ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
РОССИЙСКАЯ АКАДЕМИЯ НАРОДНОГО ХОЗЯЙСТВА И
ГОСУДАРСТВЕННОЙ СЛУЖБЫ при ПРЕЗИДЕНТЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

ОРЛОВСКИЙ ФИЛИАЛ

В.Н. Михайлов

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ

Учебно-методическое пособие

Орел, 2015

ББК 32.973.26-018.2я73
М-69

Рекомендовано к изданию Ученым советом ОФ РАНХиГС

Рецензенты:

Зав. кафедрой «Математика и информатика» Орловского филиала ФГБОУ ВПО «Российская академия народного хозяйства и государственной службы при Президенте Российской Федерации», д-р экон. наук, профессор **В.Г. Шуметов**

Зав. кафедрой «Прикладной информатики и математики» ФГБОУ ВПО «Госуниверситет-УНПК», канд. экон. наук, доцент **А.А. Федотов**

В.Н. Михайлов

М-69 **Имитационное моделирование**: Учебно-методическое пособие. – Орел: Издательство ОФ РАНХГиС, 2015. – 164 с.

В учебно-методическом пособии изложены методы имитационного моделирования в среде AnyLogic. Предлагается современный инструментарий, необходимый сегодняшним специалистам по управлению для понимания сложных систем: причинно-следственные диаграммы, системно-динамические и агентные имитационные модели. Приводятся актуальные примеры из практики государственного и делового управления.

Учебно-методическое пособие предназначено для студентов по направлению подготовки 230700 Прикладная информатика, изучающих дисциплину «Имитационное моделирование». Данный курс также может быть полезен преподавателям и аспирантам, желающим повысить свой профессиональный уровень в области имитационного моделирования.

ББК 32.973.26-018.2я73

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Успехи современных информационных технологий стимулировали рост исследований поведения сложных систем, управление которыми связано с принятием решений в условиях неопределенности. Возрастание уровня сложности ведет к тому, что проблемные ситуации становятся все более трудными для понимания и предсказания, возникает потребность достаточно адекватно описывать поведение, не прибегая к полному и детализированному структурному описанию. Не всегда возможно объяснение поведения сложной системы аналитическими математическими моделями [1,2].

При исследовании сложных систем [3] традиционными методами необходимо задать цели, указать ограничения и определить технологические, экономические или иные строго детерминированные закономерности, при этом внутренние и внешние взаимосвязи также должны выражаться математическими формами. В отличие от этого решение проблем защиты от загрязнения окружающей среды, предотвращения преступлений, роста региональных инфраструктур, анализа конъюнктуры секторов рынка товаров и услуг связано с неясными и противоречивыми целями, а также с альтернативами, диктуемыми зачастую психологическими, социальными и политическими факторами. Следовательно, объяснения поведения подобных сверхсложных объектов должны включать как строго количественные, так и качественные характеристики.

Поскольку мощность математического аппарата недостаточна для нахождения общих аналитических решений сложных проблем, связанных с управлением, то в качестве альтернативного подхода к исследованию поведения сложных систем и принятию решений в условиях неопределенности используется имитационное моделирование, которое ещё в 80-е годы прошлого века [4] прочно заняло лидирующее положение среди инструментов решения практических задач.

Термином имитация обозначается процесс экспериментирования на модели вместо проведения соответствующих опытов с реальной системой. Проведение многовариантных вычислительных экспериментов позволяет выявить некоторые качественные закономерности и сделать определенные обобщения.

Имитационное моделирование представляет собой методологию исследования меняющегося во времени динамического поведения систем в условиях неопределенности. Имитационная модель отражает временной, пространственный и логический аспекты исследуемого процесса, тогда как в других моделях, как правило, присутствует только один из них. Это сравнительно новый класс моделей, которые строятся на программировании. Таким образом, имитационное моделирование явля-

ется универсальным подходом для принятия решений в условиях неопределенности.

Программные средства имитации в своем развитии изменялись на протяжении нескольких поколений, начиная с языков моделирования [5] до генераторов программ [6], интерактивных и интеллектуальных систем [7, 8], распределенных систем моделирования.

В настоящем учебном пособии изложены методологические основы и технология имитационного моделирования, системы моделирования AnyLogic. При разработке модели на AnyLogic можно использовать концепции и средства из нескольких классических областей имитационного моделирования: динамических систем, дискретно-событийного моделирования, системной динамики, агентного моделирования. Кроме того, AnyLogic позволяет интегрировать различные подходы с целью получить более полную картину взаимодействия сложных процессов различной природы.

В данном пособии описываются четыре имитационные модели: дискретно-событийная, системно-динамическая, агентная и комбинированная. Для каждой модели приводится подробная постановка проблемы, разбирается структура, описывается процесс построения модели в среде AnyLogic и изучается ее поведение.

1. ОБЩИЕ ВОПРОСЫ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ

1.1 Модели процессов и систем

Современная парадигма научного исследования состоит в том, что реальные объекты заменяются их упрощенными представлениями, абстракциями, выбираемыми таким образом, чтобы в них была отражена суть явления, те свойства исходных объектов, которые существенны для решения поставленной проблемы [1,2]. Построенный в результате упрощения объект называется моделью. *Модель* — это упрощенный аналог реального объекта или явления, представляющий законы поведения входящих в объект частей и их связи. Построение модели и ее анализ называется *моделированием*. В научной работе моделирование является одним из главных элементов научного познания.

В практической деятельности цель построения модели — решение некоторой проблемы реального мира, которую дорого либо невозможно решать, экспериментируя с реальным объектом. Обычно исходная проблема состоит в анализе существующего или предполагаемого объекта для принятия решения по его управлению. Например, таким объектом может быть географически распределенная система поставщиков сырья, заводов, складов готовой продукции и их транспортные связи.

При построении модели как заменителя реальной системы выделяются те аспекты, которые существенны для решения проблемы, и игнорируются те аспекты, которые усложняют проблему, делают анализ очень сложным или вообще невозможным. Проблема анализа всегда ставится в мире реальных объектов.

Реальные объекты и ситуации обычно сложны, и модели нужны для того, чтобы ограничить эту сложность, дать возможность понять ситуацию, понять тенденции изменения ситуации (спрогнозировать будущее поведение анализируемой системы), принять решение по изменению будущего поведения системы и проверить его. Если модель отражает свойства системы, существенные для решения конкретной проблемы, то анализ модели позволяет вывести характеристики, которые объяснят известные и предскажут новые свойства исследуемой реальной системы без экспериментов с самой системой. С помощью моделирования получено множество впечатляющих результатов в науке, технике и на производстве.

Моделирование особенно важно именно тогда, когда система состоит из многих параллельно функционирующих во времени и взаимодействующих подсистем. Имитационная модель помогает понять сложные системы, предсказать их поведение и развитие процессов в различ-

ных ситуациях и, наконец, дает возможность изменять параметры и даже структуру модели, чтобы направить эти процессы в желаемое русло. Модели позволяют оценить эффект планируемых изменений, выполнить сравнительный анализ качества возможных вариантов решений. Такое моделирование может осуществляться в реальном времени, что позволяет использовать его результаты в различных технологиях.

При изучении естественных объектов исследователь абстрагируется от несущественных, случайных деталей, которые не просто усложняют, но могут и затемнить само явление. Соответствие модели моделируемому объекту или явлению при решении конкретной проблемы называется *адекватностью*. Адекватность определяет возможность использования приближенных результатов, полученных на модели, для решения практической проблемы реального мира. Поскольку адекватность модели определяется только возможностью использования модели для решения конкретной проблемы, адекватная модель не обязательно должна досконально отображать процессы, происходящие в моделируемой системе (или, что тоже самое, модель не обязательно должна отображать "физически правильную" картину мира).

1.2 Виды моделей

Модели можно классифицировать по различным признакам: статические и динамические, непрерывные и дискретные, детерминированные и стохастические, аналитические и имитационные и т. д.

Статические модели оперируют характеристиками и объектами, не изменяющимися во времени. В *динамических* моделях, которые обычно более сложны, изменение параметров во времени является существенным.

Статические модели обычно имеют дело с установившимися процессами, уравнениями балансового типа, с предельными стационарными характеристиками. Моделирование динамических систем состоит в имитации правил перехода системы из одного состояния в другое с течением времени. Под состоянием системы понимается набор значений ее существенных параметров и переменных. Изменение состояния системы во времени в динамических системах — это изменение значений переменных системы в соответствии с законами, определяющими связи переменных и их зависимости друг от друга во времени.

Пакет AnyLogic поддерживает разработку и анализ динамических моделей. Этот инструмент содержит средства для аналитического задания уравнений, описывающих изменение переменных во времени, дает возможность учета модельного времени и содержит средства его продвижения, здесь также имеется язык для выражения логики и описания

прогресса систем под влиянием любого типа событий, в частности, исчерпания таймаута — заданного интервала времени.

Реальные физические объекты функционируют в непрерывном времени, и лля изучения многих проблем физических систем их модели должны быть *непрерывными*. Состояние таких моделей изменяется непрерывно во времени. Это модели движения в реальных координатах, модели химического производства и т. п. Процессы движения объектов и процессы перекачки нефти в модели нефтеналивного порта являются непрерывными.

На более высоком уровне абстракции для многих систем адекватными являются модели, в которых переходы системы из одного состояния в другое можно считать мгновенными, происходящими в дискретные моменты времени. Такие системы называются *дискретными*. Примером мгновенного перехода является изменение числа клиентов банка или количества покупателей в магазине. Очевидно, что дискретные системы — это абстракция, процессы в природе не происходят мгновенно. При построении *модели* магазина для оценки, например, средней длины очереди в кассу при заданном потоке покупателей и известных характеристиках обслуживания кассиром клиентов можно абстрагироваться от этих второстепенных явлений и считать систему дискретной: результаты анализа полученной дискретной модели обычно достаточно точны для принятия обоснованных управленческих решений для подобных систем. В модели нефтеналивного порта мгновенными можно считать, например, переходы светофоров на входе в гавань из состояния "запрещено" в состояние "разрешено".

На еще более высоком уровне абстракции при анализе систем также используются непрерывные модели, что характерно для системной динамики. Потоки машин на автострадах, потребительский спрос, распространение инфекции среди населения часто удобно описывать с помощью взаимозависимостей непрерывных переменных, описывающих количества, интенсивности изменения этих количеств, степени влияния одних количеств на другие. Соотношения таких переменных выражаются обычно дифференциальными уравнениями.

Во многих случаях в реальных системах присутствуют оба типа процессов, и если оба они являются существенными для анализа системы, то и в модели одни процессы должны представляться как непрерывные, другие — как дискретные. Такие модели со смешанным типом процессов называются *гибридными*. Например, если при анализе функционирования магазина существенным является не только количество покупателей, но и пространственное их положение и перемещение покупателей, то модель в этом случае должна представлять смесь непрерывных и дискретных процессов, т. е. это гибридная модель. Другим

примером может служить модель функционирования крупного банка. Поток инвестиций, получение и выдача кредитов в нормальном режиме описывается набором дифференциальных и алгебраических уравнений, т. е. модель является непрерывной. Однако существуют ситуации, например дефолт (дискретное событие), в результате чего возникает паника у населения, и с этого момента система описывается совершенно другой непрерывной моделью. Модель данного процесса на том уровне абстракции, на котором мы хотим адекватно описать оба режима работы банка и переход между режимами, должна включать как описание непрерывных процессов, так и дискретные события, а также их взаимозависимости.

Пакет AnyLogic поддерживает описание как непрерывных, так и дискретных процессов. Что более важно, в среде AnyLogic можно строить и их нетривиальные композиции, т. е. гибридные модели, причем самым естественным образом. AnyLogic позволяет реализовать модель, фактически, на любом уровне абстракции (детальности). Выполнение гибридных моделей в AnyLogic основано на современных результатах теории гибридных динамических систем.

При моделировании сложных реальных систем исследователь часто сталкивается с ситуациями, в которых случайные воздействия играют существенную роль. *Стохастические* модели, в отличие от *детерминированных,* учитывают вероятностный характер параметров моделируемого объекта. Например, в модели нефтеналивного порта не могут быть определены точно моменты прихода в порт танкеров. Данные моменты являются случайными величинами, потому модель эта является стохастической: значения переменных величин модели, которые зависят от реализаций случайных величин, сами становятся случайными величинами. Анализ подобных моделей выполняется на компьютере на основе статистики, набираемой в ходе имитационных экспериментов при многократном прогоне модели для различных значений исходных случайных величин, выбранных в соответствии с их статистическими характеристиками.

AnyLogic содержит средства для генерации случайных величин и статистической обработки результатов компьютерных экспериментов, средства автоматического накопления реализаций и определения выборочных характеристик исследуемых статистических процессов. AnyLogic включает генераторы случайных чисел для множества распределений. Разработчик модели может использовать также свой собственный генератор случайных величин, построенный в соответствии с данными наблюдений над реальной системой.

Использование абстракций при решении проблем с помощью моделей часто состоит в применении того или иного математического ап-

парата. Простейшими математическими моделями являются алгебраические соотношения, и анализ модели часто сводится к аналитическому решению этих уравнений. Некоторые динамические системы можно описать в замкнутой форме, например, в виде систем линейных дифференциальных и алгебраических уравнений и получить решение аналитически. Такое моделирование называется *аналитическим*. При аналитическом моделировании процессы функционирования исследуемой системы записываются в виде алгебраических, интегральных, дифференциальных уравнений и логических соотношений, и в некоторых случаях анализ этих соотношений можно выполнить с помощью аналитических преобразований. Современным средством поддержки аналитического моделирования являются электронные таблицы типа MS Excel.

Однако использование чисто аналитических методов при моделировании реальных систем сталкивается с серьезными трудностями: классические математические модели, допускающие аналитическое решение, в большинстве случаев к реальным задачам неприменимы. Например, в модели нефтеналивного порта построить аналитическую формулу для оценки коэффициента использования оборудования невозможно хотя бы потому, что в системе существуют стохастические процессы, есть приоритеты обработки заявок на использование ресурсов, внутренний параллелизм в обрабатывающих подсистемах, прерывания работы и т. п. Даже если аналитическую модель удается построить, для реальных систем они часто являются существенно нелинейными, и чисто математические соотношения в них обычно дополняются логико-семантическими операциями, а для них аналитического решения не существует. Поэтому при анализе систем часто стоит выбор между моделью, которая является реалистическим аналогом реальной ситуации, но не разрешимой аналитически, и более простой, но неадекватной моделью, математический анализ которой возможен.

При *имитационном* моделировании структура моделируемой системы — ее подсистемы и связи — непосредственно представлена структурой модели, а процесс функционирования подсистем, выраженный в виде правил и уравнений, связывающих переменные, имитируется на компьютере. AnyLogic — это среда имитационного моделирования. Разнообразные средства спецификации и анализа результатов, имеющиеся в AnyLogic, позволяют строить модели, имитирующие работу моделируемой системы фактически с любой желаемой степенью адекватности, и выполнять анализ модели на компьютере без проведения аналитических преобразований.

2. ЛАБОРАТОРНЫЕ РАБОТЫ

Лабораторная работа 1. Дискретно-событийное моделирование

1. Цель работы

Получить практические навыки по дискретно-событийному моделированию на примере модели банковского отделения.

2. Краткие теоретические сведения

Библиотека Моделирования Процессов AnyLogic подерживает дискретно-событийный, или, если быть более точным, "процессный" подход моделирования. С помощью объектов Библиотеки Моделирования Процессов Вы можете моделировать системы реального мира, динамика которых представляется как последовательность операций (прибытие, задержка, захват ресурса, разделение, ...) над некими сущностями (entities, по-русски - транзакты, заявки), представляющими клиентов, документы, звонки, пакеты данных, транспортные средства и т.п. Эти сущности пассивны, они сами не контролируют свою динамику, но могут обладать определёнными атрибутами, влияющими на процесс их обработки (например, тип звонка, сложность работы) или накапливающими статистику (общее время ожидания, стоимость).

Потоковые диаграммы AnyLogic иерархичны, масштабируемы, расширяемы и объектно-ориентированы, что позволяет пользователю моделировать сложные системы любого уровня детальности. Другой важной особенностью Основной библиотеки является возможность создания достаточно сложных анимаций процессных моделей.

Создадим модель простой системы обслуживания, а именно модель банковского отделения. В банковском отделении находятся банкомат и стойки банковских кассиров, что позволяет быстро и эффективно обслуживать посетителей банка. Операции с наличностью клиенты банка производят с помощью банкомата, а более сложные операции, такие как оплата счетов – с помощью кассиров.

3. Задание на лабораторную работу

3.1 Выполняя последовательно шаги проектирования создать модель банковского отделения.

3.2 Провести вычислительный эксперимент и сделать выводы.

4. Порядок выполнения работы.

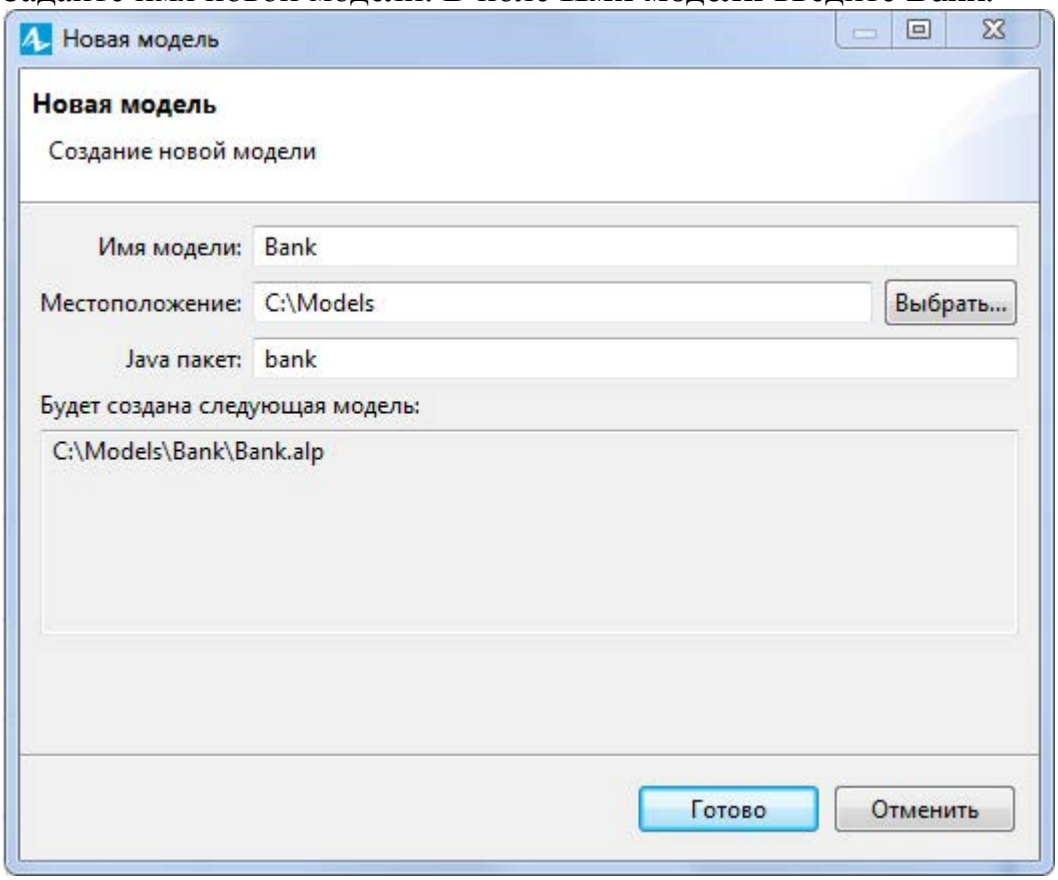
4.1 Создание простой модели

Вначале мы создадим простейшую модель, в которой будем рассматривать обслуживание людей банкоматом.

Задание 1. Создайте новую модель

Щелкните мышью по кнопке панели инструментов **Создать** . Появится диалоговое окно **Новая модель**.

Задайте имя новой модели. В поле **Имя модели** введите Bank.

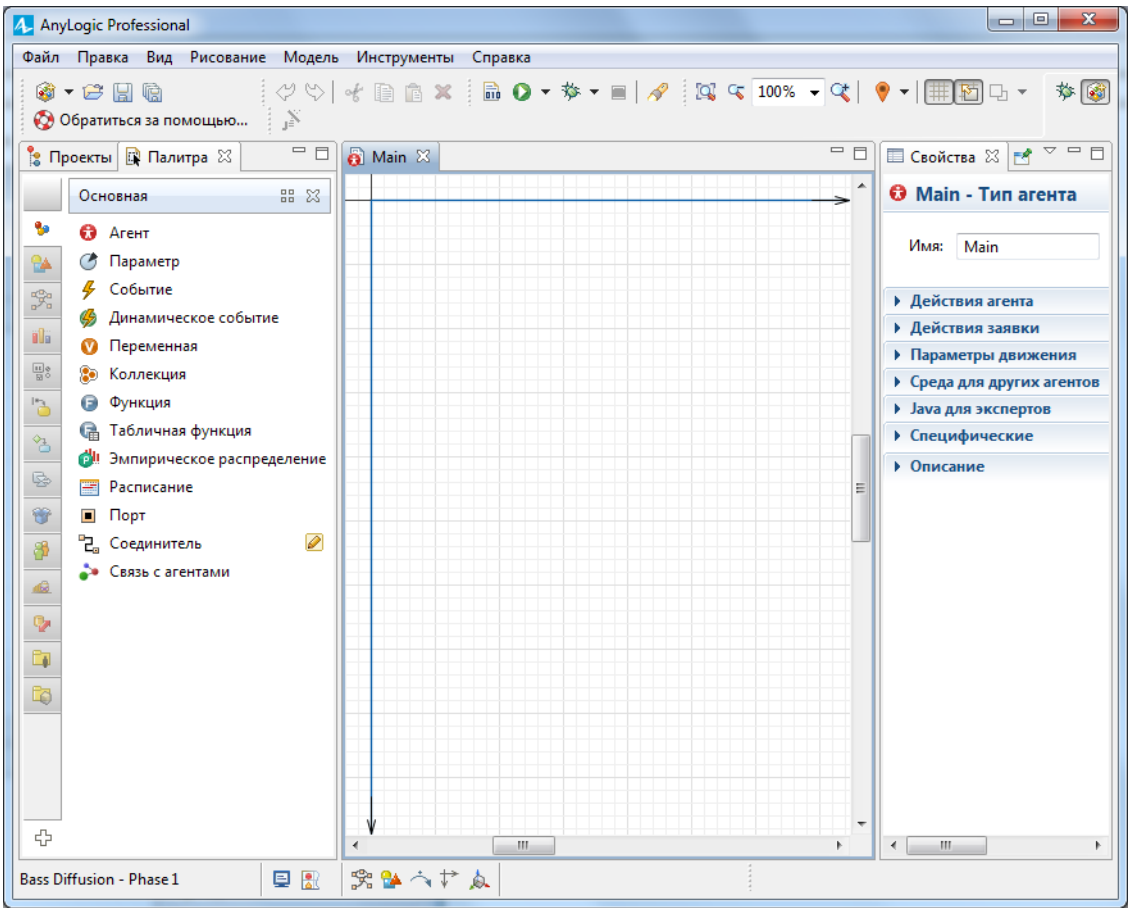


Выберите каталог, в котором будут сохранены файлы модели. Если Вы хотите сменить предложенный по умолчанию каталог на какой-то другой, Вы можете ввести путь к нему в поле **Местоположение** или выбрать этот каталог с помощью диалога навигации по файловой системе, открывающегося по нажатию на кнопку **Выбрать**.

Щелкните мышью по кнопке **Готово**, чтобы завершить процесс.

Вы создали новую модель. В ней уже имеется один тип агента Main и эксперимент Simulation. Агенты - это главные строительные блоки модели AnyLogic. В нашем случае агент Main послужит местом, где мы зададим всю логику модели: здесь мы расположим чертеж банковского отделения и зададим диаграмму процесса потока клиентов.

В центре рабочей области находится графический редактор диаграммы класса агента Main.



В левой части рабочей области находятся панель **Проекты** и панель **Палитра**. Панель **Проекты** обеспечивает легкую навигацию по элементам моделей, открытых в текущий момент времени. Поскольку модель организована иерархически, то она отображается в виде дерева. Панель **Палитра** содержит разделенные по палитрам элементы, которые могут быть добавлены на диаграмму типа агента или эксперимента.

В правой рабочей области будет отображаться панель **Свойства.** Панель **Свойства** используется для просмотра и изменения свойств выбранного в данный момент элемента (или элементов) модели. Когда вы выделяете какой-либо элемент, например, в панели **Проекты** или графическом редакторе, панель **Свойства** показывает свойства выбранного элемента.

Теперь мы можем настроить нашу модель, созданную с помощью **Мастера создания модели**.

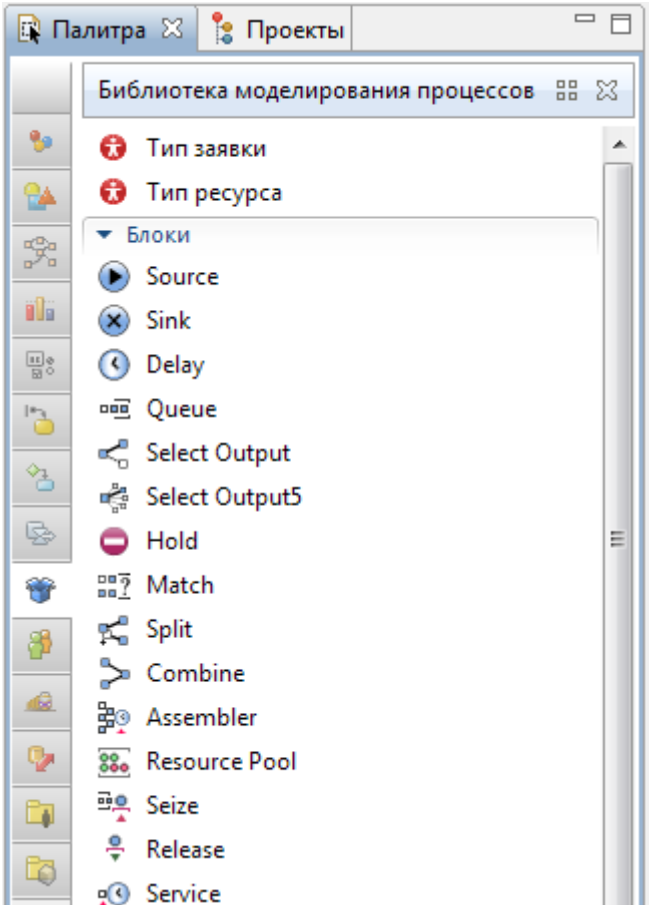
Теперь мы зададим динамику процесса, создав диаграмму из блоков **Библиотеки моделирования процессов**.

Каждый блок задает определенную операцию, которая будет производиться над проходящими по диаграмме процесса заявками.

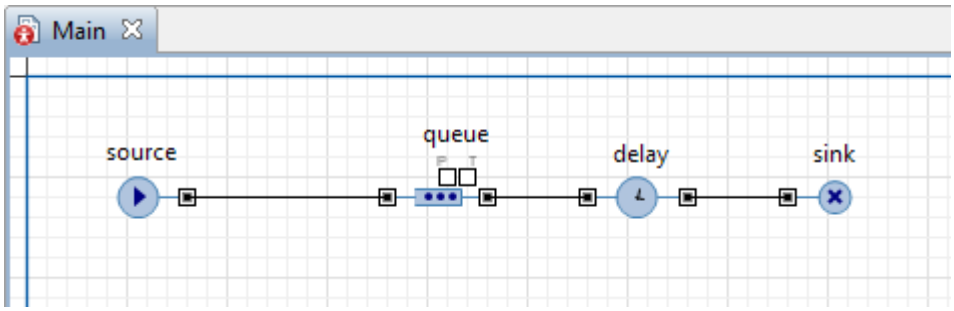
Диаграмма процесса в AnyLogic создается путем добавления объектов библиотеки из палитры на диаграмму агента, соединения их портов и изменения значений свойств блоков в соответствии с требованиями Вашей модели.

Задание 2. Создайте диаграмму процесса

Вначале откройте закладку **Библиотека моделирования процессов** панели **Палитра**, щелкнув мышью по соответствующей иконке на вертикальной панели слева от палитры (обратите внимание на указатель мыши на рисунке):



Добавьте блоки **Библиотеки моделирования процессов** на диаграмму и соедините их, как показано на рисунке. Чтобы добавить объект на диаграмму, перетащите требуемый элемент из палитры в графический редактор.



Пока вы перетаскиваете блоки и располагаете их рядом друг с другом, Вы можете видеть, как появляются соединительные линии между блоками. Будьте внимательны, эти линии должны соединять только порты, находящиеся с правой или левой стороны иконок.

Данная схема моделирует простейшую систему очереди, состоящую из источника заявок, задержки (и очереди перед задержкой) и финального уничтожения заявок.

Скажем пару слов об этих объектах диаграммы.

Объект **Source** генерирует заявки определенного типа. Обычно он используется в качестве начальной точки диаграммы процесса, формализующей поток заявок. В нашем примере заявками будут посетители банка, а объект **Source** будет моделировать их приход в банковское отделение.

Объект **Queue** моделирует очередь заявок, ожидающих приема объектами, следующими за данным в диаграмме процесса. В нашем случае он будет моделировать очередь клиентов, ждущих освобождения банкомата.

Объект **Delay** задерживает заявки на заданный период времени, представля в нашей модели банкомат, у которого посетитель банковского отделения тратит свое время на проведение необходимой ему операции.

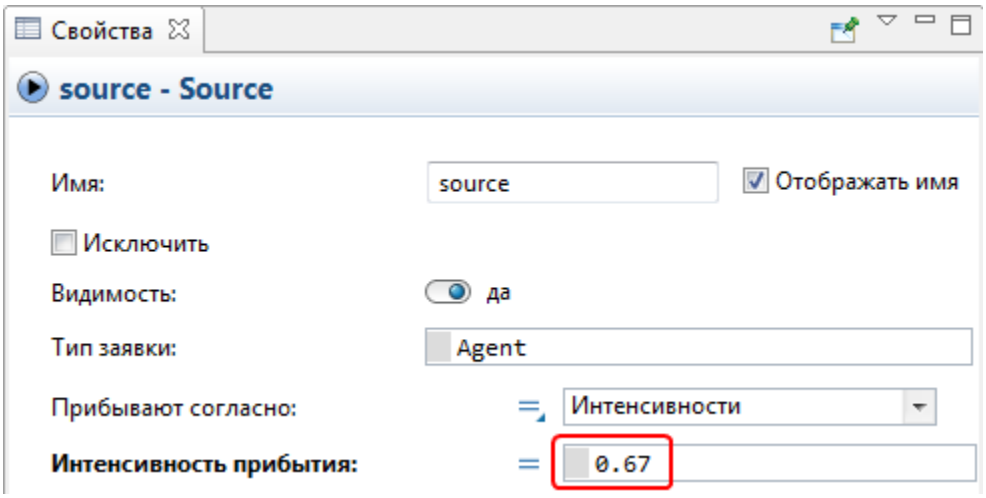
Объект **Sink** уничтожает поступившие заявки. Обычно он используется в качестве конечной точки потока заявок (и диаграммы процесса соответственно).

За детальным описанием объектов **Библиотеки моделирования процессов**, пожалуйста, обращайтесь к *Справочному руководству по Библиотеке моделирования процессов*.

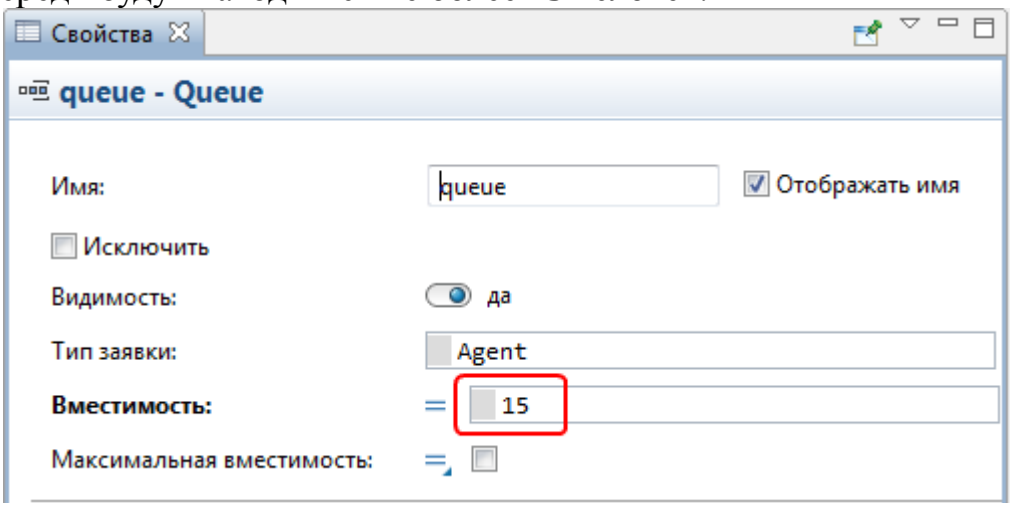
Задание 3. Настройте блоки диаграммы

Чтобы изменить свойства элемента, выделите элемент в графическом редакторе или в панели **Проекты**, щелкнув по нему мышью. Свойства элемента откроются в панели **Свойства**.

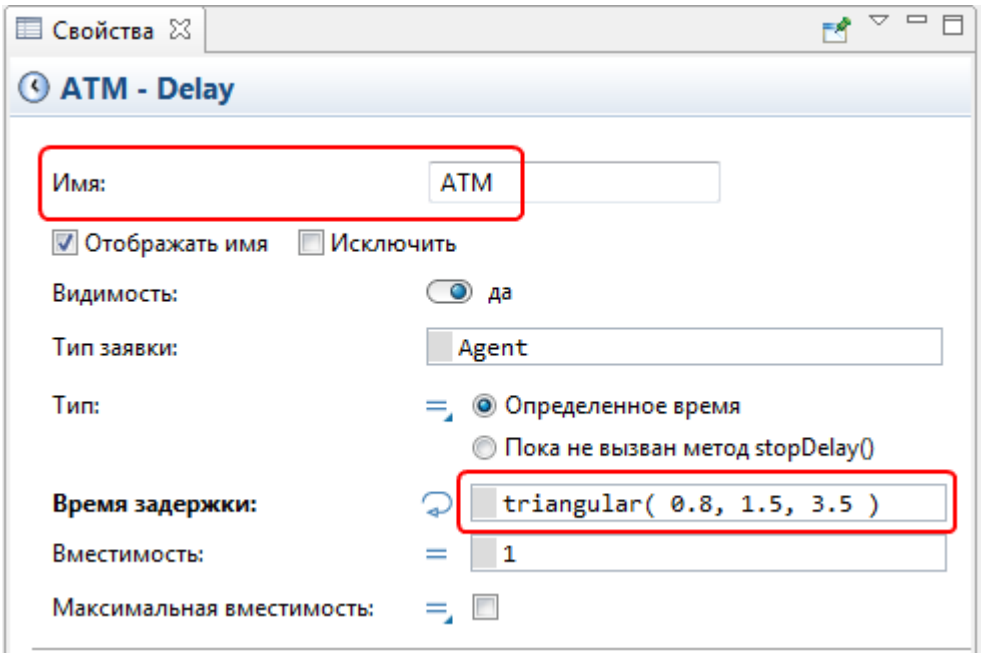
Выделите блок *source*. В панели **Свойства** укажите, как часто должны прибывать клиенты. Введите *0.67* в поле **Интенсивность прибытия**.



Измените свойства блока *queue*. Введите в поле **Вместимость** *15*. В очереди будут находиться не более 15 человек.



Измените свойства блока *delay*. Назовите объект *ATM*. Задайте время обслуживания в поле **Время задержки**, распределенное по треугольному закону со средним значением, равным *1.5*, минимальным - равным *0.8* и максимальным - *3.5* минутам.



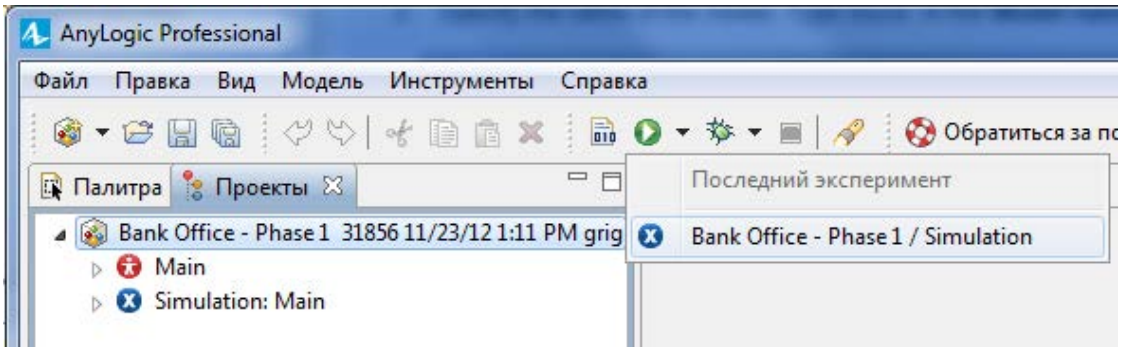
Функция triangular() является стандартной функцией генератора случайных чисел AnyLogic. AnyLogic предоставляет функции и других случайных распределений, таких как нормальное, равномерное, треугольное, и т.д.

Мы закончили моделирование простейшей системы очереди и готовы запустить созданную модель. Сначала постройте Вашу модель с помощью кнопки панели инструментов **Построить модель** (при этом в рабочей области AnyLogic должен быть выбран какой-то элемент именно этой модели). Если в модели есть какие-нибудь ошибки, то построение не будет завершено, и в панель **Ошибки** будет выведена информация об ошибках, обнаруженных в модели. Двойным щелчком мыши по ошибке в этом списке Вы можете перейти к предполагаемому месту ошибки, чтобы исправить ее.

После того, как Вы исправите все ошибки и успешно построите Вашу модель, Вы можете ее запустить. Запуская модель, вы автоматически обновляете ее.

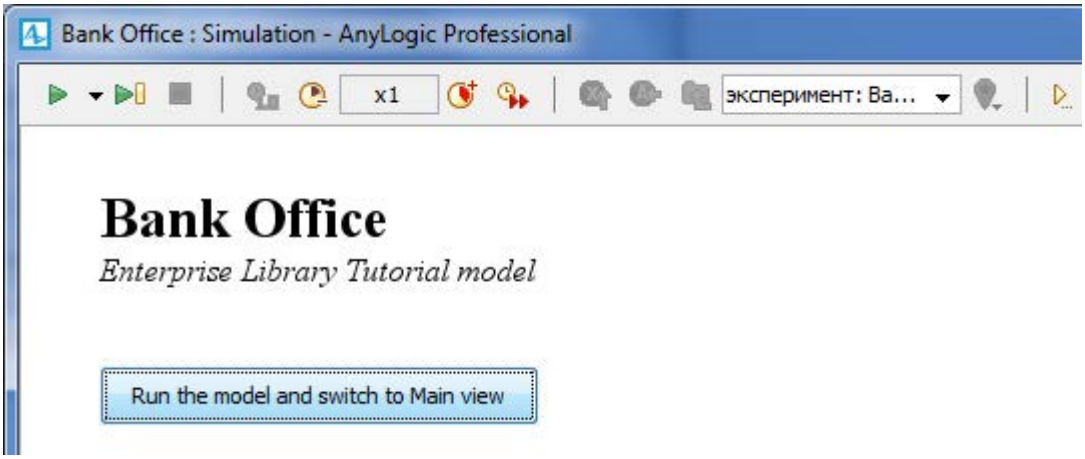
Задание 4. Запустите модель

Щелкните мышью по кнопке панели инструментов **Запустить** и выберите из открывшегося списка эксперимент, который Вы хотите запустить. Эксперимент этой модели будет называться Bank/Simulation.

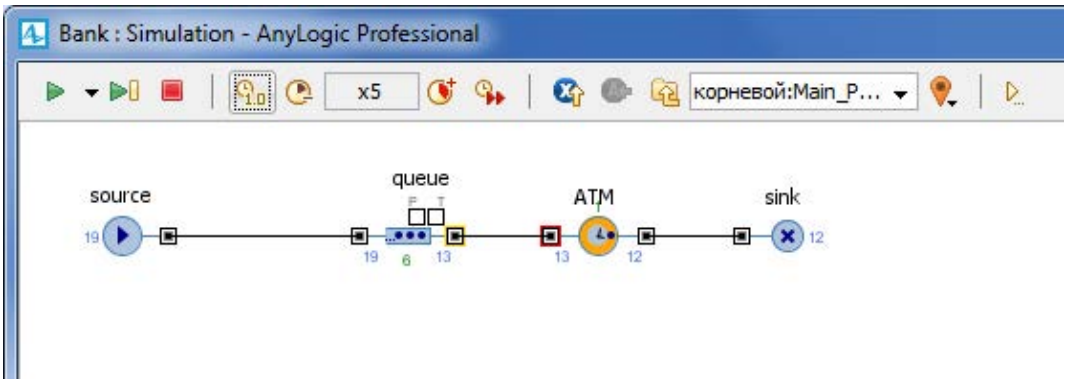


На момент запуска этого конкретного эксперимента наша модель - единственная открытая модель в рабочем пространстве. В дальнейшем будет запускаться тот эксперимент, который запускался Вами в последний раз. Чтобы выбрать какой-то другой эксперимент, Вам будет нужно щелкнуть правой кнопкой мыши по этому эксперименту в панели **Проекты** и выбрать **Запустить** из контекстного меню.

Запустив модель, Вы увидите окно презентации этой модели. В нем будет отображена презентация запущенного эксперимента.

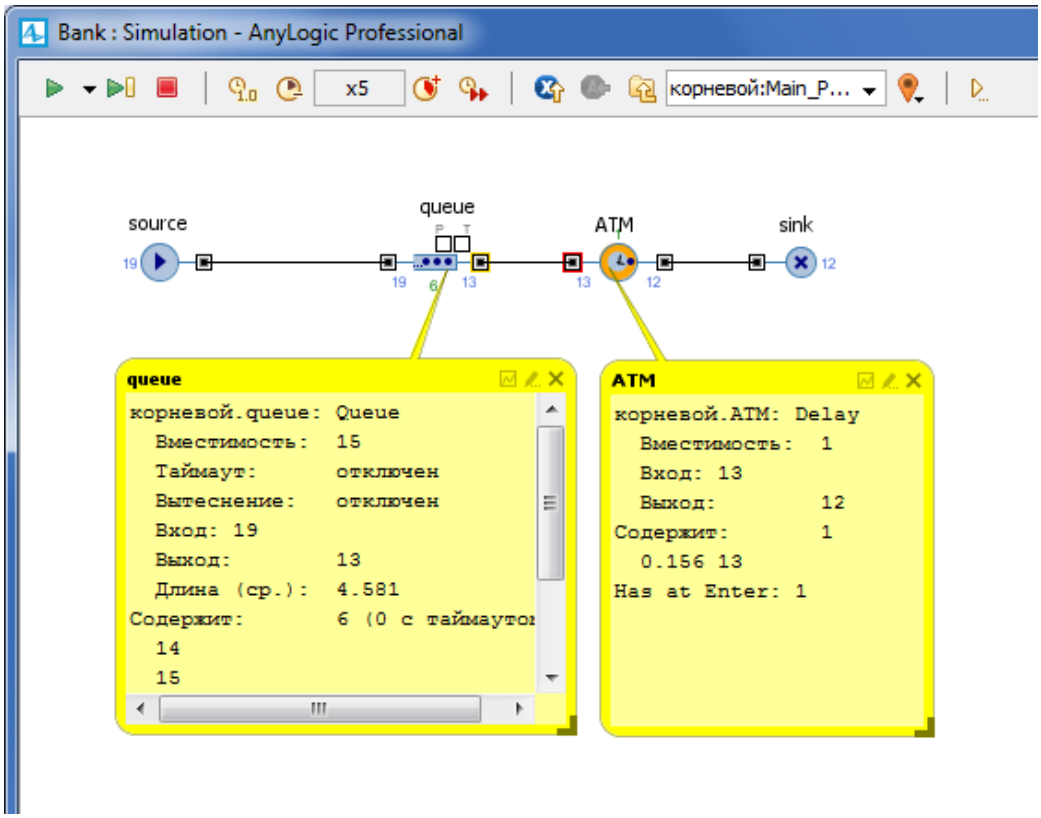


Щелкните по кнопке **Запустить модель и открыть презентацию класса Main**. Тем самым, Вы запустите модель и перейдете к презентации агента верхнего уровня запущенного эксперимента. Для каждой модели, созданной с помощью объектов **Библиотеки моделирования процессов**, автоматически создается блок-схема с наглядной визуализацией процесса, с помощью которой Вы можете изучить текущее состояние модели, например, длину очереди, количество обслуженных человек и так далее.



Вы можете изменить скорость выполнения модели с помощью кнопок панели инструментов **Замедлить** и **Ускорить**.

Вы можете следить за состоянием любого блока диаграммы процесса во время выполнения модели с помощью окна инспекта этого объекта. Чтобы открыть окно инспекта, щелкните мышью по значку блока. В окне инспекта будет отображена базовая инормация по выделенному блоку: например, для блока **Queue** будет отображена вместимость очереди, количество заявок, прошедшее через каждый порт объекта, и т.д. Строка *Contains* отображает количество заявок, находящихся в данный момент на объекте вместе с ID этих заявок.



4.2 Создание анимации модели

Хотя мы и могли анализировать работу запущенной нами только что модели с помощью диаграммы процесса, но куда удобнее было бы иметь более наглядную анимацию моделируемого нами с помощью

анимации. В этом примере мы хотим создать визуализированный план банковского отделения.

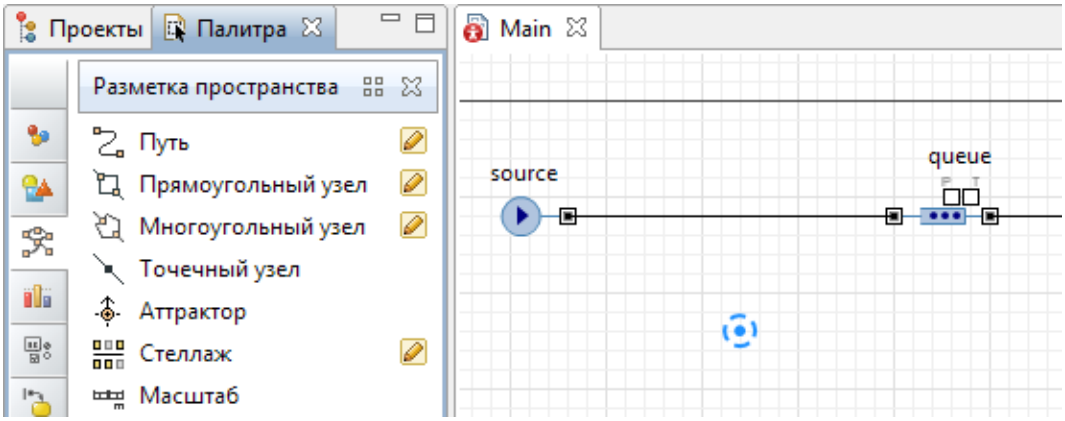
Поскольку в нашем случае нас не интересует конкретное расположение объектов в пространстве, то мы можем просто добавить чисто схематическую анимацию интересующих нас объектов - в нашем случае мы хотим видеть на анимации банкомат и ведущую к нему очередь клиентов.

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

Задание 1. Задайте фигуру анимации банкомата

Нарисуем точечный узел, обозначающий банкомат. Вначале откройте палитру **Разметка пространства** панели **Палитра**.

Перетащите элемент **Точечный узел** из палитры **Разметка пространства** в графический редактор и поместите его под блок-схемой процесса.



Выделите щелчком точечный узел в графическом редакторе, чтобы открыть для него панель **Свойства**. Мы с Вами хотим, чтобы во время моделирования менялся цвет нашей фигуры, поэтому введите выражение, которое будет постоянно вычисляться заново при выполнении модели, в поле **Цвет**:

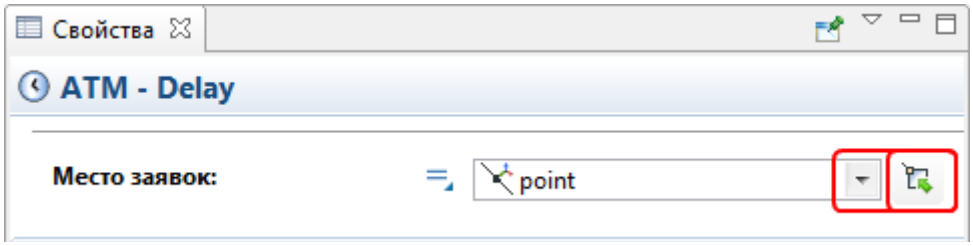
ATM.size() > 0 ? red : green

Здесь ATM – это имя нашего объекта Delay. Функция size() возвращает число человек, обслуживаемых в данный момент времени. Если банкомат занят, то цвет кружка будет красным, в противном случае - зеленым.

Выделите щелчком блок *delay*, названный нами ATM в диаграмме процесса, чтобы открыть его свойства.

Выберите точечный узел *point*, который мы только нарисовали в параметре **Местоположение заявки**. Вы можете выбрать его из выпа-

дающего списка подходящих объектов, щелкнув стрелку "вниз", или выбрать фигуру из графического редактора, предварительно щелкнув кнопку справа от поля параметра (в таком случае все неподходящие объекты в графическом редакторе будут обесцвечены).

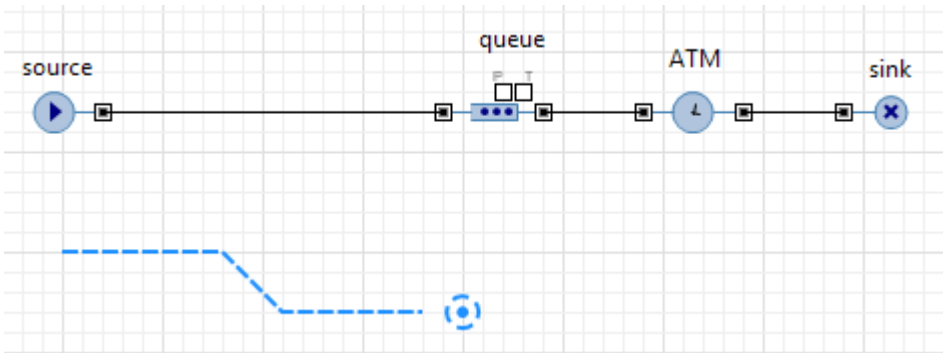


Задание 2. Задайте фигуру анимации очереди к банкомату

Нарисуем путь, обозначающий очередь к банкомату. Вначале откройте палитру **Разметка пространства** панели **Палитра**.

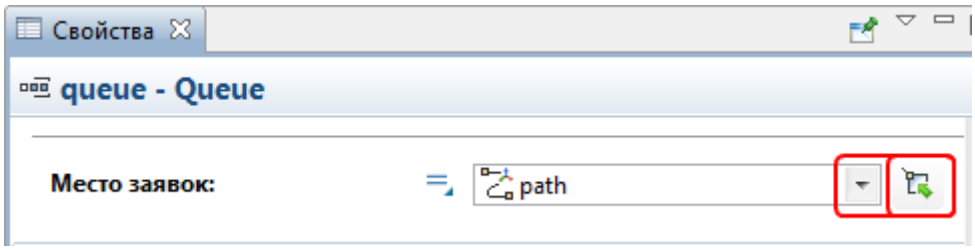
Двойным щелчком выделите элемент **Путь** палитры **Разметка пространства**, чтобы перейти в *режим рисования*.

Теперь Вы можете рисовать путь точка за точкой, последовательно щелкая мышью в тех точках диаграммы, куда Вы хотите поместить вершины ломаной. Чтобы завершить рисование, добавьте последнюю точку ломаной двойным щелчком мыши.



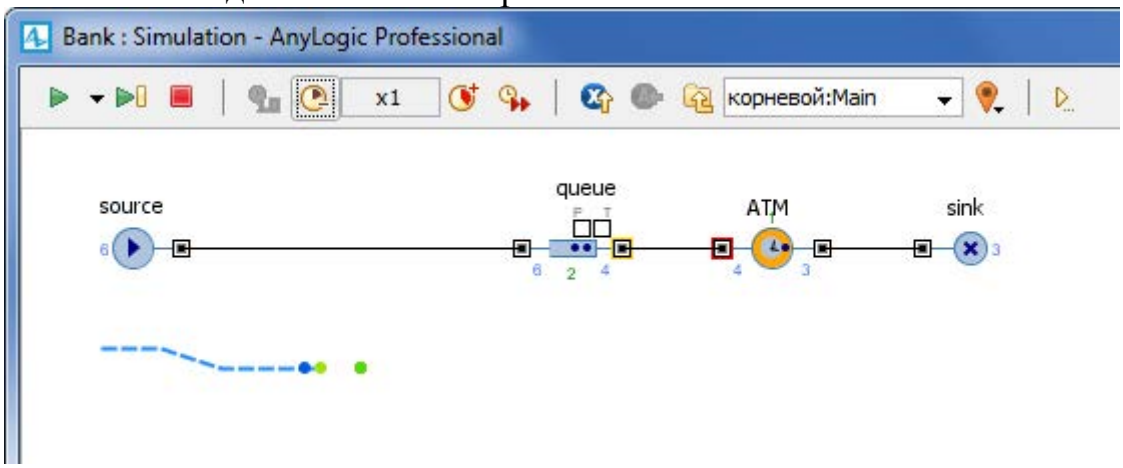
Выделите щелчком блок *queue* в диаграмме процесса, чтобы открыть для него панель **Свойства**.

Выберите путь *path*, который мы только нарисовали в параметре **Местоположение заявки**. Вы можете выбрать его из выпадающего списка подходящих объектов, щелкнув стрелку "вниз", или выбрать фигуру из графического редактора, предварительно щелкнув кнопку справа от поля параметра (в таком случае все неподходящие объекты в графическом редакторе будут обесцвечены).



Теперь Вы можете запустить модель и изучить ее поведение. Для ускорения работы модели, переключитесь в режим виртуального времени, щелкнув мышью по кнопке панели инструментов **Реальное/виртуальное время**. В режиме виртуального времени модель будет выполняться максимально быстро, без привязки модельного времени к реальному.

Запустите модель. Вы увидите, что у нашей модели теперь есть простейшая анимация - банкомат и ведущую к нему очередь клиентов. Цвет фигуры банкомата будет меняться в зависимости от того, обслуживается ли клиент в данный момент времени.



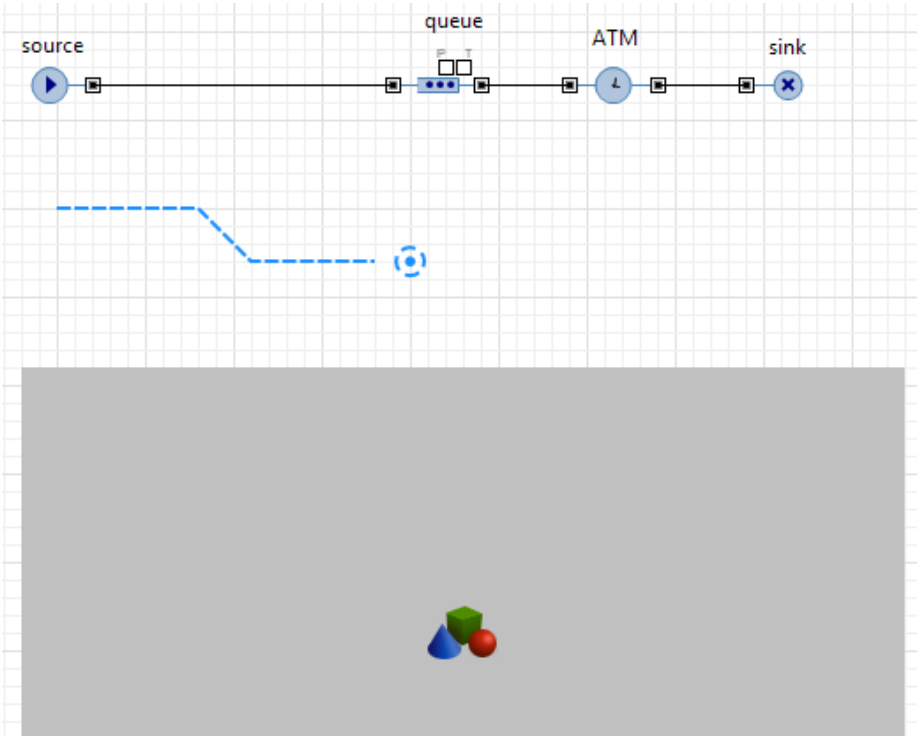
Для добавление 3D анимации нужно добавить на диаграмму активного объекта 3D Окно.

3D Окно используется для задания на диаграмме агента области, в которой во время запуска модели будет отображаться трехмерная анимация этой модели.

Задание 3. Добавьте 3D окно

Перетащите элемент **3D Окно** из секции **3D** палитры **Презентация** в графический редактор.

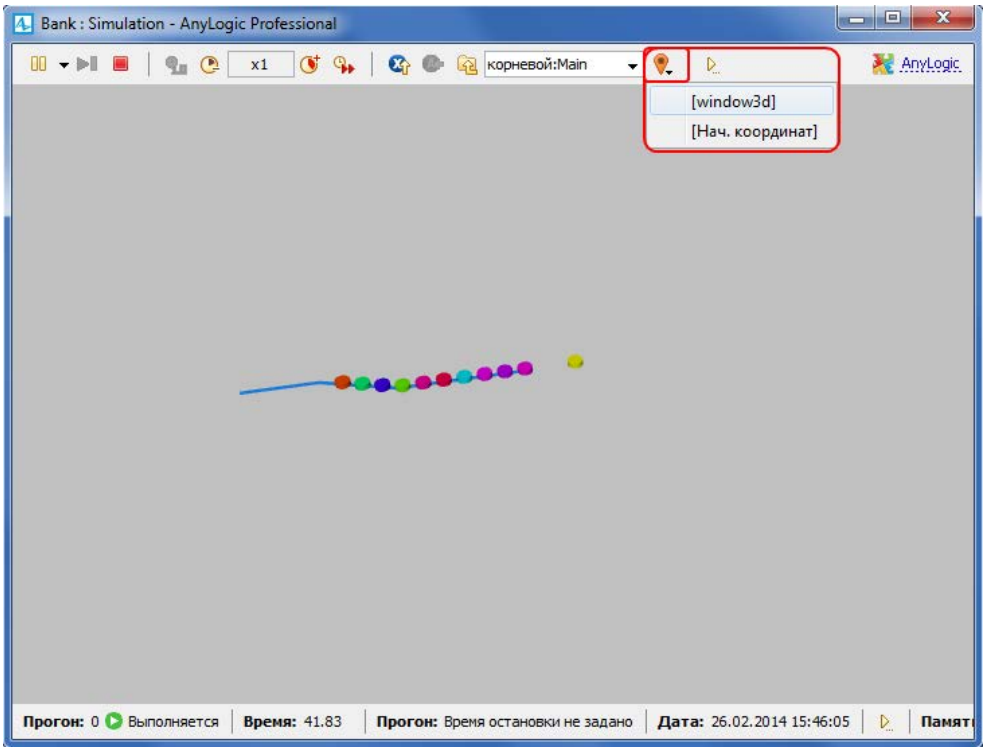
Вы увидите в графическом редакторе закрашенную серым область. Поместите ее туда, где Вы хотите видеть 3D анимацию во время прогона модели:



Задание 4. Запустите модель и опробуйте навигацию по сцене трехмерной анимации

Мы создали простейшую трехмерную анимацию и готовы к тому, чтобы запустить модель и посмотреть на результат нашей работы.

Щелкните кнопку панели инструментов **Показать область...** и выберете **[window3D]**.



Попробуйте "подвигаться" по трехмерной сцене с помощью описанных ниже команд навигации:

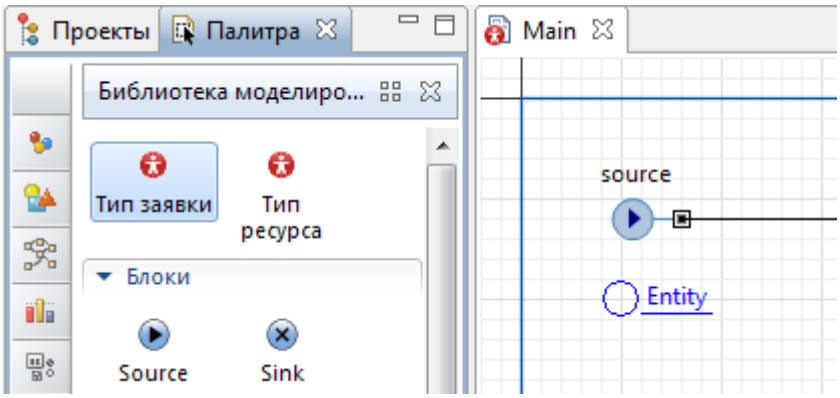
Чтобы	Выполните следующие действия
Переместить сцену	1. Нажмите левую кнопку мыши в области 3D окна и держите ее нажатой. 2. Передвиньте мышь в направлении перемещения.
Повернуть сцену	1. Нажмите клавишу **Alt** и держите ее нажатой. 2. Нажмите левую кнопку мыши в области 3D окна и держите ее нажатой. 3. Передвиньте мышь в направлении вращения.
Приблизить/отдалить сцену	1. Покрутите колесо мыши от/на себя в области 3D окна.

Теперь мы хотим задать фигуру клиента банка. По умолчанию клиенты в нашей модели обозначались цветными точками и отображались цветными цилиндрами в 3D анимации. Если мы хотим задать нестандартный тип клиента и выбрать для него красивую фигуру анимации, нам нужно создать новый тип заявки.

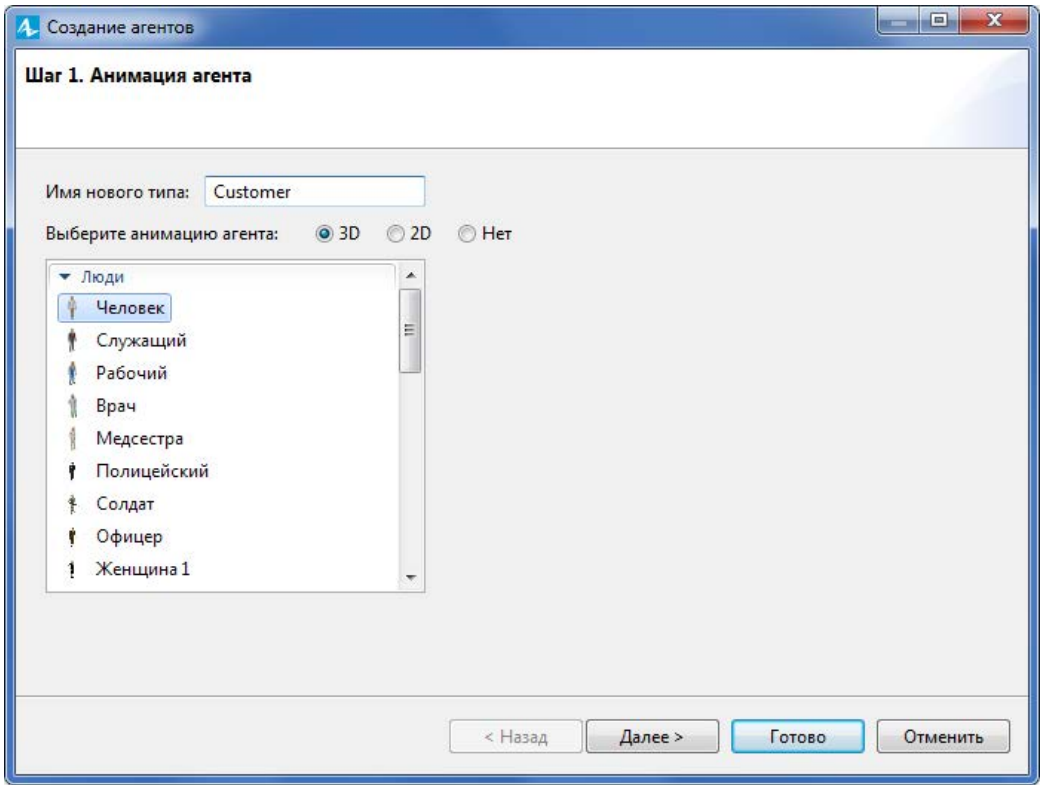
Задание 5. Создайте новый тип заявки

Откройте Библиотеку моделирования процессов в панели Палитра.

Перетащите элемент **Тип заявки** в графический редактор.



Откроется диалоговое окно Мастера создания агентов на шаге **Анимация агента**. Введите *Customer* в поле **Имя нового типа**, выберите опцию **3D** для типа анимации и фигуру анимации *Человек* из списка 3D фигур.

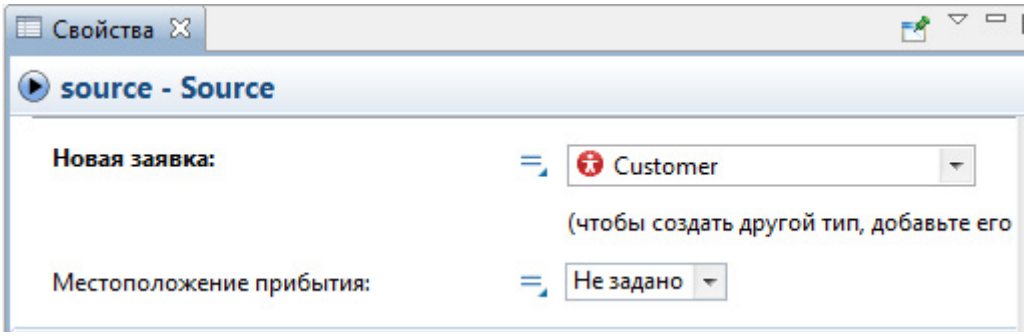


Щелкните **Готово**. Новая диаграмма Customer автоматически откроется. Вы можете найти 3D фигуру *Человек* в начале координат.

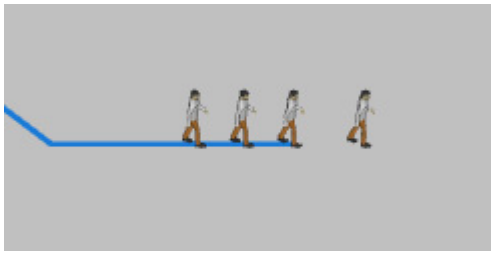
Задание 6. Настройте использование нового типа заявок в блок-схеме

На диаграмме Main, выделите блок *source* в графическом редакторе.

Выберите тип заявок Customer в выпадающем списке параметра **Новая заявка**.



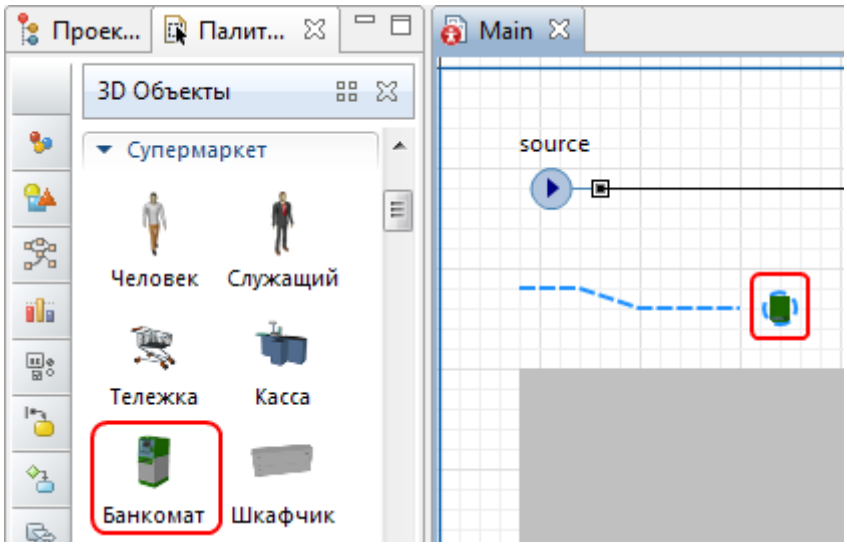
Запустите модель, чтобы увидеть анимацию клиентов в очереди.



Задание 7. Добавьте объект банкомата

Откройте палитру **3D Объекты** в панели **Палитра**.

Перетащите 3D фигуру **Банкомат** из секции палитры **Супермаркет** в графический редактор и поместите ее на точечный узел.

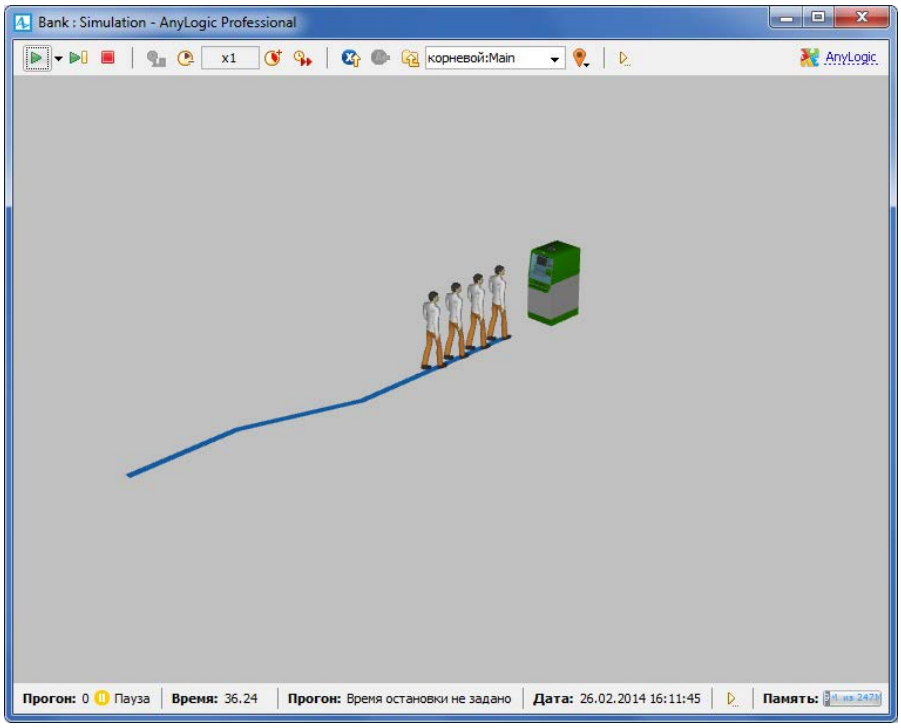


Если Вы сейчас запустите модель и проверите 3D анимацию в режите просмотра **window3D**, Вы заметите, что банкомат стоит не той стороной по направлению к очереди клиентов, и нам необходимо развернуть его в правильную сторону.

Выделите 3D объект банкомата *atm* в графическом редакторе и откройте секцию свойств **Расположение**.

Выберите из выпадающего списка параметра **Поворот Z** *0* градусов.

Запустите модель, чтобы убедиться, что фигура банкомата стоит "лицом" к клиентам.

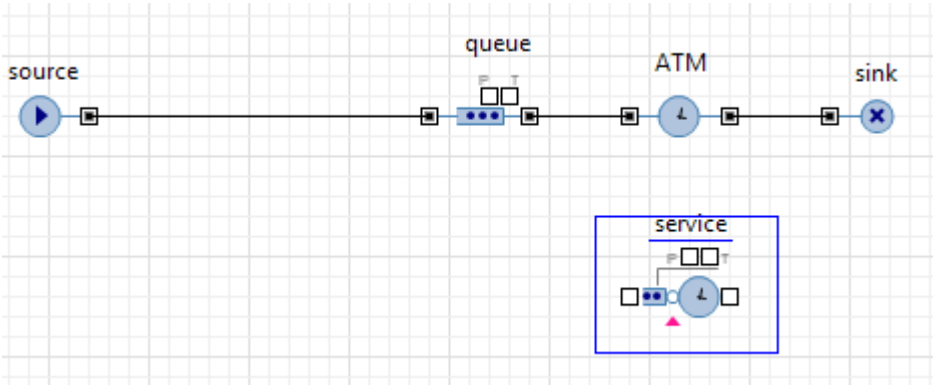


4.3 Добавление клерков

Теперь мы усложним нашу модель, добавив в нее служащих – банковских кассиров. Мы могли бы промоделировать кассиров, как и банкомат, с помощью объектов **Delay**. Но куда более удобным представляется моделирование кассиров с помощью ресурсов. Ресурс – это специальный объект **Библиотеки моделирования процессов**, который может потребоваться заявке для выполнения какой-то задачи. В каждый момент времени ресурс может быть занят только одной заявкой. В нашем примере посетителям банковского отделения (заявкам) необходимо получить помощь у банковских служащих (ресурсов).

Задание 1. Добавьте обслуживание

Откройте **Библиотеку моделирования процессов** в панели **Палитра** и перетащите на диаграмму процесса Main блок **Service**.



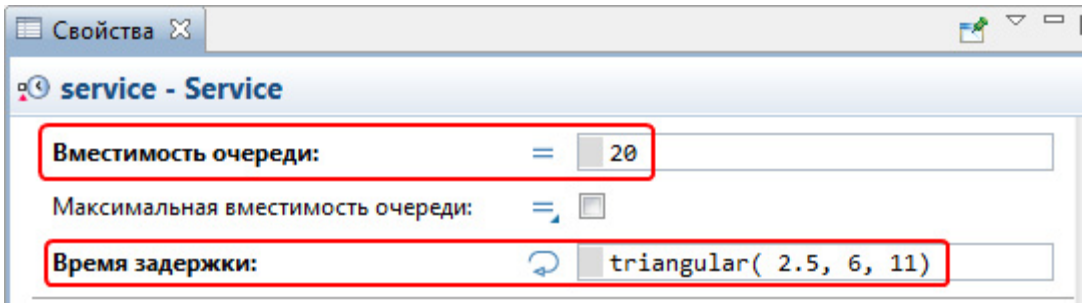
Объект **Service** захватывает для заявки заданное количество ресурсов, задерживает заявку, а затем освобождает захваченные ею ресурсы.

Перейдите в панель **Свойства** блока *service*.

Измените параметры объекта следующим образом:

Ко всем кассирам будет вести одна общая очередь. Задайте максимальное количество человек в этой очереди в поле **Вместимость очереди**: *20*.

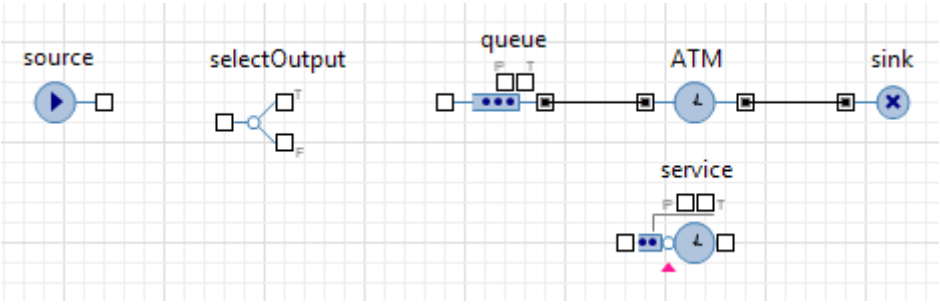
Мы полагаем, что время обслуживания имеет треугольное распределение с минимальным значением равным *2.5*, средним - *6*, и максимальным - *11* минутам. Введите в поле **Время задержки**: triangular(2.5, 6, 11)



Задание 2. Смоделируйте выбор клиентов

Удалите соединитель между блоками *source* и *queue* в диаграмме процесса.

Откройте **Библиотеку моделирования процессов** в панели **Палитра** и перетащите на диаграмму процесса Main блок **SelectOutput** в образовавшееся свободное место.

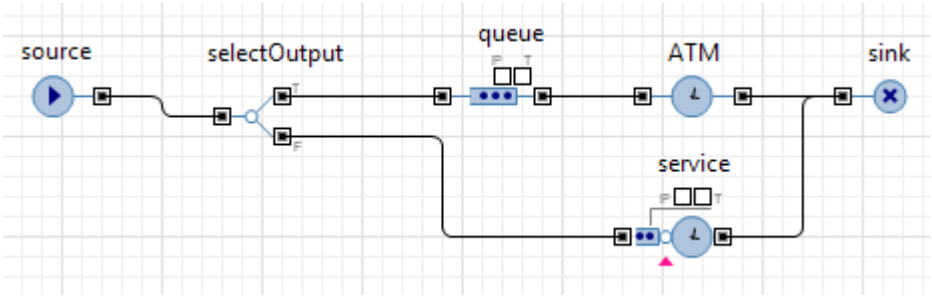


SelectOutput является блоком принятия решения. В зависимости от заданного Вами условия, заявка, поступившая в объект, будет поступать на один из двух выходных портов объекта.

Выделите блок *selectOutput* в диаграмме процесса. В панели **Свойства** этого блока выберите опцию *При выполнении условия* в параметре **Выход True выбирается**. Убедитесь, что в поле **Условие** стоит выражение randomTrue(0.5).

В этом случае к кассирам и банкомату будет приходить примерно равное количество клиентов.

Соедините блоки *selectOutput* и *service* с дургими блоками так, как показано на рисунке ниже:

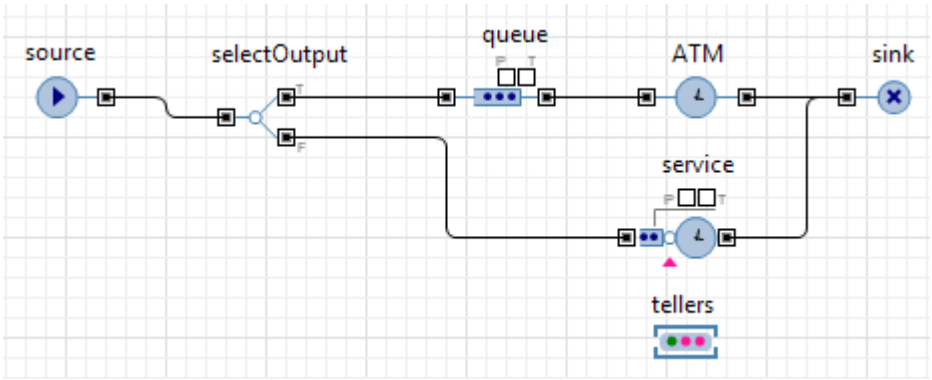


Задание 3. Добавьте ресурсы для сервиса

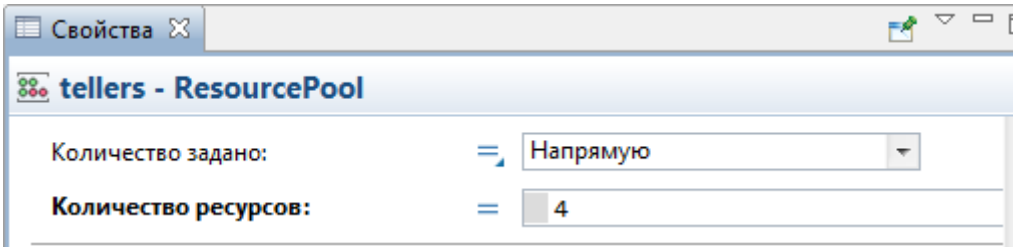
Откройте **Библиотеку моделирования процессов** в панели **Палитра** и перетащите блок **ResourcePool** на диаграмму процесса Main. Объект **ResourcePool** задает ресурсы определенного типа (в нашей модели это будут банковские клерки).

Поместите его, например, под блоком *service* и перейдите в панель **Свойства**.

Назовите объект *tellers*.



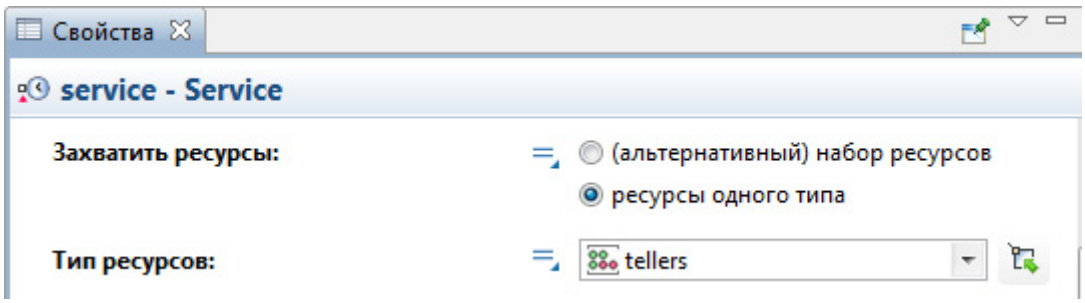
Задайте число кассиров в поле **Кол-во ресурсов**: *4*.



Блок **ResourcePool** указывается в объектах, использующих ресурсы, в нашем случае это блок **Service**. Поэтому нам необходимо изменить свойства блока *service* диаграмму процесса.

Выделите блок *service* и перейдите в панель **Свойства**. Выберите опцию *Ресурсы одного типа* в параметре **Захватить ресурсы**. Затем укажите блок *tellers*, который мы добавили на диаграмму, в параметре

Блок ResourcePool. Вы можете выбрать его из выпадающего списка подходящих объектов, щелкнув стрелку "вниз", или выбрать фигуру из графического редактора, предварительно щелкнув кнопку справа от поля параметра (в таком случае все неподходящие объекты в графическом редакторе будут обесцвечены).



Поскольку наша модель изменилась, мы должны изменить и ее анимацию.

Теперь давайте нарисуем область для ожидания и место обслуживания клиентов кассирами.

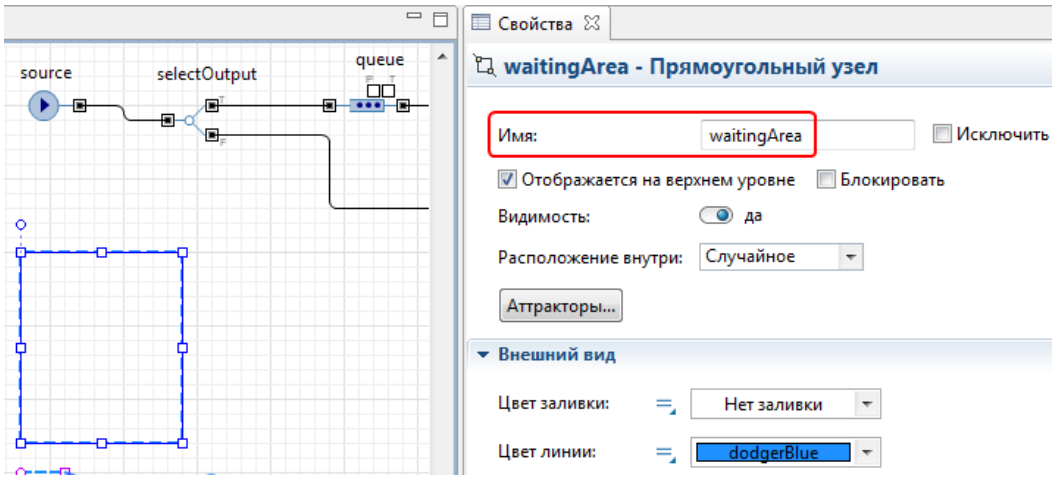
Задание 4. Задайте фигуру разметки для электронной очереди

В этот раз мы будем рисовать место ожидания клиентами, используя прямоугольный узел. Вначале откройте палитру **Разметка пространства** панели **Палитра**.

Двойным щелчком выделите элемент **Прямоугольный узел** палитры **Разметка пространства**, чтобы перейти в*режим рисования*.

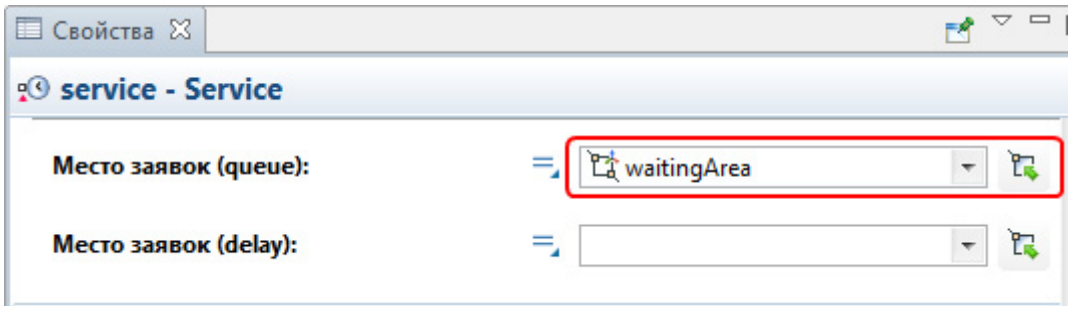
Щелкните мышью в графическом редакторе, чтобы задать вершину верхнего левого угла, затем тащите прямоугольник, не отпуская кнопки мыши. Отпустите, когда прямоугольный узел имеет нужную форму. Вы можете редактировать фигуру и после того, как ее рисование завершено

Назовите эту область *waitingArea*.



Выделите щелчком блок *service* block в диаграмме процесса и перейдите в его свойства.

Выберите только что нарисованный нами узел *waitingArea* в параметре **Местоположение заявки (queue)**.



Задание 5. Задайте фигуру разметки места обслуживания клиентов

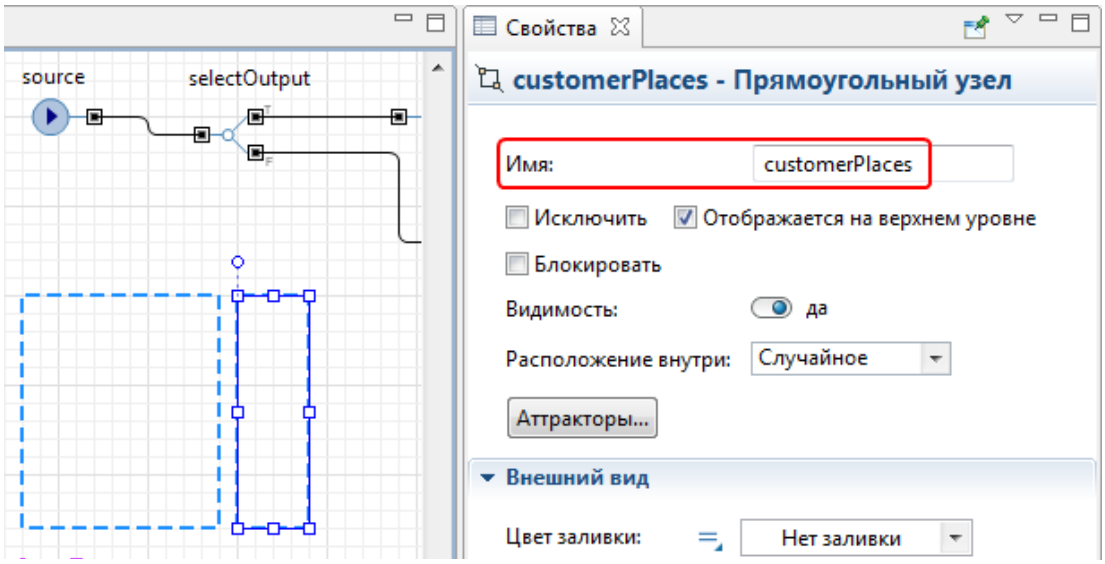
Клиентам банка требуется место, на котором они могли бы находиться во время обслуживания у кассиров. Мы нарисуем такую область, используя прямоугольный узел.

Вначале откройте палитру **Разметка пространства** панели **Палитра**.

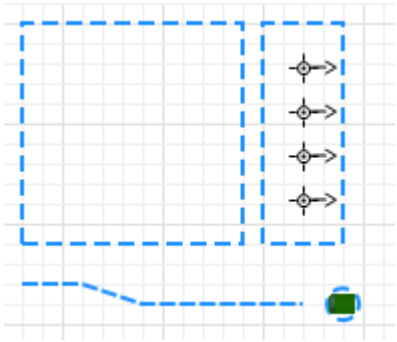
Двойным щелчком выделите элемент **Прямоугольный узел** палитры **Разметка пространства**, чтобы перейти в *режим рисования*.

Щелкните мышью в графическом редакторе, чтобы задать вершину верхнего левого угла, затем тащите прямоугольник, не отпуская кнопки мыши. Отпустите, когда прямоугольный узел имеет нужную форму. Вы можете редактировать фигуру и после того, как ее рисование завершено

Назовите эту область *customerPlaces*.

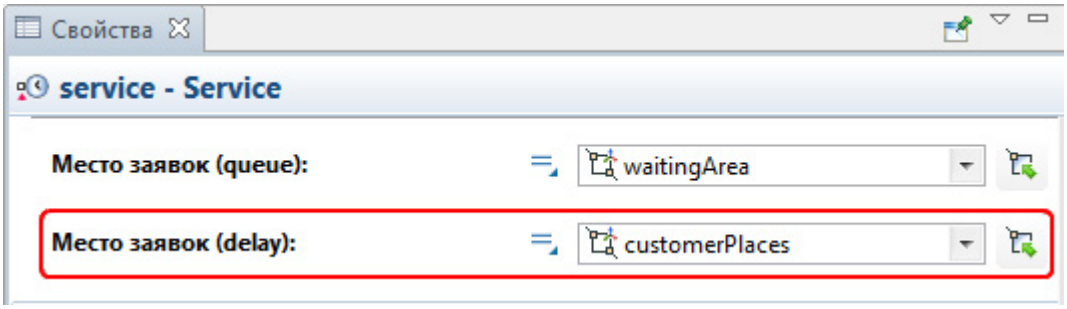


Мы будем использовать аттракторы, чтобы задать местоположение тех клиентов, которые будут обслуживаться у клерков. Выделите узел *customerPlaces* в графическом редакторе и щелкните кнопку **Аттракторы...** в свойствах узла. В открывшемся окне **Аттракторы** укажите число аттракторов *4* в режиме создания **Количество аттракторов**, затем щелкните **OK**. Вы увидите, что четыре аттрактора появились в узле *customerPlaces* на равном расстоянии друг от друга.



Теперь нам необходимо сослаться на эту фигуру в диаграмме процесса. Щелкните блок *service* и перейдите в панель **Свойства** этого блока.

Выберите нарисованный нами узел *customerPlaces* в параметре **Местоположение заявки (delay)**.



Задание 6. Задайте фигуру разметки для кассиров

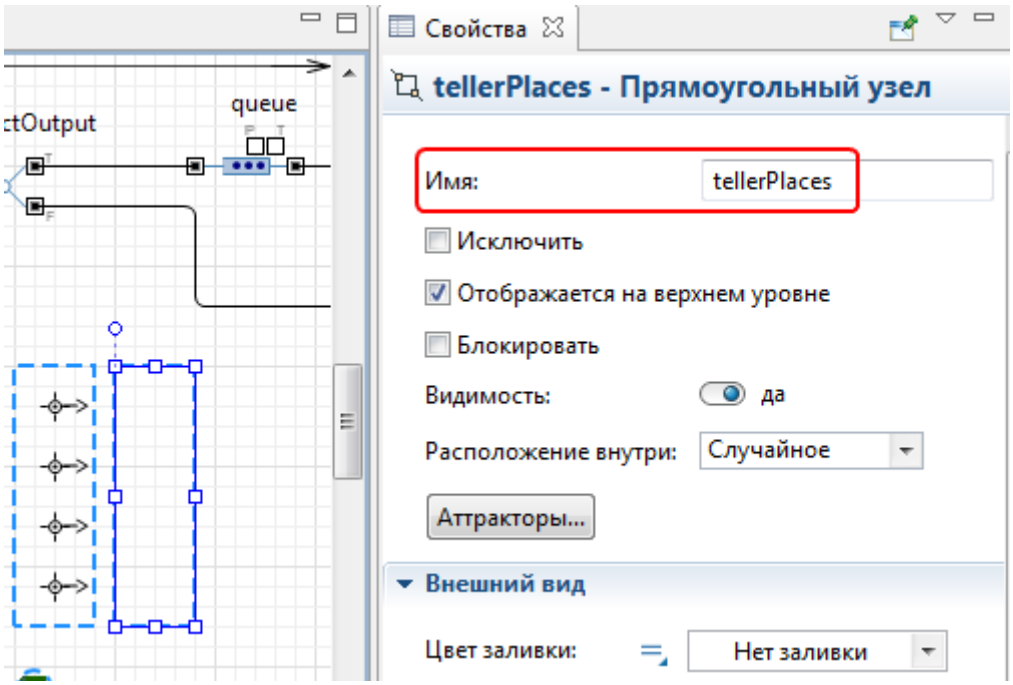
Клиентам банка требуется место, на котором они могли бы находиться во время обслуживания у кассиров. мы нарисуем такую область, используя прямоугольный узел.

Вначале откройте палитру **Разметка пространства** панели **Палитра**.

Двойным щелчком выделите элемент **Прямоугольный узел** палитры **Разметка пространства**, чтобы перейти в*режим рисования*.

Щелкните мышью в графическом редакторе, чтобы задать вершину верхнего левого угла, затем тащите прямоугольник, не отпуская кнопки мыши. Отпустите, когда прямоугольный узел имеет нужную форму. Вы можете редактировать фигуру и после того, как ее рисование завершено

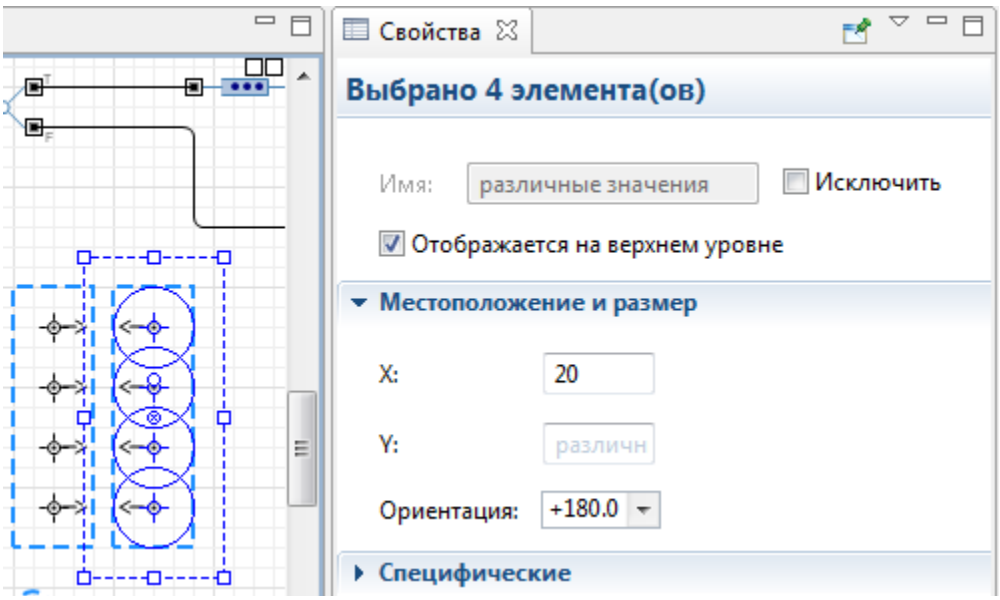
Назовите эту область *tellerPlaces*.



Мы будем использовать аттракторы, чтобы задать местоположение клерков. Выделите узел *tellerPlaces* в графическом редакторе и щелкните кнопку **Аттракторы...** в свойствах узла. В открывшемся окне **Аттракторы** укажите число аттракторов *4* в режиме создания **Количество аттракторов**, затем щелкните **OK**.

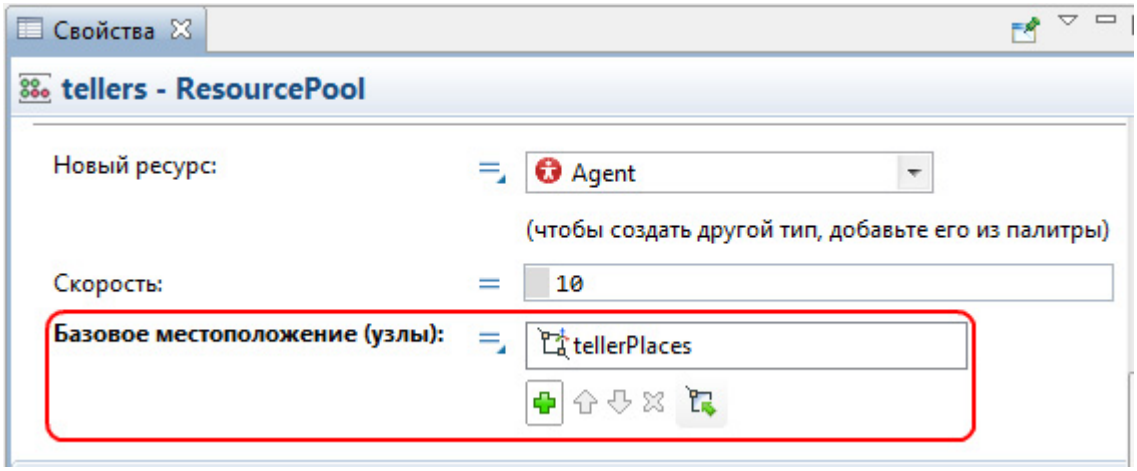
Вы увидите, что четыре аттрактора появились в узле *customerPlaces* на равном расстоянии друг от друга, но они направлены не в ту сторону. Выделите все аттракторы Shift+щелчком и выберите

+180.0 в параметре **Ориентация** секции свойств **Местоположение и размер**.



Щелкните объект *tellers* в диаграмме процесса и перейдите в его свойства.

Выберите нарисованный нами узел *tellerPlaces* в параметре **Базовое местоположение (узлы)**.



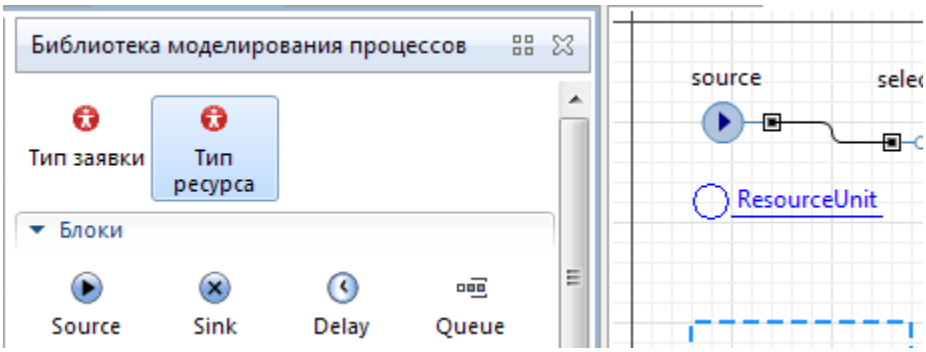
Вы можете запустить модель и наблюдать, как клиенты обслуживаются у банкоматов и проходят к кассирам.

Давайте добавим 3D фигуры клерков в нашу модель. Мы создадим новый тип ресурсов для анимации клерков.

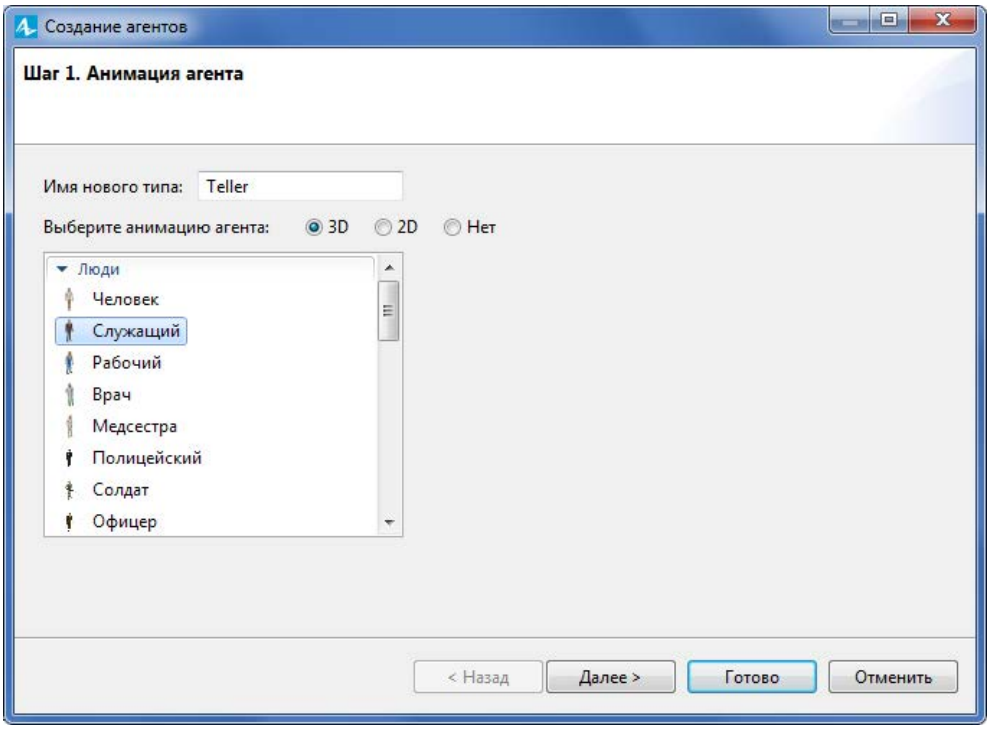
Задание 7. Создайте новый тип ресурсов

Откройте Библиотеку моделирования процессов в панели Палитра.

Перетащите элемент **Тип ресурса** в графический редактор.



Откроется диалоговое окно Мастера создания агентов на шаге **Анимация агента**. Введите *Teller* в поле **Имя нового типа**, выберите опцию **3D** для типа анимации и фигуру анимации *Служащий* из списка 3D фигур.

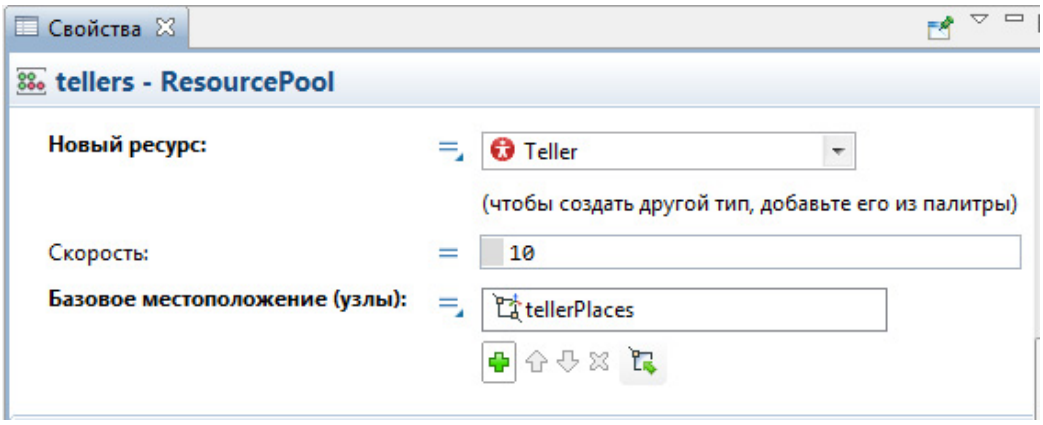


Щелкните **Готово**. Новая диаграмма Teller автоматически откроется. Вы можете найти 3D фигуру *Служащий* в начале координат. Переключитесь обратно на диаграмму Main.

Задание 8. Настройте использование нового типа ресурсов в блок-схеме

На диаграмме Main, выделите блок *tellers* в графическом редакторе.

Выберите тип ресурсов Teller в выпадающем списке параметра **Новый ресурс.**



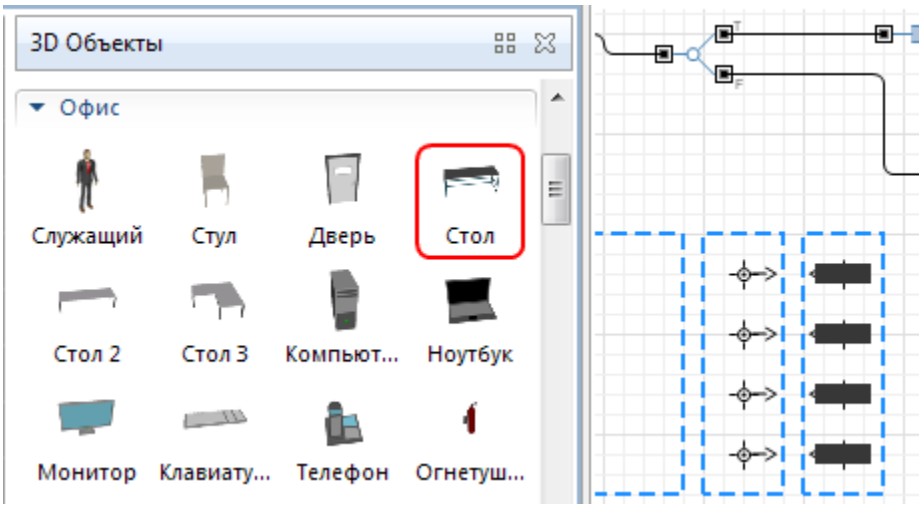
Запустите модель, чтобы увидеть получившуюся анимацию клерков.

Задание 9. Добавьте столы для клерков

Откройте палитру **3D Объекты** в панели **Палитра**.

Перетащите четыре 3D фигуры **Стол** из секции палитры **Офис** в графический редактор и поместите их в узел *tellerPlaces*.

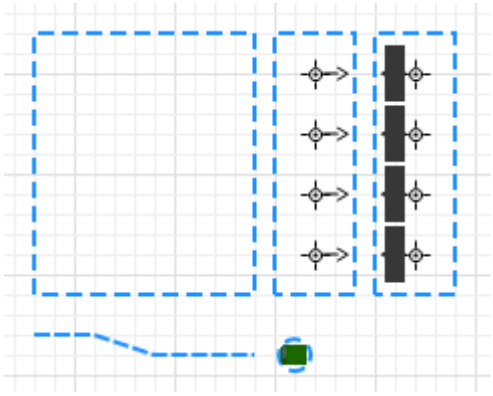
Расположите столы на аттракторах, так как аттракторы обозначают место, где стоят клерки.



Вы заметите, что они стоят не той стороной к клеркам. Выделите все столы методом Shift-щелчок и перейдите в их свойства.

В секции **Расположение** измените параметр **Поворот Z**: *-90.0* градусов.

При необходимости, выровняйте расположение всех восьми аттракторов и столов.



Теперь Вы можете запустить модель и увидеть в 3D анимации, как некоторые клиенты идут к банкомату, а другие обслуживаются у столов клерков.

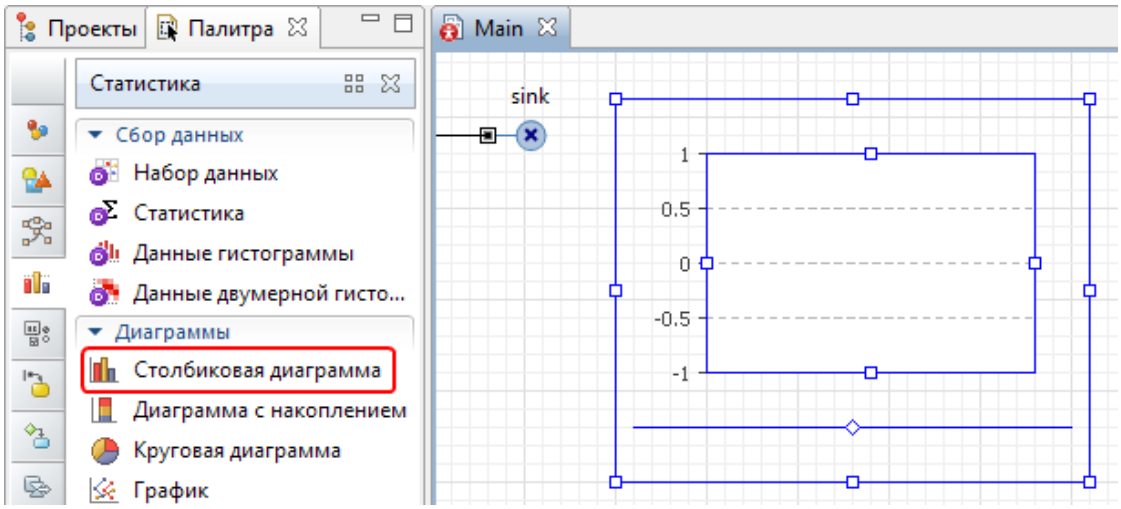
4.4 Добавление статистики модели

AnyLogic предоставляет пользователю удобные средства для сбора статистики по работе блоков диаграммы процесса. Объекты Библиотеки моделирования процессов самостоятельно производят сбор основной статистики. Все, что Вам нужно сделать - это включить сбор статистики для объекта.

Мы можем, например, просмотреть интересующую нас статистику (скажем, статистику занятости банкомата и длины очереди) с помощью диаграмм.

Задание 1. Добавьте диаграмму для отображения средней занятости банкомата

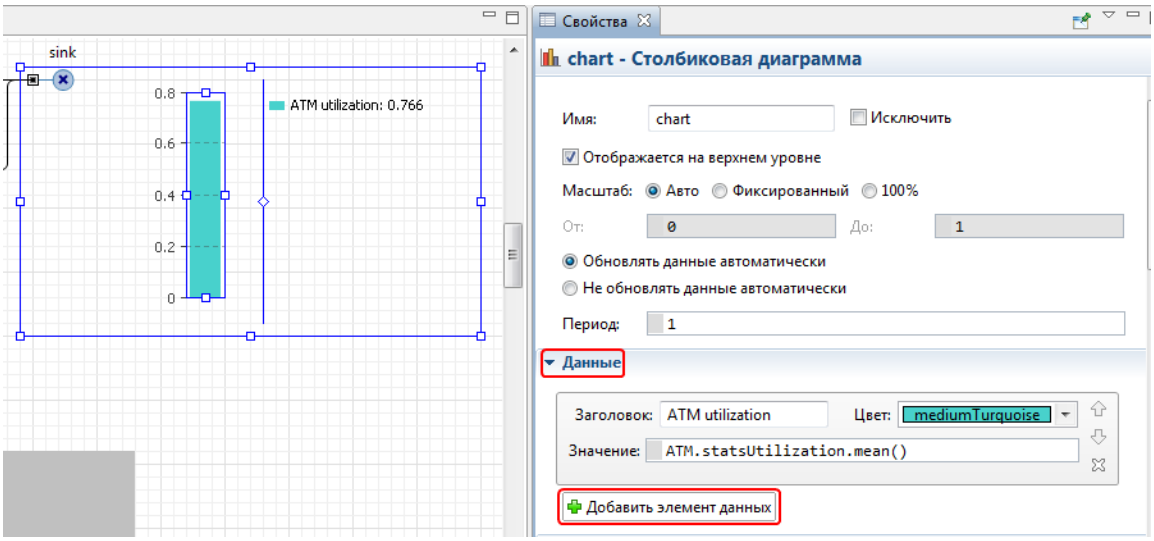
Откройте палитру **Статистика**. Эта палитра содержит элементы сбора данных и статистики, а также диаграммы для визуализации данных и результатов моделирования. Перетащите элемент **Столбиковая диаграмма** из палитры **Статистика** на диаграмму и измените ее размер:



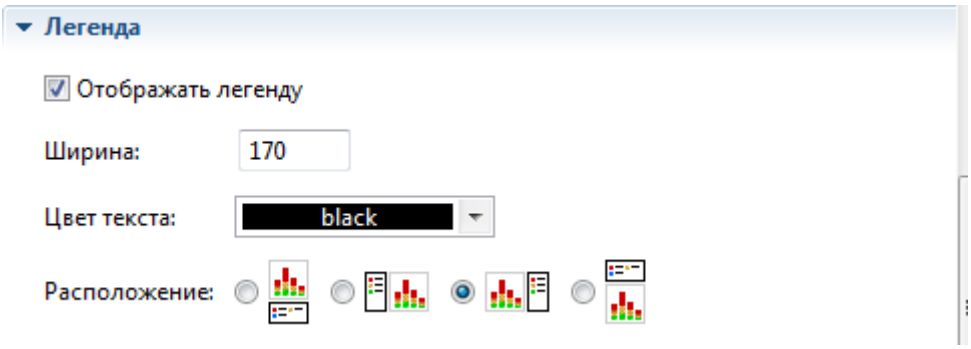
Перейдите в секцию **Данные** свойств столбиковой диаграммы. Щелкните кнопку **Добавить элемент данных**, чтобы задать данные для отображения в диаграмме.

Измените **Заголовок** на *ATM utilization*.

Введите ATM.statsUtilization.mean() в поле **Значение**. Здесь ATM - это имя нашего объекта **Dela**y. У каждого объекта **Delay** есть встроенный набор данных statsUtilization, занимающийся сбором статистики использования этого объекта. Функция mean() возвращает среднее из всех измеренных этим набором данных значений. Вы можете использовать и другие методы сбора статистики, такие, как min() или max().

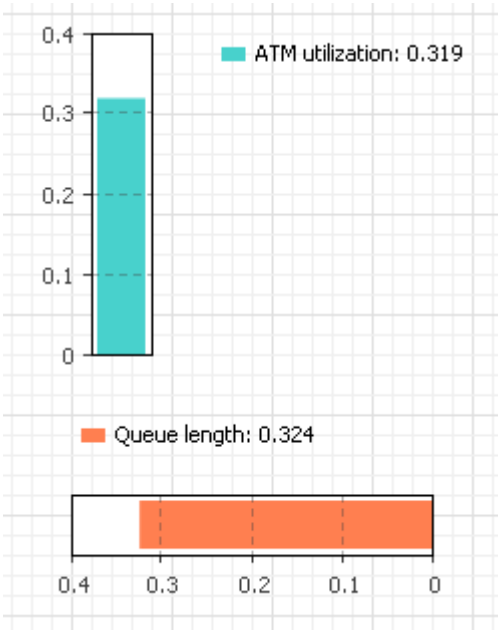


Перейдите в секцию **Легенда** панели **Свойства**. Измените расположение легенды относительно диаграммы (мы хотим, чтобы она отображалась справа).

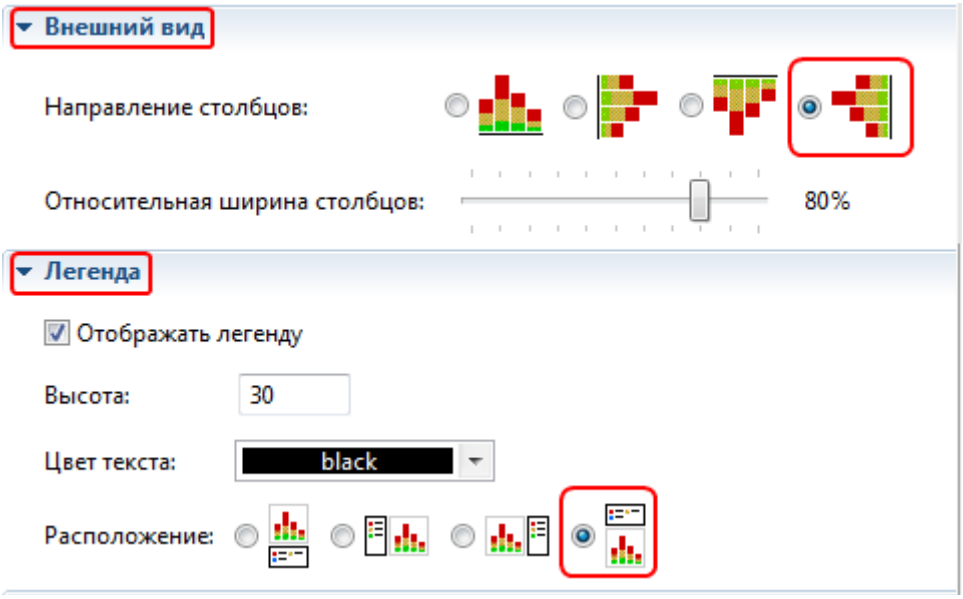


Задание 2. Добавьте диаграмму для отображения средней длины очереди

Аналогичным образом добавьте еще одну столбиковую диаграмму. Измените ее размер так, как показано на рисунке:

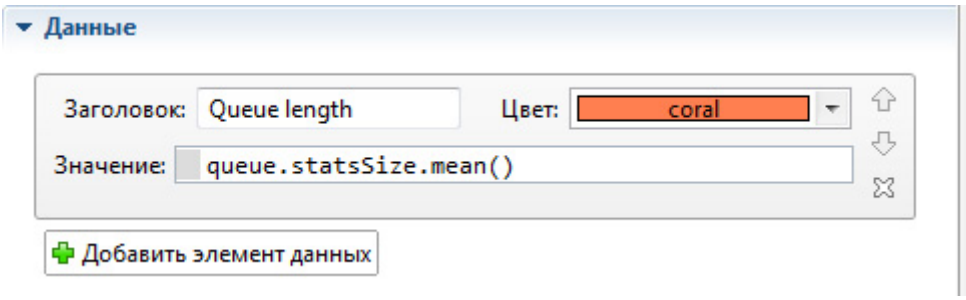


Перейдите в секцию **Внешний вид** панели **Свойства** и выберите последнюю опцию параметра **Направление столбцов**, чтобы столбцы столбиковой диаграммы росли влево. Также измените положение легенды в секции **Легенда** (как показано на рисунке ниже).

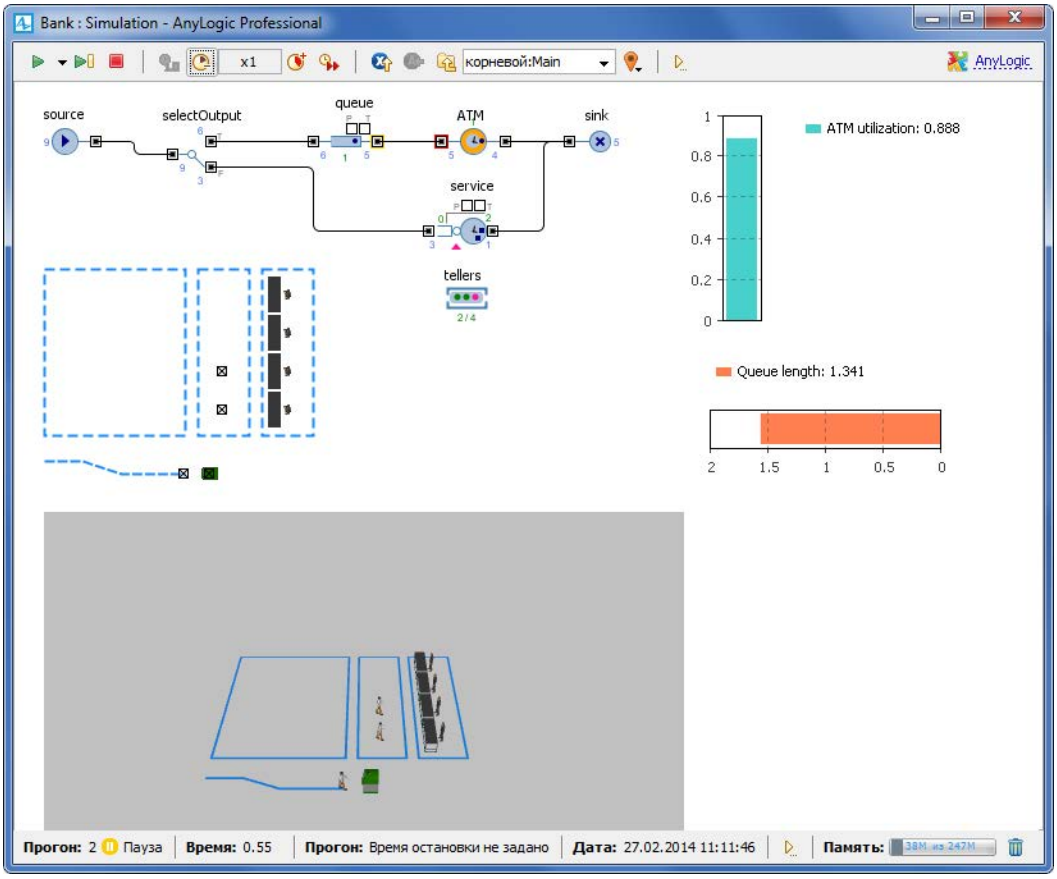


Добавьте элемент данных, который будет отображаться на диаграмме, в секции **Данные**. Задайте **Заголовок**: *Queue length* и задайте **Значение**: queue.statsSize.mean()

Здесь statsSize - это имя объекта типа "статистика" **StatisticsContinuous**, производящего сбор статистики размера очереди объекта **Queue**.



Запустите модель и пронаблюдайте за занятостью банкомата и средней длиной очереди с помощью только что созданных диаграмм.



Мы хотим знать, сколько времени клиент проводит в банковском отделении и сколько времени он теряет, ожидая своей очереди. Мы соберем эту статистику с помощью специальных объектов сбора данных и отобразим собранную статистику распределения времен обслуживания клиентов с помощью гистограмм. Для этого мы будем использовать ранее созданный тип заявки Customer.

Вначале нам необходимо добавить два параметра в нашу модель.

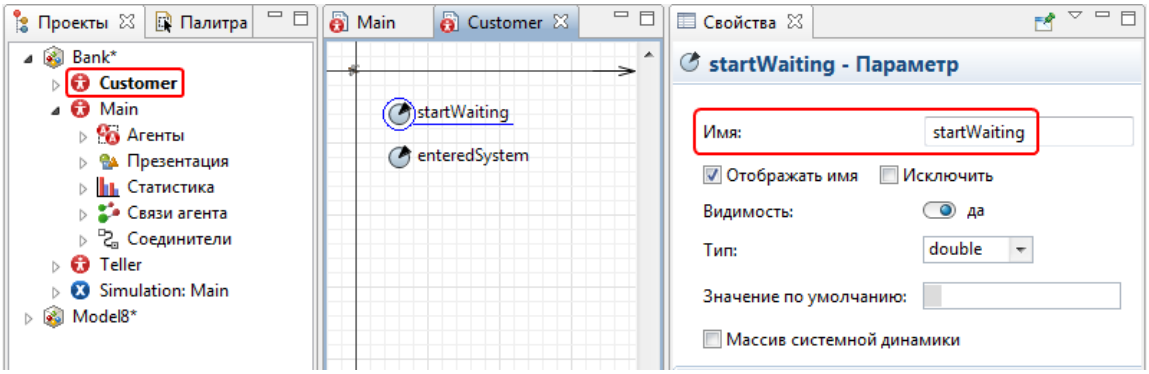
Задание 3. Добавьте параметры

Переключитесь в панель **Проекты**. Дважды щелкните тип заявки Customer, чтобы открыть его диаграмму. Нам необходимо создать параметры на диаграмме агента Customer, так как мы хотим собирать статистику клиентов по времени их обслуживания.

Откройте палитру **Основная** в панели **Палитра**.

Перетащите два элемента **Параметр** на диаграмму Customer.

Назовите параметры *startWaiting* и *enteredSystem*, оставьте тип double, заданный по умолчанию.

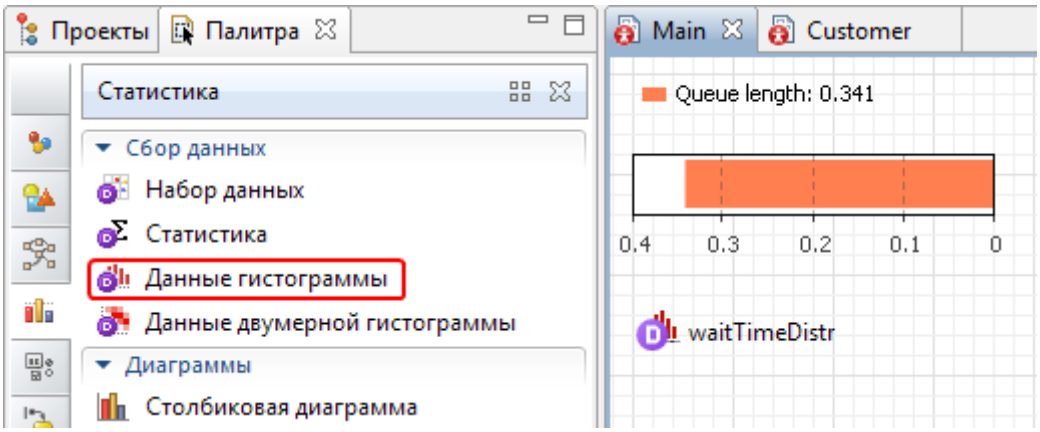


Мы продолжим разрабатывать нашу модель на диаграмме Main.

Добавьте элементы сбора статистики по времени ожидания клиентов и времени пребывания клиентов в системе. Эти элементы будут запоминать соответствующие значения времен для каждого клиента и предоставят пользователю стандартную статистическую информацию: среднее, минимальное, максимальное из измеренных значений, среднеквадратичное отклонение, доверительный интервал для среднего и т.д.).

Задание 4. Добавьте элементы сбора данных

Чтобы добавить объект сбора данных гистограммы на диаграмму, перетащите элемент **Данные гистограммы** с палитры **Статистика** на диаграмму агента Main.

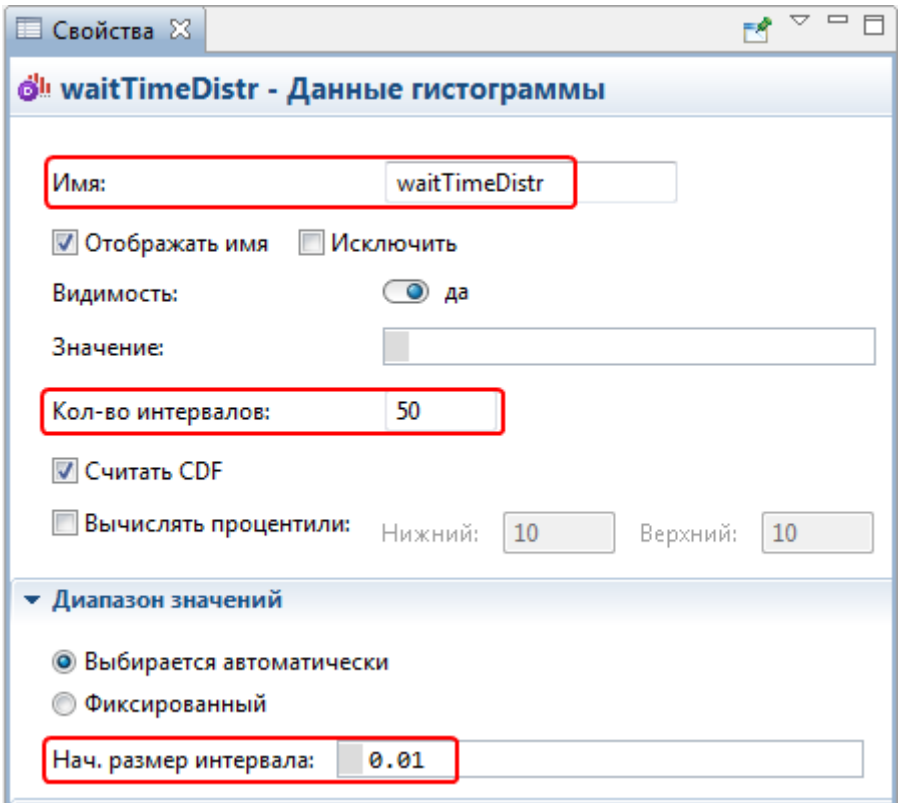


Задайте свойства элемента.

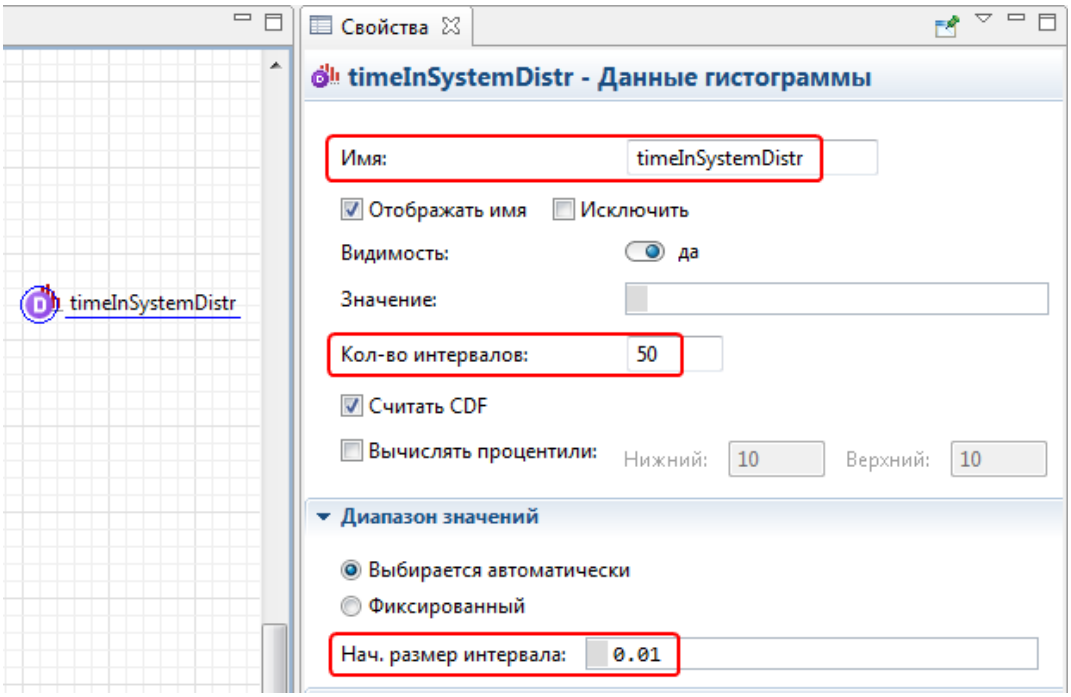
Измените **Имя** на *waitTimeDistr*.

Сделайте **Кол-во интервалов** равным *50*.

Задайте Начальный размер интервала: *0.01*.



Создайте еще один элемент сбора данных гистограммы. Ctrl+перетащите (Mac OS: Cmd+перетащите) только что созданный объект данных гистограммы, чтобы создать его копию. Измените **Имя** этого элемента на *timeInSystemDistr*.

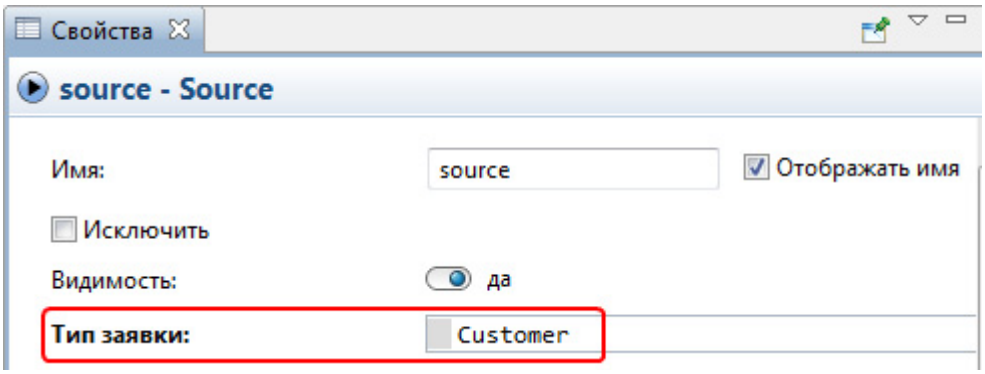


Теперь нам нужно изменить свойства блоков нашей диаграммы процесса.

Задание 5. Измените свойства блоков диаграммы процесса

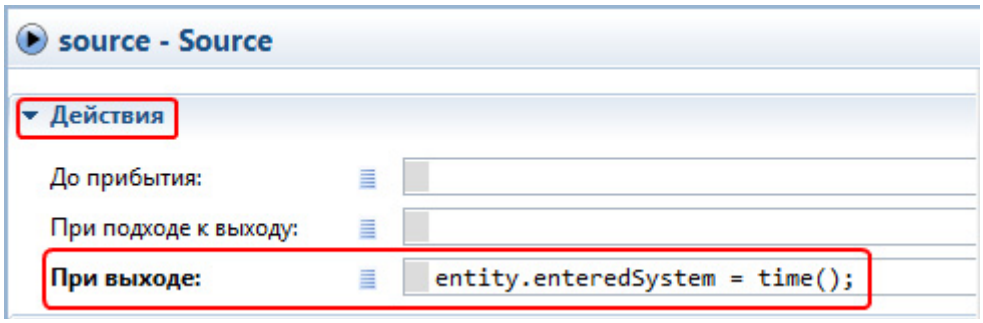
Измените свойства объекта *source*:

Введите *Customer* в поле **Тип заявки**. Это позволит напрямую обращаться к полям класса заявки Customer в коде динамических параметров этого объекта.



Убедитесь, что тип заявки Customer указан в поле **Новая заявка**. Этот объект должен продолжать создавать заявки типа Customer.

Введите entity.enteredSystem = time(); в поле действия **При выходе** в секции **Действия**. Этот код будет сохранять время создания заявки-клиента в переменной enteredSystem нашего типа заявки Customer. Функция time() возвращает текущее значение модельного времени.

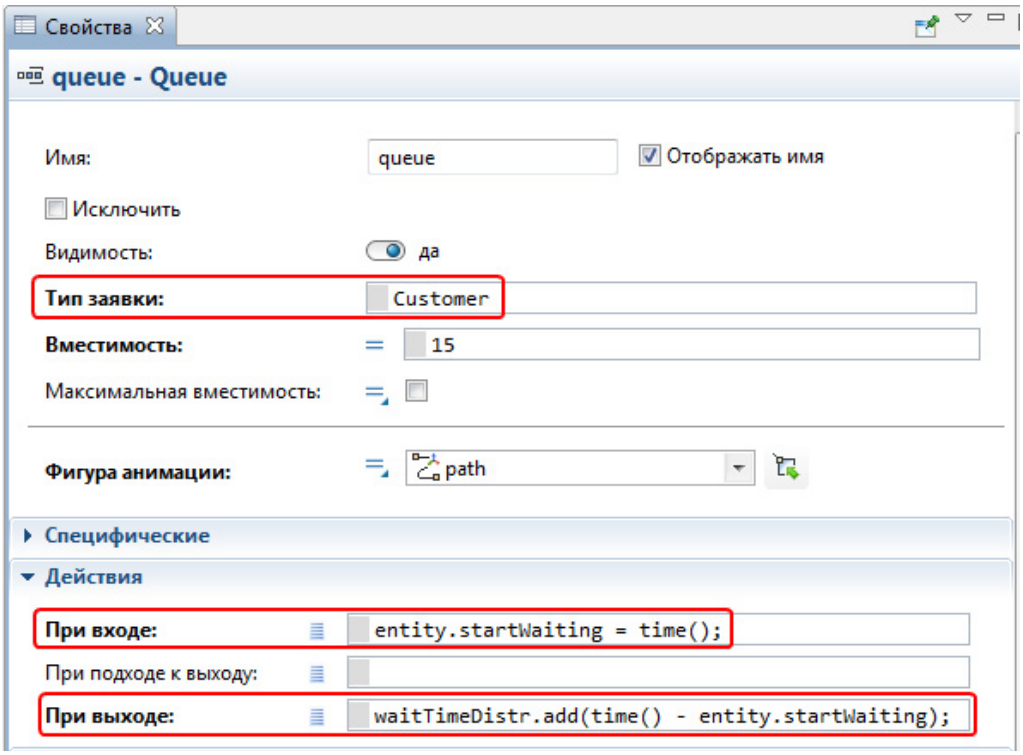


Измените свойства объекта *queue*:

Введите *Customer* в поле **Тип заявки**.

Введите entity.startWaiting = time(); в поле действия **При входе** в секции **Действия**. Этот код запоминает время начала ожидания клиентом его очереди на обслуживание в переменной startWaiting нашего типа заявки Customer.

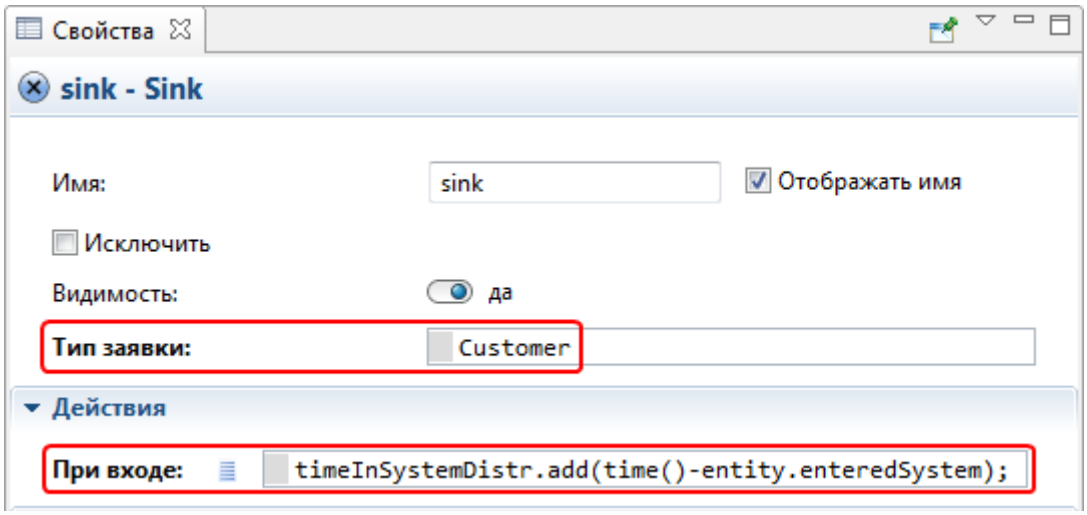
Введите waitTimeDistr.add(time() - entity.startWaiting); в поле действия **При выходе**. Этот код добавляет время, в течение которого клиент ожидал обслуживания, в объект сбора данных *waitTimeDistr*.



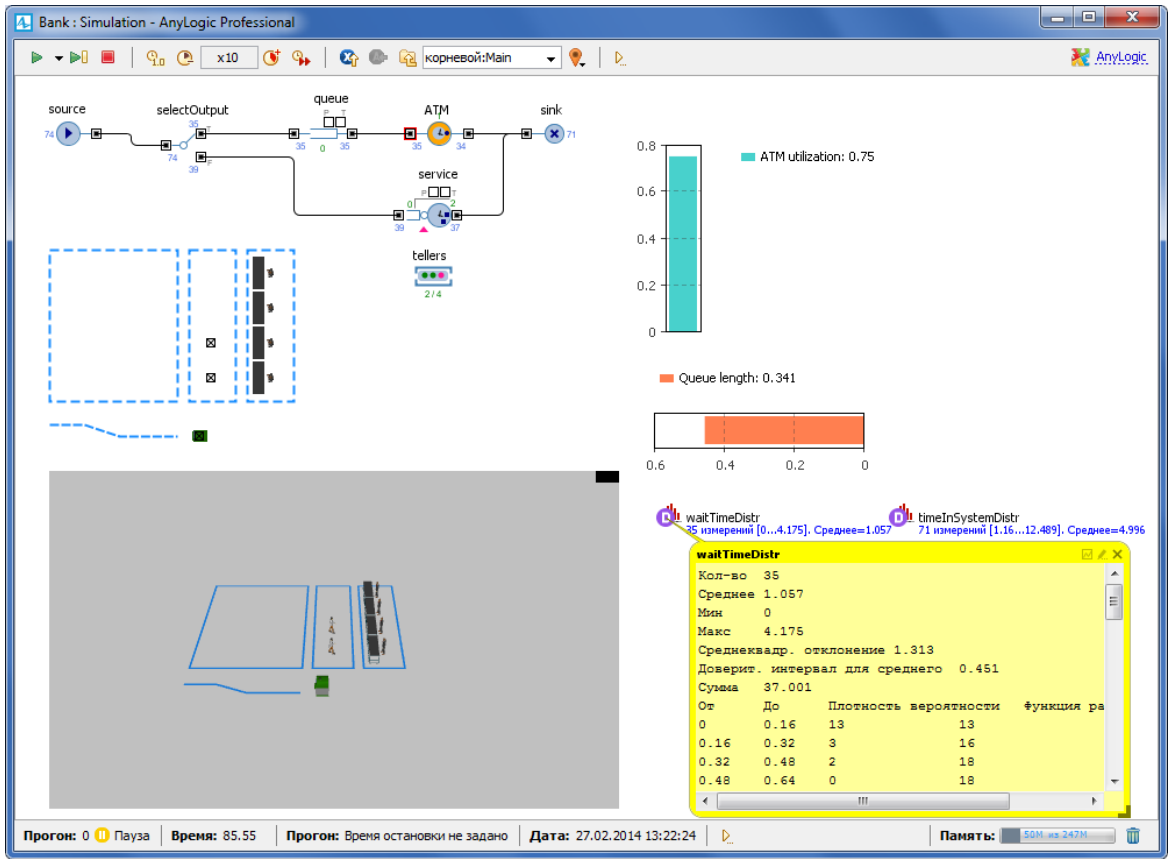
Измените свойства объекта *sink*:

Введите *Customer* в поле **Тип заявки**.

Введите timeInSystemDistr.add(time()-entity.enteredSystem); в поле действия **При входе** в секции **Действия**. Этот код добавляет полное время пребывания клиента в банковском отделении в объект сбора данных гистограммы *timeInSystemDistr*.



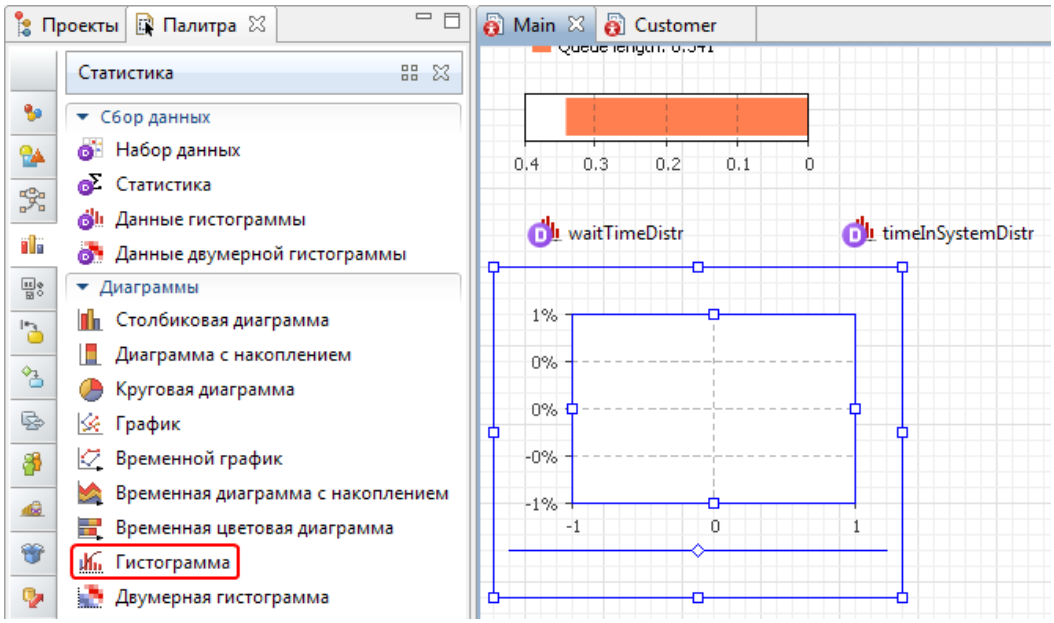
Запустите модель и просмотрите статистику с помощью окон инспекта. Открыть окно инспекта можно щелкнув мышью по значку объекта сбора данных. Здесь Вы увидите стандартные для статистического анализа данные, приведенные для значений, собранных в данном объекте сбора статистики.



Теперь давайте добавим на диаграмму нашего класса гистограммы, которые будут отображать собранную нами временную статистику.

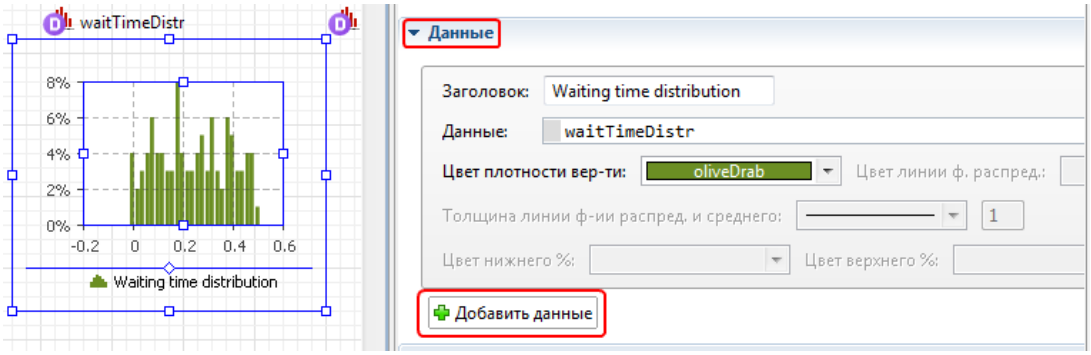
Задание 6. Добавьте две гистограммы для отображения распределений времен ожидания клиента и пребывания клиента в системе

Чтобы добавить гистограмму на диаграмму агента, перетащите элемент **Гистограмма** из палитры **Статистика** в то место графического редактора, куда Вы хотите ее поместить. Измените ее размер при необходимости.



Укажите, какой элемент сбора данных хранит данные, которые Вы хотите отображать на гистограмме: в секции **Данные** свойств гистограммы щелкните мышью по кнопке **Добавить данные** и измените **Заголовок** отображаемых данных на *Waiting time distribution*.

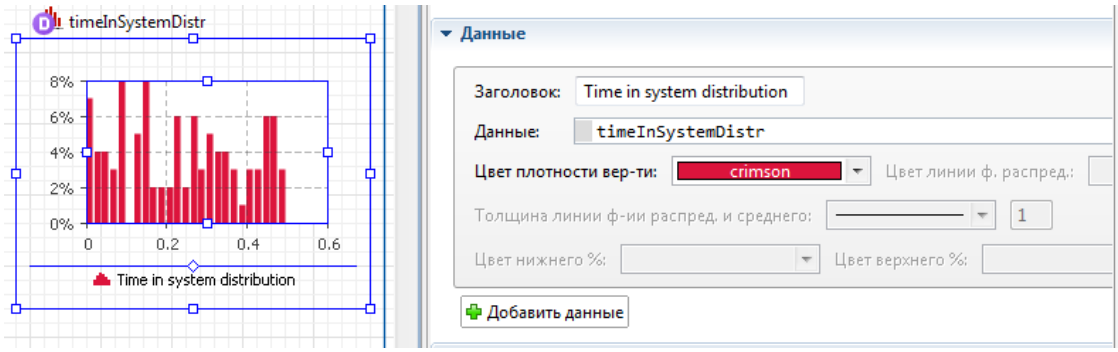
Введите в поле **Данные** имя соответствующего элемента: waitTimeDistr



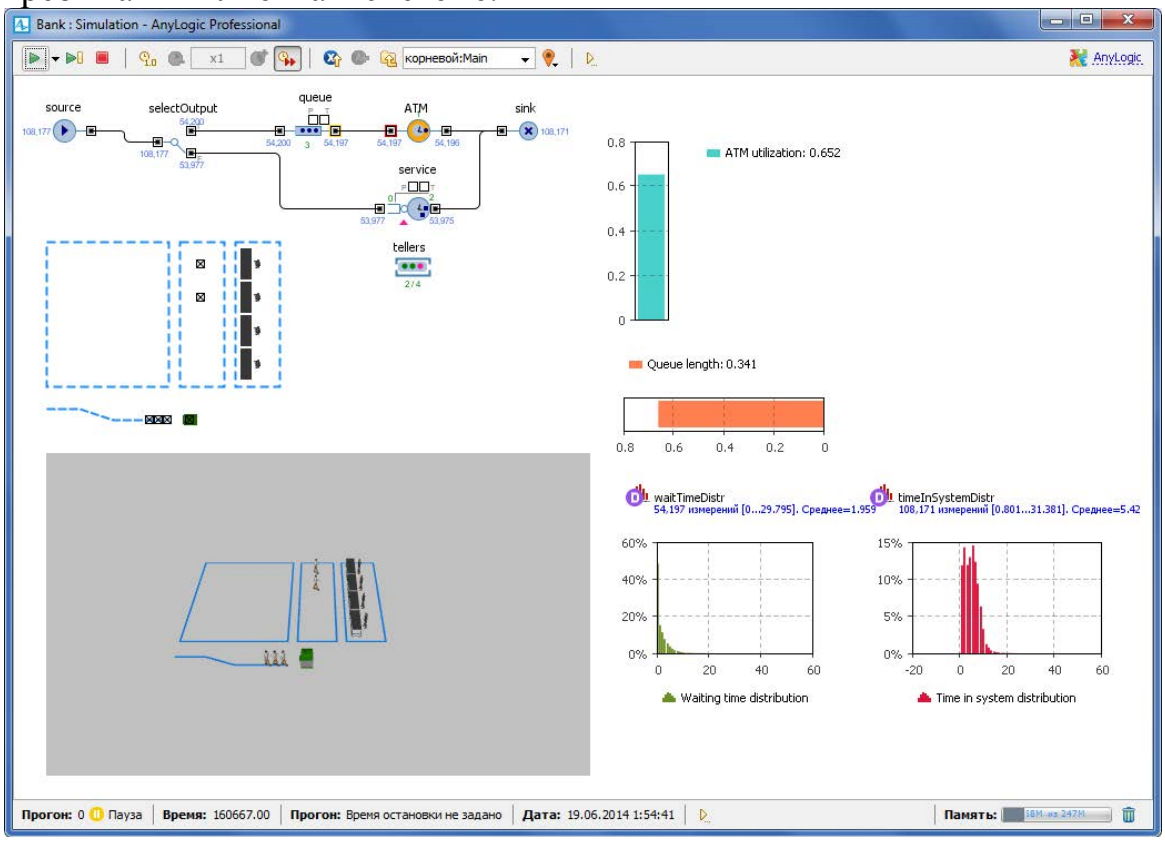
Добавьте еще одну гистограмму и расположите ее под ранее добавленной.

Измените **Заголовок** отображаемых данных на *Time in system distribution*.

В поле **Данные** введите имя элемента, хранящего данные, которые будут отображаться на гистограмме: timeInSystemDistr.



Запустите модель. Включите режим виртуального времени и понаблюдайте за тем, какой вид примет распределение времен ожидания и пребывания клиента в системе.



4.6 Проведение эксперимента.

Самостоятельно, изменяя параметры модели, сделайте выводы и оформите отчет о лабораторной работе.

Лабораторная работа 2. Моделирование динамических систем

1. Цель работы

Получить практические навыки по моделированию динамических систем на примере

2. Краткие теоретические сведения

Моделирование динамических систем по сути является прародителем системно-динамического подхода моделирования. Моделирование с помощью данного подхода используется в мехатронике, электрической, химической и других инженерных областях в качестве стандартного этапа процесса разработки. С математической точки зрения динамическая система представляет собой набор переменных состояния и алгебраических дифференциальных уравнений различного вида, заданных для этих переменных и описывающих их изменение с течением времени. В отличие от системной динамики, переменные здесь несут некоторый "физический" смысл: координаты местоположения, скорость, ускорение, сила, концентрация и.т.д., они, как это следует из их смысла, непрерывны и не являются агрегированными величинами, отражающими, например, общее количество или среднее значение нескольких сущностей.

Так же как и в случае с системной динамикой, дискретно-событийным (процессным) и агентным моделированием, AnyLogic предоставляет удобные инструменты и для тех, кто моделирует динамические системы:

AnyLogic предоставляет специальные элементы - динамические переменные для задания дифференциальных и алгебраических уравнений:

- **Накопитель** - для дифференциальных уравнений.
- **Динамическая переменная** - для формул.

AnyLogic поддерживает несколько численных методов для решения дифференциальных, алгебраических и смешанных систем уравнений. Численный метод автоматически выбирается исполняющим модулем AnyLogic в соответствии с поведением моделируемой системы. При решении дифференциальных уравнений первого порядка вначале используется метод Рунге-Кутта с фиксированным шагом. В случае систем алгебраических и смешанных уравнений AnyLogic использует метод Ньютона, варьирующий шаг интеграции для достижения необходимой точности.

AnyLogic поддерживает моделирование физических систем с помощью *Метода конечных элементов*. В этом случае обычно используются массивы и размерности типа диапазон и Вы можете ссылаться в уравнениях на следующий и предыдущий элемент . Вы можете изучить реализацию этого подхода на примере модели *"Vibrating String"*.

Более того, AnyLogic является единственным инструментом, язык моделирования которого не задает исключительно непрерывное или дискретное поведения, а может задавать модели с гибридным поведением. Вы можете сделать Вашу модель гибридной, добавив дискретные события, которые будут влиять на непрерывное поведение моделируемой системы, например, Вы можете добавить событие, отслеживающее значение непрерывно изменяющихся переменных и выполняющее некоторые действия при достижении значением переменной какого-то определенного порога; или событие, которое изменяет параметр уравнения и тем самым влияет на моделируемую динамическую систему.

Вы можете вынести переменные на интерфейс агента и связать их с интерфейсными переменными других активных объектов. Связанные переменные будут всегда иметь одинаковые значения, тем самым обеспечивая непрерывное взаимодействие объектов. Этот механизм позволяет Вам создавать объекты, аналогичные блокам, обычно используемым в блочных диаграммах - стандартном графическом языке моделирования, используемом инженерами.

Эта лабораторная работа кратко ознакомит Вас с процессом создания имитационной модели в AnyLogic. простого и наглядного пример — модель распространения нового продукта по Бассу. Вначале мы создадим классическую модель распространения инноваций Басса. Затем мы расширим нашу модель, добавив некоторые детали и продемонстрировав усовершенствованные возможности AnyLogic.

Модель Басса описывает процесс распространения продукта. Изначально продукт никому не известен, и для того, чтобы люди начали его приобретать, он рекламируется. В итоге определенная доля людей приобретает продукт под воздействием рекламы. Также люди приобретают продукт в результате общения с теми, кто этот продукт уже приобрел. Процесс приобретения нового продукта под влиянием убеждения его владельцев чем-то похож на распространение эпидемии.

3. Задание на лабораторную работу

3.1 Выполняя последовательно шаги проектирования, создать модель распространения продукта по Бассу.

3.2 Провести вычислительный эксперимент и сделать выводы.

4. Порядок выполнения работы.

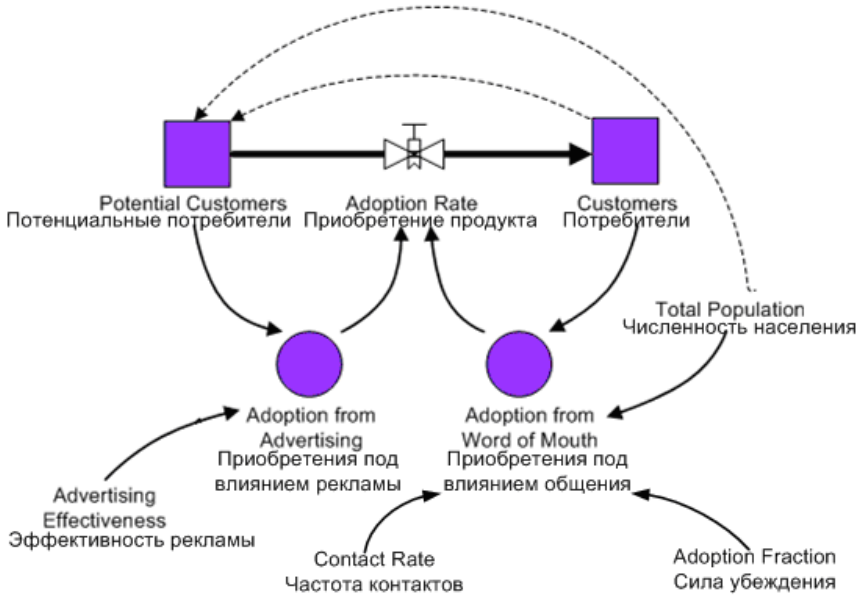
4.1 Анализ модели

Вначале мы должны проанализировать нашу модель, чтобы решить, как ее можно описать в терминах системной динамики. Мы должны определить ключевые переменные модели и то, как они влияют друг на друга, а затем создать потоковую диаграмму модели. При создании потоковой диаграммы мы должны учесть, какие переменные должны быть представлены накопителями, какие потоками, а какие – динамическими переменными.

Накопители (также называемые уровнями или фондами) представляют собой такие объекты реального мира, в которых сосредотачиваются некоторые ресурсы; их значения изменяются непрерывно. *Потоки* – это активные компоненты системы, они изменяют значения накопителей. В свою очередь, накопители системы определяют значения потоков. *Динамические переменные* помогают преобразовывать одни числовые значения в другие; они могут произвольно изменять свои значения или быть константами.

При создании потоковой диаграммы выявите переменные, которые накапливают значения с течением времени. В нашей модели численности потребителей и потенциальных потребителей продукта являются накопителями, а процесс приобретения продукта – потоком.

Системно-динамическое представление нашей модели показано на рисунке ниже. Накопители обозначаются прямоугольниками, поток—вентилем, а динамические переменные—кружками. Стрелки обозначают причинно-следственные зависимости в модели.



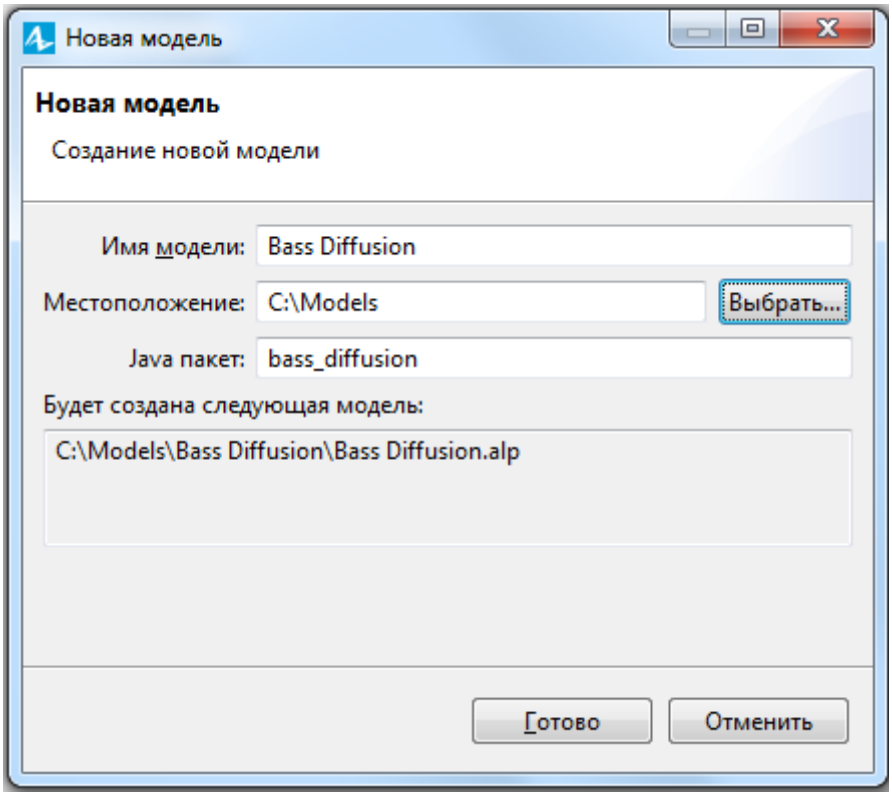
4.2 Создание новой модели

Вначале мы создадим новую модель.

Задание 1. Создайте новую модель

Щелкните мышью по кнопке панели инструментов **Создать** . Появится диалоговое окно **Новая модель**.

Задайте имя новой модели. В поле **Имя модели** введите Bass Diffusion.



Выберите каталог, в котором будут сохранены файлы модели. Если Вы хотите сменить предложенный по умолчанию каталог на какой-то другой, Вы можете ввести путь к нему в поле **Местоположение** или выбрать этот каталог с помощью диалога навигации по файловой системе, открывающегося по нажатию на кнопку **Выбрать**.

Щелкните мышью по кнопке **Готово**.

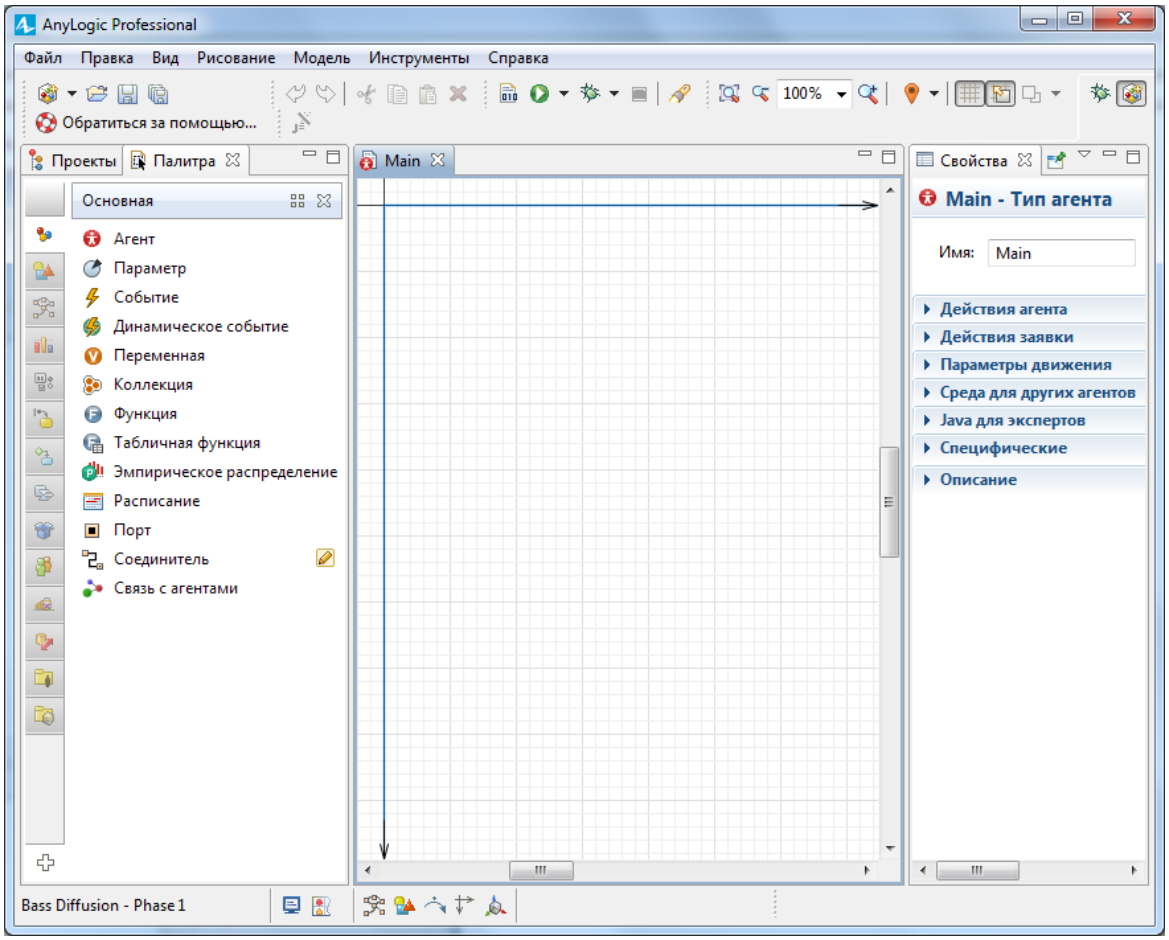
Вы создали новую модель. Если Вы еще не знакомы с пользовательским интерфейсом AnyLogic, то давайте уделим пару минут основным его компонентам:

В левой части рабочей области находятся панель **Проекты** и панель **Палитра**. Панель **Проекты** обеспечивает легкую навигацию по элементам моделей, открытым в текущий момент времени. Поскольку модель организована иерархически, то она отображается в виде дерева: сама модель образует верхний уровень дерева; эксперименты, типы агентов и Java классы образуют следующий уровень; элементы, входя-

щие в состав агентов, вложены в соответствующую подветвь типа агента и т.д. Панель **Палитра** содержит разделенные по категориям элементы, которые могут быть добавлены на графическую диаграмму типа агентов или эксперимента.

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

В центре рабочей области AnyLogic Вы увидите графический редактор диаграммы типа агентов *Main*.



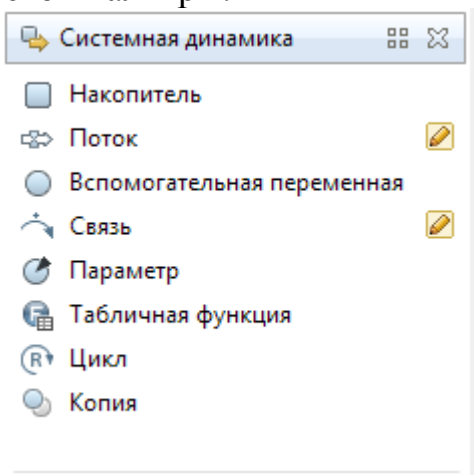
При работе с моделью не забывайте сохранять производимые Вами изменения с помощью кнопки панели инструментов **Сохранить** .

4.3 Создание накопителей

Давайте начнем создание диаграммы накопителей и потоков. И начнем мы с создания накопителей. В нашей модели их два - они моделируют численности потребителей и потенциальных потребителей продукта. Накопитель в AnyLogic задается с помощью одноименной переменной.

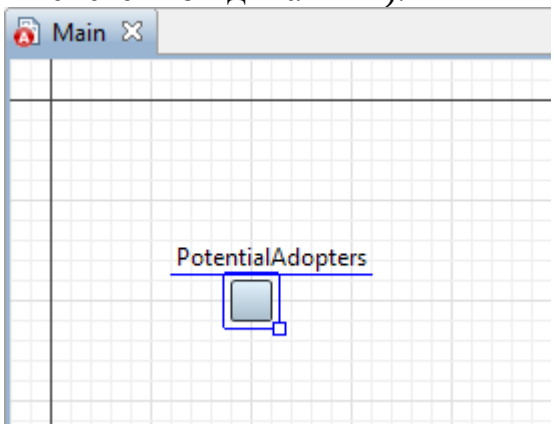
Задание 1. Создайте накопитель для моделирования численности потенциальных потребителей

Вначале откройте закладку **Системная динамика** панели **Палитра**. Чтобы открыть какую-либо закладку панели **Палитра** (именуемую в дальнейшем палитрой), нужно щелкнуть мышью по заголовку этой палитры.



Перетащите элемент **Накопитель** из палитры **Системная динамика** на диаграмму типа агентов.

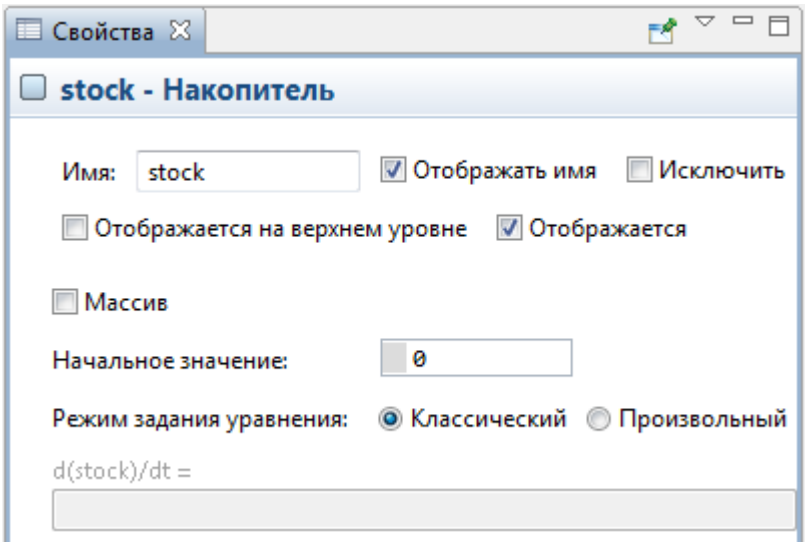
На диаграмме появится маленький голубой прямоугольник, обозначающий переменную-накопитель (что соответствует классической нотации системной динамики).



Когда Вы добавите элемент на диаграмму, он будет выделен, и его свойства будут отображены в панели **Свойства**. Здесь Вы можете изменить свойства элемента в соответствии с Вашими требованиями. Обратите внимание, что панель **Свойства** является контекстно-зависимой - она отображает свойства выделенного в текущий момент элемента. Поэтому позднее для изменения свойств элемента нужно будет предварительно щелчком мыши выделить его в графическом редакторе или в панели **Проекты**.

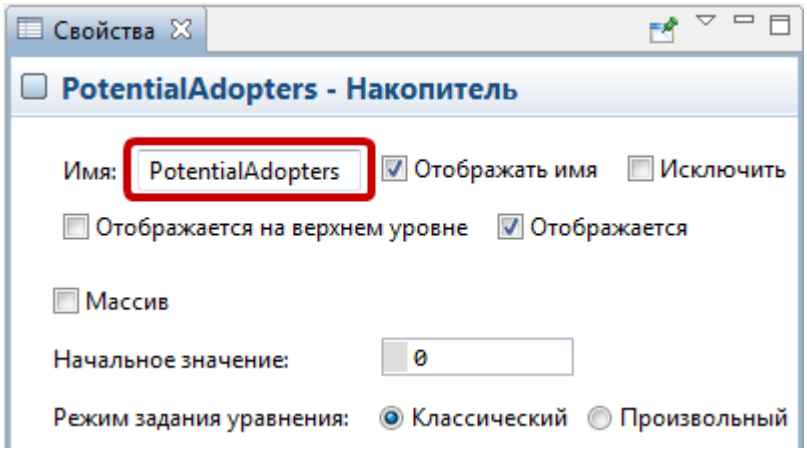
Чтобы у Вас всегда была уверенность в том, что в текущий момент в рабочем пространстве выбран именно нужный Вам элемент, и именно его свойства Вы редактируете в панели **Свойства**, обращайте

внимание на первую строку, показываемую в панели **Свойства** - в ней отображается имя выбранного в текущий момент времени элемента и его тип - в приведенном на рисунке примере это *stock* и *Накопитель* соответственно.

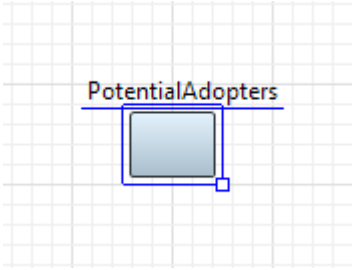


Панель **Свойства**. На данном рисунке в панели отображаются свойства выделенного накопителя.

В поле **Имя** введите новое имя накопителя: PotentialAdopters.



Немного увеличьте размер значка накопителя. Для этого выделите его щелчком мыши в графическом редакторе и перетащите в сторону появившийся в нижнем правом углу значка маркер.



Задание 2. Создайте накопитель для моделирования численности потребителей

Аналогично создайте еще один накопитель.

Чтобы создать накопитель такого же размера, проще всего склонировать существующий накопитель, перетащив его мышью с нажатой клавишей Ctrl (при этом свойства нового элемента будут теми же, что и у склонированного, но в данном случае это не важно, поскольку мы изменили только одно свойство накопителя - его имя).

Поместите новый накопитель справа от накопителя PotentialAdopters, как показано на приведенном ниже рисунке.

Назовите его Adopters.

На данный момент задание накопителей еще полностью не закончено. Позднее мы зададим начальные значения накопителей, а также интегральные формулы, согласно которым будут вычисляться их значения. Но вначале нам нужно создать поток приобретения продукта.

4.4 Добавление потока продаж продукта

Мы уже создали в нашей модели два накопителя, моделирующие численности потенциальных потребителей и потребителей продукта. Теперь пришло время задать потоки. В нашей простейшей модели есть только один поток - поток продаж продукта, увеличивающий число потребителей продукта и уменьшающий численность потенциальных потребителей.

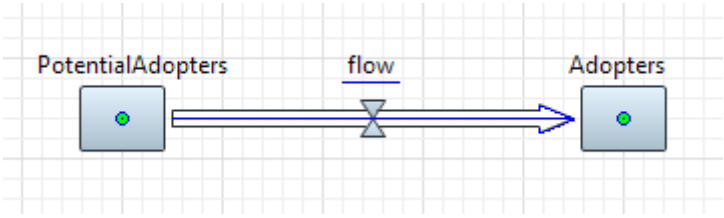
В AnyLogic поток задается переменной *поток*. Значение потока вычисляется в соответствии с заданной формулой.

Задание 1. Добавьте поток, ведущий из накопителя **PotentialAdopters** в накопитель **Adopters**

Сделайте двойной щелчок мышью по накопителю, из которого поток вытекает (PotentialAdopters), а затем щелкните по тому накопителю, в который он втекает (Adopters).

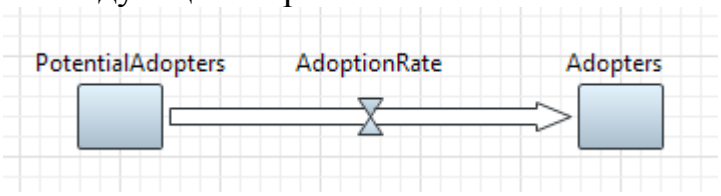
AnyLogic создаст новый поток и сделает его исходящим потоком для накопителя PotentialAdopters и входящим - для Adopters. Поток отображается в виде стрелки со значком вентиля посередине. Стрелка пока-

зывает направление потока - в данном случае поток будет уменьшать значение накопителя PotentialAdopters и увеличивать значение Adopters.

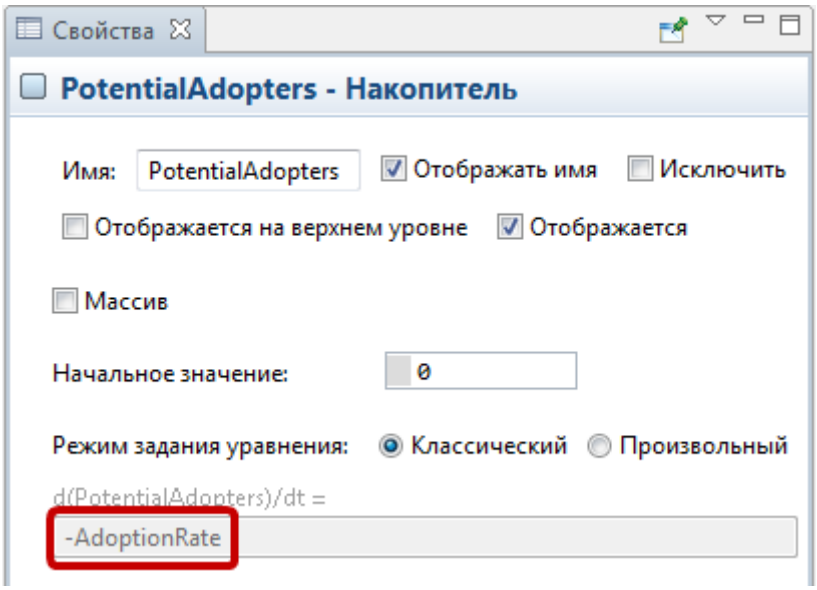


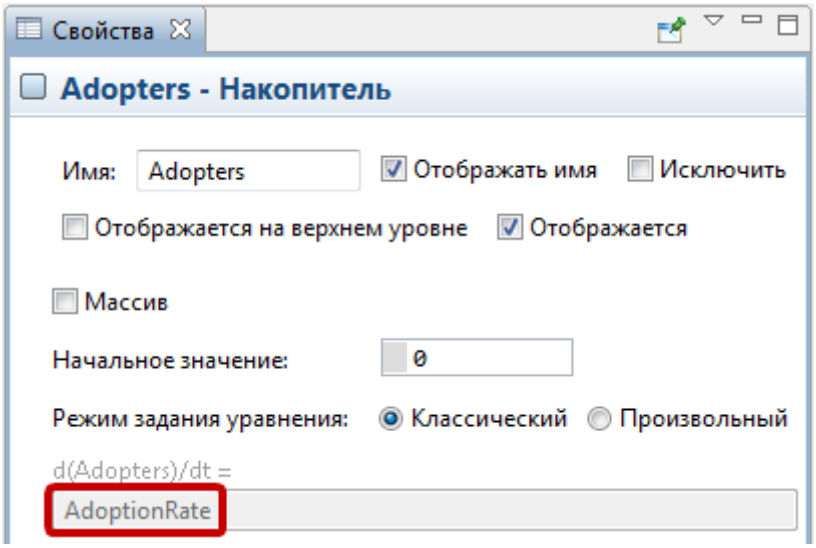
Выделите стрелку созданного потока в графическом редакторе и измените имя потока на AdoptionRate.

В результате диаграмма потоков и накопителей должна будет выглядеть следующим образом:



Можете теперь взглянуть на свойства накопителей. Формулы накопителей должны выглядеть следующим образом:





Эти формулы были автоматически заданы при добавлении потока. Значения входящих потоков, то есть потоков, которые увеличивают значение накопителя, прибавляются, а значения исходящих потоков, уменьшающих значение накопителя, вычитаются из текущего значения накопителя.

Формулу, согласно которой будет вычисляться значение потока, мы зададим чуть позднее.

4.5 Задание констант

Теперь мы зададим константы нашей модели с помощью параметров.

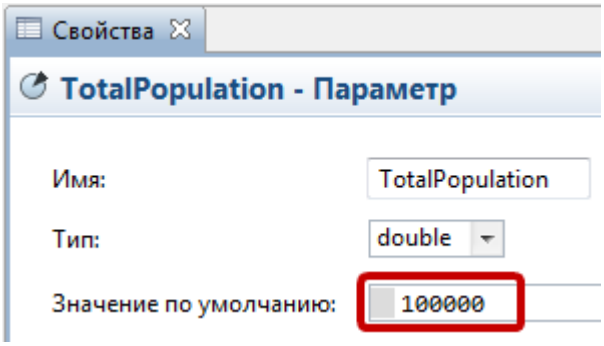
Задание 1. Создайте константу, задающую общую численность населения

Перетащите элемент **Параметр** из палитры **Системная динамика** на диаграмму типа агентов.

На странице **Основные** панели **Свойства** задайте свойства этого параметра.

Измените имя параметра. Введите TotalPopulation в поле **Имя**.

В поле **Значение по умолчанию** введите 100000. Пусть общая численность населения в нашей модели будет именно такой.



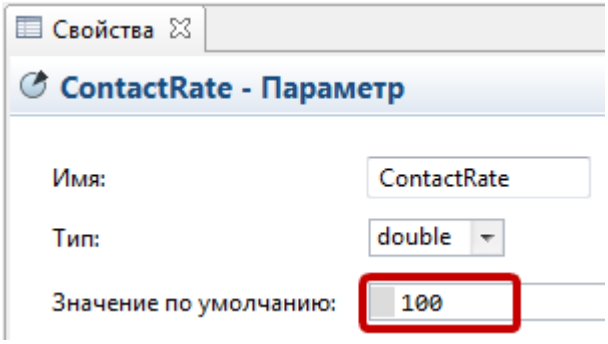
Вы можете задать краткое описание параметра на странице свойств **Описание**. Введите текст, который поможет объяснить смысл константы тем, кто в дальнейшем будет работать с этой моделью.

Частота, с которой потенциальные потребители общаются с потребителями, в нашей модели будет постоянной величиной. Поэтому мы зададим частоту контактов константой.

Задание 2. Создайте константу ContactRate, задающую общую численность населения

Аналогично создайте еще одну константу. Задайте **Имя** ContactRate.

Предположим, что каждый человек в среднем встречается со 100 людьми в год. Введите в поле **Значение по умолчанию** 100.

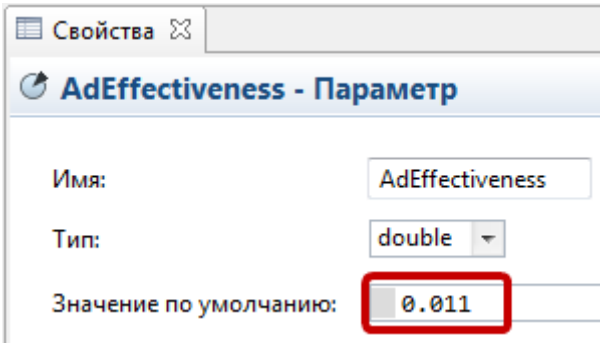


В этой модели интенсивность рекламы и вероятность того, что продукт будет приобретен под ее влиянием, полагаются постоянными. Поэтому мы зададим эффективность рекламы константой. Эффективность рекламы определяет, какая доля людей купит продукт вследствие ее влияния.

Задание 3. Создайте константу AdEffectiveness, задающую эффективность рекламы

Создайте еще одну константу и назовите ее AdEffectiveness.

Задайте значение по умолчанию 0.011.

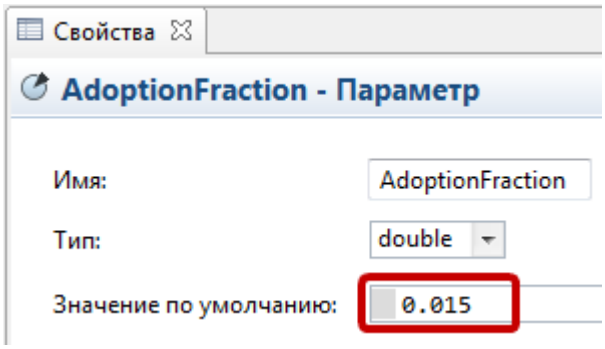


Задайте константой силу убеждения владельцев продукта, определяющую ту долю контактов, которая приводит к продажам продукта.

Задание 4. Создайте константу **AdoptionFraction**

Создайте еще одну константу и назовите ее AdoptionFraction.

Задайте значение по умолчанию 0.015.



4.6 Задание начальных значений накопителей

Теперь мы можем задать начальные значения накопителей.

Мы хотим задать общую численность людей в нашей модели (заданную параметром TotalPopulation) в качестве начального значения накопителя PotentialAdopters.

Когда Вы указываете какой-либо элемент в выражении начального значения накопителя, Вы должны вначале соединить этот элемент с накопителем с помощью связи. *Связь* позволяет явно задавать существующие зависимости между элементами диаграммы потоков и накопителей.

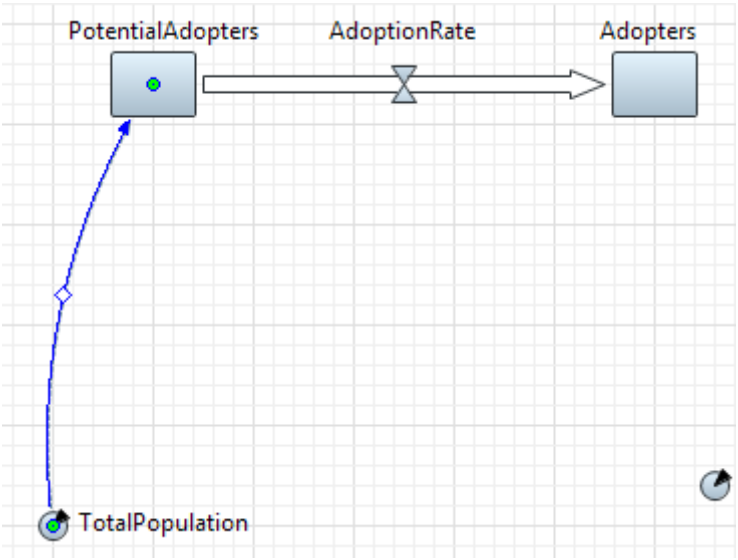
Задание 1. Чтобы задать связь

Сделайте двойной щелчок по элементу **Связь** палитры **Системная динамика**. Значок элемента при этом должен измениться на следующий: .

Сразу после этого, щелкните в графическом редакторе по элементу, который упоминается в выражении начального значения (TotalPopulation).

Затем щелкните по накопителю PotentialAdopters, к которому должна следовать создаваемая связь зависимости.

Обратите внимание, что нужно всегда рисовать связи именно в таком направлении - от независимой переменной к зависимой.



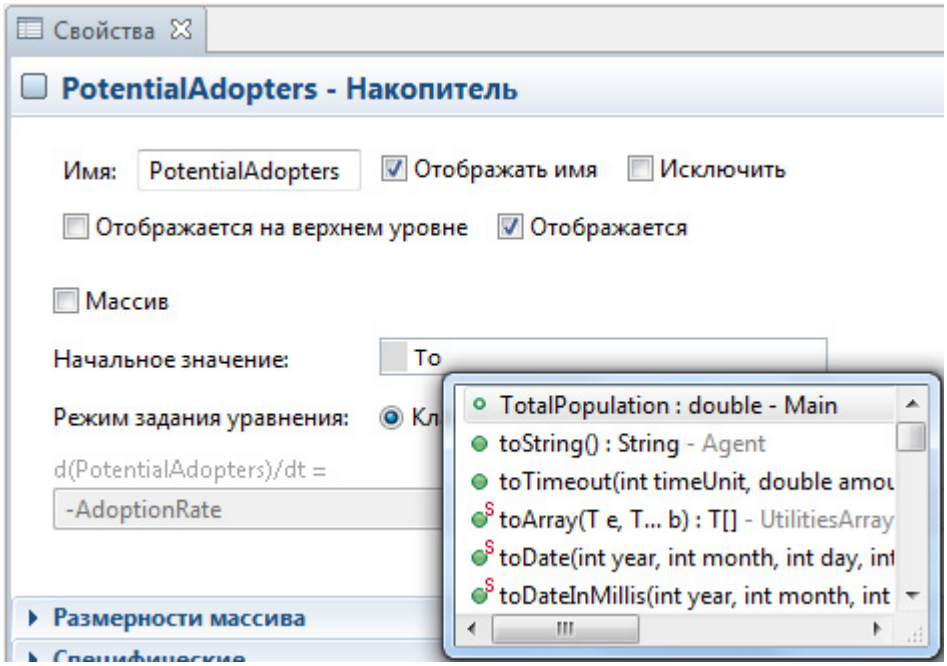
Теперь, когда мы создали связь, можно задать начальное значение накопителя, сославшись в нем на параметр TotalPopulation.

Задание 2. Задайте начальное количество потенциальных потребителей продукта

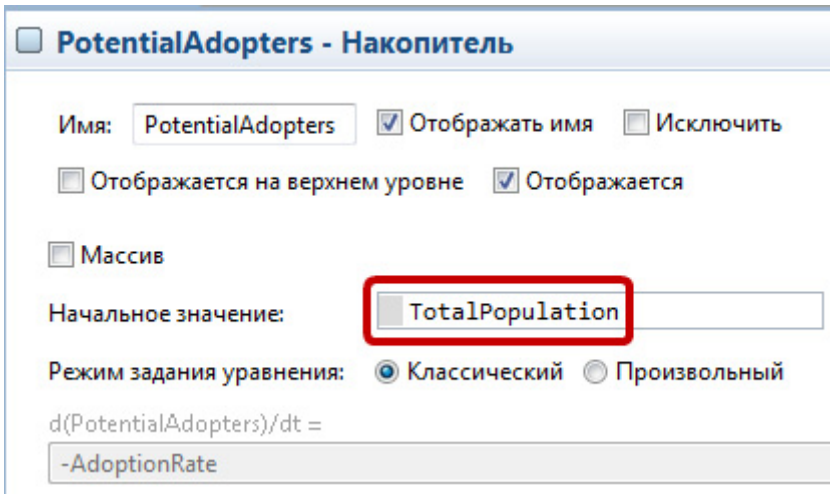
В графическом редакторе или в панели **Проекты** выделите накопитель PotentialAdopters щелчком мыши.

На странице **Основные** панели **Свойства** введите TotalPopulation в поле **Начальное значение**. Чтобы не печатать полностью имена функций и переменных в формулах, можете воспользоваться **Мастером подстановки кода**. Чтобы открыть **Мастер**, щелкните мышью в том месте поля (в нашем случае - поля **Начальное значение**, куда Вы хотите поместить имя, а затем нажмите Ctrl+пробел (Mac OS: Alt+пробел).

Появится окно **Мастера подстановки кода**, перечисляющего переменные модели и функции, доступные в текущем конексте. Прокрутите список к имени, которое Вы хотите вставить, или введите первые буквы имени, пока оно не будет выделено в списке. Двойным щелчком мыши по имени добавьте его в поле формулы.



В результате в поле **Начальное значение** должно быть добавлено имя параметра TotalPopulation, значение которого и будет определять начальное значение этого накопителя (начальную численность потенциальных покупателей продукта).



Начальное значение накопителя Adopters, моделирующего потребителей продукта, задавать не нужно, поскольку изначально число потребителей равно нулю, а накопитель по умолчанию и так инициализируется нулем.

Теперь мы закончили задание накопителей. Нам осталось добавить на диаграмму потоков и накопителей динамические переменные - и модель будет готова.

4.7 Создание динамических переменных

Нам нужно создать две динамические переменные, которые будут соответствовать двум составляющим потока приобретения продукта:

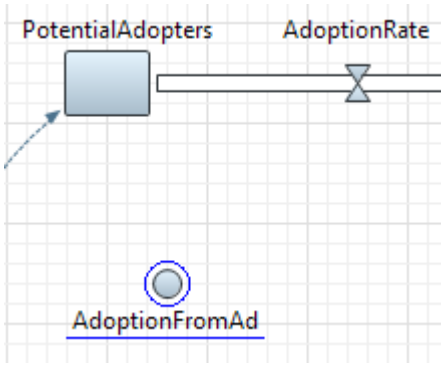
Приобретениям, совершенным под влиянием рекламы.

Приобретениям, совершенным под влиянием общения потребителей продукта с потенциальными потребителями.

Задание 1. Создайте динамическую переменную **AdoptionFromAd**

Перетащите элемент **Динамическая переменная** из палитры **Системная динамика** на диаграмму типа агентов.

На странице **Основные** панели **Свойства** введите новое **Имя** переменной: AdoptionFromAd.

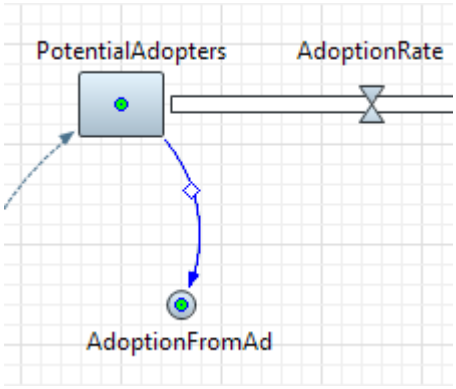


Теперь мы хотим задать формулу для этой динамической переменной. Влияние рекламы моделируется следующим образом: некий постоянный процент потенциальных клиентов AdEffectiveness всё время становятся клиентами. Их доля в AdoptionRate равна, соответственно, PotentialAdopters*AdEffectiveness.

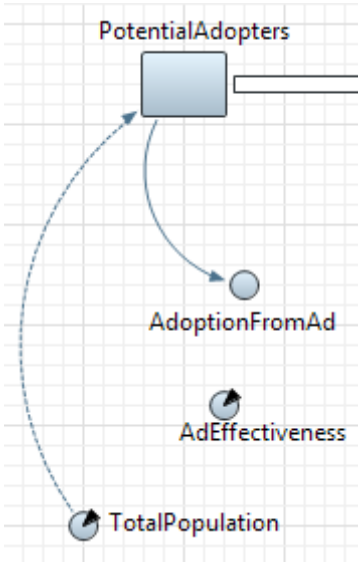
Опять, как и в случае составления выражения начального значения накопителя, когда какая-либо переменная задействована в формуле динамической переменной или потока, между этими переменными должна существовать связь.

Задание 2. Добавьте связи от двух переменных к зависимой от них **AdoptionFromAd**

Добавьте связи, ведущие от AdEffectiveness и PotentialAdopters к AdoptionFromAd.

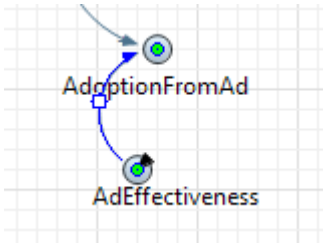


Вы можете изменить внешний вид связи (а именно - радиус ее выпуклости), перетащив мышью маркер, находящийся посередине связи (чтобы он стал виден, связь вначале надо выделить щелчком мыши). Пусть связь выглядит, как на приведенном ниже рисунке:



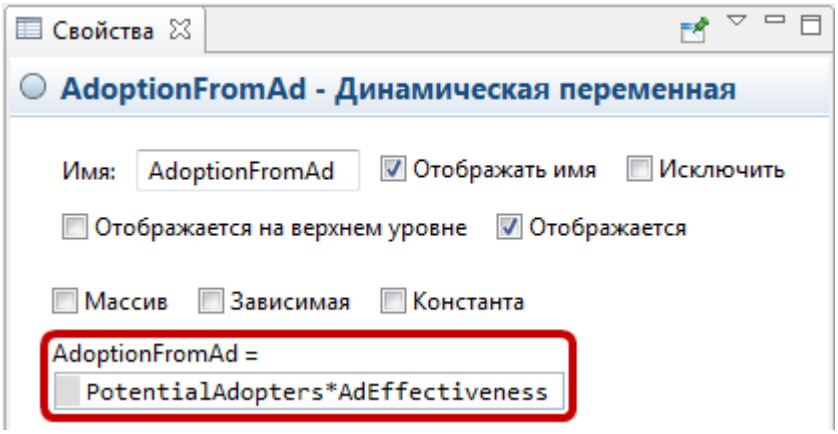
Вы могли заметить, что эта связь выглядит немного иначе, чем та, что ведет от TotalPopulation к PotentialAdopters. Связи с переменными, упоминающимися в начальных значениях накопителей, рисуются пунктирными линиями, в то время, как все остальные - сплошными.

Добавьте еще одну связь, ведущую от AdEffectiveness к AdoptionFromAd.



Задайте формулу, согласно которой будет вычисляться значение переменной. В свойствах переменной AdoptionFromAd, в по-

ле **AdoptionFromAd** = введите: AdEffectiveness*PotentialAdopters (Вы можете воспользоваться Мастером подстановки кода).



Для тех, кто не знаком с классической моделью Диффузии по Бассу, давайте попробуем самостоятельно составить формулу интенсивности продаж продукта под влиянием устного общения потребителей продукта с теми, кто данный продукт еще не приобрел.

Мы делаем предположение, что в нашей модели человек может общаться с любым другим человеком.

Количество контактов человека в единицу времени (а под единицей времени в нашей модели подразумевается год) задается параметром ContactRate. Запишем ContactRate в качестве первого сомножителя нашей формулы.

Количество людей, которые владеют продуктом, и могут убеждать остальных приобрести его, в нашей модели в каждый момент времени будет определяться значением накопителя Adopters, и поскольку каждый потребитель будет общаться в единицу времени с ContactRate людей, то количество контактов в единицу времени у всех потребителей продукта будет равно Adopters*ContactRate.

Теперь нужно учесть тот факт, что в результате общения не все те, кто еще не купил этот продукт, сразу побегут его покупать - если кого-то доводы своего знакомого, успешно пользующегося изучаемым нами продуктом, могут убедить, то кто-то может остаться к ним равнодушным, и своего решения не покупать продукт не изменить. Поэтому мы добавим в нашу формулу еще один сомножитель AdoptionFraction, задающий силу убеждения владельцев продукта, определяющую ту долю контактов, которая приводит к продажам продукта. Таким образом, наша формула приобретает вид Adopters*ContactRate*AdoptionFraction.

И наконец, нам нужно учесть, что на данный момент наша формула не учитывает того, что владельцы продукта будут общаться как с потенциальными потребителями, так и с теми, кто уже владеет продуктом. И общение с последними ни к каким новым продажам продукта не при-

ведет. Поэтому нам нужно учесть в нашей формуле и вероятность того, что тот, с кем общался потребитель, ещё не владеет интересующим нас продуктом. Эта вероятность задается так: PotentialAdopters/TotalPopulation.

В итоге наша формула будет выглядеть следующим образом: Adopters*ContactRate*AdoptionFraction*PotentialAdopters/TotalPopulation

Именно столько потенциальных потребителей будут приобретать продукт в единицу модельного времени под воздействием общения с владельцами этого продукта.

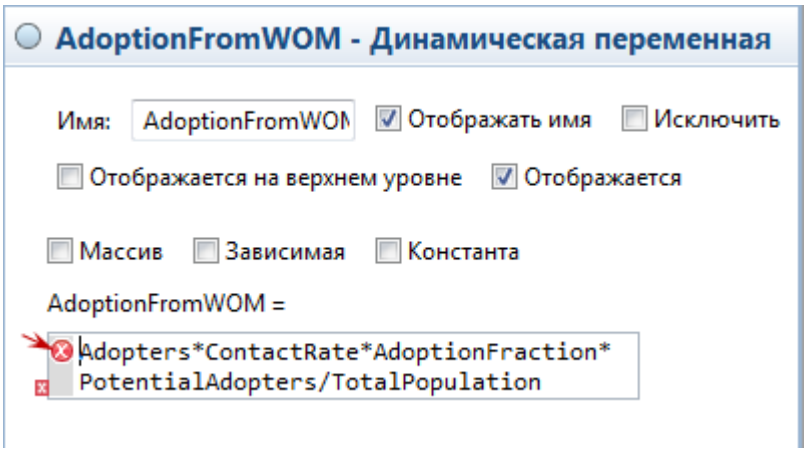
Задание 3. Создайте динамическую переменную **AdoptionFromWOM**

Аналогично создайте еще одну динамическую переменную.

Назовите ее AdoptionFromWOM.

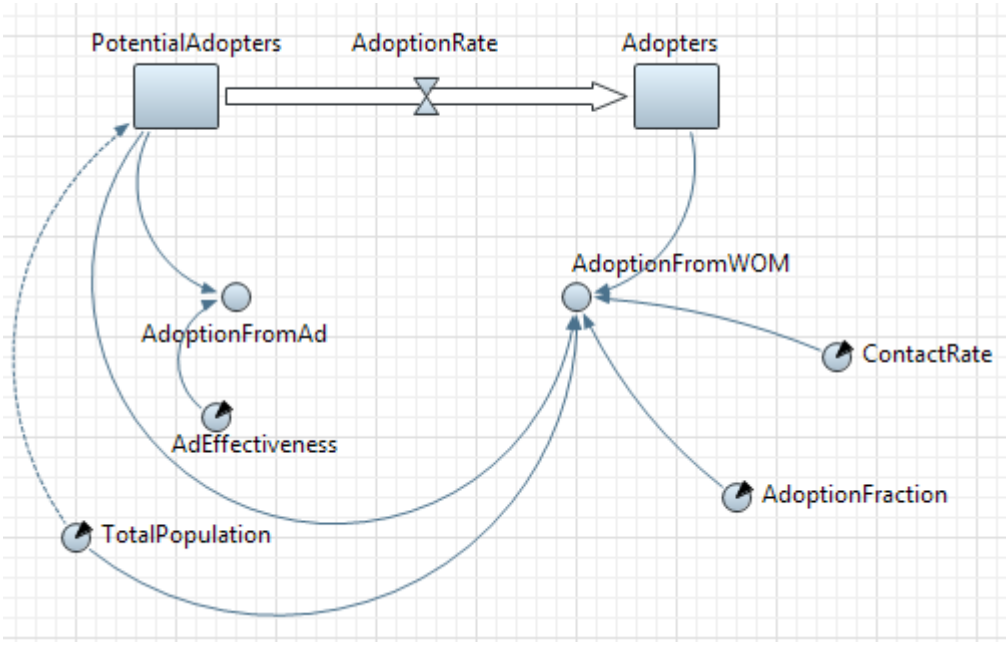
Задайте для этой переменной следующую формулу: ContactRate * AdoptionFraction * PotentialAdopters * Adopters / TotalPopulation

Вам может показаться утомительным рисовать связи для задействованных в формулах переменных и параметров. Чтобы облегчить этот процесс, AnyLogic предлагает пользователям механизм быстрого исправления ошибок, связанных с отсутствующими или избыточными связями. Когда Вы зададите указанную формулу, то щелкнув в этом поле Вы увидите индикатор ошибки:



Щелкните мышью по индикатору. Вы увидите контекстное меню, состоящее из пунктов **Добавить отсутствующую связь для: ...** . Поочередно щелкнув по всем этим пунктам Вы добавите на диаграмму все недостающие связи.

В итоге у Вас должна будет получиться диаграмма следующего вида:



Теперь мы можем задать формулу для потока приобретения продукта. Значение потока определяется суммой двух его независимых составляющих – продаж в результате рекламного влияния и продаж под влиянием общения с потребителями продукта.

Задание 4. Задайте формулу потока

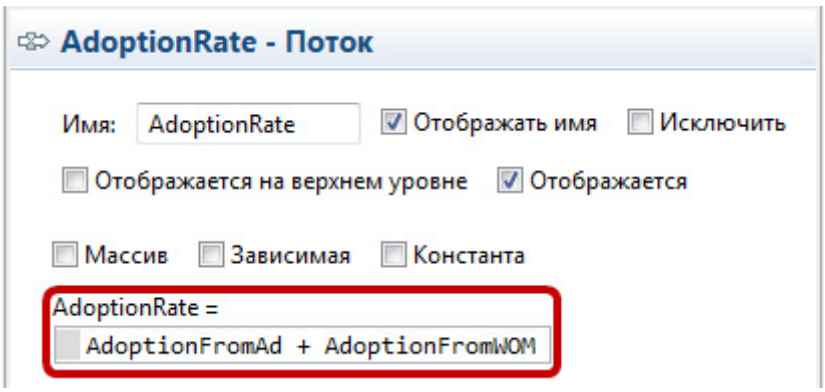
Выделите поток AdoptionRate щелчком мыши.

Перейдите на страницу **Основные** панели **Свойства**.

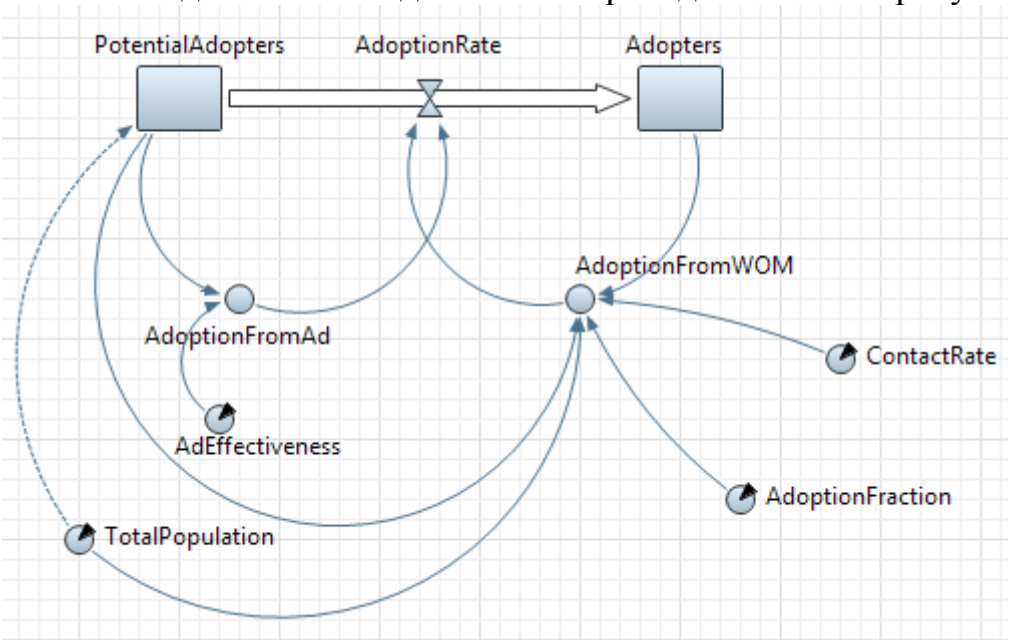
Введите правую часть формулы, по которой будет вычисляться значение потока, в поле **AdoptionRate=** :

AdoptionFromAd + AdoptionFromWOM

Добавьте соответствующие связи от этих переменных к потоку AdoptionRate.



Теперь мы закончили создание нашей модели. Диаграмма накопителей и потоков должна выглядеть как на приведенном ниже рисунке:



Связи имеют полярность, положительную или отрицательную:

Положительная связь означает, что два элемента системной динамики изменяют свои значения в одном направлении, т.е. если значение элемента, из которого направлена связь, уменьшается, значение другого элемента уменьшается тоже. Аналогично, если увеличивается значение одного элемента, то и значение зависимого от него элемента увеличивается тоже.

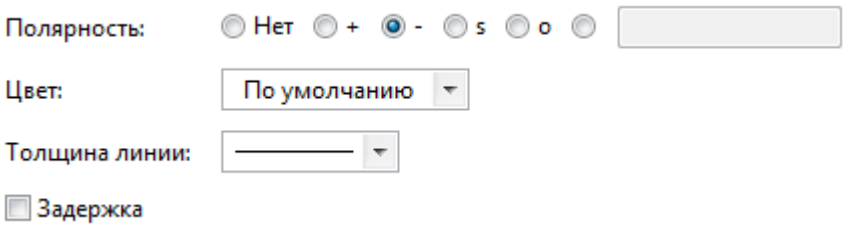
Отрицательная связь означает, что два элемента системной динамики изменяют свои значения в противоположных направлениях, т.е. если значение элемента, из которого направлена связь, уменьшается, то значение другого элемента увеличивается, и наоборот.

Вы можете добавить рядом со связями метки, которые будут обозначать полярность этих связей. Обычно полярность обозначается с помощью символов +/- рядом со стрелкой связи. Таким образом Вы можете показать, как зависимая переменная изменяет свое значение при изменении значения независимой переменной.

Задание 5. Проставьте полярности у связей

Чтобы отобразить у связи значок полярности, выделите связь и выберите нужный символ (+ или -) из группы кнопок **Полярность** на странице свойств связи:

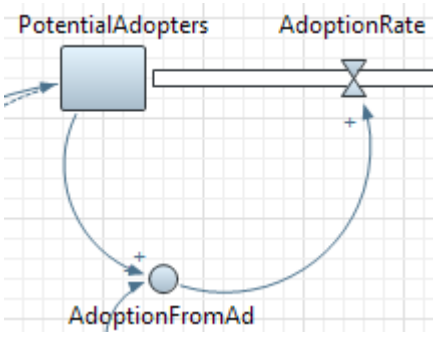
Здесь же при желании можно изменить и цвет линии связи, а также ее толщину.



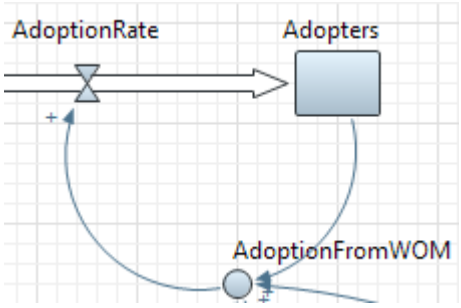
В нашей модели все связи, за исключеинем той, что ведет от TotalPopulation к AdoptionFromWOM, имеют положительную полярность.

Можно увидеть, что наша модель содержит два цикла с обратной связью: один компенсирующий и один усиливающий.

Компенсирующий цикл с обратной связью воздействует на поток приобретения продукта, вызванный рекламой. Поток приобретения продукта сокращает число потенциальных потребителей, что в свою очередь приводит к снижению интенсивности приобретения продукта.

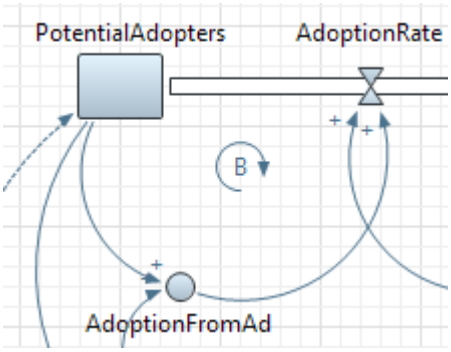


Усиливающий цикл с обратной связью воздействует на поток приобретения продукта, вызванный общением с потребителями продукта. Поток приобретения продукта увеличивает численность потребителей продукта, что приводит к росту интенсивности приобретения продукта под влиянием общения с потребителями продукта, и следовательно к росту интенсивности приобретения продукта.



Задание 6. Добавьте идентификатор цикла, вызывающего насыщение рынка

Перетащите элемент **Цикл** ® из палитры **Системная динамика** на графическую диаграмму, как показано на рисунке ниже:



Перейдите на страницу **Основные** панели **Свойства**, чтобы изменить свойства цикла.

Задайте **Направление** цикла - этот цикл направлен **Против часовой стрелки**.

В поле **Текст** введите краткое описание этого цикла, объясняющее его смысл: *Market Saturation* (или *Насыщение рынка*). Этот текст будет показан на презентации.

Из группы кнопок **Тип** выберите символ, который будет отображаться для данного цикла. Выберите символ **B** (обозначающий *Balancing*, то есть компенсирующий цикл).

Чтобы определить, является ли цикл усиливающим или уравновешивающим, Вы можете начать с предположения, что, например значение переменной А увеличивается, и проследить за изменением значений входящих в цикл переменных.

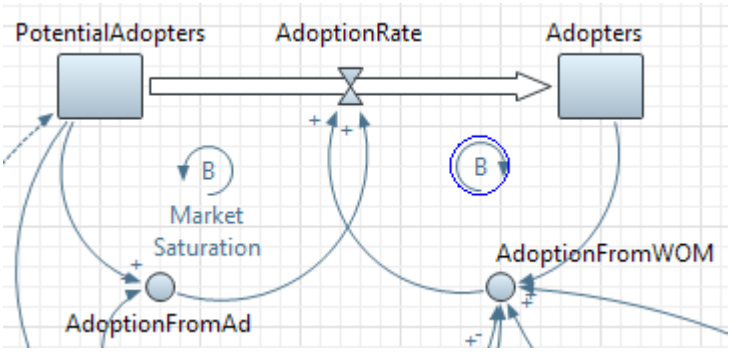
Цикл является:

усиливающим, если после прохождения по циклу Вы видите тот же результат, что был допущен при начальном предположении (в нашем случае - увеличение значения).

уравновешивающим или *компенсирующим*, если результат противоречит начальному предположению.

Задание 7. Добавьте идентификатор для цикла, задающего общение людей друг с другом

Добавьте еще один идентификатор цикла, как показано на рисунке ниже:

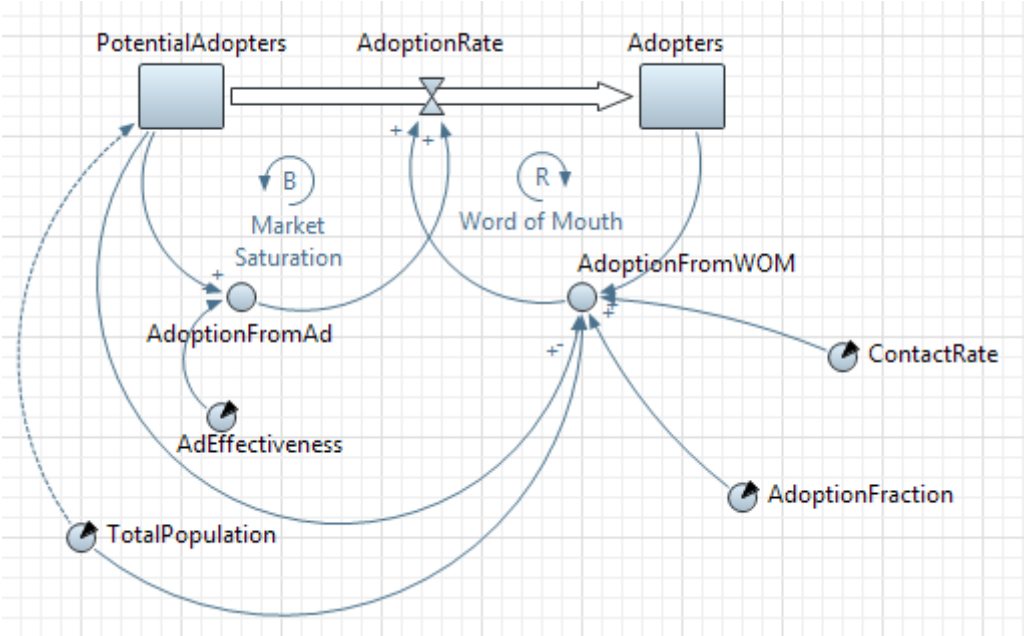


Этот цикл соответствует общению людей друг с другом. Он является усиливающим, поэтому выберите для него символ **R** (обозначающий усиливающий, *Reinforcing* цикл)

Задайте *Word of Mouth* (или *Устное общение*) в качестве текста.

Задайте **Направление** цикла - этот цикл направлен **По часовой стрелке**.

В итоге Ваша диаграмма должна выглядеть следующим образом:

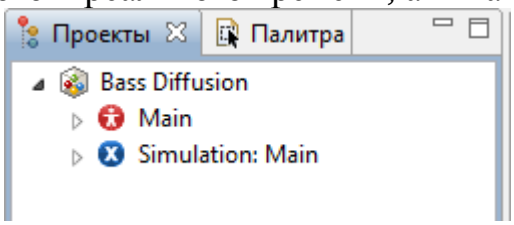


4.8 Настройка запуска модели

Вы можете сконфигурировать выполнение модели в соответствии с Вашими требованиями. Модель выполняется в соответствии с набором установок, задаваемым специальным элементом модели - *экспериментом*. Вы можете создать несколько экспериментов с различными установками и изменять рабочую конфигурацию модели, просто запуская тот или иной эксперимент модели.

В панели **Проекты** эксперименты отображаются в нижней части дерева модели. Один эксперимент, названный *Simulation*, создается по

умолчанию. Это простой эксперимент, позволяющий запускать модель с заданными значениями параметров, поддерживающий режимы виртуального и реального времени, анимацию и отладку модели.



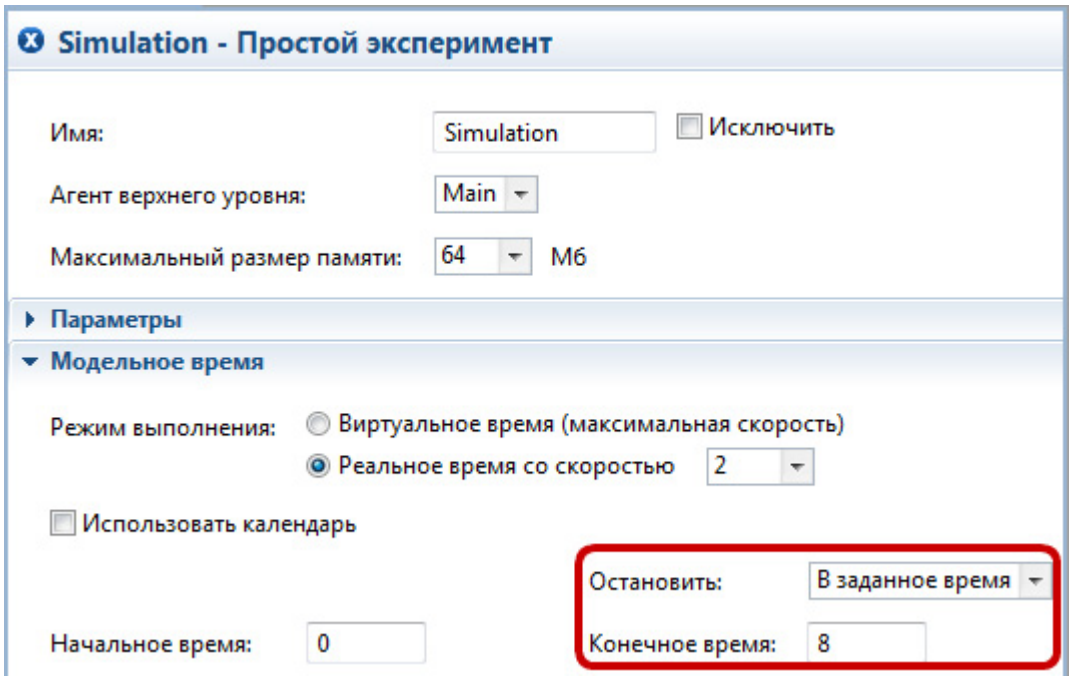
Существуют также и другие типы экспериментов (оптимизационный эксперимент, эксперимент для оценки рисков, эксперимент для варьирования параметров), которые используются в тех случаях, когда параметры модели играют существенную роль, и требуется проанализировать, как они влияют на поведение модели, или когда нужно найти оптимальные значения параметров модели.

Если мы сейчас запустим модель, то она будет моделироваться бесконечно, пока мы сами не остановим ее выполнение . Поскольку мы хотим наблюдать поведение модели только тогда, когда происходит процесс распространения продукта, нам нужно остановить модель, когда система придет в точку равновесия. Процесс распространения продукта в этой модели длится примерно 8 лет.

Задание 1. Задайте остановку модели по прошествии 8 единиц модельного времени

В панели **Проекты**, выделите эксперимент Simulation:Main щелчком мыши.

На странице **Модельное время** панели **Свойства**, выберите **В заданное время** из выпадающего списка **Остановить**. В расположенном ниже поле введите 8. Модель остановится после того, как истекут 8 единиц модельного времени.



Перед тем, как запустить модель, давайте выберем режим ее выполнения. Модель AnyLogic может выполняться либо в режиме виртуального, либо в режиме реального времени.

В *режиме виртуального времени* модель выполняется без привязки к физическому времени – она просто выполняется настолько быстро, насколько это возможно. Этот режим лучше всего подходит в том случае, когда требуется моделировать работу системы в течение достаточно длительного периода времени.

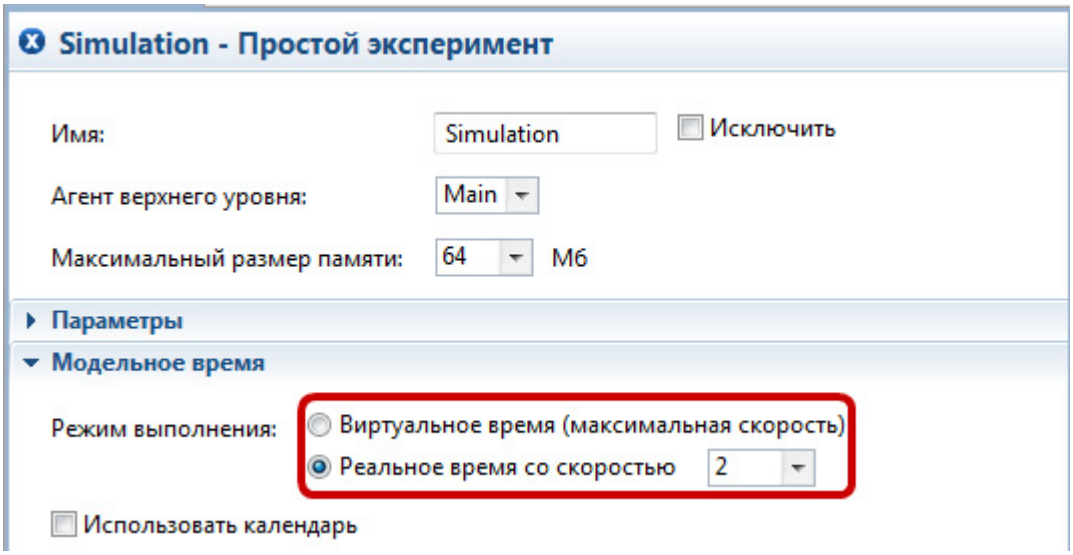
В *режиме реального времени* задается связь модельного времени с физическим, то есть задается количество единиц модельного времени, выполняемых в одну секунду. Это часто требуется, когда Вы хотите, чтобы анимация модели отображалась с той же скоростью, что и в реальной жизни.

Задание 2. Задайте выполнение модели в режиме реального времени

В панели **Проекты**, выделите эксперимент Simulation:Main щелчком мыши.

На странице Презентация панели Свойства, перейдите в раздел Модельное время и выберите опцию Реальное время со скоростью.

Задайте скорость выполнения модели, т. е., сколько единиц модельного времени будет соответствовать одной секунде реального времени. Выберите в выпадающем списке справа, скажем, 2.



И давайте теперь изменим еще одно свойство, но уже не эксперимента, а модели.

Во время моделирования мы будем изучать значения наших переменных и просматривать графики, демонстрирующие, как менялись их значения по ходу моделирования.

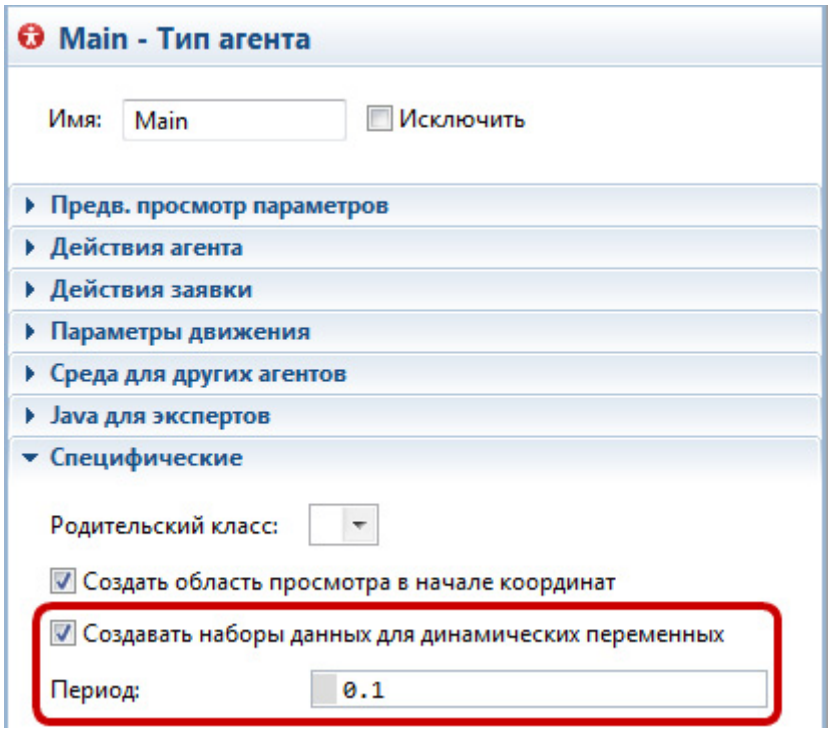
Для этого нам даже не понадобится добавлять и конфигурировать диаграммы - мы воспользуемся специальной возможностью AnyLogic - *окнами инспекта*. Дело в том, что AnyLogic автоматически запоминает значения, принимаемые динамическими переменными по ходу моделирования, в специально создаваемых для этого наборах данных. И значения этих наборов данных можно легко просмотреть в окнах инспекта, которые доступны в режиме запуска модели.

Все, что мы хотим изменить - это увеличить частоту сбора таких данных. По умолчанию новые значения переменных добавляются в наборы данных каждую единицу модельного времени. Мы же хотим, чтобы данные собирались, скажем, в 10 раз чаще.

Задание 3. Увеличьте частоту сбора данных для динамических переменных

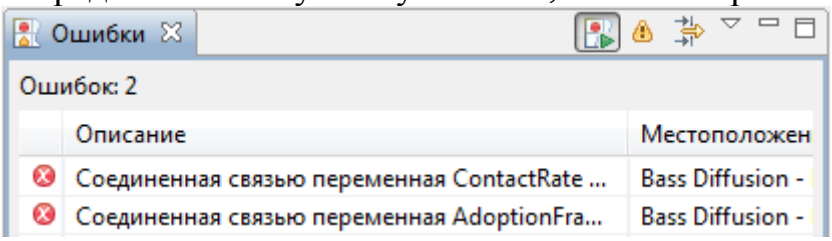
В панели **Проекты**, сделайте двойной щелчок по элементу Main.

Перейдите на страницу **Дополнительные** панели **Свойства**, промотайте ее вниз с помощью расположенного справа бегунка, и введите 0.1 в поле **Период**.



4.9 Запуск модели

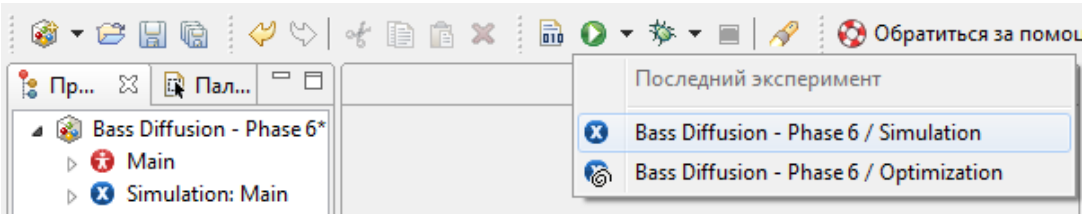
Постройте Вашу модель с помощью кнопки панели инструментов **Построить модель** (при этом в рабочей области AnyLogic должен быть выбран какой-то элемент именно этой модели). Если в модели есть какие-нибудь ошибки, то построение не будет завершено, и в панель **Ошибки** будет выведена информация об ошибках,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



После того, как Вы исправите все ошибки и успешно построите Вашу модель, Вы можете ее запустить.

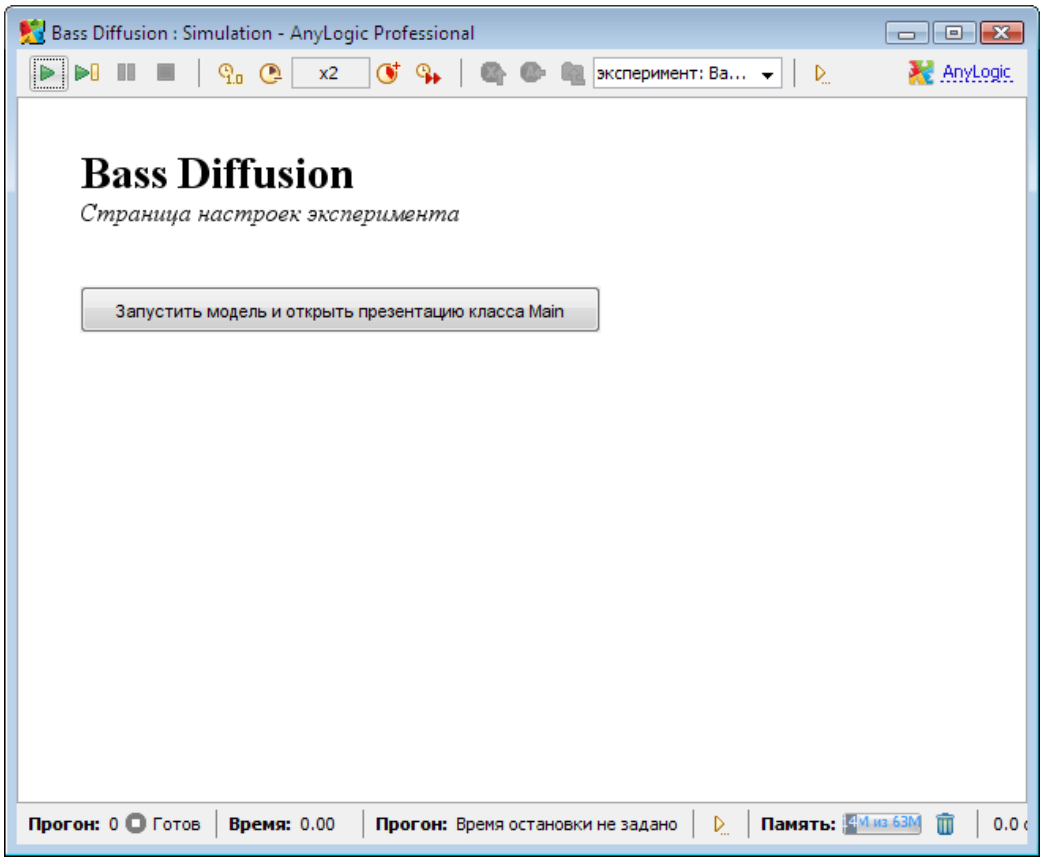
Задание 1. Запустите модель

Щелкните мышью по кнопке панели инструментов **Запустить** и выберите из открывшегося списка эксперимент, который Вы хотите запустить. Эксперимент этой модели будет называться Bass Diffusion/Simulation.

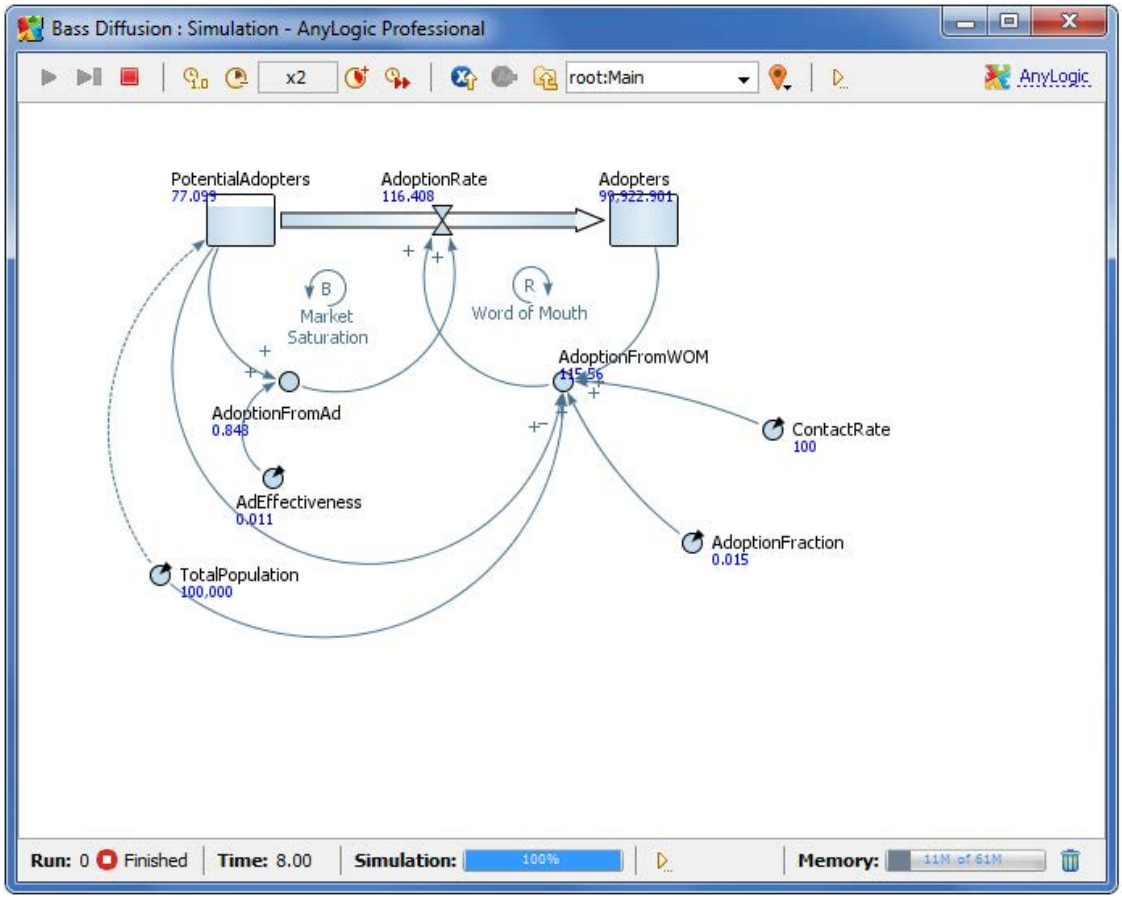


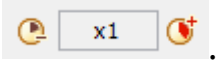
В дальнейшем по нажатию на кнопку **Запустить** (или по нажатию F5) будет запускаться тот эксперимент, который запускался Вами в последний раз. Чтобы выбрать какой-то другой эксперимент, Вам будет нужно щелкнуть мышью по стрелке, находящейся в правой части кнопки **Запустить** и выбрать нужный Вам эксперимент из открывшегося списка (или щелкнуть правой кнопкой мыши по этому эксперименту в панели **Проект** и выбрать **Запустить** из контекстного меню).

Запустив модель, Вы увидите окно презентации этой модели. В нем будет отображена презентация запущенного эксперимента. AnyLogic автоматически помещает на презентацию каждого простого эксперимента заголовок и кнопку, позволяющую запустить модель и перейти на презентацию, нарисованную Вами для главного типа агентов этого эксперимента (Main).



Щелкните по этой кнопке. Вы увидите диаграмму потоков и накопителей. Рядом с каждым элементом будет отображаться его текущее значение.



При желании Вы можете изменить скорость выполнения модели с помощью кнопок панели управления окна презентации **Замедлить** и **Ускорить** 

.

AnyLogic поддерживает различные инструменты для сбора, отображения и анализа данных во время выполнения модели. Простейшим способом просмотра текущего значения и истории изменения значений переменной или параметра во время выполнения модели является использование окна инспекта.

Задание 2. Исследуйте динамику обеих составляющих потока продаж

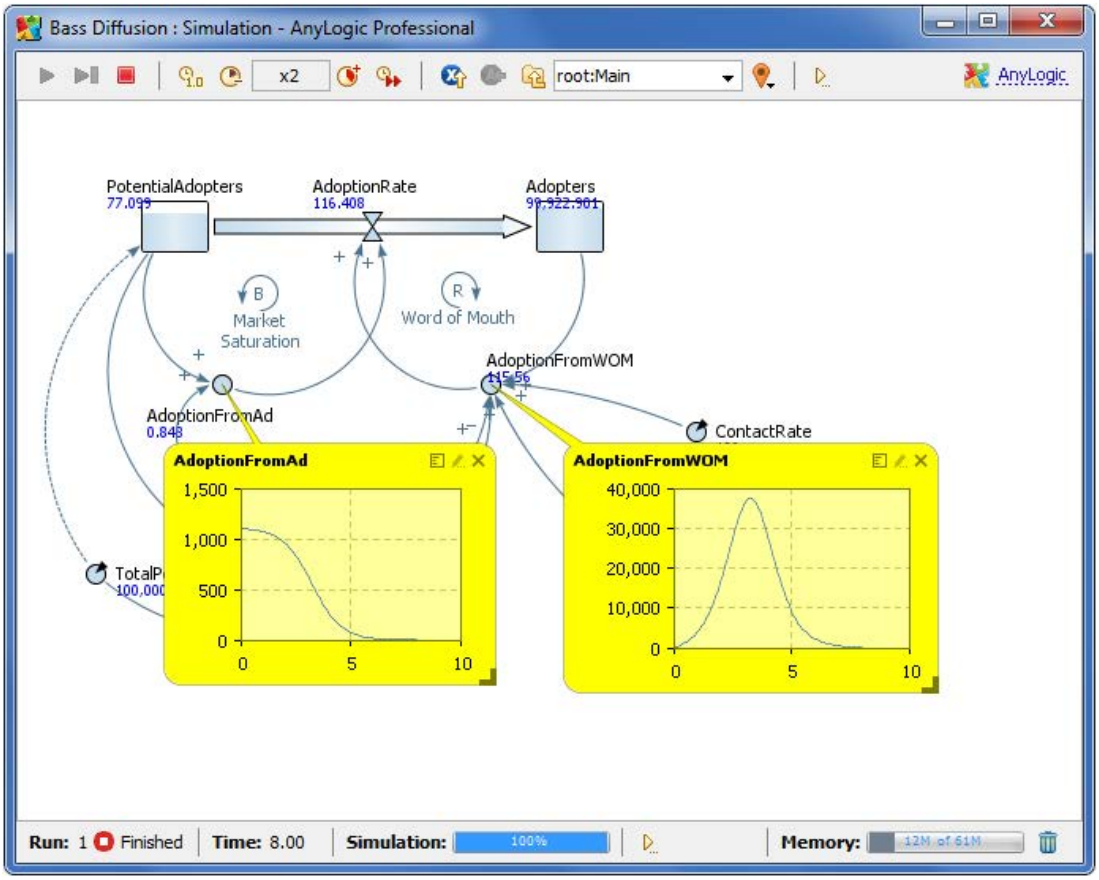
Щелкните мышью по переменной AdoptionFromAd в окне презентации. Появится желтое окошко - окно "инспекта". При желании Вы можете передвинуть это окно, перетащив его мышью за заголовок. Изменить размер можно перетащив мышью нижний правый угол окна.

По умолчанию окно инспекта находится в режиме инспекта - оно отображает текущее значение переменной. Вы можете переключить окно в режим графика, при этом оно будет отображать временной график изменений значения переменной с течением модельного времени. Текущее значение переменной также отображается и в этом режиме (рядом с началом координат графика). Окно инспекта автоматически масштаби-

руется таким образом, чтобы полностью вместить кривые графиков от начала до конца периода моделирования.

Обычно окна инспекта используются только для беглого ознакомления с проистекающими в модели процессами, для более тщательной визуализации и анализа данных используются диаграммы и объекты сбора статистики, расположенные на палитре **Статистика**. С помощью этих элементов Вы можете добавлять на презентацию любые графики, диаграммы и гистограммы и вести статистический анализ собранных данных.

Аналогичным образом откройте окно инспекта переменной AdoptionFromWOM и переключите его в режим графика.

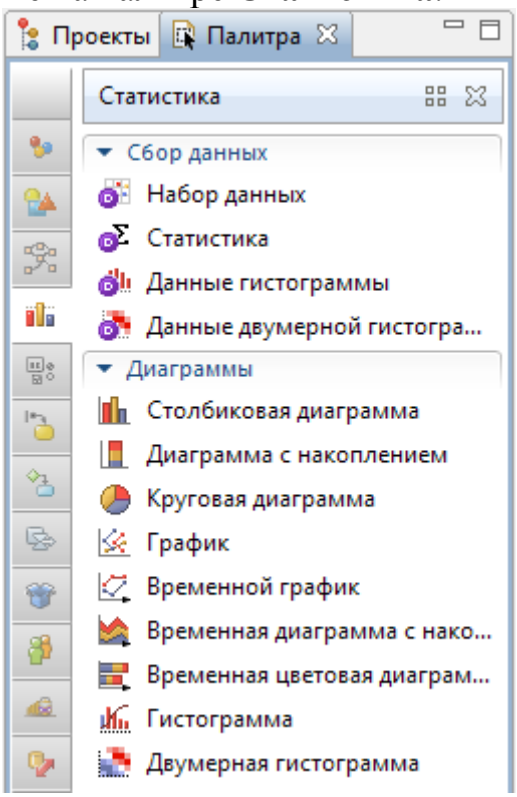


Теперь мы можем увидеть, что при внедрении нового продукта на рынок, когда число потребителей равно нулю, реклама будет являться единственным источником продаж. Наибольший рекламный эффект отмечается в начале процесса распространения продукта; он неуклонно падает по мере уменьшения численности потенциальных потребителей.

4.10 Добавление диаграмм

Как мы уже отмечали ранее, AnyLogic поддерживает различные инструменты для сбора, отображения и анализа данных во время выполнения модели. Простейшим способом просмотра истории изменения

значений переменной во время выполнения модели является использование окна инспекта. Для более тщательной визуализации и анализа данных используются *диаграммы* и *объекты сбора данных*, расположенные на палитре **Статистика**.

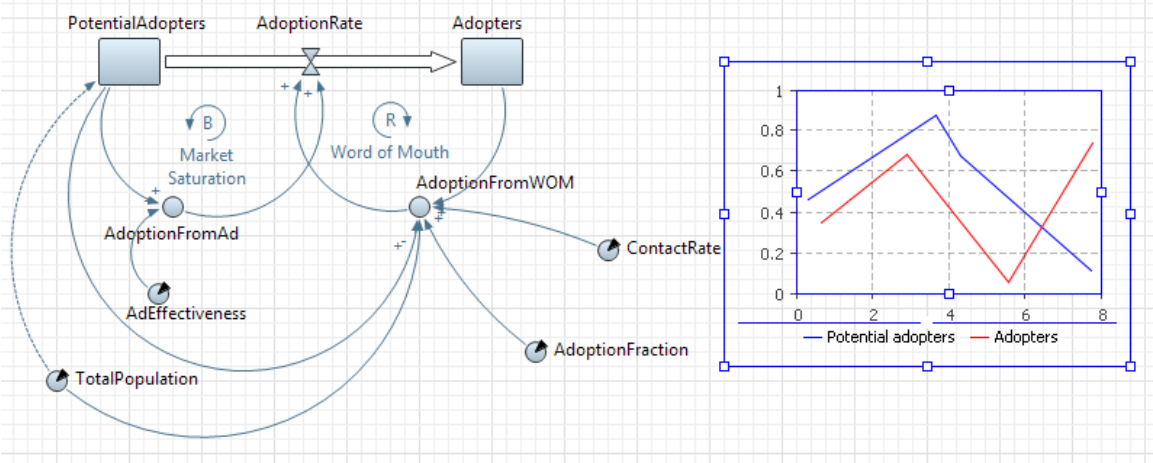


С помощью этих элементов Вы можете добавлять на презентацию любые графики, диаграммы и гистограммы и вести статистический анализ собранных данных.

Давайте добавим диаграммы, с помощью которых мы будем изучать, как изменяются со временем численности потребителей и потенциальных потребителей продукта, а также как изменяется интенсивность продаж продукта.

Задание 1. Добавьте график, отображающий динамику изменения численностей потребителей и потенциальных потребителей продукта

Перетащите элемент **Временной график** из палитры **Статистика** на диаграмму класса *Main* и измените размер графика, как показано на приведенном ниже рисунке:



Перейдите на страницу **Основные** панели **Свойства**.

В поле **Временной диапазон** задайте диапазон временной оси диаграммы (количество единиц модельного времени (N), для которого будут отображаться значения переменной): 8. Диаграмма будет отображать график только для заданного временного интервала (равного в нашем случае длительности периода моделирования).

Измените частоту обновления графика новыми данными. Введите 0.1 в поле **Период**, справа от секции **Обновлять автоматически**.

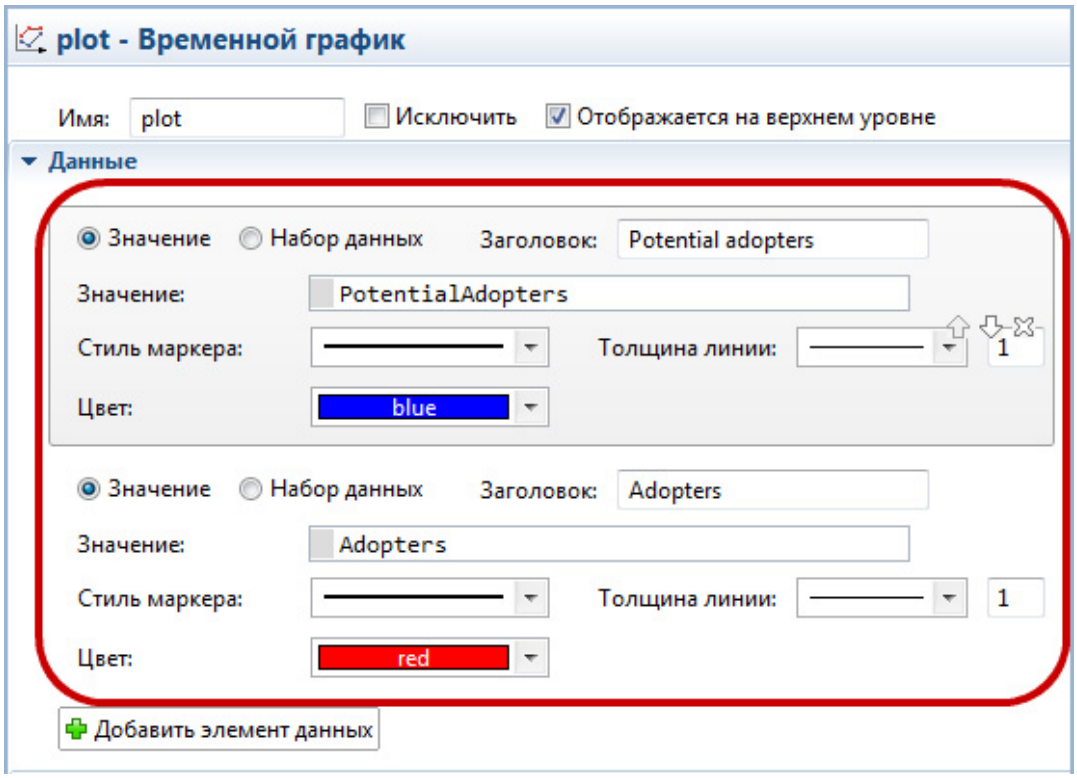
Добавьте элементы данных, историю изменения значений которых Вы хотите отображать на этом временном графике.

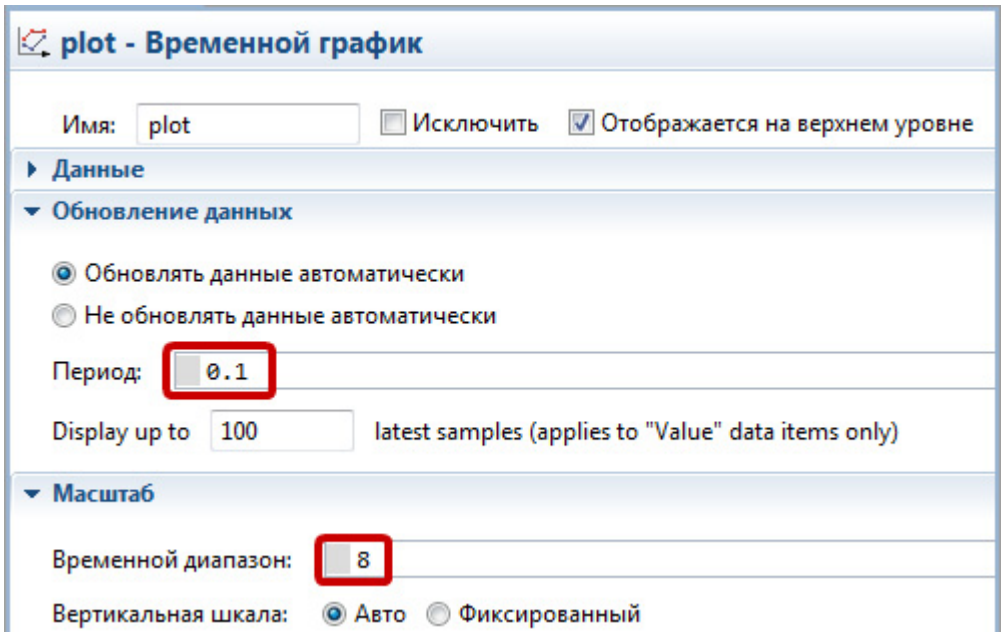
Чтобы добавить новую секцию свойств, в которой задаются свойства отображаемого на графике элемента данных, щелкните мышью по кнопке **Добавить элемент данных**.

Задайте выражение, результат вычсиления которого будет отображаться на графике. Мы хотим отображать численность потенциальных потребителей, поэтому введите в поле **Значение** имя соответствующего накопителя: PotentialAdopters.

В поле **Заголовок** введите Potential adopters. Эта строка будет отображаться в легенде диаграммы для этого элемента данных.

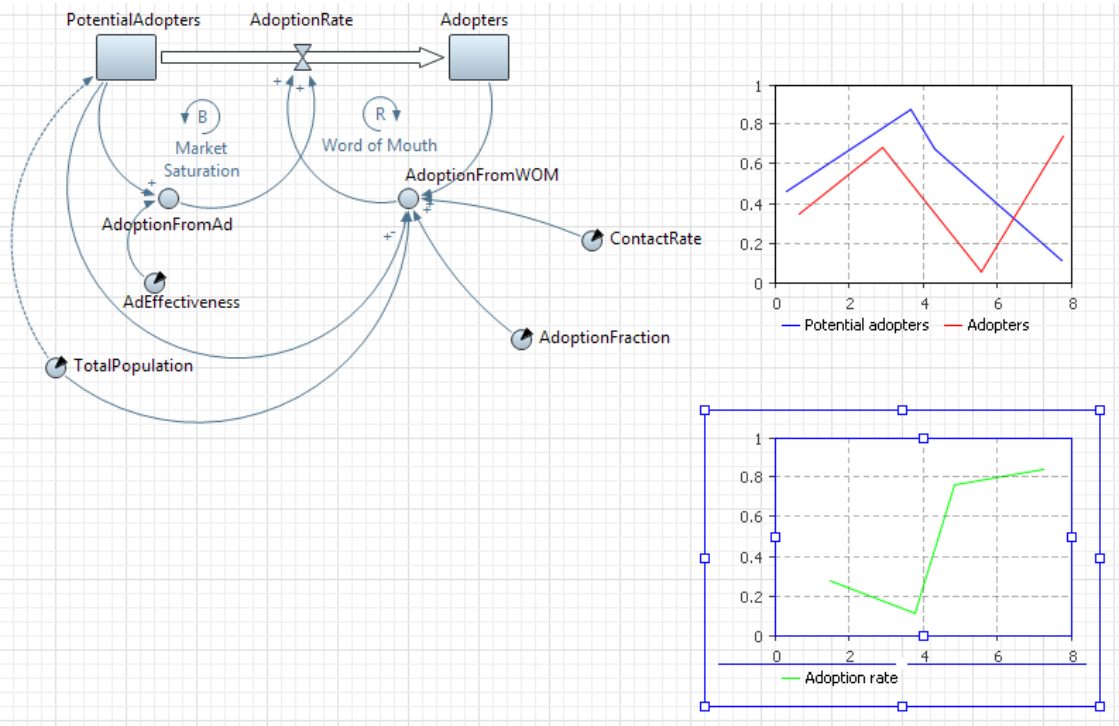
Аналогично добавьте на график еще один элемент данных, который будет отображать значение накопителя Adopters (если у Вас возникнут вопросы, то Вы можете свериться с расположенным ниже рисунком):



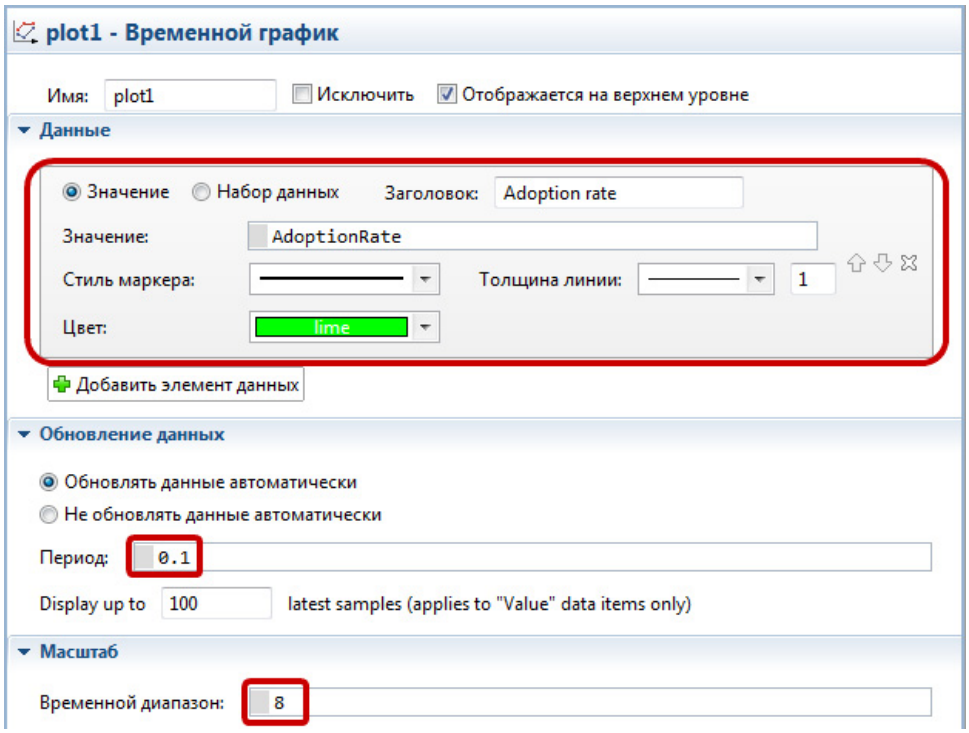


Задание 2. Добавьте график, отображающий изменение интенсивности продаж

Добавьте на диаграмму еще один временной график. Поместите его под добавленным ранее графиком:



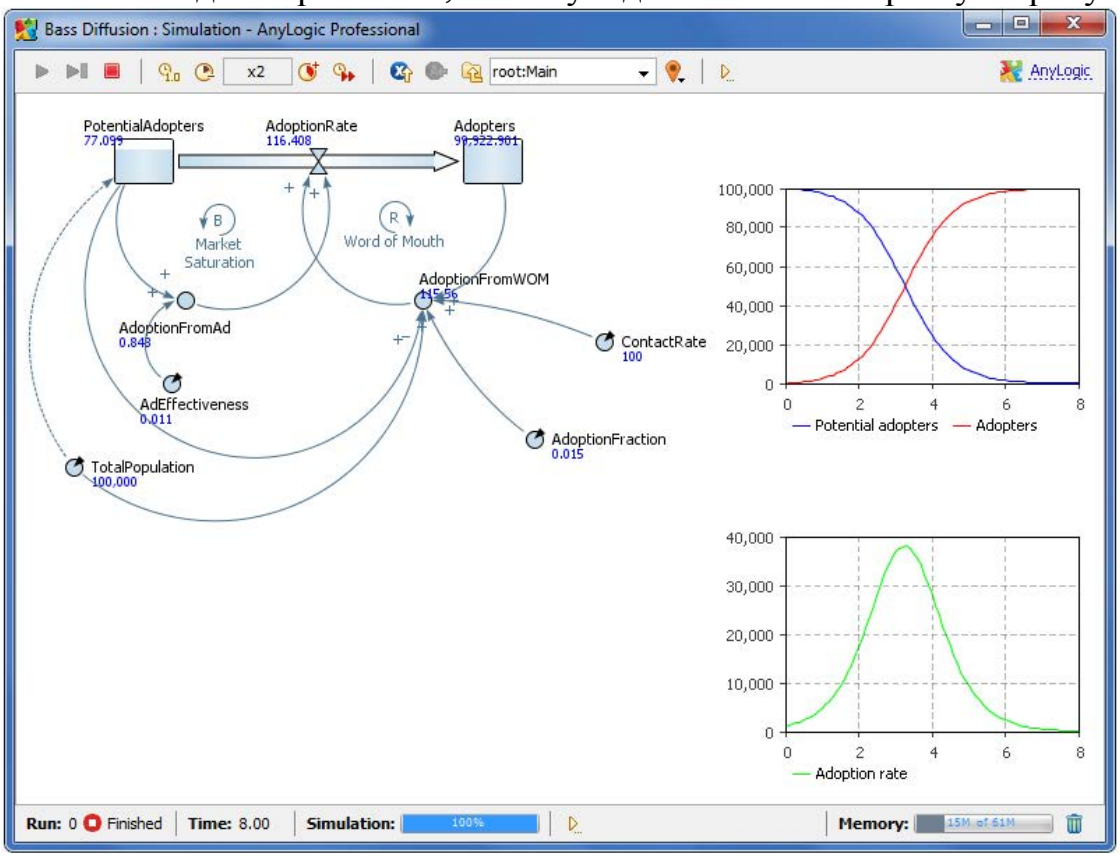
Добавьте на график новый элемент данных (в качестве **Значения** в этом случае должно быть задано имя потока AdoptionRate) и измените свойства графика, как показано на приведенном рисунке:



Теперь Вы можете запустить модель и изучить динамику изменения численностей потребителей и потенциальных потребителей продук-

та. Вы увидите классические для рассматриваемого примера системной динамики кривые S-формы.

С помощью нижнего графика Вы можете проследить, как с течением времени будет изменяться интенсивность продаж продукта. Если модель была создана правильно, то Вы увидите колоколообразную кривую:



Мы закончили создание простейшей модели системной динамики. Эта модель часто используется в классических учебниках по системной динамике, и именно поэтому она и была выбрана нами для учебного пособия. На этом примере мы хотели продемонстрировать Вам, как создаются типовые модели системной динамики в AnyLogic.

Теперь мы можем несколько усовершенствовать эту модель, попутно продемонстрировав специализированные возможности AnyLogic, реализованные в первую очередь именно для приверженцев системно-динамического метода моделирования.

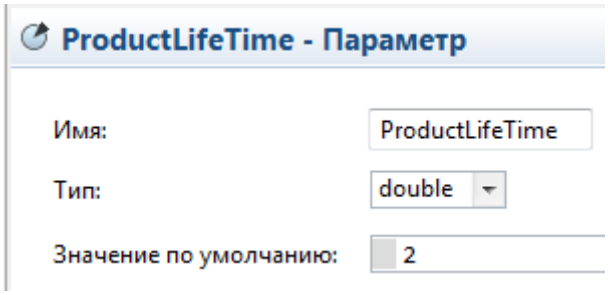
4.11 Моделирование повторных покупок

Созданная модель не учитывает того, что со временем продукт может быть израсходован или прийти в негодность, что вызовет необходимость его повторного приобретения. Мы смоделируем повторные покупки, полагая, что потребители продукта снова становятся потенциальными потребителями, когда продукт, который они приобрели, становится непригоден.

Вначале мы определим константу, задающую среднее время жизни продукта.

Задание 1. Создайте константу **ProductLifeTime**

Пусть средняя продолжительность использования нашего продукта равна двум годам. Введите **Значение по умолчанию** 2.



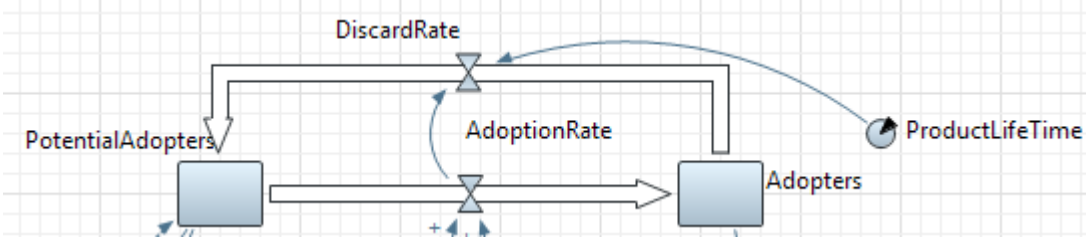
Потребители продукта снова становятся потенциальными потребителями тогда, когда продукт, который они приобрели, расходуется и перестает использоваться. Поэтому поток прекращения использования продукта является ничем иным, как потоком приобретения, задержанным на среднее время пригодности продукта.

Задание 2. Создайте поток прекращения использования продукта, ведущий из **Adopters** в **PotentialAdopters**

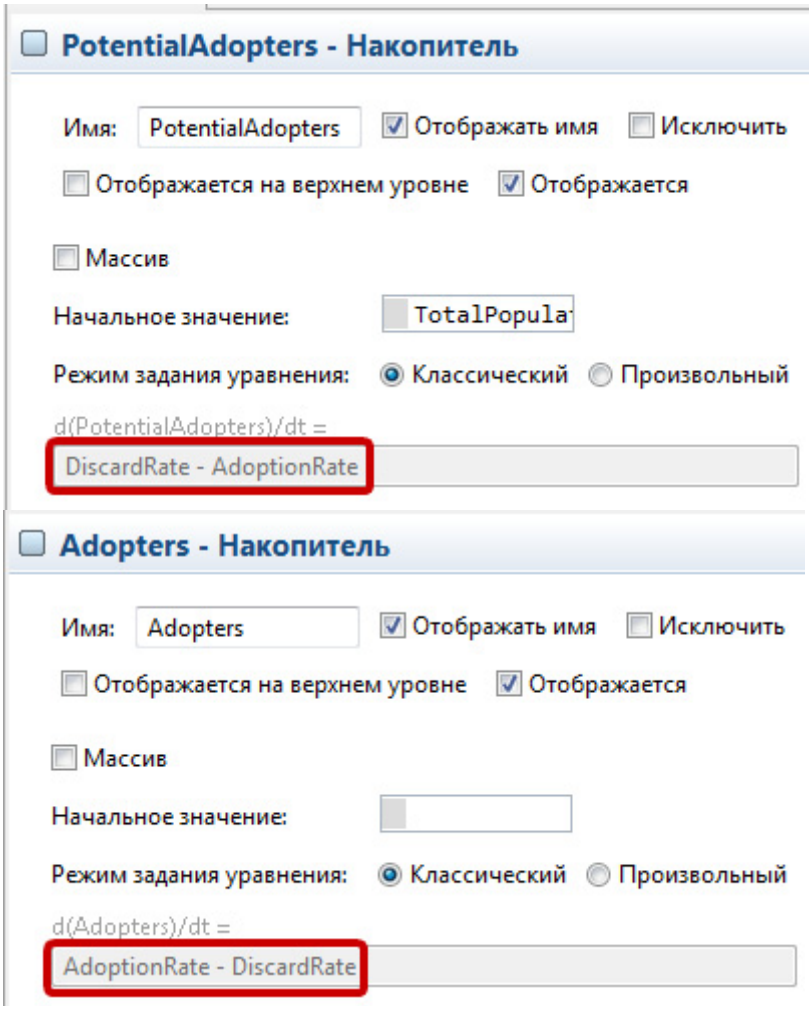
Если нарисовать новый поток прямой стрелкой, ведущей от Adopters к PotentialAdopters, то этот поток будет нарисован поверх стрелки существующего потока AdoptionRate. Поэтому давайте нарисуем поток более сложной формы (см. рисунок ниже). Для этого используем другой, более подходящий для данного случая, способ рисования потоков.

Сделайте двойной щелчок по элементу **Поток** в палитре **Системная динамика**. Значок элемента при этом должен смениться на следующий: .

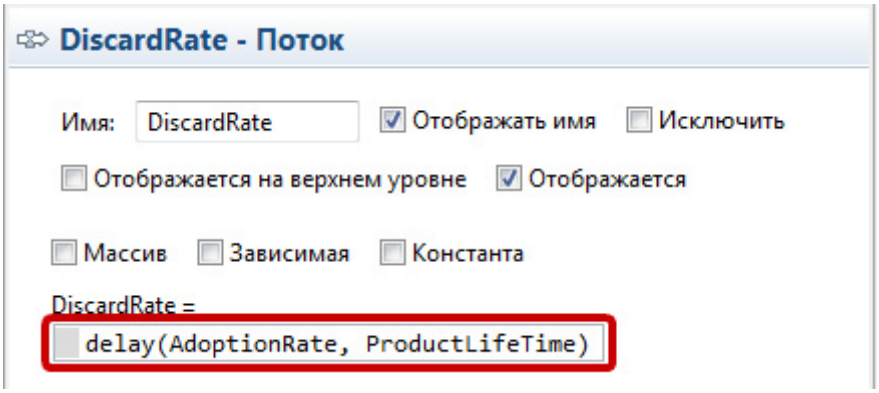
Сразу после этого щелкните мышью по накопителю Adopters, потом щелкните в промежуточных точках изгиба стрелки потока, и завершите рисование потока, сделав двойной щелчок по накопителю PotentialAdopters, в который этот поток втекает.



Назовите поток DiscardRate. Формулы накопителей после этого должны будут выглядеть следующим образом:



Задайте следующую формулу для потока DiscardRate: delay(AdoptionRate, ProductLifeTime)



Функция delay() реализует временную задержку; она имеет следующую нотацию:

delay(<задерживаемый поток>, <значение задержки>, <начальное значение>)

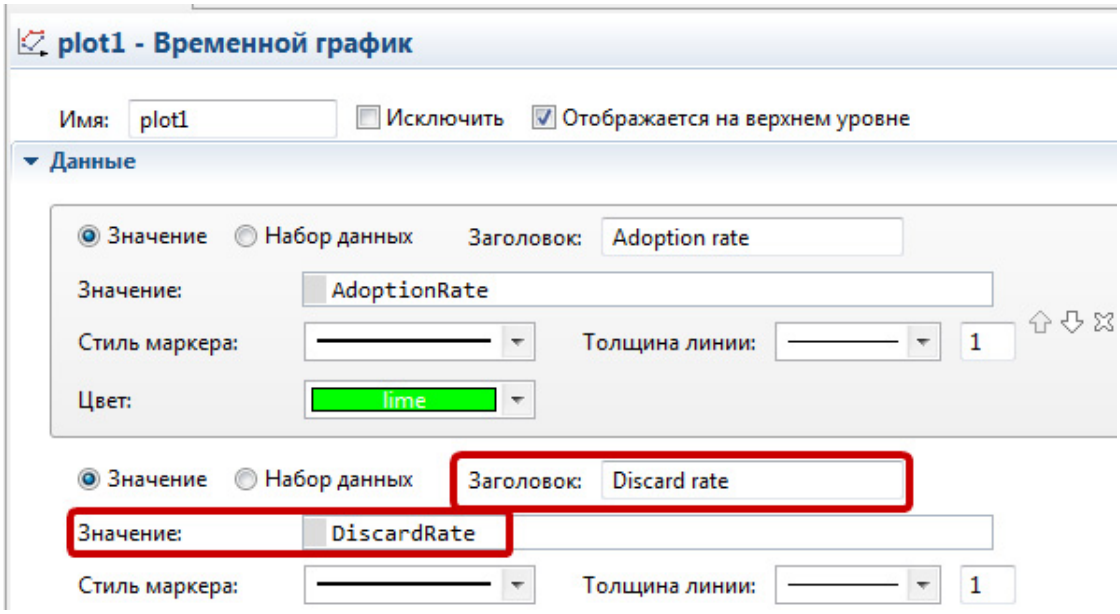
В нашем случае функция представляет собой AdoptionRate с временной задержкой ProductLifeTime. Пока не истекло время использования первого приобретенного продукта, поток равен нулю.

Как и во всех других аналогичных случаях, написание такой формулы потребует добавления ссылок от переменных AdoptionRate и ProductLifeTime к потоку DiscardRate.

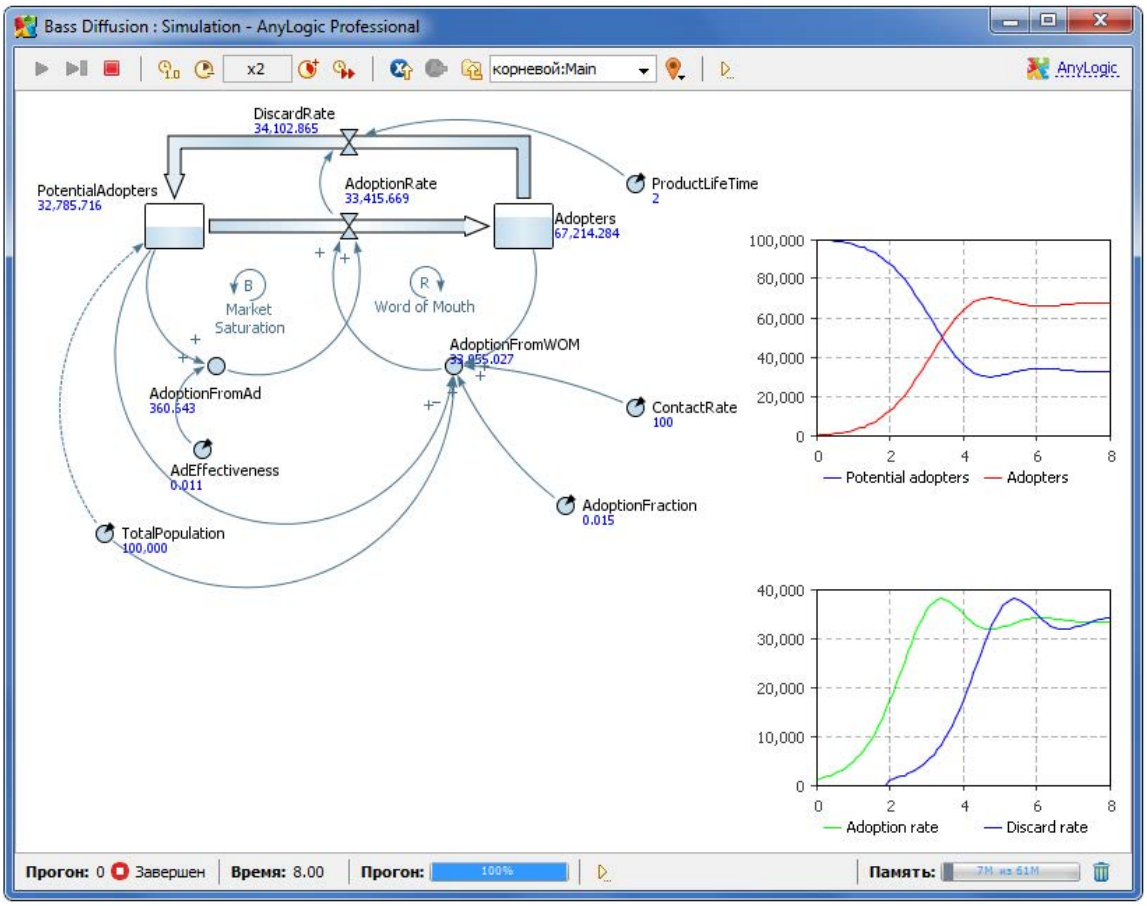
Проверить работу функции задержки проще всего с помощью диаграммы. Для этого мы добавим на график, отображающий динамику изменения интенсивности продаж, и другую интенсивность - интенсивность отказа от продукта, определяемую нашим потоком DiscardRate.

Задание 3. Добавьте на нижний график новый элемент данных, отображающий динамику изменения интенсивности DiscardRate

Добавьте на нижний график еще один элемент данных, аналогично тому, как мы это делали на предыдущем шаге учебного пособия. Задайте свойства этого элемента, как показано на рисунке:



Теперь мы закончили моделирование повторных покупок продукта. Вы можете проверить, как работает функция задержки. Запустите модель и исследуйте графики переменных AdoptionRate и DiscardRate. Вы сможете увидеть, что график потока прекращения использования продукта имеет именно тот вид, который мы и предполагали увидеть—он является ничем иным, как потоком приобретения продукта, задержанным на 2 года—время пригодности продукта.



С помощью диаграммы проследите динамику изменения численностей потребителей.

Теперь численность потенциальных потребителей не уменьшается до нуля, а постоянно пополняется по мере того, как потребители заново покупают продукты взамен непригодных. Интенсивность приобретения продукта растет, падает, и в итоге принимает какое-то значение, зависящее от средней жизни продукта и параметров, определяющих интенсивность этого потока. Наличие в модели прекращения использования продукта означает, что какая-то доля населения всегда будет оставаться потенциальными потребителями.

4.12 Моделирование цикличности спроса

В текущей модели доля контактов потребителей продукта с потенциальными потребителями, которая приводит к продажам продукта, полагается постоянной. На самом деле она изменяется, поскольку спрос на наш продукт зависит от текущего времени года. Продукт пользуется наибольшим спросом летом, в то время как зимой спрос на товар резко падает, за исключением небольшого предпраздничного периода в декабре. Давайте и промоделируем теперь сезонную цикличность спроса.

Предположим, что у нас есть экспериментальные данные того, как изменяется средний спрос на продукт в течение года. Мы добавим эти данные в нашу модель с помощью табличной функции. Табличная

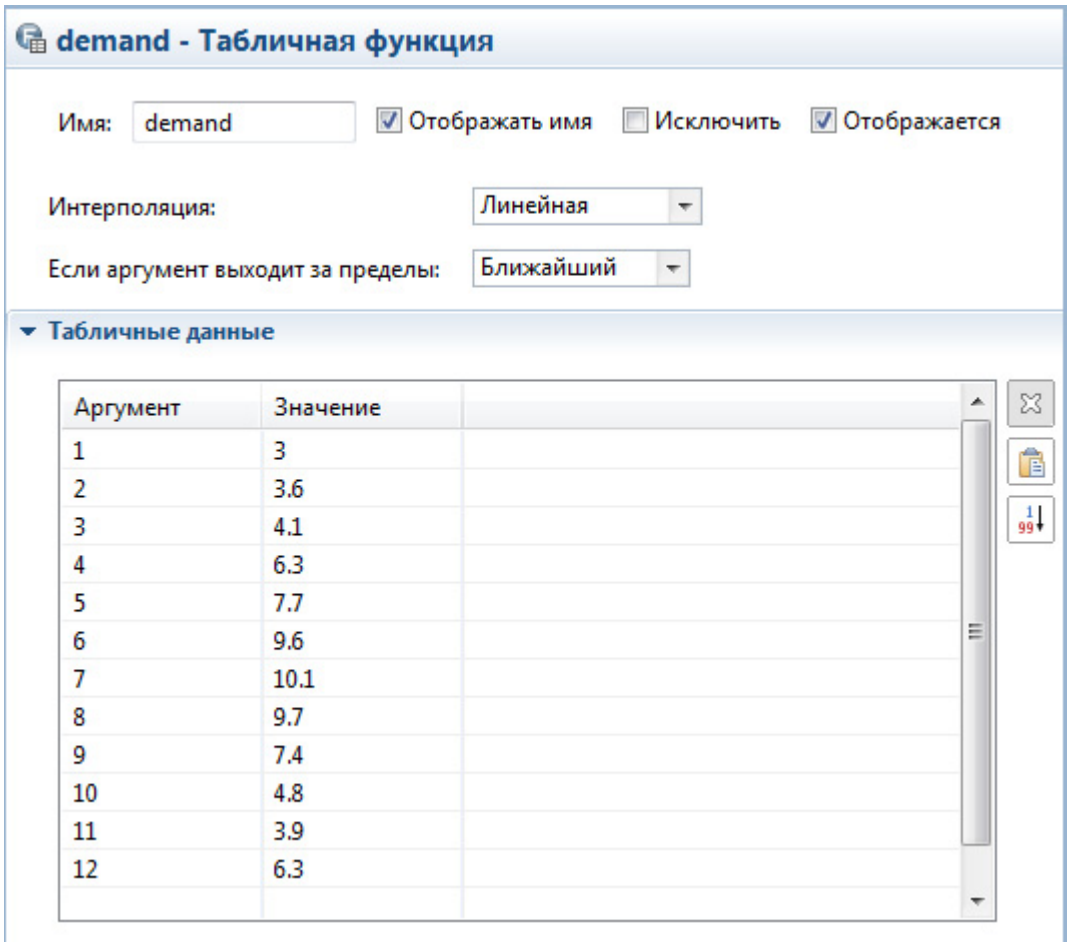
функция – это функция, заданная в табличной форме, которая может быть сделана непрерывной с помощью интерполяции и экстраполяции.

Задание 1. Задание кривой спроса с помощью табличной функции

Перетащите элемент **Табличная функция** из палитры **Системная динамика** на диаграмму класса *Main*.

Назовите функцию demand.

Задайте данные табличной функции в таблице **Табличные данные** на странице свойств функции. Каждая пара "аргумент-значение" задается в отдельной строке таблицы. Чтобы задать новую пару значений, щелкните мышью в пустой ячейке **Аргумент** и введите новый аргумент функции. Затем щелкните в соседней ячейке **Значение** справа и введите значение функции, соответствующее этому аргументу. Задайте следующие данные:

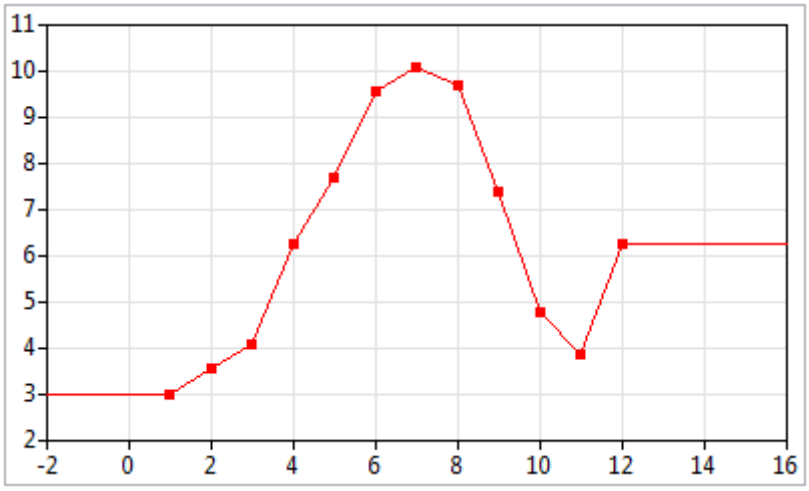


Аргумент	Значение
1	3
2	3.6
3	4.1
4	6.3
5	7.7
6	9.6
7	10.1
8	9.7
9	7.4
10	4.8
11	3.9
12	6.3

Задайте тип интерполяции. Выберите **Линейная** из выпадающего списка **Интерполяция**. Интерполяция будет производиться соединением табличных точек прямыми линиями.

Задайте тип реакции на аргументы, лежащие за пределами области допустимых значений функции. Выберите **Ближайший** из выпадающего списка **Если аргумент выходит за пределы**. В случае вызова функции с аргументом, лежащим за пределами интервала заданных значений, будет использоваться ближайший заданный аргумент функции.

Закончив задание функции, Вы можете посмотреть, как она выглядит, с помощью графика предварительного просмотра функции на странице ее свойств:



Теперь мы хотим промоделировать то, как спрос на продукт влияет на количество людей, приобретающих продукт под влиянием общения с владельцами продукта. Для этого мы создадим специальную функцию и заменим параметр AdoptionFraction вспомогательной переменной, значение которой будет вычисляться согласно этой функции.

Задание 2. Задайте функцию

Перетащите элемент **Функция** из палитры **Основная** на диаграмму класса *Main*.

Назовите функцию adoptFraction.

Функция будет возвращать вещественное значение, поэтому выберите **double** из группы кнопок **Тип возвращаемого значения**.

У функции должен быть один аргумент, с помощью которого ей будет передаваться текущее значение времени. Добавьте в таблицу **Аргументы функции** аргумент с именем time типа **double**.

На странице свойств **Код** задайте выражение функции. В поле **Тело функции** замените существующую строку на следующую:

```
return demand((time-floor(time))*12+1)/200.0;
```

Это выражение вычисляет номер текущего месяца и передает его табличной функции demand. Табличная функция возвращает значение спроса на продукт для данного месяца. В заключение, для получения значения доли людей, покупающих продукт под влиянием общения, значение спроса делится на коэффициент преобразования.

Функция floor() является одной из набора математических функций, которые AnyLogic предоставляет пользователям, наряду с такими, как sin, cos, exp, и т. д. Вводя выражения, Вы можете пользоваться Мастером подстановки кода, в котором все доступные в данном контексте функции присутствуют наряду с другими элементами модели, переменными, аргументами функций и т.д.

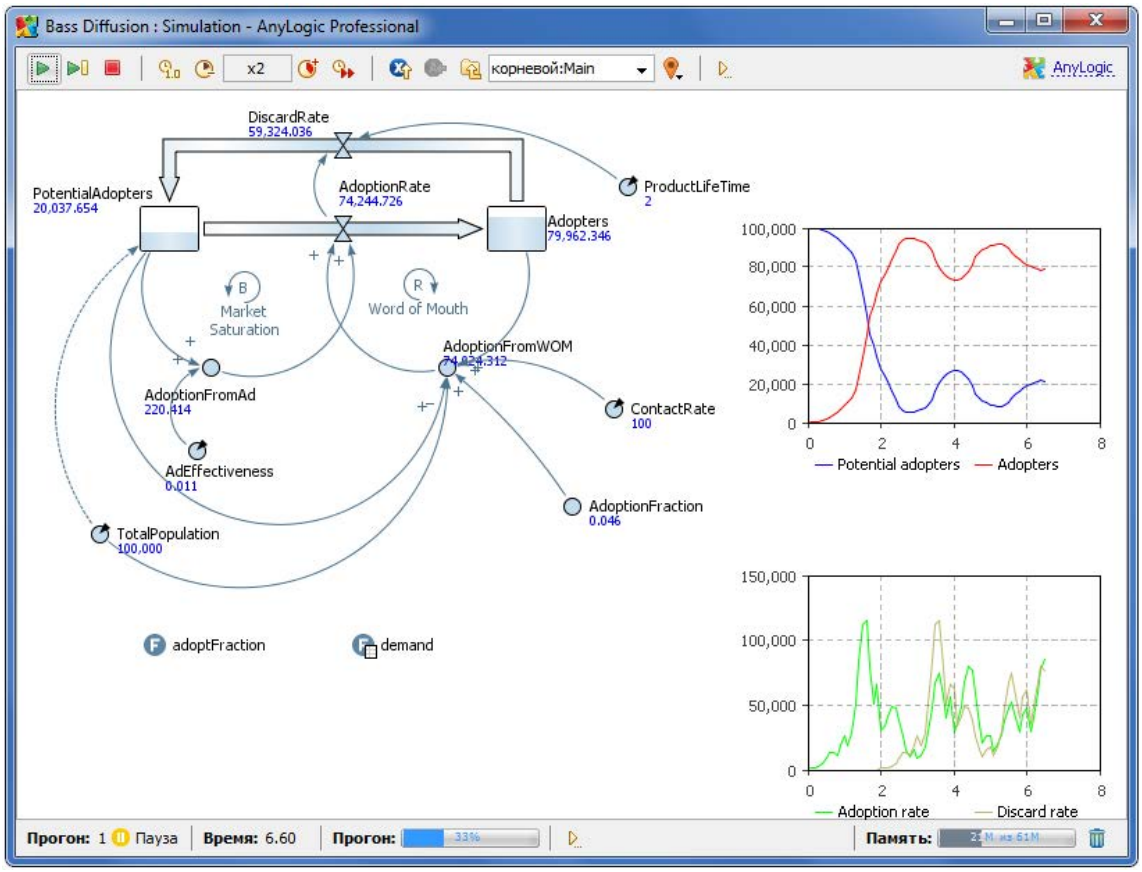
Ну и наконец, нужно будет заменить константу вспомогательной переменной, значение которой определяется нашей функцией.

Задание 3. Замените параметр AdoptionFraction вспомогательной переменной

Удалите параметр AdoptionFraction.

Добавьте вспомогательную переменную AdoptionFraction. Задайте в качестве формулы переменной adoptFraction(time()). Таким образом, значение переменной будет вычисляться согласно нашей функции. Функция принимает один аргумент, time(). Функция time() возвращает текущее значение модельного времени.

Запустите модель. Вы увидите, что теперь поведение модели колеблется около точки равновесия в силу того, что теперь колеблются значения и потока приобретения, и потока прекращения использования продукта.



4.13 Моделирование стратегии рекламной кампании

На данный момент эффективность рекламы в нашей модели полагается постоянной. На самом деле, она зависит от текущих расходов компании на рекламу. Мы хотим улучшить нашу модель, чтобы иметь возможность управлять расходами на рекламную кампанию. Изменяя месячные расходы на рекламу, мы сможем повлиять на текущую эффективность рекламы.

Задание 1. Создайте константу, задающую месячные расходы компании

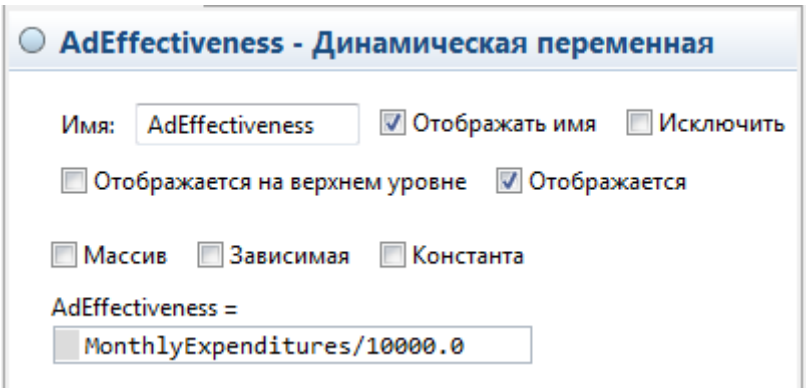
Создайте параметр MonthlyExpenditures.

Задайте Значение по умолчанию: 1100.

Задание 2. Замените параметр AdEffectiveness вспомогательной переменной

Удалите параметр AdEffectiveness.

Создайте вспомогательную переменную AdEffectiveness с формулой: MonthlyExpenditures/10000.0. Мы полагаем, что именно так эффективность рекламы зависит от текущих рекламных расходов компании.



Мы хотим вести статистику всех расходов компании. Это может быть сделано созданием специальной переменной для хранения информации о том, сколько денег было потрачено на рекламу продукта. Каждый месяц мы будем обновлять это значение с помощью специального события, добавляя значение запланированных на предстоящий месяц расходов на рекламную кампанию продукта.

Задание 3. Добавьте переменную для хранения общих расходов компании

Перетащите элемент **Переменная** из палитры **Основная** на диаграмму класса Main.

Назовите переменную TotalExpenditures.

Задание 4. Создайте событие, которое будет обновлять значение TotalExpenditures

Перетащите элемент **Событие** из палитры **Основная** на диаграмму класса Main.

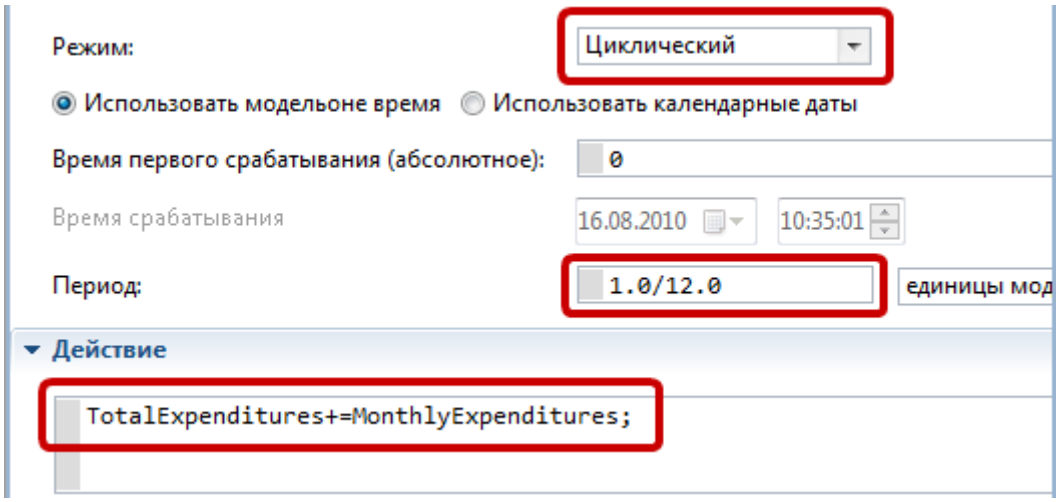
Назовите событие monthlyEvent.

Сделайте так, чтобы таймер срабатывал каждый месяц. Выберите **Циклический** из выпадающего списка **Режим**. Поскольку одна единица модельного времени в нашей модели соответствует одному году, то одному месяцу будет соответствовать выражение 1.0/12.0. Введите 1.0/12 в поле **Таймаут**.

Задайте **Действие** события:

TotalExpenditures += MonthlyExpenditures;

Этот код будет выполняться каждый раз по истечении таймаута события. Он выполняет сбор статистики, а именно добавляет значение запланированных рекламных расходов на предстоящий месяц к значению переменной TotalExpenditures.



Поскольку реклама играет значительную роль только в начальной стадии процесса завоевания рынка, мы хотим в какой-то момент времени, скажем, через 3 года остановить рекламную кампанию. Этим мы сэкономим деньгги, бесцельно тратящиеся на рекламу тогда, когда насыщение рынка будет определяться практически исключительно покупками продукта, вызванными общением потребителей с потенциальными потребителями.

Задание 5. Добавьте константу, задающую время переключения

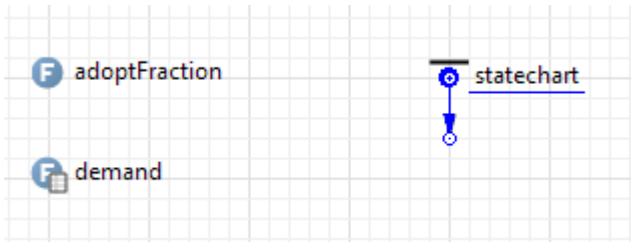
Добавьте параметр SwitchTime.

Задайте Значение по умолчанию: 3.

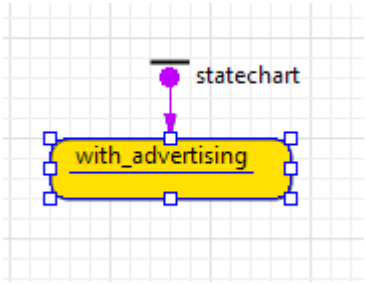
Теперь мы визуально зададим поведение системы с помощью диаграммы состояний.

Задание 6. Создайте диаграмму состояний для моделирования рекламной стратегии

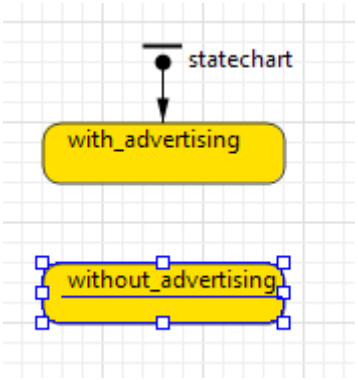
Начните задание диаграммы состояний с добавления элемента **Начало диаграммы состояний** (перетащите этот элемент на диаграмму из палитры **Диаграмма состояний**).



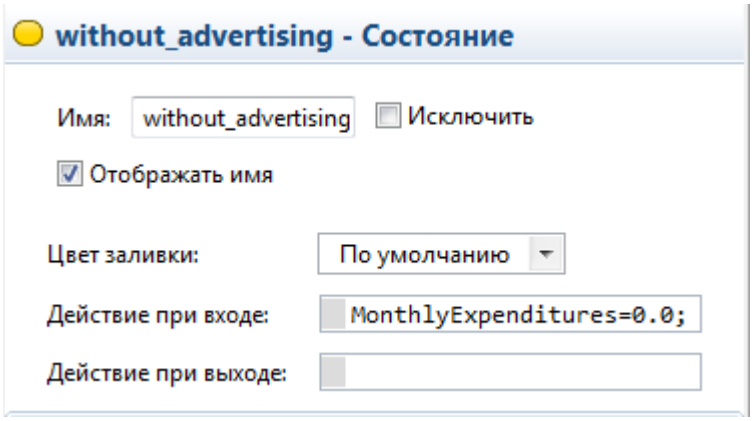
Добавьте состояние. Перетащите элемент **Состояние** из палитры **Диаграмма состояний** так, чтобы состояние присоединилось к добавленному ранее элементу (см. рисунок ниже). Измените имя состояния на with_advertising.



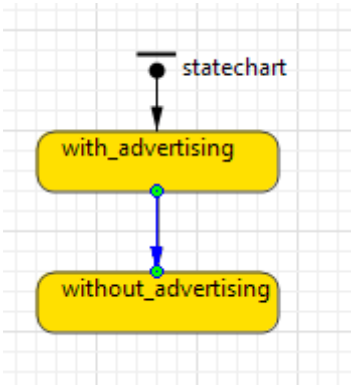
Добавьте еще одно состояние под только что созданным. Назовите его without_advertising.



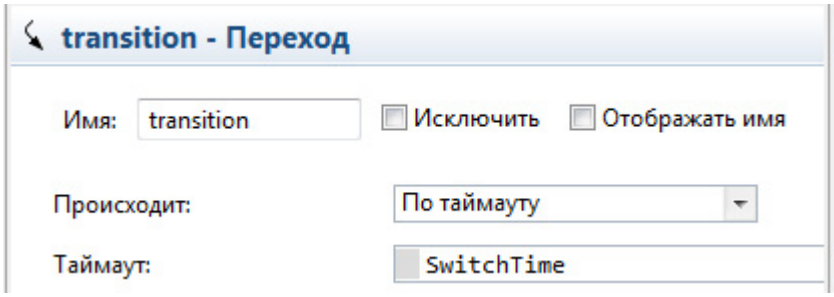
Нам нужно остановить рекламную кампанию в тот момент, когда диаграмма состояний войдет в это состояние. Поэтому напишите в поле свойства **Действие при входе** этого состояния MonthlyExpenditures=0.0;



Добавьте переход, ведущий из состояния with_advertising в состояние without_advertising. Сделайте двойной щелчок мышью по элементу **Переход** в палитре **Диаграмма состояний**. Затем нарисуйте переход, ведущий из состояния with_advertising в состояние without_advertising, щелкнув по границе верхнего состояния, а затем по границе нижнего состояния.



Укажите, что этот переход произойдет по истечении времени SwitchTime. Для этого выберите **По таймауту** из выпадающего списка **Происходит** и введите SwitchTime в поле **Таймаут**.



Теперь, когда диаграмма состояний находится в начальном состоянии with_advertising, рекламные расходы кампании определяются переменной MonthlyExpenditures. Как только диаграмма состояний покидает это состояние в момент времени SwitchTime, компания перестает рекламировать продукт.

Запустите модель и убедитесь, что рекламная кампания длится теперь только три года.

4.14 Оптимизация рекламной стратегии

Рыночная стратегия в данной модели предельно проста: в определенный момент времени компания прекращает рекламировать продукт.

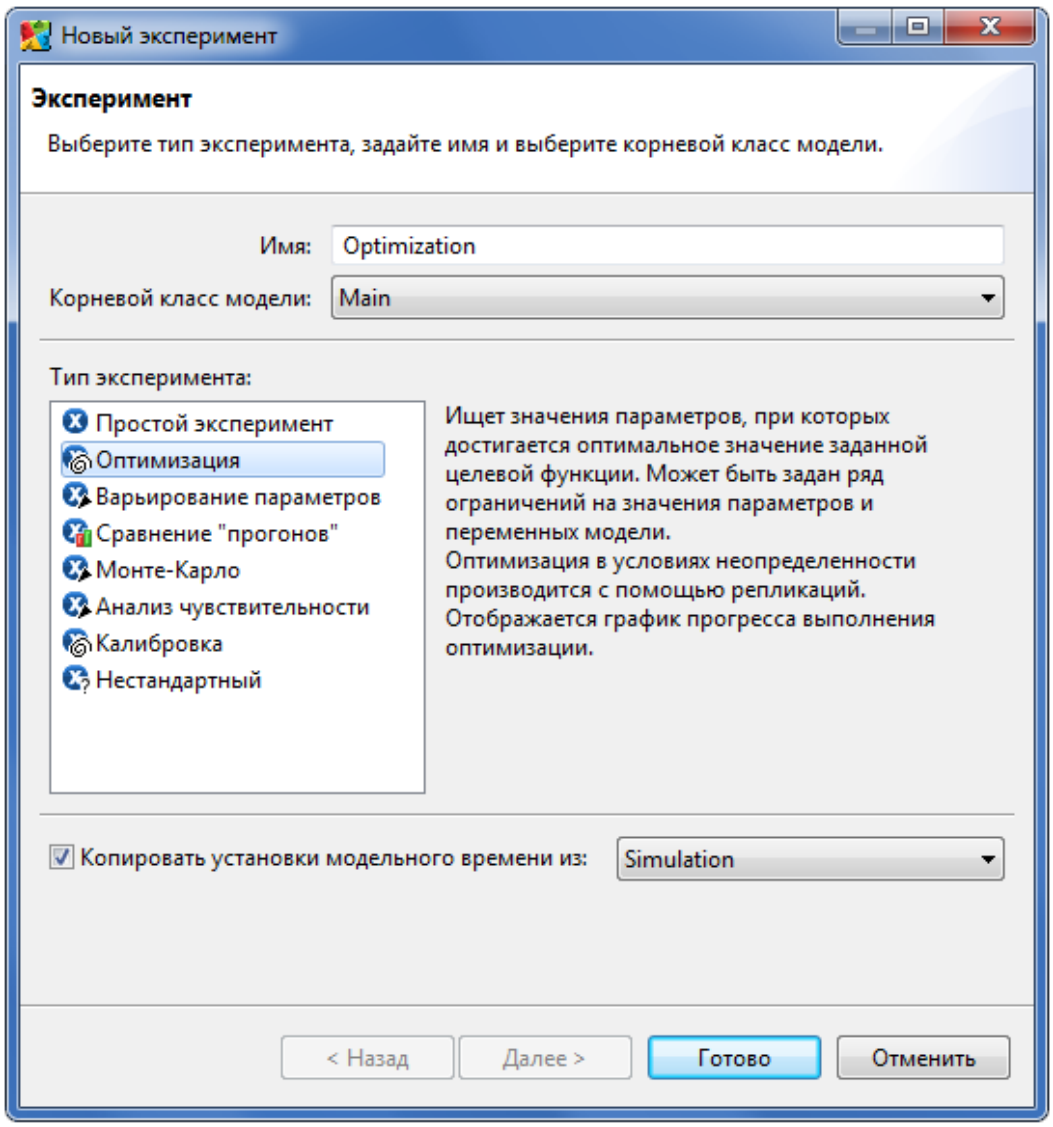
Мы же хотим найти оптимальную рыночную стратегию для достижения требуемого количества потребителей к определенному моменту времени при минимальных затратах на рекламу.

Мы можем решить эту проблему, используя оптимизацию, при которой выбранные параметры модели будут систематически изменяться для минимизации или максимизации значения целевого функционала.

Задание 1. Создайте оптимизационный эксперимент

В панели **Проекты** щелкните правой кнопкой мыши по элементу модели и выберите **Создать|Эксперимент** из контекстного меню. Откроется диалоговое окно **Новый эксперимент**.

Выберите Оптимизация из списка Тип эксперимента и нажмите Готово.



Вы увидите, что в модели будет создан еще один эксперимент, а в графическом редакторе будет открыта его диаграмма.

Мы будем оптимизировать значения параметров MonthlyExpenditures и SwitchTime. Во время оптимизации, значения параметров модели будут систематически изменяться, чтобы найти наименьшее значение переменной TotalExpenditures, выбранной в качестве целевого функционала, при котором достигается насыщение рынка к заданному моменту времени.

Задание 2. Задайте оптимизационные параметры

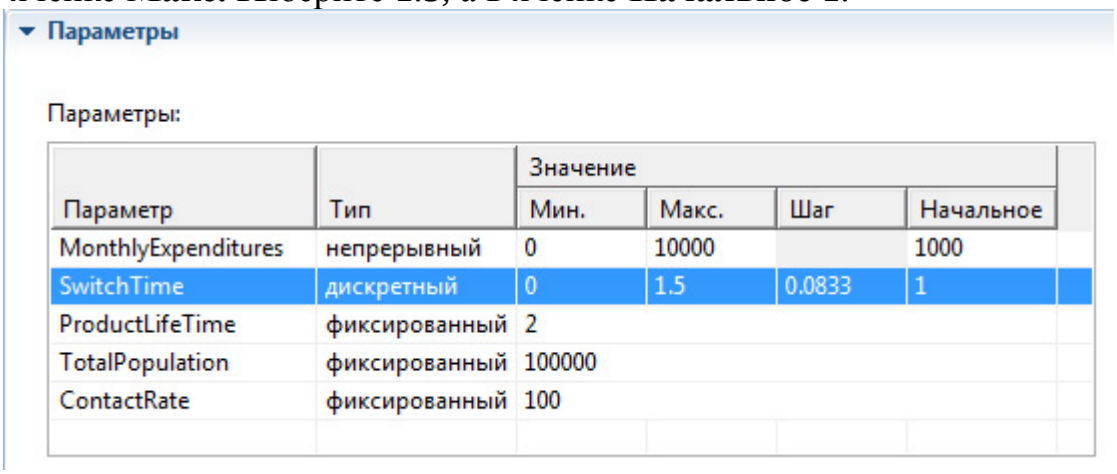
Выделите оптимизационный эксперимент в панели **Проекты** и перейдите на страницу свойств **Основные**.

Все параметры корневого активного объекта эксперимента перечислены в таблице **Параметры**, расположенной на странице основных свойств эксперимента. По умолчанию все они сделаны *фиксированными*, т.е. они не участвуют в оптимизационном процессе (их значения не меняются). Чтобы включить параметр в процесс оп-

тимизации (то есть, разрешить варьирование его значения для поиска наилучшего значения функционала), Вам нужно выбрать в ячейке **Тип** соответствующих нужным параметрам строках таблицы другую опцию вместо *фиксированный*.

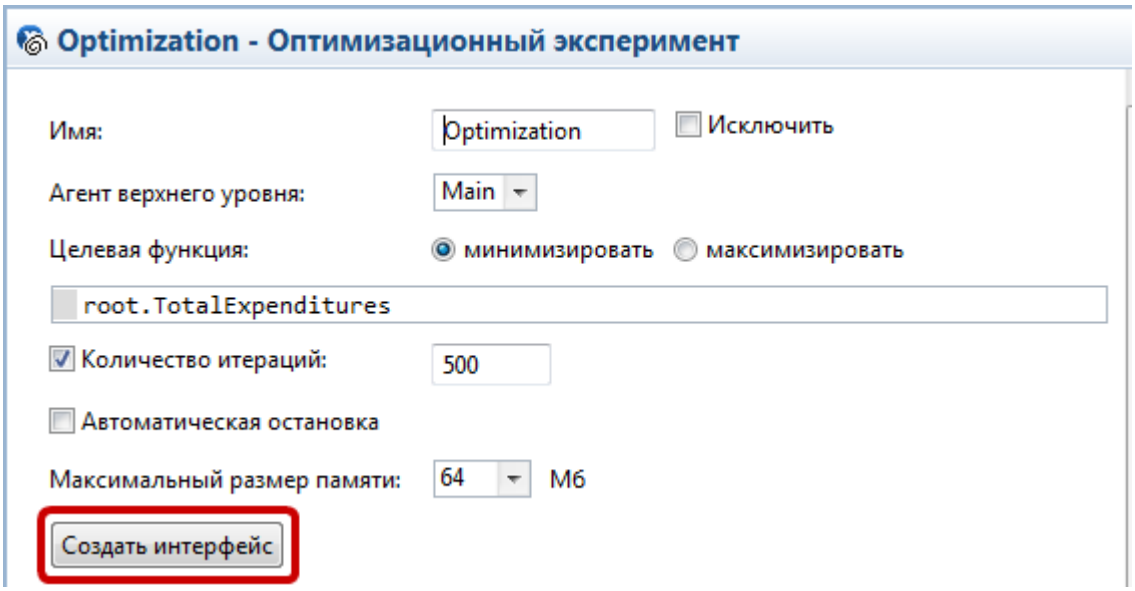
Давайте вначале сконфигурируем параметр MonthlyExpenditures. Выберите **непрерывный** в ячейке **Тип** строки **MonthlyExpenditures**. Задайте максимально возможное значение параметра в ячейке **Макс.** равным 10000, а **Начальное** значение равным 1000. Таким образом мы говорим оптимизатору, что параметр может принимать любые вещественные значения в интервале от 0 до 10000, а начнет оптимизатор процесс оптимизации со значения 1000.

Теперь сделаем то же для параметра SwitchTime. Выберите **дискретный** в ячейке **Тип**, поскольку мы хотим, чтобы этот параметр принимал только значения, соответствующие определенным временным промежуткам: один месяц, два месяца и т.д. Задайте 0.0833 в ячейке **Шаг**. Это значение соответствует одному месяцу в нашей модели, так как 1 соответствует одному году, то один месяц равен 1.0/12.0 = 0.0833. В ячейке **Макс.** выберите 1.5, а в ячейке **Начальное** 1.

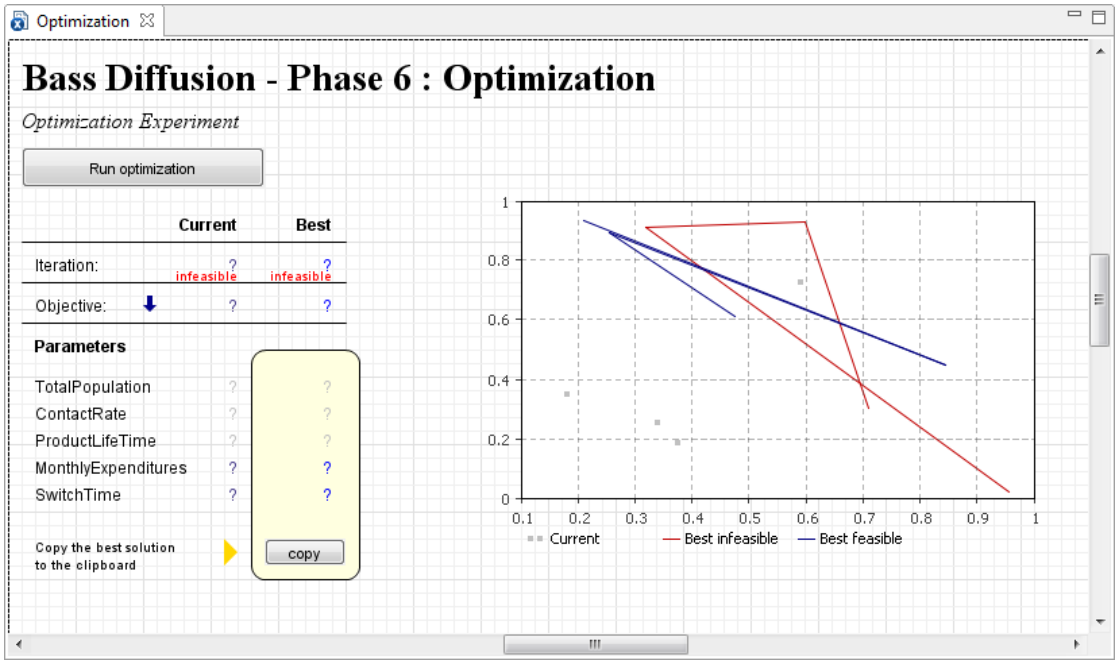


Задание 3. Создайте интерфейс эксперимента

Щелкните по кнопке **Создать интерфейс** на странице основных свойств эксперимента.



Тем самым мы создадим заданный по умолчанию интерфейс для эксперимента (см. рисунок ниже) - набор элементов управления для отображения результатов процесса оптимизации по ходу его выполнения.



Обратите внимание, что создание интерфейса удаляет все содержимое диаграммы оптимизационного эксперимента, поэтому мы рекомендуем вначале создать предлагаемый по умолчанию интерфейс, а уже потом изменять его.

Задание 4. Задайте функционал оптимизации

Мы хотим минимизировать деньги, затраченные на рекламу продукта. На странице основных свойств эксперимента введите root.TotalExpenditures в поле **Целевая функция**. Здесь мы обращаемся к корневому активному объекту эксперимента как к переменной root.

Оставьте выбранной опцию **минимизировать**.

Задание 5. Сконфигурируйте оптимизацию

На странице основных свойств эксперимента задайте максимальное количество "прогонов" модели, которое будет произведено оптимизатором. Введите 500 в поле **Количество итераций**.

Чтобы процесс оптимизации успешно выполнялся, нужно убедиться в том, что он будет заканчиваться. По умолчанию моделирование не заканчивается, поэтому оптимизатор не получит результат, который должен бы был получить по окончании выполнения каждого отдельного "прогона" модели. Поэтому нужно явно задать условие останова "прогона". Перейдите на страницу свойств эксперимента **Модельное время** и выберите опцию **В заданное время** из выпадающего списка **Остановить.** В поле ниже введите 1.5. Теперь "прогоны" модели будут завершаться по прошествии полутора единиц модельного времени.

Теперь давайте зададим дополнительное требование к результатам оптимизации, которое будет проверяться после выполнения каждого "прогона" модели. Мы хотим, чтобы по прошествии полутора лет модельного времени продукт приобрели 80000 человек.

Задание 6. Задайте the requirement for the optimization experiment

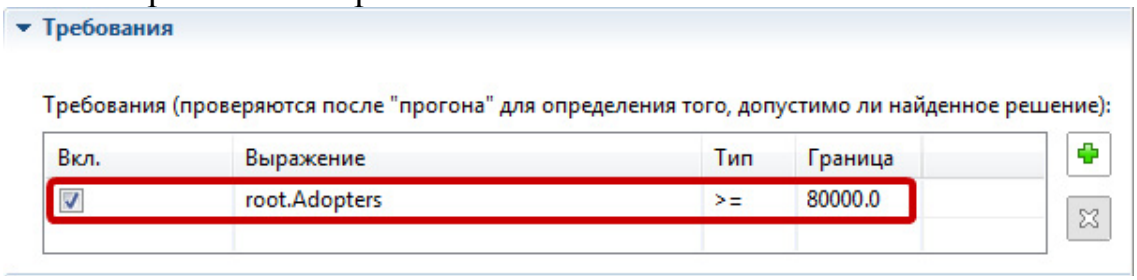
Выделите оптимизационный эксперимент в панели **Проекты** и перейдите на страницу свойств **Ограничения.** Задайте требование к результатам оптимизации в верхней строке таблицы **Требования (проверяются после "прогона" для определения того, допустимо ли найденное решение)**.

Введите root.Adopters в ячейке **Выражение**. Корневой активный объект эксперимента доступен здесь по имени root.

Выберите >= в ячейке **Тип**.

Введите 80000 в ячейке **Граница**.

Наконец, установите флажок в самом левом столбце таблицы, чтобы активировать это ограничение.



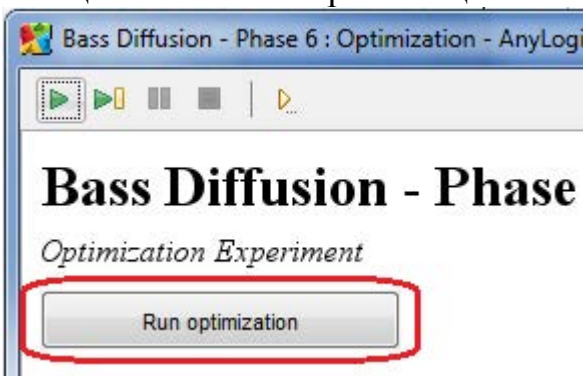
Мы закончили задание дополнительного требования к результатам оптимизации. Оно будет проверяться после каждого "прогона" модели. Если это требование не будет выполнено, то полученный в результате данного "прогона" результат будет отброшен.

Теперь модель готова к проведению оптимизации.

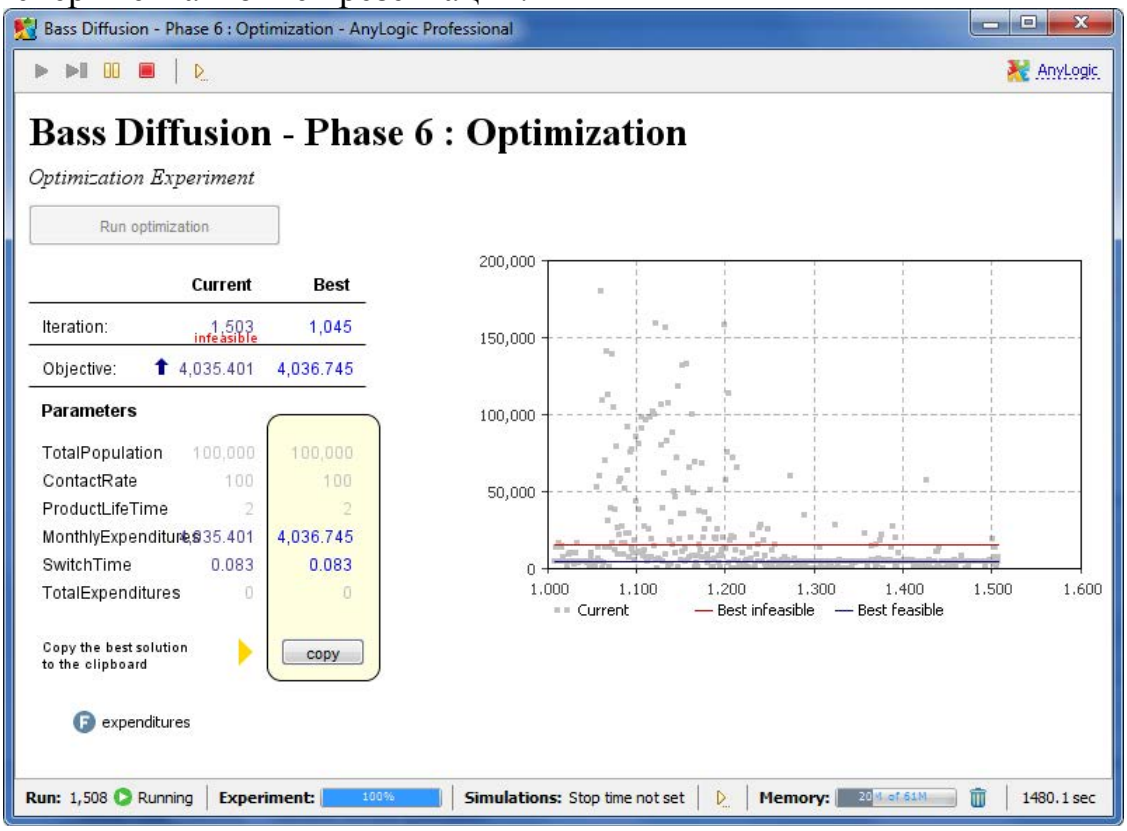
Задание 7. Запустите оптимизацию

Щелкните правой кнопкой мыши по оптимизационному эксперименту в панели **Проекты** и выберите **Запустить** из контекстного меню. Вы увидите окно презентации, отображающее презентацию запущенного эксперимента.

Запустите процесс оптимизации, щелкнув по кнопке **Запустить оптимизацию** на холсте презентации эксперимента.



AnyLogic запустит модель 500 раз, изменяя значения параметров MonthlyExpenditures и SwitchTime. Итоговая информация о результатах будет отображаться в специальных элементах управления на диаграмме эксперимента в окне презентации.



Когда процесс оптимизации модели завершится, Вы увидите, что Лучшее значение функционала равно четырем с лишним тысячам. Эксперимент в итоге выдаст оптимальные значения параметров SwitchTime

и MonthlyExpenditures, при которых было достигнуто это значение функционала.

Теперь можно обновить модель этими значениями параметров SwitchTime и MonthlyExpenditures. Сохраните полученные значения параметров в эксперименте *Simulation*, чтобы использовать в нашей модели найденную оптимальную стратегию.

Задание 8. Примените результаты оптимизации

После завершения оптимизации, щелкните по кнопке **copy** на холсте диаграммы эксперимента в окне презентации. Таким образом Вы скопируете найденные (оптимальные) значения параметров в Буфер обмена.

Закройте окно презентации и выделите эксперимент Simulation в панели **Проекты**.

Вставьте скопированные значения параметров из Буфера обмена, щелкнув по кнопке **Вставить из буфера** внизу страницы основных свойств эксперимента.

Запустите эксперимент Simulation. Теперь модель будет запущена с оптимальными значениями параметров, при которых в процессе оптимизации было получено наилучшее значение функционала. Можете проверить, что к заданному времени (1,5 года) достигается требуемое количество пользователей продукта.

Теперь мы спланировали стратегию завоевания рынка таким образом, чтобы рекламная кампания была наиболее рациональной и эффективной.

Лабораторная работа 3. Агентное моделирование

1. Цель работы

Получить практические навыки по агентному моделированию на примере агентной модели распространения продукта по Бассу

2. Краткие теоретические сведения

Пакет моделирования AnyLogic поддерживает различные подходы моделирования. В этой лабораторной работе описывается агентный подход моделирования, успешно применяемый в различных сферах деятельности, таких как экология, социология, экономика, моделирование движения и т.д. При помощи агентов моделируют рынки (агент – потенциальный покупатель), конкуренцию и цепочки поставок (агент - компания), население (агент – семья, житель города или избиратель) и многое другое. Агентные модели позволяют получить представление об общем поведении системы, исходя из предположений о поведении ее элементов, при отсутствии знания о глобальных законах – то есть в наиболее общем случае.

AnyLogic является единственным инструментом моделирования, позволяющим быстро создавать гибкие модели с агентами, взаимодействующими как друг с другом, так и со своим окружением. AnyLogic поддерживает все возможные способы задания поведения агентов – диаграммы состояний, синхронное и асинхронное планирование событий.

Эта лабораторная работа кратко ознакомит Вас с процессом создания модели в AnyLogic. Ее целью является ознакомление с интерфейсом и основными возможностями AnyLogic. Мы создадим простой и наглядный пример – модель жизненного цикла продукта, используемую для предсказания распространения новых продуктов.

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

Затем мы расширим нашу модель, добавив некоторые новые детали и продемонстрировав усовершенствованные возможности AnyLogic.

3. Задание на лабораторную работу

3.1 Выполняя последовательно шаги проектирования создать модель распространения продукта по Бассу.

3.2 Провести вычислительный эксперимент и сделать выводы.

4. Порядок выполнения работы.

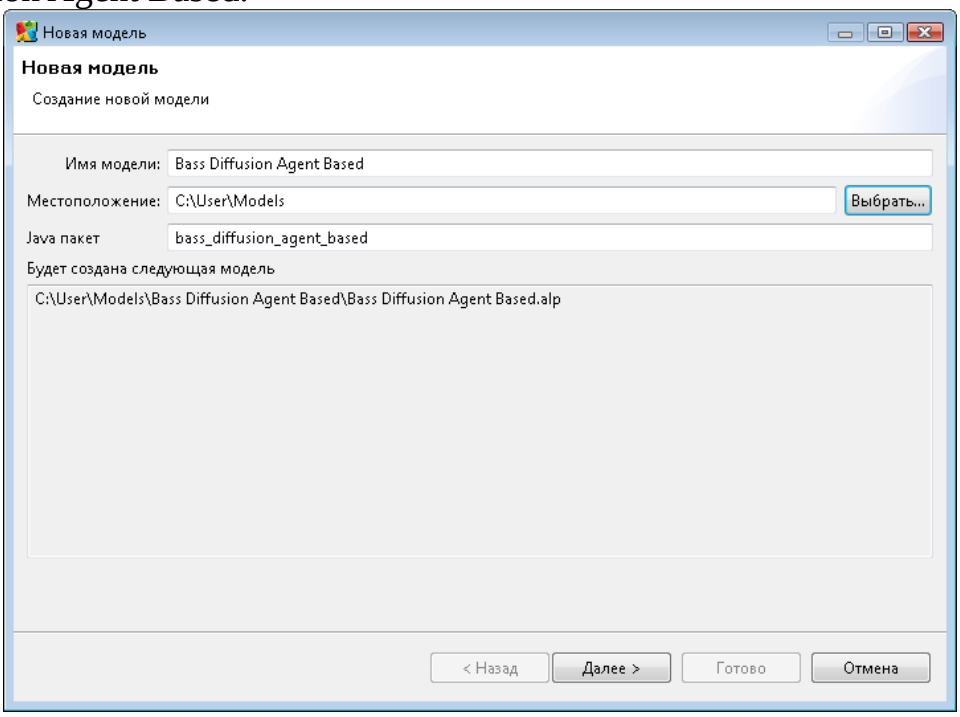
4.1 Создание простейшей агентной модели

Давайте создадим простейшую агентную модель. Начиная с версии 6.4 AnyLogic предоставляет пользователям возможность использования шаблонов моделей при создании новых моделей. Если раньше Вам приходилось всегда начинать создание модели "с чистого листа", зачастую выполняя одни и те же типовые действия для каждой новой создаваемой модели, то теперь Вы можете перепоручить выполнение первых, базовых, шагов **Мастеру создания модели**. Все, что Вам нужно - это указать, какой метод моделирования Вы будете использовать и выбрать те опции, которые Вам нужны в модели - и **Мастер** автоматически создаст простейшую модель, а Вы сможете продолжать ее разработку, лишь изменив незначительные детали.

Задание 1. Создайте новую агентную модель

Щелкните мышью по кнопке панели инструментов **Создать**  . Появится диалоговое окно **Новая модель**.

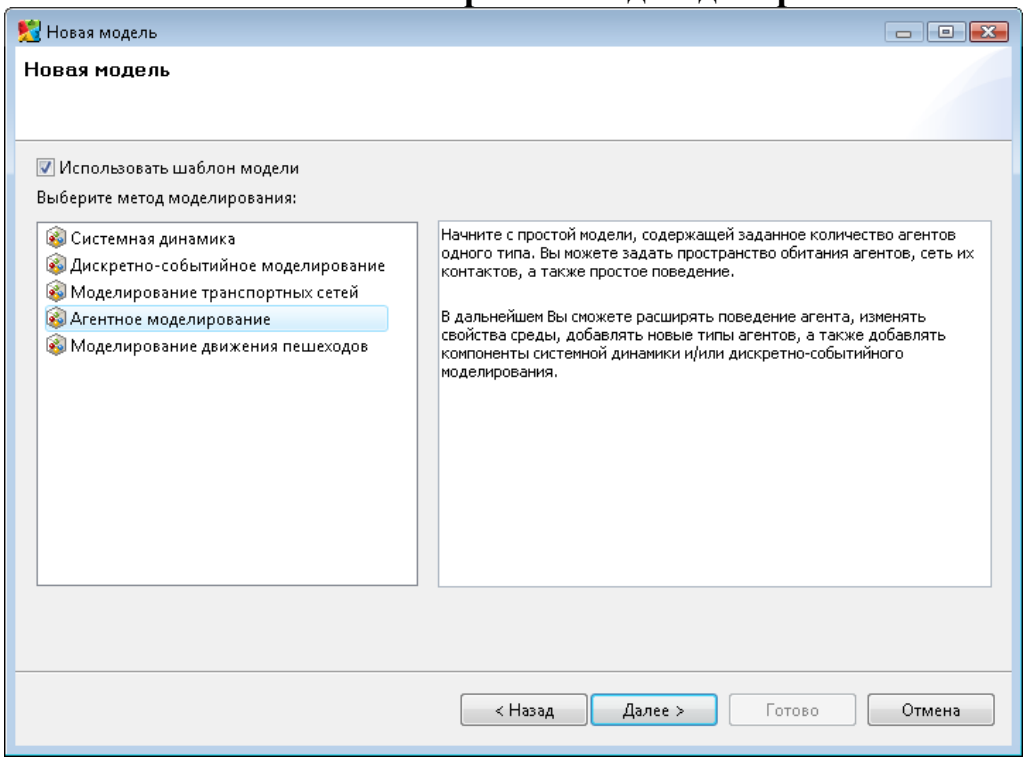
Задайте имя новой модели. В поле **Имя модели** введите Bass Diffusion Agent Based.



Выберите каталог, в котором будут сохранены файлы модели. Если Вы хотите сменить предложенный по умолчанию каталог на какой-то

другой, Вы можете ввести путь к нему в поле **Местоположение** или выбрать этот каталог с помощью диалога навигации по файловой системе, открывающегося по нажатию на кнопку **Выбрать**.

Щелкните мышью по кнопке **Далее**. Откроется вторая страница **Мастера создания модели**. Здесь Вам будет предложено выбрать шаблон модели, на базе которого Вы будете разрабатывать Вашу модель. Поскольку мы хотим создать новую агентную модель, установите флажок **Использовать шаблон модели** и выберите **Агентная модель** в расположенном ниже списке **Выберите метод моделирования**.



Щелкните мышью по кнопке **Далее**. Откроется следующая страница **Мастера создания модели**. Поскольку первым шагом при создании агентной модели всегда является создание агентов, то здесь Вам как раз предлагается задать имя типа агентов и количество агентов, которое будет изначально создано в нашей модели. Задайте в качестве имени класса *Person*. и введите в поле **Начальное количество агентов** 500. Автоматически в нашей модели будет создано 500 агентов (то есть, экземпляров класса *Person*, каждый из которых будет представлять отдельного агента).

Щелкните мышью по кнопке **Далее**. Откроется следующая страница **Мастера создания модели**. Здесь Вам будет предложено задать свойства пространства, в котором будут обитать агенты, а также выбрать фигуру анимации агента.

Установите флажок **Добавить пространство** и выберите ниже тип этого пространства: **Непрерывное**. Здесь же Вы можете задать размерности этого пространства: давайте введем в поле **Ширина** 600, а в поле **Высота** 350. Тем самым, в результате наши агенты будут располагаться каким-то образом в пределах непрерывного пространства, отображаемого на презентации моделей областью размером 600*350 пикселей.

Не меняйте значения, выбранные в выпадающих списках **Начальное расположение** и **Анимация**: пусть агенты изначально расставляются по пространству случайным образом, а анимируются с помощью фигурки человечка.

Щелкните мышью по кнопке **Далее**. Откроется следующая страница **Мастера создания модели**. Здесь Вам будет выбрать, хотите ли Вы, чтобы была задана сеть взаимосвязей агентов, и если да, то каким образом должны устанавливаться связи между агентами.

Установите флажок **Использовать сеть** и оставьте выбранной опцию **Случайное**.

Щелкните мышью по кнопке **Далее**. Откроется последняя страница **Мастера создания модели**. Установите на ней флажок **Добавить простое поведение**. Тем самым, у нашего агента будет создана диаграмма состояний (Вы сможете увидеть ее в панели предварительного просмотра модели **Мастера создания модели**). В свое время мы объясним, как задавать поведение агента с помощью диаграммы состояний.

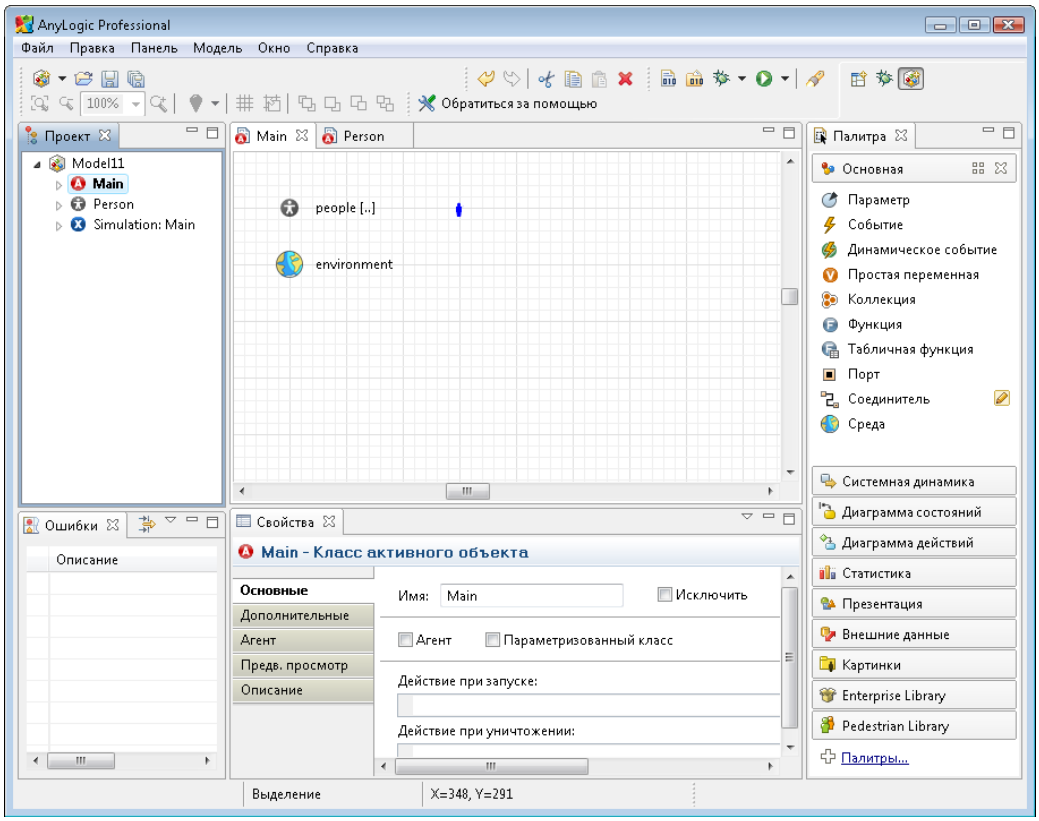
Мы закончили конфигурирование шаблона создаваемой модели. Щелкните мышью по кнопке **Готово**, чтобы закончить процесс создания модели.

Вы создали новую модель. Если Вы еще не знакомы с пользовательским интерфейсом AnyLogic, то давайте уделим пару минут основным его компонентам. В левой части рабочей области находится панель **Проекты**. Панель **Проекты** обеспечивает легкую навигацию по элементам моделей, открытым в текущий момент времени. Поскольку модель организована иерархически, то она отображается в виде дерева: сама модель образует верхний уровень дерева; эксперименты, типы агентов и Java классы образуют следующий уровень; элементы, входящие в состав агентов, вложены в соответствующую подветвь типа агента и т.д.

В правой части рабочей области отображается панель **Палитра**, а внизу - панель **Свойства**. Панель **Палитра** содержит разделенные по категориям элементы, которые могут быть добавлены на графическую

диаграмму типа агентов или эксперимента. Панель **Свойства** используется для просмотра и изменения свойств выбранного в данный момент элемента (или элементов) модели.

В центре рабочей области AnyLogic Вы увидите графический редактор. В графическом редакторе автоматически откроется диаграмма класса *Main*.



Наша модель будет содержать созданные **Мастером создания модели** классы активных объектов *Main* и *Person*. Агенты являются основными строительными блоками модели AnyLogic. Агенты могут моделировать любые объекты реального мира: машины, людей, станки, здания, аппаратное обеспечение и т.д. В нашем случае агент типа *Person* будет моделировать агентов (людей). Этот класс агента был автоматически объявлен агентом (тем самым он получил доступ к специальной функциональности агента). В панели **Проекты** такой класс отображается значком .

4.2 Моделирование продаж под влиянием рекламы

Мы (не без помощи **Мастера создания модели**) уже создали простейшую модель. Теперь давайте путем нескольких изменений приведем ее к постановке нашей простейшей задачи - пусть наша модель моделирует процесс приобретения нового продукта, но пока только под влиянием рекламной кампании, проводимой с целью выведения нового продукта на сложившийся рынок.

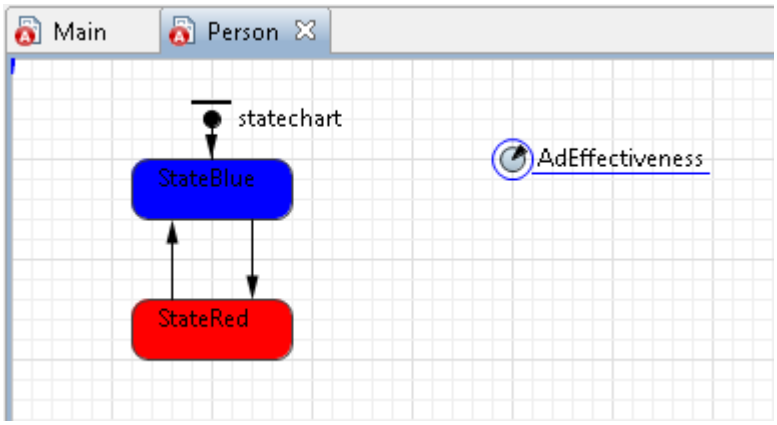
В этой модели интенсивность рекламы и вероятность того, что продукт будет приобретен под ее влиянием, полагаются постоянными. Поэтому мы зададим эффективность рекламы константой. Эффективность рекламы определяет, какая доля людей купит продукт вследствие ее влияния.

Характеристики модели задаются с помощью параметров. Мы зададим параметры в классе *Person*, потому что наши агенты задаются экземплярами именно этого класса. Задав значение параметра в классе, мы задаем его для всех агентов одновременно. Но при необходимости Вы сможете задать характеристики индивидуально для каждого агента, поскольку AnyLogic позволяет задавать различные значения параметров для разных элементов одного и того же класса.

Задание 1. Задайте подверженность человека влиянию рекламы

Откройте диаграмму класса *Person*, сделав двойной щелчок мышью по элементу *Person* в панели **Проекты**.

Перетащите элемент **Параметр** из палитры **Основная** на диаграмму класса:

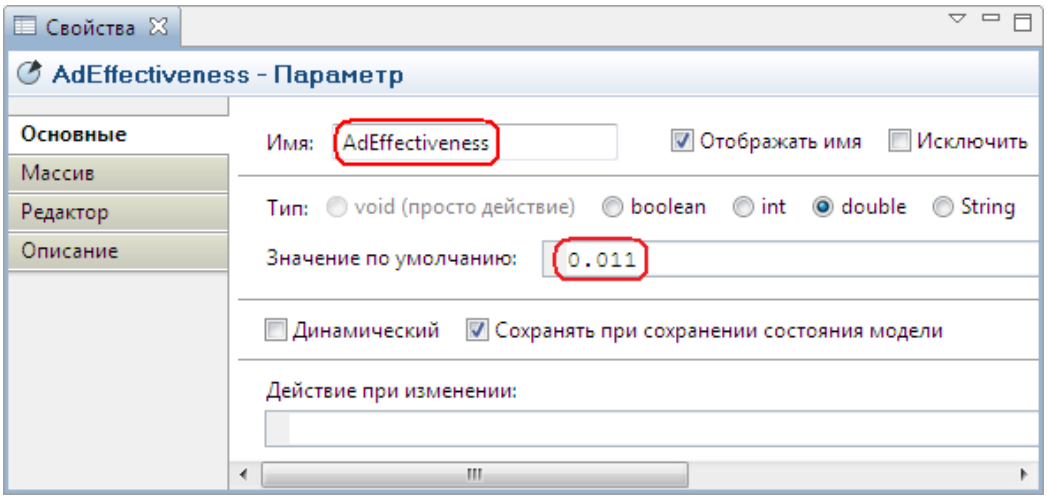


Когда Вы поместите элемент на диаграмму класса, этот элемент будет считаться выбранным, и Вы сможете изменить свойства элемента в расположенной в нижней части рабочей области панели **Свойства**. В дальнейшем для изменения свойств элемента нужно будет вначале щелчком мыши выделить его в графическом редакторе или в дереве элементов модели, отображаемом в панели **Проекты**.

Перейдите на страницу **Основные** панели **Свойства**, чтобы изменить свойства созданного параметра.

Измените имя параметра. Введите AdEffectiveness в поле **Имя**.

В поле **Значение по умолчанию** введите 0.011.



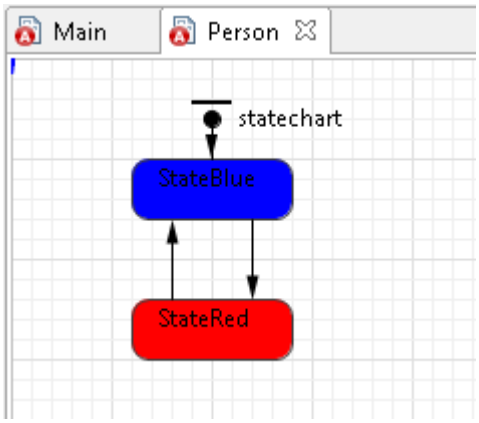
Вы можете задать краткое описание параметра на странице **Описание** панели **Свойства** (чтобы переключиться на другую страницу, щелкните мышью по вкладке с ее названием в левой части панели). Введите текст, который поможет объяснить смысл параметра тем, кто не знаком с моделью.

Поведение агента обычно описывается визуально в классе этого агента (в нашей модели это класс Person) с помощью *диаграммы состояний*.

Мастер создания моделей уже создал простейшую диаграмму состояний из двух состояний, между которыми существует два разнонаправленных перехода.

Задание 2. Измените диаграмму состояния

Откройте диаграмму класса *Person*, сделав двойной щелчок мышью по элементу *Person* в панели **Проекты**. На диаграмме класса Вы увидите следующую диаграмму состояний:



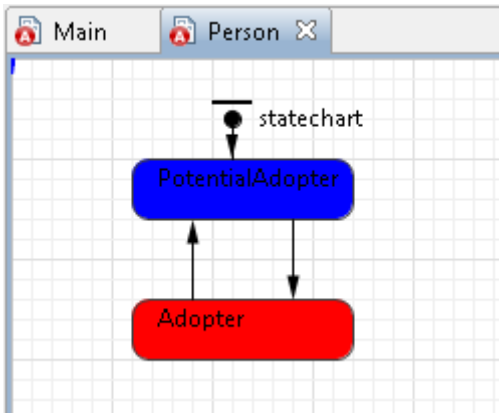
Измените имя верхнего состояния на *PotentialAdopter* (поле **Имя** на странице свойств перехода) Это начальное состояние, о чем свидетельствует элемент *Начало диаграммы состояний*, направленный в это

состояние. Если диаграмма состояний будет находиться в этом состоянии, то это будет означать, что этот человек еще не купил продукт.

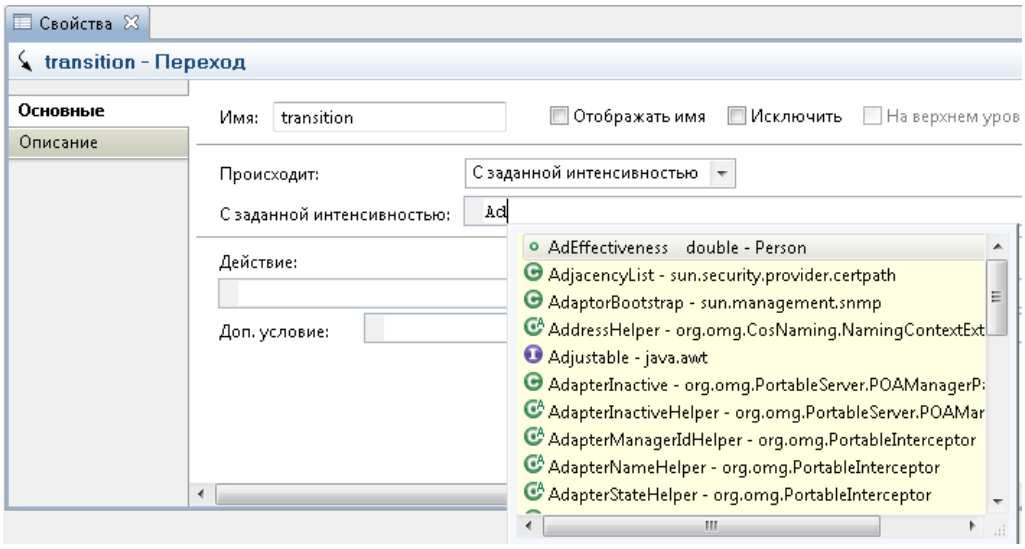
Назовите его *Adopter*. Если это состояние диаграммы будет активным, это будет означать, что этот человек уже купил продукт.

В нашей модели состояние *Adopter* должно становиться активным в момент приобретения агентом продукта. Процесс приобретения продукта этим человеком моделирует *переход*, ведущий из верхнего состояния в нижнее. Нам нужно изменить его свойства, чтобы он срабатывал в нужный нам момент времени.

Время, через которое человек купит продукт, экспоненциально зависит от эффективности рекламы продукта. Поскольку время, необходимое человеку, чтобы принять решение о покупке продукта экспоненциально зависит от подверженности этого человека влиянию рекламы, то выберите из выпадающего списка **Происходит** *С заданной интенсивностью* и введите в поле свойства **Интенсивность** этого перехода имя созданного нами только что параметра AdEffectiveness.

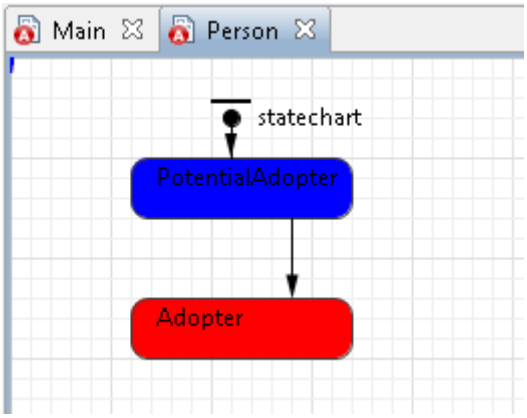


Введите AdEffectiveness в расположенном ниже поле **Интенсивность**. Чтобы не печатать полностью имена функций и переменных в формулах, можэете пользоваться **Мастером подстановки кода**. Чтобы открыть **Мастер**, щелкните мышью в том месте поля (в нашем случае - поля **Интенсивность**, куда Вы хотите поместить имя, а затем нажмите Ctrl+пробел (при работе на Mac OS: Alt+пробел). Появится окно **Мастера подстановки кода**, перечисляющего переменные модели и функции, доступные в текущем конексте. Прокрутите список к имени, которое Вы хотите вставить, или введите первые буквы имени, пока оно не будет выделено в списке. Двойным щелчком мыши по имени добавьте его в поле формулы.



Удалите переход, ведущий из нижнего состояния в верхнее, поскольку мы пока создаем простейшую модель, в которой человек, однажды приобревший продукт, навсегда остается его потребителем, и соответственно перехода из состояния *Adopter* в состояние *PotentialAdopter* пока что быть не должно.

Чтобы удалить переход, выделите его на диаграмме и нажмите Del.

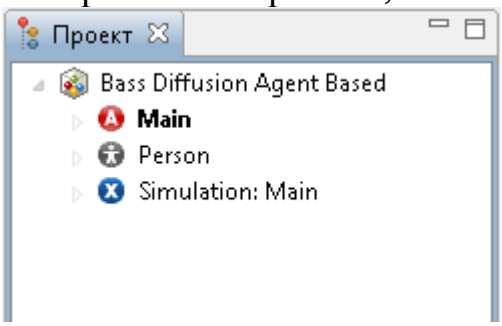


4.3 Настройка запуска модели

Вы можете сконфигурировать выполнение модели в соответствии с Вашими требованиями. Модель выполняется в соответствии с набором установок, задаваемым специальным элементом модели - *экспериментом*. Вы можете создать несколько экспериментов с различными установками и изменять рабочую конфигурацию модели, просто запуская тот или иной эксперимент модели.

В панели **Проекты** эксперименты отображаются в нижней части дерева модели. Один эксперимент, названный *Simulation*, создается по умолчанию. Это простой эксперимент, позволяющий запускать модель с

заданными значениями параметров, поддерживающий режимы виртуального и реального времени, анимацию и отладку модели.



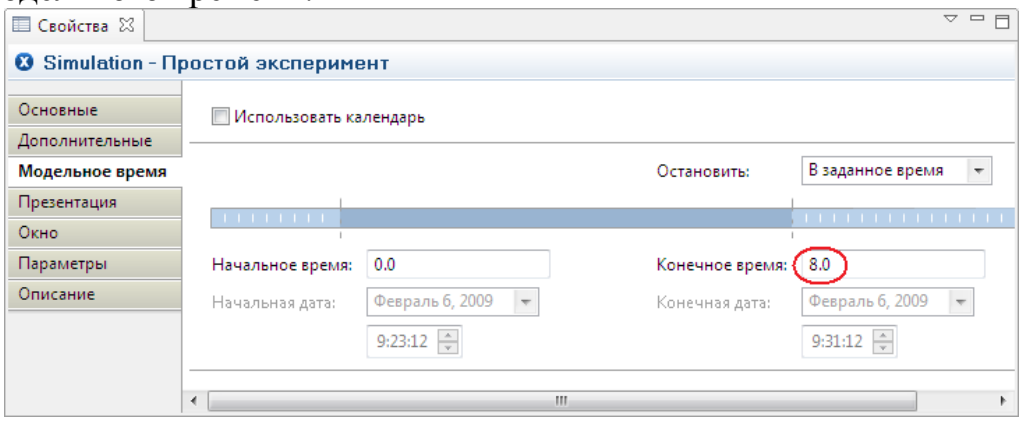
Существуют также и другие типы экспериментов (оптимизационный эксперимент, эксперимент для оценки рисков, эксперимент для варьирования параметров), которые используются в тех случаях, когда параметры модели играют существенную роль, и требуется проанализировать, как они влияют на поведение модели, или когда нужно найти оптимальные значения параметров модели.

Если мы сейчас запустим модель, то она будет работать бесконечно. Поскольку мы хотим наблюдать поведение модели только тогда, когда происходит процесс распространения продукта, нам нужно остановить модель, когда система придет в точку равновесия. Поскольку под единицей модельного времени мы будем понимать один год, а процесс распространения продукта в этой модели длится примерно 8 лет, то нам нужно будет остановить модель после 8 единиц модельного времени.

Задание 1. Задайте остановку модели по прошествии 8 единиц модельного времени

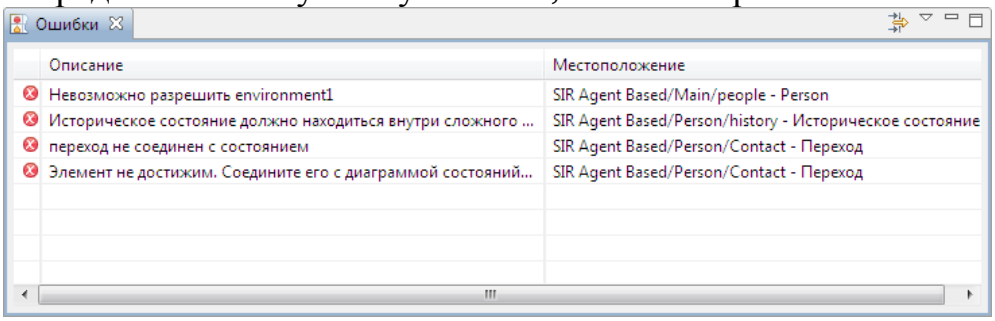
В панели **Проекты**, выделите эксперимент Simulation:Main щелчком мыши.

На странице **Модельное время** панели **Свойства**, выберите **В заданное время** из выпадающего списка **Остановить**. В расположенном ниже поле введите 8. Модель остановится после того, как истекут 8 единиц модельного времени.



4.4 Запуск модели

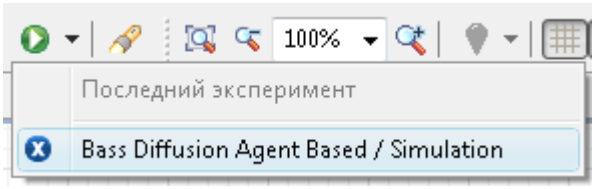
Постройте Вашу модель с помощью кнопки панели инструментов **Построить модель** (при этом в рабочей области AnyLogic должен быть выбран какой-то элемент именно этой модели). Если в модели есть какие-нибудь ошибки, то построение не будет завершено, и в панель **Ошибки** будет выведена информация об ошибках,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



После того, как Вы исправите все ошибки и успешно построите Вашу модель, Вы можете ее запустить.

Задание 1. Запустите модель

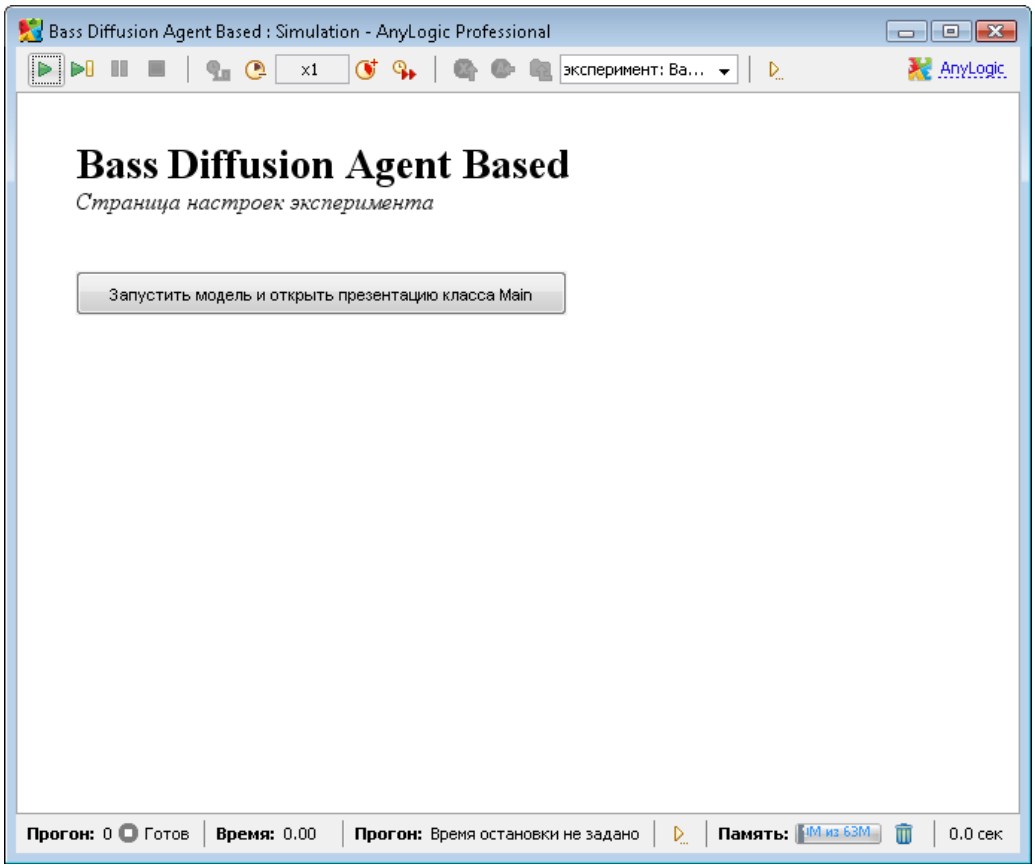
Щелкните мышью по кнопке панели инструментов **Запустить** и выберите из открывшегося списка эксперимент, который Вы хотите запустить. Эксперимент этой модели будет называться *Bass Diffusion Agent Based/Simulation*.



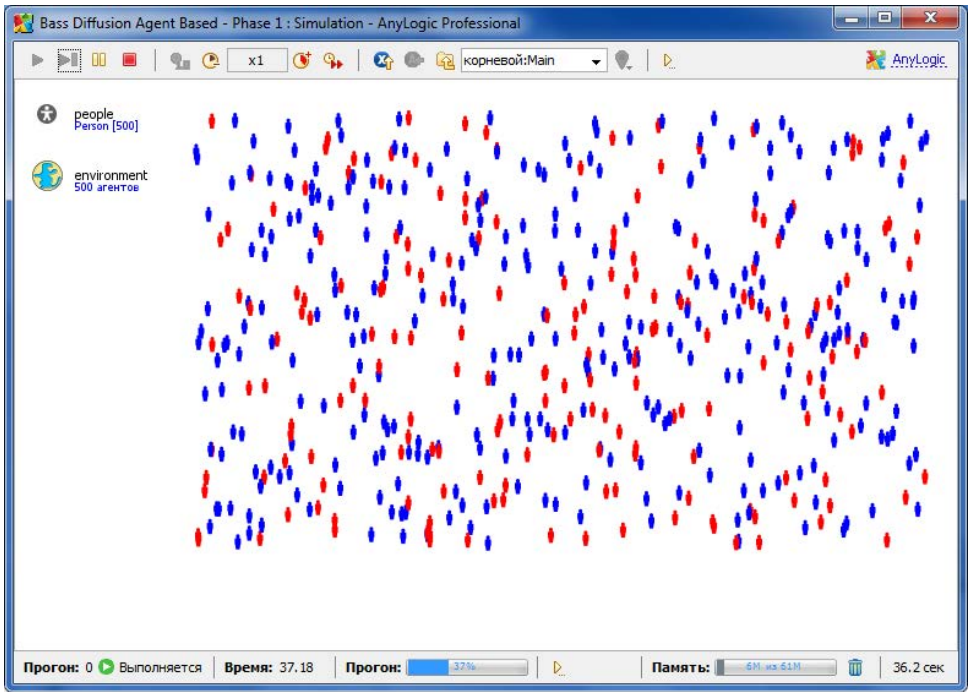
В дальнейшем по нажатию на кнопку **Запустить** (или по нажатию F5) будет запускаться тот эксперимент, который запускался Вами в последний раз. Чтобы выбрать какой-то другой эксперимент, Вам будет нужно щелкнуть мышью по стрелке, находящейся в правой части кнопки **Запустить** и выбрать нужный Вам эксперимент из открывшегося списка (или щелкнуть правой кнопкой мыши по этому эксперименту в панели **Проекты** и выбрать **Запустить** из контекстного меню).

Запустив модель, Вы увидите окно презентации этой модели. В нем будет отображена презентация запущенного эксперимента.

AnyLogic автоматически помещает на презентацию каждого простого эксперимента заголовок и кнопку, позволяющую запустить модель и перейти на презентацию, нарисованную Вами для агента верхнего уровня эксперимента (Main).



Щелкните по этой кнопке. Тем самым Вы запустите модель и перейдете к презентации корневого класса активного объекта запущенного эксперимента. На презентации Вы увидите моделируемых нами агентов. Каждый агент отображается своей фигуркой, которая меняет свой цвет в зависимости от того, приобрел ли данный агент рассматриваемый нами продукт или нет. Линиями на презентации будут соединены те агенты, между которыми существуют связи (в данный момент эти связи генерируются случайным образом).



При желании Вы можете изменить скорость выполнения модели с помощью кнопок панели управления окна презентации **Замедлить** и **Ускорить**.

4.5 Подсчет потребителей продукта

Главная задача модели распространения продукта – изучение того, как быстро люди покупают новый продукт. Поэтому сейчас мы добавим возможность отслеживания того, сколько людей уже купило продукт, а сколько – еще нет. Мы будем подсчитывать число потребителей и потенциальных потребителей продукта с помощью специальных функций сбора статистики по агентам.

Задание 1. Создайте функции сбора статистики для подсчета потенциальных потребителей продукта

Откройте диаграмму класса *Main*, сделав двойной щелчок мышью по элементу *Main* в панели **Проекты**.

Выделите на диаграмме вложенный объект *people*.

Перейдите на страницу **Статистика** панели **Свойства**.

Щелкните мышью по кнопке **Добавить ф-ю сбора статистики**. Откроется секция свойств для задания свойств новой функции сбора статистики по элементам этого реплицированного объекта (people).

Введите *potentialAdopters* в поле **Имя**. Это будет именем нашей функции.

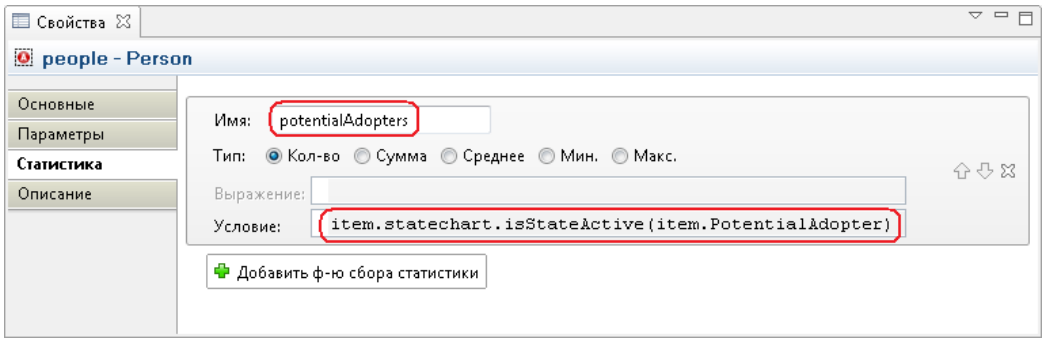
Оставьте выбранный по умолчанию **Тип** функции: **Кол-во**.

Задайте **Условие**:

item.statechart.isStateActive(item.PotentialAdopter)

Эта функция будет вести подсчет количества агентов, для которых выполняется заданное условие, т.е. тех агентов, которые находятся в текущий момент времени в состоянии *PotentialAdopter* (являются потенциальными потребителями продукта).

Здесь item - это агент (элемент реплицированного объекта people).



Задание 2. Создайте функции сбора статистики для подсчета потребителей продукта

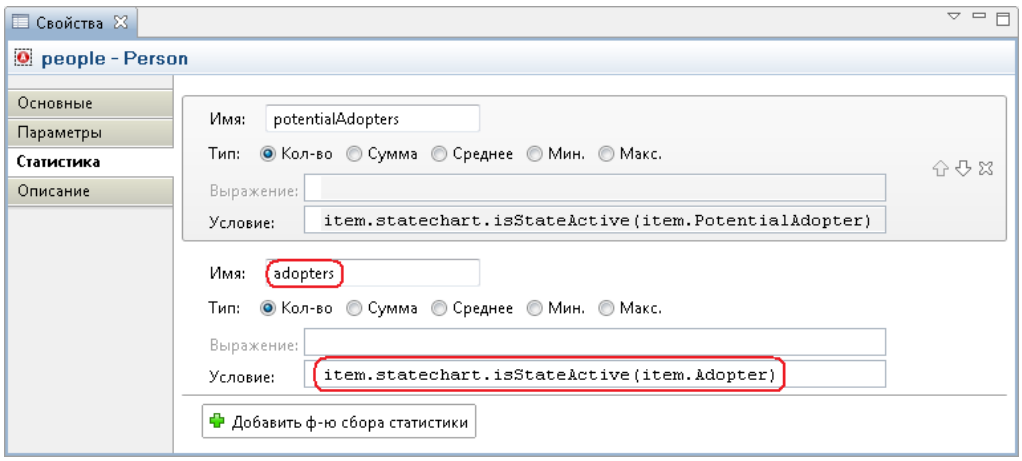
Аналогично создайте еще одну функцию сбора статистики.

Назовите ее *adopters*.

Оставьте выбранный по умолчанию **Тип** функции: **Кол-во**.

Задайте **Условие**: item.statechart.isStateActive(item.Adopter)

Эта функция будет вести подсчет количества агентов, которые находятся в состоянии *Adopter* (то есть, уже приобрели продукт).



4.6 Добавление диаграммы

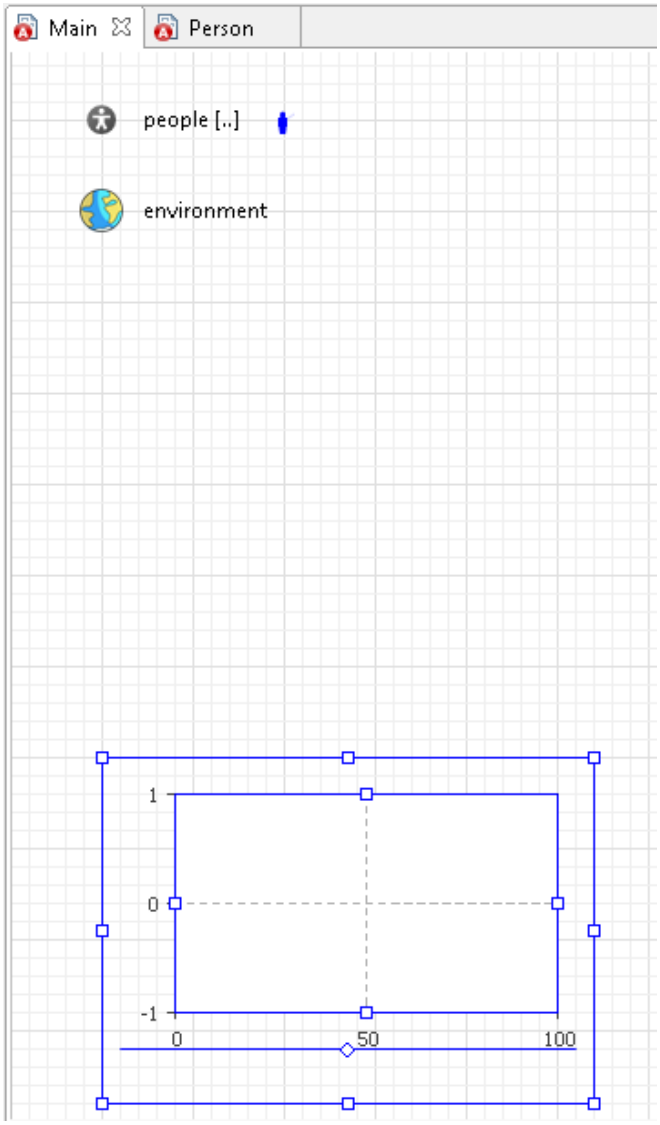
С помощью *диаграмм* Вы можете понаблюдать за динамикой моделируемого процесса. Сейчас мы создадим диаграмму, отображающую динамику изменения числа потребителей и потенциальных потребителей продукта.

Задание 1. Добавьте временной график, отображающий динамику изменения численностей потребителей и потенциальных потребителей продукта

Откройте диаграмму класса *Main*, сделав двойной щелчок мышью по элементу *Main* в панели **Проекты**.

Перетащите элемент **Временной график** из палитры **Статистика** на диаграмму класса.

Измените размер графика так, как показано на приведенном ниже рисунке:



Перейдите на страницу **Основные** панели **Свойства**.

Укажите, что именно Вы хотите отображать на графике - то есть, задайте *элементы данных* этого графика.

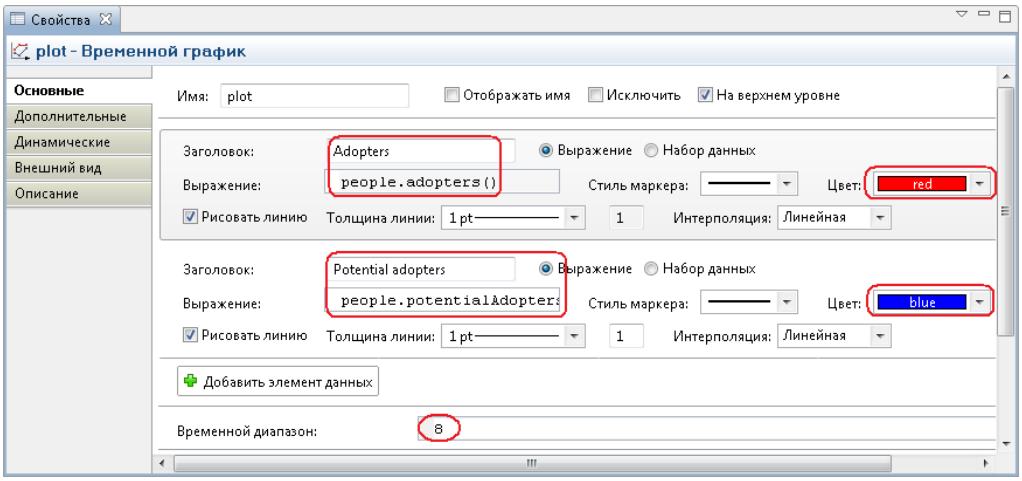
Чтобы добавить элемент данных, щелкните мышью по кнопке **Добавить элемент данных** и в открывшейся секции свойств задайте свойства этого элемента.

Введите people.potentialAdopters() в поле **Выражение**. При этом Вы можете воспользоваться помощником подстановки кода, вызываемому нажатием Ctrl+пробел при работе на Windows и Alt+пробел на Mac OS. Здесь мы задаем выражение, результат вычисления которого будет отображаться на нашем графике - в нашем случае мы помещаем здесь вызов ранее созданной нами функции сбора статистики по агентам, возвращающей текущее количесвто потенциальных потребителей продукта.

Введите *Potential adopters* в поле **Заголовок**. Эта строка будет отображаться в легенде диаграммы для данного элемента данных.

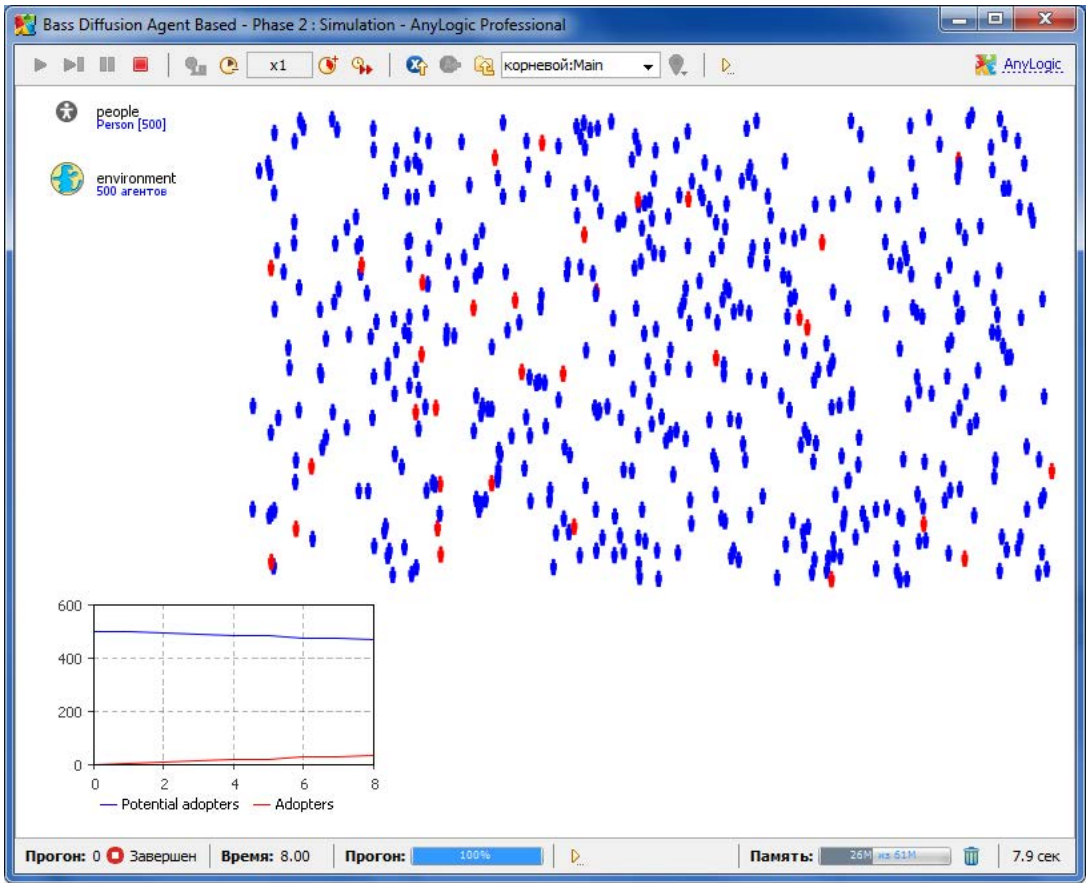
Аналогично добавьте еще один элемент данных. Пусть он отображает количество потребителей продукта, возвращаемое другой нашей статистической функцией: people.adopters(). Задайте *Adopters* в качестве заголовка этого элемента данных и измените свойства внешнего вида, как и в предыдущем случае.

Задайте **Временной диапазон**: 8. Тем самым мы задаем диапазон временной оси графика. Поскольку в текущей модели все время моделирования равно восьми единицам модельногоо времени, то мы можем ограничить и временную ось аналогичным значением.



Теперь наша диаграмма успешно добавлена и сконфигурирована на отображение численностей интересующих нас групп людей.

Запустите модель. С помощью диаграммы Вы можете понаблюдать за динамикой моделируемого процесса. Вы увидите, что под влиянием рекламы каждую единицу времени постоянная доля от общей численности потенциальных потребителей продукта приобретает изучаемый нами продукт.



4.7 Учет влияния общения людей

В текущей модели люди приобретают продукт только под влиянием рекламы. На самом деле, рекламный эффект играет значительную роль только в момент выпуска продукта на рынок. В дальнейшем все большую роль будет играть общение людей с теми своими знакомыми, которые этот продукт уже приобрели. В основном люди приобретают новые продукты именно под влиянием убеждения своих знакомых; этот процесс чем-то похож на распространение эпидемии.

Чтобы учесть влияние общения людей, мы должны внести в нашу модель небольшие изменения.

Теперь нам нужно задать еще пару новых параметров.

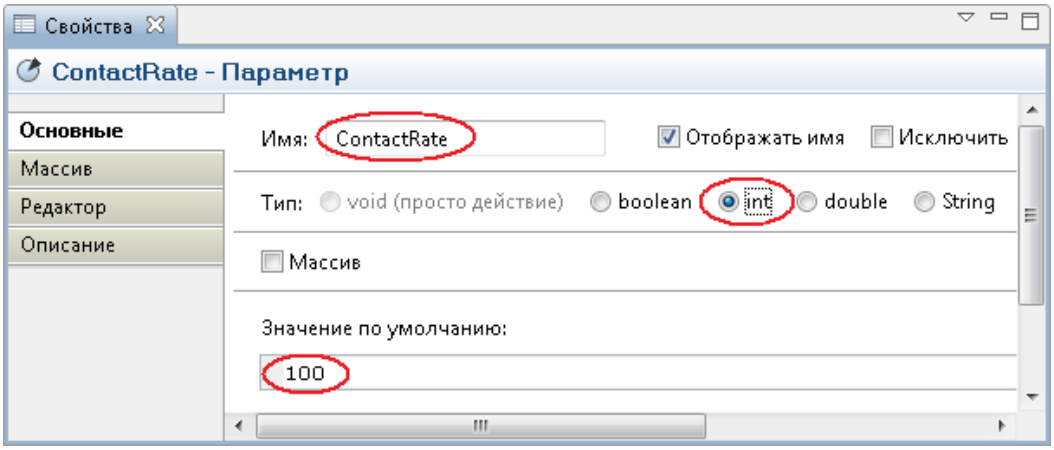
Задание 1. Задайте среднегодовое количество встреч человека

Откройте диаграмму класса *Person*, сделав двойной щелчок мышью по элементу *Person* в панели **Проекты**.

Создайте новый параметр.

Назовите его *ContactRate*.

Предположим, что человек в среднем встречается со 100 людьми в год. Выберите **Тип** *int* и введите в поле **Значение по умолчанию** 100.

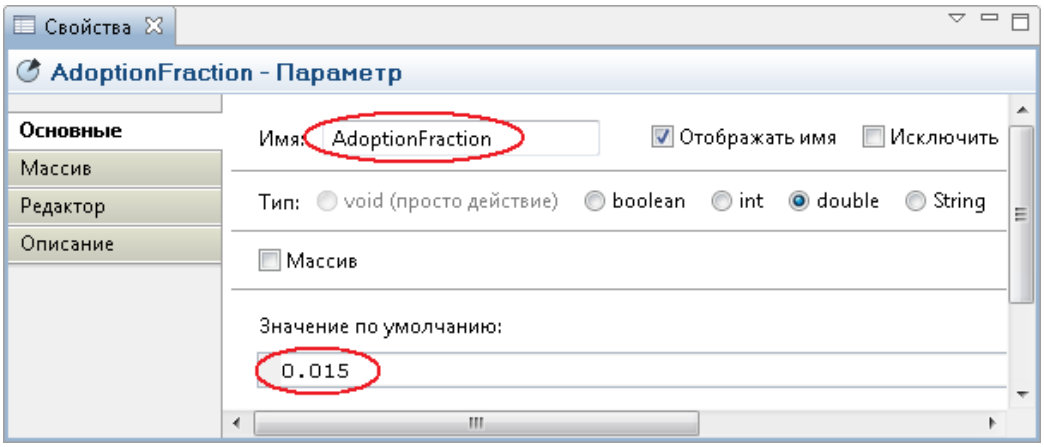


Задание 2. Задайте параметром убедительность человека

Добавьте еще один параметр, задающий силу убеждения человека - долю общавшихся с владельцем продукта людей, которая приобретет этот продукт под влиянием общения..

Назовите этот параметр *AdoptionFraction*.

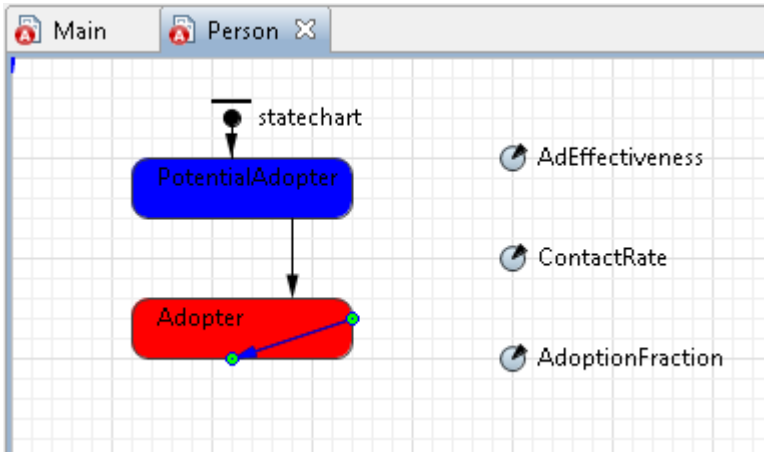
Задайте Значение по умолчанию 0.015.



Задание 3. Измените диаграмму состояний агента

Откройте диаграмму класса *Person*.

Добавьте внутренний переход в состояние *Adopter*. В данном случае проще будет нарисовать его, перейдя в специальный режим рисования. Для того, чтобы перейти в этот режим, сделайте двойной щелчок мышью по элементу **Переход** в палитре **Диаграмма состояний** (при этом его значок должен поменяться на этот:). Теперь Вы можете рисовать переход, последовательными щелчками мыши добавляя в нужных Вам местах диаграммы начальную точку перехода, затем точки его изгиба и, наконец, двойным щелчком - конечную точку перехода. Щелкните мышью вначале по одной границе состояния *Adopter*, а затем сделайте двойной щелчок мышью по другой его границе. При этом должен будет нарисоваться следующий переход:



Это внутренний переход состояния *Adopter*. Этот переход будет моделировать общение человека со своим знакомым, в результате тот может быть убежден в покупке нового продукта. Интенсивность срабатывания этого перехода будет зависеть от интенсивности общения этого человека.

Выберите на странице свойств этого перехода из выпадающего списка **Происходит** *С заданной интенсивностью* и задайте новое значение **Интенсивности** срабатывания этого перехода: ContactRate

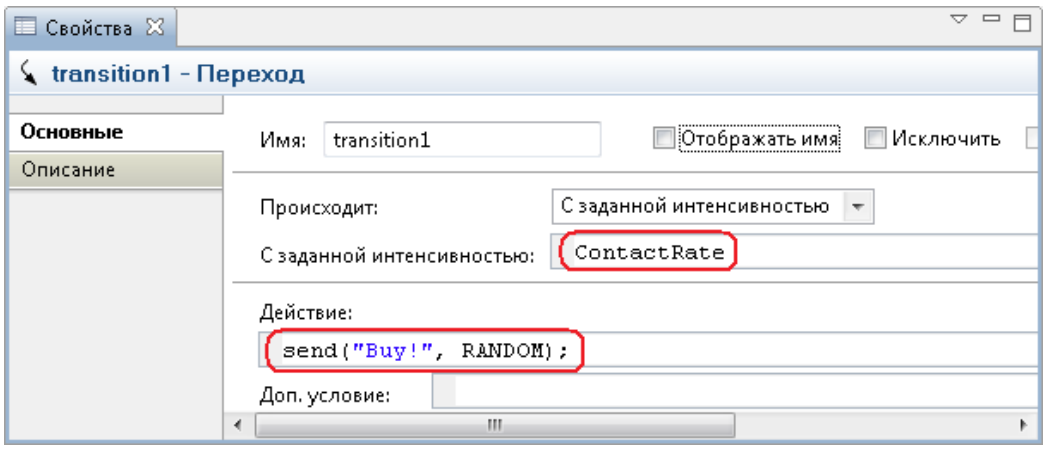
Задайте **Действие** этого перехода:

```
send("Buy!", RANDOM);
```

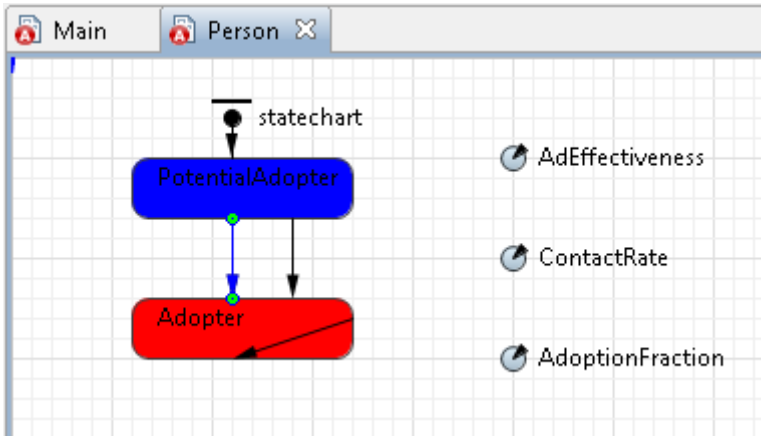
Этот переход посылает сообщение случайно выбранному человеку. Позднее мы сделаем так, что вследствие этого будет срабатывать переход диаграммы состояния этого человека, моделирующий покупку им продукта. Метод send() отсылает сообщение другому агенту. Первый аргумент задает сообщение, которое будет послано, а второй задает агента, которому это сообщение будет адресовано. В нашем случае мы посылаем сообщение какому-то случайно выбранному агенту, поэтому в каче-

стве значения этого аргумента мы используем специальную константу RANDOM.

Подробное описание этого и других методов, используемых для организации взаимодействия агентов смотрите в сатье документации Взаимодействие агентов.



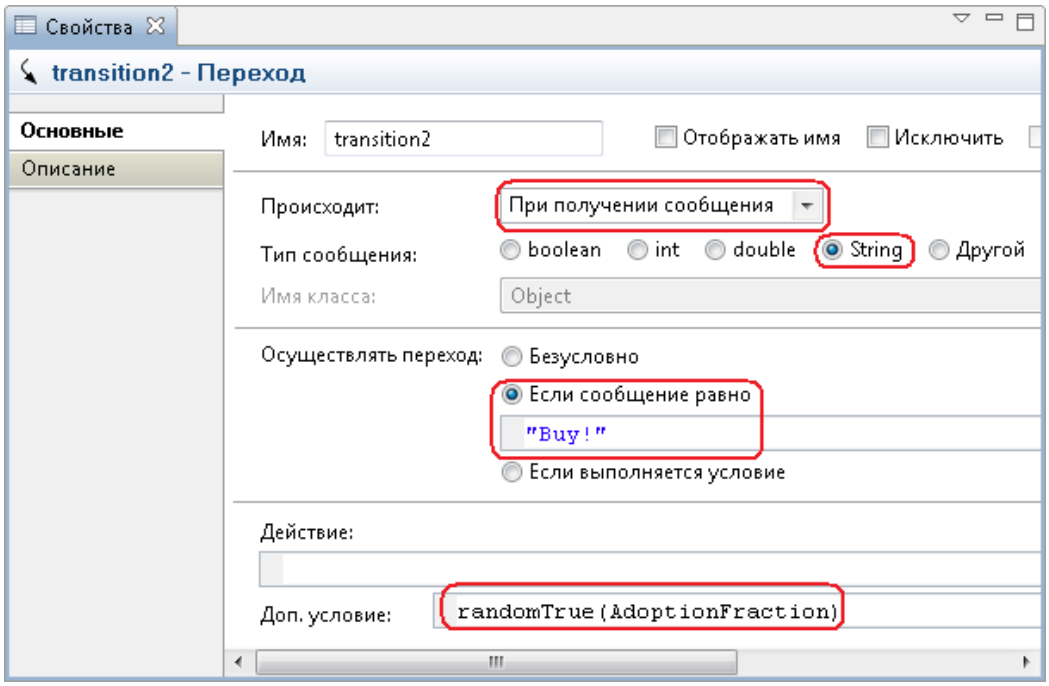
Добавьте еще один переход из состояния *PotentialAdopter* в состояние *Adopter*. Этот переход моделирует процесс приобретения продукта под воздействием общения со знакомым.



Измените свойства этого перехода. Не каждое обсуждение достоинств продукта со своим знакомым приведет к немедленному решению о приобретении этого продукта. Вероятность такого развития событий будет зависеть от того, насколько данный потенциальный потребитель подвержен внушению. В нашей модели данная характеристика задается параметром AdoptionFraction. Перейдите на страницу свойств этого перехода и введите randomTrue(AdoptionFraction) в поле **Доп. условие**. Это дополнительное условие приведет к тому, что продукт будет приобретаться с вероятностью, задаваемой параметром AdoptionFraction.

Этот переход будет срабатывать, когда диаграмма состояний этого агента получит сообщение *"Buy!"* (то есть, "Купи") от другого агента - своего знакомого. Чтобы этот переход срабатывал при получении сообщения, на странице свойств этого перехода выберите из выпадающего списка **Происходит** *При получении сообщения*.

Теперь нам нужно указать, что переход будет срабатывать только при получении сообщения соответствующего содержания. Для этого выберите из группы кнопок **Тип сообщения** опцию **String**, выберите ниже опцию **Если сообщение равно** и введите "Buy!" в расположенном ниже поле.



Теперь нам нужно изменить некоторые свойства агента, для того, чтобы получаемые им сообщения от других агентов перенаправлялись в его диаграмму состояний и обрабатывались ею в соответствии с заданной логикой.

Задание 4. Измените свойства агента

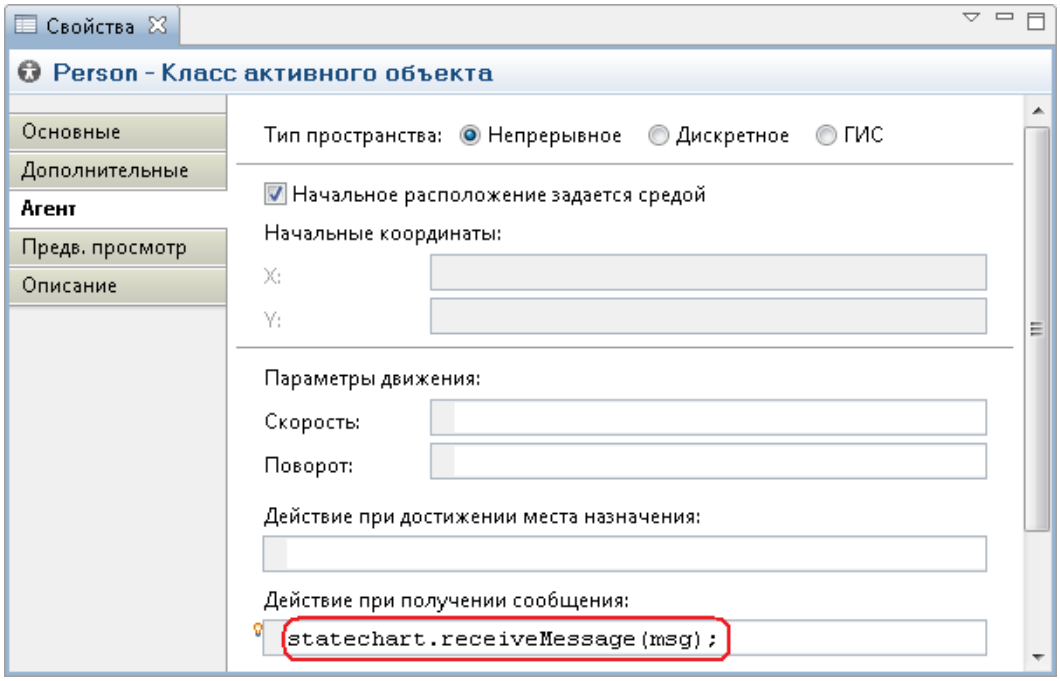
Щелкните мышью по классу *Person* в панели **Проекты**, чтобы открыть его свойства в панели **Свойства**.

Перейдите на страницу свойств **Агент**.

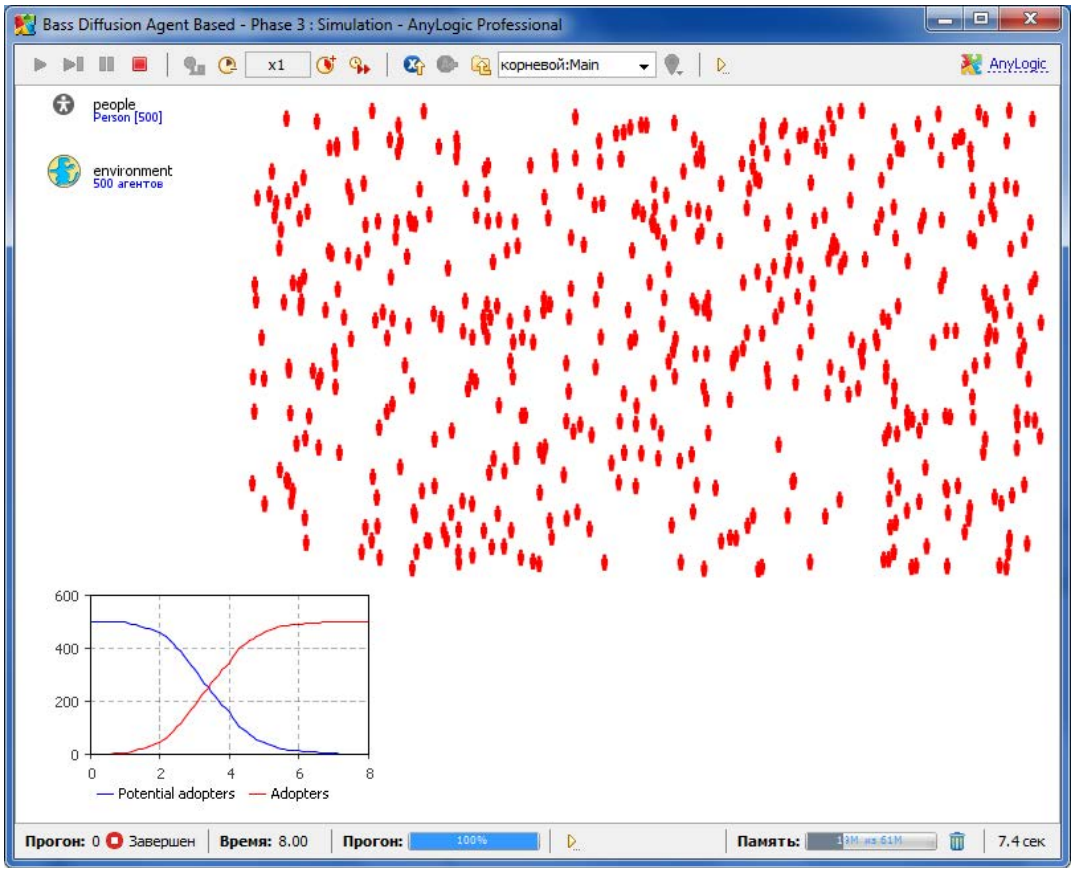
В поле **Действие при получении сообщения** введите statechart.receiveMessage(msg);

Теперь когда агент получит сообщение от какого-то другого агента, он будет перенаправлять его в свою диаграмму состояний, где оно

будет обрабатываться так, как мы с вами это задали (а именно, вызывать срабатывание перехода, моделирующего приобретение продукта под влиянием личного общения).



Запустите модель и изучите динамику процесса приобретения продукта. Вы можете увидеть, что из-за учета влияния устного общения этот процесс стал проистекать значительно быстрее.



Графики переменных представляют собой классические S-образные кривые – динамика процесса в чем-то напоминает динамику распространения заболевания. Итак, можно сказать, что мы практически абсолютно точно воспроизвели результаты, которые выдавала для данной постановки задачи системно-динамическая модель.

Сейчас люди в нашей модели случайно располагаются в прямоугольном пространстве 650x300 километров (или других условных единиц расстояния). И наша модель допускает общение любого человека с каждым, вне зависимости от того, на каком расстоянии друг от друга они находятся. Обычно же у человека есть определенный круг знакомых, которые живут в непосредственной близости к нему, и именно с ними он и общается. Поэтому мы хотим, чтобы в нашей модели общались только те люди, которые находятся не далее определенного расстояния друг от друга.

Давайте сделаем нашу модель более реалистичной, допустив возможность общения только тех людей, которые находятся друг от друга на расстоянии, не превышающем 25 километров.

Свойства формирования сетей контактов агентов, как и многие другие свойства агентной модели, задаются в объекте *среда*.

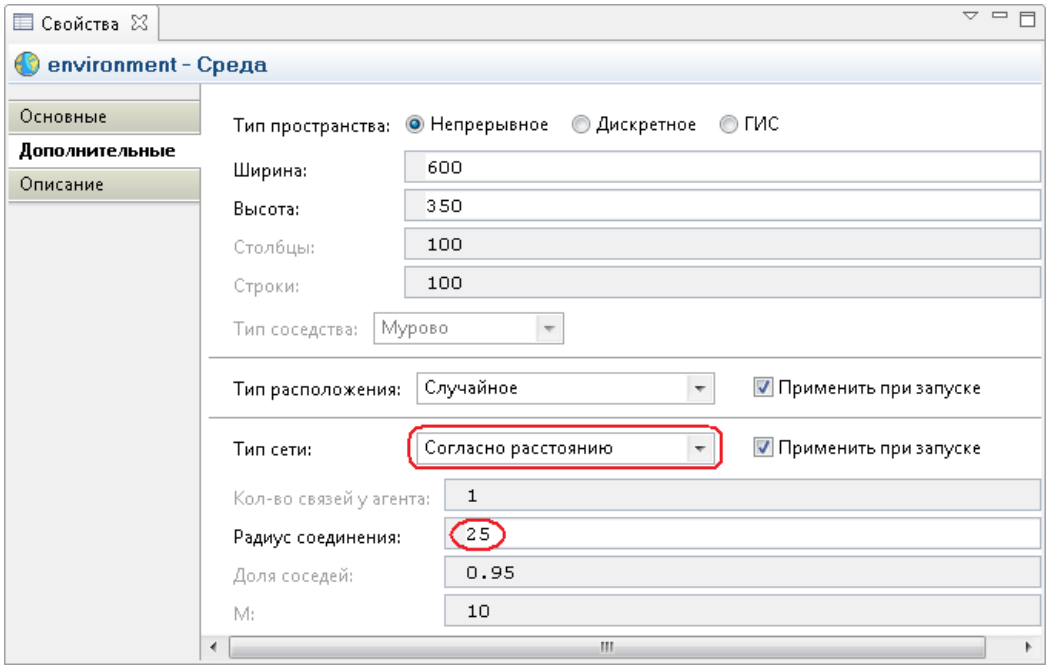
Задание 5. Измените свойства среды

Откройте диаграмму класса *Main*.

Выделите на диаграмме объект environment , задающий настройки среды, в которой обитают агенты.

Перейдите на страницу свойств **Дополнительные**.

Нам нужно изменить тип сети контактов. Выберите **Согласно расстоянию** из выпадающего списка **Тип сети** и введите 25 в расположенном ниже поле **Радиус соединения**.



Теперь нам нужно изменить диаграмму состояний агента, чтобы сообщение "Купи продукт!" отсылалось не случайно выбранному агенту, а только тому агенту, который является знакомым данного агента.

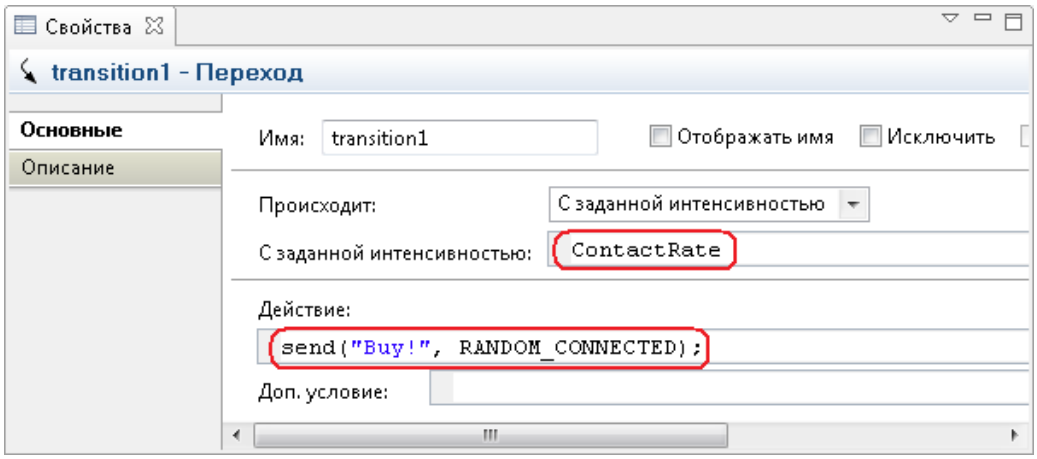
Задание 6. Измените диаграмму состояний агента

Откройте диаграмму класса *Person*.

Измените свойства внутреннего перехода состояния *Adopter*.

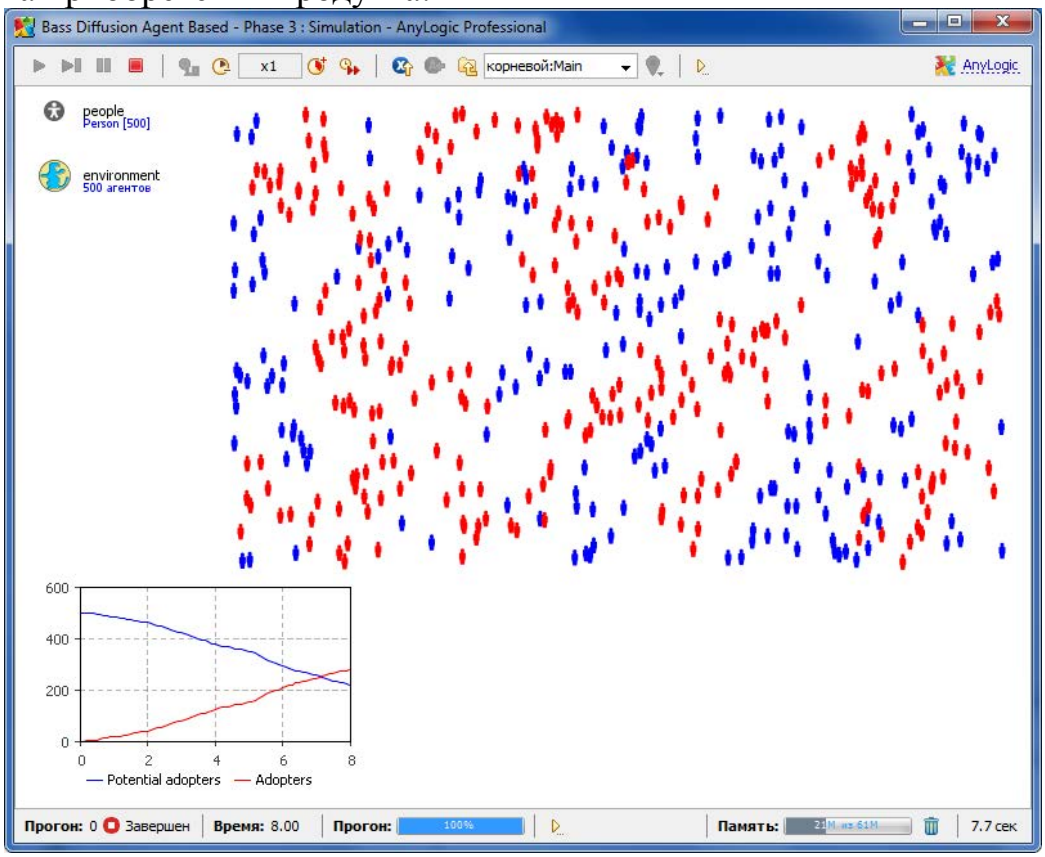
Измените **Действие** этого перехода на:

send("Buy!", RANDOM_CONNECTED);



В нашем случае мы посылаем сообщение какому-то случайно выбранному агенту из числа тех, с которым данный агент знаком, поэтому в качестве значения последнего аргумента метода send мы теперь используем специальную константу RANDOM_CONNECTED. Теперь этот переход посылает сообщение случайно выбранному знакомому этого человека.

Давайте теперь запустим модель и посмотрим, как изменилась динамика приобретения продукта:



Можно увидеть, что теперь агенты соединены только с теми, которые находятся от них на расстоянии, не превышающем 25 единиц, а сам процесс распространения продукта происходит медленнее.

4.8 Моделирование повторных покупок

Созданная модель не учитывает того, что со временем продукт может быть израсходован или прийти в негодность, что вызовет необходимость его повторного приобретения. Мы промоделируем повторные покупки, полагая, что потребители продукта снова становятся потенциальными потребителями, когда продукт, который они приобрели, становится непригоден.

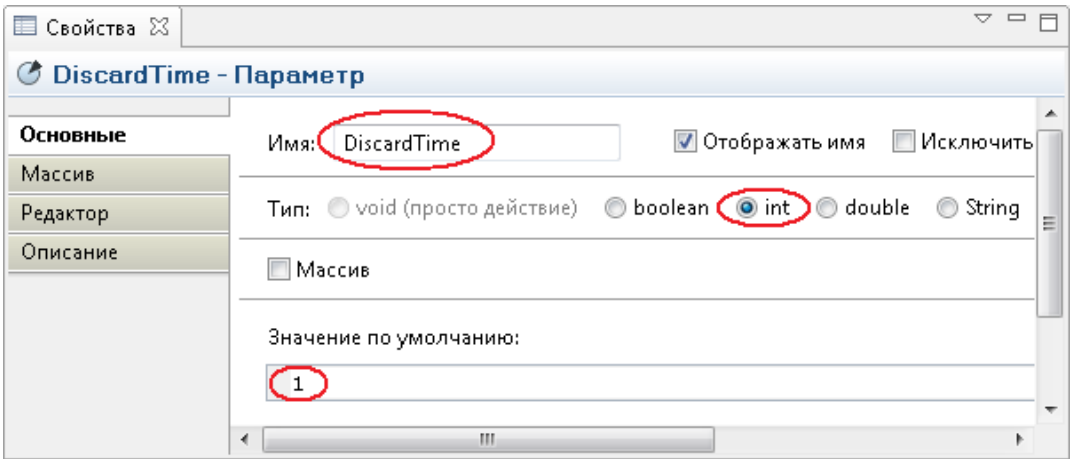
Вначале мы зададим срок службы продукта. Предположим, что средний срок службы нашего продукта - 1 год.

Задание 1. Задайте средний срок службы продукта

Откройте диаграмму класса *Main*.

Создайте параметр *DiscardTime*. Пусть средний срок службы нашего продукта равен одному году.

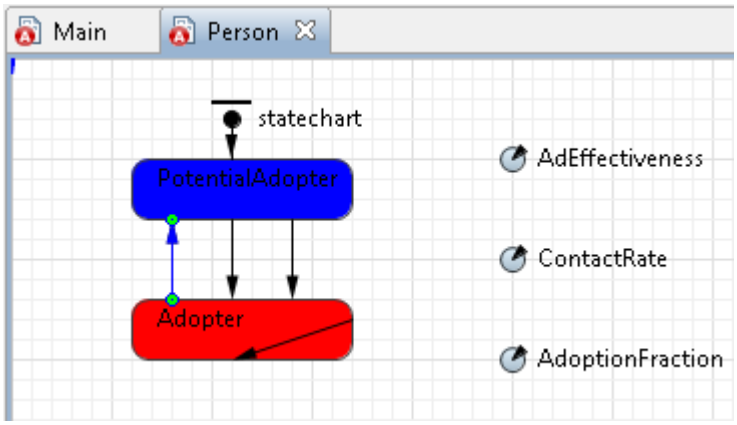
Задайте Значение по умолчанию: 1.



Задание 2. Измените диаграмму состояний

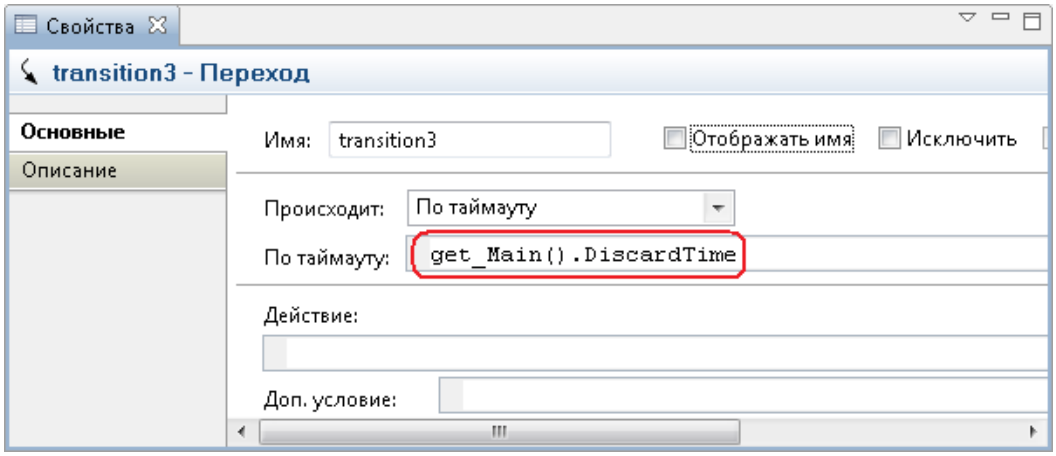
Откройте диаграмму класса *Person*, сделав двойной щелчок мышью по элементу *Person* в панели **Проекты**.

Добавьте переход из состояния *Adopter* в состояние *PotentialAdopter*.



Измените свойства перехода. Этот переход будет срабатывать по прошествии срока службы нашего продукта, заданного параметром *DiscardTime*, после того, как управление диаграммы состояний перейдет в состояние *Adopter*. Поэтому оставьте в свойстве **Происходит по** принятое по умолчанию значение *Таймауту* и введите в поле **Таймаут** get_Main().DiscardTime.

Метод get_Main() здесь возвращает экземпляр класса Main, в котором мы задали параметр DiscardTime.



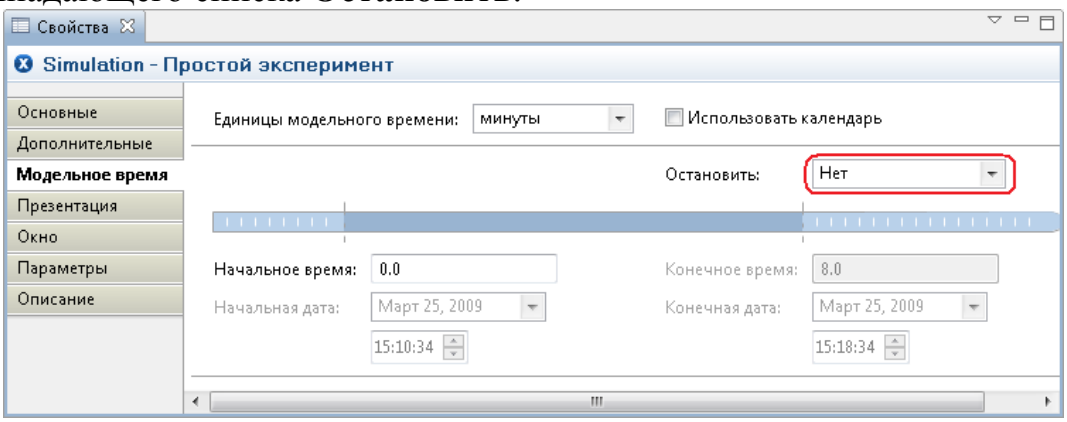
Мы закончили моделирование повторных покупок продукта.

Теперь мы хотим исследовать процесс приобретения продукта в течение более длительного периода времени. Уберите заданное ранее условие остановки модели по прошествии определенного числа единиц модельного времени, чтобы модель выполнялась бесконечно, пока ее не остановит пользователь.

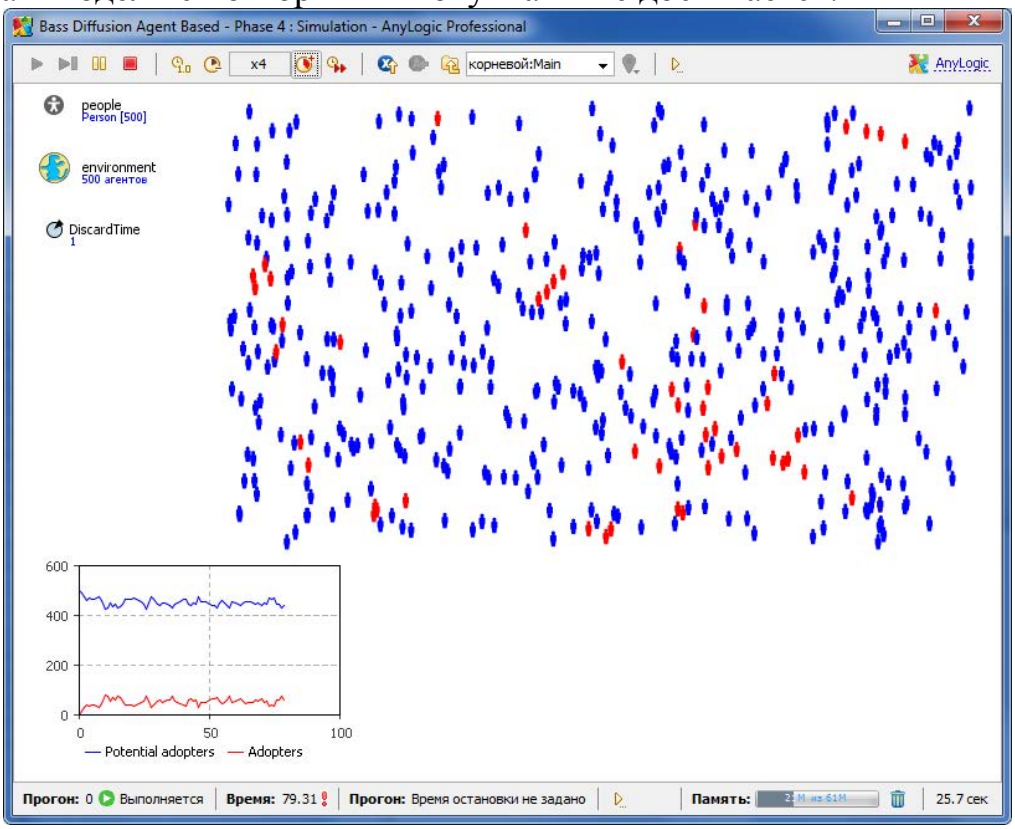
Задание 3. Удалите условие остановки модели по времени

В панели **Проекты**, выделите эксперимент *Simulation:Main* щелчком мыши.

На странице **Модельное время** панели **Свойства**, выберите **Нет** из выпадающего списка **Остановить**.



Запустите модель и с помощью диаграммы проследите динамику изменения числа потребителей продукта. Мы видим, что насыщение рынка в модели с повторными покупками не достигается.



Лабораторная работа 4. Комбинированное моделирование

1. Цель работы

Получить практические навыки по комбинированному моделированию, используя в одной модели как агентный подход, так и моделирование динамических систем, на примере модели потребительского рынка и цепочки поставок.

2. Краткие теоретические сведения

AnyLogic позволяет создавать имитационные модели с помощью различных подходов моделирования: Системной динамики, Агентного, а также Дискретно-событийного (или Процессного) моделирования. Более того, Вы можете совмещать различные методы в одной модели: помещать агентов в окружение, чья динамика задана в стиле Системной динамики, использовать диаграммы процесса или системную динамику для задания внутренней структуры агента и т.д. Благодаря своему уникальному языку моделирования AnyLogic поддерживает любые способы комбинирования различных подходов в одной модели.

Выбор архитектуры модели (как разделять модель на компоненты, что агрегировать, какой уровень детальности выбрать, какое поведение естественнее задавать с помощью диаграммы процесса, а какое - с помощью диаграммы состояний и т.д.) зависит в основном от опыта и интуиции разработчика моделей. Эта лабораторная работа позволит научить пошагово создавать модель, совместно использующую системную динамику и агентное моделирование. Создав такую модель, впоследствии Вам будет легче строить различные многоподходные модели.

Мы создадим модель потребительского рынка и цепочки поставок.

Рынок будет моделироваться, исходя из предположений, аналогичных тем, что делаются при моделировании классических моделей распространения продукта/инновации, например, модели Басса с продуктом с ограниченным сроком эксплуатации и повторными покупками для замены вышедших из строя продуктов. Но при этом вместо одного мы рассмотрим сразу два конкурирующих друг с другом продукта.

Есть два альтернативных продукта: A и B, производимые различными (и конкурирующими) компаниями. Продукты эквивалентны, т.е. могут легко использоваться один вместо другого. Цены на товары одинаковы и поэтому не имеют значения.

Потребители (общей численностью **Total Population = 1000**) изначально не являются пользователями ни одного из рассматриваемых

продуктов, но потенциально заинтересованы в продукте (являются потенциальными пользователями).

Потребители приобретают продукт под влиянием рекламы и общения с пользователями этого продукта.

Реклама порождает спрос на продукт среди потенциальных потребителей. Эффективность рекламы **Advertizing Effectiveness = 0.011** задает процент потенциальных пользователей, которые принимают решение о приобретении определенного продукта (A или B) под влиянием рекламы в течение дня. Обе компании проводят рекламные кампании.

Потребители общаются друг с другом. Общаясь с потенциальными пользователями продукта, владельцы продукта могут убедить их в необходимости приобретения продукта. В среднем один пользователь продукта успешно убеждает за день одного своего собеседника.

По истечении времени **Discard Time = uniform(17,23)** дня продукты приходят в негодность, что порождает немедленный спрос на продукт того же бренда, что и только что вышедший из эксплуатации.

Если человек хочет приобрести продукт (например, A), но этого продукта нет в наличии в течение максимально допустимого времени ожидания **Maximum Waiting Time = 2** дня, то этот человек готов приобрести продукт другой компании, если он доступен (A или B); то же для B.

У каждой компании (и A, и B) есть своя цепочка поставок, используемая для поставки товаров конечным пользователям. Цепочки поставок достаточно просты и работают следующим образом:

Потребитель может приобрести продукт только у ритейлера (изначально обладающего определенным количеством продукта (**Initial Retailer Stock = 100** единиц)).

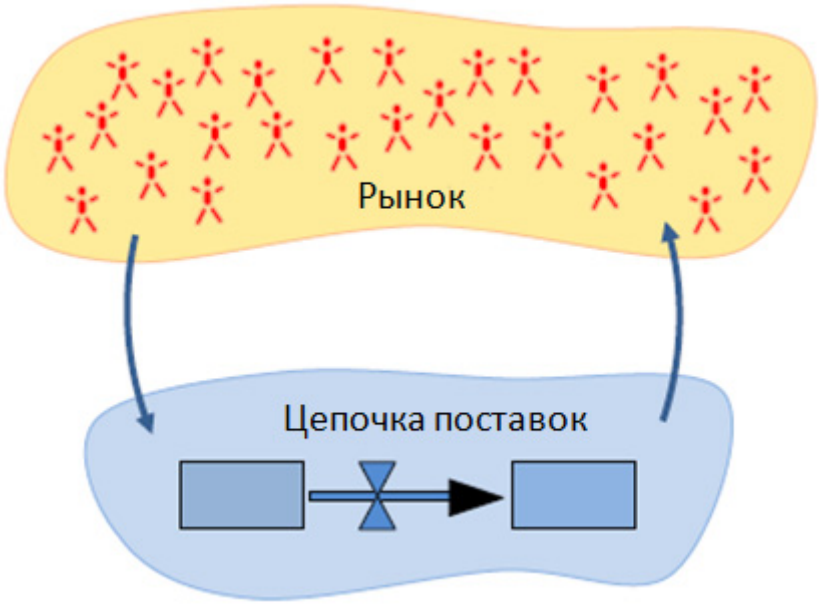
Продукт производится производителем. Производитель выпускает **Production Rate** единиц продукта в день, и эта интенсивность производства может меняться, например, она может подгоняться под текущий спрос (известный производителю).

Доставка готовой продукции ритейлеру занимает **Delivery Time = 2** дня.

В качестве результата модель должна показывать доли рынка для продуктов A и B, спрос (т.е. количество человек, которые хотят приобрести продукт(ы), но не могут этого сделать в силу их отсутствия у ритейлеров), а также уровни запасов цепочек поставок.

Мы будем моделировать потребительский рынок с помощью агентного подхода моделирования: каждый потребитель будет являться агентом. Цепочки поставок для обоих продуктов будут заданы в стиле системной динамики. Обратите внимание, что постановка задачи позво-

ляет выбрать и другие комбинации подходов моделирования - это просто одна из них.



Лучше всего создавать имитационные модели итеративно, т.е. за несколько фаз, в конце каждой из которых создается готовая к запуску модель. В нашем случае имеет смысл сделать это в следующем порядке:

Начнем создание модели рынка с рассмотрения только одного продукта (A)

Построим модель поведения потребителя

Населим рынок потребителями

Предположим, что продукт есть в наличии

Добавим цепочку поставок для продукта A

Добавим продукт B

3. Задание на лабораторную работу

3.1 Выполняя последовательно шаги проектирования создать модель потребительского рынка и цепочки поставок.

3.2 Провести вычислительный эксперимент и сделать выводы.

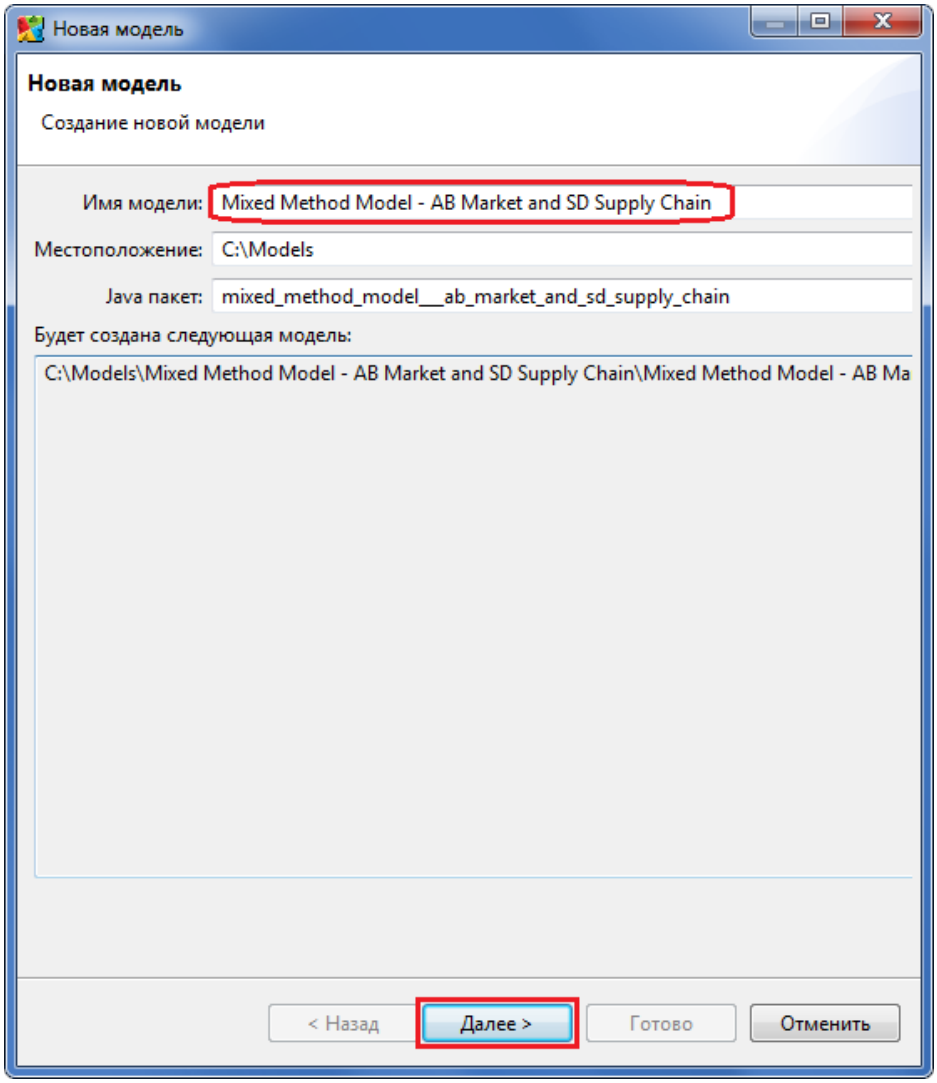
4. Порядок выполнения работы.

4.1 Создание 1000 агентов

В этой фазе мы создадим новую модель AnyLogic, воспользовавшись готовым шаблоном агентной модели. В итоге мы получим модель, в которой будут моделироваться потребители численностью 1000 человек (с простой анимацией каждого потребителя). На начальном этапе у

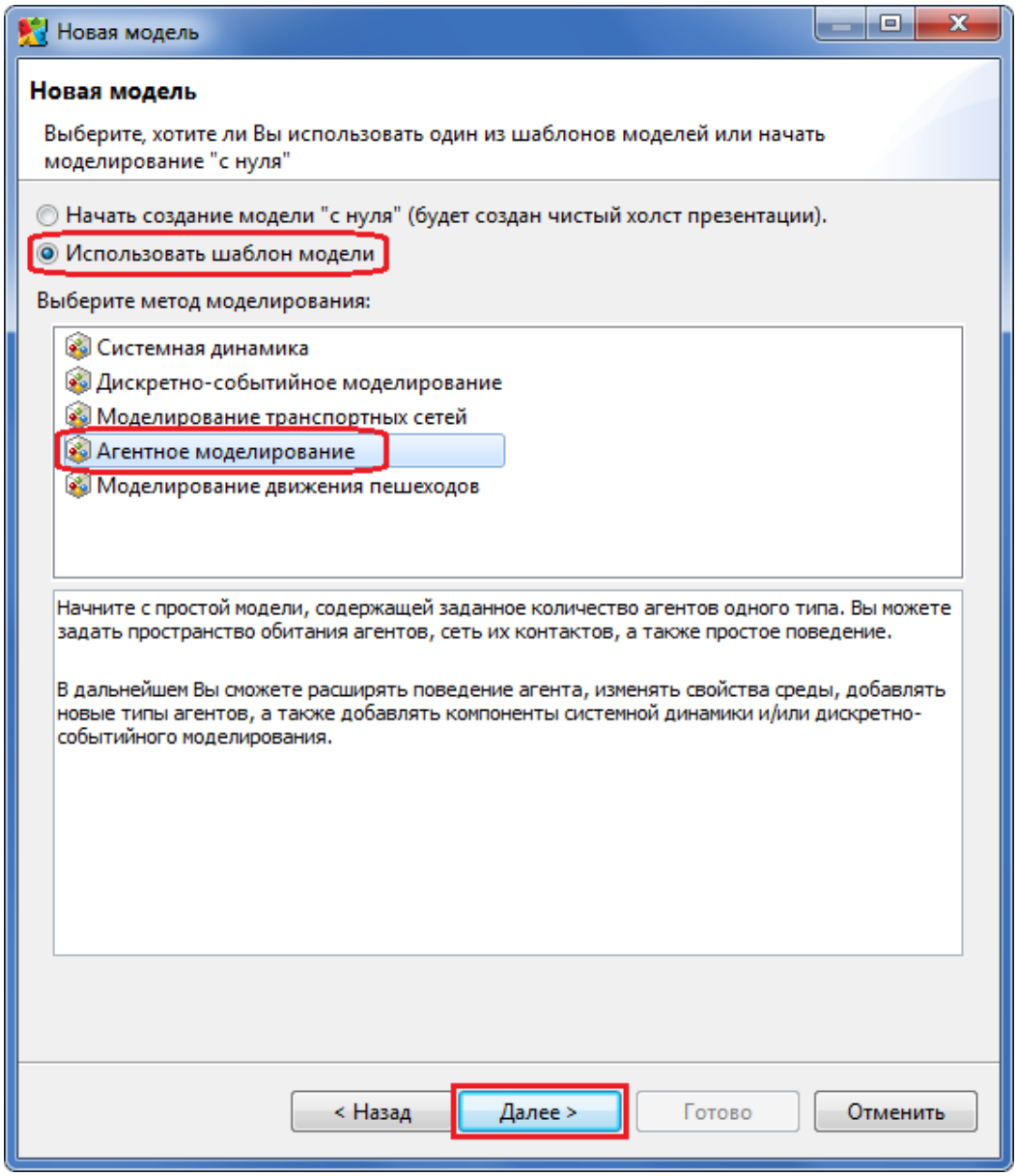
потребителей не будет задано никакого поведения, и поэтому пока что в этой модели ничего не будет происходить.

Создайте новую модель Mixed Method Model - AB Market and SD Supply Chain:



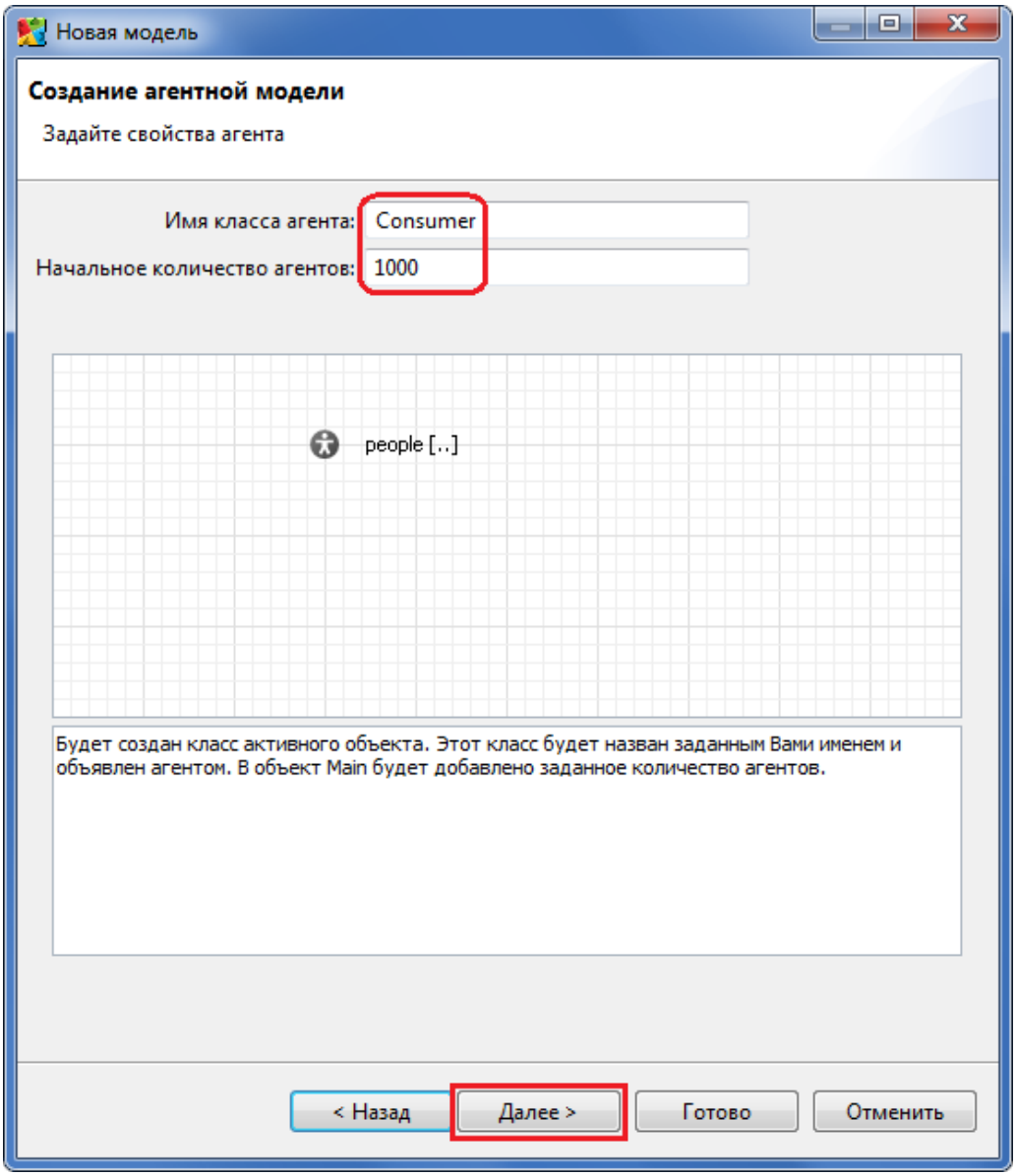
Когда Вы создаете новую модель, AnyLogic предлагает Вам на выбор набор шаблонов готовых моделей, с помощью которых Вы можете минимизировать количество действий, требуемых при начальном построении модели. Каждый такой шаблон содержит базовые конструкции, которые Вам могут обычно понадобиться при создании модели с помощью того или иного метода моделирования: системной динамики, дискретно-событийного (дискретного) подхода, агентного подхода, пешеходной динамики и т.д.

Выберите шаблон **Агентное моделирование** и перейдите к следующей странице мастера для конфигурации создаваемой модели.



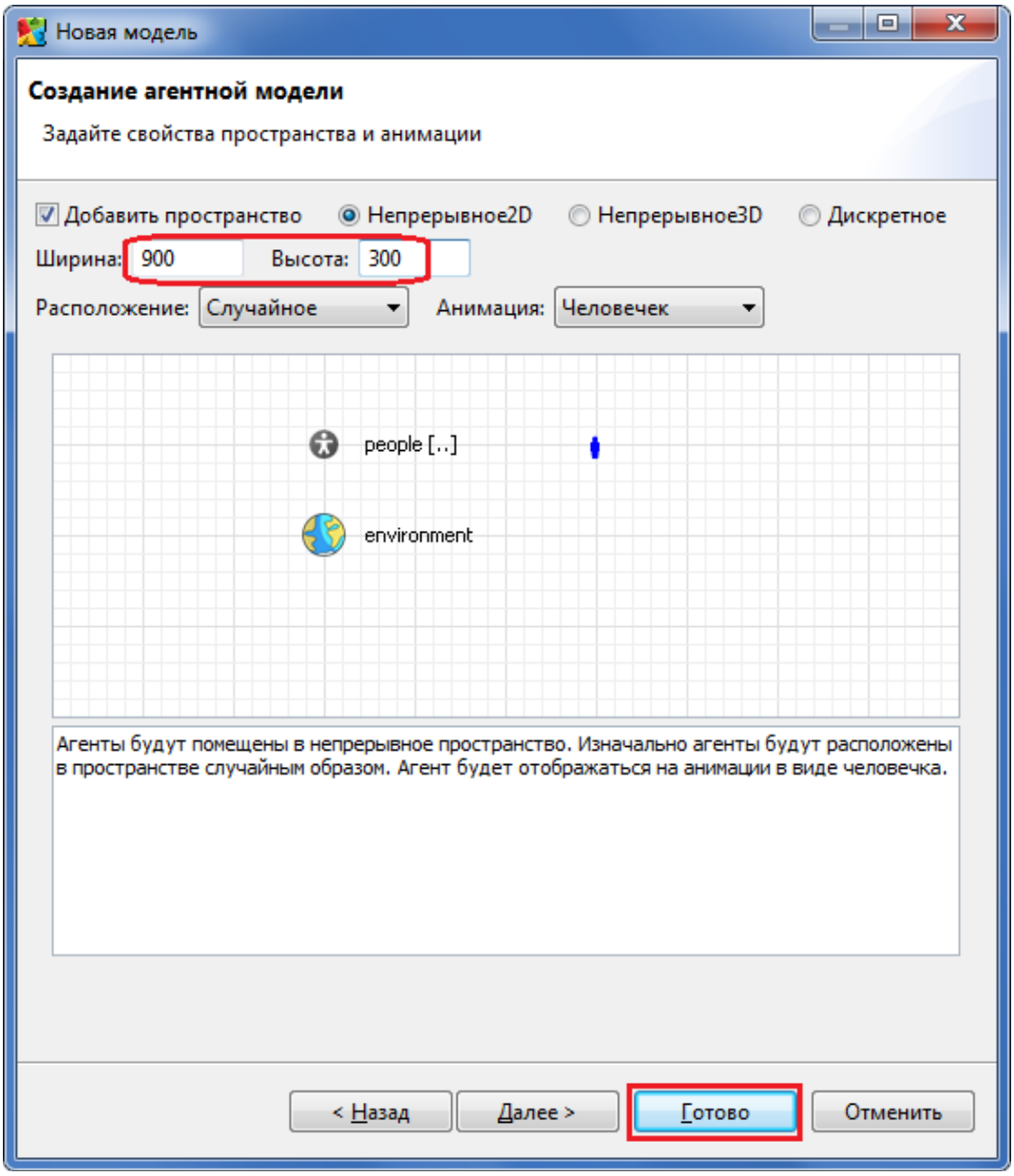
Нам нужно создать отдельный класс для потребителей, в котором мы будем задавать поведение потребителя. Заключив это поведение в класс, мы можем создать множество экземпляров этого класса, которые будут представлять отдельных потребителей. Задайте *Consumer* в качестве имени этого класса агента. Класс будет автоматически объявлен как **Агент** (на странице основных свойств этого класса будет установлен флажок **Агент**). Это позволит использовать некоторые встроенные возможности AnyLogic по поддержке агентов, например, механизм взаимодействия агентов, выбрать тип пространства и легко задать их расположение в нем и т.д.

Задайте Начальное количество агентов в модели: *1000*.



На следующей странице Мастера создания модели, оставьте в поле **Анимация**: **Человечек**. Здесь Вы выбираете фигуру, которая будет визуально представлять потребителя на анимации модели.

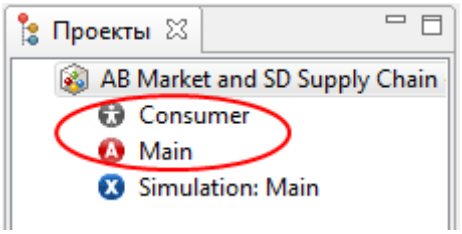
Измените размер прямоугольной области, в которой будут отображаться агенты на анимации. Задайте **Ширину** 900 и **Высоту** 300.



Обычно управляемое событиями поведение агента задается с помощью диаграммы состояний, и Мастер создания моделей предлагает также создать и диаграмму состояний, задающую весьма простое поведение. Но мы не будем использовать эту опцию, поскольку мы хотим нарисовать диаграмму состояний с нуля.

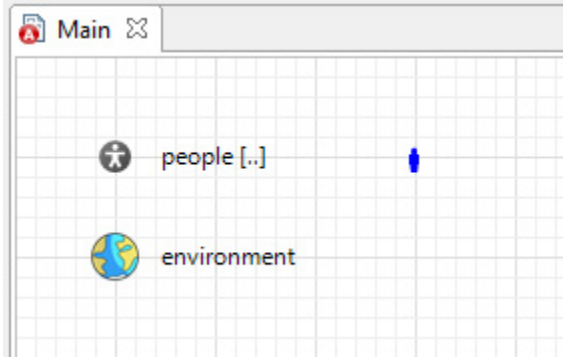
Завершите процесс создания модели.

Теперь мы закончили создание простой агентной модели. Шаблон агентной модели содержит два класса: один для агента (в нашем случае это *Consumer*) и один для окружающей среды (*Main*).



В новой модели всегда есть класс активного объекта *Main*, чья диаграмма автоматически открывается в графическом редакторе. Объект *Main* представляет собой верхний уровень иерархии нашей модели, здесь мы зададим взаимодействие между агентной и системно-динамической частями нашей модели.

Вы можете увидеть на диаграмме класса *Main* элемент *environment*, объект *people*, а также фигуру анимации потребителя.



Переместите анимацию вложенного объекта потребителя на диаграмме класса *Main* в точку (50,250).

Местоположение этой фигуры анимации будет задавать верхний левый угол прямоугольной области, в которой позднее будет визуализироваться население потребителей. Остальные элементы модели мы будем располагать над этой областью.

Объект *people* представляет собой реплицированный объект, изначально содержащий 1000 элементов (поскольку именно такое количество мы задали в Мастере создания моделей). Квадратные скобки [..] рядом с именем означают, что этот объект - реплицированный.

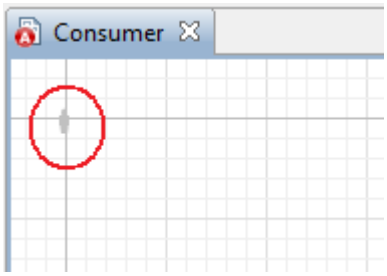
Каждый элемент этого реплицированного объекта моделирует одного отдельного потребителя. Количество таких элементов - потребителей может быть изменено в свойстве **Количество** этого вложенного объекта.

Измените имя вложенного объекта с *people* на *consumers*. *consumers* будет именем этой популяции агентов.

Элемент **Среда** поддерживает различные типы пространств, расположение агентов в этих пространствах, сети и взаимодействие агентов. В нашем случае среда нужна нам для того, чтобы расположить ани-

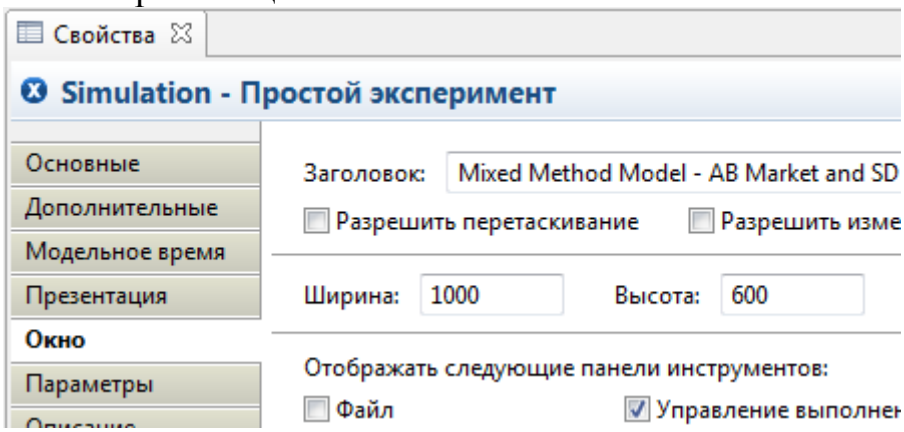
мации агентов и чтобы промоделировать общение людей друг с другом с помощью встроенного механизма агентного взаимодействия. Этот элемент указан в качестве среды для агентов, заданных реплицированным объектом *consumers* (имя этого элемента указано в свойстве **Среда** объекта *consumers*).

На диаграмме класса потребителя *Consumer* Вы увидите небольшое изображение человечка (рядом с началом координат, в верхнем левом углу). Смените цвет заливки этой картинки (в действительности, являющейся кривой) на серый. Каждый потребитель будет отображаться на анимации модели такой фигуркой. Цвет фигуры, равно как и другие ее свойства, могут быть впоследствии изменены, индивидуально для каждого потребителя.



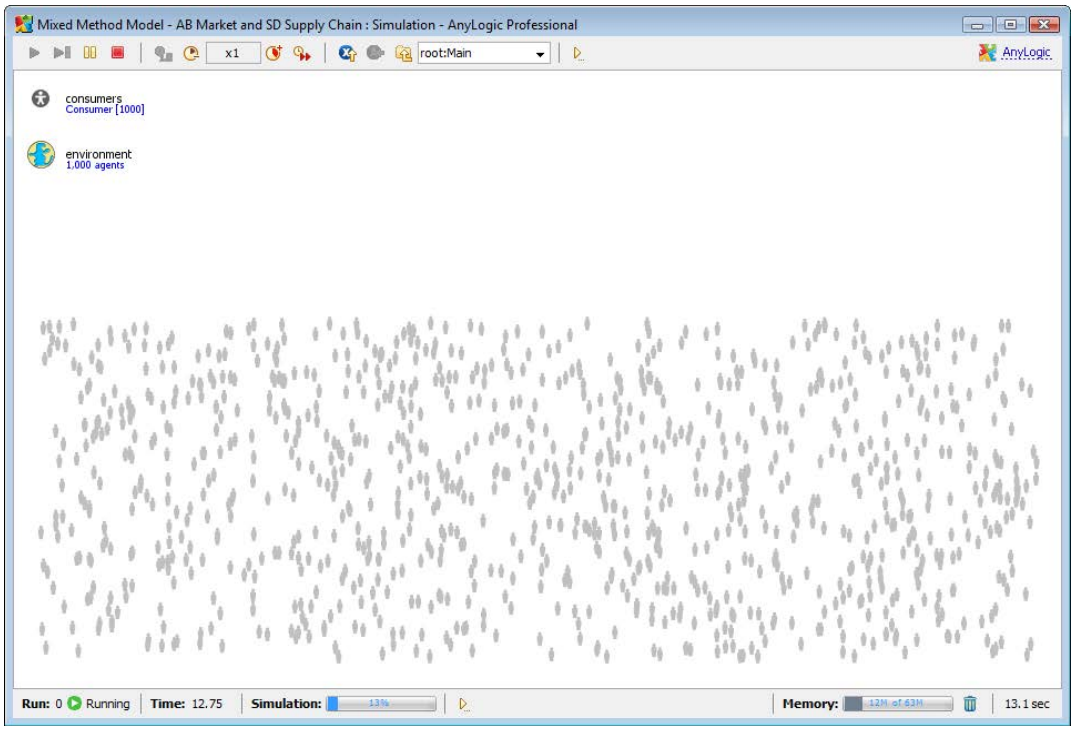
Щелкните в дереве проектов по элементу эксперимента *Simulation:Main*, перейдите на страницу его свойств **Окно** и задайте размер окна презентации, введя в поле **Ширина** 1000, а в поле **Высота** 600.

Таким образом Вы задаете начальный размер окна презентации модели. Теперь область, в которой будут отображаться агенты, поместится в окно презентации.



Запустите модель.

Вы увидите окно наподобие показанного на рисунке ниже. В модели нет динамики, поэтому с течением времени ничего не происходит.

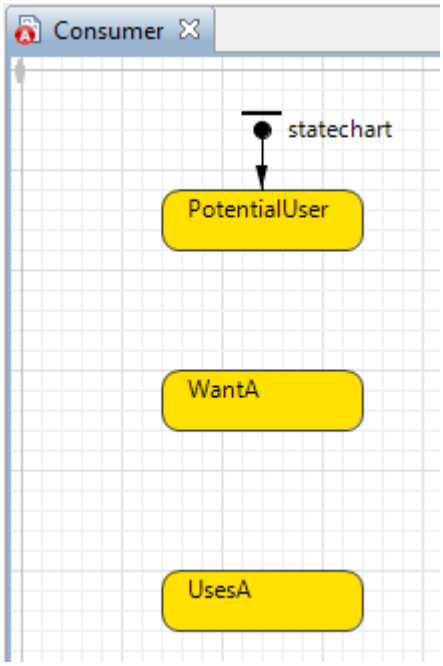


4.2 Задание простого поведения агента

Исходя из постановки задачи, мы зададим поведение потребителя как последовательность трех состояний: *PotentialUser*, *WantA*, *UsesA*. Мы полагаем, что продукт будет всегда в наличии, поэтому переход из состояния *WantA* в состояние *UsesA*будет мгновенным и не будет требовать выполнения каких-либо условий для срабатывания. Эффект от рекламы будет моделироваться как связанная с переходом из состояния *PotentialUser* в состояние *WantA* стохастическая задержка. На данном этапе мы не будем учитывать ни повторные покупки продукта, ни общение людей друг с другом.

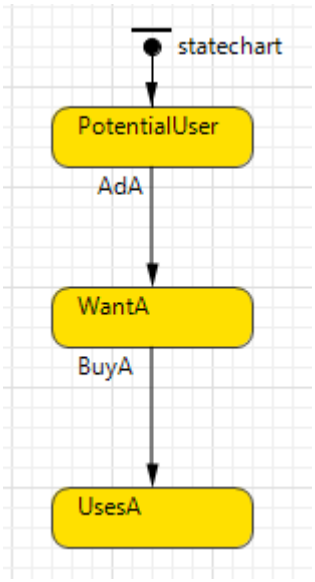
Нарисуйте в редакторе класса *Consumer* три состояния (расположите их снизу вверх в следующем порядке): *PotentialUser*, *WantA*, *UsesA*. Добавьте элемент *Начало диаграммы состояний*, чтобы он указывал на состояние *PotentialUser*.

Состояния диаграммы альтернативны друг другу: в каждый отдельный момент времени агент (в нашем случае – потребитель) может находиться только в одном состоянии. Начало диаграммы состояний указывает на начальное состояние диаграммы, которым, очевидно, является *PotentialUser* (то есть, потенциальный пользователь). Имя начала диаграммы состояний играет роль имени диаграммы в целом. В одном активном объекте может быть задано сразу несколько диаграмм состояний, но в данной модели нам достаточно одной.



Добавьте переход, ведущий из состояния *PotentialUser* в состояние *WantA* и назовите его *AdA*. Добавьте переход из состояния *WantA* в состояние *UsesA* и назовите его *BuyA*. Установите у обоих переходов флажок **Отображать имя** и измените расположение меток с именами переходов на диаграмме.

Переходы определяют то, как объект изменяет свое состояние. Переход может сработать, например, по приходу агента в точку назначения, по выполнению условия, истечению таймаута и т.д. Переход *AdA* будет моделировать произведенный рекламой эффект - принятие решения о необходимости приобретения продукта, а *BuyA* – событие приобретения продукта А.



Задайте следующие свойства для перехода *AdA*: **Происходит**: *С заданной интенсивностью*, **Интенсивность**: 0.011

Срабатывающий с заданной интенсивностью переход в AnyLogic представляет собой по сути переход, срабатывающий по экспоненциально распределенному таймауту. Когда управление диаграммы состояний переходит в состояние *PotentialUser*, происходит генерация значения согласно экспоненциальному распределению, которое и присваивается таймауту этого перехода. Поэтому у потребителей будут различные времена принятия решения о покупке продукта под влиянием рекламы, но при этом в одну единицу модельного времени (день) в среднем решение о покупке будет принимать 1.1% потенциальных пользователей продукта.

Задайте следующие свойства для перехода *BuyA*: **Происходит**: *По таймауту*, **Таймаут**: 0.

Поскольку мы подразумеваем, что продукт всегда доступен (есть в наличии), то все, кто хочет приобрести продукт, могут это сделать сразу же – поэтому, как только управление диаграммы состояний перейдет в состояние *WantA*, оно тут же, с нулевой задержкой, перейдет дальше, в состояние *UsesA*.

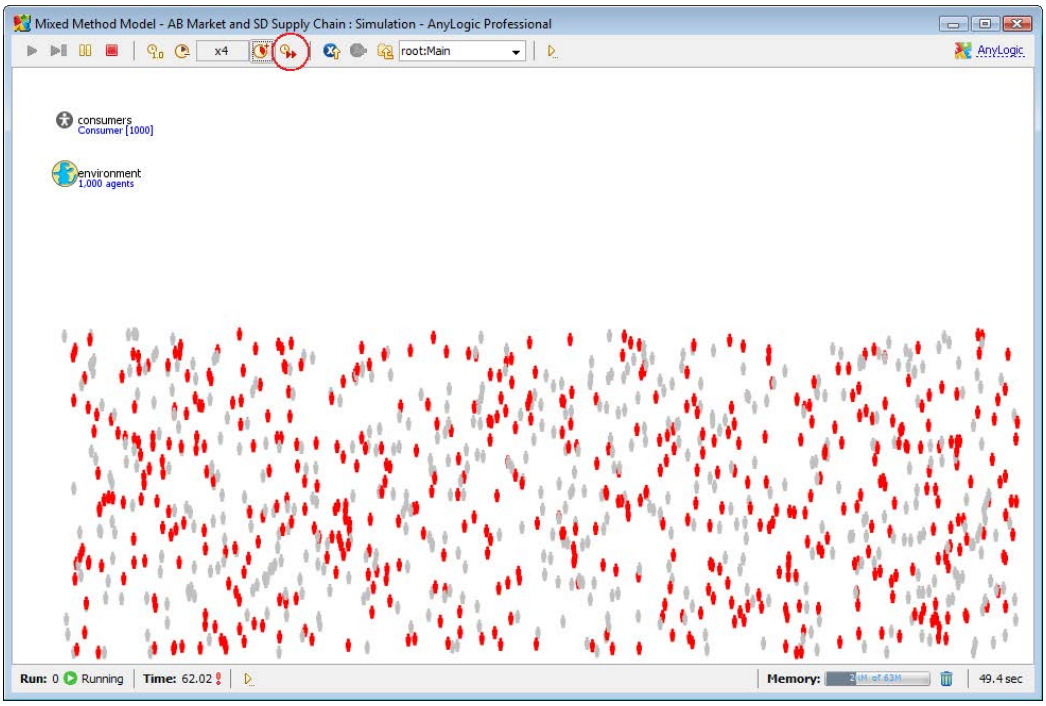
В поле **Действие при входе** состояния *WantA* введите: person.setFillColor(pink);

В поле **Действие при входе** состояния *UsesA* введите: person.setFillColor(red);

Таким способом мы изменяем цвет фигуры анимации потребителя, чтобы визуально отобразить смену его состояния. Как Вы можете видеть, все элементы модели и презентации класса активного объекта доступны один из другого. person здесь - это имя кривой, автоматически нарисованной для задания агента Мастером создания модели. Обратите внимание, что если Вы зададите другую фигуру в качестве презентации объекта класса Consumer, то Вам нужно будет здесь ссылаться уже на ее имя.

Запустите модель. По прошествии некоторого времени можете нажать кнопку панели инструментов **Реальное/виртуальное время**, чтобы ускорить выполнение модели.

Вы сможете увидеть, как фигурки потребителей постепенно закрашиваются красным цветом – таким образом виден производимый рекламой эффект. Хотя каждый потребитель проходит через состояние *WantA*, которому соответствует розовый цвет, фигурок розового цвета не видно, поскольку потребители мгновенно покидают это состояние, чтобы перейти в состояние *UsesA*.



Откройте свойства эксперимента *Simulation:Main*. На странице свойств **Основные** выберите опцию **Случайное начальное число (уникальные "прогоны")**. На странице **Модельное время** выберите из выпадающего списка **Остановить**: **Никогда**.

Поскольку наша модель стохастична (как Вы помните, источниками случайности в ней являются срабатывающие с заданной интенсивностью переходы *AdA* в диаграммах состояний потребителей), то результаты моделирования будут зависеть от генератора случайных чисел. Выбрав опцию **Случайное начальное число**, Вы говорите Anylogic, что для каждого "прогона" модели должны использоваться различные последовательности случайных чисел.

Запустите модель еще раз.

Вы увидите, что в конечном счете все потребители приобретут продукт.

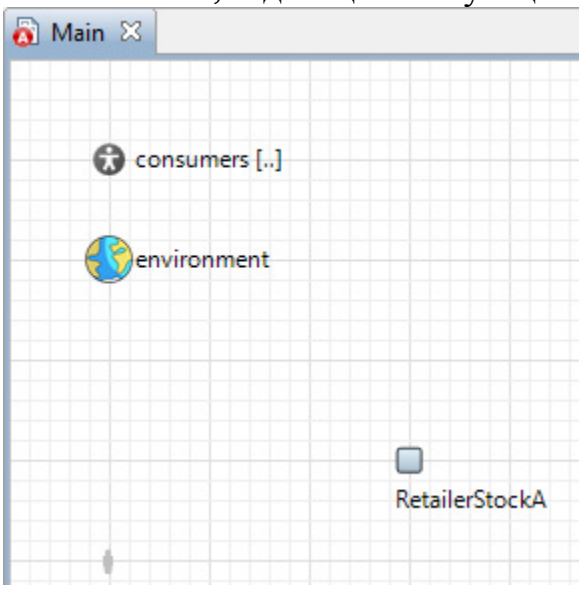
4.3 Добавление ритейлера с некоторым начальным количеством продукта A

В этой фазе мы добавим первый (и фактически последний) элемент цепочки поставок для продукта A: накопитель ритейлера. У накопителя (элемента Системной динамики) будет определенное начальное значение (моделирующее начальное количество продукта), поэтому некоторые потребители смогут приобрести продукт. Мы изменим поведение потребителя так, что переход из состояния *WantA* в состояние *UsesA* будет возможен только если в накопителе будет хотя бы одна единица продукта. В результате перехода потребителя по данному переходу будет уменьшаться количество доступного товара. Это будет первым местом

взаимодействия агентного и системно-динамического компонентов модели.

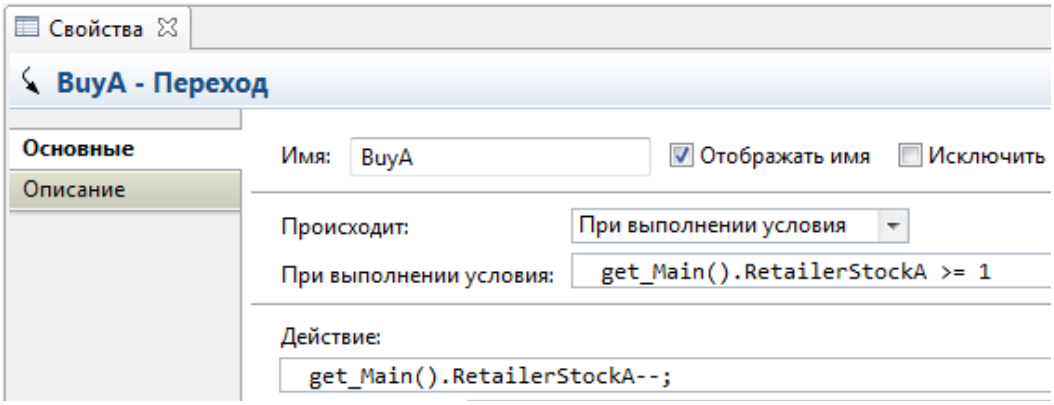
Откройте редактор класса *Main* и добавьте накопитель *RetailerStock*. Задайте его начальное значение равным *100*.

Это первый элемент Системной динамики в нашей модели. Мы создаем его в классе *Main* – на том же уровне иерархии, где мы создали вложенный объект, задающий популяцию агентов-потребителей.



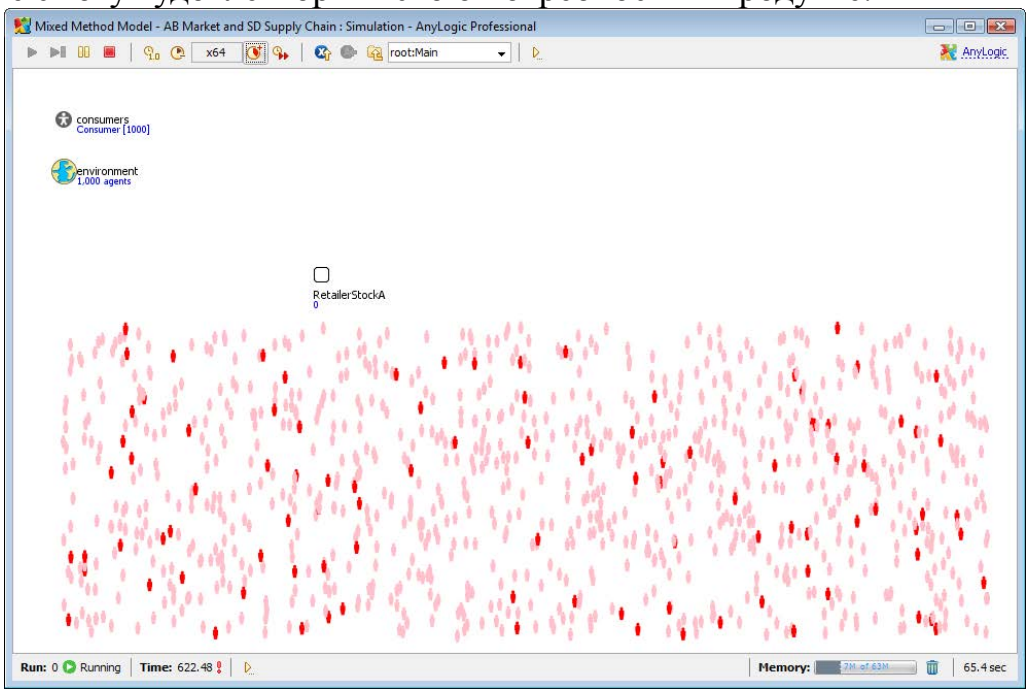
Перейдите в редакторе класса *Consumer* и измените свойства перехода *BuyA*: пусть теперь он **Происходит** *При выполнении условия*. Задайте **Условие**: get_Main().RetailerStockA >= 1 и **Действие**: get_Main().RetailerStockA--;

Таким образом мы реализуем механизм взаимодействия агентной части модели и системно-динамической: будучи в состоянии *WantA* потребитель постоянно отслеживает значение накопителя *RetailerStockA*; если накопитель содержит хотя бы один продукт, то переход происходит, и как результат, из накопителя "извлекается" одна единица товара. Обратите внимание, что поскольку накопитель находится на один уровень выше относительно того места, где задано поведение потребителя (в объекте *Main*, который содержит объекты потребителей), то вначале мы получаем доступ к объекту класса *Main*, и только затем – к самому накопителю, поэтому мы и используем префикс get_Main(), который "переносит" нас в контекст контейнера объекта типа *Consumer* - объект класса *Main*.



Запустите модель в режиме виртуального времени.

Вы увидите, что 100 потребителей смогут приобрести продукт A, в то время, как остальные постепенно будут перекрашены в розовый цвет, поскольку запасы ритейлера скоро подойдут к концу, и многие потребители не смогут удовлетворить свою потребность в продукте.



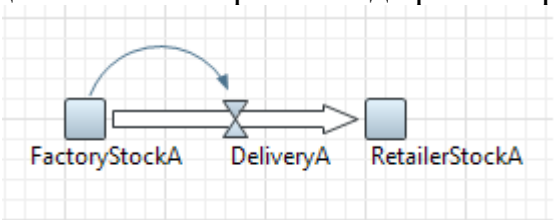
4.4 Добавление СД цепочки поставок для продукта A

Мы добавим оставшуюся часть цепочки поставок для продукта A, отвечающую за производство и доставку. Цепочка поставок будет задана диаграммой потоков и накопителей Системной динамики: начнется она с потока производства Production , ведущего в накопитель производителя FactoryStock, а затем, через поток доставки Delivery - в накопитель ритейлера RetailerStock. Вначале мы сделаем интенсивность производства постоянной, а потом - переменной, подстраивающейся под спрос на продукт. Это будет вторым местом взаимодействия системно-

динамической и агентной частей модели друг с другом. Чтобы обеспечивать спрос в системно-динамической части, мы добавим в популяцию потребителей функцию сбора статистики: она будет подсчитывать число потребителей, находящихся в состоянии *WantA*. Каждый день значение интенсивности производства будет пересчитываться заново в соответствии с этим числом.

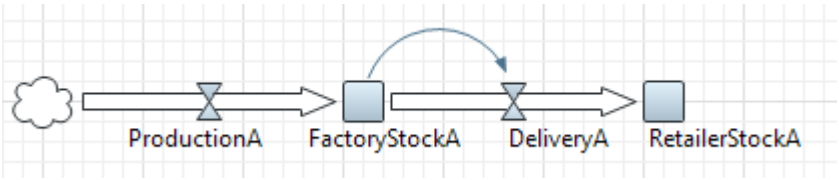
В редакторе класса *Main* добавьте накопитель *FactoryStockA* слева от накопителя *RetailerStockA*. Нарисуйте поток из *FactoryStockA* в *RetailerStockA*, сделав двойной щелчок по первому элементу и затем одиночный по второму. Назовите созданный поток *DeliveryA*. Задайте формулу **DeliveryA =** FactoryStockA / 2

Таким образом мы промоделируем двухдневную задержку, вызванную доставкой товара с завода ритейлеру.



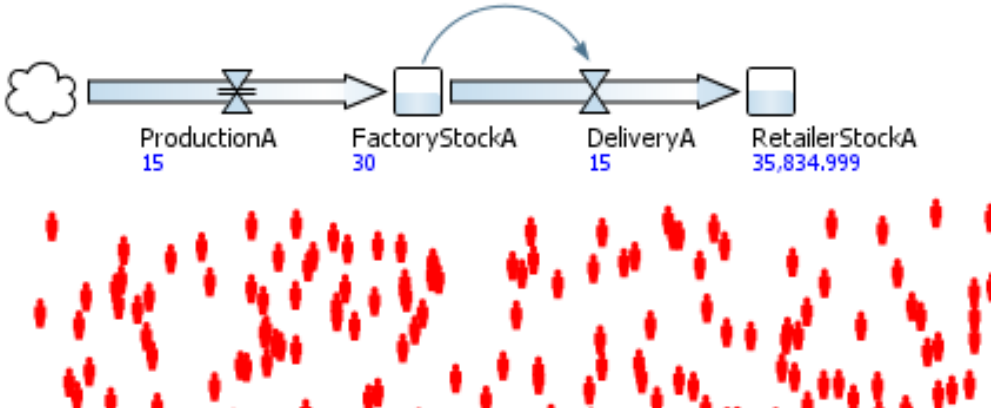
Добавьте поток *ProductionA* слева от *FactoryStockA* и сделайте его входящим в накопитель *FactoryStockA*. Пусть *ProductionA* = 15. Установите для этого потока флажок **Константа**, чтобы сделать значение этого потока постоянным.

На данный момент мы будем полагать интенсивность производства постоянной.



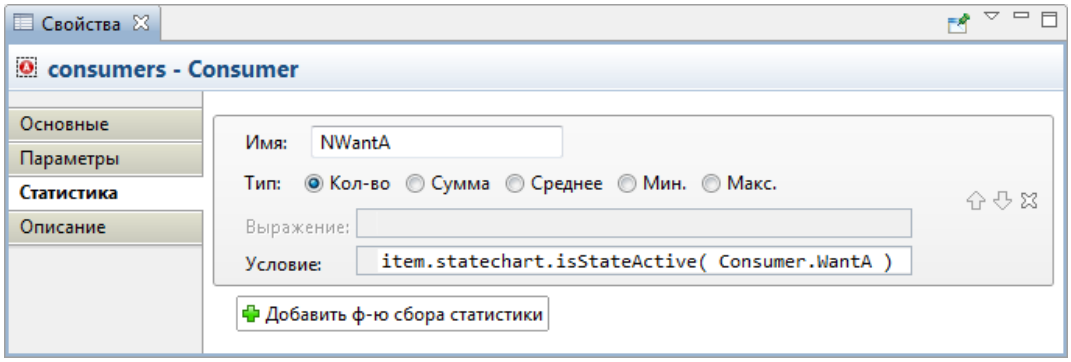
Запустите модель.

Вы увидите, что теперь спрос полностью удовлетворяется, и более того, товаров производится намного больше, чем нужно. Очевидно, что когда все станут пользователями продукта, цепочка поставок будет продолжать увеличивать значение товаров на складе ритейлера.



В редакторе класса *Main* щелкните по объекту *consumers* и перейдите на страницу свойств **Статистика** этого объекта. Добавьте функцию сбора статистики с именем *NWantA*. Оставьте выбранным **Тип** функции: *кол-во*. В качестве условия задайте: item.statechart.isStateActive(Consumer.WantA)

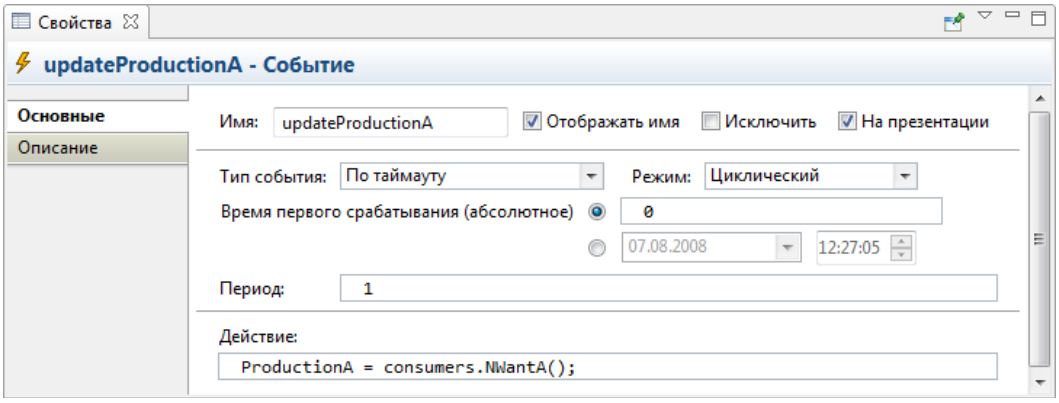
Поскольку потребительский рынок в данной модели полностью дезагрегирован, нам нужно пройти в цикле по всем агентам, чтобы сосчитать, сколько из них хотят приобрести продукт, т.е. находятся в состоянии *WantA*. Функция сбора статистики с типом *кол-во* делает именно это: проходит в цикле по популяции агентов и подсчитывает, для скольких из них выполняется заданное условие. В выражении условия item является рассматриваемым в текущий момент времени агентом; statechart - имя диаграммы состояний потребителя, isStateActive - стандартный метод диаграммы состояний, а WantA - имя состояния, заданного внутри агента, поэтому перед его именем и используется префикс item.



В редакторе класса *Main* добавьте событие рядом с потоком *ProductionA*. Назовите его *updateProductionA*. В свойствах этого события задайте **Тип события**: *По таймауту*, **Режим**: *Циклический*, **Время**

первого срабатывания (абсолютное): 1, **Действие**: ProductionA = consumers.NWantA();

Задав такое событие, мы добавим еще одну связь между системно-динамической и агентной частями модели: теперь в начале каждого дня интенсивность производства будет изменяться в соответствии с количеством потребителей, которые желают приобрести продукт. Можно использовать и более сложные формулы, чем просто “интенсивность = спрос”, но нас устраивает и этот простейший случай. NWantA() - это имя функции сбора статистики, которую мы задали на предыдущем шаге.

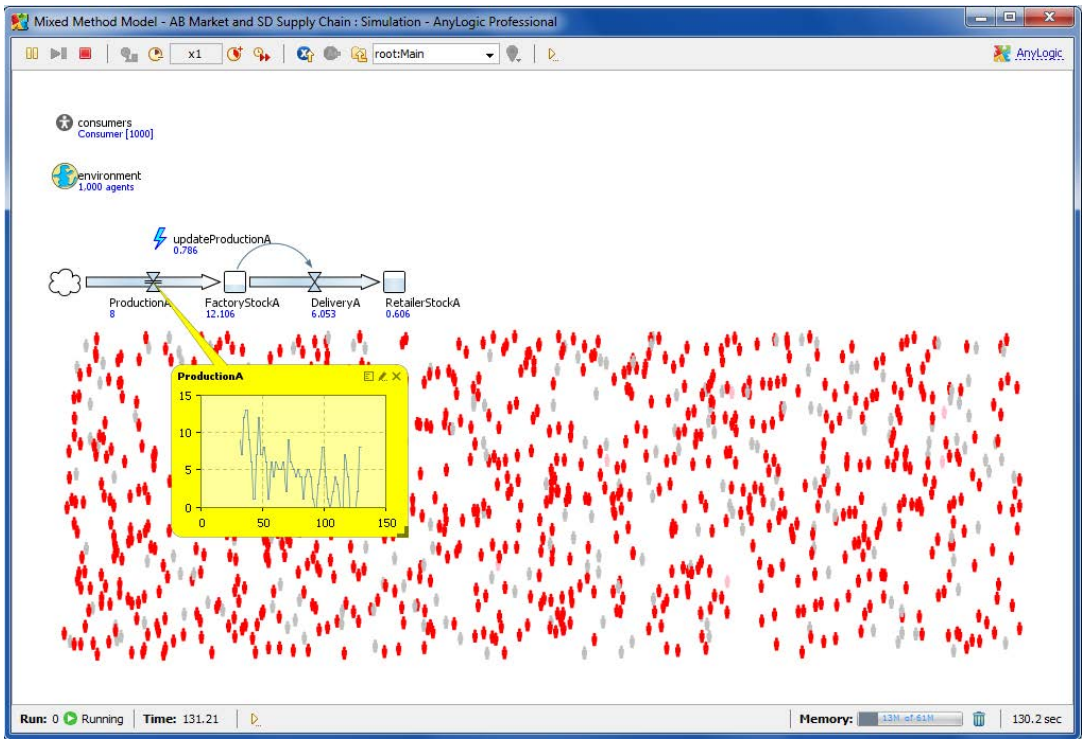


В формуле потока *ProductionA* удалите 15 и напишите 0.

Теперь интенсивность производства полностью управляется событием, поэтому формулу можно удалить.

Запустите модель. Щелкните мышью по значку потока *ProductionA*, чтобы показать окно инспекта для этого элемента. Затем переключите это окно в режим отображения графика, щелкнув по маленькой кнопке с изображением графика в верхнем правом углу окна инспекта.

Вы сможете увидеть, как колеблется интенсивность производства, что типично для цепочек поставок с их неизбежными задержками.



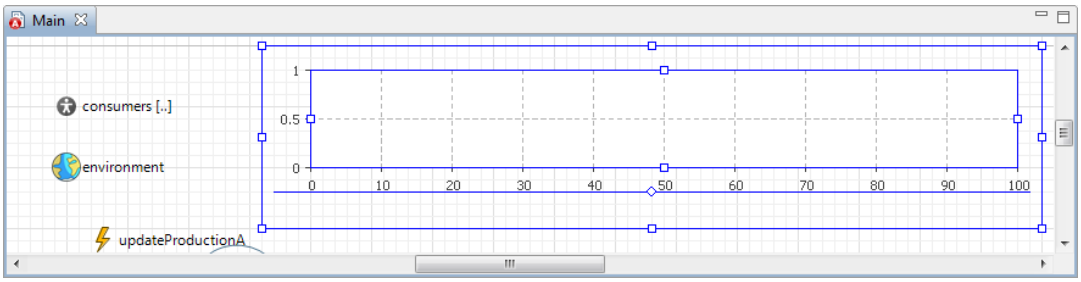
4.5 Визуализация динамики спроса и пользовательской базы

Сейчас давайте провизуализируем динамику порождаемого рынком неудовлетворенного спроса, а также численности пользователей продукта. Для этого нам нужно добавить функцию сбора статистики по численности пользователей продукта (а именно, функцию, подсчитывающую количество тех агентов, которые находятся в состоянии *UsesA*).

В редакторе класса *Main* щелкните по объекту *consumers* и перейдите на страницу его свойств **Статистика**. Добавьте еще один элемент сбора статистики NUseA типа **Кол-во** с **Выражением**: item.statechart.isStateActive(Consumer.UsesA) (которое можно скопировать из элемента *NWantA* и затем слегка изменить).

Перетащите элемент **Временная диаграмма с накоплением** из палитры **Статистика** на диаграмму класса *Main*

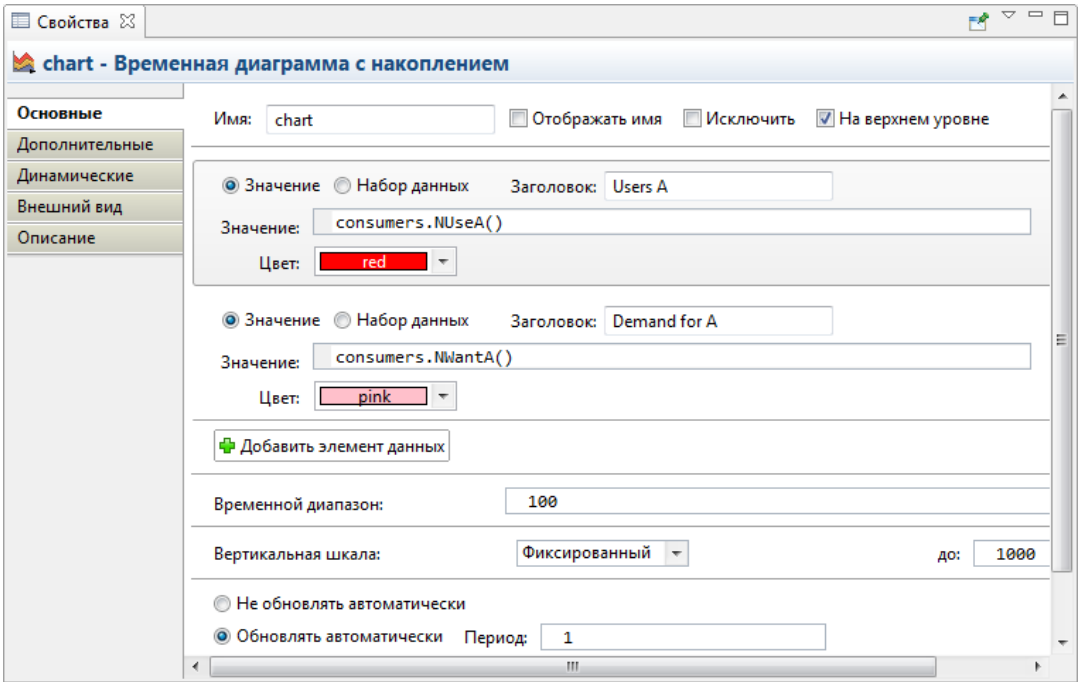
Временная диаграмма с накоплением является наиболее подходящим типом диаграммы для визуализации динамики изменения численностей долей населения во времени.



В свойствах диаграммы добавьте два новых элемента данных: один со **Значением** consumers.NUseA(), **Заголовком** *Users A* и красным цветом, а другой со **Значением** consumers.NWantA(), **Заголовком** *Demand for A* и розовым цветом.

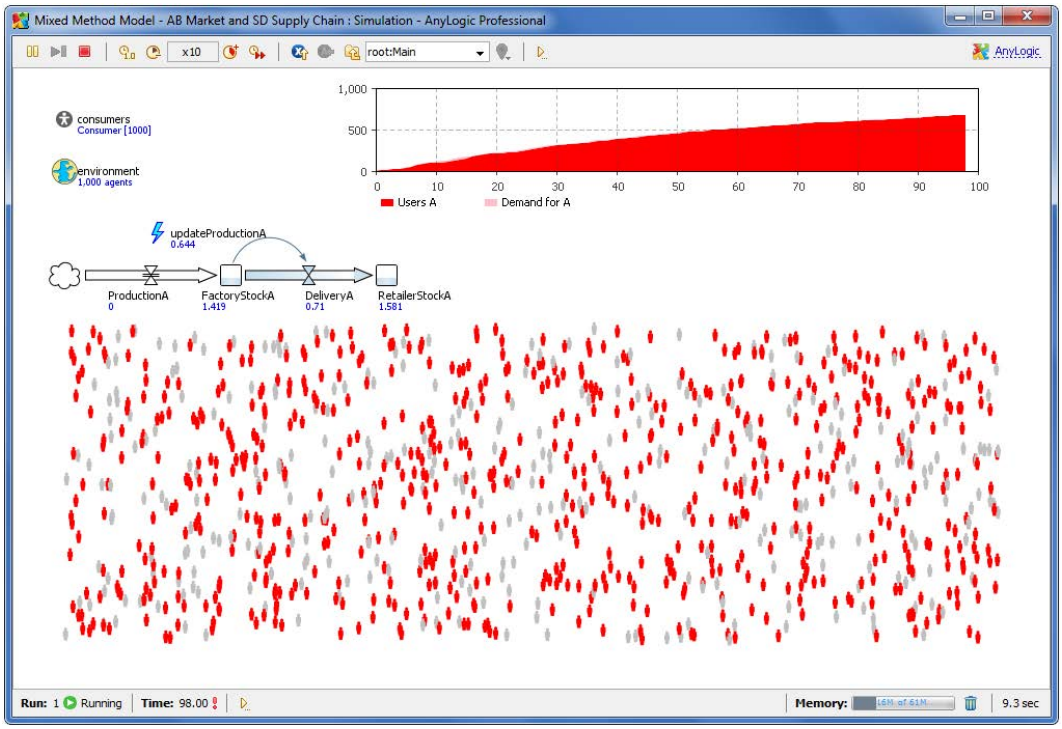
Сделайте вертикальную шкалу диаграммы **Фиксированной**, до *1000*.

Таким образом мы задаем, что будет показываться на диаграмме. По умолчанию временной диапазон диаграммы равен 100. Диаграмма будет автоматически обновлять значения каждую единицу модельного времени.



Запустите модель.

Вы увидите, как растет пользовательская база продукта A. Цепочка поставок работает хорошо, наблюдается лишь небольшой слегка колеблющийся неудовлетворенный спрос.



4.6 Учет общения потребителей друг с другом

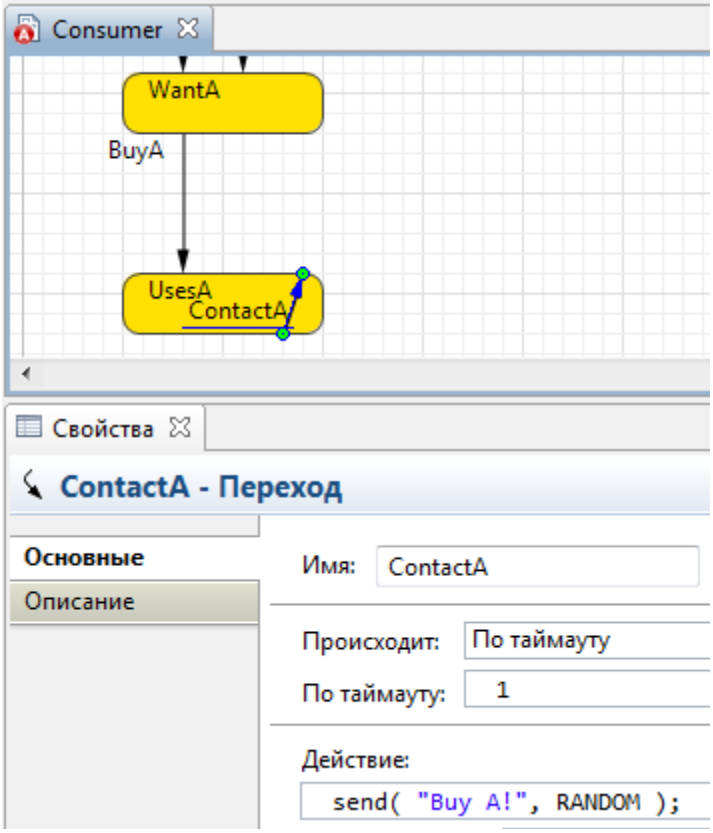
В этой фазе агенты начнут общаться друг с другом. Нас интересует общение пользователей продукта с потенциальными пользователями. Для этого мы добавим циклический переход внутрь состояния *UsesA*. Переход будет срабатывать периодически и при каждом своем срабатывании агент-пользователь будет посылать сообщение случайно выбранному агенту, говоря тому, что продукт А стоит того, чтобы его приобрести. Если этот случайно выбранный агент будет являться потенциальным пользователем (т.е. будет находиться в состоянии *PotentialUser*), то он отреагирует на получение такого сообщения переходом в состояние *WantA*. Для этого, очевидно, нужно будет добавить еще один переход из *PotentialUser* в *WantA*, который будет срабатывать по полученииии соответствующего сообщения.

Перейдите в редактор класса *Consumer* и нарисуйте переход внутри состояния *UsesA* так, чтобы он начинался и заканчивался на границе этого состояния и полностью находился внутри этого состояния. Назовите его *ContactA*. Оставьте без изменения свойства срабатывания этого перехода: *По таймауту* с таймаутом 1. Задайте **Действие**: send("Buy A!", RANDOM);

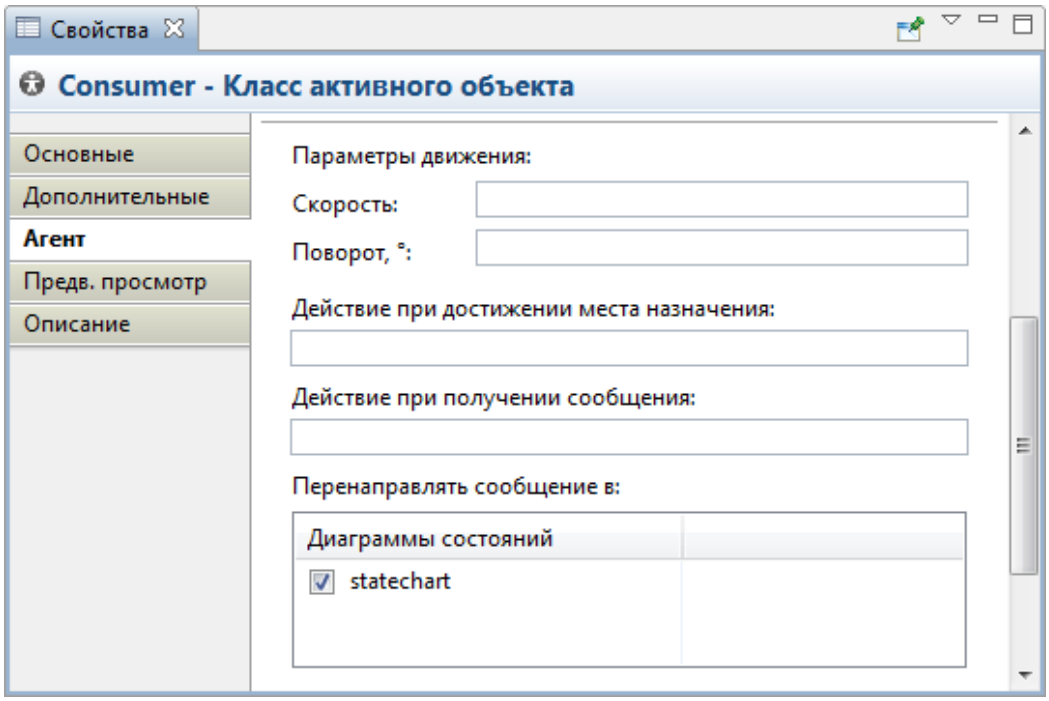
Внутренний переход является циклическим переходом, который не выводит диаграмму состояний из того состояния, в котором он задан.

В действии этого перехода потребитель случайно выбирает другого потребителя (не обязательно потенциального пользователя!) и посылает тому текстовое сообщение “Buy A!” (т.е."Купи продукт A!").

Переход *ContactA* сработает по истечении заданного таймаута. Мы предполагаем, что в течение дня пользователь убеждает одного человека. При желании Вы можете сделать этот переход срабатывающим с заданной интенсивностью. Вы можете непосредственно моделировать все контакты, тогда в случае, например, 5 контактов в день, интенсивность контактов будет равна 5. Поскольку не все контакты успешны, т.е. убеждают потенциальных пользователей приобрести продукт A, то Вы можете моделировать только “успешную долю” этих контактов и сделать их более редкими, умножив их интеннсивность на коэффициент, задающий силу убеждения, скажем, 0.015. Мы же используем срабатывающий по таймауту переход, потому что такой подход нам кажется проще.



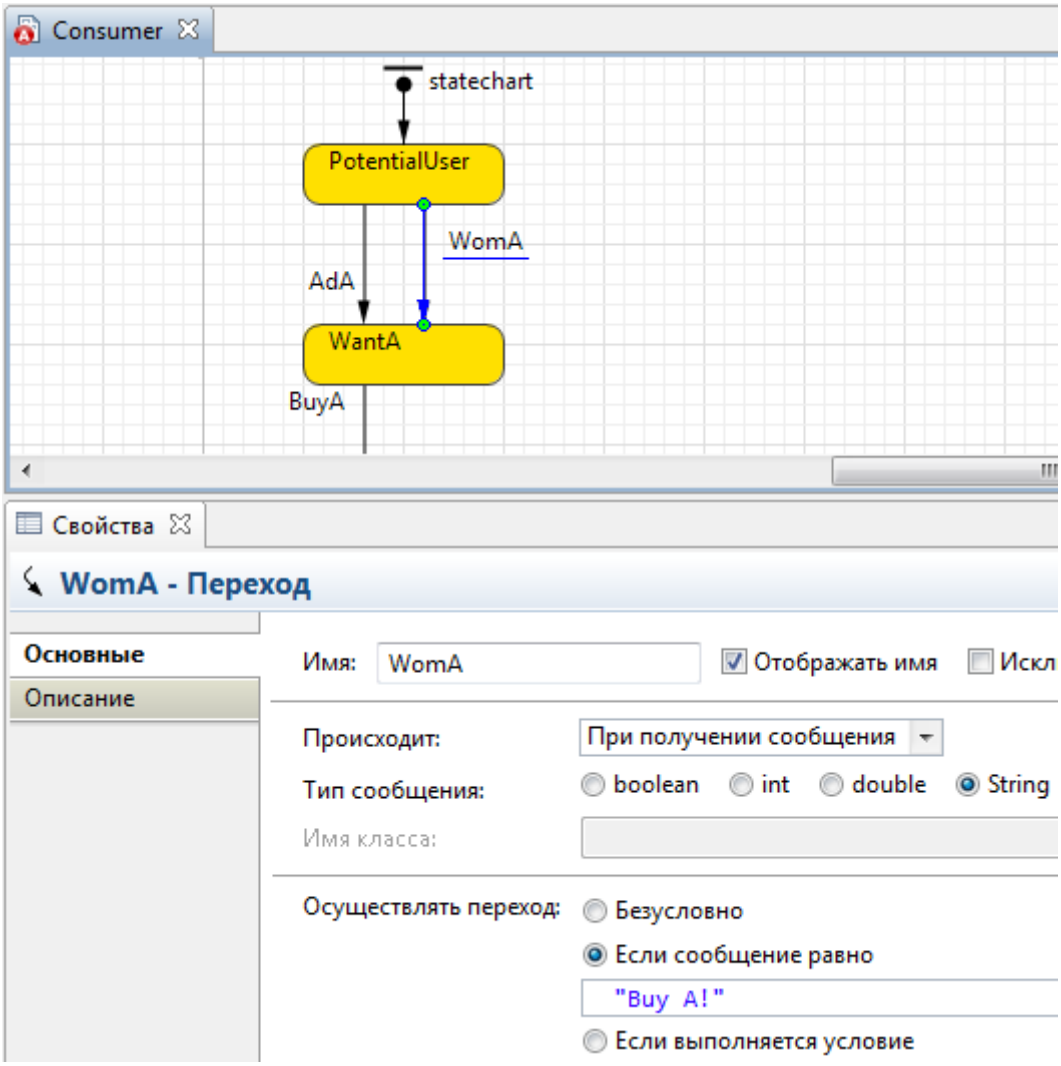
Перейдите на страницу свойств **Агент** класса *Consumer*. В таблице **Перенаправлять сообщение в** выберите флажок в строке, соответствующей диаграмме состояний *statechart*:



На предыдущем шаге мы сделали так, что агенты-пользователи продукта теперь периодически контактируют с другими потребителями, отсылая им сообщения “Buy A!” (т.е."Купи продукт A!"). В секции свойств агента **Перенаправлять сообщение в** мы выбираем диаграммы состояний, которые будут получать и обрабатывать получаемые агентом сообщения. То есть мы перенаправляем полученное сообщение в диаграмму состояний *statechart*. Для реакции на это сообщение мы добавим в нашу диаграмму состояний еще один переход.

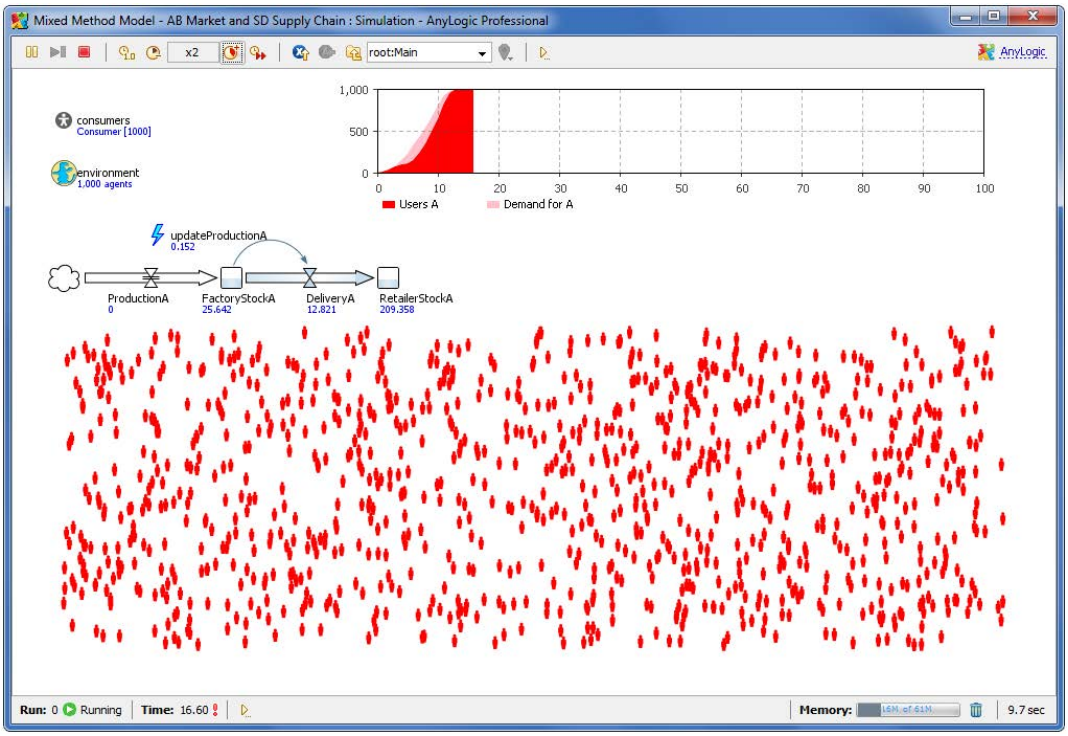
Добавьте еще один переход в диаграмму состояний класса *Consumer*: из состояния *PotentialUser* в состояние *WantA*, назовите его *WomA*. Пусть этот переход срабатывает *По получении сообщения*, типом сообщения пусть будет **String**, а само сообщение должон соответствовать строке “Buy A!”

Мы закончили моделирование общение людей, в результате которого происходит убеждение в необходимости покупки продукта. Сообщение, приходящее от другого агента, перенаправляется в диаграмму состояний, и если диаграмма находится в состоянии *PotentialUser*, то в результате получения такого сообщения произойдет мгновенный переход в состояние *WantA*. Если потребитель будет находиться в каком-то другом состоянии, то сообщение будет проигнорировано.



Запустите модель.

Вы увидите, что насыщение рынка теперь происходит намного быстрее. Диаграмма показывает широко известную кривую распространения продукта S-формы. Неудовлетворенный спрос во время пикового интереса к продукту довольно незначителен.

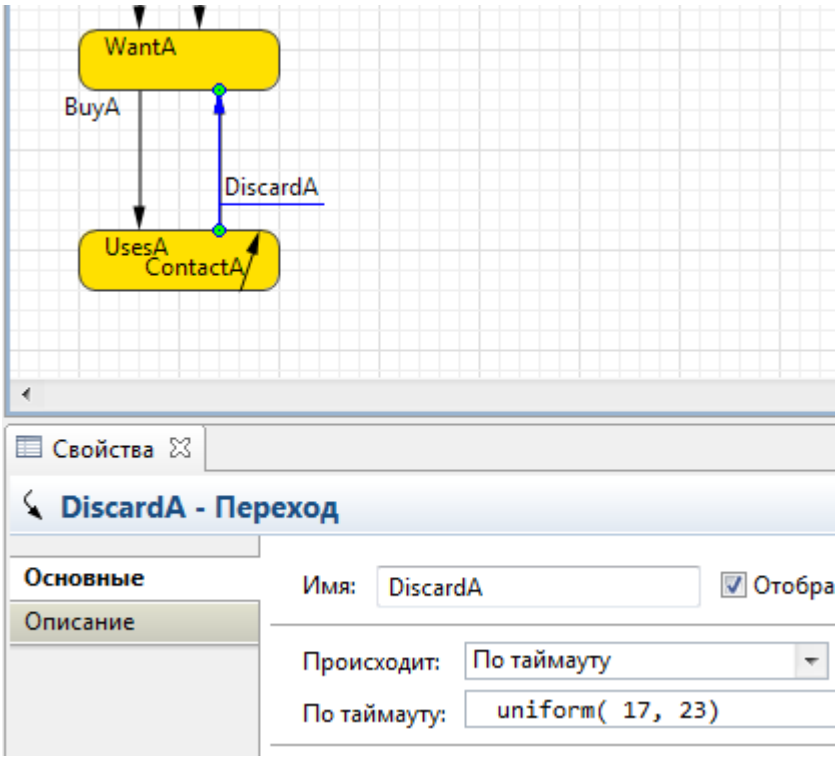


4.7 Добавление повторных покупок продукта

Это очень простая фаза. Поскольку спустя определенное время продукт приходит в негодность, и пользователю нужно приобрести ему замену, то мы добавим переход из состояния *UsesA* в состояние *WantA*, срабатывающий по постоянному таймауту DiscardTime. Добавив такой переход, мы ограничиваем время каждого пребывания в состоянии *UsesA* временем DiscardTime.

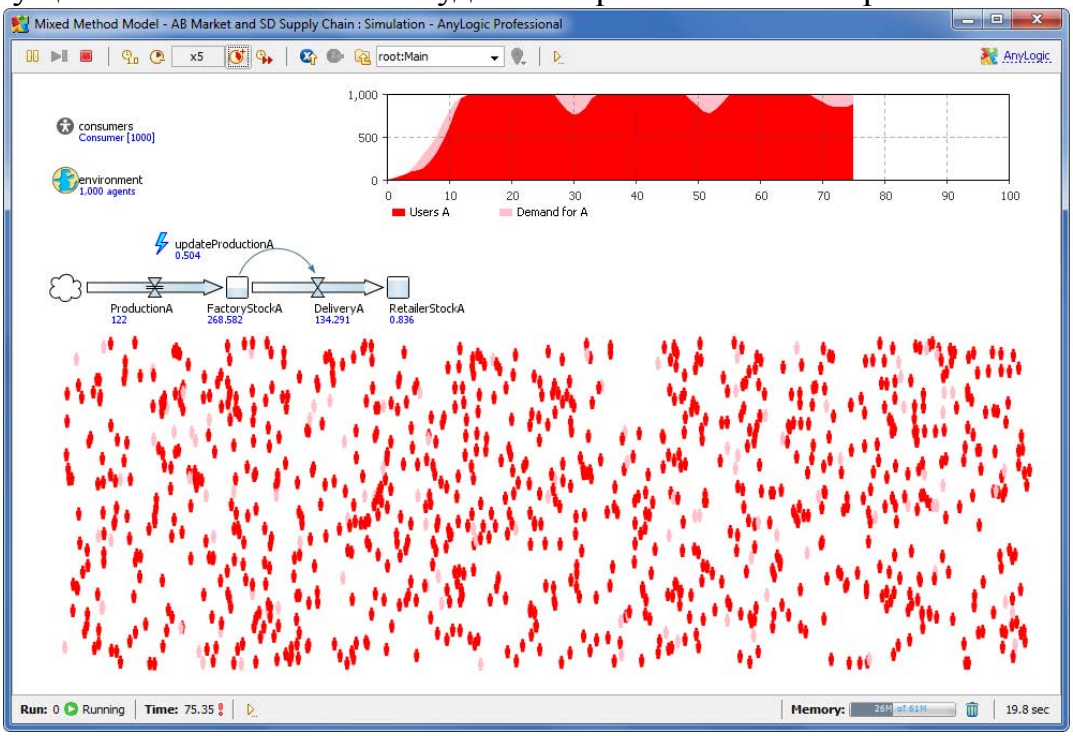
Добавьте в диаграмму состояний потребителя переход, ведущий из состояния *UsesA* в состояние *WantA*, назовите его *DiscardA*. Пусть этот переход **Происходит** *По таймауту*, с таймаутом uniform(17, 23).

Когда управление диаграммы состояний перейдет в состояние *UsesA*, будет вычислено значение стохастического таймаута, и начнется обратный отсчет для перехода *DiscardA*. Время использования определенной единицы товара (т.е. время, проведенное в состоянии *UsesA*) поэтому будет равномерно распределено между значениями 17 и 23. Обратите внимание, что срабатывание перехода *ContactA* не будет сбрасывать текущее значение таймаута, поскольку этот переход является внутренним. После срабатывания перехода *DiscardA* потребитель перейдет в состояние *WantA*, что будет означать, что он желает немедленно приобрести замену пришедшему в негодность продукту.



Запустите модель.

После наступления насыщения рынка (когда все потребители являются владельцами продукта), можно наблюдать периодически порождаемый цепочкой поставок дефицит товара, вызванный прихождением товара в негодность у большинства пользователей. Цепочка поставок в ее текущем состоянии не может удовлетворить эти пики спроса.



4.8 Добавление в модель продукта B

Теперь давайте добавим в модель продукт B, конкурирующий с продуктом A. Мы сделаем это простым копированием цепочки поставок для продукта A и переименованием ее элементов. Зависящая от типа продукта часть диаграммы состояний потребителя также будет скопирована, так, что при выходе из состояния *PotentialUser* будет две альтернативы: переход в состояние *WantA* или состояние *WantB*. На данном этапе мы не будем учитывать максимально возможное время ожидания продукта. Для того, чтобы отобразить динамику изменившегося рынка, нам также придется добавить на нашу диаграмму два новых элемента данных.

В редакторе класса *Consumer* выделите два состояния *WantA* и *UsesA* и все переходы (другими словами, выделите все элементы диаграммы состояний за исключением состояния *PotentialUser* и начала диаграммы состояний). Сделайте копию выделенных элементов и расположите переходы так, как показано на рисунке ниже.

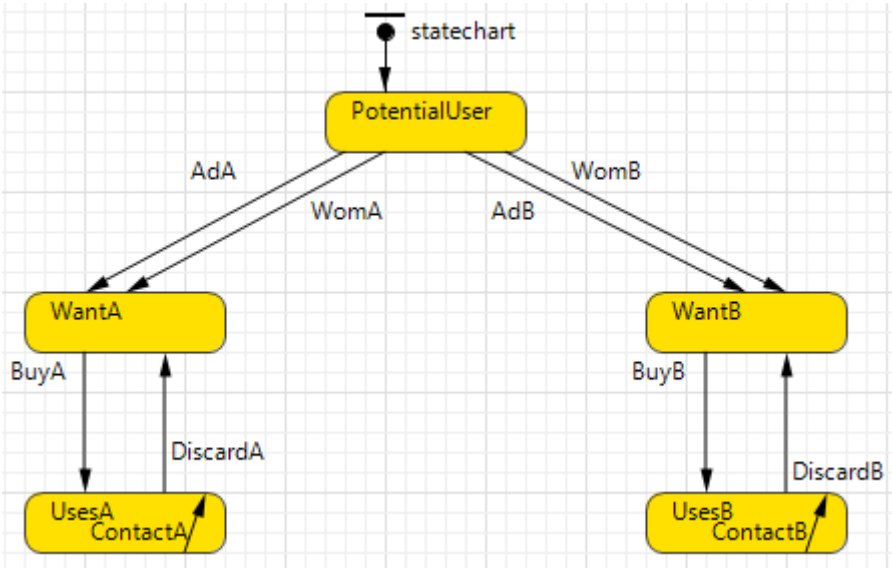
Переименуйте “*A1*” в окончаниях имен скопированных элементов на “*B*”.

В переходах *WomB* и *ContactB* поменяйте “Buy A!” на “Buy B!”.

В свойствах перехода *BuyB* поменяйте *RetailerStockA* на *RetailerStockB.*

В полях **Действие при входе** состояний *WantB* и *UsesB* измените цвета на *lightBlue* и *blue*.

Только что мы создали альтернативный путь в поведении потребителя. Мы предполагаем для продукта B такую же эффективность рекламы (0.011), такую же интенсивность общения и силу убеждения (хотя мы, конечно, могли бы изменить значение любого из этих параметров). Обратите внимание, что структура новой диграммы состояний позволяет потенциальному пользователю делать выбор продукта только в самом начале; после того, как выбор был сделан, поменять его нельзя.



В редакторе класса *Main* щелкните по объекту *consumers* и перейдите на страницу свойств **Статистика** этого объекта. Добавьте еще два элемента для сбора статистики: *NWantB* and *NUseB* с условиями

item.statechart.isStateActive(Consumer.WantB) и

item.statechart.isStateActive(Consumer.UsesB) соответственно.

Эти элементы сбора статистики будут подсчитывать спрос на продукт B и количество пользователей продукта B.

В редакторе класса *Main* выделите всю цепочку поставок для продукта A (все элементы системной динамики), а также событие *updateProductionA,* скопируйте все эти элементы и вставьте их справа. Переименуйте:

событие updateProductionA1 на updateProductionB, накопитель ProductionA1 на ProductionB и т.д.

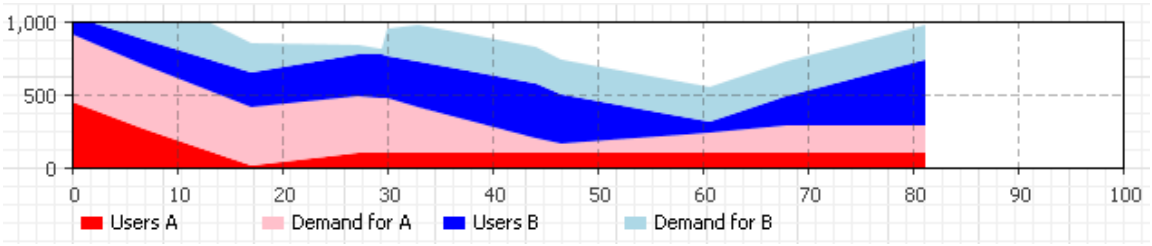
Измените действие *updateProductionB* на ProductionB = consumers.NWantB();

Поменяйте A на B во всех формулах элементов цепочки поставок для продукта B.

Излишне говорить, что такое копирование и переименование нельзя назвать элегантным способом расширения модели. И естественно, AnyLogic предлагает куда более правильный способ реплицирования

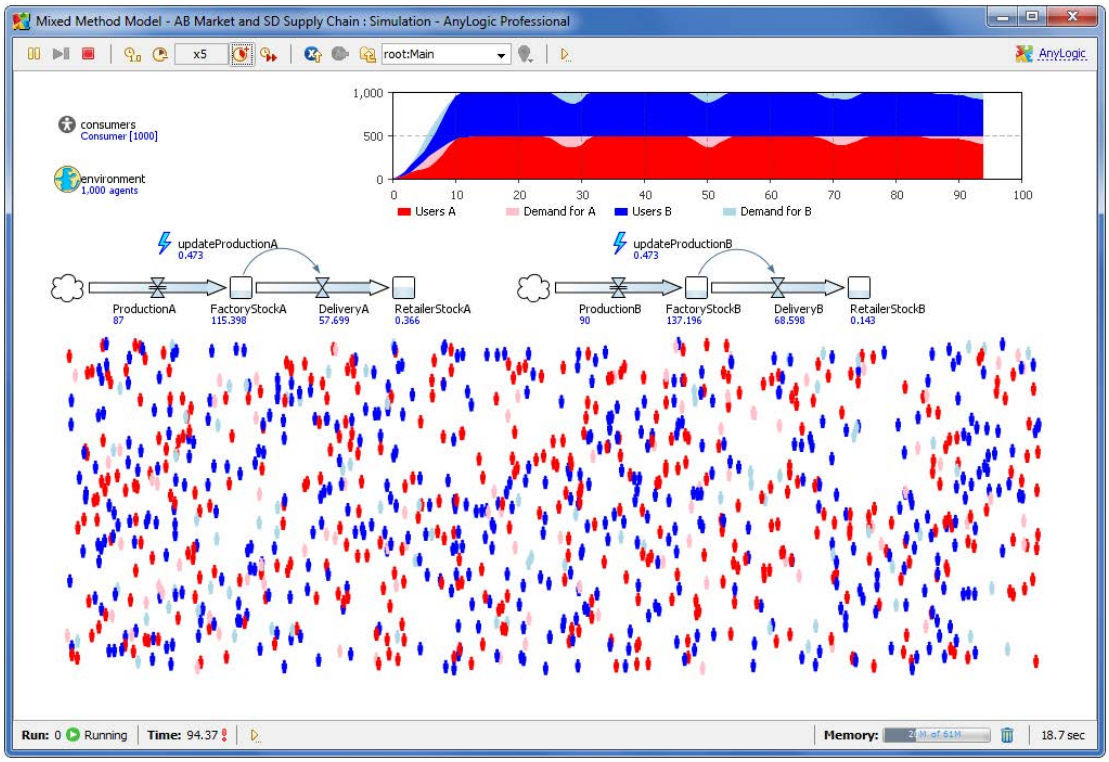
отдельных частей модели: Вы можете поместить цепочку поставок и соответствующее событие в новый класс активного объекта, вынести в параметры класса, например, спрос, и создать два экземпляра этого класса в классе *Main*: один для продукта A и один для B. Но мы не будем делать этого в целях сохранения простоты этого учебного пособия.

В редакторе класса *Main* добавьте на диаграмму два новых элемента данных. Задайте у этих элементов **Значения** consumers.NWantB() и consumers.NUseB() и **Заголовки** “*Demand for B*” и “*Users B*”.



Запустите модель.

Вы увидите одинаковую динамику распространения для продуктов A и B. Обратите внимание, что доля рынка определяется только начальным выбором того или иного продукта, и не может быть изменена впоследствии в силу текущей структуры диаграммы состояний потребителя.

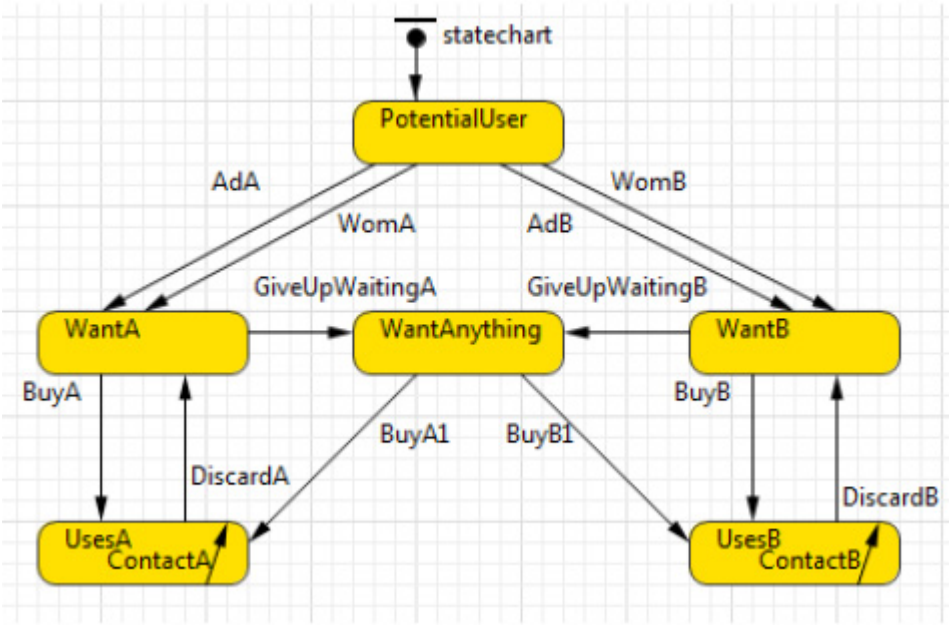


4.9 Учет смены предпочитаемого продукта при долгом ожидании

Согласно нашей постановке задачи, если потребитель слишком долго безуспешно ждет появления в продаже какого-то определенного продукта, то он теряет терпение и согласен приобрести любой продукт (A или B), который быстрее появится в наличии, поэтому становится возможной смена предпочтений пользователя. Чтобы учесть такое поведение, достаточно лишь слегка изменить диаграмму состояний потребителя: нужно добавить два срабатывающих по таймауту перехода из состояний *WantA* и *WantB* в новое состояние *WantAnything*. В свою очередь, из этого состояния в состояния *UsesA* и *UsesB* будут вести два альтернативных перехода, срабатывающих при покупке продукта того или иного типа.

Добавьте новое состояние *WantAnything* в середину диаграммы состояний потребителя. Нарисуйте два перехода, ведущие в это состояние из состояний *WantA* и *WantB* соответственно. Назовите их *GiveUpWainingA* и *GiveUpWaitingB*. Оба перехода должны срабатывать **По таймауту**, с таймаутом, равным *2* единицам времени. Создайте копию перехода *BuyA* (этот новый переход будет называться *BuyA1*) и

сделайте так, чтобы он вел из состояния *WantAnything* в состояние *UsesA*. Аналогичным образом создайте копию перехода *BuyB*. Оставьте свойства срабатывания переходов без изменений. В свойстве **Действие при входе** состояния *WantAnything* напишите person.setFillColor(gold);



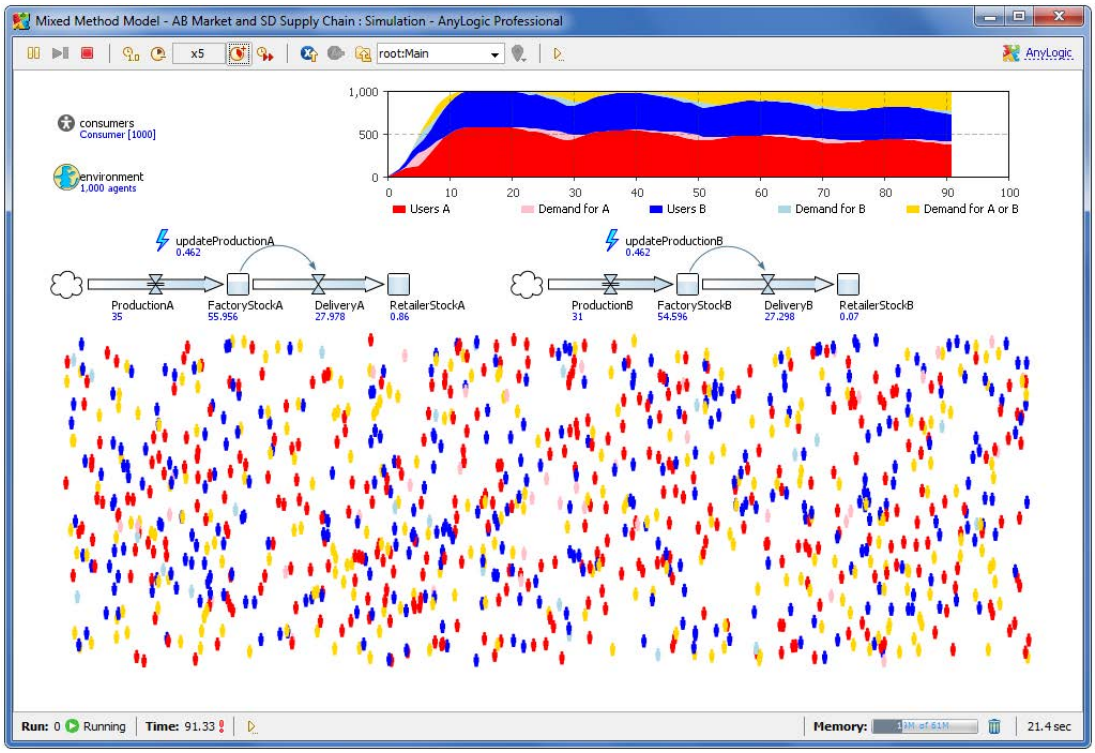
Срабатывающие по истечению таймаута переходы ограничивают время ожидания определенного продукта до 5 единиц времени. Перейдя в состояние *WantAnything*, потребитель начинает отслеживать доступность обоих продуктов и купит тот продукт, который появится в наличии первым.

В редакторе класса *Main* щелкните по объекту *consumers* и добавьте еще одну функцию сбора статистики (на странице свойств **Статистика**). Назовите ее *NWantAny* и задайте условие item.statechart.isStateActive(item.WantAnything).

Добавьте на диаграмму еще один элемент данных со **Значением** consumers.NWantAny() и золотым цветом *gold*.

Запустите модель.

Вы увидите, что с течением времени растет неудовлетворенный спрос. Это вызвано неточностями вычисляемых цепочками поставок оценок спроса: интенсивность производства для каждого продукта зависит только от количества потребителей, ожидающих появления этого определенного продукта, и не принимает во внимание тех потребителей, которые находятся в состоянии *WantAnything*.



Измените действия событий *updateProductionA* и *updateProductionB* следующим образом: напишите

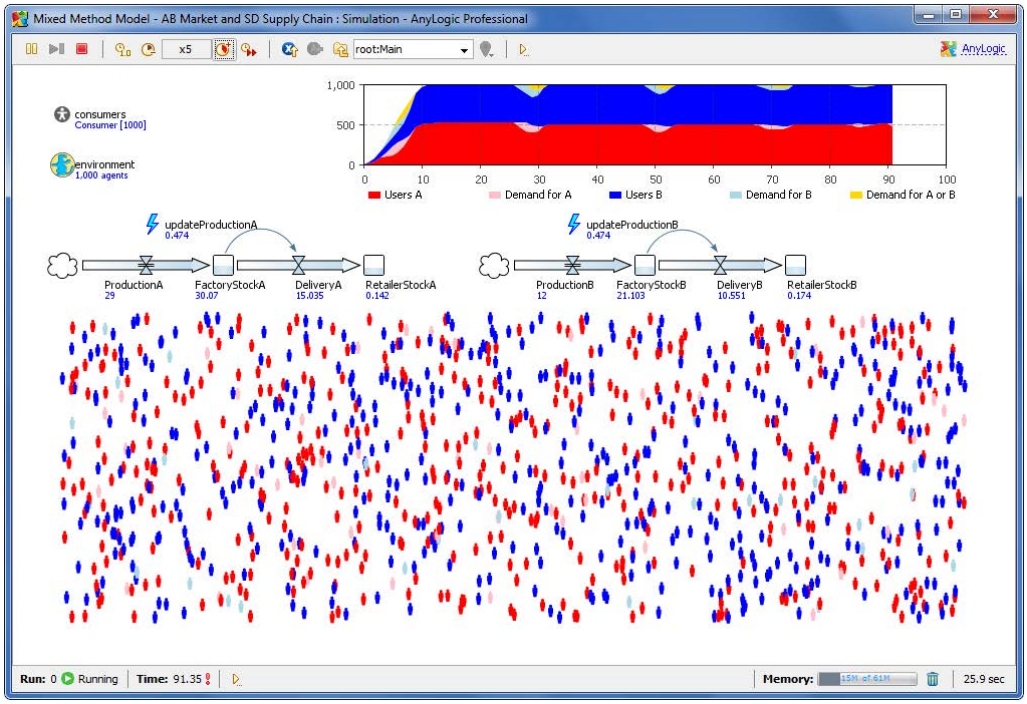
ProductionA = consumers.NWantA() + consumers.NWantAny(); и

ProductionB = consumers.NWantB() + consumers.NWantAny();

Таким образом мы учитываем при анализе спроса и тех потребителей, которые готовы приобрести любой продукт.

Запустите модель.

Теперь цепочки поставок работают лучше, хотя периодически возникают непродолжительные периоды дефицита. Вы также можете увидеть, что доли рынка немного колеблются, т.е. у потребителей происходит смена предпочтений. Но в целом отношение долей рынка для двух продуктов остается примерно равным 50/50, потому что модель абсолютно симметрична.



4.10 Проведение эксперимента

На данный момент модель полностью соответствует нашей постановке задачи. Теперь Вы можете провести с ней любые эксперименты на Ваше усмотрение.

Например, Вы можете изменить политики цепочек поставок и изучить, как они влияют на итоговое распределение долей рынка. Может иметь смысл отслеживать не только удовлетворение спроса, но также и уровни запасов в накопителях (которые, очевидно, хочется держать на минимальном уровне).

Цепочка поставок может быть промоделирована не с помощью системной динамики, а с помощью дискретно-событийного подхода моделирования (путем использования объектов Основной библиотеки AnyLogic). В этом случае будет намного проще моделировать время доставки и может в целом стать легче более детально промоделировать процесс.

Другие интересные эксперименты могут быть проведены с поведением потребителя и с численностью потребителей. Вы можете добавить в модель социальную сеть, контакты в которой могут происходить только между людьми, которые знают друг друга. Это очевидно повлияет на процесс распространения продукта. Вы можете сделать потребителей неидентичными друг другу, добавив им параметры (например, различную подверженность рекламе и чужому мнению, различные уровни лояльности брендам) или внеся изменения на уровне поведения (например, некоторые потребители могут разочароваться в продукте, распространять негативные отзывы о нем и т.д.).

СПИСОК ЛИТЕРАТУРЫ

1. Бир С. Кибернетика и управление производством: пер. с англ. – М.: Наука, 1985. – 391с.

2. Шеннон Р. Имитационное моделирование систем – искусство и наука: пер. с англ. – М.: Мир, 1978. – 418 с.

3. Бусленко Н.П. Моделирование сложных систем. – М.: Наука, 1978. – 399 с.

4. Соломатин Н.А. Имитационное моделирование в оперативном управлении производством/ Н.А. Соломатин, Г.В. Беляев, В.Т. Петроченко. - М.: Машиностроение, 1984. – 208 с.

5. Киндлер Е. Языки моделирования: пер. с чеш. – М.: Энергоатомиздат, 1985. – 288 с.

6. Нейлор Т. Машинные имитационные эксперименты с моделями экономических систем: пер. с англ. – М.: Мир, 1975. – 500 с.

7. Прицкер А. Введение в имитационное моделирование и язык СЛАМ II: пер. с англ. – М.: Мир, 1987. – 646 с.

8. Томашевский В.Н. Имитационное моделирование в среде GPSS/ В.Н. Томашевский, Е.Г. Жданова. – М.: Бестселлер, 2003. – 416 с.

9. Клейнрок Л. Теория массового обслуживания // М.: Машиностроение. 1979. – 432 с.

10. Советов Б.Я. Моделирование систем / Б.Я. Советов, С.А. Яковлев. – М.: Высшая школа, 1985. – 217 с.

11. Форрестер Дж. Основы кибернетики предприятия: Индустриальная динамика: пер. с англ. – М.: Прогресс, 1971. – 340 с.

12. Емельянов А.А. Имитационное моделирование экономических процессов: Учеб. пособие / А.А. Емельянов, Е.А. Власова, Р.В. Дума; Под ред. А.А. Емельянова. – М.: Финансы и статистика, 2002. – 368 с: ил.

13. Лоу А М., Кельтон В.Д. Имитационное моделирование. 3-е издание. – СПб.: Питер, Киев: BHV, 2004. – 847 с.

14. Рыжиков Ю. И. Имитационное моделирование. Теория и технология. – СПб.: КОРОНА принт, 2004. – 384 с.

15. Аристов С.А. Имитационное моделирование экономических систем: Учеб. пособие. – Екатеринбург: Изд-во Урал. гос. экон. ун-та. 2004. – 123 с.

16. Карпов Ю. Имитационное моделирование систем. Введение и моделирование с AnyLogic 5. – СПб.: БХВ-Петербург, 2005. – 400 с: ил.

17. Киселева М. В. Имитационное моделирование систем в среде AnyLogic: учебно- методическое пособие / М.В. Киселёва. Екатеринбург: УГТУ – УПИ, 2009. – 88 с.

18. Каталевский Д.Ю. Основы имитационного моделирования и системного анализа в управлении: Учебное пособие. – М.: Издательство Московского университета, 2011. – 304 с., ил.

19. Осоргин А.Е. AnyLogic 6. Лабораторный практикум / А.Е.Осоргин. – Изд. 2-е, перераб. и доп. – Самара: ПГК, 2012. – 110 с.

Михайлов Валерий Николаевич

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ

Учебно-методическое пособие

Подписано в печать 30.12.2014 г. Формат 60x84 1/16.
Бумага офсетная. Печать ризография. Усл. печ. л 10,25.
Тираж 200 экз. (1-й завод – 50 экз.) Заказ № 127.
Отпечатано с готового оригинал-макета в издательстве ОФ РАНХиГС
г. Орел, ул. Панчука, 1.