[Рынок и цепочка поставок (Агенты + СД)](http://127.0.0.1:64791/help/nav/1_3)

**Введение**

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 и выходит за рамки этого учебного пособия. Это же пособие призвано научить пошагово создавать модель, совместно использующую системную динамику и агентное моделирование. Создав такую модель, впоследствии Вам будет легче строить различные многоподходные модели.  
  
Учебное пособие подразумевает определенную степень знакомства со средой разработки моделей AnyLogic: инструкции в большинстве случаев даются в стиле: “Создайте состояние PotentialUser”, а не “Откройте палитру **Диаграмма состояний**, перетащите элемент **Состояние** из палитры на диаграмму графического редактора и смените имя состояния на PotentialUser”.

**Постановка задачи**

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.e. 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.

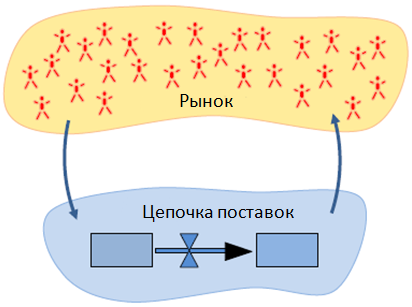
У каждой компании (и A, и B) есть своя цепочка поставок, используемая для поставки товаров конечным пользователям. Цепочки поставок достаточно просты и работают следующим образом:

* 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.e. количество человек, которые хотят приобрести продукт(ы), но не могут этого сделать в силу их отсутствия у ритейлеров), а также уровни запасов цепочек поставок.

**План**

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



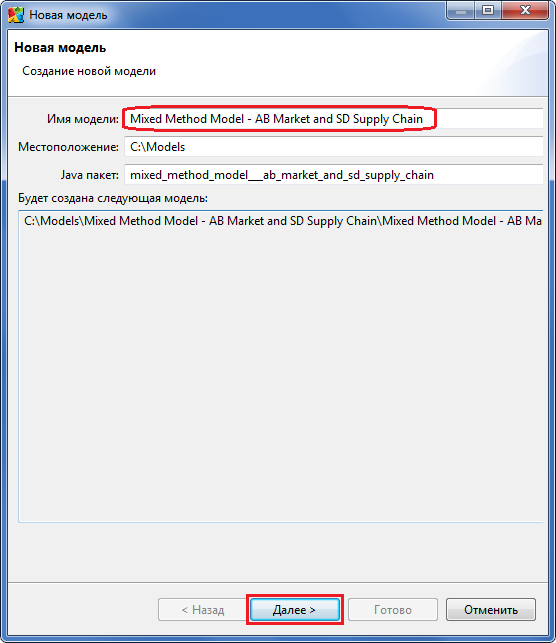
Лучше всего создавать имитационные модели итеративно, т.e. за несколько фаз, в конце каждой из которых создается готовая к запуску модель. В нашем случае имеет смысл сделать это в следующем порядке:

1. Начнем создание модели рынка с рассмотрения только одного продукта (A)
2. Построим модель поведения потребителя
3. Населим рынок потребителями
4. Предположим, что продукт есть в наличии
5. Добавим цепочку поставок для продукта A
6. Добавим продукт B

**Шаг 1. Создание 1000 агентов**

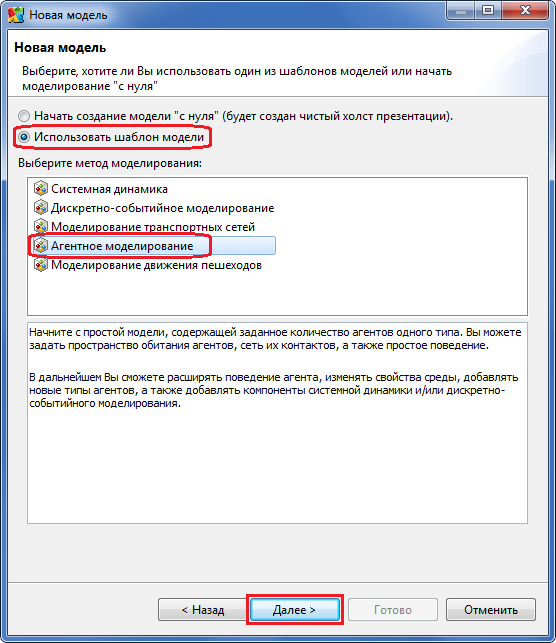
В этой фазе мы создадим новую модель AnyLogic, воспользовавшись готовым шаблоном агентной модели. В итоге мы получим модель, в которой будут моделироваться потребители численностью 1000 человек (с простой анимацией каждого потребителя). На начальном этапе у потребителей не будет задано никакого поведения, и поэтому пока что в этой модели ничего не будет происходить.

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

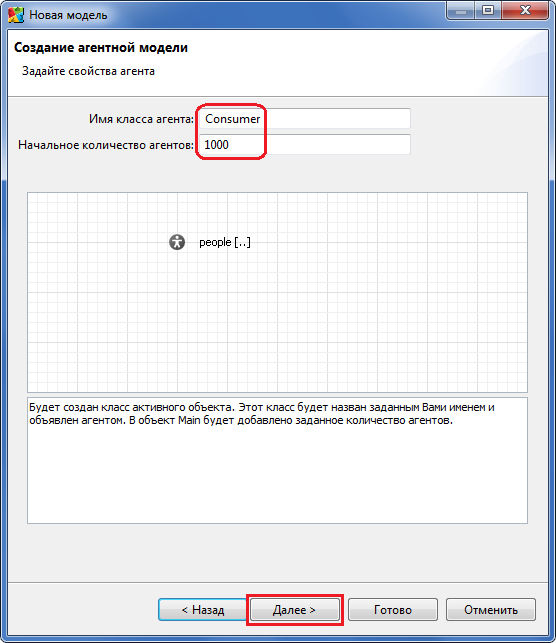


Когда Вы создаете новую модель, AnyLogic предлагает Вам на выбор набор шаблонов готовых моделей, с помощью которых Вы можете минимизировать количество действий, требуемых при начальном построении модели. Каждый такой шаблон содержит базовые конструкции, которые Вам могут обычно понадобиться при создании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

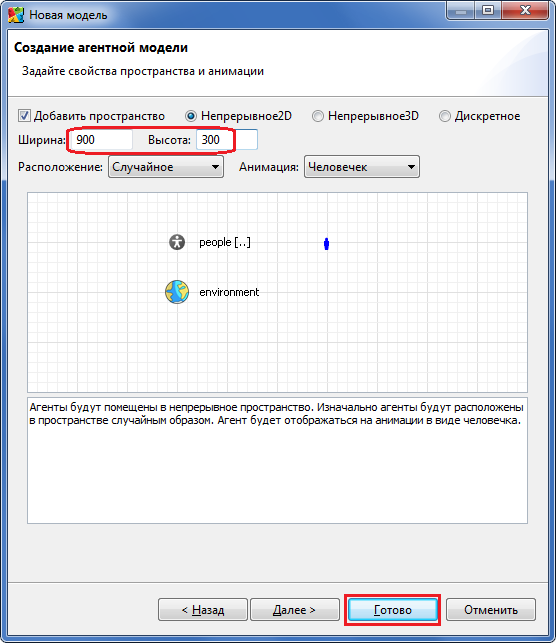
1. Выберите шаблон **Агентное моделирование** и перейдите к следующей странице мастера для конфигурации создаваемой модели.



1. 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*.

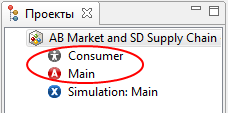


1. На следующей странице Мастера создания модели, оставьте в поле **Анимация**: **Человечек**. Здесь Вы выбираете фигуру, которая будет визуально представлять потребителя на анимации модели.   
   Измените размер прямоугольной области, в которой будут отображаться агенты на анимации. Задайте **Ширину** 900 и **Высоту** 300.



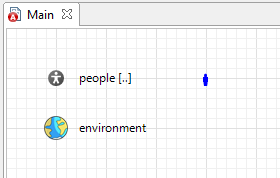
1. Обычно управляемое событиями поведение агента задается с помощью диаграммы состояний, и Мастер создания моделей предлагает также создать и диаграмму состояний, задающую весьма простое поведение. Но мы не будем использовать эту опцию, поскольку мы хотим нарисовать диаграмму состояний с нуля.  
   Завершите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



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

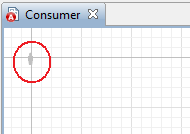
Местоположение этой фигуры анимации будет задавать верхний левый угол прямоугольной области, в которой позднее будет визуализироваться население потребителей. Остальные элементы модели мы будем располагать над этой областью.

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.

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

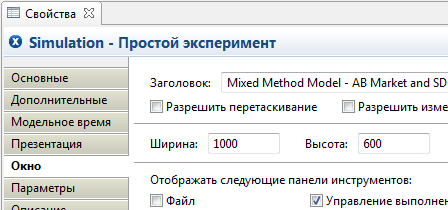
Элемент **Среда** поддерживает различные типы пространств, расположение агентов в этих пространствах, сети и взаимодействие агентов. В нашем случае среда нужна нам для того, чтобы расположить анимации агентов и чтобы промоделировать общение людей друг с другом с помощью встроенного механизма агентного взаимодействия. Этот элемент указан в качестве среды для агентов, заданных реплицированным объектом *consumers* (имя этого элемента указано в свойстве **Среда** объекта *consumers*).

1. 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.



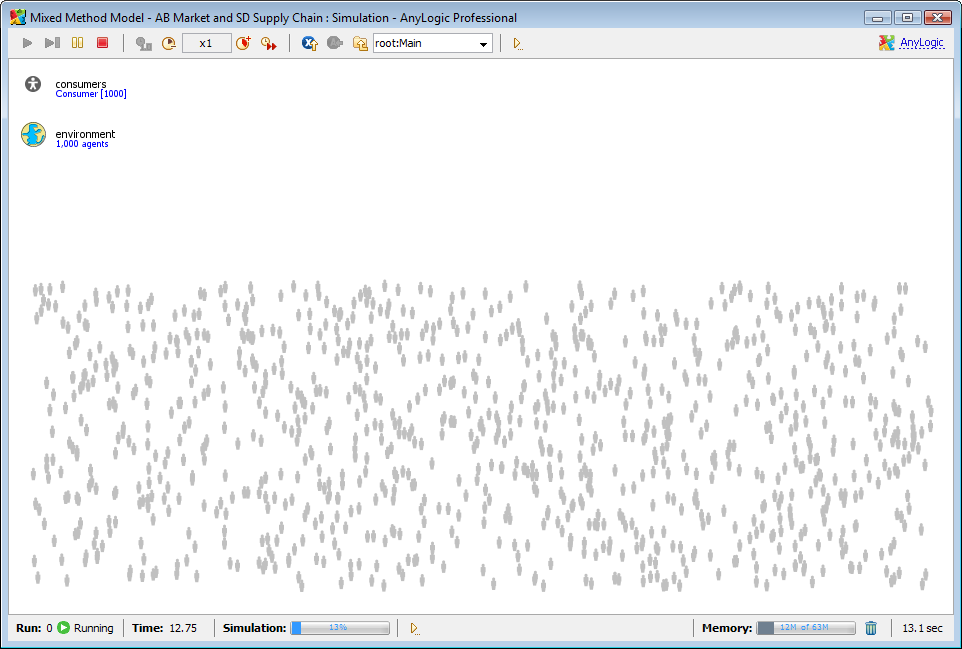
1. Щелкните в дереве проектов по элементу эксперимента *Simulation:Main*, перейдите на страницу его свойств **Окно** и задайте размер окна презентации, введя в поле **Ширина** 1000, а в поле **Высота** 600.

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



1. Запустите модель.

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

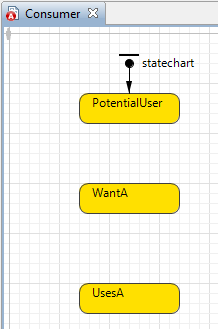


**Шаг 2. Задание простого поведения агента**

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.

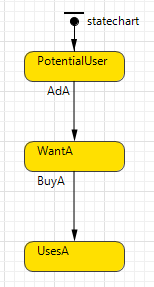
1. Нарисуйте в редакторе класса *Consumer* три состояния (расположите их снизу вверх в следующем порядке): *PotentialUser*, *WantA*, *UsesA*. Добавьте элемент [*Начало диаграммы состояний*](http://127.0.0.1:64791/help/topic/com.xj.anylogic.help/html/statecharts/Statechart%20Entry%20Point.html), чтобы 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.



1. 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 A.



1. 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.

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

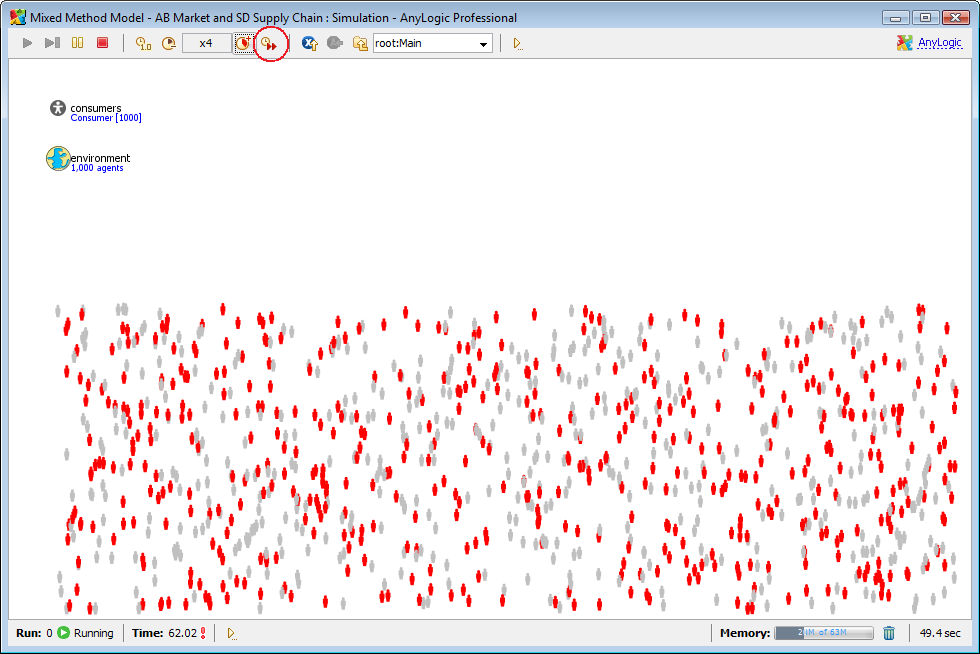
Поскольку мы подразумеваем, что продукт всегда доступен (есть в наличии), то все, кто хочет приобрести продукт, могут это сделать сразу же – поэтому, как только управление диаграммы состояний перейдет в состояние *WantA*, оно тут же, с нулевой задержкой, перейдет дальше, в состояние *UsesA*.

1. 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.

1. 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*.



1. 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.

1. Запустите модель еще раз.

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

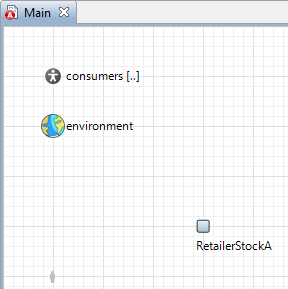
**Контрольная модель:** [AB Market and SD Supply Chain - Phase 2](javascript:liveAction(%22com.xj.anylogic.ui%22,%20%22com.xj.anylogic.actions.help.OpenSampleModelFromHelpAction%22,%22com.xj.anylogic.examples;basicmodels;AB%20Market%20and%20SD%20Supply%20Chain%20-%20Phase%202%22))

**Шаг 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.

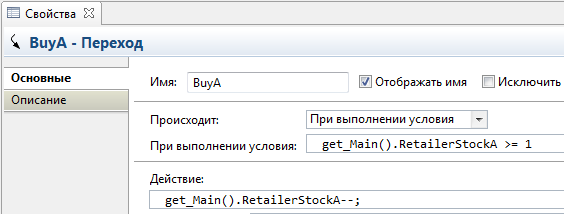
1. Откройте редактор класса *Main* и добавьте [накопитель](http://127.0.0.1:64791/help/topic/com.xj.anylogic.help/html/sd/Adding%20Stocks.html) *RetailerStock*. Задайте его начальное значение равным *100*.

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.



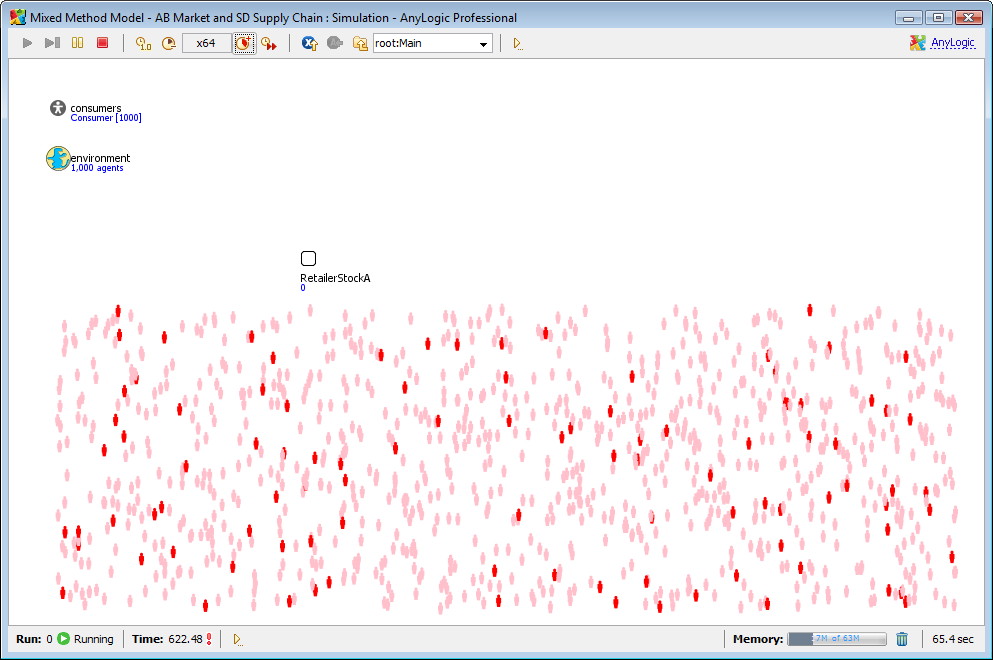
1. Перейдите в редакторе класса *Consumer* и измените свойства перехода *BuyA*: пусть теперь он **Происходит** *При выполнении условия*. Задайте **Условие**: get\_Main().RetailerStockA> = 1 и **Действие**: get\_Main().RetailerStockA--;

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*.



1. Запустите модель в режиме виртуального времени.

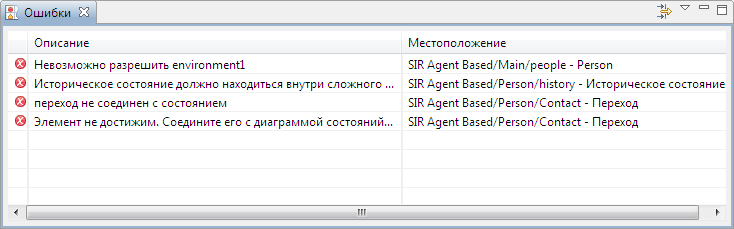
Вы увидите, что 100 потребителей смогут приобрести продукт A, в то время, как остальные постепенно будут перекрашены в розовый цвет, поскольку запасы ритейлера скоро подойдут к концу, и многие потребители не смогут удовлетворить свою потребность в продукте.



**Контрольная модель:** [AB Market and SD Supply Chain - Phase 3](javascript:liveAction(%22com.xj.anylogic.ui%22,%20%22com.xj.anylogic.actions.help.OpenSampleModelFromHelpAction%22,%22com.xj.anylogic.examples;basicmodels;AB%20Market%20and%20SD%20Supply%20Chain%20-%20Phase%203%22))

**Шаг 4. Запуск модели**

Постройте Вашу модель с помощью кнопки панели инструментов **Построить модель** http://127.0.0.1:64791/help/topic/com.xj.anylogic.help/html/SDT/images/Build_co.gif(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.

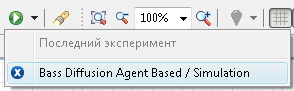


Панель Ошибки

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

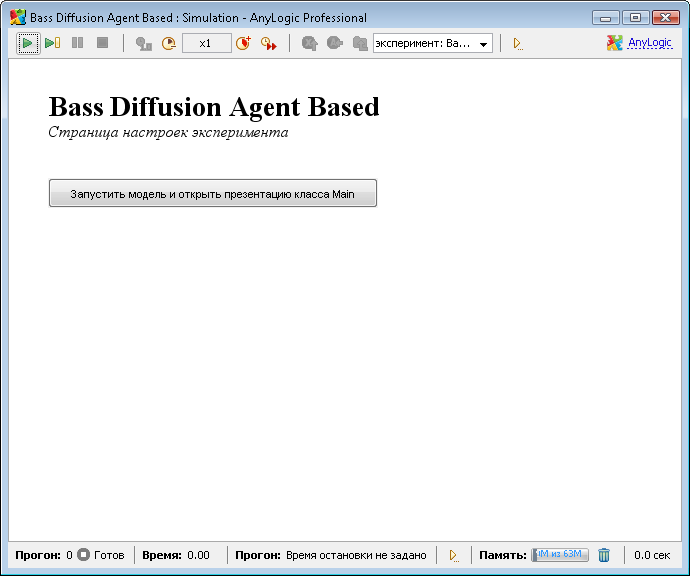
http://127.0.0.1:64791/help/topic/com.xj.anylogic.help/html/howto.gif**Запустите модель**

1. Щелкните мышью по кнопке панели инструментов **Запустить** http://127.0.0.1:64791/help/topic/com.xj.anylogic.help/html/running/Run%20the%20Simulation.files/Run_co.gifи выберите из открывшегося списка эксперимент, который Вы хотите запустить. Эксперимент этой модели будет называться *Bass Diffusion Agent Based/Simulation*.

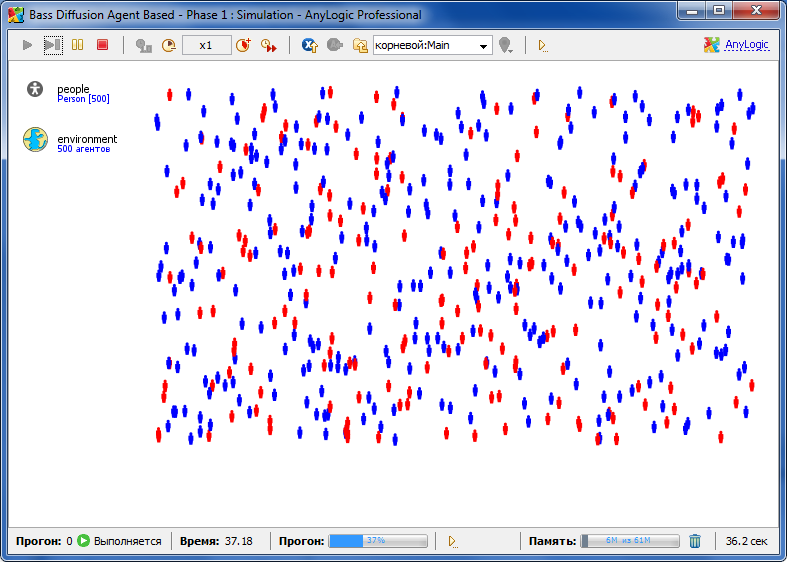


В дальнейшем по нажатию на кнопку **Запустить** http://127.0.0.1:64791/help/topic/com.xj.anylogic.help/html/running/Run%20the%20Simulation.files/Run_co.gif(или по нажатию F5) будет запускаться тот эксперимент, который запускался Вами в последний раз. Чтобы выбрать какой-то другой эксперимент, Вам будет нужно щелкнуть мышью по стрелке, находящейся в правой части кнопки **Запустить** http://127.0.0.1:64791/help/topic/com.xj.anylogic.help/html/running/Run%20the%20Simulation.files/Run_co.gifи 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 главного класса активного объекта этого 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**.

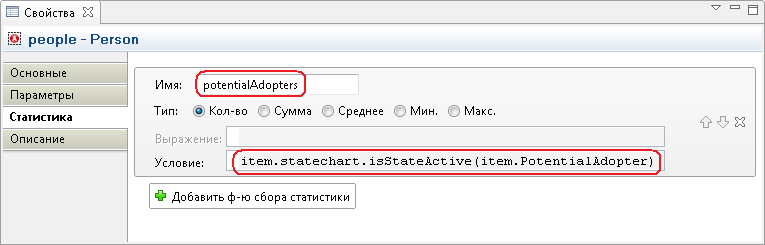
**Контрольная модель:** [Bass Diffusion Agent Based - Phase 1](javascript:liveAction(%22com.xj.anylogic.ui%22,%20%22com.xj.anylogic.actions.help.OpenSampleModelFromHelpAction%22,%22com.xj.anylogic.examples;basicmodels;Bass%20Diffusion%20Agent%20Based%20-%20Phase%201%22))

**Шаг 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.

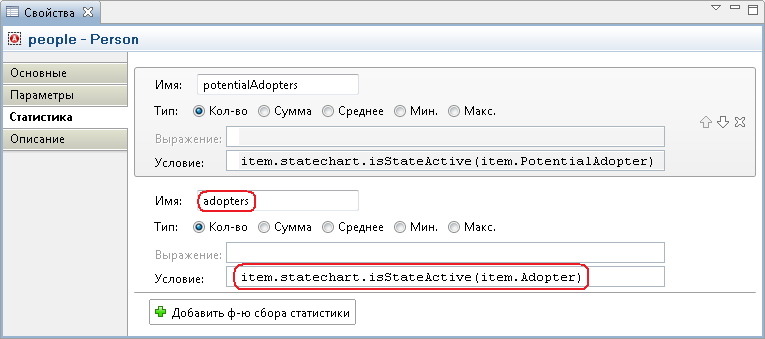
http://127.0.0.1:64791/help/topic/com.xj.anylogic.help/html/howto.gif**Создайте функции сбора статистики для подсчета потенциальных потребителей продукта**

1. Откройте диаграмму класса *Main*, сделав двойной щелчок мышью по элементу *Main* в панели **Проекты**.
2. Выделите на диаграмме вложенный объект *people*.
3. Перейдите на страницу **Статистика** панели **Свойства**.
4. 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).
5. Введите *potentialAadopters* в поле **Имя**. Это будет именем нашей функции.
6. Оставьте выбранный по умолчанию **Тип** функции: **Кол-во**.
7. Задайте **Условие**:  
   item.statechart.isStateActive(item.PotentialAdopter)  
   Эта функция будет вести подсчет количества агентов, для которых выполняется заданное условие, т.e. тех агентов, которые находятся в текущий момент времени в состоянии *PotentialAdopter* (являются потенциальными потребителями продукта).  
   Здесь item - это агент (элемент реплицированного объекта people).



http://127.0.0.1:64791/help/topic/com.xj.anylogic.help/html/howto.gif**Создайте функции сбора статистики для подсчета потребителей продукта**

1. Аналогично создайте еще одну функцию сбора статистики.
2. Назовите ее *adopters*.
3. Оставьте выбранный по умолчанию **Тип** функции: **Кол-во**.
4. Задайте **Условие**: item.statechart.isStateActive(item.Adopter)  
   Эта функция будет вести подсчет количества агентов, которые находятся в состоянии *Adopter* (то есть, уже приобрели продукт).

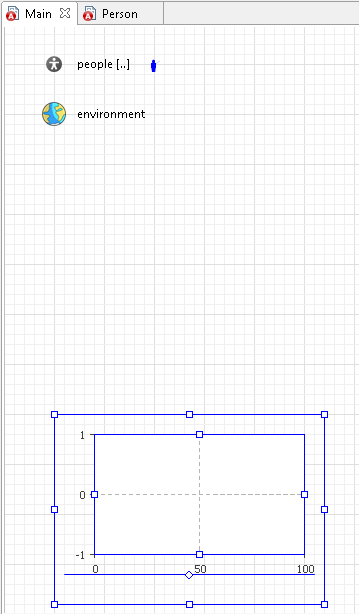


**Шаг 6. Добавление диаграммы**

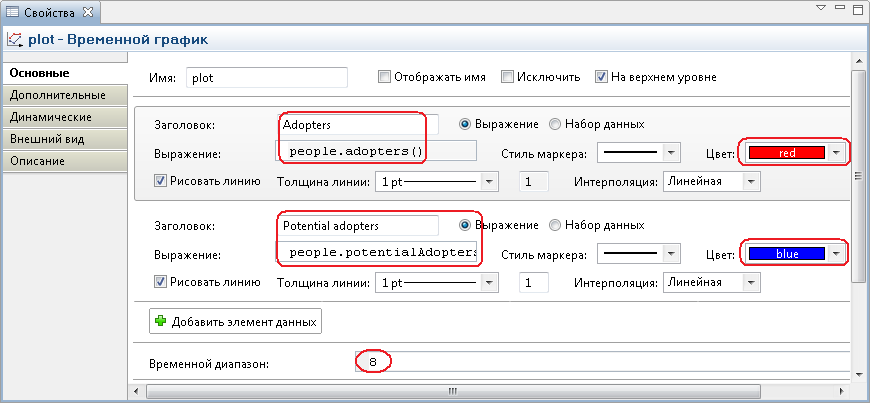
С помощью [*диаграмм*](http://127.0.0.1:64791/help/topic/com.xj.anylogic.help/html/analysis/Charts.html) Вы можете понаблюдать за динамикой моделируемого процесса. Сейчас мы создадим диаграмму, отображающую динамику изменения числа потребителей и потенциальных потребителей продукта.

http://127.0.0.1:64791/help/topic/com.xj.anylogic.help/html/howto.gif**Добавьте временной график, отображающий динамику изменения численностей потребителей и потенциальных потребителей продукта**

1. Откройте диаграмму класса *Main*, сделав двойной щелчок мышью по элементу *Main* в панели **Проекты**.
2. Перетащите элемент **Временной график**http://127.0.0.1:64791/help/topic/com.xj.anylogic.help/html/analysis/images/TimePlot_edit.gif из палитры **Статистика** на диаграмму класса.
3. Измените размер графика так, как показано на приведенном ниже рисунке:

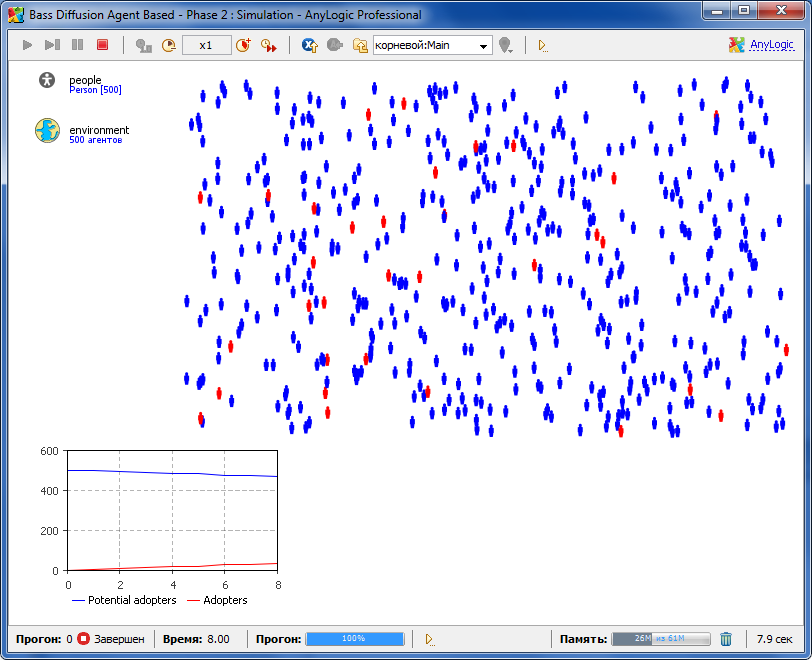


1. Перейдите на страницу **Основные** панели **Свойства**.
2. Укажите, что именно Вы хотите отображать на графике - то есть, задайте *элементы данных* этого графика.
3. Чтобы добавить элемент данных, щелкните мышью по кнопке **Добавить элемент данных** и в открывшейся секции свойств задайте свойства этого элемента.
4. 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.
5. Введите  *Potential adopters* в поле **Заголовок**. Эта строка будет отображаться в легенде диаграммы для данного элемента данных.
6. 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.
7. 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.



**Шаг 7. Учет влияния общения людей**

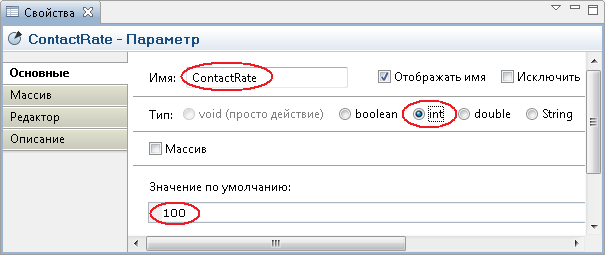
В текущей модели люди приобретают продукт только под влиянием рекламы. На самом деле, рекламный эффект играет значительную роль только в момент выпуска продукта на рынок. В дальнейшем все большую роль будет играть общение людей с теми своими знакомыми, которые этот продукт уже приобрели. В основном люди 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.

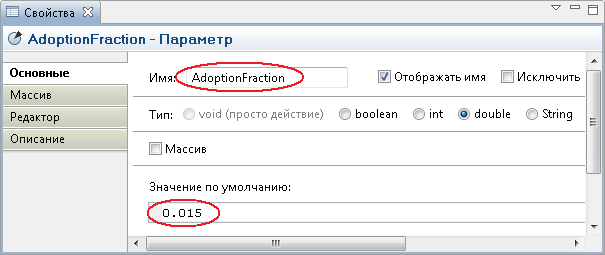
http://127.0.0.1:64791/help/topic/com.xj.anylogic.help/html/howto.gif**Задайте среднегодовое количество встреч человека**

1. Откройте диаграмму класса *Person*, сделав двойной щелчок мышью по элементу *Person* в панели **Проекты**.
2. Создайте новый параметр.
3. Назовите его *ContactRate*.
4. Предположим, что человек в среднем встречается со 100 людьми в год. Выберите **Тип** *int* и введите в поле **Значение по умолчанию** 100.



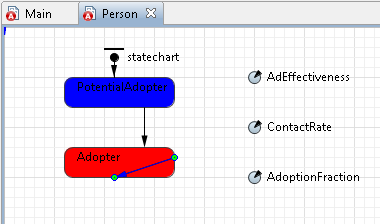
http://127.0.0.1:64791/help/topic/com.xj.anylogic.help/html/howto.gif**Задайте параметром убедительность человека**

1. Добавьте еще один параметр, задающий силу убеждения человека - долю общавшихся с владельцем продукта людей, которая приобретет этот продукт под влиянием общения..
2. Назовите этот параметр *AdoptionFraction*.
3. Задайте **Значение по умолчанию** 0.015.

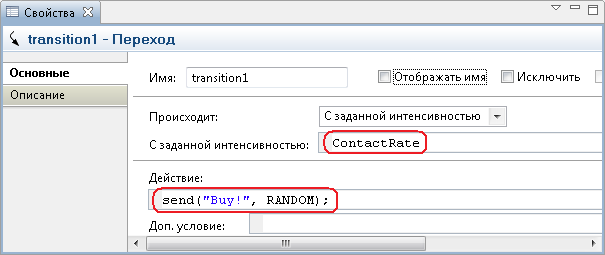


http://127.0.0.1:64791/help/topic/com.xj.anylogic.help/html/howto.gif**Измените диаграмму состояний агента**

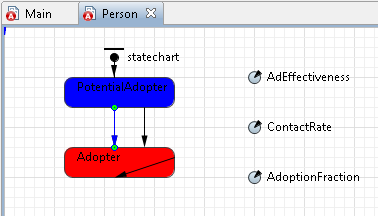
1. Откройте диаграмму класса *Person*.
2. Добавьте внутренний переход в состояние *Adopter*. В данном случае проще будет нарисовать его, перейдя в специальный режим рисования. Для того, чтобы перейти в этот режим, сделайте двойной щелчок мышью по элементу **Переход** http://127.0.0.1:64791/help/topic/com.xj.anylogic.help/html/statecharts/Transition.files/image002.gif в палитре **Диаграмма состояний** (при этом его значок должен поменяться на этот: http://127.0.0.1:64791/help/topic/com.xj.anylogic.help/html/presentation/images/ToolActive_co.gif). 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:



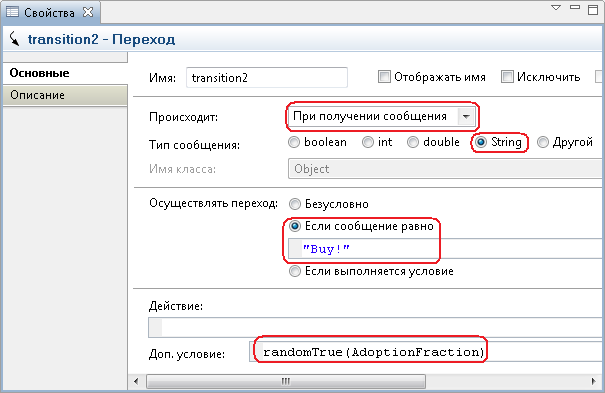
1. Это [внутренний переход](http://127.0.0.1:64791/help/topic/com.xj.anylogic.help/html/statecharts/Transition.html#inner) состояния *Adopter*. Этот переход будет моделировать общение человека со своим знакомым, в результате тот может быть убежден в покупке нового продукта. Интенсивность срабатывания 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
2. 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 качестве значения этого аргумента мы используем специальную константу RANDOM.   
   Подробное описание этого и других методов, используемых для организации взаиомодействия агентов смотрите в сатье документации [Взаимодействие агентов](http://127.0.0.1:64791/help/topic/com.xj.anylogic.help/html/agentbased/Communication.html).



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



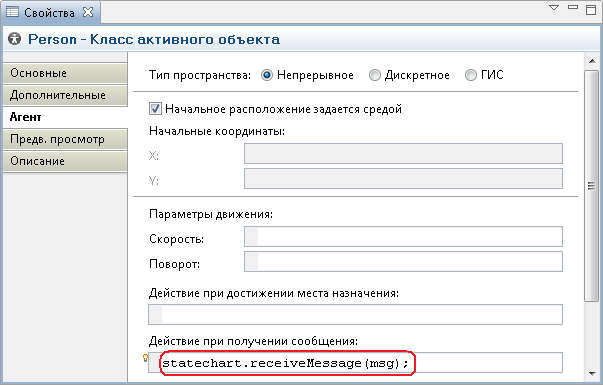
1. Измените свойства этого перехода. Не каждое обсуждение достоинств продукта со своим знакомым приведет к немедленному решению о приобретении этого продукта. Вероятность такого развития событий будет зависеть от того, насколько данный потенциальный потребитель подвержен внушению. В нашей модели данная 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.
2. 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*.
3. 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.



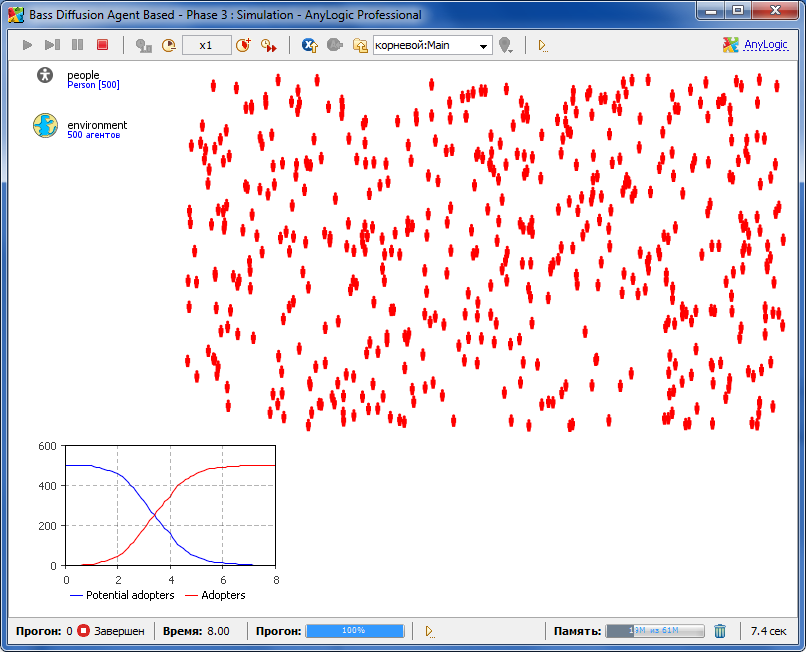
Теперь нам нужно изменить некоторые свойства агента, для того, чтобы получаемые им сообщения от других агентов перенаправлялись в его диаграмму состояний и обрабатывались ею в соответствии с заданной логикой.

http://127.0.0.1:64791/help/topic/com.xj.anylogic.help/html/howto.gif**Измените свойства агента**

1. Щелкните мышью по классу *Person* в панели **Проекты**, чтобы открыть его свойства в панели **Свойства**.
2. Перейдите на страницу свойств **Агент**.
3. 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 сотсояний, 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.

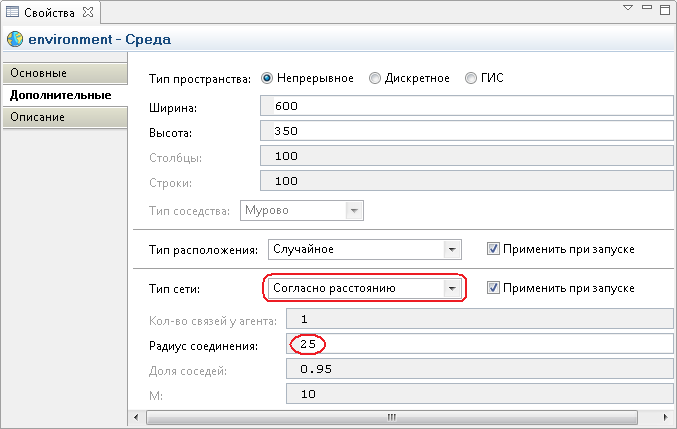
Сейчас люди в нашей модели случайно располагаются в прямоугольном пространстве 650x300 километров (или других условных единиц расстояния). И наша модель допускает общение любого человека с каждым, вне зависимости от того, на каком расстоянии друг от друга они находятся. Обычно же у человека есть определенный 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*.

http://127.0.0.1:64791/help/topic/com.xj.anylogic.help/html/howto.gif**Измените свойства среды**

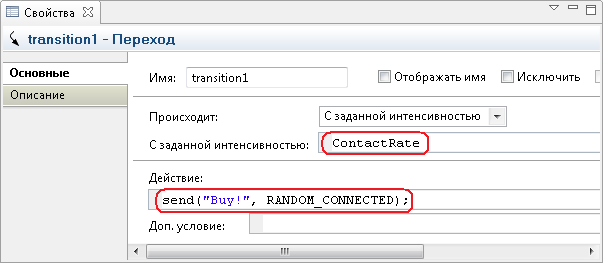
1. Откройте диаграмму класса *Main*.
2. Выделите на диаграмме объект environment http://127.0.0.1:64791/help/topic/com.xj.anylogic.help/html/agentbased/environment.png, задающий настройки среды, в которой обитают агенты.
3. Перейдите на страницу свойств **Дополнительные**.
4. Нам нужно изменить тип сети 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.

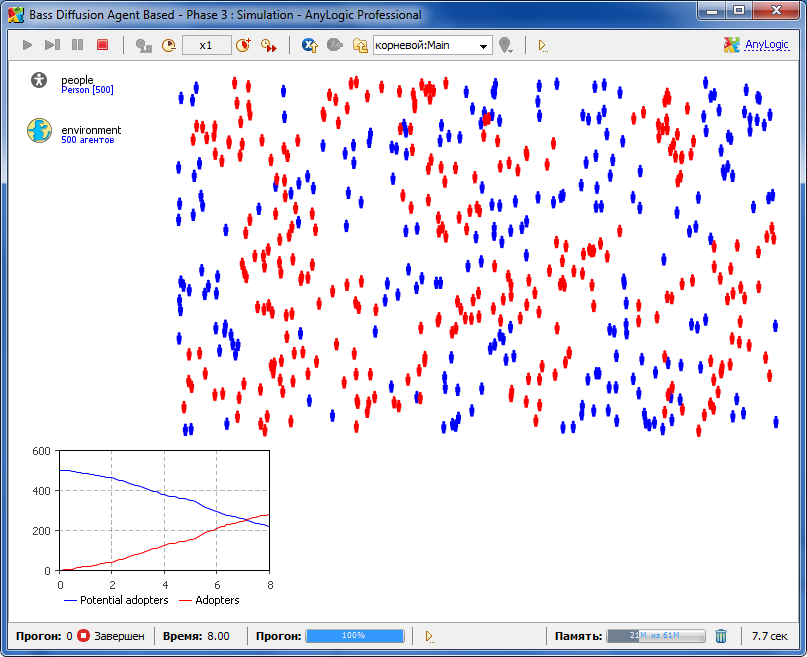
http://127.0.0.1:64791/help/topic/com.xj.anylogic.help/html/howto.gif**Измените диаграмму состояний агента**

1. Откройте диаграмму класса *Person*.
2. Измените свойства внутреннего перехода состояния *Adopter*.
3. Измените **Действие** этого перехода на:  
   send("Buy!", RANDOM\_CONNECTED);



1. 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.

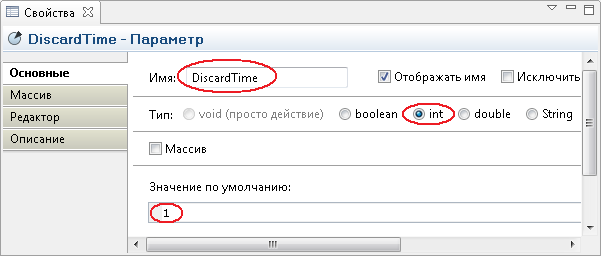
**Шаг 8. Моделирование повторных покупок**

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.

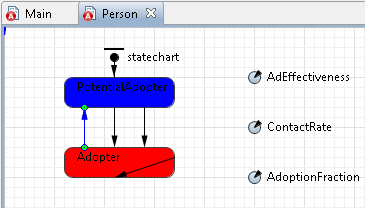
http://127.0.0.1:64791/help/topic/com.xj.anylogic.help/html/howto.gif**Задайте средний срок службы продукта**

1. Откройте диаграмму класса *Main*.
2. Создайте параметр *DiscardTime*. Пусть средний срок службы нашего продукта равен одному году.
3. Задайте **Значение по умолчанию**: 1.

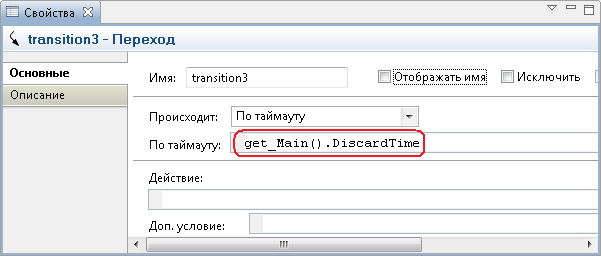


http://127.0.0.1:64791/help/topic/com.xj.anylogic.help/html/howto.gif**Измените диаграмму состояний**

1. Откройте диаграмму класса *Person*, сделав двойной щелчок мышью по элементу *Person* в панели **Проекты**.
2. Добавьте переход из состояния *Adopter* в состояние *PotentialAdopter*.



1. 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.

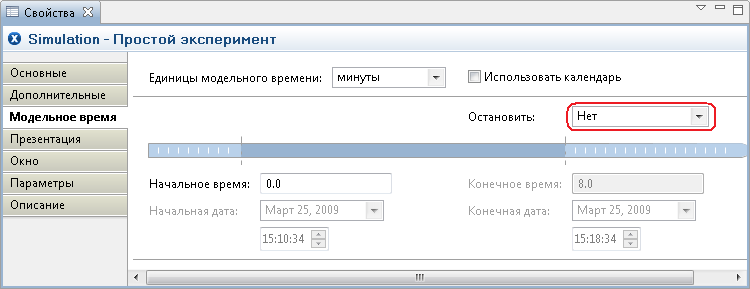


Мы закончили моделирование повторных покупок продукта.

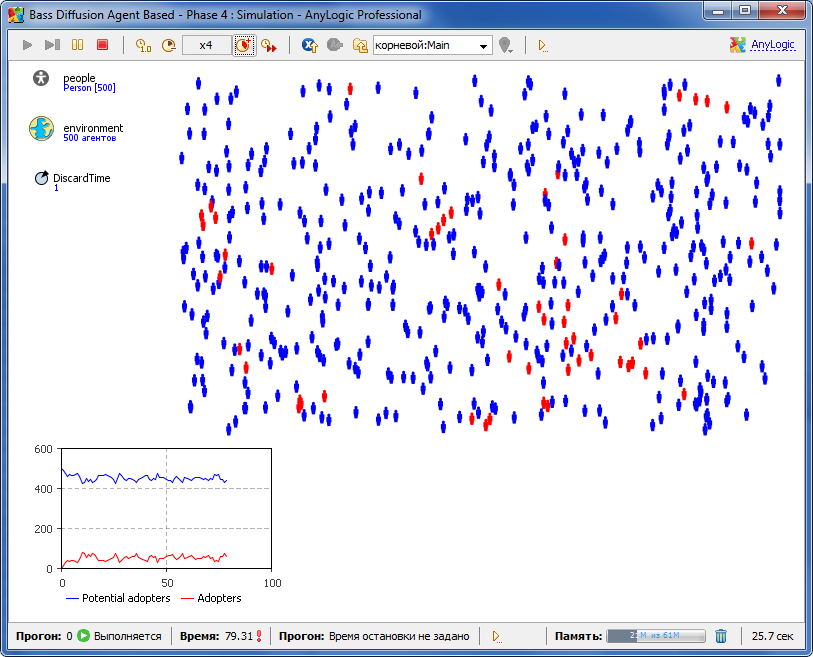
Теперь мы хотим исследовать процесс приобретения продукта в течение более длительного периода времению. Уберите заданное ранее условие остановки модели по прошествии определенного числа единиц модельного времени, чтобы модель выполнялась бесконечно, пока ее не остановит пользователь.

http://127.0.0.1:64791/help/topic/com.xj.anylogic.help/html/howto.gif**Удалите условие остановки модели по времени**

1. В панели **Проекты**, выделите эксперимент *Simulation:Main* щелчком мыши.
2. 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.



**Контрольная модель:** [Bass Diffusion Agent Based - Phase 4](javascript:liveAction(%22com.xj.anylogic.ui%22,%20%22com.xj.anylogic.actions.help.OpenSampleModelFromHelpAction%22,%22com.xj.anylogic.examples;basicmodels;Bass%20Diffusion%20Agent%20Based%20-%20Phase%204%22))