



**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА
(САМАРСКИЙ УНИВЕРСИТЕТ)»**

**ИНСТИТУТ ИНФОРМАТИКИ, МАТЕМАТИКИ И ЭЛЕКТРОНИКИ
Факультет информатики
Кафедра программных систем**

А.В. Баландин

Основы компьютерного моделирования в AnyLogic

Учебное пособие

**Самара
2024**









УДК 681.3.066
ББК 32.973.26-018.2

Баландин А.В. Основы компьютерного моделирования в AnyLogic. Учебное пособие - Самар. ун-т. Самара, 2024. 48 с.

В пособии даны основы графо-символической разработки компьютерных моделей в системе AnyLogic в виде виртуального активного объекта иерархической структуры – агента, поведение которого управляется модельным временем, и моделируемыми событиями. Описан интерфейс пользователя и средства графического редактора системы AnyLogic для построения диаграмм агентных классов, специфицированных системами алгебраических и/или дифференциальных уравнений, а также диаграммами состояний реагирующих систем. Описаны этапы подготовки и проведения экспериментов с компьютерной моделью для исследования характеристик предмета моделирования.

Учебное пособие предназначено обучающимся направления 02.03.02 – Фундаментальная информатика и информационные технологии для самостоятельного изучения и освоения средств моделирования системы AnyLogic, необходимых для выполнения заданий лабораторных работ и курсовой работы по дисциплине «Моделирование информационных процессов и систем».

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ	5
КОНЦЕПЦИЯ АГЕНТНОГО МОДЕЛИРОВАНИЯ	5
Понятие агента	5
Эксперименты с моделью	6
НАЧАЛО РАБОТЫ.....	7
Старт ANYLOGIC	7
Справочная система ANYLOGIC	8
Создание модели	9
Открытие модели	11
Сохранение моделей.....	11
Закрытие моделей.....	11
Переименование элементов модели	12
Управление элементами модели.....	12
Комментирование элементов модели	12
ПОСТРОЕНИЕ И ИСПОЛНЕНИЕ МОДЕЛЕЙ.....	14
Агентные типы и агенты	14
Создание нового агента типа	14
Иерархический агентный тип.....	14
<i>Агенты верхнего уровня.....</i>	<i>15</i>
Эксперименты с моделью	16
<i>Типы экспериментов.....</i>	<i>17</i>
Простой эксперимент	17
Оптимизация	17
Варьирование параметров.....	17
<i>Модельное время</i>	<i>17</i>
Единицы модельного времени.....	18
Выбор режима времени и скорости выполнения модели	18
Задание начального и конечного времени моделирования	19
<i>Запуск модели</i>	<i>20</i>
<i>Простой эксперимент.....</i>	<i>21</i>
<i>Оптимизационный эксперимент.....</i>	<i>21</i>
<i>Эксперимент варьирования параметров.....</i>	<i>23</i>
ГРАФИЧЕСКОЕ МОДЕЛИРОВАНИЕ ДИНАМИЧЕСКИХ СИСТЕМ	24
Графические элементы построения D-схем.....	24
 <i>-Параметр.....</i>	<i>25</i>
Статический параметр	26
Динамический параметр.....	27
Параметр-действие	29
 <i>-Переменная.....</i>	<i>29</i>
 <i>-Динамическая переменная.....</i>	<i>31</i>
Программное управление динамической переменной	31
Функциональное использование динамической переменной	32
Параметрическое использование динамической переменной	32
 <i>Накопитель.....</i>	<i>33</i>
 <i>Поток</i>	<i>34</i>
 <i>Связь</i>	<i>35</i>
Полярность связи	36
Создание связи	37
Механизм быстрого добавления отсутствующих связей	38
 <i>Функция.....</i>	<i>39</i>
<i>Определение функции.....</i>	<i>39</i>
Аргументы функции.....	40
Уровень доступа к функции.....	41
 <i>Табличная функция.....</i>	<i>41</i>
<i>Диаграммы действий.....</i>	<i>41</i>
УПРАВЛЕНИЕ ВЫПОЛНЕНИЕМ МОДЕЛИ.....	43
<i>Событие.....</i>	<i>43</i>

	4
<i>Диаграмма состояний</i>	44
Элементы диаграммы состояний	44
Порядок выполнения действий в диаграмме состояний	45
ПРИЛОЖЕНИЕ 1	47
МАТЕМАТИЧЕСКИЕ ФУНКЦИИ	47
ПРИЛОЖЕНИЕ 2	48
МАТЕМАТИЧЕСКИЕ КОНСТАНТЫ	48
СИСТЕМНЫЕ ФУНКЦИИ	48

Введение

Компьютерное моделирование является единственным и в то же время сложным, и зачастую трудоёмким способом априорной оценки характеристик разрабатываемой информационной системы. Своевременное получение результатов моделирования для оценки принимаемых проектных решений в значительной степени зависит от возможностей и качества используемых инструментальных средств моделирования. Одним из таких средств является система компьютерного моделирования AnyLogic, разработанная в 1991 году российской компанией XJ Technologies (www.xjtek.ru). С момента своего создания система AnyLogic постоянно совершенствуется, став универсальным инструментом, предназначенным для моделирования и исследования характеристик различных систем, включая и информационные системы¹.

Технология моделирования в AnyLogic основана на представлении предмета моделирования как иерархически связанного множества *активных виртуальных объектов - агентов*, параллельно функционирующих во времени и взаимодействующих между собой посредством разделения общих данных или передачи данных. Иерархическая связь агентов выражается в том, что построенные агенты, моделирующие составную часть сложной системы – агенты нижних уровней, в инкапсулированном виде используются для конструирования агентов верхних уровней. Благодаря графическому пользовательскому интерфейсу AnyLogic для задания структуры, свойств и поведения агентов, а также выразительных анимационных возможностей презентации результатов проведения экспериментов с агентами обеспечивается высокая производительность создания сложных моделей и эффективность их анализа.

При разработке модели в большинстве случаев нельзя обойтись без написания программных фрагментов на языке Java. Важно, однако, что язык Java настолько интегрирован в пользовательский интерфейс AnyLogic, что программные фрагменты "естественно" встраиваются в структуру создаваемого агента. При разработке моделей с простыми типами данных и процедурами поведения доля программирования обычно не велика, а реализованный в AnyLogic способ включения в модель программных фрагментов в основном не требует от пользователя знаний специфических особенностей языка Java. Тем не менее, при создании сложных моделей могут понадобиться знания основ объектно-ориентированного программирования и отладки программ на языке Java. Разработанная в AnyLogic графическая модель в итоге преобразуется в генерируемый Java-апплет, который запускается на выполнение при проведении вычислительных экспериментов с моделью. Но можно создавать и автономные JAR-файлы.

Концепция агентного моделирования

Понятие агента

Логически моделирование в AnyLogic представляется как построение аналога предмета моделирования в виде многоуровневого множества иерархически связанных, параллельно функционирующих во времени и взаимодействующих путём обмена данными динамических информационных объектов, называемых *агентами*. Построение модели начинается с создания исходных агентов, используя диаграммы различных типов, составляющие графический язык моделирования AnyLogic. Построенные исходные агенты можно инкапсулировать и тиражировать, порождая тем самым агенты одного класса, которые могут использоваться как конструктивные элементы при построении агентов других уровней иерархии. В результате одни агенты многоуровневого множества инкапсулируют другие агенты этого же множества более низких уровней. Построенное в итоге многоуровневое множество агентов и является аналогом предмета моделирования - *моделью*.

¹ Компания AnyLogic предоставляет для обучения студентов и самообразования бесплатную версию AnyLogic (Personal Learning Edition - PLE), которую можно установить на свой компьютер, скачав дистрибутив с официального сайта компании AnyLogic (<http://www.anylogic.ru>).

Многоуровневое моделирование позволяет рассматривать предмет моделирования на различных уровнях детализации. В качестве примера многоуровневой модели можно назвать модель организации хранения данных в файловой системе компьютера. Аналогами агентов на разных уровнях иерархии в данном случае выступают различного типа носители информации: диски, папки, файлы. В результате пользователь может рассматривать файловую систему на разных уровнях детализации - от устройства до файла. Однако объекты файловой системы являются статическими агентами, они не обладают способностью самостоятельно изменять свои свойства во времени. Принципиальное отличие агента от объекта файловой системы в том, что объект-агент обладает поведением (изменяет свойства) во времени и в этом смысле каждый агент на заданном промежутке времени являет собой самостоятельный процесс, параллельно развивающийся со всеми остальными процессами-агентами. Наблюдение за поведением модели во времени интерпретируется как эксперимент с моделью. С этой точки зрения модель следует рассматривать как иерархическое множество развивающихся во времени взаимодействующих параллельных процессов.

При моделировании пользователь, в общем случае, наделяет каждый агент набором свойств, процедур поведения и способов взаимодействия с другими агентами. В соответствии с концепцией агентно-ориентированного моделирования создание агента неявно порождает определение *класса агентов* (абстрактный *агентный тип*), после чего тиражирование агента рассматривается как создание объектов соответствующего агентного типа.

В иерархической структуре модели различают *простые* и *составные* агенты. Простые агенты составляют нижний уровень иерархической структуры модели. Они не содержат в своей структуре объекты агентных типов. Составные агенты включают в себя объекты агентных типов и в этом смысле являются *макроагентами*. В итоге модель в целом является макроагентом с многоуровневой структурой.

Согласно концепции AnyLogic, каждый созданный тип агента по умолчанию наделяет каждый объект этого агентного типа стандартным графическим образом – *простейшей презентацией*, и *контактами* для связи с другими агентами. AnyLogic не ограничивается этим, он предоставляет средства, позволяющие заменить образ по умолчанию специально разработанной пользователем презентацией. Поэтому в модели агенты любого уровня имеют либо визуальное представление по умолчанию, либо явно созданную пользователем презентацию.

Эксперименты с моделью

Целью построения модели является исследование поведения эндогенных параметров предмета моделирования во времени. Для этого AnyLogic предоставляет средства экспериментирования с моделью. Как объект экспериментирования каждая модель характеризуется набором настраиваемых экзогенных управляемых параметров, которые используются для установки начального состояния модели. В рамках объектно-ориентированной технологии моделирования AnyLogic спецификация эксперимента с моделью рассматривается как создание особого логического объекта – объекта класса «эксперимент» с моделью. Объект-эксперимент (далее просто *эксперимент*) характеризует начальное состояние модели, модельное время, в соответствии с которым будут изменяться динамические экзогенные параметры модели и другие характеристики. Каждый созданный (специфицированный) эксперимент наделяется именем и сохраняется в системе.

Выполнение в системе запущенного эксперимента заключается в изменении динамических экзогенных параметров модели в модельном времени. На презентации модели пользователь может наблюдать изменение всех параметров модели. С одной и той же моделью AnyLogic позволяет создать и сохранить эксперименты с различными именами и тем самым получать и сохранять различные версии настройки параметров одной и той же модели как параметры эксперимента с моделью. При запуске эксперимента все агенты, составляющие модель, выполняются как взаимодействующие параллельные процессы. При этом реализация эксперимента выглядит как «прогон» модели с соответствующей настройкой параметров и выбранной моделью времени. В процессе прогона на презентации модели автоматически визуализируются изменения во времени

значений эндогенных параметров. Каждый эксперимент по умолчанию сохраняется в проекте модели как именованный элемент проекта типа "*простой эксперимент*". По умолчанию при создании проекта модели в него сразу включается простой эксперимент с именем Simulation с заданными при создании агентов начальными значениями управляемых экзогенных параметров модели.

Начало работы

Старт AnyLogic

При запуске AnyLogic на экране появляется начальная страница, которая содержит краткое описание основных возможностей программы (Рисунок 1).

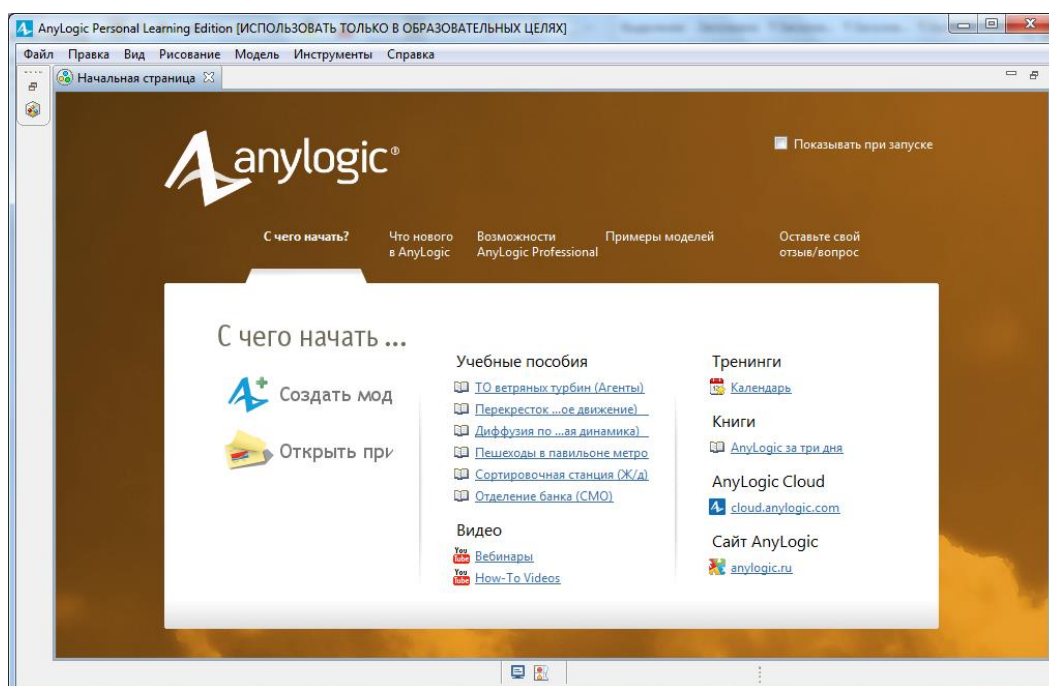


Рисунок 1 Начальная страница

По умолчанию начальная страница отображается при каждом запуске AnyLogic. Однако, можно отключить её отображение через настройки AnyLogic или, просто сбросив флаг "**Показывать при запуске**" в окне начальной страницы.

Для перехода в рабочий режим закройте панель (вкладку) начальной страницы, после чего откроется окно рабочей области AnyLogic (Рисунок 2).

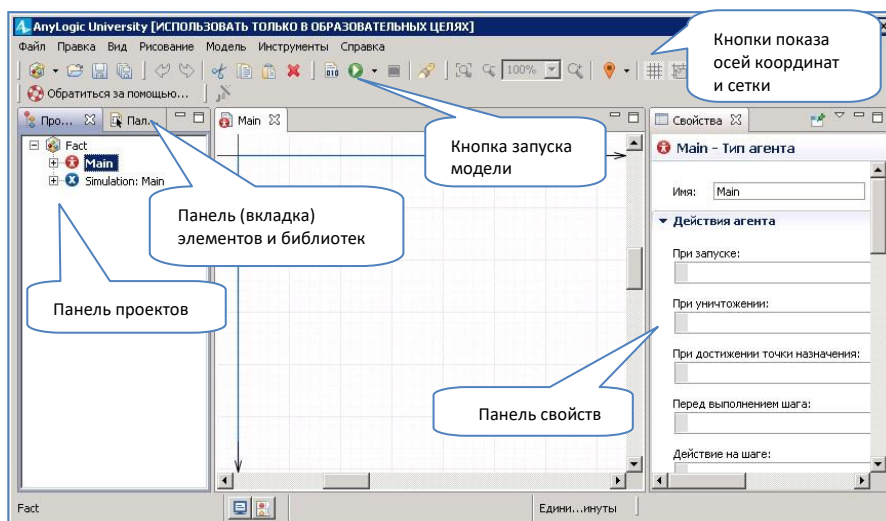


Рисунок 2 Рабочая область AnyLogic

Рабочая область включает в себя набор панелей (вкладок) и окно графического редактора (обычно располагающееся в центре рабочей области), составляющих пользовательский интерфейс для управления созданием модели и проведения экспериментов. Пользователь может управлять их расположением в рабочей области по своему усмотрению. Каждая из панелей может при необходимости включаться или отключаться с помощью меню Панель. Наиболее актуальными для пользователя являются следующие панели:

- **Панель Проект.** Отображает иерархическую агентно-ориентированную структуру модели и обеспечивает наглядную навигацию по агентам модели.
- **Панель Палитра.** Включает в себя набор примитивов графо-символического языка конструирования моделей, разбитых на семантические группы, отображаемые на разных вкладках панели (**Основная**, **Презентация**, **Элементы управления** и др.).
- **Панель Свойства.** Отображает свойства выделенного элемента модели.
- **Панель Ошибки.** Отображает ошибки, возникшие при построении модели, и помогает их локализовать.

Справочная система AnyLogic

AnyLogic предоставляет пользователю мощную справочную систему, которую необходимо использовать как пособие для самостоятельного освоения системы. Для работы со справочной документацией необходимо выполнить команду меню **Справка/Справка AnyLogic**, откроется окно **Справка** (Рисунок 3).

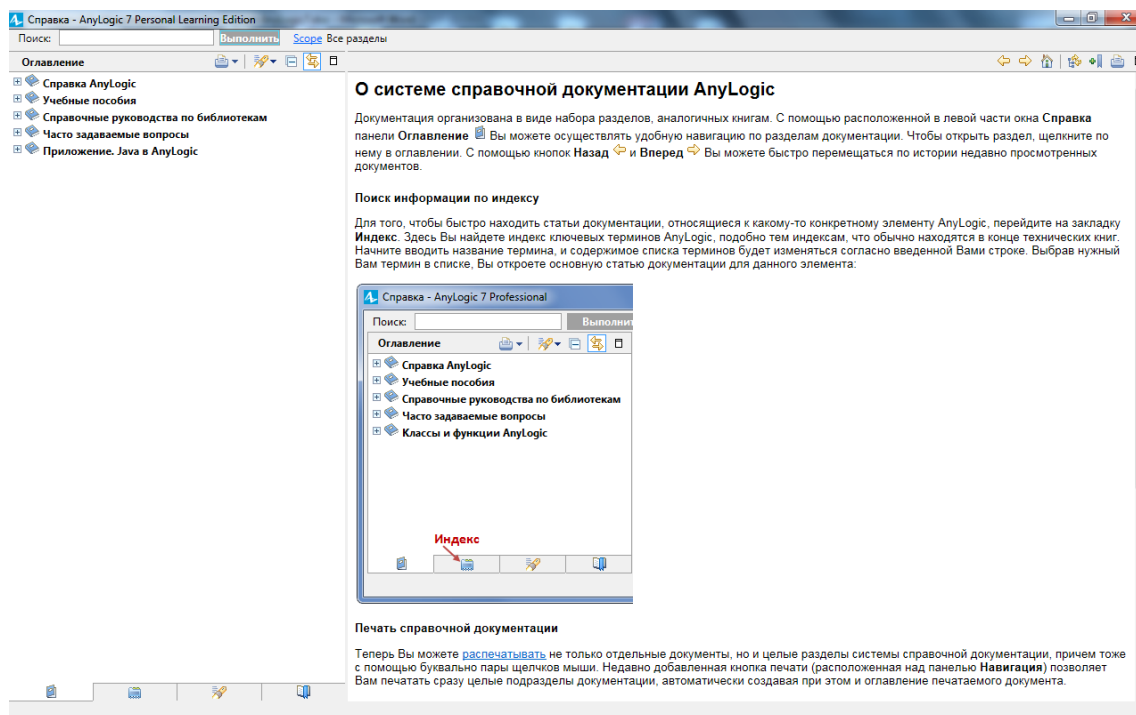


Рисунок 3 Окно справочной системы

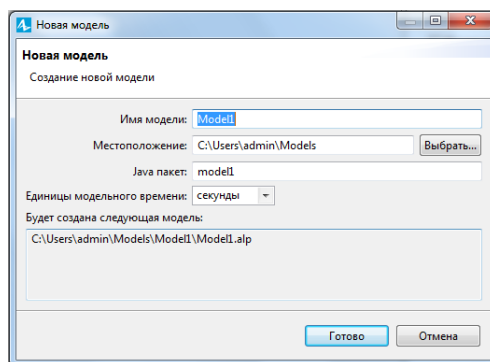
Справочная система AnyLogic поддерживает функции навигации, текстового поиска по ключевому слову или строке текста, создания личных закладок пользователя и печати документации (Рисунок 3). Также поддерживается механизм контекстно-зависимой справки, обеспечивающий быстрый доступ к краткой информации о конкретном элементе, с которым пользователь работает в текущий момент времени.

Создание модели

Запустите AnyLogic. Процесс моделирования в AnyLogic начинается с создания нового проекта модели с единой для всех проектов структурой. Проект модели (или просто модель) создаётся и сохраняется в AnyLogic целиком в одном файле с именем, совпадающим с заданным при создании проекта модели именем и расширением **.alp**. По умолчанию создаваемая модель получает имя **Model** и сохраняется в файле с именем **Model.alp** в папке **Model**, которая создаётся автоматически в указанном пользователем месте файловой системы компьютера. Например, если для размещения модели в ОС Windows указывается **Рабочий стол**, то полное имя файла с моделью будет **C:\Users\admin\Desktop\Model\Model.alp**.

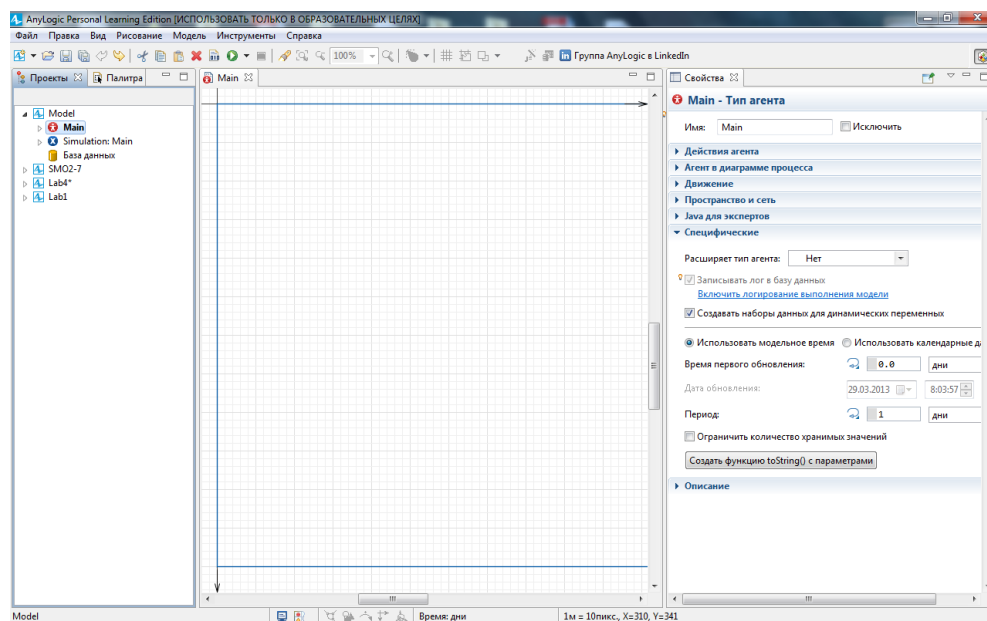
Чтобы создать новую модель:

1. Выполните пункты меню **Файл|Создать|Модель** из главного меню. Появится диалоговое окно **Новая модель**:

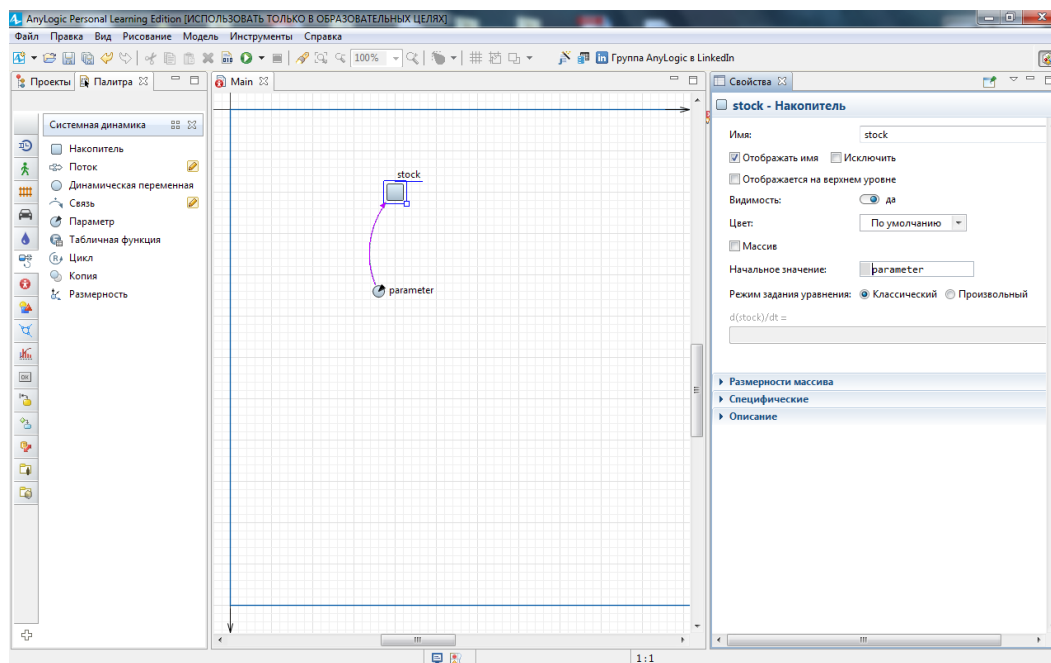


2. Введите имя новой модели и задайте местоположение каталога, в котором будут храниться файлы модели в поле **Местоположение**.
3. Остальные поля оставьте без изменения.
4. Щелкните мышью по кнопке **Готово**.

Имя созданной модели появляется на панели **Проект**, и является самым верхним элементом проекта (корнем дерева модели - **root**). Дерево "пустой" модели уже включает в себя один пустой агент (основа для построения модели) с именем **Main**. Для управления исполнением созданной модели по умолчанию создаётся один (простой) эксперимент с именем **Simulation**. В рабочей области окна AnyLogic откроется окно графического редактора с пустой диаграммой агента **Main**:



Переключитесь на панель **Палитра**. Конструирование диаграммы агента **Main** осуществляется с использованием пиктограмм конструктивных элементов языка AnyLogic в окне графического редактора агента **Main**. Пиктограммы выбираются в палитре и переносятся в окно графического редактора с мышью:



Открытие модели

Чтобы открыть ранее созданную модель:

1. Выберите пункты меню **Файл|Открыть...** из главного меню. Откроется диалоговое окно **Открыть**.
2. Выберите нужный файл модели и щелкните по кнопке **Открыть**.

Можно открыть одновременно несколько моделей. Когда открывается модель, она добавляется в текущий набор открытых моделей панели **Проекты**. При запуске AnyLogic автоматически открывает все модели, присутствовавшие в этом наборе на момент последнего закрытия AnyLogic.

Сохранение моделей

Чтобы сохранить текущую модель:

1. Выберите пункты **Файл|Сохранить** из главного меню.

Чтобы сохранить текущую модель под другим именем:

1. Выберите пункты **Файл|Сохранить как...** из главного меню. Откроется диалоговое окно **Сохранить как**.
2. Введите новое имя модели и задайте каталог, в котором будут храниться файлы этой модели.
3. Щелкните мышью по кнопке **Сохранить**.

Чтобы сохранить все открытые модели

1. Выберите пункты **Файл|Сохранить все** из главного меню.

Закрытие моделей

Чтобы окончательно завершить работу с моделью, можете закрыть её, убрав из текущего набора открытых моделей.

Чтобы закрыть модель

1. Выберите любой элемент модели, которую хотите закрыть, и выберите пункты **Файл|Закрыть** из главного меню.

Переименование элементов модели

Помимо простого переименования элемента, когда Вы просто изменяете его имя в панели **Свойства**, AnyLogic поддерживает механизм переименования элемента, производящий также соответствующее переименование всех ссылок на этот элемент в данной модели. Такое переименование не влияет на поведение модели, оставляя ее семантику прежней. Это может пригодиться, например, в следующем случае: пусть Вам нужно переименовать переменную модели, задействованную в формулах, определяющих значения других переменных модели. Если Вы просто переименуете эту переменную, то соответствующие формулы станут некорректными, а если Вы воспользуетесь механизмом переименования с заменой всех ссылок на элемент, то имя этой переменной будет изменено и во всех соответствующих формулах, и семантика модели останется прежней.

Чтобы переименовать элемент модели (а также все ссылки на его имя)

1. Щелкните правой кнопкой мыши по модели в панели **Проект**, выберите **Переименовать** из контекстного меню и в открывшемся диалоговом окне введите новое имя элемента в поле **Новое имя**. Если Вы хотите предварительно просмотреть результаты планируемого переименования, оставьте выбранным флажок **Предварительный просмотр изменений**. Щелкните по кнопке **Переименовать**.
2. Появится диалоговое окно **Предварительный просмотр изменений**. Оставьте выделенными в списке **Будут сделаны следующие изменения** только те ссылки на элемент, которые Вы также хотите переименовать, и щелкните по кнопке **ОК**.

Управление элементами модели

Можно управлять набором элементов модели, выбирая для каждого из них, хотите ли Вы отображать его имя на диаграмме в окне графического редактора, а также должен ли этот элемент отображаться на презентации во время выполнения модели.

Чтобы спрятать/отобразить имя элемента в графическом редакторе:

1. Выделите этот элемент в графическом редакторе или в панели **Проект**.
2. На вкладке **Основные** панели **Свойства** сбросьте/установите флажок **Отображать имя**.

Чтобы спрятать/отобразить элемент на презентации агента (или эксперимента) во время работы модели:

1. Выделите этот элемент в графическом редакторе или в панели **Проект**.
2. На вкладке **Основные** панели **Свойства** сбросьте/установите флажок **На презентации**.

Можно временно исключить любой элемент из модели. Эта возможность полезна для отладки моделей, поскольку позволяет легко изменять структуру модели, исключая одни элементы модели и добавляя другие. Исключенные элементы, тем не менее, продолжают отображаться в окне графического редактора, и при необходимости можно в любой момент включить ранее исключенные элементы обратно в модель.

Чтобы исключить/включить элемент в модели:

1. Выделите этот элемент в окне графического редактора или в панели **Проект**.
2. На вкладке **Основные** панели **Свойства** установите/сбросьте флажок **Исключить**.

Комментирование элементов модели

Элементы модели можно сопровождать комментариями, отражающими семантику элемента, это делает как элемент, так и модель в целом более понятной.

Чтобы задать описание элемента:

1. Выберите элемент модели.
2. На вкладке **Описание** панели **Свойства** введите текст комментария.

Построение и исполнение моделей

Агентные типы и агенты

Агенты используются в AnyLogic в качестве аналогов любых реальных или виртуальных динамических сущностей предмета моделирования (люди, машины, здания, аппаратные и программные системы, вычислительные сети, процессы, явления и т.д.), характеризующихся помимо свойств и поведением во времени. Для спецификации агентов в модели создаются агентные типы (классы агентов). Агенты как структурные элементы модели являются динамическими объектами соответствующих агентных типов. Язык моделирования AnyLogic располагает широким набором встроенных агентных типов (представленных на панели **Палитра**), которые, как правило, используются для конструирования пользователем с помощью графического редактора AnyLogic собственных агентных типов (в виде различных диаграмм), включаемых в проект модели. Каждому типу агентов сопоставляется некий графо-символический примитив (значок, пиктограмма), который представляет агентный тип в дереве проекта, а соответствующих агентов этого типа - в окне графического редактора.

Вновь созданный агентный тип является «пустым» - ему соответствует пустое окно графического редактора и не сформированы свойства и процедуры поведения (действия) агентов данного типа (отображаются на панели **Свойства**). При формировании диаграммы агентового типа используются графо-символические примитивы ранее созданных агентных типов, находящихся на панели **Палитра**, или непосредственно в дереве проекта модели. В процессе формирования диаграммы агентового типа окно графического редактора заполняется пиктограммами агентов и связями между ними. Диаграмма агентного типа может формироваться постепенно и включать в себя всё новые графические примитивы, характеризующие свойства и процедуры поведения моделируемых сущностей. По ходу построения агентного типа с ним по умолчанию ассоциируется экземпляр агента этого типа, с которым можно проводить эксперименты.

Создание нового агентового типа

Типы агентов создаются в модели пользователем или могут быть взяты из тематических библиотек агентных типов AnyLogic на панели **Палитра**. Проект вновь созданной модели по умолчанию содержит пустой агентный тип **Main** и связанный с ним простой эксперимент **Simulation**. В некоторых случаях может быть достаточным для моделирования использовать лишь один агентный тип **Main**, который можно при необходимости переименовать. Если одного агентового типа недостаточно, то можно создать необходимое количество других агентных типов.

Чтобы создать новый тип агентов:

1. В панели **Проект**, щелкните правой кнопкой мыши по модели, с которой Вы работаете в данный момент, и выберите **Создать|Тип агента** из контекстного меню.
2. Откроется диалоговое окно **Создание агентов**.
3. Задайте имя нового типа. Начинайте имя типа с заглавной буквы по правилу названия классов в Java.
4. Щелкните по кнопке **Готово**. Новый тип появится в проекте модели.

Иерархический агентный тип

AnyLogic позволяет создавать иерархический агентный тип. Это следует понимать так, что при создании диаграммы очередного агентового типа в неё можно включать/добавлять в виде пиктограмм экземпляры агентов других уже имеющихся в проекте агентных типов (вложенные агенты). Количество уровней иерархической вложенности агентов неограниченно. Использование в диаграмме вложенных агентов позволяет "прятать" технические детали реализации диаграммы агентного типа на различных уровнях рассмотрения.

Чтобы добавить вложенный агент в диаграмму создаваемого агентного типа:

1. Пиктограмму нужного агентного типа перетащите мышью из панели **Проект** (или из раздела панели **Палитра**) в окно графического редактора с диаграммой агентного типа и добавьте её в диаграмму.
2. На диаграмме появится новый значок, соответствующий пиктограмме агентного типа - значок вложенного агента.

Если нужно выяснить, агентом какого типа является вложенный агент, можно выделить его значок в диаграмме и прочитать имя его типа в панели **Свойства** (в поле **Тип**).

Если Вы добавили вложенный агент и хотите уточнить или отредактировать какие-то детали его внутренней структуры (диаграммы типа), то можно прямо на текущей диаграмме с помощью мыши выделить вложенный агент и двойным щелчком открыть окно графического редактора с диаграммой типа этого вложенного агента.

Агенты верхнего уровня

В проекте модели на самом верхнем уровне располагаются все созданные пользователем агентные типы. При этом по умолчанию агент любого агентного типа в AnyLogic называют – *агент верхнего уровня*. Логически агент верхнего уровня можно обособленно рассматривать как подмодель. Это означает, что диаграмму его агентного типа можно редактировать отдельно и обособленно проводить эксперименты с выбранным агентом верхнего уровня. Это даёт возможность при необходимости любую подмодель рассматривать и исследовать самостоятельно. Для этого следует в свойствах выделенного эксперимента выбрать один из *агентов верхнего уровня*. (Рисунок 4).

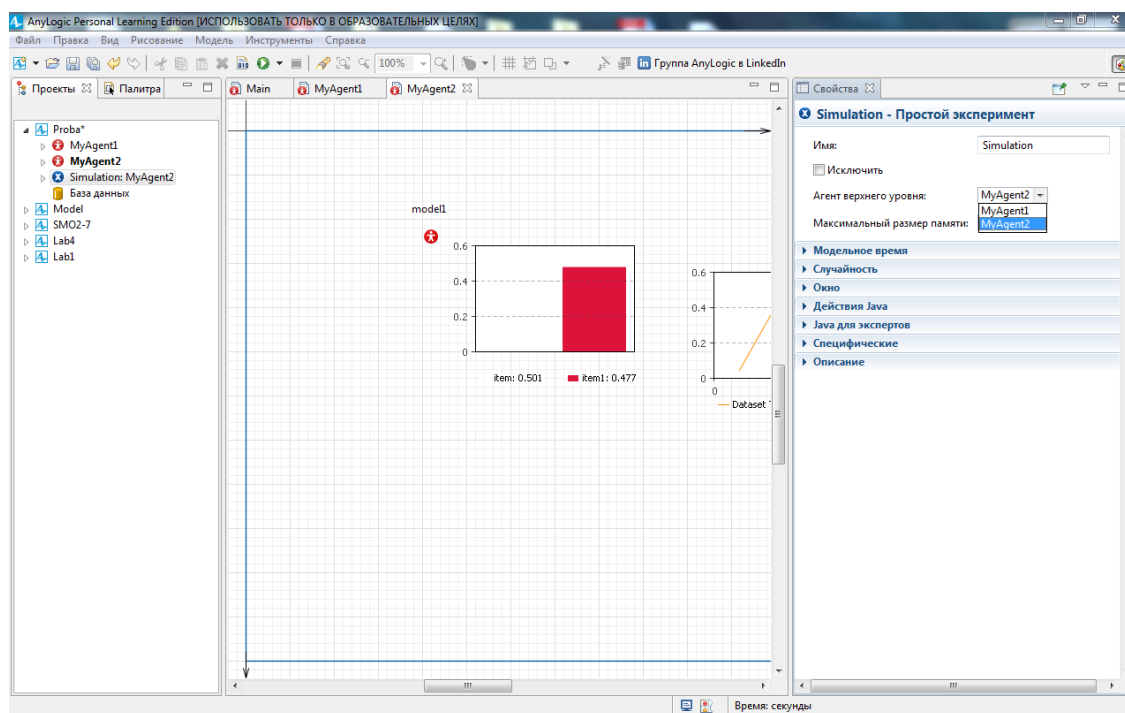


Рисунок 4 Выбор в эксперименте агента верхнего уровня

В AnyLogic можно легко масштабировать модели, создавая в одном проекте модели различные эксперименты с разными агентами верхнего уровня. В итоге, запуская тот или другой эксперимент, можно экспериментировать с различными подмоделями.

Чтобы в модели выбрать агент верхнего уровня:

1. В панели **Проект** щелкните по выбранному эксперименту.

2. В окне выбранного эксперимента на панели **Свойства** из выпадающего списка **Агент верхнего уровня** выберите имя агента.

Эксперименты с моделью

Разработка модели M_S осуществляется с целью исследования её поведения во времени. Как предмет исследования модель M_S в целом представляет собой набор множеств параметров вместе параметром времени $t - \langle X, V, H, Y, t \rangle$, связанных различными, не всегда формально заданными непосредственно в модели, отношениями. Для обозначения всего множества выделенных и представленных в M_S отношений над параметрами будем использовать запись $\otimes_S \langle X, V, H, Y, t \rangle$, а также положим, что $M_S \equiv \otimes_S \langle X, V, H, Y, t \rangle$.

Множество всех параметров модели состоит из подмножества независимых (экзогенных) и зависимых (эндогенных) параметров. Независимые параметры могут быть *управляемыми* или *неуправляемыми*. Значения управляемых параметров можно изменять произвольно. Значения неуправляемых параметров произвольно задаваться не могут. Если неуправляемый параметр является эндогенным, то его значение формируется отношениями, связывающими его с другими параметрами модели. Если неуправляемый параметр является экзогенным, то его значение в модели формируется интерактивными отношениями, связывающим параметр с окружением модели - «внешней средой», к которой относится и параметр времени - t . Среди эндогенных особо выделяют параметры, которые характеризуют исследуемые свойства модели.

Исследование свойств модели, построенной в AnyLogic, осуществляется посредством вычислительных экспериментов с моделью на компьютере. В течение эксперимента выполняется численная реализация заданных в модели отношений и вычисление значений параметров, имитирующая поведение модели во времени. Простой эксперимент может заключаться в том, чтобы для заданных исходных значений управляемых параметров получить зависимость характеристик модели от времени. Более сложные эксперименты могут планироваться с целью нахождения таких значений управляемых параметров, при которых характеристики модели достигают в результате заданных или экстремальных значений. В этом случае потребуется многократный «прогон» модели при различных исходных значениях управляемых параметров и сопоставление получаемых значений характеристик между собой. Очевидно, что эффективность проведения таких экспериментов существенно зависит от наличия в системе специальных средств циклического прогона модели и автоматического варьирования значениями управляемых параметров. AnyLogic предоставляет пользователю широкий спектр средств, позволяющих проводить с моделью эксперименты с различными целями.

Технологически подготовка эксперимента с моделью выражается в создании и включении в проект модели специальных объектов, называемых *экспериментами*. Каждый созданный эксперимент наследует параметры модели типа **Параметр**, получая возможность заменить в эксперименте заданные в модели начальные значения таких параметров собственными начальными значениями. Иными словами, они становятся управляемыми параметрами прогонов эксперимента. При этом в самой модели начальные значения этих параметров сохраняются. Кроме задания начальных значений параметров в эксперименте выбирается и устанавливается модель изменения параметра времени: монотонное («реальное») или спорадическое (виртуальное) время.

Тик часов реального времени по умолчанию устанавливается равным 1с, а начальное значение – равным 0. Однако начальное значение и величину тика при необходимости можно увеличивать или уменьшать.

Тик спорадического времени имеет неопределённое значение, так как завершение очередного тика и увеличение времени связано с событием изменения текущего состояния модели.

Таким образом, проект модели, помимо набора созданных агентов, содержит набор созданных различных экспериментов с собственными настройками параметров, включая модель и параметры времени.

Типы экспериментов

Версия AnyLogic для обучения предоставляет возможность подготавливать и осуществлять следующие типы экспериментов:

- Простой эксперимент.
- Оптимизация.
- Варьирование параметров.

Простой эксперимент

Простой эксперимент заключается в предварительном задании значений параметров модели, и однократном прогоне при запуске модели. Один простой эксперимент с именем *Simulation* всегда создается по умолчанию при создании модели.

Оптимизация

В этом эксперименте осуществляется многократный прогон модели с автоматически изменяющимися в очередном прогоне значениями параметров с целью нахождения их оптимальных значений (с учетом заданных ограничений), при которых заданная целевая функция достигает экстремума.

Варьирование параметров

Этот тип эксперимента заключается в многократном прогоне модели с автоматически изменяющимися в очередном прогоне значениями параметров. Этот эксперимент позволяет сравнить поведение модели при разных значениях параметров, оценить степень влияния отдельных параметров на поведение модели. Кроме того, можно осуществлять несколько прогонов модели с фиксированными значениями параметров, если требуется оценить влияние случайных факторов в стохастических моделях.

Модельное время

В AnyLogic агенты в рамках единой модели существуют и развиваются во времени как параллельные процессы. Важное значение для агента имеют различные возникающие во времени события, которые оказывают влияние на их выполнения. Например, для согласования параллельно развивающихся агентов может потребоваться задержка выполнения агентов на заданный период времени, или ожидание наступления агентом заданного момента времени для завершения своей работы. Кроме того, моменты времени непосредственно используются для получения значений динамических переменных или вычисления функций, зависящих от времени.

Известно, что в каждой вычислительной системе имеются встроенные часы физического (астрономического) времени, позволяющие получать значения текущей календарной даты и времени с точностью до секунды (и даже до миллисекунды). Однако использование физического времени в явном виде для моделирования не всегда является продуктивным. Например, если изучается некий исторический процесс, реальный темп развития которого выражается минимум в годах (единица изменения времени – 1 год), то моделирование его на компьютере в физическом времени лишается практического смысла (слишком долго). Может оказаться и так, что на исследуемом физическом объекте реальные процессы развиваются с темпом в 1 микросекунду. В этом случае вычислительная система может просто не успевать выполнять компьютерную модель в таком темпе (слишком быстро). Следовательно, очевидно, что в таких случаях не продуктивно использовать для моделирования часы физического времени. Поэтому для моделирования используется абстрактный аналог физического времени – *модельное время*.

Сходство между физическим и модельным временем заключается лишь в общем способе получения значений меток времени, а именно, - подсчёт количества прошедших к "текущему моменту" единиц времени. При этом физическое время имеет реальную ("ощущаемую" физически) протяжённость единицы времени, а единица модельного времени имеет абстрактную протяжённость, в течение которой все параллельно выполняемые агенты "пересчитывают" своё состояние, после чего модельные часы увеличиваются на единицу. Из этого следует, что модельное время может либо опережать физическое время, либо отставать от него. Это, в конечном счете, зависит от производительности вычислительной системы.

AnyLogic использует часы физического времени, которые идут с интервалом в 1 секунду. Эти часы являются внешними по отношению ко всем моделям и могут использоваться для связи темпа модельного времени с физическим. Выражается эта связь в том, что задается количество единиц модельного времени, выполняемых в течение одной секунды физического времени. Возможно, модельное время будет искусственно придерживаться, чтобы удовлетворить это требование. Возможно и то, что модельное время окажется медленнее физического, тогда эта зависимость будет формально нарушаться. Однако на корректность результатов моделирования это не будет оказывать существенного влияния. Например, скорость анимации предмета моделирования на экране всего лишь не будет соответствовать скорости изменения его образа в реальной жизни.

AnyLogic предоставляет возможность выбора для модели режима *реального* или *виртуального* модельного времени. Если в модели выбран *режим реального времени*, то темп модельного времени связывают с темпом физического времени. А именно, задается коэффициент связи темпа модельного времени с физическим, значение которого можно выбрать в диапазоне от 1/512 до 512. Если пользователь выбирает коэффициент равный единице, то он хочет, чтобы модельное время соответствовало физическому времени. Иначе, темп времени в модели будет медленнее или быстрее физического.

Если в модели выбран *режим виртуального времени*, интенсивность изменения модельного времени никак не связана с интенсивностью изменения физического времени. Текущий отсчёт виртуального времени изменяется тогда, когда со значением текущего отсчета выполнены в модели все одномоментные действия. Этот режим позволяет не ждать наступления очередного физического момента времени, если в этом нет необходимости, и модель может прожить длинную виртуальную жизнь за короткий период физического времени.

Единицы модельного времени

AnyLogic предоставляет пользователю следующую систему мер времени для интерпретации единицы модельного времени: *миллисекунды, секунды, минуты, часы, дни и недели*. Выбор пользователем конкретной меры модельного времени обычно связан с семантикой исследуемого предмета моделирования. Во многих экспериментах конкретная мера модельного времени вообще может быть не существенной ("подходит" любая единица измерения времени). В этом случае можно задать любую единицу времени

Задание единицы модельного времени

1. В панели **Проект**, выделите эксперимент, для которого Вы хотите изменить единицу модельного времени.
2. Перейдите на страницу **Модельное время** панели **Свойства**.
3. Выберите нужные Вам единицы модельного времени из выпадающего списка **Единицы модельного времени**.

Выбор режима времени и скорости выполнения модели

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* задаётся 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.

Чтобы задать режим времени и скорость выполнения

1. В панели **Проект**, выделите эксперимент, для которого Вы хотите изменить скорость выполнения модели.
2. На странице **Презентация** панели **Свойства** выберите режим времени с помощью кнопок, расположенных в секции **Режим выполнения**.
3. Если Вы хотите, чтобы модель выполнялась в режиме виртуального времени, выберите опцию **Виртуальное время (максимальная скорость)**.
4. Если же Вы хотите, чтобы модель выполнялась в режиме реального времени, выберите опцию **Реальное время со скоростью** и задайте скорость выполнения модели (количество выполняемых в секунду единиц модельного времени AnyLogic) в выпадающем списке справа.

Можно менять режим времени и скорость выполнения модели прямо во время ее выполнения с помощью кнопок панели управления окна презентации. В частности, запустив модель в режиме реального времени, Вы сможете менять скорость выполнения модели, меняя коэффициент скорости моделирования. Коэффициент 1х означает, что модель будет выполняться со скоростью, заданной в свойствах текущего эксперимента; 2х означает, что модель будет выполняться в два раза быстрее заданной скорости, и т.д. Например, если будет задана скорость выполнения модели, равная 6 единицам модельного времени в секунду, то при коэффициенте 2х в 1 секунду будет выполняться 12 единиц модельного времени.

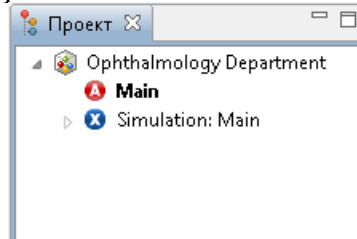
Задание начального и конечного времени моделирования

Чтобы задать начальное и конечное время моделирования

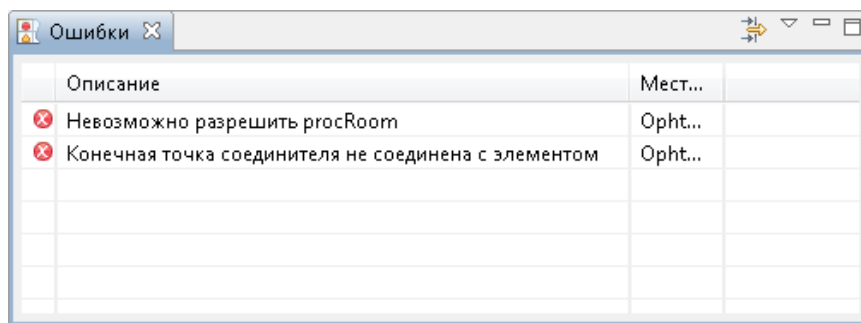
1. В панели **Проект**, выделите нужный эксперимент.
2. Перейдите на страницу **Модельное время** панели **Свойства**.
3. По умолчанию модель будет работать 100 единиц модельного времени и затем остановится. Если Вы хотите, чтобы модель работала до тех пор, пока Вы сами ее не остановите, выберите **Нет** из выпадающего списка **Остановить**.
4. Если же Вы хотите, чтобы моделирование было прекращено в какой-то определенный момент модельного времени, то Вы можете задать момент остановки либо как календарную модельную дату, либо как количество единиц модельного времени, по прошествии которого модель должна быть остановлена. В этом случае выберите **В заданное время** из выпадающего списка **Остановить** и задайте момент остановки моделирования с помощью расположенных ниже элементов управления.
5. Если Вы хотите задать интервал моделирования как количество единиц модельного времени, убедитесь, что флажок **Использовать календарь** сброшен, и введите конечное время моделирования в поле **Конечное время**.
6. Если же Вы хотите задать интервал моделирования с помощью календарных дат, установите флажок **Использовать календарь** и выберите начальную и конечную дату моделирования с помощью элементов управления **Начальная дата** и **Конечная дата**.

Запуск модели

Перед запуском выбранного эксперимента можно предварительно задать значения параметров модели. Можно создать несколько экспериментов с различными установками и изменять рабочую конфигурацию модели, просто запуская тот или иной эксперимент модели. В панели **Проект** эксперименты отображаются в нижней части дерева модели.




Прежде чем запустить модель на выполнение необходимо построить её исполняемый программный образ. Это можно сделать с помощью кнопки панели инструментов **Построить модель** (при этом фокус AnyLogic должен быть в рабочей области на каком-то элементе именно этой модели). Если при построении модели выявляются какие-нибудь ошибки, то построение не будет завершено, и в панель **Ошибки** будет выведена информация об ошибках, обнаруженных в модели.

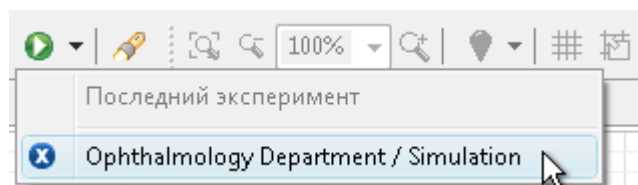




Панель Ошибки

Двойным щелчком мыши по выделенной строке ошибки в этом списке инициируется переход к предполагаемому месту ошибки. После исправления всех выявленных ошибок модель будет успешно построена и её можно будет запустить.

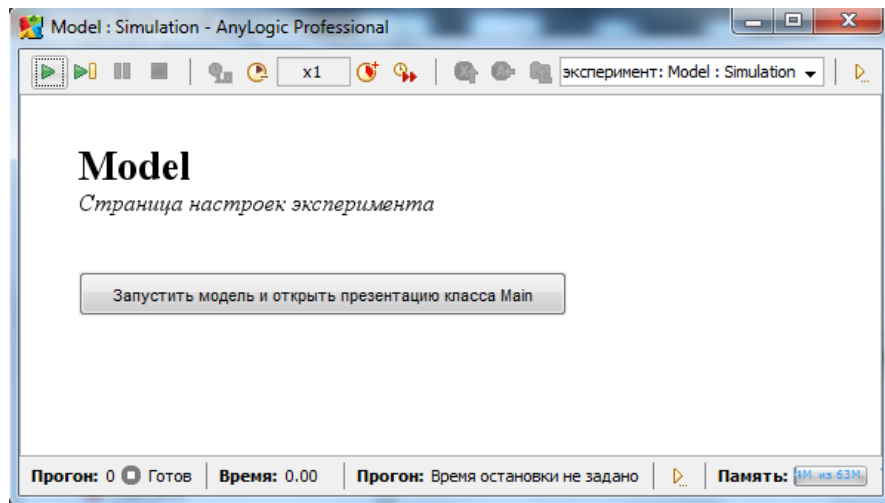
Действия при запуске модели


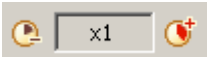
1. Щелкните мышью по кнопке панели инструментов  - запустить, и выберите из открывшегося списка **Последний эксперимент** имя эксперимента, который Вы хотите запустить. Например, это может выглядеть так:



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

В результате открывается окно эксперимента. В нем при выполнении прогона модели будет отображена презентация запущенного эксперимента. В начале в окно каждого простого эксперимента AnyLogic автоматически помещает название модели и кнопку **Запустить модель и открыть презентацию класса Main**:



При нажатии кнопки запуска модель запускается, в результате чего осуществляется переход на презентацию выбранного агента верхнего уровня этого эксперимента (по умолчанию - Main). При желании можно изменять режим модельного времени с реального на виртуальный – кнопка **Реальное/виртуальное время** - , а в режиме реального времени можно изменять скорость выполнения модели с помощью кнопок **Замедлить** и **Ускорить** - .

Простой эксперимент

При создании модели автоматически создается один простой эксперимент, названный Simulation. Он запускает модель с заданными значениями параметров, поддерживает режимы виртуального и реального времени, анимацию, отладку модели. Вы можете создать несколько экспериментов с разными настройками и легко менять текущие настройки запуска модели, запуская тот или иной эксперимент.

Чтобы создать простой эксперимент

1. В панели **Проект**, щелкните правой кнопкой мыши по модели, для которой Вы хотите создать новый эксперимент, и выберите **Создать|Эксперимент** из контекстного меню. Появится диалоговое окно **Новый эксперимент**.
2. Выберите **Простой эксперимент** из списка **Тип эксперимента**.
3. Введите имя эксперимента в поле **Имя**.
4. Выберите корневой агент для этого эксперимента из выпадающего списка **Агент верхнего уровня**.
5. Если Вы хотите применить к создаваемому эксперименту временные установки другого эксперимента, оставьте установленным флажок **Копировать установки модельного времени из** и выберите эксперимент из расположенного справа выпадающего списка.
6. Щелкните мышью по кнопке **Готово**.

Оптимизационный эксперимент

Чтобы создать оптимизационный эксперимент

1. В панели **Проект**, щелкните правой кнопкой мыши по элементу модели и выберите **Создать|Эксперимент** из контекстного меню. Появится диалоговое окно **Новый эксперимент**.

2. Выберите **Оптимизация** из списка **Тип эксперимента**.
3. Введите имя эксперимента в поле **Имя**.
4. Выберите корневой агент для этого эксперимента из выпадающего списка **Агент верхнего уровня**.
5. Если Вы хотите применить к создаваемому эксперименту временные установки другого эксперимента, оставьте установленным флажок **Копировать установки модельного времени из** и выберите эксперимент из расположенного справа выпадающего списка.
6. Щелкните мышью по кнопке **Готово**.

Основные свойства

Имя – Имя эксперимента. ⚠ Поскольку AnyLogic генерирует для каждого эксперимента соответствующий Java класс, при задании имени эксперимента нужно руководствоваться правилами названия классов в Java. Начинайте имя эксперимента с заглавной буквы.

Агент верхнего уровня – Здесь задается агент верхнего уровня этого эксперимента. Агент этого класса будет играть роль корня иерархического дерева агентов модели, запускаемой этим экспериментом.

Исключить – Если опция выбрана, то эксперимент будет исключен из модели.

Создать интерфейс – Кнопка удаляет содержимое презентации эксперимента и создает интерфейс эксперимента заново согласно его текущим установкам (набору оптимизационных параметров и их свойствам и т.д.). ⚠ Создавайте новый интерфейс только *после того*, как закончите задание параметров эксперимента.

Генератор случайных чисел – В стохастических моделях важную роль играет заданное начальное число генератора случайных чисел - от этого будет зависеть, будут ли "прогоны" стохастической модели уникальными или воспроизводимыми.

Случайное начальное число (уникальные "прогоны") – Если опция выбрана, то "прогоны" модели будут уникальными и не смогут быть воспроизведены в силу того, что при каждом новом запуске модели генератор случайных чисел будет инициализироваться другим числом.

Фиксированное начальное число (воспроизводимые "прогоны") – Если опция выбрана, то генератор случайных чисел модели будет всегда инициализироваться одним и тем же начальным числом (оно задается в поле **Начальное число**), поэтому все запуски модели будут идентичными и воспроизводимыми.

Целевая функция – Функционал оптимизации (целевая функция, которую необходимо **минимизировать** или **максимизировать**). Для доступа к параметрам корневого агента эксперимента следует использовать имя **root**. Например, **root.param**

Количество итераций – Если опция выбрана, то оптимизация будет остановлена, если будет выполнено максимальное количество итераций, заданное в расположенном справа поле.

Автоматическая остановка – Если опция выбрана, то оптимизация будет остановлена, если значение целевой функции перестанет существенно улучшаться (эта опция называется *автоостановом оптимизации*).

Параметры – Здесь пользователь задает набор параметров, которые будут варьироваться оптимизатором (оптимизационные параметры). В таблице перечислены все параметры корневого агента. Чтобы разрешить варьирование параметра оптимизатором, перейдите на соответствующую строку таблицы **Параметры**, щелкните мышью в ячейке **Тип** и выберите тип параметра, отличный от значения **фиксированный**. Список возможных значений будет меняться в зависимости от типа параметра: **набор**, **int**, **дискретный** для целочисленных параметров типа *int*, **непрерывный** и **дискретный** для вещественных параметров типа *double* и т.д. Задайте диапазон допустимых значений параметра. Введите нижнюю границу диапазона в ячейке **Мин** и верхнюю - в ячейке **Макс**. Для параметров **набор** и **дискретный** нужно также указать в ячейке **Шаг** величину шага (инкремента), с помощью которого будут определяться допустимые значения данного параметра (первое допустимое значение равно заданной нижней границе интервала, следующее равно сумме первого значения и заданного шага и т.д.).

Ограничения

Здесь пользователь может задать ограничения и требования - дополнительные ограничения, накладываемые на параметры и найденные решения оптимизатором.

Требования (проверяются после "прогона" для определения того, допустимо ли найденное решение) – В этой таблице Вы можете задать требования - дополнительные ограничения, накладываемые на получаемые оптимизатором решения. Если на момент окончания прогона модели заданные требования будут выполняться, то полученный оптимизатором набор значений параметров будет признан допустимым, и результат оптимизации будет запомнен оптимизатором. Если ограничения удовлетворены не будут, то найденные значения параметров и решение будут считаться недопустимыми. Выражение требования может содержать любые арифметические операции над данными модели. В выражении также можно вызывать созданные Вами функции, а также предопределенные функции (sin(), cos(), sqrt() и т.д.). Агент верхнего уровня эксперимента доступен здесь как root. Вы можете отключить требование, сделав его неактивным, сбросив флажок в ячейке **Вкл.** соответствующей строки таблицы.

Эксперимент варьирования параметров

AnyLogic позволяет оценить степень и характер влияния отдельных параметров на поведение модели. Для этого не нужно самостоятельно запускать модель раз за разом, вручную менять значения параметров между запусками, и пытаться отследить какие-то закономерности, анализируя результаты каждого запуска по отдельности. С помощью *эксперимента варьирования параметров* модель будет автоматически запускаться заданное количество раз с варьирующимися значениями выбранных параметров. Изучить и сравнить поведение модели при разных значениях параметров Вы сможете с помощью диаграмм AnyLogic.

Чтобы провести эксперимент варьирования параметров

1. Создайте новый эксперимент варьирования параметров.
2. Настройте этот эксперимент, т.е. выберите параметры, которые Вы хотите варьировать, и задайте диапазоны возможных значений этих параметров.
3. Запустите эксперимент.

Чтобы создать эксперимент варьирования параметров

1. В панели **Проект**, щелкните правой кнопкой мыши по модели, для которой Вы хотите создать новый эксперимент, и выберите **Создать|Эксперимент** из контекстного меню.
2. Появится диалоговое окно **Новый эксперимент**.
3. Выберите **Варьирование параметров** из списка **Тип эксперимента**.
4. Введите имя эксперимента в поле **Имя**.
5. Выберите корневой агент для этого эксперимента из выпадающего списка **Корневой класс модели**.
6. Если Вы хотите применить к создаваемому эксперименту временные установки другого эксперимента, оставьте установленным флажок **Копировать установки модельного времени из** и выберите эксперимент из расположенного справа выпадающего списка.
7. Щелкните мышью по кнопке **Готово**.









Графическое моделирование динамических систем

При моделировании информационных систем, являющихся системами реального времени, к которым относятся системы промышленной автоматизации, робототехнические устройства, встроенные системы управления, например, летающими аппаратами и т.п., своё окружение они рассматривают как физическую среду, выступающую в качестве источника и/или приёмника физических данных, формирующих информационную нагрузку системы. Состояние физического объекта обычно выражается в изменяющихся во времени его физических параметрах в ответ на посланные управляющие воздействия. При этом система должна обеспечивать сбор данных о состоянии системы и выдачу управляющих воздействий на физический объект в режиме реального времени. Разрабатывать и отлаживать программные приложения для систем реального времени, используя реальный физический объект, практически либо невозможно, либо крайне неэффективно. Поэтому реальный физический объект заменяют программной моделью, имитирующей поведение физического объекта во взаимодействии с разрабатываемой информационной системой. Во многих случаях предмет моделирования представляется как непрерывно-детерминированная динамическая система.

Моделирование непрерывно-детерминированных динамических систем связано с необходимостью отражения в модели различного рода факторов и отношений между ними, характеризующих свойства и поведение предмета моделирования в виде алгебраических и/или дифференциальных уравнений. Для непрерывно-детерминированных моделей характерно то, что выделяемые при анализе предмета моделирования факторы и отношения между ними имеют некоторый физический смысл: координаты местоположения, скорость, ускорение, сила, концентрация вещества и т.п., и характеризуются непрерывно изменяющимися во времени детерминированными значениями, а параметр времени соответствует модели непрерывного времени. Обычно количественные факторы выражаются вещественными значениями, а отношения между ними - алгебраическими или дифференциальными уравнениями. В итоге модель составляют набор абстрактных параметров, моделирующих различные выделенные факторы предмета моделирования, связи между параметрами и способ их изменение во времени.

Графические элементы построения D-схем

AnyLogic предоставляет широкую палитру графических элементов, которые могут быть использованы для формализации предмета моделирования как динамической системы – *D-схемы*. В AnyLogic D-схема представляется графическими диаграммами, элементы которых формализуют различные *факторы* и *отношения* между ними, выделяемые при анализе предмета моделирования. И факторы, и отношения между ними специфицируются в виде значков графических элементов, которые могут быть использованы для построения D-схем, располагаются в окне **Палитра** на панелях **Агент** или **Системная динамика** и выглядят следующим образом:

-  - параметр,
-  - переменная,
-  - динамическая переменная,
-  - накопитель,
-  - поток,
-  - связь.
-  - функция,
-  - табличная функция,

-Параметр


Тип **Параметр** позволяет выражать в модели свойства предмета моделирования как эндогенные или экзогенные параметры. Важной особенностью использования параметров в модели является то, что при создании в проекте объекта "эксперимент" они автоматически наследуются этим объектом как параметры настройки эксперимента. Тип **Параметр** предоставляет разработчику широкий выбор способов формирования и/или изменения значений параметра. Параметр в модели может интерпретироваться как:

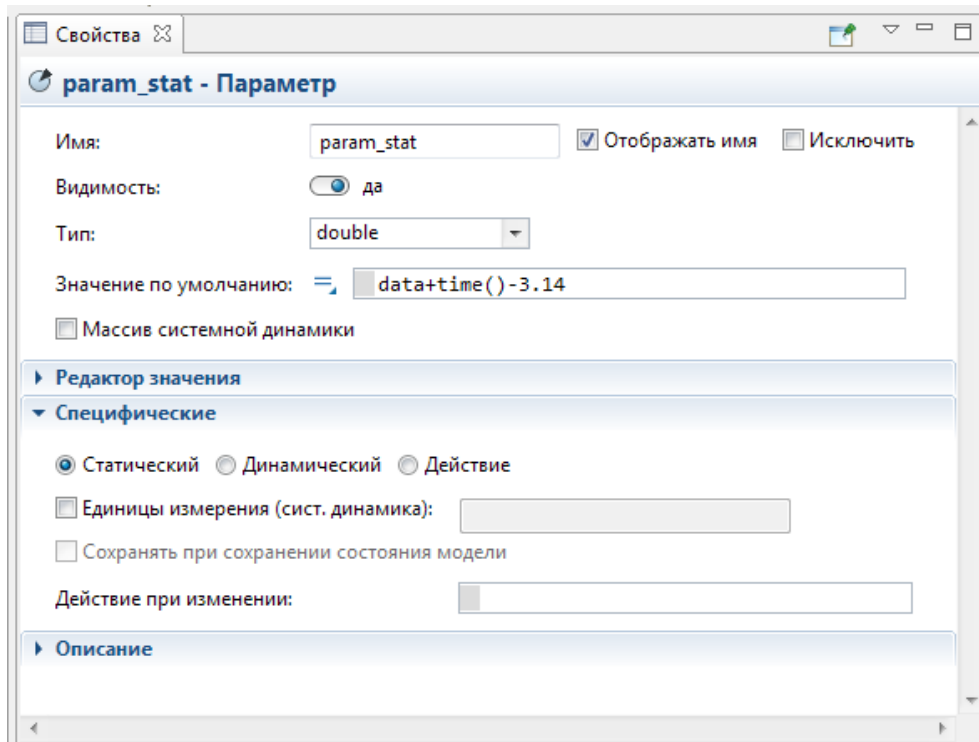
- экзогенный статический параметр заданного типа, хранящий значение, которое меняется операцией присвоения при выполнении в модели программного кода;
- эндогенный динамический параметр, значение которого формируется как результат вызова функции с именем параметра, возвращающей значение заданного типа;
- абстрактная процедура, выполняемая в результате вызова функции с именем параметра, не имеющей возвращаемого значения.

AnyLogic позволяет создавать в модели параметры различных типов:

- базовых типов языка Java,
- системных и созданных в модели прикладных агентных типов,
- стандартной библиотеки классов языка Java²
- абстрактных классов на языке Jav.

Чтобы включить параметр в диаграмму агентного типа необходимо:

1. Перетащить элемент  - **Параметр**, из палитры в окно графического редактора на диаграмму редактируемого агентного типа и перейти в окно **Свойства** для настройки его свойств.
2. Специфицировать способ использования параметра в модели и настроить его свойства на панели **Свойства**:



The screenshot shows the 'Свойства' (Properties) window for a parameter named 'param_stat'. The window has a title bar with standard OS controls and a close button. Below the title bar, the parameter name 'param_stat - Параметр' is displayed with a small icon. The main area contains several settings:

- Имя:** A text field containing 'param_stat'.
- Видимость:** A toggle switch set to 'да' (yes).
- Тип:** A dropdown menu showing 'double'.
- Значение по умолчанию:** A text field containing 'data+time()-3.14'.
- Массив системной динамики:** An unchecked checkbox.
- Редактор значения:** A section with a 'Специфические' (Specific) sub-section.
 - Статический / Динамический / Действие:** Three radio buttons, with 'Статический' (Static) selected.
 - Единицы измерения (сист. динамика):** An empty text field.
 - Сохранять при сохранении состояния модели:** An unchecked checkbox.
 - Действие при изменении:** An empty text field.
- Описание:** A section with an empty text area.

² Стандартные Java-классы можно найти по адресу <http://java.sun.com/javase/6/docs/api/>

Статический параметр

По умолчанию включённый в диаграмму параметр является экзогенным статическим. Выражение, заданное при определении параметра, используется для получения его начального значения. При доступе где-либо в модели к статическому параметру его значение автоматически *не пересчитывается* в соответствии с заданным выражением. Однако можно явно изменять значения статической переменной в выполняемом программном коде, и даже интерпретировать такие изменения как возникающие события, связывая с ними инициацию выполнения заданного программного кода.

Изменение значения статического параметра

Тип **Параметр** с признаком **Статический** хорошо подходит для отражения в модели экзогенных факторов предмета моделирования. Для изменения значения параметра во время прогона модели должен явно выполняться программный код с операцией присваивания статическому параметру нового значения. Например, в некотором программном коде целочисленному параметру **param_stat** можно явно присвоить значение 10:

```
param_stat=10;
```

Событие изменения значения статического параметра

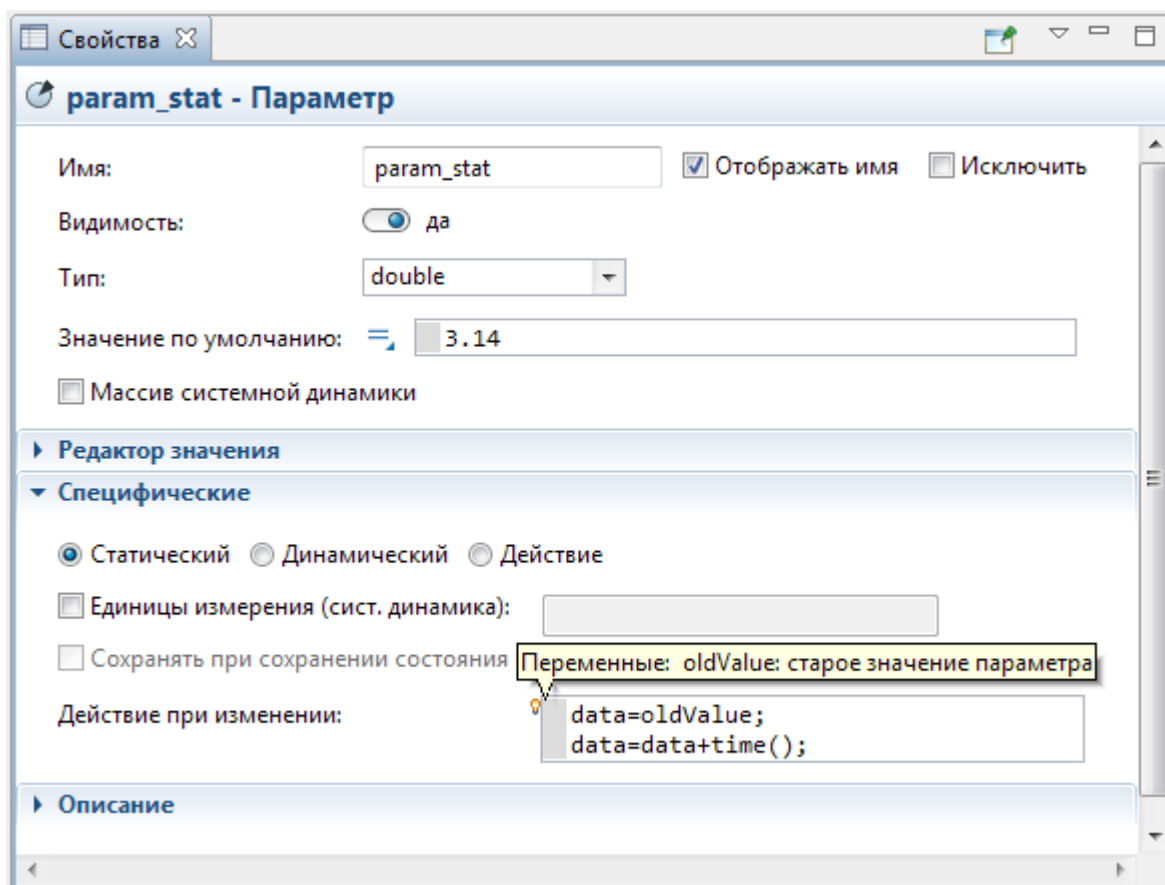
Факт изменения значения статического параметра операцией присваивания не рассматривается в модели как ожидаемое событие. Если же в модели такое событие требует реакции, то изменение значения статического параметра следует выполнить не присваиванием, а специальным методом:

```
set_<имя_статического_параметра>(<значение>)
```

В AnyLogic для каждого статического параметра такой метод изменения его значения генерируется автоматически. Например, если при выполнении модели требуется текущее значение статического параметра с именем **param_stat** изменить на значение 123 и одновременно инициировать в модели запланированную реакцию в виде программного действия, то для параметра следует программно выполнить метод:

```
set_param_stat(123); /*изменение значения параметра с именем  
param_stat, инициирующее действие*/
```

Такое изменение параметра инициирует выполнение программного кода, заданного в свойствах статического параметра в поле - **Действие при изменении**:



Замечания:

- В поле программного кода **Действие при изменении** доступна системная локальная переменная с именем **oldValue**, которая сохраняет старое (заменённое) значение параметра.
- **Действие при изменении** статического параметра не будет выполнено, если новое значение статического параметра окажется равным текущему значению данного параметра!

Динамический параметр

Динамический параметр подходит для отражения в модели эндогенного фактора предмета моделирования, значение которого формируется в зависимости от других факторов. Для этого параметру нужно назначить свойство *динамический* и обращаться к нему за значением в форме вызова функции с именем динамического параметра (с аргументами или без аргументов):

<имя_динамического_параметра>(<список_значений_аргументов>)

Параметр становится динамическим, если в свойствах **Специфические** выбрана опция **Динамический**:

Свойства ✕

param_din - Параметр

Имя: ☒ Отображать имя ☐ Исключить

Видимость: ☒ да

Тип:

Значение по умолчанию:

☐ Массив системной динамики

Аргументы:

Имя	Тип
a1	int
a2	double

▶ Редактор значения

▼ Специфические


☐ Статический ☒ Динамический ☐ Действие

☐ Единицы измерения (сист. динамика):

▶ Описание

В поле **Значение по умолчанию** следует разместить выражение для вычисления значения динамического параметра. Выражение может содержать формальные параметры (аргументы), которые определяются в области **Аргументы**. Можно задать любое выражение, результат вычисления, возвращаемый функцией, будет иметь тот же тип, что и заданный тип параметра. При обращении к динамическому параметру в форме вызова функции формальные аргументы заменяются фактическими значениями, а в качестве значения параметра функция возвращает значение вычисленного выражения, приведённого к заданному типу параметра.

Чтобы пользователь мог отличать в модели, какие параметры объектов являются динамическими, а какие нет, в панели свойств объекта справа от метки параметра рисуется специальный значок, говорящий о том, что параметр используется как динамический:

Время задержки: 

Вместимость:

Максимальная вместимость:

Каждый раз, когда осуществляется обращение к такому параметру, его значение будет пересчитываться заново в соответствии с заданным выражением (с заданными аргументами), то есть динамический параметр ведёт себя как динамическая переменная. Но принципиальное отличие динамического параметра от динамической переменной заключается в том, что с помощью динамических параметров можно параметризовать объекты одного и того же агентного

типа разными выражениями (значениями по умолчанию). У динамической переменной такой возможности нет. Использование динамических параметров при создании агентного типа позволяет вводить в модель однотипные объекты с различным поведением их динамических параметров и настраивать их при подготовке экспериментов.

Параметр-действие

При необходимости можно использовать параметр для явной инициации выполнения заданного программного кода (абстрактной процедуры) при обращении к нему в программном коде в форме вызова функции типа **void** с именем параметра (не возвращающей значения) с аргументами или без аргументов. Для этого параметру нужно назначить свойство **Действие**, сформировать программный код процедуры по умолчанию и обращаться к нему в форме вызова функции с именем параметра и с аргументами или без аргументов:

<имя_динамического_параметра>(<список_значений_аргументов>)

Параметр становится функцией, если в свойствах **Специфические** выбрана опция **Действие**:

param_proc - Параметр

Имя: ☒ Отображать имя ☐ Исключить

Видимость: ☒ да

Действие по умолчанию:

Аргументы:

Имя	Тип
a1	int
a2	double

Редактор значения

Специфические

☐ Статический ☐ Динамический ☒ Действие

☐ Единицы измерения (сист. динамика):

Описание


В поле **Действие по умолчанию** размещается программный код с формальными аргументами, а не выражение как у динамического параметра.

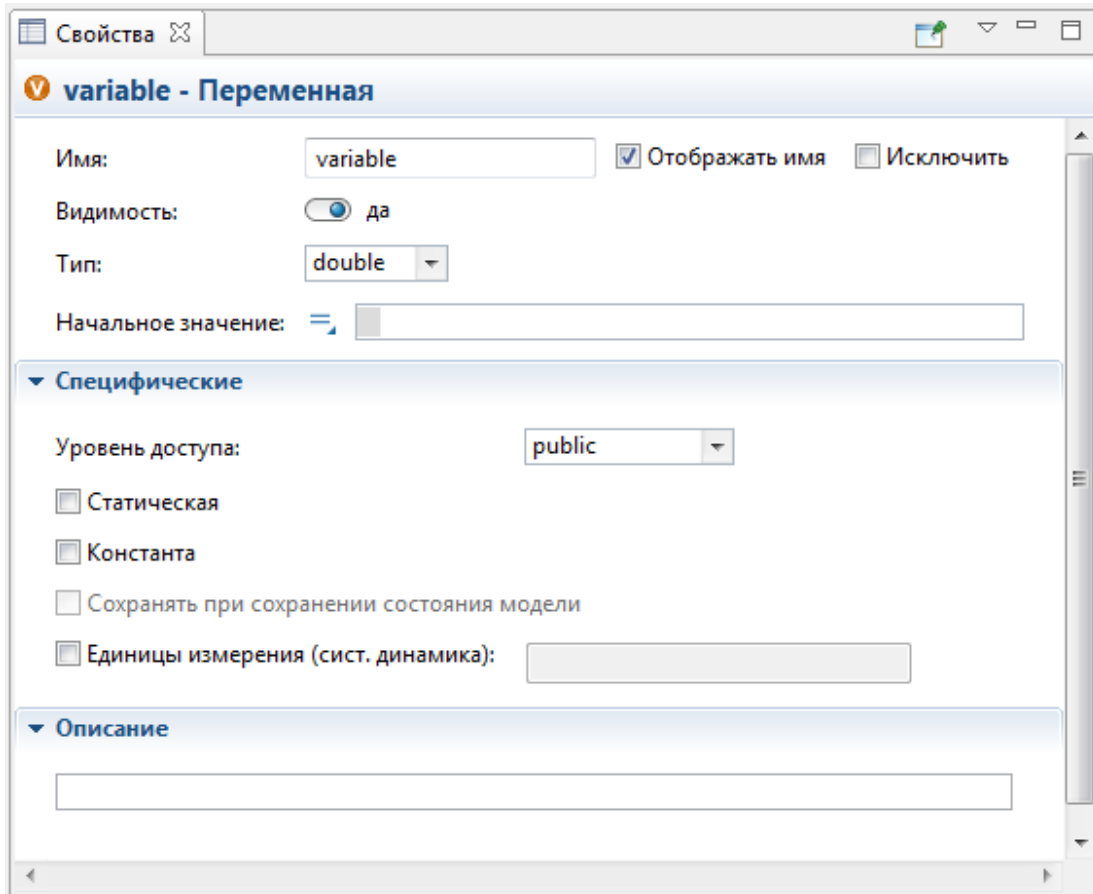
v-Переменная

Тип **Переменная** (или **Простая переменная**) предоставляет элементарный способ отражения в модели экзогенных факторов предмета моделирования, не характеризующихся сложным

поведением. Переменная может быть любого скалярного типа или даже Java-класса. В отличие от параметра, при создании эксперимента переменная не наследуется в качестве параметра настройки эксперимента. Так как переменная моделирует экзогенный фактор, то способы изменения значения переменной во время выполнения эксперимента с моделью ограничивается обычной операцией присвоения либо в программном коде, либо с помощью элементов управления значениями переменных, заданных на презентации.

Для включения переменной в модель требуется:

1. Перетащить элемент **Переменная**  из палитры **Агент** на диаграмму типа агентов.
2. В панели **Свойства** задать имя переменной в поле **Имя**. Это имя будет использоваться для идентификации и доступа к переменной:



3. С помощью выпадающего списка **Тип** можно задать для переменной один из базовых типов (**int**, **double**, **boolean**, **String**), или задать тип любого Java класса (в этом случае нужно выбрать опцию **Другой** и ввести имя класса в расположенном справа поле).
4. Если в течение всего моделируемого периода времени переменная должна всегда иметь одно и то же значение, можно объявить её *константой*, выбрав флажок **Константа** в секции свойств **Специфические**. Сделав переменную константой, она будет защищена от изменения во время выполнения модели.
5. Если объявить переменную *статической*, выбрав в секции свойств **Специфические** опцию **Статическая**, то переменная будет иметь одно и то же значение во всех имеющихся в модели экземплярах данного класса (например, во всех агентах популяции). Следует, однако, иметь в виду, что нельзя использовать статические переменные, если планируется проводить сложный эксперимент (оптимизационный эксперимент, эксперимент варьирования параметров и т.д.).
6. В поле **Начальное значение** можно задать выражение для получения начального значения переменной. Это значение может быть изменено во время выполнения модели операцией присваивания. Если переменная объявлена константой, то нельзя будет использовать в выражении для начального значения факторы, представленные как *параметры*, *накопители*, *потоки* и *динамические переменные*.

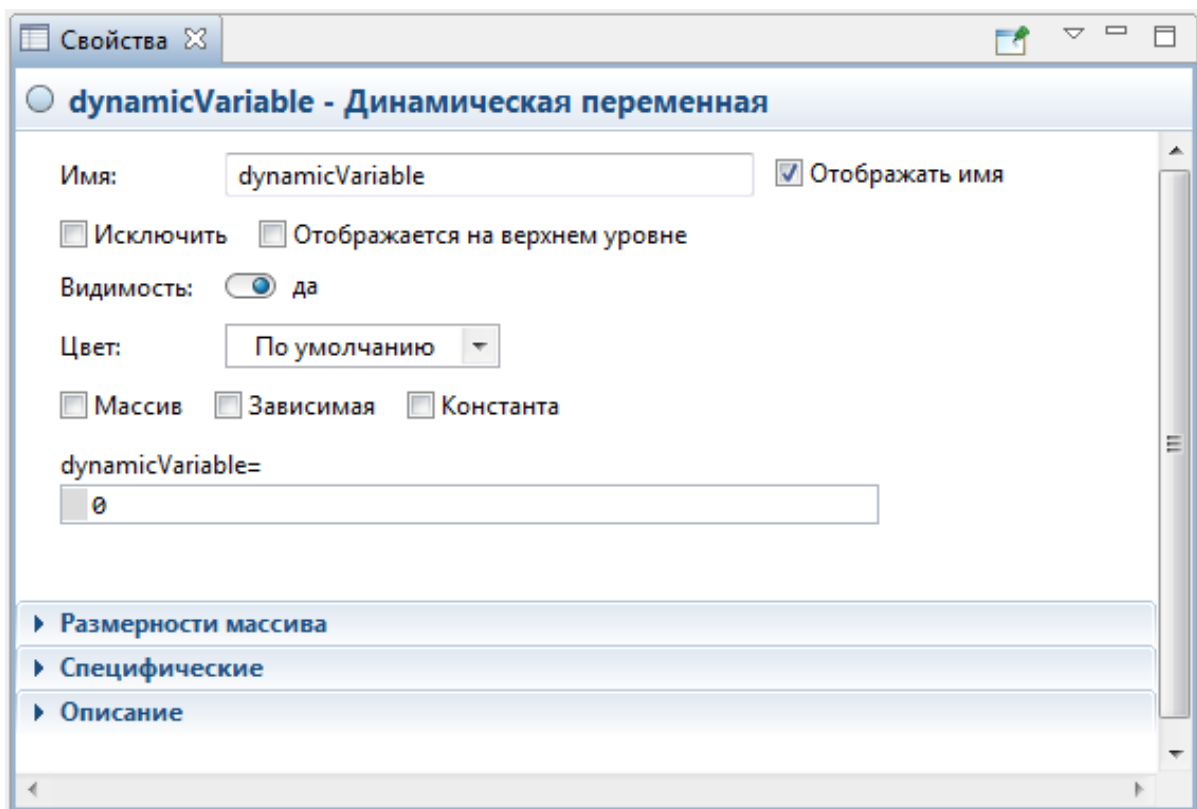
●-Динамическая переменная

Тип **Динамическая переменная** предназначен для определения в модели эндогенного фактора (или далее - "динамической переменной"), значение которого всегда формируется как результат вычисления выражения, заданного при определении. Это выражение связывает данный фактор с другими факторами модели - как экзогенными, так и эндогенными. При определении в типе агента динамической переменной её *тип* явно не задаётся. Неявно (по умолчанию) тип динамической переменной ассоциируется с *типом результата* заданного выражения. Но если в выражении используются только базовые типы числовых значений, то неявно типом результата является **double**.

Важно отметить, что каждый раз при использовании где-то в модели динамической переменной её значение будет каждый раз предварительно автоматически пересчитываться в соответствии с заданным для неё исходным выражением. Этим динамическая переменная принципиально отличается от факторов типа **Параметр** или **Переменная**, при обращении к которым получают их текущие значения, так как изменить значение параметра или переменной можно только явным выполнением программного кода.

Чтобы добавить динамическую переменную в диаграмму, выполните следующее:

1. Перетащите элемент **Динамическая переменная** ● из палитры **Системная динамика** на диаграмму типа агента.
2. Перейдите на панель **Свойства** и настройте свойства переменной:



Программное управление динамической переменной

Можно ли в программном коде изменить динамическую переменную? Да, можно, выполнив операцию присвоения. Но при первом же обращении к ней в другом месте её текущее значение будет автоматически вычислено по заданному исходному выражению. Некорректное использование динамических переменных приводит к "необъяснимым" изменениям их текущих значений.

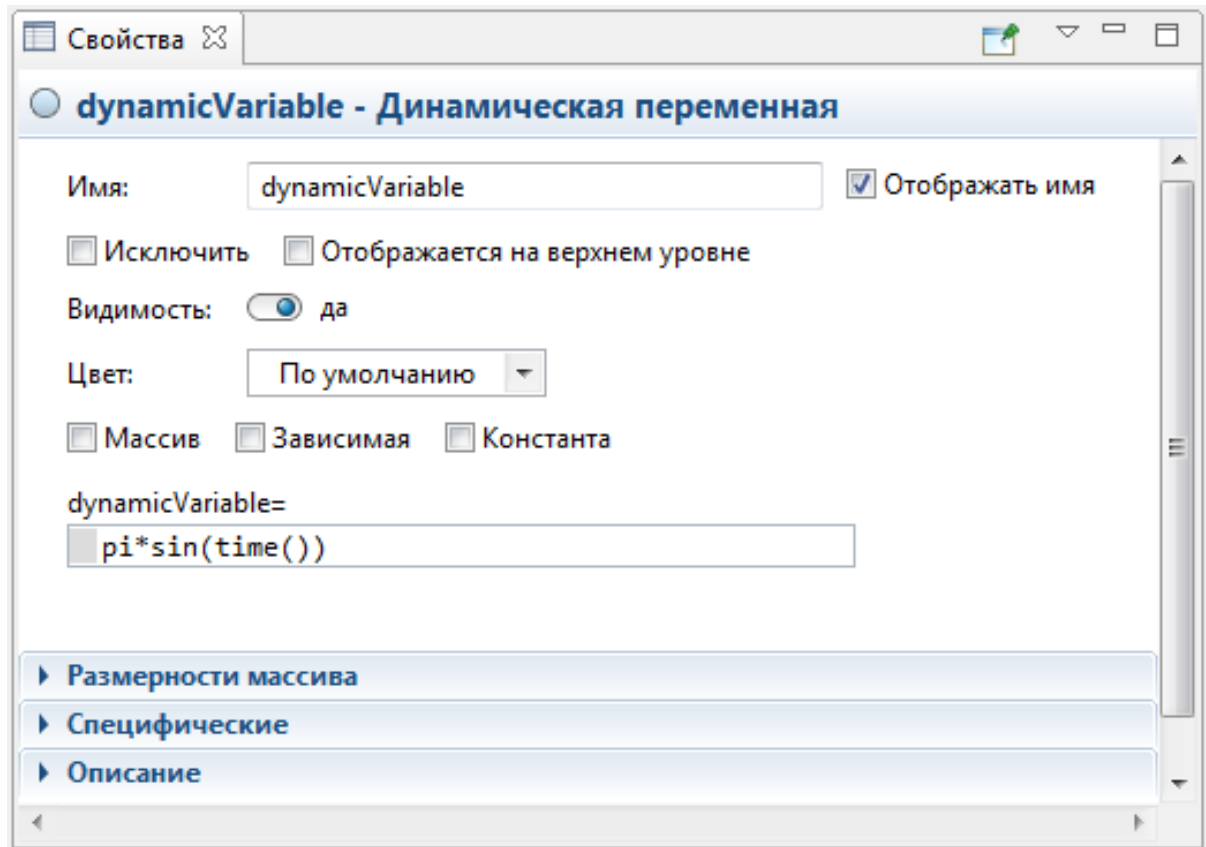
Способ изменения и использования значений динамической переменной при выполнении модели зависит от выбора её вида при включении в создаваемый или редактируемый тип агента.

Динамическая переменная может иметь один из двух исключаящих друг друга видов использования её значения при выполнении модели:

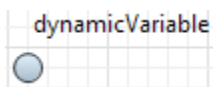
- функциональный (по умолчанию),
- параметрический (явно выбираемый).

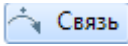
Функциональное использование динамической переменной

При выборе функционального вида (по умолчанию) значение динамической переменной задаётся в свойствах в форме любого *вычисляемого выражения*, с типом которого ассоциируется значение, полученное динамической переменной. Например:



На диаграмме агентного типа графический символ функциональной динамической переменной будет иметь вид:



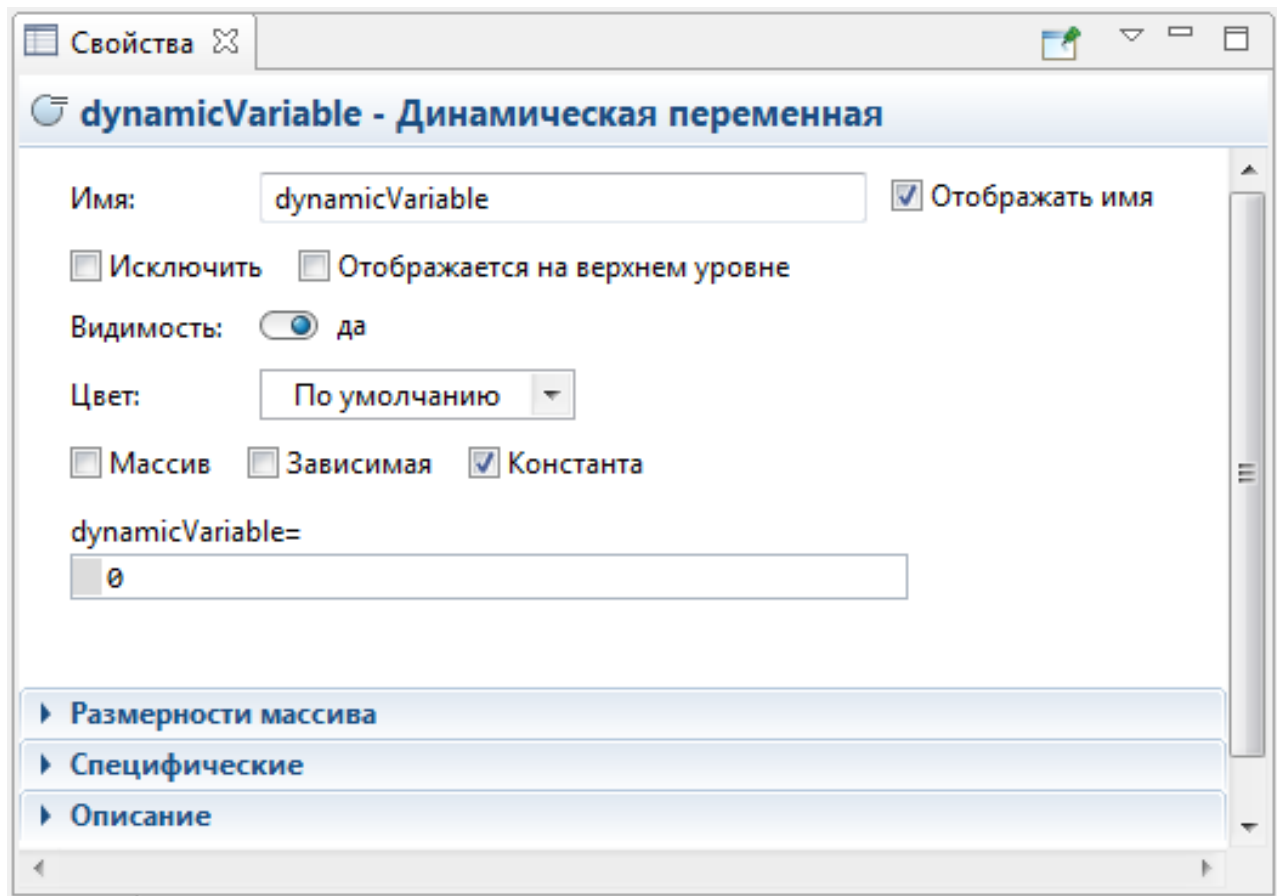
Корректное задание на диаграмме типа агента функциональной динамической переменной требует соединить её связями со всеми входящими в заданное выражение и представленными на диаграмме факторами. Связи выражаться дугами , направленными от заданных в выражении факторов в сторону зависящей от них динамической переменной. В противном случае при построении модели AnyLogic укажет на наличие ошибок.

Параметрическое использование динамической переменной

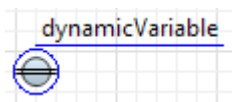
Параметрическое использование динамической переменной отменяет автоматический пересчёт её текущего значения при доступе. При параметрическом использовании динамической переменной её текущее значение будет задаваться последней явно выполненной в некотором программном коде операцией присваивания. Такую динамическую переменную ассоциируют с "константой" в том смысле, что при её использовании в качестве члена вычисляемого выражения

она автоматически обновляться не будет (в отличие от функциональной динамической переменной). Изменить значение параметрической динамической переменной можно только явно выполнив операцию присваивания в некоторой процедуре действия.

Динамическая переменная становится параметрической, если в качестве её свойств явно выбрать опцию **Константа**:



На диаграмме типа агента параметрическая динамическая переменная будет представлена графически в следующем виде:



Выражение, заданное в свойствах, только инициализирует начальное значение параметрической динамической переменной, которое в дальнейшем можно будет изменить только явно, программно выполняя операцию присвоения значения в процедурах действий.

Накопитель

Накопители используются в динамических системах для моделирования факторов H_t , функционально зависящих от времени, но явный вид функциональной зависимости не известен, и неявно задаётся дифференциальным уравнением в форме Коши с производной этой функции по времени:

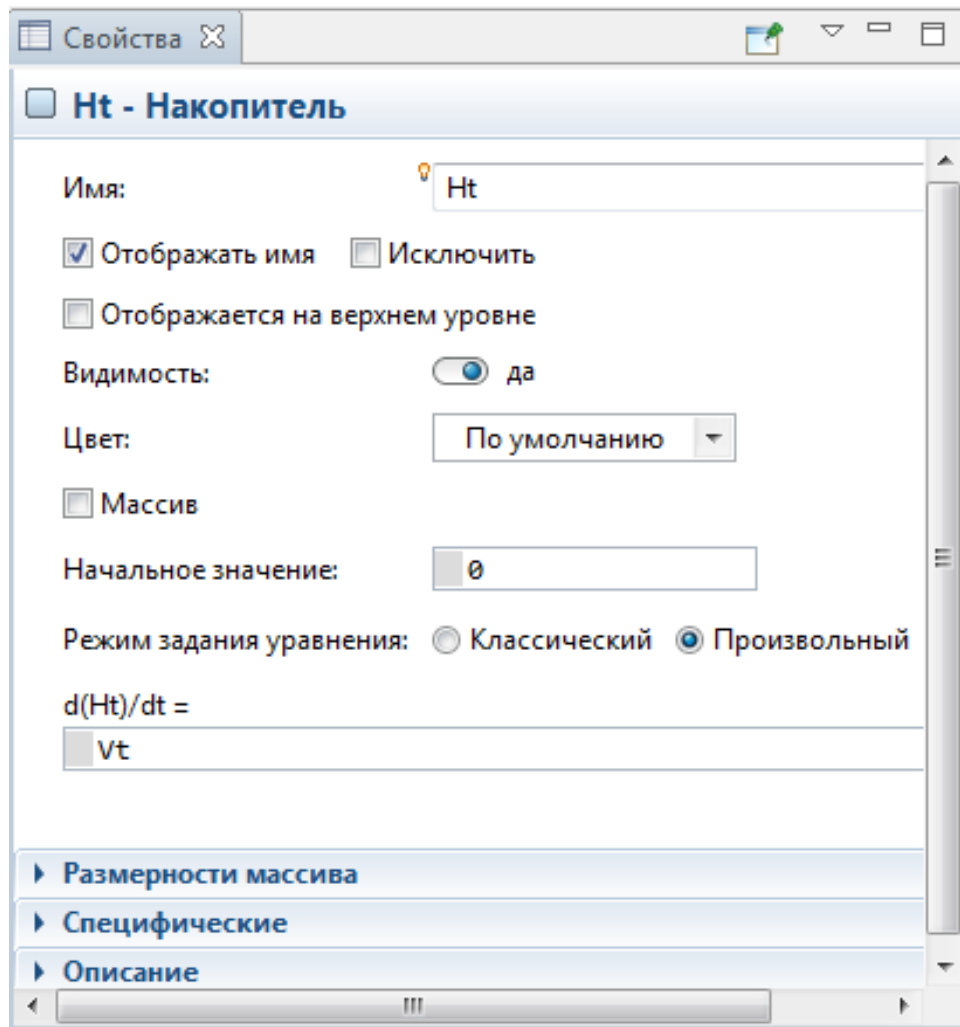
$$\frac{dH(t)}{dt} = V(t).$$

Данная запись означает, что значение накопителя Ht в процессе эксперимента с моделью будет автоматически вычисляться для каждого момента модельного времени и будет изменяться во времени согласно функции $H(t)$, удовлетворяющей этому дифференциальному уравнению. Нахождение значения функции $H(t)$ в текущий момент времени осуществляется встроенным в

AnyLogic численным методом интегрирования дифференциального уравнения. В процессе выполнения эксперимента с моделью

Чтобы создать накопитель

1. Перетащите элемент **Накопитель** из палитры **Системная динамика** в окно графического редактора диаграммы формируемого типа агента.
2. Перейдите на панель **Свойства** и задайте дифференциальное уравнение в произвольном режиме:



3. Введите в поле формулу дифференциального уравнения вычисления интенсивности изменения значения накопителя.

Поток

Поток – это тип динамической переменной, используемой, как правило, в моделях *системной динамики*. Динамические переменные используются для моделирования фактора, участвующего в формировании интенсивности изменения значения некоторого накопителя. Переменная *поток* не может существовать самостоятельно без связи с некоторым накопителем. При этом у накопителя выбирается *классический* режим задания дифференциального уравнения, правая часть которого формируется автоматически как сумма связанных с ним входящих и исходящих потоков:

$$\langle \text{входящий поток 1} \rangle + \langle \text{входящий поток 2} \rangle + \dots - \langle \text{исходящий поток 1} \rangle - \langle \text{исходящий поток 2} \rangle \dots$$

AnyLogic отображает получившиеся зависимости между переменными диаграммы потоков и накопителей с помощью стрелок.

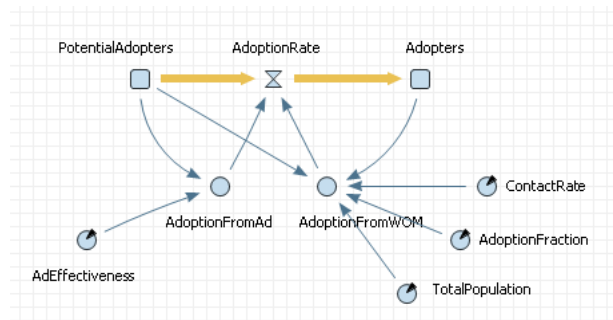

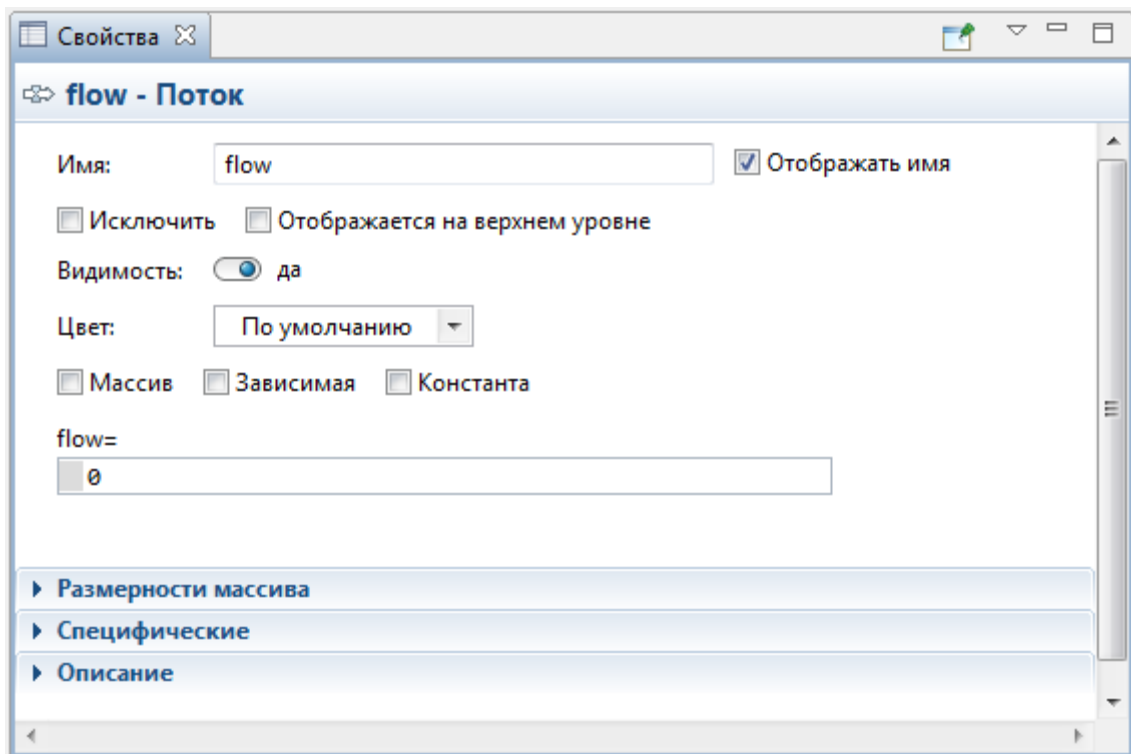


Рисунок 5 Пример диаграммы потоков и накопителей

Зависимости между накопителями и потоками отображаются в виде толстых стрелок. Входящие потоки отображаются в виде стрелок, направленных от переменной-потока к переменной-накопителю, в то время как исходящие потоки наоборот - от накопителя к потоку. Тонкие стрелки отображают зависимости переменных. Стрелка, направленная от переменной **A** к переменной **B** означает, что переменная **A** упоминается в формуле переменной **B**.

Чтобы создать поток

1. Перетащите элемент **Поток**  из палитры **Системная динамика** в окно графического редактора диаграммы агентного типа.
2. Перейдите на панель **Свойства** и настройте свойства потока (см. справку AnyLogic):



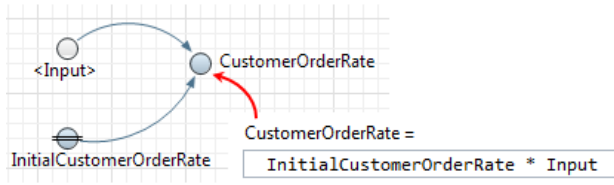
Связь

Связь используется для задания зависимости между элементами диаграммы: параметрами, переменным, накопителями и потоками. AnyLogic не рисует стрелки взаимосвязей автоматически по заданным формулам, а предлагает пользователю самостоятельно графически задавать связи

между зависимыми элементами диаграммы (но для удобства работы AnyLogic поддерживает механизм быстрого добавления отсутствующих связей, см. ниже).

Есть два типа зависимостей:

- Какой-то элемент (это может быть накопитель, поток, вспомогательная переменная или параметр) упоминается в формуле потока или вспомогательной переменной. Такой тип связи отображается сплошной линией:



- Какой-то элемент системной динамики упоминается в выражении начального значения накопителя. Этот тип связи отображается пунктирной линией:



Если какой-то элемент А упоминается в формуле или в выражении начального значения элемента В, то Вам нужно будет вначале соединить эти элементы связью, направленной из элемента А в элемент В, и только затем задать выражение, в котором фигурирует переменная А, в свойствах элемента В.

Полярность связи

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. При этом считается, что значения остальных элементов не изменяются.





- 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. При этом считается, что значения остальных элементов не изменяются.

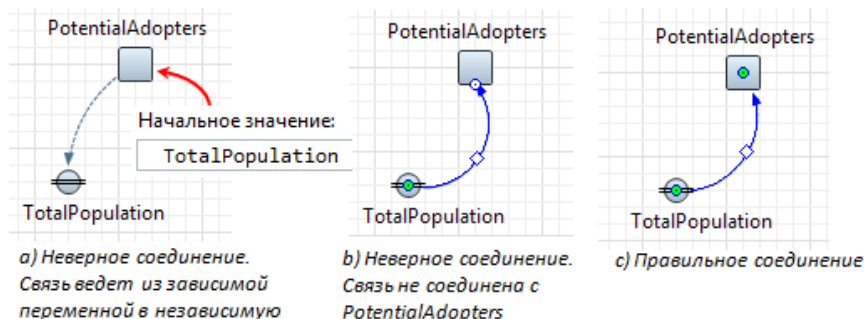


Вы можете добавить рядом со связями метки, которые будут обозначать полярность этих связей. Обычно полярность обозначается с помощью символов + или - рядом со стрелкой связи. Таким образом, Вы можете показать, как зависимая переменная изменяет своё значение при изменении значения независимой переменной.

Создание связи

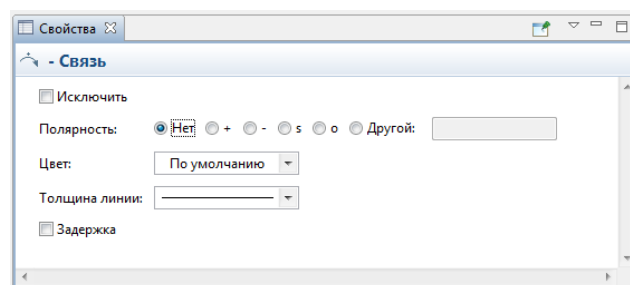
Чтобы добавить связь между двумя элементами системной динамики:

1. Сделайте двойной щелчок по элементу **Связь**  в палитре **Системная динамика**, так что значок этого элемента изменится на .
2. Сразу после этого щёлкните в графическом редакторе по тому элементу, который упоминается в формуле или поле начального значения.
3. Чтобы закончить рисование связи, щёлкните по элементу, в свойствах которого упоминается элемент, от которого Вы рисуете эту связь.
4. Когда Вы закончите рисование связи, она будет выделена в графическом редакторе. Если Вы успешно нарисовали связь, соединив ею оба требуемых элемента, то Вы увидите на конечных точках маленькие зелёные кружки (см. рисунок с ниже). В противном случае, если Вы увидите хотя бы один белый кружок (см. рисунок *b*), то это будет означать, что соответствующий конец связи не был успешно соединён. Чтобы исправить это, нужно будет перетащить эту конечную точку связи на требуемый элемент диаграммы потоков и накопителей (пока индикатор соединения не станет зелёным).



Обратите внимание, что важно рисовать связь в правильном направлении - от исходной (независимой) переменной к зависимой, в уравнении которой упоминается исходная (а не наоборот, как на рисунке *a* выше).

5. Перейдите в панель **Свойства**, чтобы задать свойства связи:

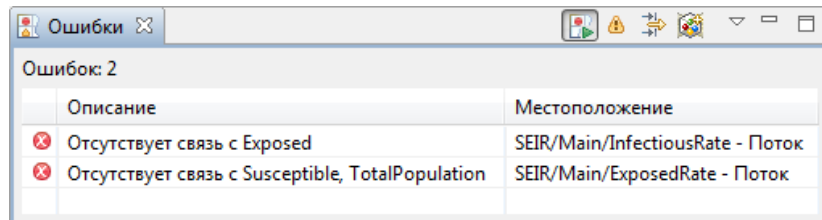


Механизм быстрого добавления отсутствующих связей

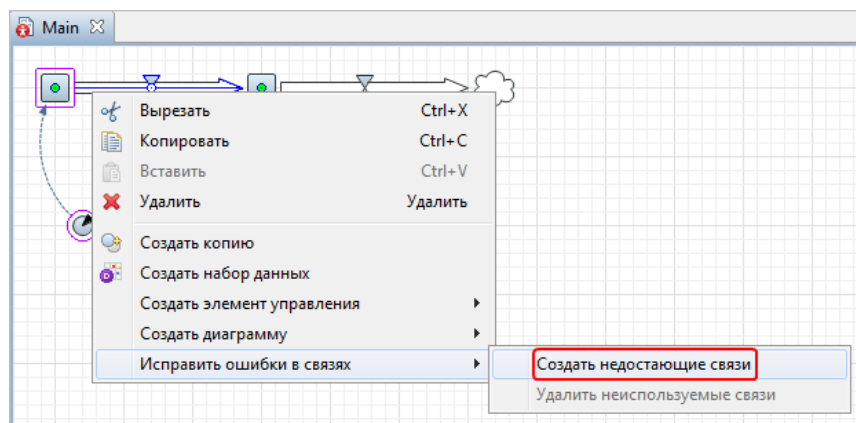
AnyLogic поддерживает удобный механизм быстрого добавления отсутствующих на диаграмме связей.

Чтобы добавить на диаграмму несколько отсутствующих связи

1. Если в формуле или выражении начального значения какого-либо элемента модели системной динамики упоминаются некоторые переменные, но они не соединены с этим элементом связями, AnyLogic отобразит это в панели **Ошибки**:





2. Найдите элемент, содержащий такое выражение, в графическом редакторе и щёлкните по нему правой кнопкой мыши.
3. В контекстном меню Вы увидите опцию **Исправить ошибки в связях**. Выберите далее **Создать недостающие связи**, чтобы добавить все отсутствующие связи для этого элемента.



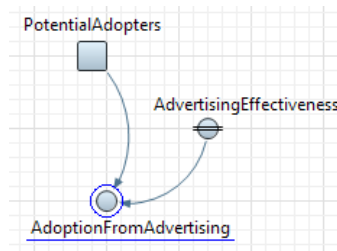
4. Все необходимые связи будут тут же созданы автоматически. **Панель Ошибки** закроется, поскольку конфликты будут разрешены.
5. Если у элемента имеются лишние связи, Вы можете их удалить с помощью опции **Удалить неиспользуемые связи** в контекстном меню.

Чтобы добавить на диаграмму отсутствующую связь

1. Допустим, что в формуле или выражении начального значения, скажем, переменной variableA упоминается переменная variableB, которая не соединена связью с переменной variableA.
2. Перейдите в панель **Свойства** переменной, которая вызывает ошибку (variableA) и щелкните мышью в поле формулы. В левой части поля Вы увидите индикатор ошибки.

AdoptionFromAdvertising =   AdvertisingEffectiveness * PotentialAdopters

3. Щелкните по этому индикатору левой кнопкой мыши. Вы увидите контекстное меню. Выберите из него пункт **Создать связь для [имя_переменной]**. Соответствующая связь будет нарисована на диаграмме.




Ф Функция

AnyLogic позволяет пользователям создавать и графически отображать на диаграмме свои собственные функции. С помощью функций можно единожды задать определённую последовательность действий (обычно - вычислений, возвращающих результат), которую можно будет выполнять в разных агентах и в разные моменты жизни модели. Функция может быть полностью написана на языке Java. Однако AnyLogic поддерживает и определение функции в графическом виде. Для этого используются *диаграммы действий* - структурированные блок-схемы, позволяющие задавать алгоритмы графически в стиле структурного программирования. Диаграммы действий облегчают задание алгоритмов, делая необязательным знание синтаксиса Java-операторов. Использование диаграмм действий даёт ещё одно преимущество: графическое представление алгоритма упрощает понимание функции пользователями модели.

Определение функции

Чтобы задать функцию

1. Перетащите элемент **Функция**  из палитры **Основная** на диаграмму агентного типа (или эксперимента).
2. Перейдите на страницу **Основные** панели **Свойства** и задайте свойства функции.
3. Введите имя функции в поле **Имя**.
4. Если функция только выполняет какие-то действия, но не возвращает никакого результат проведённых вычислений, укажите, что эта функция не возвращает ничего, выбрав в группе кнопок **Тип возвращаемого значения** **void**.
5. Если Вам нужно, чтобы функция возвращала результат проведённых ею вычислений, то Вам нужно будет указать, какого типа будет возвращаемое значение. Функция может возвращать как значение одного из наиболее часто используемых типов (**int**, **double**, **boolean**, **String**), так и значение любого другого Java класса (в этом случае Вам нужно будет выбрать опцию **Другой** и ввести имя класса в расположенном справа поле).
6. Функция может быть объявлена *статической*. Чтобы сделать функцию статической, установите флажок **Статическая**. Статическая функция не требует создания экземпляров того агентного типа, в котором она задана. Статическая функция *моя_функция*, объявленная в агентном типе *MyClass*, будет доступна из любого места модели как:

```
MyClass.моя_функция().
```

7. Перейдите на страницу **Код** панели **Свойства** и введите тело функции в поле **Тело функции**. Последняя строка должна возвращать значение с помощью оператора **return**:

```
return <выражение, возвращающее значение заданного типа>;
```

Основные свойства

Имя – Имя функции.

Отображать имя – Если опция выбрана, то имя функции будет отображаться в графическом редакторе.

Исключить – Если опция выбрана, то функция будет исключена из модели.

На верхнем уровне – Если опция выбрана, то функция будет видна на презентации класса, в который будет вложен данный агент. Поскольку функции не могут отображаться на диаграмме класса верхнего уровня, этот флажок всегда сброшен и недоступен.

На презентации – Если опция выбрана, то функция будет отображаться на презентации во время выполнения модели.

Уровень доступа – Уровень доступа к функции. Есть четыре уровня доступа:

- **private** – функция доступна только из этого агентного типа,
- **protected** - функция доступна из этого агентного типа и его подклассов,
- **default** - функция доступна из любого места модели,
- **public** - функция доступна из всех открытых моделей.

Статическая – Если опция выбрана, то функция будет *статической*. Статическая функция не требует создания экземпляров того агентного типа, в котором она задана. Статическая функция *моя_функция*, объявленная в агентном типе MyClass, будет доступна из любого места модели как MyClass.моя_функция().

Тип возвращаемого значения– Тип возвращаемого функцией значения. Если Вам нужно, чтобы функция возвращала результат проведенных ею вычислений, то Вам нужно будет указать здесь, какого типа будет возвращаемое значение. Функция может возвращать как значение одного из наиболее часто используемых типов (**int**, **double**, **boolean**, **String**), так и значение любого другого (и это уникальная особенность AnyLogic) Java класса (в этом случае Вам нужно будет выбрать опцию **Другой** и ввести имя класса в расположенном справа поле). Если функция не возвращает ничего, выберите **void**.

Аргументы функции – Здесь Вы можете задать аргументы функции, с помощью которых Вы сможете передавать функции данные, необходимые для вычислений. Каждый аргумент задается в отдельной строке таблицы.

Код

Тело функции – Введите здесь тело функции. Функции пишутся на языке Java, поэтому в Вашем распоряжении имеются все преимущества этого языка, такие, как, например, условные операторы (if-then-else), циклические операторы (while, for), операторы ветвления (switch) и т.д. Также Вы можете оперировать здесь аргументами функции. Последняя строка должна возвращать значение с помощью оператора **return**.

Аргументы функции

Если нужно передавать функции какие-то данные, необходимые для проведения вычислений, можно использовать для этого *аргументы функции*.

Чтобы задать аргументы функции

1. Выберите функцию в графическом редакторе или в панели **Проект**.
2. Задайте аргументы в таблице **Аргументы функции** на странице **Основные** панели **Свойства**. Каждый аргумент задаётся в отдельной строке таблицы.
3. Введите имя аргумента в ячейке **Имя**.
4. Укажите тип аргумента в ячейке **Тип**. Щелкните мышью в ячейке и выберите нужный Вам тип из выпадающего списка, либо же введите его самостоятельно.

Если у функции заданы аргументы, то при каждом вызове этой функции будет нужно передавать ей значения этих аргументов (в том же порядке, в каком они заданы в таблице **Аргументы функции**).

Порядок следования аргументов в таблице можно изменять с помощью кнопок  и .

Чтобы удалить аргумент, выделите соответствующую строку в таблице и щелкните по кнопке .

Уровень доступа к функции

По умолчанию функции могут вызываться из любого места модели, в которой они заданы. Можно ограничить доступ к функции, изменив её уровень доступа с помощью выпадающего списка **Уровень доступа**. Есть четыре уровня доступа:

- private* – функция доступна только из этого агентного типа,
- protected* - функция доступна из этого агентного типа и его подклассов,
- default* - функция доступна из любого места модели,
- public* - функция доступна из всех открытых моделей.

Табличная функция

AnyLogic поддерживает специальный тип функций – табличные функции. *Табличная функция* – это функция, заданная в табличной форме. Она может быть сделана непрерывной с помощью интерполяции и экстраполяции. Табличные функции обычно используются для задания сложных нелинейных зависимостей, которые не могут быть описаны с помощью стандартных функций, или для приведения собранных с какой-то периодичностью и заданных в виде таблицы экспериментальных данных к непрерывному виду.

Табличная функция работает следующим образом: пользователь задает функцию путем задания пар значений (аргумент, значение), т.е. определенного количества базовых точек графика XY. Основываясь на этих данных и на выбранном типе интерполяции, AnyLogic строит табличную функцию. Для того, чтобы получить значение, которое принимает табличная функция для заданного аргумента, вызовите табличную функцию (так же, как и обычную функцию) по ее имени, передав ей в качестве параметра значение аргумента. Если аргумент будет лежать за областью допустимых значений, то функция вернет значение в соответствии с заданным Вами поведением (задается в свойстве функции **Если аргумент выходит за пределы**).

Чтобы задать табличную функцию


1. Перетащите элемент **Табличная функция**  из палитры **Агент** на диаграмму типа агентов (или эксперимента).
2. Перейдите в панель **Свойства** и задайте свойства табличной функции.
3. Введите имя функции в поле **Имя**. Задайте данные табличной функции в секции **Табличные данные**.
4. Выберите тип интерполяции функции из выпадающего списка Интерполяция.
5. Выберите требуемый тип поведения функции при выходе аргумента за пределы заданного диапазона из выпадающего списка **Если аргумент выходит за пределы**.

График табличной функции можно увидеть в секции предварительного просмотра на странице свойств этой функции. Красная область на этом графике обозначает область недопустимых значений функции.

Диаграммы действий

Моделирование не может существовать без возможности задания процедур обработки данных, изменяющих состояние агента. В окне графического редактора с диаграммой агентного типа кроме пиктограмм параметров, переменных, накопителей и потоков, характеризующих свойства предмета моделирования, одновременно можно располагать и графические структуры, моделирующие поведение предмета моделирования (так называемые *параллельные активности*). К ним относятся *диаграммы действий* и *диаграммы состояний*. В процессе эксперимента с агентом все диаграммы, характеризующие и свойства и поведение агента, существуют параллельно, одновременно во времени и согласовано, как разные составляющие единого организма.

Диаграммы действий - структурированные блок-схемы. Диаграммы действий используются для визуального задания функций, описывая алгоритм их выполнения в наглядной форме. Диаграммы действий в большинстве случаев делают необязательным знание особенностей синтаксиса Java-операторов при определении сложных функций.

Диаграмма действий собирается из блоков, расположенных на странице **Диаграмма действий** панели **Палитра**:



Диаграмма действий

Добавляя на диаграмму этот блок, Вы создаёте простейшую диаграмму действий, состоящую из начальной точки (задаваемой собственным блоком "диаграмма действий") и блока "вернуть значение". Блок "диаграмма действий" задаёт основные свойства определяемой функции - её тип возвращаемого значения, аргументы, уровень доступа и т.д.



Код

Блок **Код** позволяет добавлять в Вашу диаграмму действий *фрагменты кода*. С помощью таких блоков Вы можете задавать на языке Java последовательность действий, которые необходимо выполнить в процессе выполнения алгоритма.



Решение (If.. Else)

Блок **Решение (If.. Else)** является простейшим способом ветвления алгоритма. Он обеспечивает выполнение *фрагментов кода* в соответствии с условием.

У блока есть две исходящие ветви - *true* и *false*. С помощью других блоков диаграммы действий Вы можете задать последовательность действий для каждой из этих ветвей. Когда управление дойдет до данного блока, будет приниматься решение о том, по какой ветви блока управление пойдёт дальше. Если заданное для блока *условие* будет выполнено, то будет выбрана ветвь *true*. В противном случае - ветвь *false*.



Локальная переменная

Используется для объявления новой локальной переменной в диаграмме действий. Локальная переменная будет видна не во всей диаграмме действий, а только в той ее части, которая следует за точкой объявления переменной.

Цикл While является одним из трех блоков диаграммы действий, предназначенных для реализации циклов итераций. Циклы необходимы для того, чтобы повторить некоторые действия несколько раз.



Цикл While

Цикл While выполняется до тех пор, пока заданное для этого цикла *условие* будет истинно (принимает значение *true*). Как только условие принимает значение "ложно", цикл завершается и идёт переход к следующему блоку диаграммы действий.

Цикл While очень похож на **Цикл Do While**. Единственным отличием является то, что истинность выражения проверяется не в конце каждой итерации, а в начале. Следовательно, **Цикл While** может не выполниться ни разу (истинность выражения проверяется в начале каждой итерации, и если с самого начала значением выражения будет ложно, то выполнение цикла будет сразу же прекращено).

Цикл Do While является одним из трех блоков диаграммы действий, предназначенных для реализации циклов итераций. Циклы необходимы для того, чтобы повторить некоторые действия несколько раз.



Цикл Do While

Цикл Do While выполняется до тех пор, пока заданное для этого цикла *условие* будет истинно (принимает значение *true*). Как только условие принимает значение "ложно", цикл завершается и идёт переход к следующему блоку диаграммы действий.

Цикл Do While очень похож на **Цикл While**. Единственным отличием является то, что истинность выражения проверяется не в начале каждой итерации, а в конце. Следовательно, первая итерация цикла **Do While** выполнится обязательно (истинность выражения проверяется только в конце итерации).

Цикл. Есть две формы цикла:



Цикл For

Итератор по коллекции: итеративно проходит по всем элементам указанной коллекции. На каждой итерации выполняется заданное действие, в котором доступен очередной элемент коллекции.

Цикл со счетчиком: выполняет заданные для этого цикла действия несколько раз, до тех пор, пока не выполнится заданное условие.




Вернуть значение (Return)

Блок **Вернуть значение (Return)** играет две роли: во-первых, он определяет, какое значение будет возвращать диаграмма действия (если ее *тип возвращаемого значения* не *void*), и во-вторых, немедленно возвращает это значение, завершая тем самым процесс.



Выход из цикла (Break)

Блок **Выход из цикла** управляет выполнением цикла. Он останавливает текущую итерацию цикла (и опционально также выходит из этого цикла, не выполняя не только текущую, но и оставшиеся итерации).

Создание диаграммы действий начинается с размещения в любом месте окна графического редактора диаграмм значка  - **Диаграмма действий**. При этом будет создана базовая конструкция, состоящая из начальной точки (собственно блока **Диаграмма действий**) и блока **Вернуть значение (Return)**. После этого Вы можете добавлять согласно задаваемому Вами алгоритму другие блоки диаграммы действий в созданную структуру.

Вызов диаграммы действий

Для вызова функции, заданной с помощью диаграммы действий, используется тот же синтаксис, что и для самих функций. Вы должны указать имя диаграммы действий, за которым следуют скобки, в которых через запятую перечисляются передаваемые диаграмме значения аргументов (если они есть), например:

```
мояФункция ()
move (15, 35)
```

Управление выполнением модели


AnyLogic предоставляет набор встроенных функций, позволяющих программно управлять выполнением модели (см. справку AnyLogic):

- по событию приостановить выполнение модели;
- возобновить выполнение при срабатывании события соответствующего перехода диаграммы состояний;
- по событию завершить выполнение модели.


Функция может быть вызвана в любой из параллельных активностей (диаграмм) агента модели: из потока выполнения модели, из кода действия элемента управления, из действия по щелчку фигуры. Например, часто используемой функцией является функция **finishSimulation()** возвращает true и модель сразу останавливается.

Функции программного управления выполнением моделью обычно включаются в состав кодов действий на панели свойств, связанных с такими компонентами диаграмм как *события*, *состояния* и *переходы*, находящимися на палитре **Агент**.

Событие

События являются самым простым способом планирования действий в модели. Событие обозначается на диаграмме значком .

Чтобы задать событие:

1. Перетащите элемент **Событие**  из палитры **Агент** на диаграмму типа агентов.
2. Вы можете изменить его имя в небольшом текстовом редакторе, который появится справа от элемента в графическом редакторе. Закончив ввод нового имени, нажмите Enter.
3. В панели **Свойства**, с помощью выпадающего списка **Тип события**, выберите нужный тип события и сконфигурируйте его с помощью приведённых ниже свойств, специфичных для данного типа события.

4. Задайте действие события в секции **Действие** свойств события. Здесь можно написать Java код или задать это **действие** с помощью функции или диаграммы действий и поместить сюда вызов этой функции который будет выполняться при возникновении этого события.

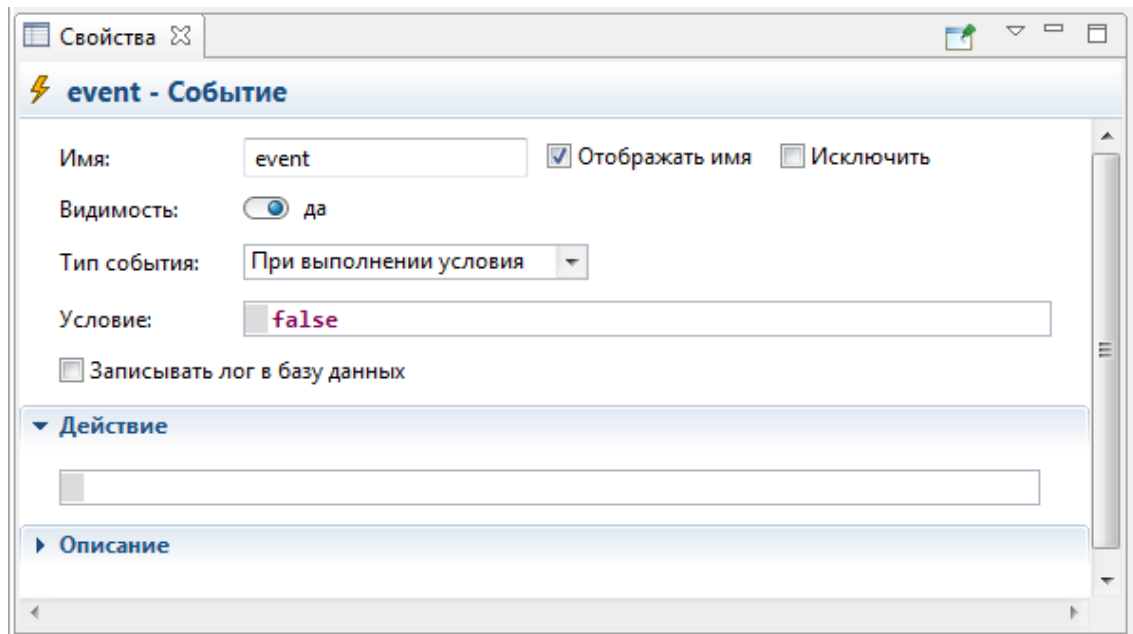







Диаграмма состояний

При усложнении модели в ней накапливается много всяких событий, ассоциированных с различными видами поведения агентов. В качестве событий могут выступать истечение заданного таймута, получение агентом сообщения, выполнение в текущий момент времени заданного логического условия над параметрами модели и т.п. Исследователю становится затруднительно ясно понимать событийно-поведенческую структуру модели. В этом случае сложное поведение агента следует представить в виде нескольких качественно различных поведений (состояний), последовательно сменяющих друг друга при возникновении определённых событий. Для этого в окно графического редактора для спецификации поведения формируемого агентного типа включается "параллельная активность" в виде так называемой *диаграммы состояний*. Диаграмма состояний позволяет графически выразить состояния формируемого агентного типа, и ассоциировать с ними события, возникновение которых являются причинами переходов агентов этого типа из текущего состояния в последующие, а с переходами ассоциировать действия, выполняемые в модели при смене агентами состояний. Состояния могут быть иерархическими, т.е. содержать внутренние диаграммы состояний. Графический язык диаграмм состояний AnyLogic сохраняет графический вид, атрибуты и семантику выполнения, определённую в языке моделирования UML.

Элементы диаграммы состояний

Диаграмма состояний рисуется на диаграмме типа агентов с помощью следующих элементов:

-  Начало диаграммы состояний
-  Состояние
-  Переход
-  Указатель начального состояния
-  Конечное состояние

◇ Ветвление

Ⓜ Историческое состояние

Формирование диаграммы осуществляется путём перетаскивания в окно графического редактора и добавления на диаграмму соответствующих элементов из палитры **Агент/Диаграмма состояний** (Рисунок 6).

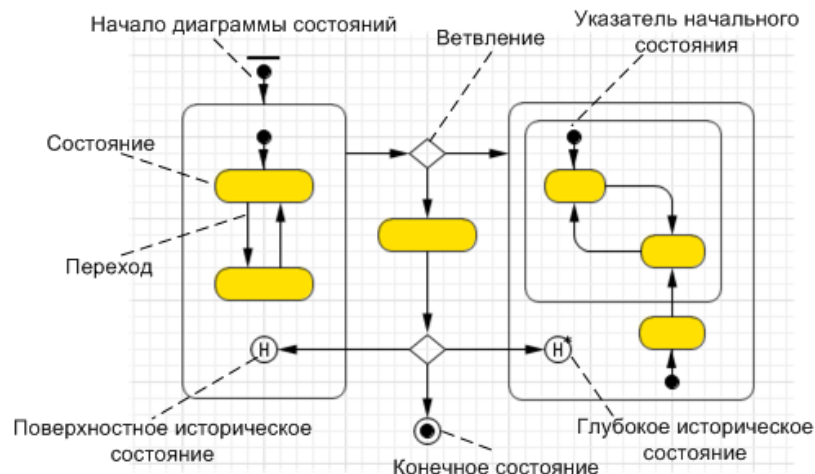


Рисунок 6 Пример диаграммы состояний

Порядок выполнения действий в диаграмме состояний

Очень важно точно знать, в каком именно порядке выполняются действия элементов диаграммы состояний. Для этого предлагается изучить приведённый ниже алгоритм.

При срабатывании перехода выполняются следующие действия (в указанном порядке):

1. Действия при выходе из состояния, начиная с текущего простого состояния, и дальше вверх по иерархии состояний, заканчивая тем сложным состоянием, на уровень иерархии которого и передаётся управление.
2. Действие перехода.
3. Действия при входе в состояние, начиная со сложного состояния, которое получает управление, и дальше, вниз по иерархии состояний, вплоть до простого состояния или псевдосостояния, в которое передаётся управление.
4. Если управление передаётся в псевдосостояние, то выполняется код действия псевдосостояния, а затем управление немедленно передаётся другому состоянию, и описанный выше алгоритм выполняется сначала.



Действия состояний и переходов выполняются за нулевое модельное время. Поэтому они не могут содержать синхронизационных операций и операций задержки и не могут вызывать методы, явно или неявно содержащие такие операции.

Подробное описание элементов диаграммы состояний следует искать в справке AnyLogic. Ниже приведён пример иерархической диаграммы состояний и порядка выполнения в ней действий (Рисунок 7).

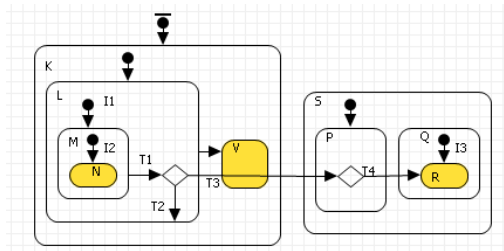


Рисунок 7 Пример порядка выполнения действий

Предположим, что состояние N, представленное на рисунке, является текущим простым состоянием, и срабатывает переход T1. Тогда действия выполняются в следующем порядке:

1. Действие при выходе из состояния N
2. Действие при выходе из состояния M
3. Действие перехода T1
4. Действие состояния ветвления

Затем, в зависимости от дополнительных условий переходов, будет выбран переход T2 или T3. Если будет выбран переход T2, то выполнятся следующие действия:

5. Действие перехода T2
6. Действие указателя начального состояния I1 (действия при входе и выходе из состояния L не выполняются, поскольку управление остается в этом состоянии)
7. Действие при входе в состояние M
8. Действие указателя начального состояния I2
9. Действие при входе в состояние N

Если выбирается переход T3, то выполняются следующие действия:

10. Действие при выходе из состояния L
11. Действие при выходе из состояния K (действия состояния V не выполняются)
12. Действие перехода T3
13. Действие при входе в состояние S
14. Действие при входе в состояние P
15. Действие состояния-ветвления
16. Действие при выходе из состояния P
17. Действие перехода T4 (дополнительное условие перехода должно быть равно true, поскольку это единственный выход из ветвления)
18. Действие при входе в состояние Q
19. Действие указателя начального состояния I3
20. Действие при входе в состояние R

Приложение 1

Полный перечень системных функций представлен в <https://help.anylogic.ru/index.jsp>

Математические функции

Тип возвращаемого значения	Имя	Описание
double	abs(double a)	Возвращает значение по модулю от вещественного (типа double) значения.
int	abs(int a)	Возвращает значение по модулю от целочисленного (типа int) значения.
double	acos(double a)	Возвращает арккосинус значения; возвращаемое значение лежит в интервале от 0.0 до π . <i>Параметры:</i> a - значение, для которого нужно вернуть арккосинус.
double	asin(double a)	Возвращает арксинус значения; возвращаемое значение лежит в интервале от $-\pi/2$ до $\pi/2$. <i>Параметры:</i> a - значение, для которого нужно вернуть арксинус.
double	atan(double a)	Возвращает арктангенс значения; возвращаемое значение лежит в интервале от $-\pi/2$ до $\pi/2$. <i>Параметры:</i> a - значение, для которого нужно вернуть арктангенс.
double	atan2(double y, double x)	Возвращает угол theta, полученный из преобразования прямоугольных координат (x, y) в полярные координаты (r, theta).
double	cbrt(double a)	Возвращает кубический корень из вещественного значения.
double	ceil(double a)	Возвращает наименьшее (ближайшее к минус бесконечности) вещественное (типа double) значение, большее или равное аргументу и равное математическому целому числу.
double	cos(double a)	Возвращает тригонометрический косинус угла. <i>Параметры:</i> a - угол, в радианах
double	cosh(double x)	Возвращает гиперболический синус от вещественного (типа double) значения. <i>Параметры:</i> x - значение, для которого нужно вернуть гиперболический косинус.
double	exp(double a)	Возвращает число Эйлера e, возведенное в степень a (вещественное число).
double	expm1(double x)	Возвращает $e^x - 1$.
double	floor(double a)	Возвращает наибольшее (ближайшее к плюс бесконечности) вещественное (типа double) значение, меньшее или равное аргументу и равное математическому целому числу.
double	gammaLog(double x)	Возвращает натуральный логарифм Гамма функции от x: $\ln(\Gamma(x))$. Гамма функция является расширением факториальной функции, принимающей все положительные значения x. Если n является положительным целым числом, то: $\Gamma(n) = (n - 1)!$. Функция gammaLog может использоваться, например, в системно-динамических моделях, вычисляющих комбинаторные факторы.
int	getExponent(double d)	Возвращает несмещенное значение экспоненты аргумента.
double	hypot(double x, double y)	Возвращает $\sqrt{x^2 + y^2}$ без промежуточного переполнения или потери значимости.
double	limit(double min, double x, double max)	Возвращает x, если значение лежит в интервале [min,max], иначе возвращает ближайшую границу.
double	limitMax(double x, double max)	Возвращает x, если его значение меньше или равно значению аргумента max, иначе возвращает max, то есть значение x ограничено справа значением max.
double	limitMin(double min, double x)	Возвращает x, если его значение больше или равно значению аргумента min, иначе возвращает min, то есть значение x ограничено слева значением min.
double	log(double a)	Возвращает натуральный логарифм (по основанию e) вещественного аргумента.
double	log10(double a)	Возвращает логарифм аргумента по основанию 10.
double	log1p(double x)	Возвращает натуральный логарифм суммы аргумента и 1.
double	max(double a, double b)	Возвращает наибольший из двух вещественных аргументов.
int	max(int a, int b)	Возвращает наибольший из двух целочисленных аргументов.
double	min(double a, double b)	Возвращает наименьший из двух вещественных аргументов.
int	min(int a, int b)	Возвращает наименьший из двух целочисленных аргументов.
double	pow(double a, double b)	Возвращает значение первого аргумента, возведенное в степень второго аргумента.
double	quantum(double value, double quantizer)	Возвращает число, меньшее (по модулю) или равное значению value, которое наиболее близко к value и кратно величине квантователя quantizer. Если quantizer меньше или равен нулю, то возвращает value. <i>Параметры:</i> value - значение; quantizer - квантователь, задающий интервал между возможными значениями, возвращаемыми функцией: quantizer, quantizer*2, ...
double	random()	Возвращает положительное вещественное значение, лежащее в интервале [0.0, 1.0).
double	rint(double a)	Возвращает вещественное значение, ближайшее по значению к аргументу и равное математическому целому числу.
double	scalb(double d, int scaleFactor)	Возвращает $d \times 2^{\text{scaleFactor}}$.
double	signum(double d)	Возвращает знаковую функцию (сигнум) аргумента; 0, если аргумент равен нулю, 1.0, если аргумент больше, чем ноль, -1.0, если аргумент меньше, чем ноль.
double	sin(double a)	Возвращает тригонометрический синус угла. <i>Параметры:</i> a - угол, в радианах
double	sinh(double x)	Возвращает гиперболический синус от вещественного значения. <i>Параметры:</i> x - значение, для которого нужно вернуть гиперболический синус.
double	sqrt(double a)	Возвращает корректно округленный положительный квадратный корень вещественного числа.
double	tan(double a)	Возвращает тригонометрический тангенс угла. <i>Параметры:</i> a - угол, в радианах
double	tanh(double x)	Возвращает гиперболический тангенс вещественного (типа double) значения.
double	toDegrees(double anggrad)	Преобразует измеренный в радианах угол к приблизительно эквивалентному углу, измеренному в градусах. <i>Параметры:</i> anggrad - угол, в радианах
double	toRadians(double angdeg)	Преобразует измеренный в градусах угол к приблизительно эквивалентному углу, измеренному в радианах. <i>Параметры:</i> angdeg - угол, в градусах
double	xidz(double a, double b, double x)	Пытается поделить первый аргумент на второй. Если результат - бесконечность или не является числом, то функция возвращает третий аргумент, если нет - то возвращает результат деления.
double	zidz(double a, double b)	Пытается поделить первый аргумент на второй. Если результат - бесконечность или не является числом, то функция возвращает 0, если нет - то возвращает результат деления.

Приложение 2

Математические константы

Тип	Имя	Значение	Описание
double	infinity	1d / 0d	Константа, хранящая положительное бесконечное число типа double (Double.POSITIVE_INFINITY).
double	-infinity	- 1d / 0d	Отрицательное бесконечное число типа double (Double.NEGATIVE_INFINITY).
double	E	2.718281828459045	Вещественное значение, ближайшее, чем какое бы то ни было к математической константе e , основанию натурального логарифма.
double	PI	3.141592653589793	Вещественное значение, ближайшее, чем какое бы то ни было к математической константе π , выражающей отношение длины окружности к длине её диаметра.

Системные функции

double	time()	Возвращает текущее значение модельного времени (в единицах модельного времени).
boolean	finishSimulation()	Завершает выполнение модели.