 0.3. В результате получим уже следующие значения (*рис. 5.21*).

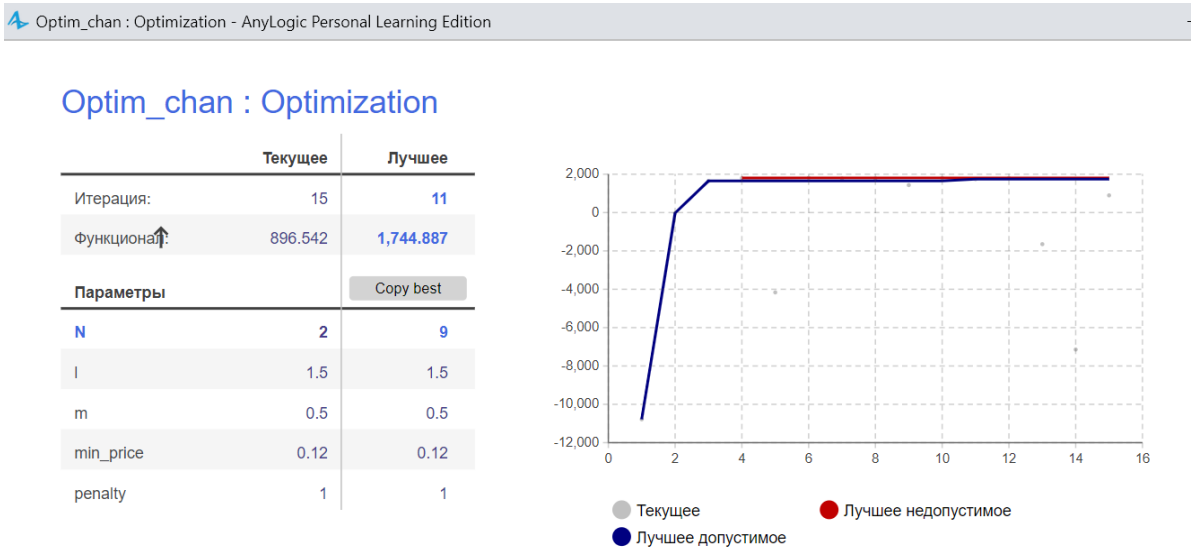


Рис. 5.21. Результаты моделирования

Анализируя результаты, видим, что в данном случае оптимальное значение уже 9, а не 12. Очевидно, что, начиная со следующего значения, будут нарушаться ограничения на ресурсы.

ЗАКЛЮЧЕНИЕ

В пособии изложен теоретический материал, необходимый для изучения дисциплины «Моделирование». Основное внимание уделено особенностям среды Anylogic, предназначенной для имитационного моделирования и получению статистики функционирования систем любой сложности.

В пособии рассматриваются основные подходы, на которых базируется данная среда: дискретно-событийное моделирование, системная динамика и агентное моделирование. Каждый раздел снабжен большим количеством примеров, достаточным для освоения функционирования тех или иных блоков.

Следует отметить, что возможности среды AnyLogic значительно шире, чем те, что рассмотрены в данном пособии. В частности, были опущены железнодорожная и другие библиотеки, рассмотрены не все объекты представленных библиотек, не описано взаимодействие среды с базами данных, использование аппарата численных методов и т.д. Все эти особенности среды выходят за рамки данного курса и, в связи с этим, могут быть рекомендованы желающим для самостоятельного изучения (см. [1]).

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Григорьев И. AnyLogic за три дня [Электронный ресурс] // AnyLogic.ru. Режим доступа: https://www.anylogic.ru/resources/books/free-simulation-book-and-modeling-tutorials/

2. Боев, Д. В. Компьютерное моделирование: Пособие для практических занятий, курсового и дипломного проектирования в AnyLogic / Д.В, Боев - СПб: ВАС, 2014 – 432 с.

3. Каталевский, Д.Ю. Основы имитационного моделирования и системного анализа в управлении: учебное пособие; 2-е изд., перераб. и доп. / Д.Ю. Каталевский. — М.: Издательский дом «Дело» РАНХиГС, 2015. — 496 с., ил.

4. Карпов, Ю.Г. Имитационное моделирование систем. Введение в моделирование с AnyLogic 5 / Ю.Г.Карпов. – СПб.: БХВ-Петербург, 57 2005.

5. Советов, Б.Я. Моделирование систем: Учеб. Для ВУЗов – 3-е изд., перераб. и доп. / Б.Я. Советов, С.А. Яковлев. – М.: Высш. шк., 2001. – 343 с.

6. AnyLogic [Электронный ресурс]. Режим доступа: http://www.anylogic.ru/

ПРИЛОЖЕНИЕ

Функции Anylogic для задания различных законов распределения

Как правило, длительность обслуживания представляет собой случайную величину. В этом случае для ее задания необходимо использовать одну из наиболее подходящих вероятностных функций. Наиболее часто используются следующие распределения из теории вероятностей.

1. **Равномерное распределение**. Это означает, что время обслуживания может быть с одинаковой вероятностью любой величиной из некоторого временного интервала. График случайной величины, распределенной по равномерному закону в интервале [a,b], имеет вид:

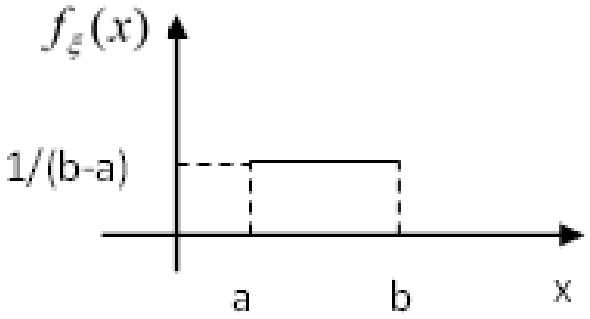


Рис.1. Случайная величина, распределенная по равномерному закону

Например, если сказано, что длительность выполнения некоторой работы составляет 25±3 с, то это означает, что соответствующая случайная величина имеет равномерный закон распределения в интервале [22,28].

Для моделирования равномерного закона распределения используется функция ***Uniform***, имеющая два параметра: нижнюю и верхнюю границы возможных значений случайной величины.

2. **Экспоненциальное распределение**. Данное распределение является одним из самых распространенных для моделирования простейших обслуживающих систем. Например, с помощью экспоненциального закона можно смоделировать случайную величину, описывающую длительность обслуживания клиента на кассе, на автозаправочной станции и т.д. График случайной величины, распределенной по экспоненциальному закону, имеет вид (рис.2).

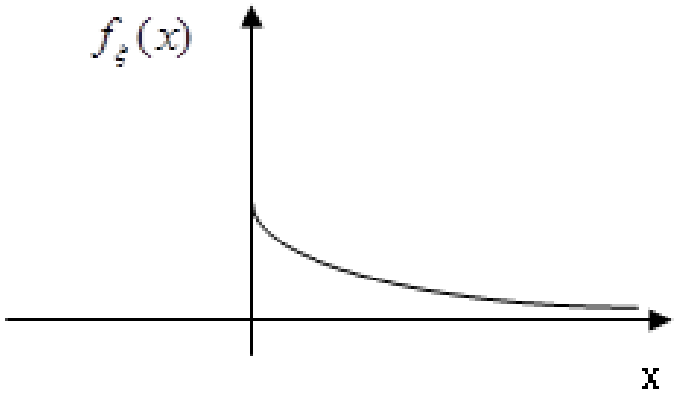


Рис.2. Экспоненциальный закон распределения

Экспоненциальный закон имеет единственный параметр λ, обозначающий интенсивность обслуживания. Интенсивность и длительность обслуживания связаны между собой обратной зависимостью:

$$T_{обсл}=1/\lambda. \quad (1)$$

Для моделирования случайной величины, распределенной по экспоненциальному закону, используется функция ***exponential***. Например, если известно, что длительность обслуживания распределена экспоненциально и составляет в среднем 2 мин, то необходимо воспользоваться функцией exponential(0.5).

Можно также оставить в качестве параметра функции exponential значение (1/T), где T – время обслуживания. Однако, в этом случае их необходимо представить в вещественном виде (иначе произойдет целочисленное деление). Например, в предыдущем случае функцию можно вызвать следующим образом:

exponential(1.0/2.0).

3. **Нормальный закон распределения**. Нормальный закон распределения является самым распространенным законом в теории вероятностей. Он также достаточно часто используется для моделирования случайных величин. График нормально распределенной случайной величины имеет следующий вид (рис.3).

4.

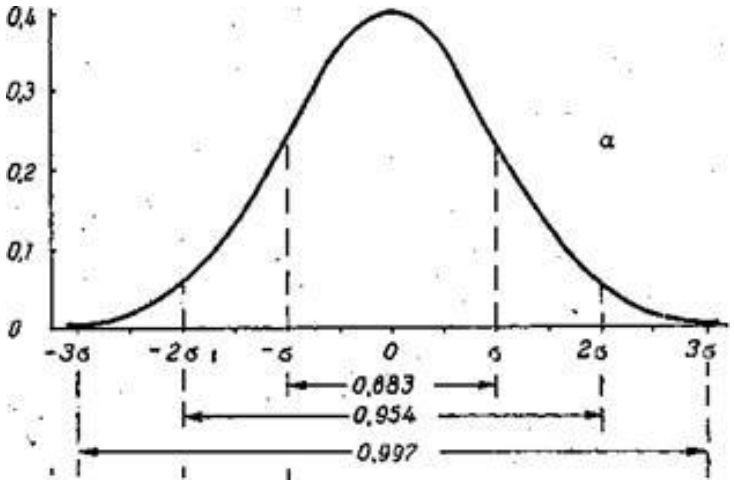


Рис. 3. Нормальный закон распределения

Случайная величина, распределенная по нормальному закону, задается двумя параметрами: ожидаемым значением a (математическим ожиданием) и стандартным отклонением σ (корнем из дисперсии). В Anylogic нормальное распределение задается функцией ***normal(σ, a)***, где первый параметр – стандартное отклонение; второй – ожидаемое значение.

5. **Другие законы**. В Anylogic существует возможность задания большого числа случайных величин. В частности, может потребоваться треугольное (triangle), бета (beta) и другие распределения. Для получения справочной информации обо всех поддерживаемых распределениях необходимо в справке набрать «вероятностные распределения». Выбрав нужную функцию, можно получить справку о ней и ее параметрах.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 3
1. ТЕОРЕТИЧЕСКИЕ АСПЕКТЫ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ . 4
1.1. Модель. Причины использования моделей 4
1.2. Требования, предъявляемые к модели. Функции модели 5
1.3. Классификация моделей 7
1.4. Аналитическое моделирование 10
1.5.Особенности имитационного моделирования 17
1.6. Дискретно-событийное моделирование 20
2. ОБЗОР ОСНОВНЫХ БИБЛИОТЕК Anylogic 24
2.1. Назначение и возможности среды Anylogic 24
2.2 Библиотека моделирования процессов и простейшая дискретно-событийная модель 26
2.3. Диаграмма состояний 43
2.4. Пешеходная библиотека 48
2.5. Моделирование дорожного движения 55
2.6. Системная динамика 61
2.7. Блоки для группового обслуживания заявок 67
2.8. Разметка пространства 74
2.7. Статистика 36
2.8. Параметры 41
3. МОДЕЛИРОВАНИЕ ОБСЛУЖИВАНИЯ С РЕСУРСАМИ 79
3.1. Использование ресурсов 79
3.2. Хранение ресурсов 84
3.3. Перемещение ресурсов 85
4. АГЕНТНОЕ МОДЕЛИРОВАНИЕ 91
4.1. Основы агентного моделирования 91
4.2. Взаимодействие агентов 93
4.3. Области видимости 98
4.4. Примеры 99
5. ЭКСПЕРИМЕНТЫ В AnyLogic 104
5.1. Варьирование параметров 104
5.2. Оптимизационный эксперимент 108
ЗАКЛЮЧЕНИЕ 119
БИБЛИОГРАФИЧЕСКИЙ СПИСОК 120
ПРИЛОЖЕНИЕ 121