МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВПО «БАШКИРСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. М.АКМУЛЛЫ»

Р.Ф.МАЛИКОВ

Практикум по имитационному моделированию сложных систем в среде AnyLogic 6

Уфа 2013

УДК 004.93 (0.75.8)
ББК 32.973.26.018.2 я 73
М 19

Маликов, Р. Ф. Практикум по имитационному моделированию сложных систем в среде AnyLogic 6 [Текст]: учеб. пособие / Р. Ф. Маликов. – Уфа: Изд-во БГПУ, 2013. – 296с.

В пособии рассмотрены основные понятия, виды и инструментальные средства моделирования, основные этапы разработки компьютерных моделей сложных систем. Основное внимание уделено многоподходному инструменту моделирования объектов и процессов реального мира AnyLogic 6 и технологиям построения имитационных моделей по методологиям системной динамики (физические процессы), дискретно-событийного моделирования (транспортные сети, полиграфические процессы) и пешеходной динамики (системы массового обслуживания) в среде AnyLogic.

Предназначено для бакалавров и магистров, обучающихся по направлениям подготовки 051000 - Профессиональное обучение (по отраслям), по направлению 230400 - «Информационные системы и технологии», а также аспирантов, научных работников и инженеров специализирующимся в области математического моделирования сложных систем.

Рецензенты: В.Е. Гвоздев, д-р техн. наук, проф. (УГАТУ)
С.И. Спивак, д-р физ.-мат. наук, проф. (БашГУ)

ISBN 978-5-87978-862-4

© Издательство БГПУ, 2013

© Р.Ф.Маликов, 2013

СОДЕРЖАНИЕ

ВВЕДЕНИЕ..... 5

ГЛАВА 1. МЕТОДОЛОГИЧЕСКИЕ ОСНОВЫ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ СЛОЖНЫХ СИСТЕМ..... 8

1.1. Исходные понятия и определения……. 8

1.2. Разновидности моделирования …… 15

1.3. Классификация систем компьютерного моделирования 29

1.4. Системный анализ и этапы имитационного моделирования сложных систем……. 32

1.5. Проектирование и разработка имитационных моделей сложных объектов ……. 46

1.6. Основные направления и перспективы развития имитационного моделирования…… 52

ГЛАВА 2. СРЕДА ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ ANYLOGIC 6… 58

2.1. Общие сведения о системе имитационного моделирования AnyLogic 6…… 58

2.2. Базовые инструменты для разработки модели в среде AnyLogic 6…….. 64

ГЛАВА 3. СИСТЕМНАЯ ДИНАМИКА..... 72

3.1. Методология системной динамики……. 72

3.2. Моделирование задачи системной динамики «Ассимиляция этносов» ……. 78

ГЛАВА 4. МОДЕЛИРОВАНИЕ ДИНАМИЧЕСКИХ СИСТЕМ 85

4.1. Колебания маятника Фуко……. 86

4.2. Пространственный осциллятор…… 92

4.3. Связанные маятники…… 98

ГЛАВА 5. ДИСКРЕТНО-СОБЫТИЙНОЕ МОДЕЛИРОВАНИЕ ПРЕДПРИЯТИЙ ЗДРАВООХРАНЕНИЯ 105

5.1. Методология дискретно-событийного моделирования… 105

5.2. Дискретно-событийная модель стоматологической клиники …… 106

ГЛАВА 6. ДИСКРЕТНО-СОБЫТИЙНОЕ МОДЕЛИРОВАНИЕ ТРАНСПОРТНЫХ ПОТОКОВ И СЕТЕЙ..... 129

6.1. Модель дорожного перекрестка…… 130

6.2. Модель дорожного движения на трех перекрестках … 140

6.3. Модель дорожно-транспортной развязки с железнодорожным переездом…… 151

6.4. Модель трубовидной транспортной развязки…… 155

ГЛАВА 7. ДИСКРЕТНО-СОБЫТИЙНОЕ МОДЕЛИРОВАНИЕ ПОЛИГРАФИЧЕСКИХ ПРОЦЕССОВ 170

7.1. Имитационная модель подготовки макета издания (допечатная подготовка) ……. 176

7.2. Моделирование печатных процессов…… 211

7.2.1. Анимационная модель печатного процесса ……. 212

7.2.2. Модель печатного цеха при наличии трех офсетных машин…… 217

7.3. Моделирование послепечатных процессов……. 229

7.3.1. Первая анимационная модель послепечатного процесса …… 231

7.3.2. Вторая имитационная модель работы послепечатного цеха ……. 236

ГЛАВА 8. МОДЕЛИРОВАНИЕ ДВИЖЕНИЯ ПЕШЕХОДОВ 245

8.1. Пешеходная динамика покупателей в магазине…… 245

8.2. Пешеходная динамика зрителей в кинотеатре…….. 273

Литература…… 282

Приложение…… 294

ВВЕДЕНИЕ

Благодаря интенсивному развитию информатики и компьютерных технологий стало намного проще решать сложные задачи, требующие больших временных и финансовых затрат. Намного упростить их решение возможно с использованием моделирования.

Одним из наиболее распространенных и удобных способов моделирования сложных систем является имитационное компьютерное моделирование объектов и процессов реального мира.

Невозможно сразу моделировать какой-либо процесс, для этого необходимо специальное обучение способам, приемам и технологиям компьютерного имитационного моделирования.

Специалист, приступая к решению задачи, должен знать основы динамических процессов, подходы и методы решения сложных процессов и систем, в том числе аналитических и имитационных, а также знать конкретные информационные системы моделирования и используемые в них языки программирования. Среди множества сред аналитического моделирования основными являются: Maple, MathCAD, MATLAB+Simulink, и другие.

При обучении моделированию сложных систем могут быть использованы различные среды и методологии разработки аналитических и имитационных моделей сложных систем: MvStudium, MATLAB, Arena, GPSS, Extend, iThink Analyst, Process Model и др. Особое место среди сред разработки компьютерных моделей сложных систем принадлежит много подходной среде моделирования имитационных моделей – AnyLogic [1-2]. Разные средства спецификации и анализа результатов, имеющиеся в AnyLogic, позволяют строить модели (динамические, дискретно-событийные, агентные), имитирующие практически любой реальный процесс (а также строить и многие другие модели), выполнять анализ моделей на компьютере без проведения реальных экспериментов и самостоятельных сложных вычислений. Но для возможности оперировать этой программной средой и получать при моделировании верные результаты, пользователь AnyLogic должен овладеть технологией работы в среде, понять ее функциональные особенности, в этих целях мы представляем практикум по разработке компьютерных моделей сложных систем в среде AnyLogic.

Практикум по имитационному моделированию различных процессов и систем в среде AnyLogic 6 состоит из восьми глав.

В первой главе приведены основные понятия, используемые при моделировании сложных систем, классификация видов и инструментальных

средств моделирования. Здесь же даны основные этапы разработки компьютерных моделей сложных систем, а также примеры технических заданий и подходы к проектированию сложных систем в среде BPWin.

Во второй главе кратко рассказывается об инструменте моделирования объектов и процессов реального мира AnyLogic 6.

Третья глава посвящена методологии и примеру разработки имитационной модели системной динамики.

В четвертой главе приведены лабораторные работы по построению моделей физических процессов и систем. Для построения этих моделей используется методология системной динамики. Представленные лабораторные работы позволят студентам строить имитационные и анимационные модели физических процессов. Студенты, выполняя задания по шагам, в конечном счете, приходят к построению той или иной модели с 3D – анимацией.

В пятой главе рассмотрена методология дискретно-событийного моделирования на примере стоматологической клиники как системы массового обслуживания и построена имитационная модель в среде AnyLogic 6.

Шестая глава посвящена построению дискретно-событийных имитационных моделей транспортных систем и сетей. Здесь представлены лабораторные работы по созданию имитационных и анимационных моделей движения транспорта для разных случаев дорожных сетей.

В седьмой главе приведены примеры разработки имитационных и анимационных моделей полиграфических процессов и систем, приводятся технологии построения имитационных моделей допечатных, печатных и послепечатных процессов, позволяющие исследовать их статистические характеристики.

В восьмой главе приведены примеры построения имитационных моделей по парадигме системной и пешеходной динамики (модели пешеходной динамики покупателей в магазине и зрителей в кинотеатре).

Представленные имитационные модели построены в первом приближении и являются учебными моделями, соответственно требуют уточнения и соответствующей доработки после анализа и обсуждения со специалистами в соответствующей отрасли моделирования. На примере разработки демонстрационных и обучающих моделей мы показали некоторые возможности среды имитационного моделирования Anylogic.

В первой главе студентам предлагаются контрольные вопросы по изложенной теме. Остальные главы являются фактически описанием лаборатор-

ных работ. В конце пособия приведена рекомендуемая литература и приложения.

Данное пособие предназначено для бакалавров, магистров, преподавателей, разработчиков компьютерных моделей и научных работников, занимающихся разработками компьютерных моделей сложных систем.

В основу данного пособия легли разработки учебных имитационных моделей в среде AnyLogic, выполненных автором и его студентами в рамках курсового и дипломного проектирования в Институте профессионального образования и информационных технологий БГПУ им. М.Акмуллы: М.В.Аккужиным, В.Х.Хазиевым, Е.Н.Васильевой, А.А.Аглиуллиным, А.Р.Аглиуллиной, Я.П.Андреевой, А.С.Нургутдиновым, С.В.Борковой, А.Д.Здрюмовой, Д.З.Янбердиным и др.

ГЛАВА 1

МЕТОДОЛОГИЧЕСКИЕ ОСНОВЫ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ

1.1. Исходные понятия и определения

Теория основ математического и компьютерного моделирования предполагает содержательное и формальное определение категорий, определений и понятий с целью построения математических моделей сложных систем.

Основными методологическими категориями теоретических основ моделирования являются понятия[1]: ***объект, класс, отношение (связь), система, элемент, структура.***

Определение понятия ***объект*** имеет различное толкование в зависимости от области рассмотрения[2]. Если мы рассматриваем область имитационного моделирования, то в стратегии объектно-ориентированного подхода объект является первым важным понятием. Объект – это некоторая сущность в виртуальном пространстве, обладающая определенным состоянием и поведением, имеющая заданные значения свойств (атрибутов) и операций над ними.

Следующим важным понятием объектно-ориентированного подхода является ***класс.*** Родственные по определенным характеристикам, поведению объекты объединяются в классы. В зависимости от характеристик одни и те же объекты могут быть в различных классах.

В одном из разделов современной математики «теории категорий» объект используется как термин для обозначения элементов произвольной категории, играющих роль множеств, групп, топологических пространств и т. п. Здесь также вводится понятие класса объектов и проводится изучение свойств отношений между математическими объектами, не зависящих от внутренней структуры объектов.

Понятие ***отношение*** определяет взаимное положение объектов, связи между объектами в виде иерархических, ассоциативных, алгоритмических, табличных и других структур.

Понятие ***система*** является основополагающим в теории математического моделирования. Существует несколько десятков различных определений понятия «система», используемых в зависимости от контекста, области знаний и целей исследования. Изучением систем занимаются научные дис-

[1] http://ustenko.fromru.com/part4.html
[2] http://ru.wikipedia.org

циплины как *системология, кибернетика, системный анализ, теория систем, системная динамика* и другие[3].

Система – это 1) целое, созданное из частей и элементов целенаправленной деятельности и обладающее новыми свойствами, отсутствующими у элементов и частей, его образующих; 2) объективная часть мироздания, включающая схожие и совместимые элементы, образующие особое целое, которое взаимодействует с внешней средой; 3) объективное единство закономерно связанных друг с другом предметов, явлений, сведений, а также знаний о природе, обществе и т. п. Допустимы и многие другие определения. Общим в них является то, что система есть некоторое правильное сочетание наиболее важных, существенных свойств изучаемого объекта. Каждый объект, чтобы его можно было считать системой, должен обладать четырьмя основными свойствами или признаками (целостностью и делимостью, наличием устойчивых связей, организацией и эмерджентностью).

Элемент – это простейшая неделимая часть системы, а ее свойства определяются конкретной задачей. Элемент всегда связан с самой системой. Элемент сложной системы может быть в свою очередь сложной системой в другой задаче.

Подсистема – компонент системы – объединение элементов, но по масштабу меньше, чем система в целом.

Система может включать большой перечень элементов и ее целесообразно разделить на ряд подсистем.

Признаками системы являются множество составляющих ее элементов, единство главной цели для всех элементов, наличие связей между ними, целостность и единство элементов, наличие структуры и иерархичности, относительная самостоятельность и наличие управления этими элементами. Термин «организация» в одном из своих лексических значений означает также «систему», но не любую систему, а в определенной мере упорядоченную, организованную.

Понятие "подсистема" выработано для анализа сложноорганизованных, саморазвивающихся систем, когда между элементами и системой имеются "промежуточные" комплексы, более сложные, чем элементы, но менее сложные, чем сама система. Они объединяют в себе разные части (элементы) системы, в своей совокупности способные к выполнению единой (частной) программы системы. Будучи элементом системы, подсистема в свою очередь оказывается системой по отношению к элементам, ее составляющим. Анало-

[3] http://ru.wikipedia.org/wiki

гично обстоит дело с отношениями между понятиями "система" и "элемент": они переходят друг в друга. Иначе говоря, система и элемент относительны. С этой точки зрения вся материя представляется как бесконечная система систем. "Системами" могут быть системы отношений, детерминаций и т.п.

Наряду с представлением об элементах в представление о любой системе входит и представление о ее структуре.

Структура – это совокупность устойчивых отношений и связей между элементами. Сюда включается общая организация элементов, их пространственное расположение, связи между этапами развития и т.п.

По своей значимости для системы связи элементов (даже устойчивые) неодинаковы: одни малосущественны, другие существенны, закономерны. Структура прежде всего – это закономерные связи элементов. Среди закономерных наиболее значимы интегрирующие связи (или интегрирующие структуры). Они обусловливают интегрированность сторон объекта. В системе производственных отношений, например, имеются связи трех родов: относящиеся к формам собственности, к обмену деятельностью и к распределению. Все они существенны и закономерны. Но интегрирующую роль в этих отношениях играют отношения собственности (иначе формы собственности). Интегрирующая структура является ведущей основой системы.

Существует ряд подходов к выделению систем по сложности и масштабу. Например, для систем управления удобно пользоваться классификацией по числу (количеству) элементов:

малые (10-10^3 элементов);

сложные (10^4- 10^7 элементов);

ультрасложные (10^8 - 10^{30} элементов);

суперсистемы (10^{30} - 10^{200} элементов).

Большая система – это всегда совокупность материальных и энергетических ресурсов, средств получения, передачи и обработки информации, людей, которые принимают решение на разных уровнях иерархии. В настоящее время для понятий «сложная система» и «большая система» используют такие определения:

- *сложная система* – упорядоченное множество структурно взаимосвязанных и функционально взаимодействующих *разнотипных систем,* которые объединены структурно в целостный объект функционально разнородными взаимосвязями для достижения заданных целей в определенных условиях;
- *большая система* объединяет *разнотипные сложные системы.*

Тогда определение системы можно записать так «***система – это упорядоченное множество структурно взаимосвязанных и функционально взаимодействующих однотипных элементов любой природы, объединенных в целостный объект, состав и границы которого определяются целями системного исследования».***

Характерные особенности больших систем:

- значительное количество элементов;
- взаимосвязь и взаимодействие между элементами;
- иерархичность структуры управления;
- наличие человека в контуре управления и необходимость принятия решений в условиях неопределенности.

Описание динамики системы или ее поведения составляет основу любой имитационной модели. В качестве исходных данных для решения этой задачи используются результаты, полученные на этапе разработки концептуальной модели системы. К ним относятся:

– определение принадлежности моделируемой системы одному из известных классов;

– описание рабочей нагрузки системы;

– выбор уровня детализации представления системы в модели и ее декомпозиция.

Все последующие действия исследователя по созданию модели могут быть отнесены к этапу ее формализации, который в общем случае предполагает:

– выбор метода отображения динамики системы (на основе событий, процессов или транзактов);

– формальное (математическое) описание случайных факторов, подлежащих учету в модели;

– выбор механизма изменения и масштаба модельного времени.

Рассмотрим устоявшиеся понятия в имитационном моделировании: *«**процесс**», «**работа**», «**событие**», «**транзакт**».*

Работа *(активность)* – это единичное действие системы по обработке (преобразованию) входных данных. В зависимости от природы моделируемой системы под входными данными могут пониматься информационные данные или какие-либо материальные ресурсы.

Под ***процессом*** понимают логически связанный набор работ. Некоторые процессы могут рассматриваться как работы в процессе более высокого

уровня. Любой процесс характеризуется совокупностью статических и динамических характеристик.

К статическим характеристикам процесса относятся:

- длительность;
- результат;
- потребляемые ресурсы;
- условия запуска (активизации);
- условия остановки (прерывания).

Статические характеристики процесса не изменяются в ходе его реализации, однако при необходимости любая из них может быть представлена в модели как случайная величина, распределенная по заданному закону.

Динамической характеристикой процесса является его состояние (активен или находится в состоянии ожидания).

Моделирование в терминах процессов проводится в тех случаях, если система оценивается по каким-либо временным показателям, либо с точки зрения потребляемых ресурсов.

Например, при оценке производительности вычислительной сети обработка заданий может быть представлена в модели как совокупность соответствующих процессов, использующих ресурсы сети (оперативную память, пространство на жестких дисках, процессорное время, принтеры и т.д.).

Если модель строится с целью изучения причинно-следственных связей, присущих системе, динамику системы целесообразно описывать в терминах событий.

Событие представляет собой мгновенное изменение некоторого элемента системы или состояния системы в целом. Событие характеризуется:

- условиями (или законом) возникновения;
- типом, который определяет порядок обработки (дисциплину обслуживания) данного события;
- нулевой длительностью.

События подразделяют на две категории:

- события следования, которые управляют инициализацией процессов (или отдельных работ внутри процесса);
- события изменения состояний (элементов системы или системы в целом).

Механизм событий используется в качестве основы построения моделей, предназначенных для исследования причинно-следственных связей в

системах при отсутствии временных ограничений. К таким задачам можно отнести, например, некоторые задачи по оценке надежности.

Еще один способ имитационного моделирования систем основан на использовании понятия транзакта или сущности.

Транзакт или сущность — это некоторое сообщение (заявка на обслуживание), которое поступает извне на вход системы и подлежит обработке.

В некоторых случаях, например, при моделировании автоматизированных систем управления удобно проследить функционирование системы относительно алгоритма обработки транзакта(сущности) . В рамках одной имитационной модели могут рассматриваться транзакты(сущности) нескольких типов. Каждый транзакт(сущность) характеризуется соответствующим алгоритмом обработки и необходимыми для его реализации ресурсами системы. Прохождение транзакта (сущности) по системе можно в некоторых случаях рассматривать как последовательную активизацию процессов, реализующих его обработку («обслуживание заявки»).

Чтобы построить качественную компьютерную модель сложной системы необходимо уметь:

- определенным способом представить в модели динамику (движение) системы. Это может быть описано посредством событий, работ, процессов, транзактов;
- определить способ изменения модельного времени. Здесь выделяют моделирование с постоянным шагом и моделирование по особым состояниям.

В большинстве случаев конечной целью моделирования является оптимизация каких-либо параметров системы.

Виды имитационного эксперимента:

- исследование относительного влияния различных факторов на значения выходных характеристик системы;
- нахождение аналитической зависимости между интересующими исследователя выходными характеристиками и факторами;
- отыскание оптимальных значений параметров системы (так называемый «экстремальный эксперимент»);
- сравнение альтернатив для принятия решений;
- оптимизация системы для оценки и выработки оптимальной стратегии;
- анализ ситуаций и обучение в различных отраслях через виртуальные имитационные модели игр;

– визуализация и анимация работы разрабатываемого объекта.

Вид эксперимента влияет не только на выбор схемы его формализации, но также на построение плана эксперимента и выбор метода обработки его результатов.

С точки зрения организации взаимодействия исследователя с моделью (по способу взаимодействия с пользователем), в ходе эксперимента имитационные модели делятся на автоматические и диалоговые.

Автоматическими называются имитационные модели[4], взаимодействие пользователя с которыми сводится только к вводу исходной информации и управлению началом и окончанием работы моделей.

Диалоговыми называются имитационные модели, позволяющие исследователю активно управлять ходом моделирования; приостанавливать сеанс моделирования, изменять значения параметров модели, корректировать перечень регистрируемых данных и т. д.

Компьютерная модель (англ. computer model), или численная модель (англ. computational model) — это (1) компьютерная программа, работающая на отдельном компьютере, суперкомпьютере или множестве взаимодействующих компьютеров (вычислительных узлов), реализующая абстрактную модель некоторой системы; это (2) модель, выполненная с помощью компьютерных информационных, схематичных, электронных устройств и технологий и сетей; это (3) созданный за счет ресурсов компьютера виртуальный образ, качественно и количественно отражающий внутренние свойства и связи моделируемого объекта, иногда передающий и его внешние характеристики; это (4) модель, воспроизводящая моделируемый объект программными средствами на компьютере.

Разработке компьютерной модели предшествуют мысленные, вербальные, структурные, математические и алгоритмические модели.

Компьютерные модели подразделяются на аналитические и имитационные. Компьютерные модели различаются по видам применения: обучающие, научно-исследовательские, научно-технические для исследования процессов и явлений, реальных объектов и промышленные, встроенные в производственный процесс или адекватно моделирующие производственные процессы на компьютерах. Имитационные модели не только отражают реальность с той или иной степенью точности, но и имитируют ее. Эксперимент с моделью либо многократно повторяется при разных исходных данных, чтобы изучить и оценить последствия каких-либо действий на реальную обстанов-

[4] http://info-tehnologii.ru/IMIT_MOD/index.html

ку, либо проводится одновременно со многими другими похожими объектами, но поставленными в разные условия.

Имитационное моделирование при изучении сложных систем является практически основным доступным методом получения информации о поведении системы в условиях неопределенности.

Компьютерные модели сложных систем подразделяются условно на следующие виды:

- структурно-функциональные, которые представляют собой условный образ объекта (технологические диаграммы, сетевые графики, структурные схемы, ГИС, табличный способ, анимационные и мультипликационные), описанный с помощью программных и компьютерных технологий;
- имитационные, представляющие собой программу или комплекс программ, позволяющий воспроизводить процессы функционирования объекта в разных условиях.
- Комбинированные, с возможностями наблюдения и исследования объекта на динамических условных образах модели и имитационных моделях объекта.

Существует множество программных комплексов, которые позволяют проводить построение и исследование моделей (моделирование). Каждая программная среда имеет свой инструментарий и позволяет работать с определенными видами информационных моделей. Поэтому перед исследователем возникает нелегкий вопрос выбора наиболее удобной и эффективной среды для решения поставленной задачи. Надо сказать, что одну и ту же задачу можно решить, используя различные среды программирования и моделирования.

От выбора программной среды зависит алгоритм построения компьютерной модели, а также форма его представления. Например, это может быть блок-схема. Руководствуясь блок-схемой, задачу можно решить в разных средах. В среде программирования – это программа, записанная на алгоритмическом языке. В прикладных средах – это последовательность технологических приемов, приводящая к решению задачи.

1.2. Разновидности моделирования

Моделирование в широком смысле – это изучение объектов познания с помощью их моделей; построение и изучение моделей реально существую-

щих предметов, процессов или явлений с целью получения объяснений этих явлений, а также для предсказания явлений, интересующих исследователя.

Исследователь имеет дело с моделью, а не его оригиналом. Классификацию видов моделирования можно проводить по разным признакам: по характеру моделей, по характеру моделируемых объектов, по сферам приложения моделирования. Мы выделим следующие виды моделирования [19, 26, 28, 87, 119, 124]:

- информационное (концептуальное) моделирование – процесс описания информации об объекте, с помощью формализованных, неформализованных языков, образно-иллюстративных материалов и фиксированные в реальном материале эти представления и факты;
- эстетическое моделирование – процесс описания информации и объектов и явлений через ощущения и восприятия человека посредством живописи, декоративно-прикладного искусства и музыки;
- физическое моделирование – процесс разработки, конструирование натурных, физических, аналоговых или масштабных моделей объектов и исследование свойств и картины поведения объекта и реальных явлений на этих моделях;
- математическое (аналитическое и имитационное) моделирование;
- компьютерное моделирование.

В узком содержательном смысле, под моделированием мы будем понимать ряд процессов:

- процесс описания или формализации объекта-оригинала, преследующей целью – создание аналога (модели), адекватного объекту;
- процесс конструирования или проектирования объекта или его модели;
- изменение существующей модели в целях создания другой модели, более адекватной объекту-оригиналу;
- процесс проведения эксперимента на модели в целях: прогнозирования и выяснения картины возможного поведения объекта-оригинала; для изучения различных характеристик и свойств объекта.

Рассмотрим более подробно виды математического и компьютерного моделирования. Математическое моделирование подразделяется на аналитическое и имитационное.

Аналитическое моделирование. Под аналитическим моделированием мы будем понимать процесс формализации реального объекта и нахождение его решения в аналитических функциях.

Модель, сформулированная на языке математики, физики, химии или другой науки с помощью системы специализированных символов с точными правилами сочетаемости, называется аналитической моделью, чаще всего они представляются в виде формул, неравенств, линейных и нелинейных уравнений, в том числе дифференциальных и интегро-дифференциальных уравнений и их комбинаций.

Специалисты, занимающиеся математическим моделированием, исследование объекта или явления обычно начинают с поиска возможных аналитических решений упрощенной математической модели, используя различные приближения, т.е. на самом деле решают упрощенную задачу (модель).

Полученные аналитические решения для упрощенной модели удовлетворительно характеризует суть явлений. Аналитические решения позволяют понять и наглядно представить основные закономерности, особенно при изучении нового явления или процесса.

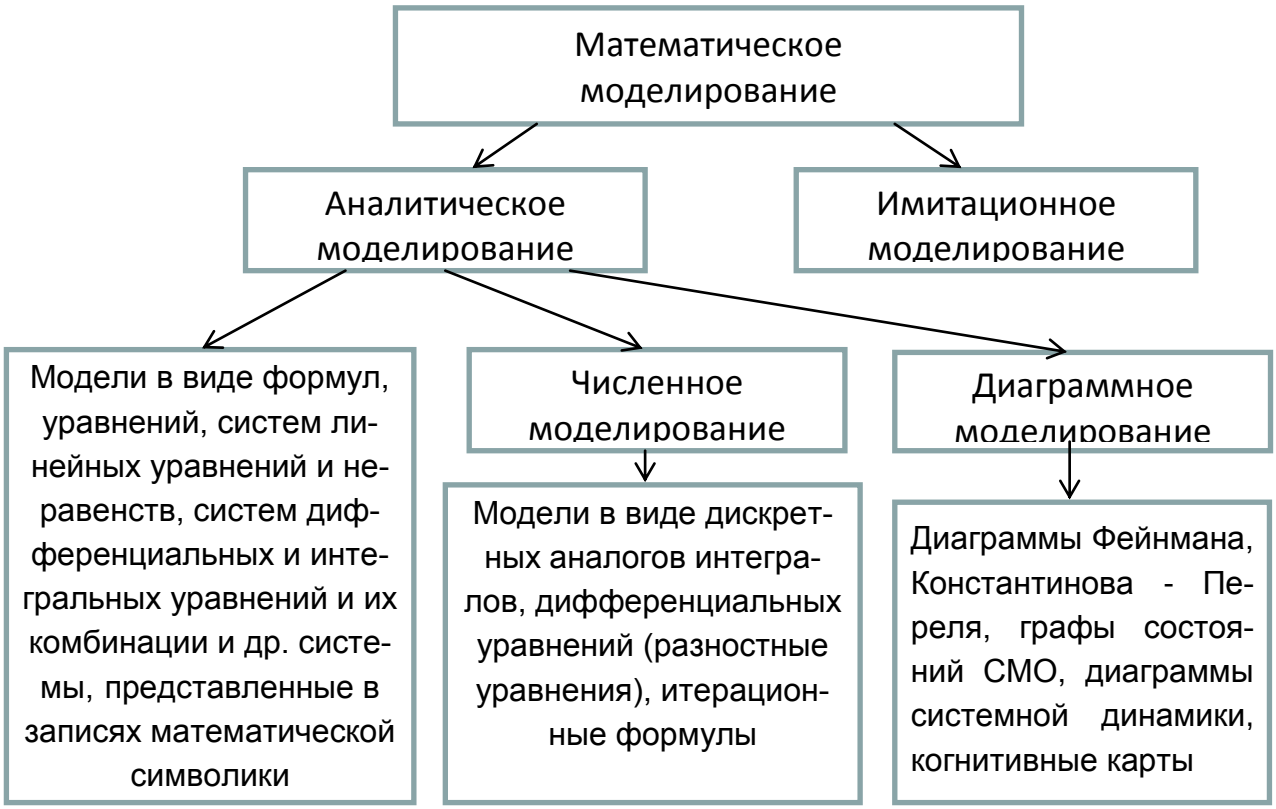


Рис. 1.1 Виды математического моделирования

Однако возможности нахождения аналитического решения при исследовании непростых моделей ограничены, поэтому решения часто строятся в виде алгебраических итерационных формул. Итерационные модели, представленные в виде алгебраических уравнений, можно решать приближенно, используя численные методы.

Процедуру построения математической модели какого-либо реального явления или процесса и нахождение численного решения с помощью итерационных формул часто называют численным моделированием.

Теорию аналитического моделирования реальных процессов и технологии разработки компьютерных моделей можно изучать по книгам [9, 14, 19, 26, 28, 74, 87, 92, 119, 120, 135].

Компьютерное моделирование – это применение компьютерных технологий решения математических моделей на электронно-вычислительных машинах.

Появление компьютеров позволило ускорить процесс нахождения решения математических моделей. Аналитические, численные и другие методы реализованы на ЭВМ. Были разработаны множество компьютерных технологий моделирования. Это технологии моделирования на языках программирования, в системах компьютерной математики и схемотехнического моделирования. С помощью этих технологий создаются компьютерные вычислительные установки. Компьютерная имитация позволяет исследовать модель, как в определенные моменты времени, так и в течение продолжительных периодов времени. Для нахождения решений (характеристик) при моделировании требуется его многократное воспроизведение с последующей обработкой, чаще всего с помощью компьютерных средств визуализации. В результате использования этих технологий мы получаем «компьютерное решение» рассматриваемой задачи.

Имитационное моделирование. В связи со стремительным развитием информационных и компьютерных технологий возможности моделирования реальных объектов расширились. Появились новые методы и технологии, позволяющие моделировать сложные объекты и процессы в промышленности, здравоохранении, в экономических и социальных системах, в науке и других сферах. Появление новых систем (пакетов) моделирования привело к созданию нового типа компьютерных моделей – «имитационных моделей».

Под имитационным моделированием понимается *«разработка модели системы в виде программы для компьютера и проведение экспериментов с программой, вместо проведения экспериментов с реальной системой или объектом».*

Имитационное моделирование применяется, когда невозможно построить аналитическую модель системы, учитывающую причинные связи, последствие, нелинейности, стохастические переменные, когда необходимо имитировать поведение системы во времени, рассматривая различные

возможные сценарии ее развития при изменении внешних и внутренних условий.

Таким образом, имитационное моделирование – это высокоуровневая информационная технология с применением компьютеров и чаще всего используется при моделировании сложных систем и написано достаточное количество работ по разным подходам и точки зрения к имитационному моделированию [4, 6, 20, 22-23, 46, 57-72, 79, 84, 88, 103-106, 111- 115, 123, 129-131, 134].

Появление новых современных программных продуктов существенно снижает требования к разработчику модели, и открывают для специалистов широкого профиля, не обладающих навыками программирования, возможность разработки моделей, в том числе и для достаточно сложных систем. В то же время увеличивает требования к постановщику задач. Вышесказанное предопределяет чрезвычайно широкие возможности по применению методов имитационного моделирования при изучении социальных явлений, образовательной деятельности, при обучении управленческих кадров и т.д. специалистами в этих отраслях. Однако существуют другая опасность в разработке и интерпретации результатов имитационных моделей сложных систем. Речь идет о разработке плохих и уродливых моделей [20].

Модель представляет собой упрощенное отображение реальности – это менее детальное, менее сложное, менее подробное воспроизведение реально существующего объекта, системы или феномена, процесса. При этом, когда мы говорим о моделировании социальных феноменов, модель представляет не просто упрощение реальности, а отображение реальности через призму определенного теоретического подхода, или мнения эксперта.

Имитационное моделирование условно может быть представлено различными разновидностями или направлениями, соответственно имеющими свои методологии (рис. 1.2). Рассмотрим эти направления:

Статистическое (численное) моделирование является разновидностью имитационного моделирования. Изначально оно появилось в теории случайных процессов и математической статистике как способ вычисления статистических характеристик случайных процессов путем многократного воспроизведения течения процесса с помощью модели этого процесса [22, 54, 98, 123]. Этот подход к исследованию реального процесса был назван методом статистических испытаний (методом Монте-Карло). Модели здесь строятся для явлений и систем объектов, входы и (или) функциональные соотношения между различными компонентами которой содержат элементы

случайности или полностью случайных процессов, подчиняющиеся вероятностным законам.

Рис.1.2. Разновидности (направления) имитационного моделирования

Реализация решения вероятностной модели реального объекта осуществляется на ЭВМ. Машинная имитация позволяет исследовать модель, как в определенные моменты времени, так и в течение продолжительных периодов времени. Для нахождения устойчивых решений (характеристик) при численном статистическом моделировании требуется его многократное воспроизведение с последующей статистической обработкой. Здесь проводится имитация воздействия многочисленных случайных факторов на различные элементы модели. Каждое воздействие на процесс в модели представляется в виде «розыгрыша» случайного явления с помощью процедуры, дающей случайный результат. Множество таких реализаций в ходе одного варианта имитации дает одну реализацию (историю) процесса. Затем вычисляются средние статистические характеристики по многим историям [23, 54, 87, 98].

Статистическое моделирование в зависимости от области применения подразделяется на несколько направлений (рис.1.3).

Вероятностное моделирование (методы Монте-Карло) это направление развивается как способ решения математических задач – вычисление интегралов, решение систем линейных уравнений, решение дифференциальных уравнений.

Вероятностно-имитационное моделирование – применение теории вероятностей и методов Монте-Карло для построения имитационных моделей в

молекулярной, статистической, квантовой, нейтронной физике, геофизике, газовой динамике, химической кинетике, в передаче и защите информации, в моделях массового обслуживания, финансовой математики, математической биологии и др.

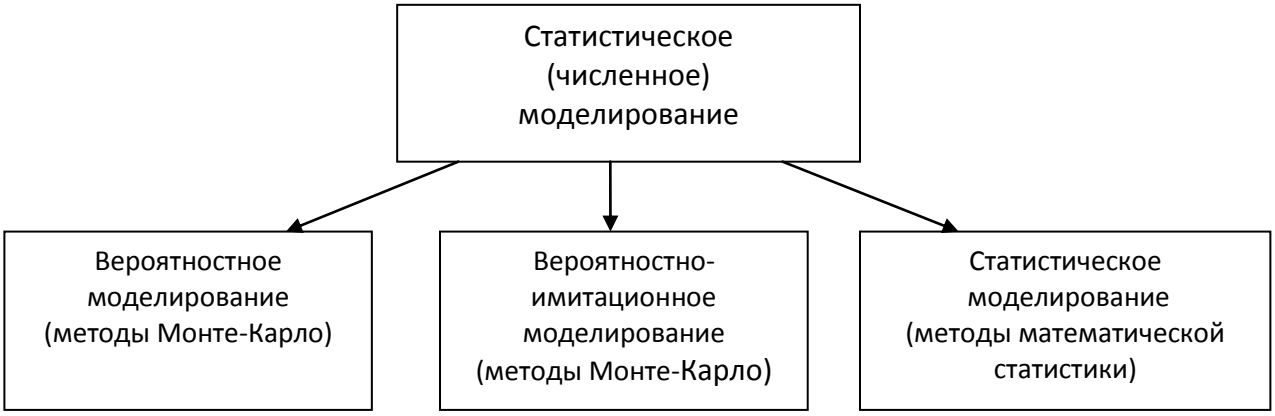


Рис.1.3. Классификация видов статистического
численного моделирования по типу решаемых задач

Статистическое моделирование – применение математической статистики для статистического оценивания и прогнозирования, корреляционно-регрессионного и многомерного статистического анализа, оптимизации систем, определения экстремума функций большого числа переменных и др. в различных отраслях производства и науки.

В экономических и социальных науках чаще всего используется статистическое моделирование. Здесь при разработке модели решаются задачи формализации экспертного знания и различных теоретических концепций, при разработке модели максимально исчерпывающе описываются положения теоретической концепции или результаты экспертного анализа (например, в форме мозгового штурма, социологического экспертного опроса и проч.). Сама по себе подобная формализация является важным результатом, и моделирование в этом свете может рассматриваться в качестве своеобразного формального языка и выступать в качестве определенного аналога математики для естественных наук. Результатом применения статистического моделирования является возможность проведения прогноза. Прогнозирование можно считать одним из наиболее ценных приложений имитационного моделирования.

Динамические системы. Под *динамической системой* будем понимать любой объект, процесс или явление, для которого однозначно определено понятие состояния как совокупности некоторых величин и задан закон, ко-

торый описывает изменение начального состояния с течением времени, двигающуюся в пространстве и изменяющуюся во времени. Динамическими объектами могут быть механические, производственные, физические, химические, биологические объекты, вычислительные процессы и др.

Динамические системы описываются различными способами: дифференциальными уравнениями, дискретными отображениями, марковскими цепями, графическими образами и др. Они классифицируются в зависимости от вида оператора отображения и структуры фазового пространства. Различают линейные и нелинейные, непрерывные и дискретные операторы в соответствии определяются системы линейные и нелинейные, системы с *дискретным* временем и системы с *непрерывным* временем.

В основе методологии *моделирования динамических систем* и построения объектно-ориентированных моделей в технических системах лежит агрегативный подход, который был заложен в 1960-70-х годах гениальным советским ученым Н.П.Бусленко, здесь сложная система представлялась в виде агрегата (черного ящика), имеющего множество входных и выходных сигналов и воздействующих управляющих сигналов. Математически агрегат задается совокупностью множеств Т, Х, Г, У, Z и случайными операторами Н и G , где Т – множество моментов времени, Х, Г, У – множества входных, управляющих выходных сигналов агрегата, Н и G операторы переходов и выхода. Этот подход широко используется при исследовании сложных индивидуальных управленческих систем, к которым относятся АСУ[4, 9, 50-52, 75, 81, 99, 132, 145]. Агрегативные системы позволяют описать широкий круг объектов исследования с отображением системного характера этих сложных объектов, с возможностью расчленения сложной системы на конечное число подсистем, с сохранением связей между ними и взаимодействия частей. В теории автоматического управления основным объектом изучения являются системы управления сложными динамическими (техническими) объектами и ее элементами. Математические модели систем автоматического управления и ее элементов представляются в виде уравнений динамики (движения), которые записываются либо в форме дифференциальных, интегральных и разностных уравнений, либо в виде уравнений «вход-выход» (в общем случае матричных уравнений) в пространстве состояний, благодаря которому они нашли широкое применение в инженерной практике. Описание динамических систем и элементов в пространстве состояний позволяет легко перейти к уравнениям для моделирования на ЭВМ, а также провести моделирование систем автоматического управления в виде струк-

турных схем с помощью аппарата передаточных функций и динамических звеньев [50, 52, 99, 145]. Для моделирования динамических систем используются так называемые среды схемотехнического моделирования: VISSIM, SIMULINK+MATLAB, PowerSim, Multisim, LabView, Easy5, MvStudium и др.

Дискретно-событийное моделирование обязано своим рождением Дж. Гордону, который в начале 1960-х спроектировал и реализовал на IBM систему дискретно-событийного программирования GPSS (Global Purpose Simulation System). Основной объект в этой системе — пассивный транзакт (заявка на обслуживание), который может определенным образом представлять собой работников, клиентов, покупателей, детали, сырье, документы, сигналы и т. п. «Перемещаясь» по модели, транзакты становятся в очереди к одноканальным и многоканальным устройствам, захватывают и освобождают эти устройства, расщепляются, уничтожаются и т. д. Таким образом, дискретно-событийную модель можно рассматривать как глобальную схему обслуживания заявок. Аналитические результаты для большого количества частных случаев таких моделей рассматриваются в теории массового обслуживания.

Этот подход используется для описания функционирования системы (процесса) из одного состояния в другое дискретным образом в виде события. Подход к построению имитационных моделей, предлагающий аппроксимировать реальные процессы такими событиями, и называется "дискретно-событийным" моделированием (discrete event modeling) [6, 17, 32, 37, 57-58, 79, 117, 129, 134]. Этот вид моделирования чаще всего используется для производственных процессов, где динамика системы может быть представлена как последовательность операций («процессное моделирование»). Данный подход широко применяется в теории массового обслуживания, который изучает широкий класс случайных процессов в системах распространения информации, информационно-коммуникативных системах (компьютеры, интернет, связь и т.д.) и в различных отраслях массового обслуживания (железнодорожный, автомобильный транспорт, аэропорты, поликлиники, санаторные и лечебные учреждения, любые торговые предприятия, сферы обслуживания и др.) [15, 53, 58, 63, 88, 94, 117].

Системная динамика – парадигма моделирования, где для исследуемой системы строятся графические диаграммы причинных связей и глобальных влияний одних параметров на другие во времени, а затем созданная на основе этих диаграмм модель имитируется на компьютере [121]. Метод ос-

нован Дж. Форрестером в 1950-х годах и используется для анализа сложных систем с нелинейными обратными связями [138-140]. По сути, такой вид моделирования более всех других парадигм помогает понять суть происходящего выявления причинно-следственных связей между объектами и явлениями. С помощью системной динамики строят модели общества, мировой динамики, бизнес-процессов, развития города, модели производства, динамики популяции, экологии и развития эпидемии и другие [2, 84, 100-101, 108, 121]. Системная динамика – это подход имитационного моделирования, своими методами и инструментами позволяющий понять структуру и динамику сложных систем. Также системная динамика – это метод моделирования, использующийся для создания точных компьютерных моделей сложных систем для дальнейшего использования с целью проектирования более эффективной организации и политики взаимоотношений с данной системой. Системная динамика главным образом используется в долгосрочных, стратегических моделях и принимает высокий уровень абстракции. Люди, продукты, события и другие дискретные элементы представлены в моделях Системной Динамики не как отдельные элементы, а как система в целом.

Агентное моделирование (agent-based model (ABM)) – разновидность имитационного моделирования, современный метод, позволяющий исследовать работу децентрализованных агентов и то, как такое поведение определяет поведение всей системы в целом [60, 62, 66, 91].

В отличие от системной динамики аналитик определяет поведение агентов на индивидуальном уровне, а глобальное поведение возникает как результат деятельности множества агентов (моделирование «снизу вверх»).

Ингредиенты агентного моделирования.

Агент – это некая сущность, обладающая активностью, автономным поведением, может принимать решения в соответствии с некоторым набором правил, взаимодействовать с окружением, а также самостоятельно изменяться. Агентами могут быть: люди (потребители, жители, работники пациенты, доктора, клиенты, солдаты и др.); транспорт, оборудование (автомобили, краны, самолёты, вагоны, станки, …); нематериальные вещи (проекты, продукты, инновации, идеи, инвестиции…,); организации (компании, политические партии, страны, …).

Среда – некоторое пространство, в котором находятся агенты, характеризуемая своими состояниями и факторами, агенты находятся в

определенном месте этого пространства, с возможностью ориентирования и передвижения в данном пространстве;

Правила взаимодействия – законы взаимодействия агентов в окружающей среде, с процедурами принятия решения и выбора стратегии при очередном шаге взаимодействия.

Задача имитационного моделирования при агентном подходе заключается в определении характеристик состояния агентов и среды, изучения поведения агентов при различных ситуациях взаимодействия и изменяющихся состояниях среды.

Через изучение поведения множества агентов в некотором пространстве согласно некоторым правилам взаимодействия, прогнозирование поведения системы в целом.

Цель создания агентных моделей — получить представление об этих глобальных правилах, общем поведении системы, исходя из предположений об индивидуальном, частном поведении ее отдельных активных объектов и взаимодействии этих объектов в системе.

Агентный подход позволяет исследовать задачи коллективного взаимодействия, эффективно решать задачи прогнозирования. Агентные системы позволяют исследовать процессы самоорганизации, дают возможность естественного описания сложных систем, обладают высокой гибкостью [1].

Методология когнитивного моделирования, предназначенная для анализа и принятия решений в *плохо определенных ситуациях*, была предложена американским исследователем Р. Аксельродом[5]. Изначально когнитивный анализ сформировался в рамках социальной психологии, а именно – когнитивизма, занимающегося изучением процессов восприятия и познания. Применения разработок социальной психологии в теории управления привело к формированию особой отрасли знаний – когнитологии, концентрирующейся на исследовании проблем управления и принятия решений.

Под когнитивными технологиями понимается широкий спектр технологий рационализации и формализации интеллектуальных систем создания и функционирования знаний, экспертизы, коммуникации и принятия решения [3, 18, 25, 86]. Когнитивные информационные технологии представляет собой совокупность методов, приемов, действий, процессов, осуществляемых в

[5] Structure of Decision. The cognitive Maps of Political Elites / Ed. R. Axelrod. N.Y.: Princeton, 1976.

определенной последовательности, инструментальных средств (ПК), позволяющих преобразовать входную информацию в варианты управленческого решения.

В триаде «теория - натурный эксперимент – машинный имитационный эксперимент», последний блок является быстроразвивающимся научным методом, который применяется практически во всех высокотехнологических отраслях для моделирования сверхсложных систем. Когнитивная наука в широком смысле слова — совокупность наук о приобретении, хранении, преобразовании и использовании знания.

В настоящее время когнитивный подход является направлением исследования больших систем, к которым относится социально-экономические, политические, экологические системы. Методология когнитивного моделирования развивается в направлении совершенствования аппарата анализа, моделирования и поиска решений в слабоформализуемых и плохо структурированных ситуациях при отсутствии или неполной информации о процессах, происходящих в таких ситуациях и условиях быстрых перемен [3, 25, 40-42].

Ситуационное моделирование. Ситуационное моделирование(situational simulation), ситуационное управление (management situations) – направление исследований и принятия решений, развиваемое с 60-х гг. прошлого столетия. В России эти исследования связаны с авторами работ [3, 110, 116, 126, 130, 141].

Ситуационное управление – это метод управления сложными техническими и организационными системами, основанный на идеях теории искусственного интеллекта: представление знаний об объекте управления и способах управления им на уровне логико-лингвистических моделей, использование обучения и обобщения в качестве основных процедур при построении процедур управления по текущим ситуациям, использование дедуктивных систем для построения многошаговых решений [18, 110, 130].

Ситуационное моделирование как метод исследования ситуаций, включающий в себя построение модели реальной ситуации и проведение с ней различного рода мысленных экспериментов: прогнозирования направлений ее развития и (или) “проигрывание” на ней предполагаемых решений по управлению ситуацией с целью выбора оптимального. В целях исследования ситуаций различного вида разрабатывают компьютерные аналитические и имитационные модели для прикладных применений, например, различные компьютерные игровые модели для общества, бизнеса, производства под-

разделяются на деловые, экономические, военные, образовательные. С помощью таких моделей можно разрешать конфликтные ситуации, оказывать психологическую помощь, проигрывать поведение объекта и системы в различных ситуациях.

Особо ценное практическое значение приобретает создание многовариантных ситуационных моделей как перспективной, так и ретроспективной направленности.

Многовариантное ситуационное моделирование предоставляет его субъекту “поисковое поле”, на котором происходит многовариантный выбор: например, избрание того или иного тактического приема, использование той или иной типовой версии. Многовариантное моделирование постоянно вызывает потребность выбирать, сравнивать, искать альтернативы, находить лучшие решения и средства их реализации. Основными достоинствами многовариантного моделирования являются его многофакторность и многофункциональность, гибкость и продуктивность. Разработка такого рода компьютерной имитационной модели для принятия решения в многовариантном ситуационном поле является одним из важных направлений имитационных исследований.

Логическая последовательность процедуры ситуационного моделирования может быть сведена к следующим его этапам:

а) постановка проблемы (определение задач моделирования);

б) построение модели ситуации (моделирование ситуации), т.е. заполнение структурных блоков конкретным содержанием;

в) абстрагирование от несущественных для исследования обстоятельств;

г) диагностика ситуации;

д) учет динамических факторов;

е) определение ряда возможных альтернативных решений по управлению ситуацией;

ж) “проигрывание” решений на модели и выбор оптимального.

Заметим, что далеко не всегда процесс ситуационного моделирования включает в себя вышеназванные этапы.

Для формализации и описания ситуаций моделей используют различные подходы:

- логика предикатов;

- дискретные ситуационные сети, представляющую семантическую сеть, в которой ситуация описывается ориентированным графом;

- универсальный семантический код, в котором используется конструкция SAO, где S – субъект совершает A – действие над объектом O ;

- RX – коды, которые представляют собой язык бинарных отношений.

Одним из направлений ситуационного моделирования является разработка компьютерных и игровых имитационных моделей своеобразных тренажеров, позволяющих проводить обучения по производственным, транспортным, психологическим, педагогическим, управленческим и др. ситуациям. В основе данных тренажеров лежит имитационная модель, которая позволяет обучающимся применять определенный набор инструментов воздействия и управления и воспроизводит реакцию социальной системы на соответствующее управляющее воздействие. Тренажеры такого рода (виртуальные игровые имитационные модели) уже получили широкое распространение в зарубежной практике.

Результатом развития направления ситуационного моделирования является ситуационное управление как научного подхода к анализу и решению задач ситуационного отображения информации. Разработаны системы ситуационного отображения информации, которые позволяют решать задачи управления динамическими объектами (центры управления космическими станциями, цифровые системы анализа и изображений, предназначенные для интерпретации различных политических и военных ситуаций, системы опознавания различных объектов и др.). Эти системы позволяют строить изображения ситуаций, возникающих в предметной области, на основе которых можно принимать управляющие решения в рамках поставленной задачи.

Важность развития этого направления показывает и тот факт, что в каждом регионе созданы ситуационные центры, для обобщения и моделирования различных видов ситуаций.

Ситуационный центр теоретически представляется в виде совокупности интеллектуально организованных рабочих мест с автоматизированными операциями:

- поступления и пополнения информации;
- процедурами построения имитационных и тренажерных моделей;
- анализа ситуаций;
- прогона моделей,
- графического представления проигранных сценариев
- экспертных систем прогнозирования.

В соответствии с поставленными задачами ситуационного центра возникают направления развития ситуационного моделирования. В зависимости

от класса и масштаба задач ситуационные центры подразделяются на стратегические и оперативные.

Стратегические ситуационные центры решают сложные, масштабные, ответственные задачи, направленные на структурную и функциональную перестройку и ориентированы на объекты класса: регион, отрасль, крупное предприятие (холдинг), ведомство и т.п. Руководители, имеющие стратегические ситуационные центры владеют большей информацией и возможностями ситуационного прогноза.

Оперативные ситуационные центры, решают задачи свертки оперативной информации в ситуационную модель, дающее лицу принимающему решение (ЛПР) возможность маневрирования в пределах своей компетентности (области интересов) или бизнеса. Эти центры настроены на объекты класса: предприятие (компания), проект, крупная акция, процесс и др. Здесь строятся имитационные модели компании или отрасли, позволяющие осуществлять оперативный контроль и анализ состояния компании, существенно улучшить процесс управления компанией. Для ситуационного моделирования применимы методологии когнитивного моделирования.

1.3. Классификация систем компьютерного моделирования

Имитационная модель представляется в виде компьютерной программы, компьютерной установки, которая описывает структуру и воспроизводит поведение реальной системы во времени. Имитационная модель позволяет получать подробную статистику о различных аспектах функционирования системы в зависимости от входных данных.

Имитационные модели могут создаваться в виде программ технологией прямого программирования, а также с помощью систем компьютерного моделирования, представленных на рис. 1.4.

В системах компьютерной математики, технического и имитационного моделирования предусмотрены возможности создания статистических (монте-карловских) имитационных моделей (генераторы случайных чисел, генераторы случайных величин, распределений и т.д.) и компьютерных имитационных моделей сложных систем.

В настоящее время идет стремительное развитие направления разработки инструментальных средств имитационного моделирования (ИСИМ) целенаправленно поддерживающих те или иные методологии и направления имитационного моделирования сложных систем (ИМСС):

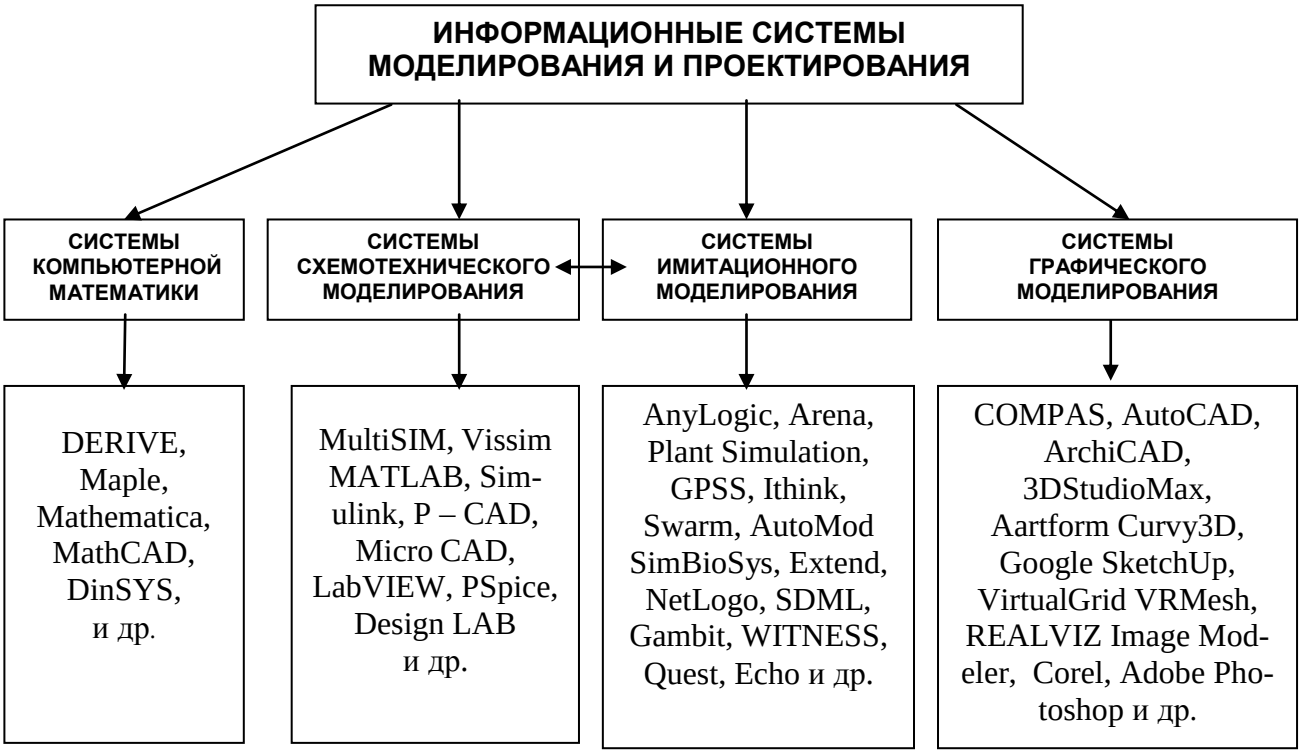


Рис. 1.4. Условная классификация информационных систем по типу решаемых задач

- AnyLogic — программного обеспечения для имитационного моделирования сложных систем и процессов, позволяющего поддерживать направление агентного моделирования, дискретно-событийного моделирования и разработки моделей системной динамики (разрабатывается российской компанией (англ. XJ Technologies) «Экс Джей Технолоджис»);
- GPSS (англ. General Purpose Simulation System — общецелевой системы моделирования) — языка объектно-ориентированного программирования, используемого для имитационного моделирования систем массового обслуживания, различных информационных процессов и разработки имитационных моделей в сети интернет;
- Arena – разработываемого компанией Systems Modeling Corporation программного обеспечения для имитационного моделирования, позволяющего создавать подвижные компьютерные модели, используя которые можно адекватно представить очень многие реальные системы;
- Plant Simulation — программной среды имитационного моделирования систем и процессов, предназначенного для оптимизации материалопотоков, загрузки ресурсов, логистики и метода управления для всех уровней планирования от целого производства и сети производств до отдельных линий и участков;

- SimBioSys: C++ – оболочки агентно-базового эволюционного моделирования в биологических и общественных науках;
- системы моделирования SWARM и его расширения MAML (Multi-Agent Modelling Language) для моделирования искусственного мира;
- пакетов Ascape(Agent Landscape) и RePast (Recursive Porous Agent Simulation Toolkit), написанных на платформе языка Java, для поддержки агентно-базового моделирования;
- NetLogo и MIMOSE (Micro- and Multilevel Modelling Software) информационных систем, предназначенных для создания имитационных моделей и технологий моделирования в общественных науках;
- SPSS, Statistica, PilGrim, Z-Tree – систем статистического моделирования для исследования экономических, педагогических и психологических явлений и процессов.

Перечень программного обеспечения и инструментальных средств имитационного моделирования можно посмотреть на сайте http://dic.academic.ru/dic.nsf, а также сайте Национального общества имитационного моделирования: www.simulation.su.

В отрасли имитационного моделирования реальных объектов условно выделились четыре основных направления: моделирование динамических систем, дискретно-событийное моделирование, агентное моделирование и системная динамика. В таблице 1.2 приведены языки и средства автоматизации имитационного моделирования, которые однозначно, а некоторые условно можно отнести к соответствующим подходам (методологиям) имитационного моделирования.

Таблица 1.2

Инструментальные средства имитационного моделирования			
Динамические системы	Системная динамика	Дискретное событийное моделирование	Агентное моделирование
Dynamo, PowerSim, MIMIC, АРТОН MIDAS, PACTOLUS, CSSL, СЛАМ, GASP, НЕДИС, МИКС, MATLAB+Simulink, Multisim VisSim, LabView, Easy5, MvStudium и др.	AnyLogic, Arena, SimBioSys, eM-Plant, Tecnomatix, Plant Simulation, SimuLab, VenSim, PowerSim, Pilgrim, Dynamo, Stella, Ithink и др.	AnyLogic, Arena, Extend, PowerSim Studio, Witness, ProModel, Pilgrim, Taylor Simulation, GPSS, SimScript, Quest, SIMULA, SIMUL8, Modelling, SimProcess, AutoMod, Enterprise Dynamics, FlexSim и др.	AnyLogic, Swarm+MAML, SimAgent, SimBioSys, C++, Java, AgentSpeak, Oz, TeleScript, RePast, NetLogo, Ascape, Mason и др.

Как видно из таблицы систем имитационного моделирования достаточно много, однако, не все перечисленные программные продукты доступны для использования. Многие программные продукты, представленные в таблице 1.2, по автоматизации имитационного моделирования не используются в России.

Это связано с тем, что отсутствуют представительства разработчиков этих систем имитационного моделирования или большинство из представленных инструментальных средств являются коммерческими и недоступны по причине дороговизны продукта как для университетов, так и для коммерческих IT – компаний занимающихся имитационными исследованиями. Наиболее широко используемые зарубежные системы имитационного моделирования в России: Arena (www.interface.ru), GPSS World (www.gpss.ru), платформа ARIS (www.softwareag.com/ru), VISSIM, VISUM (www.ptv-vision.ru), Quest Delmia Solution (www.3ds.com) и др.; свободно распространяемые и студенческие продукты имитационного моделирования Simplex3, Plant Simulation и др.

Системная динамика и дискретно-событийное моделирование – традиционные устоявшиеся подходы, агентное моделирование – относительно новый подход. Подход динамического моделирования позволяет увидеть поведение модели во времени при движении в прошлое (для получения исторического результата) и в будущее (для выявления возможных исходов). При изменении параметров модели можно наблюдать причины успеха или неудачи, находить оптимальные решения. Математически, системная динамика и динамические системы оперируют в основном с непрерывными во времени процессами, тогда как дискретно-событийное и агентное моделирование – в основном с дискретными.

Для разработки моделей сложных систем по данным подходам используются среды имитационного моделирования, разработанные в России: AnyLogic (www.anylogic.ru), Pilgrim (www.mfpa.ru), Rand Model Designer (www.mvstudium.com), расширенный редактор GPSS (www.elina-computer.ru) и др.

1.4. Системный анализ и этапы имитационного моделирования сложных систем

Большинство изучаемых и подлежащих моделированию объектов являются сложными системами. Характерные признаки сложной системы – невозможность рассмотрения отдельно каждого элемента (без установления связей с другими элементами и внешней средой), неопределенность, прояв-

ляющаяся в большом числе возможных состояний системы, неопределенность достоверности исходной информации, разнообразие вариантов путей достижения конечной цели функционирования системы, адаптивность (приспосабливаемость системы к возмущающим факторам воздействия внешней среды). Эти особенности вызывают необходимость использования методологии системного анализа при создании имитационной модели сложного объекта. Для анализа сложных объектов и процессов рассматривают системные направления, включающие в себя следующие термины: системный подход, системные исследования, системный анализ.

Системный подход. Этот термин начал применяться в первых работах, в которых элементы общей теории систем использовались для практических приложений. Заимствованные при этом понятия теории систем вводились не строго, не исследовался вопрос, каким классом систем лучше отобразить объект, какие свойства и закономерности этого класса следует учитывать при конкретных исследованиях и т. п. Иными словами, термин «системный подход» практически использовался вместо терминов «комплексный подход», «комплексные исследования». Под системным подходом понимается — направление методологии научного познания, в основе которого лежит рассмотрение объекта как системы: целостного комплекса взаимосвязанных элементов[6]; совокупности взаимодействующих объектов[7]; совокупности сущностей и отношений (Холл А. Д., Фейджин Р. И., поздний Берталанфи)[8].

В настоящее время системный подход рассматривается как начальная фаза современного системного анализа, как стадия (имеющих несколько этапов) первоначального качественного анализа проблемы решения и постановки задач исследования.

Системные исследования. В работах под этим названием понятия теории систем используются более конструктивно: определяется класс систем, вводится понятие структуры, а иногда и правила ее формирования и т. п. Это был следующий шаг в системных направлениях. В поисках конструктивных рекомендаций появились системные направления с разными названиями: системотехника, системология и др. Для их обобщения стал применяться термин «системные исследования». Часто в работах использовался аппарат

[6] Блауберг И.В., Мирский Э.М., Садовский В.Н. Системный подход и системный анализ // Системные исследования. – М., 1982. – С. 47-64.
[7] Берталанфи Л. Общая теория систем: критический обзор // Исследования по общей теории систем. – М., 1969. – С.23-82.
[8] http://ru.wikipedia.org/wiki

исследования операций, который к тому времени был больше развит, чем методы конкретных системных исследований[9].

Системный анализ. В настоящее время системный анализ является наиболее конструктивным направлением анализа систем[10]. Здесь предлагается методология проведения исследования, делается попытка выделить этапы исследования и предложить методику выполнения этих этапов в конкретных условиях. Особое внимание при анализе систем уделяется определению целей системы, вопросам формализации представления целей.

Системный анализ в широком смысле – это методология (совокупность методических приемов) постановки и решения задач построения и исследования систем, тесно связанная с математическим моделированием [28, 33, 46, 71, 80, 97, 144]. В более узком смысле системный анализ – методология формализации сложных (трудно формализуемых, плохо структурированных) задач. Системный анализ возник как обобщение приемов, накопленных в задачах исследования операций и управления в технике, экономике, военном деле. Соответствующие модели и методы заимствовались из математической статистики, математического программирования, теории игр, теории массового обслуживания, теории автоматического управления. Системный анализ – это целенаправленная творческая деятельность человека, на основе которой обеспечивается представление исследуемого объекта в виде системы. Системный анализ характеризуется упорядоченным составом методических приемов исследования.

Системный анализ – направление анализа, содержащее методику разделения процессов на этапы и подэтапы, систем на подсистемы, целей на подцели и т. д. В системном анализе выработана определенная последовательность действий (этапов) при постановке и решении задач, которую будем называть алгоритмом (методикой) системного анализа. Эта методика помогает более осмысленно и грамотно ставить и решать прикладные задачи. Если на каком-то этапе возникают затруднения, то нужно вернуться на один из предыдущих этапов и изменить (модифицировать) его. Если и это не помогает, то это значит, что задача оказалась слишком сложной и ее нужно разбить на несколько более простых подзадач, т. е. провести декомпозицию. Каждую из полученных подзадач решают по той же методике.

Метод системного анализа с успехом применяется к решению самых разных проблем в практически любой области – от проблем корпоративного управления и принятия управленческих решений, до моделей в области со-

[9] http://ru.convdocs.org/docs/index-12468.html?page=7#187925
[10] http://www.tssa.pisem.net/

циологии, экономики, физики, информатики, биологии и т.п. Более того, в последние годы этот метод используется как один из основных подходов к анализу и построению структуры диссертационных исследований в любых отраслях.

Остановимся на этих ранних этапах основного содержания деятельности системного аналитика. Для всех следующих этапов имитационного моделирования эта работа важна. Именно здесь же специалист по имитационному моделированию может проявить себя как системный аналитик, который владеет таким искусством, как моделирование.

На рис. 1.5 представлена схема проведения имитационного исследования сложной системы, предложенная Р.Шенноном [46, 144].

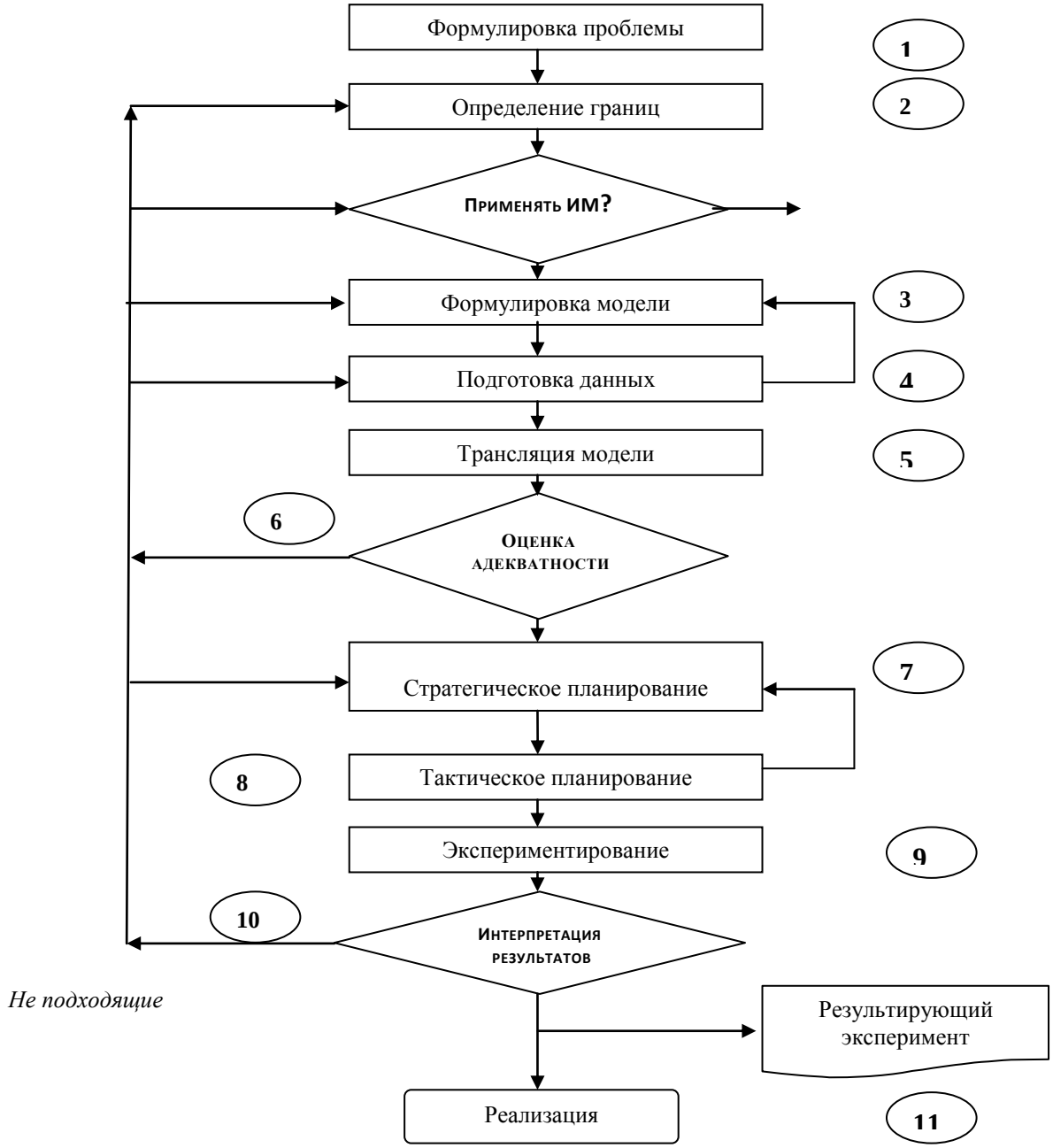


Рис.1.5. Этапы разработки компьютерной модели сложной системы

В каждом цикле разработки компьютерных моделей сложных объектов можно выделить следующие этапы.

1. Формулировка проблемы

Здесь проводится описание исследуемой проблемы и определение целей исследования. Постановка задачи, формулировка и установление иерархии целей и подзадач. Изучение поведения системы в целом.

Результатом этого этапа должно быть документированное содержательное описание объекта моделирования. Иначе говоря, построение полной информационной модели объекта или системы.

Анализ проблемы начинается с детального изучения всех его аспектов функционирования. Здесь очень важно понимание деталей, поэтому надо взаимодействовать с экспертами, либо быть хорошим специалистом в конкретной предметной области. Данная система связана весьма тесно с другими системами, поэтому важно правильно и четко определить задачи, но при этом задача моделирования разбивается на частные задачи.

Разработаны алгоритмы системного подхода к решению проблемы, они представлены в виде ряда этапов (см. рис.1.6).

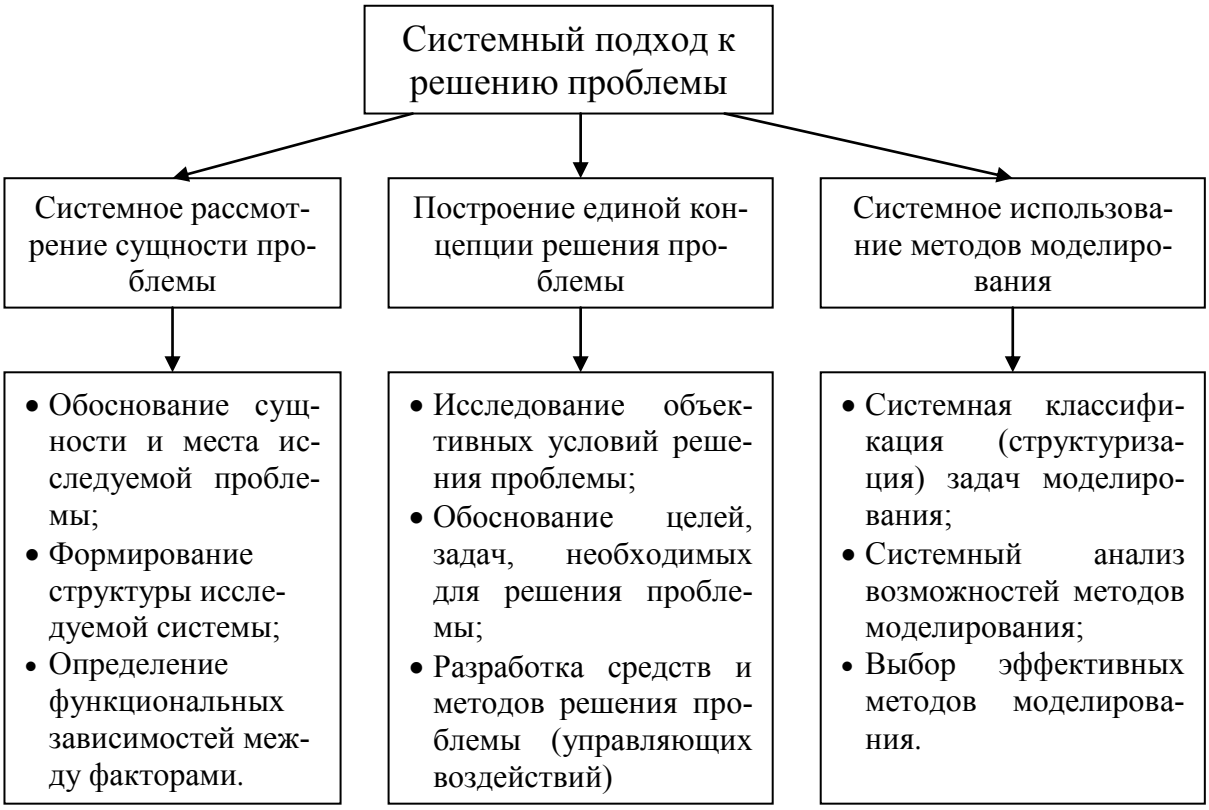


Рис. 1.6. Содержание системного подхода к решению проблем

Первый и самый решающий шаг при создании абсолютно любой модели моделирования состоит в обосновании ее целевого назначения. Воз-

можно применение метода декомпозиции целей, который предполагает разделение целого на части: задач – на подзадачи, целей – на подцели и т.д. Если использовать такой подход на практике, то он приводит к иерархическим древовидным структурам (т.е. построение дерева целей).

Остановимся на более употребляемых категориях целей в имитационном исследовании: предсказание, сопоставление альтернатив, оценка, оптимизация и др. Эксперименты по моделированию проводятся с разнообразными целями:

- прогноз – критика поведения системы при некоем предполагаемом сочетании рабочих условий;
- сравнение альтернатив – сравнение соперничающих систем, рассчитанных на выполнение конкретной функции, либо же на сравнение 1-го и более предлагаемых рабочих принципов либо методик;
- выявление многофункциональных соотношений – определение зависимости между двумя или более действующими факторами, с одной стороны, и откликом этой системы, с другой стороны;
- анализ чувствительности – обнаружение из чуть большего числа работающих факторов тех, которые в большей степени воздействуют на все поведение системы;
- оценка – определение, как буквально система предлагаемой структуры станет подходить неким конкретным аспектам;
- оптимизация – конкретное определение сочетания работающих величин и их причин, обеспечивающих наилучший отклик всей системы в целом;
- демонстрация – показ возможностей модели и имитационных исследований системы.

2. Определение границ

Логико-математическое описание моделируемой системы в соответствии с формулировкой проблемы. Определение границ системы и внешней среды, т.е. выделение системы из окружающей среды. Определение входных параметров и выходных характеристик системы.

В каждой модели существует некоторая комбинация составляющих как переменные, параметры, компоненты, функциональные зависимости, ограничения, целевые функции (аспекты).

При описании моделируемой системы и процессов, определяются основные *параметры и переменные* модели. *Параметрами* являются величины, которые исследователь может выбирать произвольно, в отличие от пере-

менных модели, которые могут принимать значения, определяемые видом заданной функции.

Компонентами системы считают составные части, которые образуют систему. Иногда компонентами считают также элементы системы или ее подсистемы. Система определяется как группа или совокупность объектов, объединенных некоторой формой регулярного взаимодействия или взаимозависимости для выполнения заданной функции.

Функциональные зависимости описывают поведение переменных и параметров в пределах компоненты или же выражают соотношения между компонентами системы. Эти соотношения по природе являются либо детерминистскими, либо стохастическими.

Ограничения представляют собой устанавливаемые пределы изменения значений переменных или ограничивающие условия их изменений. Они могут вводиться либо разработчиком, либо устанавливаться самой системой вследствие присущих ей свойств.

Целевая функция (функция критерия) представляет собой точное отображение целей или задач системы и необходимых правил оценки их выполнения. Выражение для целевой функции должно быть однозначным определением целей и задач, с которыми должны соизмеряться принимаемые решения (см.предыдущий этап).

3. Формулировка и разработка модели

Включает в себя разработку концептуальной модели и формализацию построенной концептуальной модели.

3.1. На этом стадии работы результатом деятельности разработчика компьютерной модели является создание полной концептуальной модели.

Концептуальная (содержательная) модель – это абстрактная модель, определяющая структуру моделируемой системы, свойства ее элементов и причинно-следственные связи, присущие системе и существенные для достижения цели моделирования. Построение концептуальной модели включает в себя декомпозицию системы, определение и выделение основных компонент, элементов и подсистем (построение модели состава). Концептуальная модель – это логико-математическое описание смоделированной системы в соответствии с формулировкой проблемы.

Главным содержанием этого шага является переход от настоящей системы к логической схеме ее функционирования, формулировка всеобщего плана модели. В этом шаге приводится алгоритмизация функционирования

ее составляющей и отображение объекта в терминах математических понятий.

Итогом работы на предоставленном шаге является избранный метод формализации моделируемой системы, документированное концептуальное отображение. Если формируют малые модели, то этот шаг совмещается с шагом составления содержательного описания моделируемой системы, на котором уточняется способ имитационного опыта. Построение концептуальной модели наступает с определения действия наружной среды, на базе цели моделирования устанавливаются рубежи моделируемой системы, выдвигаются гипотезы и укрепляются все дозволения (предположения), какие нужны для построения имитационной модели, обсуждается степень детализации моделируемых процессов.

Найти систему можно как совокупность взаимосвязанных частей. Определение системы зависит от того, кто определяет систему и от цели моделирования. На данном этапе выполняется декомпозиция системы, определяются более значительные взаимодействия между ними, в смысле сформулированной трудности, составляющие системы (выполняется структурный анализ моделируемой системы), выявляются главные аспекты функционирования моделируемой системы (составляется многофункциональная модель), приводится изображение внешней среды. Выделение подсистем либо декомпозиция системы (объекта моделирования) – это процедура анализа. Составляющие таковой модели должны быть реально существующим фрагментом в системе, а сложная система разбивается на части, которая охраняет при этом связи, обеспечивающие взаимодействие. Разрешено составить многофункциональную схему, проясняющая специфику динамических процессов, какие проистекают в рассматриваемой системе, и принципиально найти, какие будут вынесены во внешнюю обстановку, какие элементы будут введены в модель и какие взаимосвязи будут постановлены между ними.

Итак, вначале находится “элементарность” – составляется наиболее простое дерево целей, строится упрощенная конструкция модели. Далее проводится постепенная детализация модели. Используется метод «от простого к сложному». Сначала строится простая модель, затем она усложняется. Здесь применяется принцип итеративного построения модели (иерархический метод), когда по мере исследования системы по модели, в ходе ее разработки, модель меняется в результате добавления новых, либо исключения некоторых её элементов и/или взаимосвязей между ними.

Как перейти от реальной системы к её упрощенному (приближенному) описанию? Упрощение, приближение – основной прием любого моделирования. Избранный уровень детализации должен позволять абстрагироваться от неточно определенных, из-за аспектов функционирования реальной системы, вследствие недочета информации.

Под упрощением (приближением) понимается пренебрежение несущественными деталями, либо принятие догадок о наиболее простых соотношениях (к примеру, предположение о линейной зависимости между переменными). При моделировании выдвигаются гипотезы, догадки, относящиеся к взаимосвязи между компонентами и переменными системы.

Иным аспектом разбора реальной системы является абстракция, она содержит в себе существенные качества поведения объекта, однако не обязательно в такой же форме и столь подробно, как это имеет место в реальной системе.

После анализа элементов и подсистем приступаем к их соединению и объединению в единое целое. В концептуальной модели должно быть корректно отражено их взаимодействие. Композиция есть операция синтеза, агрегирование (при системном моделировании это не всегда сводится к элементарной сборке компонентов). В ходе операции выполняется введение отношений между элементами (к примеру, уточняется конструкция, приводится описание отношений, упорядочение и др.).

Таким образом, системное исследование построено на сочетании операций анализа и синтеза: проведение анализа взаимосвязей подсистем, установление связей между компонентами, подсистемами, элементами (построение модели структуры связей); принятие гипотез и допущений, физическая схематизация, иначе говоря, построение общей структуры системы с учетом всех подсистем и связей.

Для разработки концептуальной модели часто используются информационные системы проектирования BPwin, Erwin, Rational Rose, CASE Аналитик, ARIS Toolset и др., в которых можно провести контекстную и функциональную декомпозицию системы, потоков данных, управляющих потоков, определить структуру данных, построить диаграммы «сущность-связь».

3.2. Формализация построенной концептуальной модели осуществляется с помощью языка или аппарата математических методов, в том числе и имитационных технологий. В зависимости от сложности объекта и цели моделирования выбирается один из подходов аналитического или имитационного моделирования (статистический подход, динамическая система, дис-

кретно-событийное, мультиагентное моделирование, системная динамика, когнитивное, SIE – моделирование). В рамках выбранного подхода проводится разработка математического описания объекта моделирования. Результатом этого этапа является разработка технического проекта компьютерной установки для моделирования.

Процесс формализации сложной системы включает следующие виды работ:

- выбор способа формализации;
- составление формального описания системы.

В процессе построения модели можно выделить 3 уровня ее представления:

- неформализованный (этап 2) – концептуальная модель;
- формализованный (этап 3) – формальная модель;
- программный (этап 4) – имитационная модель.

Каждый уровень отличается от предыдущего степенью детализации моделируемой системы и способами описания ее структуры и процесса функционирования. При этом уровень абстрагирования возрастает.

Концептуальная модель – это систематизированное содержательное описание моделируемой системы (или проблемной ситуации) на неформальном языке. Неформализованное описание разрабатываемой имитационной модели включает определение основных элементов моделируемой системы, их характеристики и взаимодействие между элементами на собственном языке. При этом могут использоваться таблицы, графики, диаграммы и т.д. Неформализованное описание модели необходимо как самим разработчикам (при модификации, проверке адекватности модели и т.д.), так и для взаимопонимания со специалистами других профилей. Концептуальная модель содержит исходную информацию для системного аналитика, выполняющего формализацию системы и использующего для этого определенную методологию и технологию на основе формализованного описания осуществляется разработка более строгого и подробного формализованного описания.

Формализация объекта исследования осуществляется на основе той методологии имитационного моделирования, которая подходит к данной системе. Наблюдается множество схем (концепций) формализации и структуризации, которые пошли в применение в имитационном моделировании. Такие таблицы формализации исходят из различных понятий об изучаемых процессах и ориентируются на разные математические теории. Отсюда множество схем формализации и трудности отбора подходящей для описа-

ний данного предмета моделирования. В настоящее время существуют разработанные подходы и методологии формализации имитационного моделировании: методы статистического моделирования, моделирования динамических систем, системной динамики, дискретно-событийного моделирования и др.

Имитационная модель – преобразование формализованного описания в программу – имитатор, построенную в соответствии с некоторой методикой в средах программирования или моделирования. Аналогичная схема имеет место и при выполнении имитационных экспериментов: содержательная постановка отображаются на формальную модель, после чего вносятся необходимые изменения и дополнения в методику направленного вычислительного эксперимента. Основная задача этапа формализации – дать формальное описание сложной системы, свободное от второстепенной информации, имеющейся в содержательном описании, алгоритмическое представление объекта моделирования.

Цель формализации при аналитическом моделировании – построить аналитическую модель в виде каких-либо уравнений (линейных, дифференциальных, интегро-дифференциальных и др.).

Цель формализации при имитационном моделировании – получить формальное представление логико-математической модели, т.е. алгоритмов поведения компонентов сложной системы и отразить на уровне моделирующего алгоритма взаимодействие между компонентами.

Таким образом, концептуальное или формальное описание модели сложной системы на уровне формализации это построение имитационной компьютерной модели «программы – имитатора» в соответствии с некоторой методикой программирования с применением языков и систем автоматизации моделирования. Выбор инструментального средства для построения компьютерной модели является основным моментом в имитационном исследовании сложной системы.

4. Подготовка данных

Включает идентификацию, спецификацию и сбор данных.

Идентификация – статистический анализ модели, статистическое оценивание неизвестных параметров.

Спецификация – определение конечных целей моделирования; определение набора экзогенных и эндогенных переменных; определение состава системы уравнений, их структур; формулировка исходных предпосылок, ог-

раничений. Спецификация опирается на имеющиеся экономические теории, специальные знания, интуицию исследователя.

В подготовку данных входит сбор и анализ исходных данных для моделирования. Если трассировку и программирование имитационной модели можно выполнять на гипотетических данных, то предстоящее экспериментальное исследование нужно выполнять на реальном потоке данных, так как от этого зависит адекватность модели реальной системе и точность получаемых результатов моделирования. Здесь перед разработчиком имитационной модели встают два вопроса: где и каким образом получить и собрать исходную информацию, и как обработать собранные данные о жизненной системе.

Основные способы получения исходных данных:

- из экспериментальных данных (физический эксперимент);
- из документации на систему (финансовая и техническая документация для промышленных систем, данные отчетов, статистические сборники, к примеру, для социально-экономических систем и др.);
- из литературных источников по рассматриваемой системе.

Бывает так, что для задания начальной информации нужно вести предварительный, априорный синтез данных, натурные опыты на моделируемой системе либо на ее прототипах, а время от времени исходные данные имеют все шансы не быть, а моделируемая система исключает вероятность физического эксперимента, и тогда дают различные приемы предварительного объединения данных. При моделировании информационных систем длительность выполнения информационного требования оценивается на основании трудоемкости (реализуемых на ЭВМ) алгоритмов. К таким способам относят различные процедуры, основанные на общем обзоре проблематики, анкетировании, интервьюировании, большом применении методов экспертного оценивания.

Второй вопрос связан с проблемами идентификации входных данных для стохастических систем. Имитационное моделирование является весьма эффективным аппаратом исследования стохастических систем, т.е. таких систем, динамика которых зависит от случайных факторов: входные (либо выходные) переменные стохастической модели, случайные величины, векторы, функции, случайные процессы. Именно поэтому появляются лишние трудности, связанные с синтезом уравнений относительно неизвестных законов распределения и определением вероятностных характеристик (математических ожиданий, дисперсий, корреляционных функций и т.п.) для ана-

лизируемых процессов и их параметров. Необходимость статистического анализа при анализе и сборе входных данных связана с целью определения вида функциональных зависимостей, описывающей входные данные, оценкой конкретизированных значений параметров этих зависимостей, а также проверкой значимости параметров. Для подбора теоретических распределений случайных величин используют известные приемы математической статистики, основанные на определении параметров эмпирических распределений и проверке статистических гипотез, с применением критериев согласия, согласуются ли экспериментальные данные с известными законами распределения.

5. Трансляция модели

Трансляция модели – это перевод модели со специальных имитационных языков или языка математики на язык программирования, на котором будет реализована прикладная программа, соответствующая компьютерной модели. Алгоритмизация и программная реализация, т.е. строится программный комплекс моделирования объекта исследования. Проводится отладка компьютерной модели.

6. Оценка адекватности (верификация и валидация)

Верификация – это установка правильности разработанной программы, формальное, либо практическое доказательство ее правильной работоспособности на ЭВМ. На этом этапе проводится испытание, корректировка, проверка модели, комплексное тестирование компьютерной модели на адекватность объекту моделирования.

Валидация – это оценка требуемой точности и адекватности имитационной модели.

После реализации имитационной модели на ЭВМ, необходимо проводить испытания для оценки достоверности модели. В периоде испытания и исследования разработанной имитационной модели организуется комплексное тестирование модели (testing) – планируемый итеративный процесс, направленный на поддержку операций верификации и валидации имитационных моделей и данных.

Если в случае проведенных процедур модель окажется недостаточно подлинной, то может быть осуществлена калибровка имитационной модели (в моделирующий алгоритм встраиваются калибровочные коэффициенты) с целью снабжения адекватности модели. В более трудных случаях достижимы большие итерации на ранние периоды с целью снятия дополнительной информации о моделируемом объекте или доработки имитационной модели.

Наличие ошибок во взаимодействии компонентов модели возвращает исследователя на этап создания имитационной модели. Причиной этого может быть с самого начала упрощенная модель процесса или явления, что приводит к неадекватности модели объекту. В случае, если выбор способа формализации оказался весьма неудачным, то должно повторить этап создания концептуальной модели с учетом свежей информации и приобретенного опыта. Наконец, если оказалось , что информации об объекте мала, то нужно вернуться к периоду составления полного описания системы и конкретизировать его с учетом итогов испытания.

7-8. Стратегическое и тактическое планирование

На этом этапе проводится стратегическое и тактическое планирование машинного эксперимента. Результатом является составленный план эксперимента и проведенный вычислительный эксперимент («прогоны» компьютерной модели с различными начальными данными). Здесь проводится определение условий машинного эксперимента с имитационной моделью, а также параметров при тестировании модели, результаты по входным данным.

9. Постановка экспериментов. На данном этапе предполагается прогон программы имитационной модели на ЭВМ для получения выходных данных или результатов, позволяющих оценить адекватность построенной модели. Здесь также необходимо определить условия, в которых будет осуществляться тестирование, проверка работоспособности и возможности функционирования; параметры, на которые надо обратить внимание при тестировании модели. Параметры могут быть связаны со способностью модели реагировать на какие-либо стохастические воздействия, на неверные входные данные, либо полное их отсутствие, на неверные действия персонала.

Далее проводится вычислительный эксперимент на имитационной модели. На последних стадиях имитационного моделирования необходимо вести стратегическое и тактическое планирование имитационного эксперимента. Организация направленного вычислительного эксперимента на имитационной модели считает выбор и использование различных аналитических приемов для обработки итогов имитационного исследования. Ради этого применяются способы планирования вычислительного эксперимента, статистический, регрессионный и дисперсионный анализ, методы оптимизации. Организация и проведение опыта требует корректного использования возможных аналитических приемов. Согласно полученным результатам проведенное обследование должно позволить сделать выводы, достаточные для

принятия решений по обозначенным на ранних стадиях проблемам и задачам.

10. Анализ результатов моделирования

Обработка, визуализация и интерпретация результатов машинного компьютерного эксперимента, предполагает рассмотрение и изучение результатов имитационного эксперимента для подготовки выводов о возможности применения имитационной модели для решения некоторой проблемы.

11. Реализация и документирование

На основе построенной имитационной модели можно дать рекомендации о принятии того или иного управленческого решения и документально отразить процесс функционирования модели и полученные результаты. Анализ основных проблем, возникающих при создании модели сложной системы.

Более подробно перечнем работ на каждом этапе разработки имитационных моделей сложных систем можно ознакомиться в работах авторов [46, 60, 72, 106, 124].

1.5. Проектирование и разработка имитационных моделей сложных объектов

Основные стадии и этапы в стадиях для разработки компьютерных моделей сложных систем как разновидности программного обеспечения или информационной системы может быть кратко представлено следующим образом [10, 11, 21, 30, 43, 77, 136]:

1. Предпроектная стадия – стадия формирования требований к автоматизированной системе
   - этап разработки концепции автоматизированной системы;
   - этап разработки и утверждения технического задания.
2. Стадия проектирования и разработки программного обеспечения:
   - этап разработки эскизного и технического проекта автоматизированной системы;
   - этап проектирования программного обеспечения;
   - этап проектирования интерфейса;
   - этап реализации программного обеспечения (создание программного кода);
   - этап создания и оформления документации.
3. Стадия внедрения.
4. Период сопровождения или пользовательский период.

Эти стадии создания автоматизированных систем или программного продукта применимы также для разработки компьютерных установок для моделирования.

Разработка технического задания является обязательной, в приложении приведены образцы оформления технического задания на разработку компьютерных моделей сложных объектов для учебных целей [приложение 1].

Практически любая современная крупная программная система разрабатывается с применением CASE-технологий по крайней мере на этапах анализа и моделирования, что связано с большой сложностью данной проблематики и со стремлением повысить эффективность работ [29, 30, 142].

Целью систем анализа и проектирования является определение системных требований и свойств, создание проекта и архитектуры информационной системы, а также детальная «калька» проекта, включающая алгоритмы и определения структур данных [10, 11, 77, 136, 142].

Системы проектирования баз данных обеспечивают логическое моделирование данных, автоматическое преобразование моделей данных в Третью Нормальную Форму, автоматическую генерацию схем Базы данных и описаний форматов файлов на уровне программного кода.

В таблице 1.1 приведены некоторые CASE- средства анализа и проектирования программных систем. Здесь приведены производители этих средств, нотации и диаграммы, которые они поддерживают.

Эти системы поддерживают те или иные методологии анализа и проектирования информационных систем, которые определяют руководящие указания для оценки и выбора проекта программного продукта, последовательность шагов выполнения работы, правила распределения и назначения операций и методов.

Таблица 1.1. Средства проектирования

Система проектирования	Производитель ПО	Нотация DFD	Поддержка	
			Поддержка методологии	Поддержка диаграмм
Системы анализа и проектирования				
BPWin	Logic Works	Гейн-Сарсон	IDEF0/SADT IDEF3/SADT	Функциональной декомпозиции
ProKit *Workbench	MDIS	Гейн-Сарсон	Собственная STRADIS	Потоков данных в нотации Гейна-Сарсона, сущность-связь, структурные карты Константайна
CASE. Аналитик	Эйтекс	Гейн-Сарсон	структурного системного анализа Гейн-Сарсона	функциональной декомпозиции, потоков данных, управляющих потоков, структуры данных, сущность-связь
CASE /4/0	MicroTO	Йодан	анализа и проек-	Функциональной декомпозиции, пото-

	OL	(расшир.)	тирования систем реального времени Уорда-Меллера	ков данных, переходов состояний, карты Джексона
De-sign/IDEF	Meta Software	-	IDEF0, IDEF1 IDEF1X, IDEF/CPN	сущность-связь
Visible Analyst Workbench	Visible Systems	Гейн-Сарсон, Йодан	Информационного моделирования Мартина	Функциональной декомпозиции, сущность-связь, потоков данных в нотации Йодана и Гейна-Сарсона, структурные карты Константайна
Rational Rose	Rational Software	UML, также поддерживается нотация Буча и OMT-2.	Собственная методология "Rational Objectory Process", UML	вариантов использования, взаимодействия объектов, последовательности взаимодействий, переходов состояний, классов, пакетов, компонентов, размещения
UML 2.x	Object Management Group	Собственная	UML	Классов, компонентов, композитной/составной структуры, кооперации (UML 2.0), объектов, пакетов, профилей (UML 2.2), деятельности, состояний, прецедентов, коммуникации (UML 2.0) / кооперации (UML 3.x), обзора взаимодействия (UML 2.0), последовательности, синхронизации (UML 2.0)
ARIS Toolset	Software AG	Нотация ARIS eEPC	UML	сущность-связь, описания процессов, информационных потоков
Системы проектирования баз данных и файлов				
Design-er/2000	Oracle	Гейн-Сарсон	Собственная CASE*Method	сущность-связь, потоков данных, иерархии функций, взаимодействия модулей структурные карты Джексона
ERWin	Logic Works	Гейн-Сарсон	IDEF3/SADT	сущность-связь
EasyCASE	Evergreen CASE Tools	Гейн-Сарсон, Йодан	структурного системного анализа Гейн-Сарсон	сущность-связь, потоков данных, переходов состояний, структурные карты в нотации Константайна
Vantage Team Builer	CAY-ENNE	Йодан	Структурного анализа и проектирования Чена и Йодана,	сущность-связь в нотации Чена, потоков данных в нотации Йодана, переходов состояний, структурные карты Константайна
Chen Toolkit	Chen & Asssociates	Чена	Чена	сущность-связь, потоков данных, переходов состояний

S-Designor	Sybase/ Powersoft		Информационного моделирования	сущность-связь
SILVERRUN	Computer Systems Advisers	произвольная	DATARUN	потоков данных BPM, сущность-связь ERX и RDM

Современный подход к проектированию информационных систем предполагает создание модели исходной информационной системы, описывающей все необходимые аспекты её функционирования. Применение моделей позволяет сократить сроки проектирования, улучшить качество проекта за счёт устранения большого числа ошибок в решении стратегических вопросов уже на ранних стадиях работы.

При создании такой модели обычно применяется функциональная методология [142]. Она предполагает рассмотрение системы в виде набора функций, преобразующих входной поток информации в выходной.

Функциональная модель – описание системы с помощью IDEF0. Данная модель предназначена для описания существующих бизнес-процессов, в которой используются как естественный, так и графический языки. Для передачи информации о конкретной системе источником графического языка является сама методология IDEF0.

Методология IDEF0 предназначена для построения иерархической системы диаграмм – единичных описаний фрагментов системы. Сначала проводится описание системы в целом и ее взаимодействия с окружающим миром (контекстная диаграмма), после чего проводится функциональная декомпозиция – система разбивается на подсистемы и каждая подсистема описывается отдельно (диаграммы декомпозиции). Затем каждая подсистема разбивается на более мелкие и так далее до достижения нужной степени подробности.

Каждая IDEF0-диаграмма содержит блоки и дуги. Блоки – функции моделируемой системы. Дуги связывают блоки вместе и отображают взаимодействия и взаимосвязи между ними.

Функциональные блоки (работы) на диаграммах изображаются прямоугольниками, означающими поименованные процессы, функции или задачи, которые происходят в течение определенного времени и имеют распознаваемые результаты.

Каждая сторона блока имеет особое назначение. Левая сторона блока предназначена для входов, верхняя – для управления, правая – для выходов, нижняя – для механизмов. Такое обозначение отражает определенные сис-

темные принципы: входы преобразуются в выходы, управление ограничивает или предписывает условия выполнения преобразований, механизмы показывают, что и как выполняет функция.

Остановимся на малых интегрированных средствах проектирования информационных систем. Типичный представитель малых интегрированных средств моделирования – комплект программных продуктов Platinum Technology (CA/ Platinum/Logic Works), основанный на популярных пакетах BPwin и Erwin [29, 30, 77, 142].

BPWin. Компания LogicWorks, разработчик BPwin, сейчас входящий в Computer Associates, работает на рынке технологий моделирования уже более 20 лет. Для проведения анализа и реорганизации сложных систем и процессов Logic Works предлагает CASE-средство верхнего уровня – BPwin, который поддерживает 3 методологии:

- IDEF0 (функциональная модель),
- IDEF3 (WorkFlow Diagram) – только диаграммы процессов,
- DFD (DataFlow Diagram) – диаграммы потоков данных.

Функциональная модель предназначена для описания существующих систем и процессов (так называемая модель AS-IS) и идеального положения вещей – того, к чему нужно стремиться (модель TO-BE).

Интеграция выполняется как путем слияния нескольких моделей, так и посредством переключения на различные методологии в процессе разработки отдельных диаграмм информационной модели. Предусмотрено расширение возможностей анализа систем как в самом пакете BPwin (функционально-стоимостный анализ), так и с помощью экспорта данных в другие пакеты. BPwin автоматизирует задачи, связанные с построением моделей развития, обеспечивая семантическую строгость, необходимую для гарантирования правильности и непротиворечивости результатов.

ERwin. Поддерживает несколько разновидностей методологии информационного моделирования, основанной на ER-диаграммах (сущность – связь). Интеграция моделей BPwin с моделями ERwin выполняется путем обмена данными через функции экспорта/импорта.

Основной из трех методологий является IDEF0, она относится к семейству IDEF, которое было введено в 1973 году Россом под названием SADT (Structured Analysis and Design Technique). Хотя основной акцент использования малых интегрированных систем делается в применении к бизнес-процессам на предприятиях, эти технологии применимы для декомпозиции и проектирования широкого класса систем, в том числе информационных сис-

тем, в частности для проектирования сайтов, порталов, систем и программных комплексов компьютерных установок для моделирования реальных объектов. Для информационных систем применение IDEF0 имеет своей целью определение требований и указание функций для последующей разработки системы, отвечающей поставленным требованиям и реализующей выделенные функции. Применительно к уже существующим системам IDEF0 может быть использована для анализа функций, выполняемых системой, и отображения механизмов, посредством которых эти функции выполняются.

Первая диаграмма в иерархии диаграмм IDEF0 всегда изображает функционирование информационной системы в целом (рис.1.7).

Такая диаграмма называется контекстной. В контекст входит описание цели моделирования, области (описания того, что будет рассматриваться как компонент системы, а что как внешнее воздействие) и точки зрения (позиции, с которой будет строиться модель). Функциональную модель дорожного движения по принципу «Зеленой волны» представлена на рис.1.8 и ее декомпозиция на рис. 1.9.

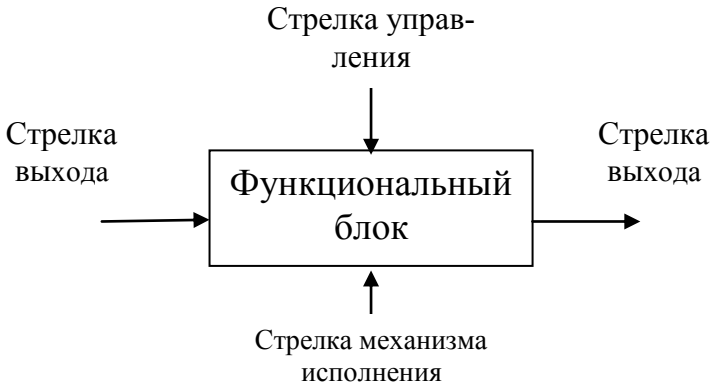


Рис. 1.7. Соединение стрелок со сторонами функционального блока

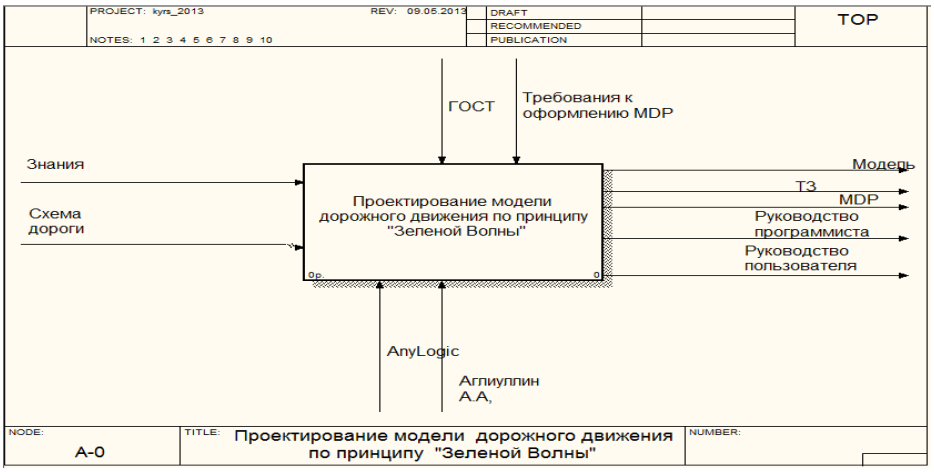


Рис.1.8. Функциональная модель дорожного движения

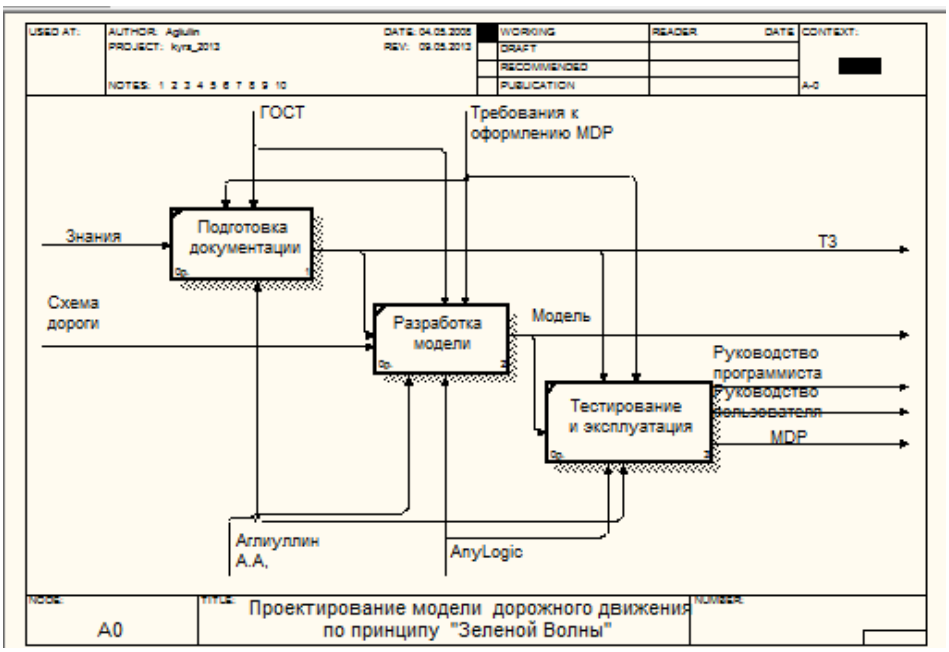


Рис.1.9. Модель декомпозиции

1.4. Основные направления и перспективы развития имитационного моделирования

Имитационное моделирование в настоящее время является одним из основных видов моделирования и исследования для экономических, производственных, экологических, демо- и энтографических систем, систем массового обслуживания и др. Оно заключается в создании модели-имитатора работы сложных (чаще всего при наличии стохастических факторов) систем и процессов при неполных знаниях о ряде процессов в моделируемых объектах. Одними из наиболее эффективных методов исследования указанных объектов и систем являются методы имитационного и комплексного моделирования. Имитационное моделирование (ИМ) — это метод исследования, который основан на том, что анализируемая динамическая система заменяется имитатором и с ним проводятся эксперименты для получения информации об изучаемой системе. Роль имитатора выполняет компьютерная программа или комплекс программ для ЭВМ, которая называется имитационной моделью.

Анализ перечисленных ранее особенностей формального описания и исследования сложных объектов и систем показывает, что при моделировании и управлении данными объектами и системами следует базироваться на концепциях и принципах, положенных в основу современных технологий системного (комплексного) моделирования. Более того, как показывает анализ, при решении актуальных в современных условиях проблем структурно -

функционального синтеза облика гибридных интеллектуальных систем управления (ГИСУ) СОТО целесообразно рассматриваемые технологии системного моделирования, традиционно связанные с количественными вычислениями, дополнить интеллектуальными информационными технологиями, ориентированными на символьную обработку информации. К указанным информационным технологиям принято относить [108]:

− технологии мультиагентного моделирования;

− технологии экспертных систем (Expert Systems) или систем, основанных на знаниях (Knowledge-Based Systems);

− технологии нечёткой логики (Fuzzy Logic);

− технологии искусственных нейронных сетей (Artificial Neural Networks);

− технологии вывода, основанного на прецедентах CBR (Case Based Reasoning) - технологии;

− технологии естественно-языковых систем и онтологии;

− технологии ассоциативной памяти;

− технологии когнитивного картирования и операционного кодирования;

− технологии эволюционного моделирования.

Несмотря на рост производительности современных компьютеров, их мощности не хватает для моделирования задач, связанных с самолето- и автомобилестроением, логистикой, сборочным производством и т. п., когда имитационные прогоны моделей могут длиться часами. Одним из вариантов решения этой проблемы является использование параллельного и распределенного дискретно-событийного моделирования. В этой области есть даже стандарт HLA – High Level Architecture.

Вопросам практического использования результатов имитационного и комплексного моделирования сложных процессов и систем уделяется особое внимание на многих конференциях, в частности на конференциях ИММОД [108].

Если говорить о направлениях построения и использования имитационных моделей, то они следующие:

1. Модели, предназначенные для выявления функциональных соотношений, т.е. для определения природы и закономерностей между двумя или несколькими факторами с одной стороны и откликом системы с другой стороны.

2. Модели, предназначенные для прогноза, т.е. для оценки поведения системы при некотором сочетании рабочих условий, в том числе и по времени.
3. Модели для экспертной оценки предлагаемой структуры или конфигурации системы по предлагаемым некоторым критериям или совокупностью аксиом, сформулированным с помощью экспертов. Эта совокупность является как бы основой разработанной экспертной системы (ядром) – имитационной модели.
4. Модели, позволяющие сопоставлять конкурирующие системы, рассчитанные на выполнение определенной функции, или сопоставление различных предлагаемых рабочих методик, принципов. Эти модели сравнения альтернатив разрабатываются для принятия решений, разрабатываются в виде утилиты, запускаются регулярно при принятии оперативных решений.
5. Модели для оптимизации в целях определения сочетания действующих факторов и их величин, при котором обеспечивается наилучший отклик системы в целом. Может использоваться как инструмент для оценки и сравнения вариантов предполагаемых изменений или выработки оптимальной стратегии. Эти модели широко применяются при решении задач управления (планирования, проектирования) и часто называются интерактивными оптимизационными моделями (системами).
6. Модели виртуальных игр, предназначенные для обучения студентов, управленческого персонала, сотрудников. Разработка моделей виртуальной реальности – инструментов для компьютерных имитационных игр для различных отраслей, военных, экономических, государственных и межгосударственных отношений.
7. Модели для анализа чувствительности, т.е. для выявления тех факторов, которые в наибольшей степени влияют на общее поведение системы.
8. Модели, встроенные в производственный процесс, в технические и автоматизированные установки, запускающиеся автоматически при выполнении соответствующих операций.
9. Модели, созданные для динамической визуализации (демонстрации) проектируемого объекта в целях аргументации разрабатываемого проекта для руководства или потенциальных заказчиков (воспроизводящую виртуальную иллюзию процесса развития системы).

Говоря о моделировании как исследовательском методе, и имитационном (компьютерном) моделировании как его разновидности, необходимо ос-

тановиться на основных областях применения, в которых могут решаться отраслевые задачи при помощи данного метода, о его преимуществах и особенно – ограничениях.

Активнее всего имитационным моделированием интересуются в металлургии, нефтегазовой отрасли, производстве стройматериалов, пищевых продуктов, в различных областях массового обслуживания (аэропорты, транспорт, медицина, торговые предприятия, сферы обслуживания и др.).

Анализ результатов конференций ИММОД показывает, что сводный перечень основных областей применения выглядит следующим образом [108]:

- предоставление услуг связи, сети передачи данных;
- управление подвижными (космическими) объектами, воздушным и автотранспортным движением;
- разработка ситуационных имитационных моделей полета военно-воздушного транспорта;
- организация промышленного производства (ГПС, обувное, мелкосерийное, сборка персональных компьютеров, разработка программного обеспечения);
- проектирование рыбообрабатывающих комплексов на судах промыслового флота;
- информационное противоборство, блочное шифрование;
- динамика популяции зверей и животных;
- региональные экономические системы;
- лечебно-эвакуационные мероприятия в авиадивизии;
- налоговое и пенсионное законодательство, обращение граждан в органы государственного управления;
- подготовка специалистов по управлению железнодорожным транспортом и магистральными трубопроводами;
- исследования в области экосистем (водных, морских, лесных, степных и др.);
- исследования демографических и этнических систем.

Из области применения имитационного моделирования можно сформировать основные тематические направления имитационных исследований:

- теоретические основы и методология имитационного и комплексного моделирования;
- методы оценивания качества моделей и полимодельных комплексов;
- методы и системы распределенного моделирования;

- моделирование глобальных процессов;
- разработка средств автоматизации и визуализации имитационного моделирования;
- системная динамика (с обязательным наличием имитационной составляющей);
- практическое применение моделирования и инструментальных средств автоматизации моделирования, принятие решений по результатам моделирования;
- виртуальное и цифровое производство – промышленное моделирование;
- имитационное моделирование в обучении и образовании.

Применение имитационного моделирования достаточно медленно развивается из-за сложности рассматриваемых объектов и трудоемкости создания имитационных моделей, из-за отсутствия профессиональных кадров в области имитационного моделирования и ряду других причин. Особенно сложно обстоит дело в областях социальных и гуманитарных наук, где слабое применение методов имитационного моделирования может быть объяснено из-за причин, изложенных выше, нелинейностью и многофакторностью социальных процессов, сложностью взаимосвязей, а также из-за необоснованных ожиданий, возлагаемых на модель, которые не могут оправдаться в реальности. Тем не менее, формализацией социальных систем в настоящее время занимаются, определяются математические подходы и получены некоторые результаты [45, 91, 100-101, 107, 121].

Одна из причин неуспеха прогнозирования, планирования и проектирования социальных систем в настоящее время (в том числе крупных и сложных социально-экономических объектов, регионов) состоит в том, что:

- отсутствуют компетентные в отрасли специалисты по системному анализу и имитационным исследованиям;

- отсутствуют соответствующие структуры в виде Центров, Институтов занимающихся имитационными исследованиями;

- отсутствуют возможности самостоятельного владения инструментами имитационных исследований у большинства органов управления в силу их стоимости и необходимости наличия высокооплачиваемых компетентных специалистов со стороны;

Предлагаемые подходы к моделированию и проектированию будущего (крупные региональные и всероссийские проекты в регионах, финансируемые крупными компаниями и государством) для уровня регионов и крупных компаний:

- информирование научной и производственной общественности о причинах, полезности и перспективах применения имитационного моделирования систем и технологий проектирования будущего;
- для регионов – создание соответствующих структур – Центров, Институтов имитационных, ситуационных, когнитивных исследований;
- широкое распространение систем проектирования и имитационного моделирования будущего и практики их применения;
- эффективное обучение применению технологий проектирования и моделирования будущего в университетах;
- воспроизводство и «расширенное производство» специалистов, владеющих методологиями и технологиями имитационного моделирования и проектирования будущего, в том числе создание новых образовательных программ по имитационному моделированию и проектированию будущего.

Контрольные вопросы

1. Что мы понимаем под объектом и как могут формироваться классы?
2. Что такое система, ее признаки и сущность?
3. Чем отличается сложная система от большой системы?
4. Что такое связь и структура системы?
5. Что такое процесс и ее основные характеристики?
6. Дайте определение события и ее характеристик? Что такое транзакт?
7. Какие виды имитационного эксперимента и применения компьютерных моделей Вы знаете?
8. Какие разновидности имитационного моделирования существуют на сегодняшний день?
9. Какие инструментальные средства используются для построения имитационных моделей?
10. Как соотносятся понятия системный подход и системный анализ?
11. Каковы этапы имитационного моделирования?
12. Какие среды проектирования информационных систем применяются для проектирования имитационных моделей?
13. Каковы основные направления построения и использования имитационных моделей существуют на сегодняшний день?
14. Перечислите области применения имитационного моделирования?

ГЛАВА 2

СРЕДА ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ ANYLOGIC 6

2.1. Общие сведения о системе имитационного моделирования AnyLogic

Вначале 1990-х гг. в компьютерной науке наблюдался большой интерес к построению математически трактуемого описания взаимодействия параллельных процессов. Небольшая группа учёных из Санкт-Петербургского Политехнического университета занималась разработкой пакета программ для анализа корректности параллельных и распределённых систем[11]. Такой программный продукт был создан и новый инструмент был назван COVERS (Параллельная Верификация и Моделирование). В этом пакете анализируемая система процессов задавалась графически, с помощью описания её структуры и поведения отдельных параллельных компонентов, которые могли взаимодействовать с окружением — с другими процессами и средой.

В 1998 г. успех этого исследования вдохновил лабораторию организовать коммерческую компанию с миссией создания нового программного обеспечения для имитационного моделирования. Акцент при разработке ставился на прикладные методы: моделирование стохастических систем, оптимизацию и визуализацию модели. Новое программное обеспечение, выпущенное в 2000 г., было основано на последних преимуществах информационных технологий: объектно-ориентированный подход, элементы стандарта UML, языка программирования Java, современного GUI и т.д.

Продукт получил название AnyLogic, потому что он поддерживал три известных направления имитационного моделирования (рис.2.1):

Рис 2.1. Три подхода имитационного моделирования

[11] http://ru.wikipedia.org/wiki/AnyLogic

А также любую комбинацию этих подходов в пределах одной модели. Первой версии был присвоен индекс 4 – Anylogic 4.0, так как нумерация продолжила историю версий предыдущей разработки – COVERS 3.0.

- системная динамика;
- дискретно-событийное (процессное) моделирование;
- агентное моделирование.

Огромный шаг вперёд был сделан в 2003 году, когда был выпущен AnyLogic 5, ориентированный на бизнес-моделирование. С помощью AnyLogic стало возможным разрабатывать модели в следующих областях:

– производство;
– логистика и цепочки поставок;
– рынок и конкуренция;
– бизнес-процессы и сфера обслуживания;
– здравоохранение и фармацевтика;
– управление активами и проектами;
– телекоммуникации и информационные системы;
– социальные и экологические системы;
– пешеходная динамика;
– оборона.

Версия AnyLogic 6 написана на языке программирования Java в популярной среде разработки Eclipse и является кросс-платформенным программным обеспечением, работает как под управлением операционной системы Windows, так и под Mac, OS и Linux.

AnyLogic включает в себя графический язык моделирования, а также позволяет пользователю расширять созданные модели с помощью языка Java. Интеграция компилятора Java в AnyLogic предоставляет более широкие возможности при создании моделей, а также создание Java апплетов, которые могут быть открыты любым браузером. Эти апплеты позволяют легко размещать модели AnyLogic на веб-сайтах. В дополнение к Java апплетам, AnyLogic Professional поддерживает создание Java приложений, в этом случае пользователь может запустить модель без инсталляции AnyLogic.

Система AnyLogic, разработанная компанией XJTechnologies (Россия), это среда компьютерного моделирования общего назначения. Это комплексный инструмент, охватывающий основные в настоящее время направления моделирования: дискретно-событийное, системной динамики, агентное. Ис-

пользование AnyLogic дает возможность оценить эффект конструкторских решений в сложных системах реального мира.

Отечественный профессиональный инструмент имитационного моделирования AnyLogic нового поколения, разработан на основе современных концепций в области информационных технологий и результатов исследований в теории гибридных систем и объектно-ориентированного моделирования. Построенная на их основе инструментальная система AnyLogic не ограничивает пользователя одной единственной парадигмой моделирования, что является характерным для существующих на рынке инструментов моделирования. В AnyLogic разработчик может гибко использовать различные уровни абстрагирования и различные стили и концепции и смешивать их при создании одной и той же модели.

Программный продукт AnyLogic основан на объектно-ориентированной концепции. Объектно-ориентированный подход к представлению сложных систем является лучшим на сегодняшний день методом управления сложностью информации, эта концепция позволяет простым и естественным образом организовать и представить структуру сложной системы. Таким образом, идеи и методы, направленные на управление сложностью, выработанные в последние десятилетия в области создания программных систем, позволяют разработчикам моделей в среде AnyLogic организовать мышление, структурировать разработку и, в конечном счете, упростить и ускорить создание моделей.

Другой базовой концепцией AnyLogic является представление модели как набора взаимодействующих параллельно функционирующих активностей. Такой подход к моделированию интуитивно очень понятен и естественен во многих приложениях, поскольку системы реальной жизни состоят из совокупности активностей, взаимодействующих с другими объектами. Активный объект AnyLogic — это объект со своим собственным функционированием, взаимодействующий с окружением. Он может включать в себя любое количество экземпляров других активных объектов. Активные объекты могут динамически порождаться и исчезать в соответствии с законами функционирования системы. Так могут моделироваться социальные группы, холдинги компаний, транспортные системы и т. п.

Графическая среда моделирования AnyLogic поддерживает проектирование, разработку, документирование модели, выполнение компьютерных экспериментов с моделью, включая различные виды анализа — от анализа

чувствительности до оптимизации параметров модели относительно некоторого критерия.

В результате AnyLogic не ограничивает пользователя одной-единственной парадигмой моделирования, что является характерным фактически для всех инструментов моделирования, существующих сегодня на рынке. В AnyLogic разработчик может гибко использовать различные уровни абстрагирования, различные стили и концепции, строить модели в рамках той или иной парадигмы и смешивать их при создании одной и той же модели, использовать ранее разработанные модули, собранные в библиотеки, дополнять и строить свои собственные библиотеки модулей. При разработке модели на AnyLogic можно использовать концепции и средства из нескольких «классических» областей моделирования, например, в агентной модели использовать методы системной динамики для представления изменений состояния среды или в непрерывной модели динамической системы учесть дискретные события. В AnyLogic легко строятся подобные модели с требуемым уровнем адекватности, позволяющие ответить на многие вопросы, интересующие исследователя. Богатые возможности анимации и визуального представления результатов в процессе работы модели позволяют понять суть процессов, происходящих в моделируемой системе, упростить отладку модели.

Многие крупные компании как отечественные Билайн, Газпром, Сбербанк России, Русский алюминий, Северсталь и т.д., так и зарубежные General Motors, Mitsubishi, McDonalds, IBM и др. являются клиентами компании и используют AnyLogic для своих потребностей и исследований (рис.2.2).

Сайт компании AnyLogic (www.anylogic.ru) позволяет продемонстрировать возможности данного продукта. На сайте представлены различные демо-версии имитационных моделей по различным отраслям (рис. 2.3).

Для каждого из отраслей на сайте приведены различные примеры демо-моделей (рис.2.4), которые можно прогнать в реальном времени. На сайте runthemodel.com приведены различные имитационные модели для различных отраслей логистики, промышленности, социальной динамики, транспортных систем, бизнеса, пешеходной и городской динамики, экологических и социальных систем и др., разработанные на основе разных методологий в среде AnyLogic. Описания разработок имитационных моделей изложены в работах и книгах по имитационному моделированию [1,2,15-16, 64, 66, 69, 88, 94, 102, 111, 127] и приведены на сайтах www.anylogic.ru и www.runthemodel.ru.

Рис. 2.2. Клиенты компании AnyLogic

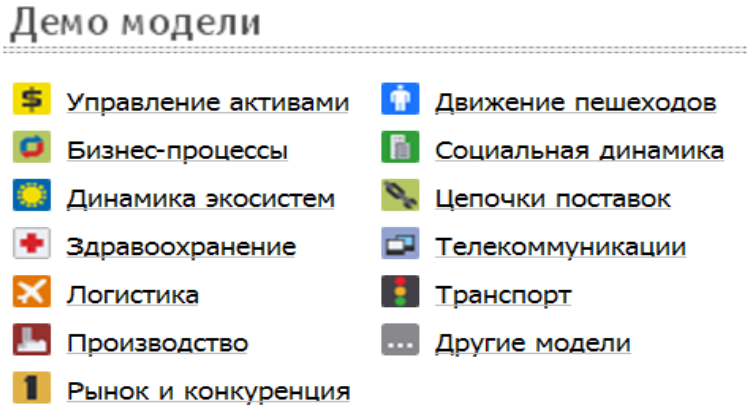


Рис.2.3. Отрасли и сферы использования программного продукта

Рис.2.4

Графическая среда AnyLogic 5 построена по тому же принципу, что и в Rockwell Arena. Моделирующие конструкции располагаются в палитрах (аналог шаблонов в Arena). Для создания модели, как и в Arena, моделирующие конструкции перетаскивают в область модели и соединяют.

2.2. Базовые инструменты для разработки модели в среде AnyLogic

Окно редактора AnyLogic имеет следующие элементы (рис. 2.5):

- Панели инструментов;
- Панель Проекты (дерево всех объектов проекта);
- Панель Палитра;
- Панель Ошибки;
- Окно свойств.
- Окно графического редактора.

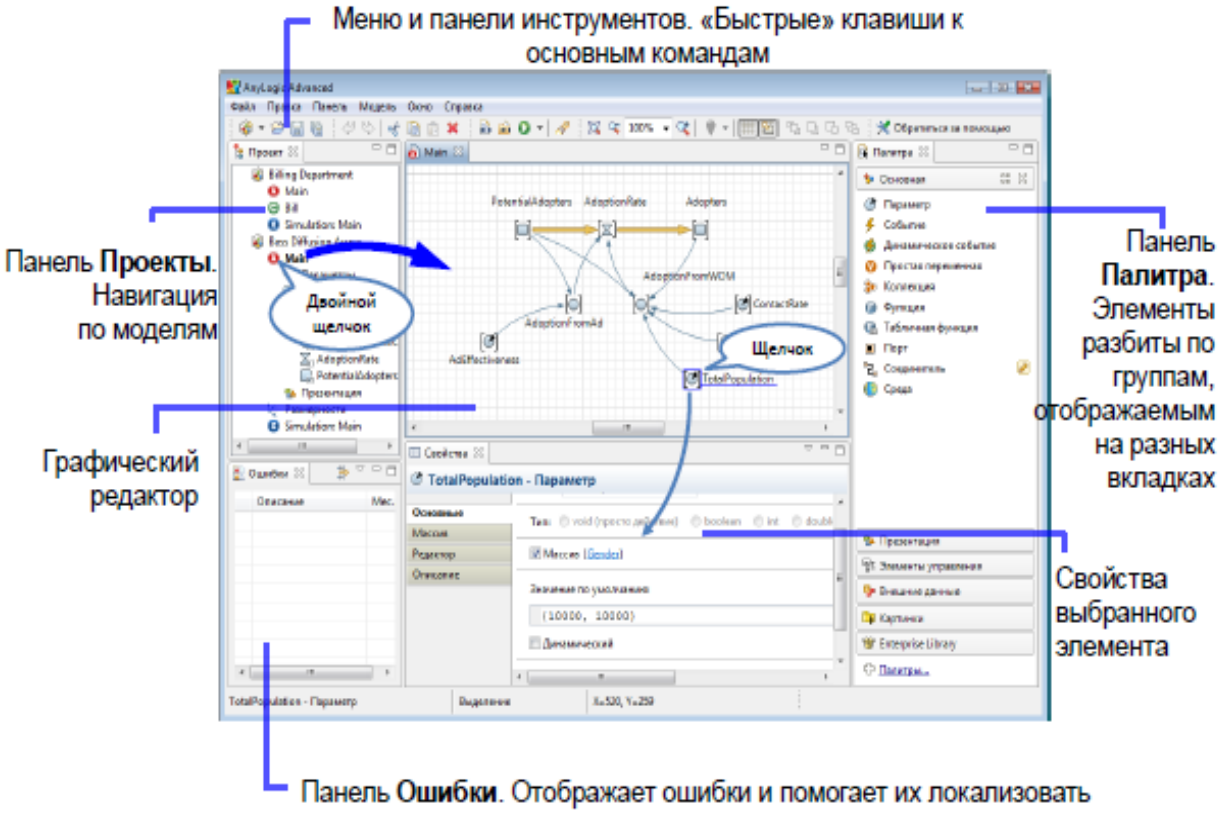


Рис. 2.5. Окно редактора AnyLogic

Панели инструментов обеспечивают быстрый доступ к большинству функций, автоматически подстраиваются под текущий режим, а также могут быть адаптированы пользователем. В AnyLogic существуют следующие панели инструментов:

Стандартная (Создать, Открыть модель, Сохранить модель, Сохранить все модели)

Построение (Построить модель, Построить все, Отладка, Запуск)

Редактирование (Отменить,Повторить, Вырезать, Скопировать, Вставить, Удалить)

Рисование (100%, Отдалить, Масштаб, Приблизить, Отобразить/скрыть сетку)

Панель Проект. Дерево элементов модели.

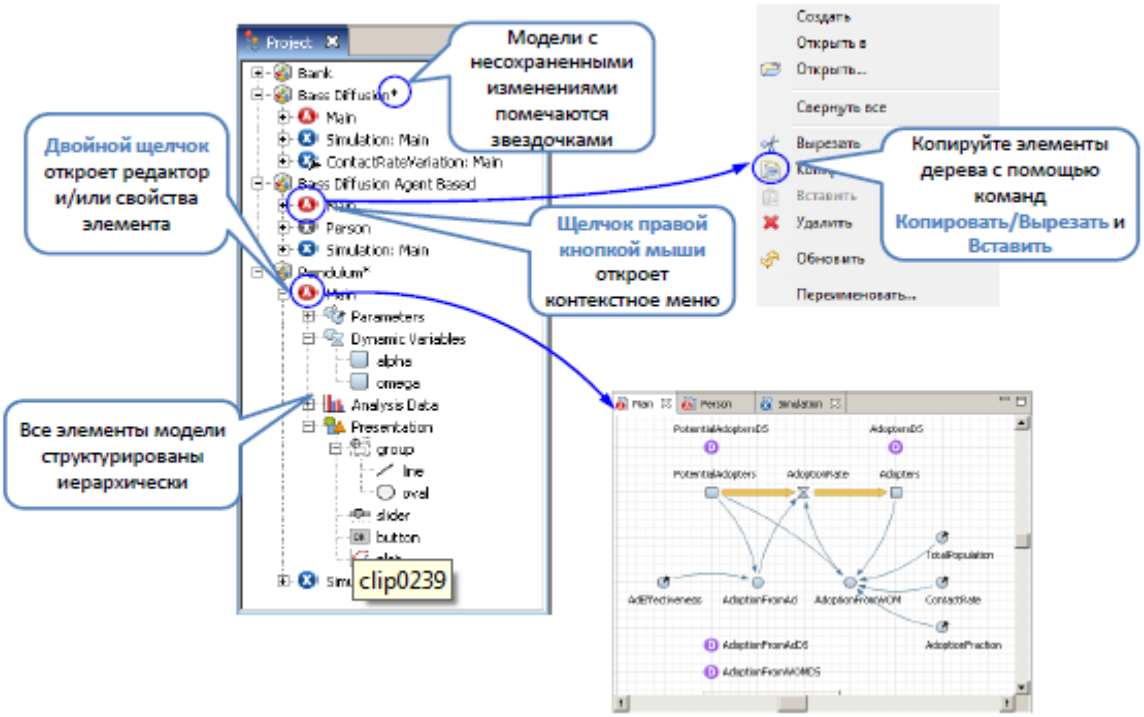


Рис. 2.6. Панель Проекты

Панель Палитры состоит из нескольких вкладок (палитр), каждая из которых содержит элементы, относящиеся к определенной задаче:

• основная содержит основные элементы, с помощью которых Вы можете задать динамику модели, ее структуру и данные;

• системная динамика содержит: элементы диаграммы потоков и накопителей, а также параметр, соединитель и табличную функцию;

• диаграмма состояний содержит блоки диаграмм, позволяющих графически задавать поведение объекта;

• диаграмма действий содержит блоки структурированных блок-схем, позволяющих задавать алгоритмы визуально;

• статистика содержит элементы, используемые для сбора, анализа и отображения результатов моделирования;

• презентация содержит элементы для рисования презентаций: примитивные фигуры, а также элементы управления, для придания презентации интерактивности;

• внешние данные содержит инструменты для работы с базами данных и текстовыми файлами;

• картинки содержит набор картинок наиболее часто моделируемых объектов: человек, грузовик, фура, погрузчик, склад, завод и т. д.

Панель Ошибки находится в нижнем левом углу. Она позволяет быстро с помощью двойного щелчка отобразить ошибку (курсор расположится там, где будет находиться ошибка) и локализовать ее.

Окно свойств. В окне свойств редактора AnyLogic для каждого выделенного элемента модели указываются его свойства (параметры). При выделении какого-либо элемента в любом из окон редактора (в окне структуры, окне поведения, окне анимации или в окне классов) справа появляется окно свойств, показывающее параметры именно этого выделенного элемента. Каждый класс может иметь произвольное количество параметров различных типов (int, real, Boolean и др.). Каждый экземпляр класса может иметь собственные значения параметров, которые могут быть изменены динамически в процессе выполнения модели.

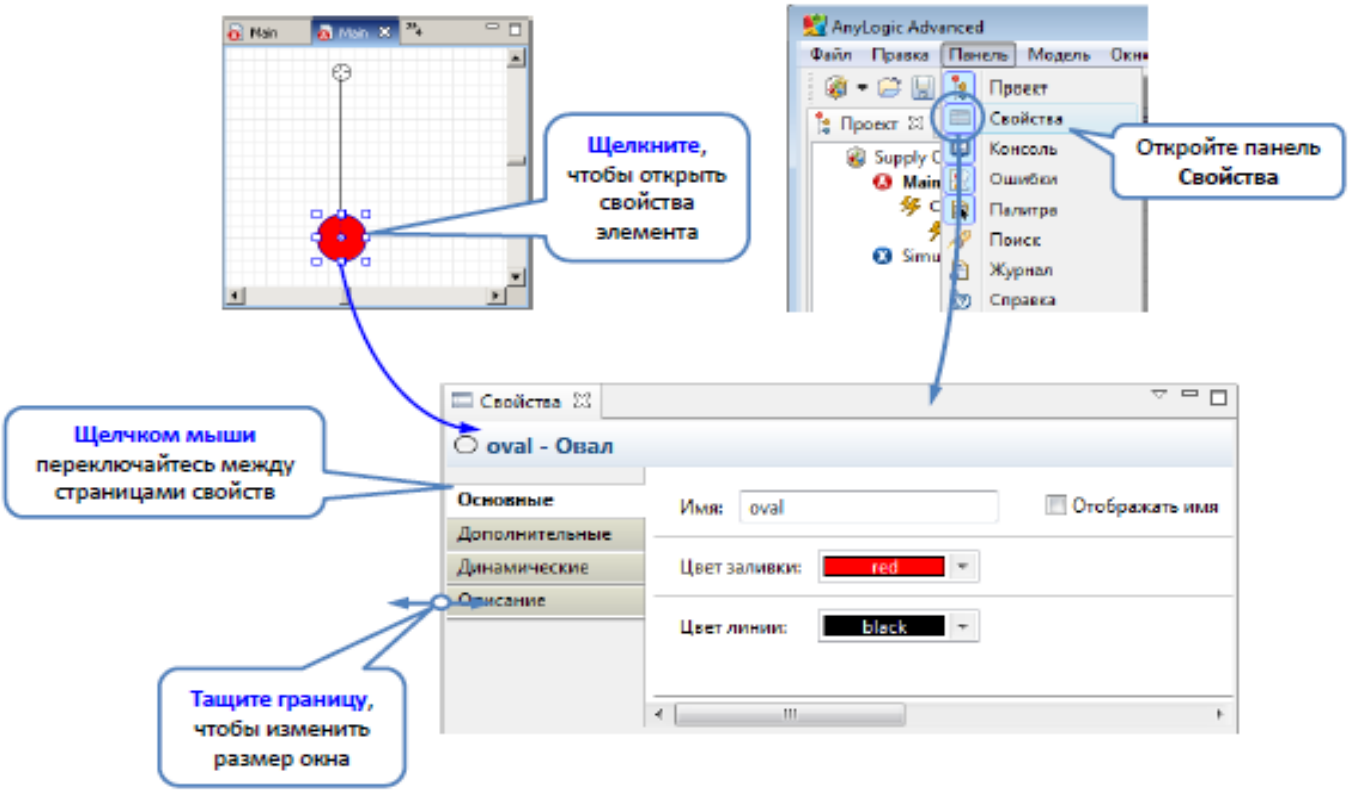


Рис.2.7. Окно свойств

В ***Окне графического редактора*** активного объекта для модели строится двухмерное или трехмерное анимационное представление, которое помогает понять, что происходит с моделью с течением времени. Именно в этом окне визуально представляется имитация поведения моделируемой

системы. Элементы анимационной картинки имеют свои параметры, которые могут быть связаны с переменными и параметрами модели. Изменение переменных модели во времени ведет к изменению графического образа, что позволяет пользователю наглядно представить динамику моделируемой системы с помощью динамически меняющейся графики.

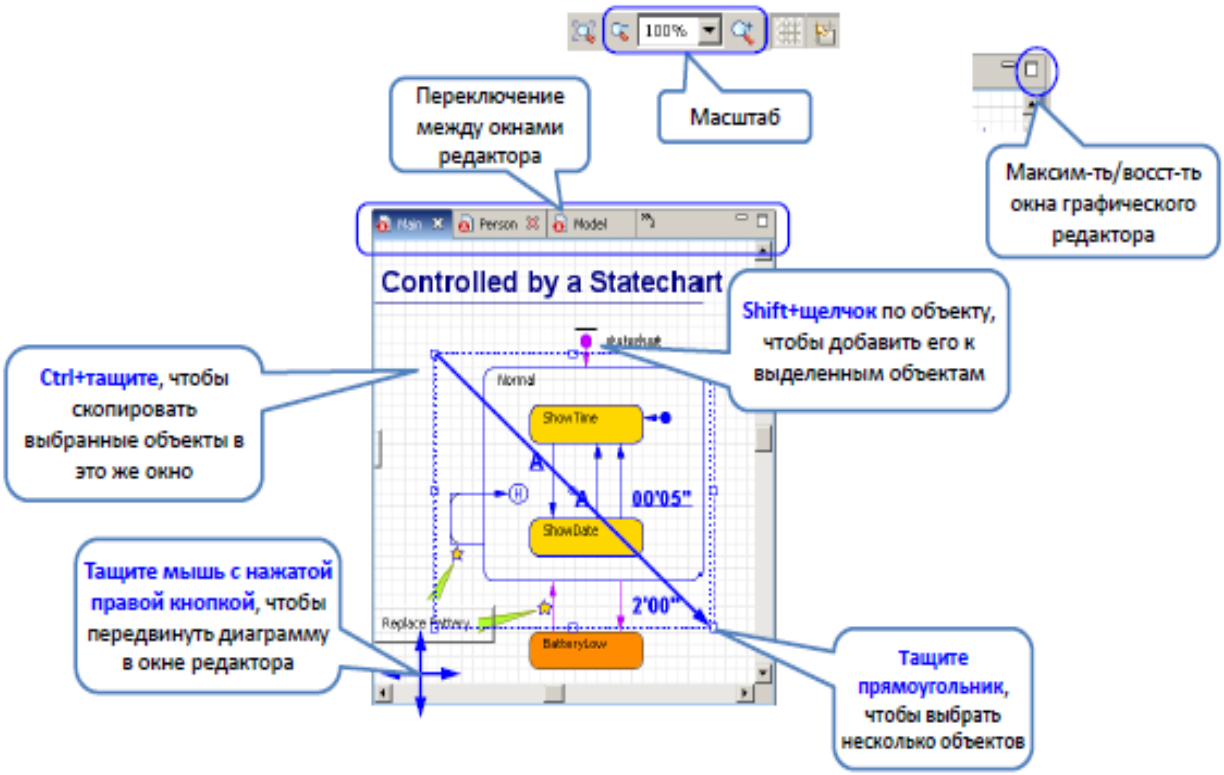


Рис. 2.8. Окно графического редактора

Исполняемая модель. При исполнении модели запустится компилятор, который построит исполняемый код модели в языке Java, оттранслирует его и затем запустит модель на исполнение.

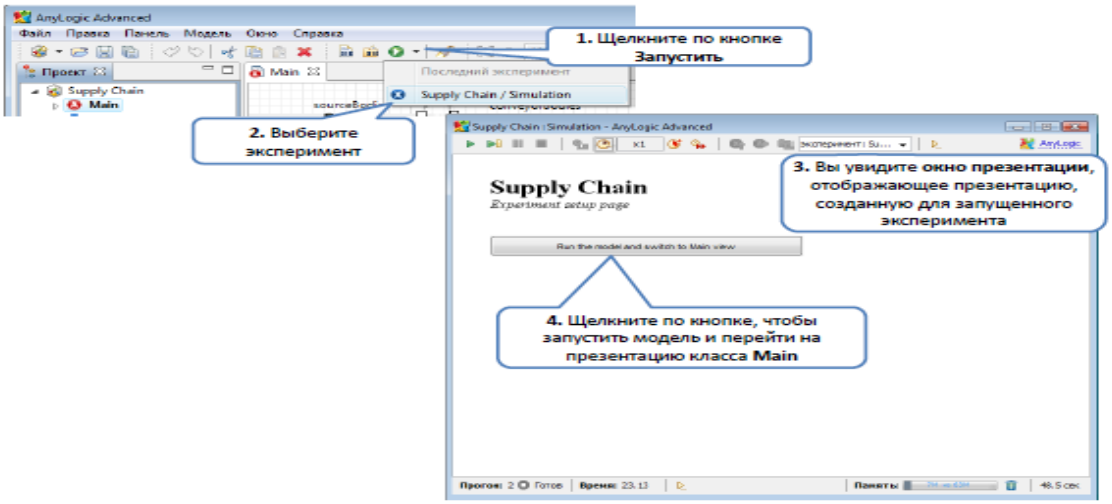


Рис. 2.9. Запуск модели

При этом откроется окно презентации (рис.2.10).

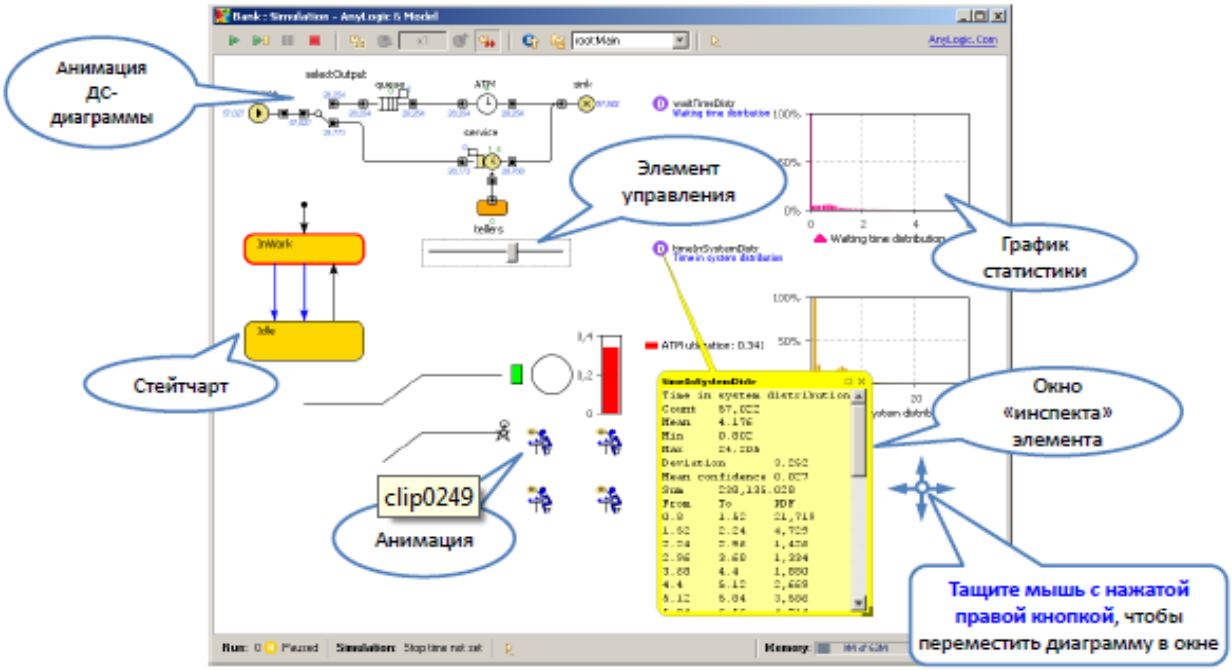


Рис.2.10. Окно презентации

AnyLogic имеет исключительно развитый базовый язык дискретного и смешанного дискретно-непрерывного моделирования, на основе которого разработаны стандартные библиотеки:

- Enterprise Library - конструкции для построения дискретно-событийных моделей
- Dynamic Systems Library (системная динамика)
- Material Flow Library (потоки материалов).

Модель и анимация быстро строятся в стиле drag-and-drop и очень гибко параметризуется. Реализация стандартных объектов открыта для пользователя, их функциональность может быть как угодно расширена, вплоть до создания собственных библиотек. Используя иерархию и регулярные структуры объектов, можно создавать масштабируемые модели.

С помощью библиотеки Enterprise Library пакета AnyLogic можно быстро создавать сложные дискретно-событийные модели, такие как:

- модели производственных процессов;
- модели систем обслуживания (банки, аэропорты и т.д.);
- модели бизнес-процессов с оценкой затрат операций;
- модели логистики и цепочек доставки.

Библиотека объектов Enterprise Library позволяет создавать гибкие модели с наглядной визуализацией моделируемого процесса и возможностью

сбора необходимой статистики, разработана для поддержки дискретно-событийного моделирования в таких областях, как производство, цепи поставок, логистика и здравоохранение. Используя Enterprise Library, можно смоделировать системы реального мира с точки зрения заявок (сделок, клиентов, продуктов, транспортных средств, и т. д.), процессов (последовательности операций, очередей, задержек), и ресурсов. Процессы определены в форме блочной диаграммы.

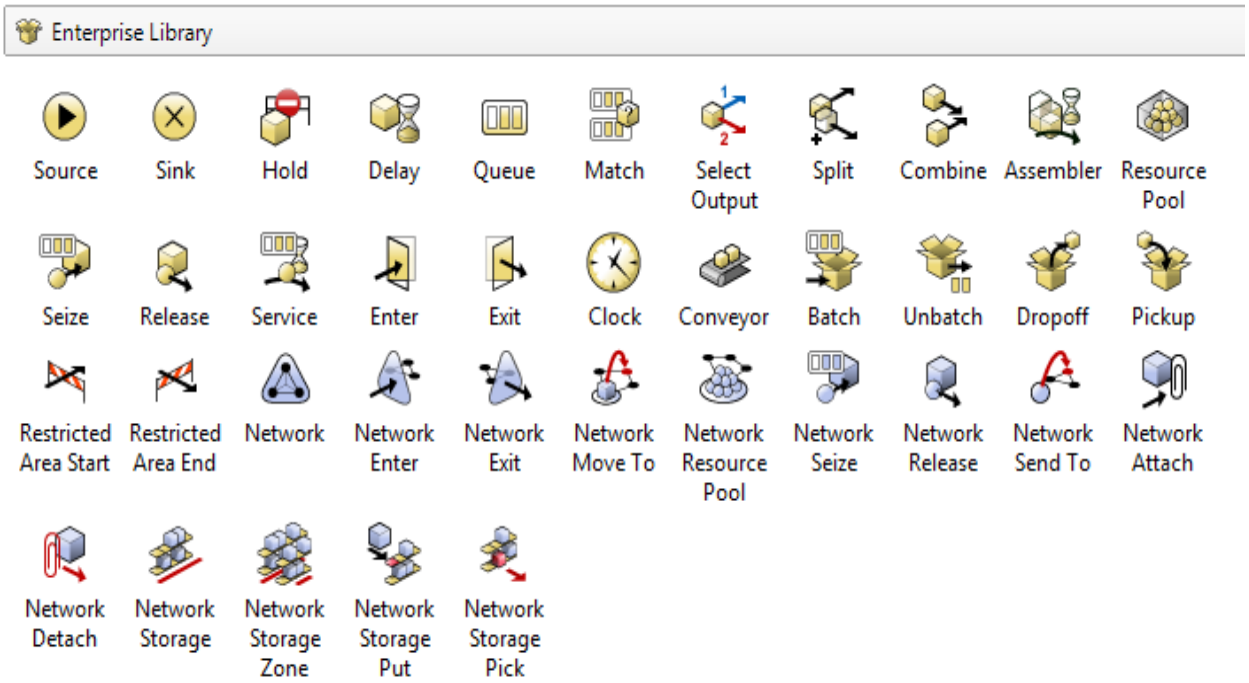


Рис. 2.11. Библиотека Enterprise Library.

– Pedestrian Library создана для моделирования пешеходных потоков в «физической» окружающей среде. Это позволяет создавать модели с большим количеством пешеходного трафика (как станции метро, проверки безопасности, улицы и т.д.). Модели поддерживают учёт статистики плотности движения в различных областях. Это гарантирует приемлемую работу пунктов обслуживания с ограничениями по загруженности, оценивает длину простаивания в определённых областях и обнаруживает потенциальные проблемы с внутренней геометрией – такие как эффект добавления слишком большого числа препятствий – и другими явлениями. В моделях, созданных с помощью Pedestrian Library, пешеходы двигаются непрерывно, реагируя на различные виды препятствий (стены, различные виды областей) так же, как и обычные пешеходы. Пешеходы моделируются как взаимодействующие агенты со сложным поведением. Для быстрого описания потоков пешеходов

Pedestrian Library обеспечивает высокоуровневый интерфейс в виде блочной диаграммы.

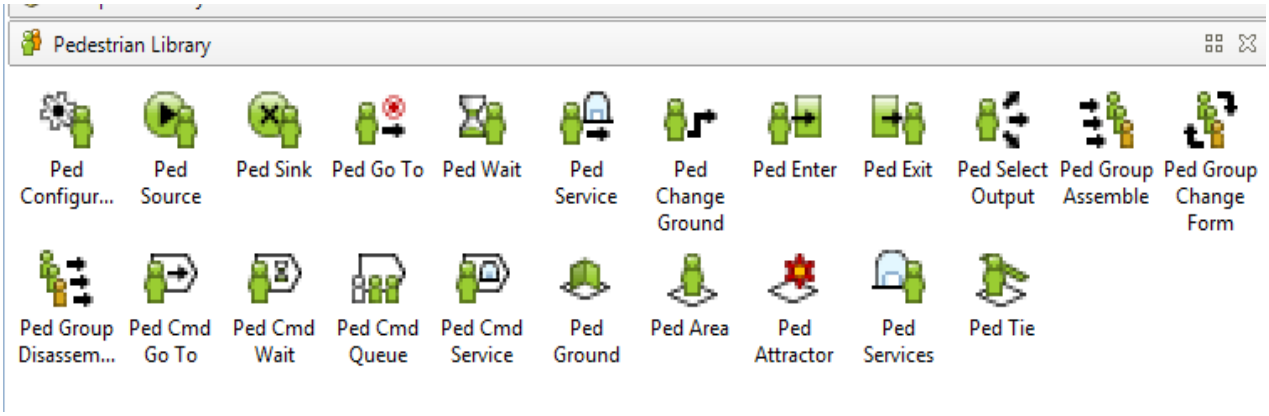


Рис. 2.12. Библиотека Pedestrian Library

– Rail Yard Library поддерживает моделирование, имитацию и визуализацию операций сортировочной станции любой сложности и масштаба. Модели сортировочной станции могут использовать комбинированные методы моделирования (дискретно-событийное и агентное моделирование), связанные с действиями при транспортировке: погрузками и разгрузками, распределением ресурсов, обслуживанием, различными бизнес-процессами.

Детализировать моделирующие конструкции можно, выделив их и изменив параметры, используя панель свойств.

AnyLogic поддерживает подход агентного моделирования, в качестве агентов могут быть: люди – потребители, жители, работники, пациенты, доктора, клиенты, солдаты и др.; транспорт, оборудование – автомобили, краны, самолёты, вагоны, станки и др.; нематериальные вещи – проекты, продукты, инновации, идеи, инвестиции и др.; организации – компании, политические партии, страны и др. Эти объекты в информационной системе AnyLogic могут создаваться и уничтожаться динамически, перемещаться, общаться друг с другом, иметь поведение, знания, цели, стратегию — то есть обладают всеми свойствами агентов.

Агентный подход используют для моделирования социальных систем, в частности, моделируют рынки (агент - потенциальный покупатель), конкуренцию и цепочки поставок (агент - компания), население (агент - семья, житель города или избиратель) и много другое. Только агентный подход позволяет получить представление об общем поведении системы, исходя из предположений о поведении её элементов при отсутствии знания о глобальных законах — то есть в наиболее общем случае.

AnyLogic основан на Java и базируется на платформе Eclipse - современном стандарте для бизнес-приложений. Благодаря Eclipse AnyLogic работает на всех распространенных операционных системах (Windows, Mac, Linux и т.д.).

В редакторе AnyLogic возможно разработать анимацию и интерактивный графический интерфейс модели. Анимация может быть иерархической и поддерживать несколько перспектив. Например, есть возможность определить глобальный взгляд на процесс производства, а также детальные анимации конкретных операций и переключаться между ними. Имеется возможность использования различных видов технологических решений для реализации анимации (рис.2.13).

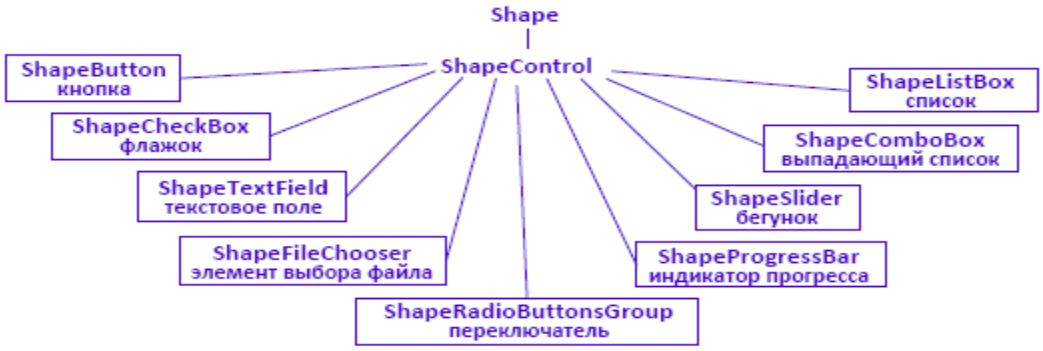


Рис.2.13

В AnyLogic пользователю доступно 35 стандартных теоретических распределений, также можно определить собственные распределения. AnyLogic позволяет строить как стохастические, так и детерминированные модели и проводить анализ результатов моделирования. С моделью могут быть проведены различные эксперименты:

- моделирование (simulation);
- оптимизация (optimization);
- эксперименты Монте-Карло;
- анализ чувствительность;
- эксперименты по сценарию пользователя.

В AnyLogic встроен оптимизатор OptQuest. Комбинируя эвристики, нейронные сети и математическую оптимизацию, OptQuest позволяет находить значения дискретных и непрерывных параметров модели, соответствующие максимуму или минимуму целевой функции, в условиях неопределённости и при наличии ограничений. Для создания отчетов в AnyLogic отведена специальная палитра «Статистика», в которой содержатся конструкции для сбора данных по ходу работы модели. В этой палитре также находятся различные диаграммы, графики и гистограммы.

ГЛАВА 3

СИСТЕМНАЯ ДИНАМИКА

3.1. Методология системной динамики

Системная динамика – парадигма моделирования, где для исследуемой системы строятся графические диаграммы причинных связей и глобальных влияний одних параметров на другие во времени, а затем созданная на основе этих диаграмм модель имитируется на компьютере. Такой вид моделирования помогает понять суть происходящего выявления причинно-следственных связей между объектами и явлениями. Системная динамика применяется для решения производственных, организационных и социально-экономических задач.

Системная динамика как метод имитационного моделирования включает в себя:

- структуризацию объекта;
- построение системной диаграммы объекта, где указываются связи между элементами;
- определение переменных для каждого элемента и темпов их роста;
- принятие гипотез о зависимости каждого темпа роста от переменных и формальное описание этих гипотез;
- процесс оценки введенных параметров с помощью имеющейся статистики.

Для построения и исследования моделей с помощью метода системной динамики в 1958 г. был разработан специальный язык программирования DYNAMO (DYNAmic MOdels).

Модель Форрестера включает следующие элементы:

- *уровни (ресурсы, резервуары)* характеризуют накопленные значения величин внутри системы. Это могут быть заготовки, комплектующие и готовая продукция, товары на складе, товары в пути, банковская наличность, страховые межоперационные запасы, производственные площади, численность работающих, финансовые ресурсы и т.д. Уровни применимы не только к физическим величинам. Например, уровень осведомленности существенен при принятии решения. Уровни удовлетворения, оптимизма и негативных ожиданий влияют на экономическое поведение. Уровни представляют собой значения переменных, накопленные в результате разности между входящими и исходящими

потоками. Каждый уровень описывается его переменной величиной, зависящей от разности входящих и исходящих потоков. Например, производительность транспортной системы может быть выражена количеством товаров в пути (уровень) и константой (запаздывание на время транспортировки). Более сложный пример: решение о найме рабочих может быть связано с уровнями имеющейся рабочей силы, среднего темпа поступления заказов, числа работников, проходящих курс обучения, числа вновь принятых работников, задолженности по невыполненным заказам, уровня запасов, наличия оборудования и материалов. На диаграммах в модели изображаются прямоугольниками;

- *темпы потоков* – скорости изменения уровней, перемещающие содержимое одного уровня к другому. Примерами могут быть потоки материалов, заказов, денежных средств, рабочей силы, оборудования, информации. Изображаются сплошными стрелками;
- *функции решений (винтили),* которые регулируют темпы потока между уровнями. Функция решения может иметь форму простого уравнения, определяющего реакцию потока на состояние одного или двух уровней, и обеспечивают заданный темп потока в связи. Изображаются двумя треугольниками в виде бабочки;
- *линии задержки (запаздывания)* – служат для имитации задержки потоков. Характеризуются параметрами среднего запаздывания и типом неустановившейся реакции. Второй параметр характеризует отклик элемента на изменение входного сигнала. Разные типы линий задержки имеют различный динамический отклик.
- *каналы информации*, соединяющие функции решений (винтили) с уровнями. Изображаются штриховыми стрелками.
- *вспомогательные переменные* – располагаются в каналах информации между уровнями и функциями решений и определяют некоторую функцию и могут иметь размерность уровней либо темпов и в процессе имитации вычисляются совместно с группой переменных, которые имеют такую же размерность. Их обозначают кружком большего размера.
- Постоянные параметры модели обозначают кружочком малого размера.

На рис. 3.1 приведена модель работы отделения банка. Здесь хорошо представлены резервуары (уровни), темпы потоков, вспомогательные переменные и постоянные параметры.

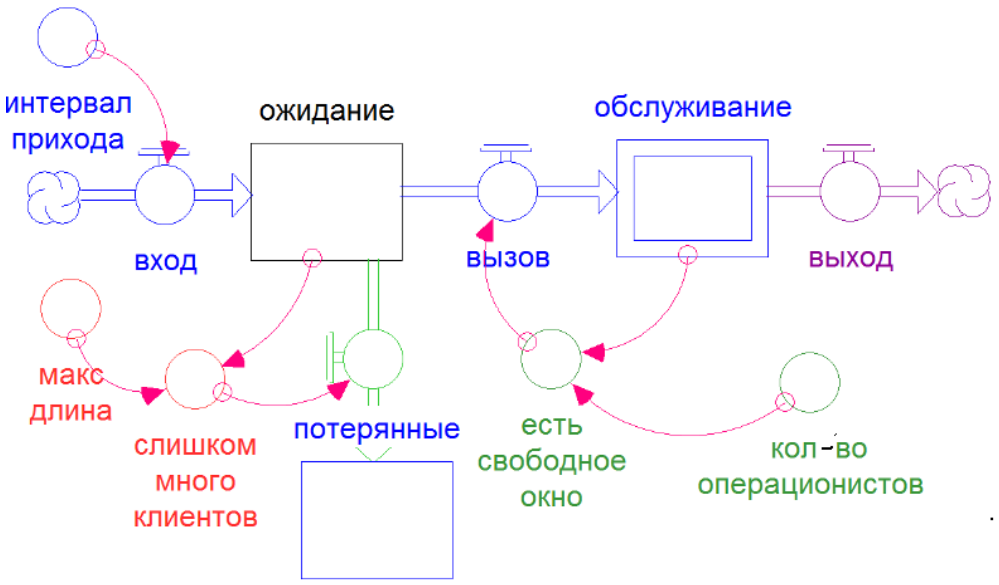
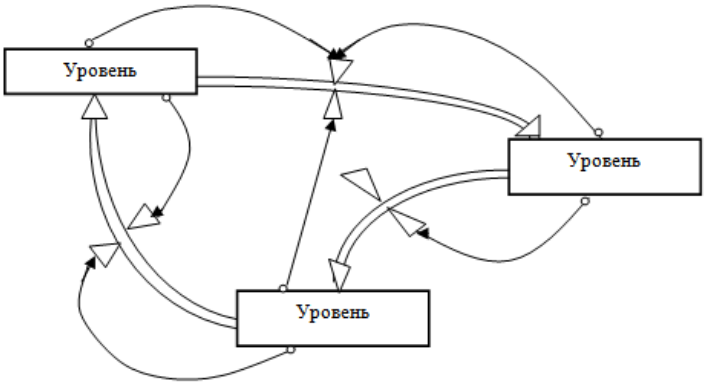


Рис. 3.1. Схема модели системной динамики работы отделения банка

На рисунке 3.2 представлена базовая структура модели Форрестера, которая показывает только одну сеть с элементарной схемой информационных связей между уровнями и темпами. Чтобы отразить деятельность всего промышленного предприятия, необходимо построить несколько взаимосвязанных сетей. Для предприятий можно выделить шесть типов сетей, представляющих различные типы переменных: заказы, материалы, денежные средства, рабочую силу и оборудование, соединенных воедино с помощью сети информации. Любая из этих сетей может быть разбита еще на несколько отдельных частей.



- уравнения темпов, - канал потока, - источники информации.

Рис. 3.2. Базовая структура модели Форрестера

Базовая структура модели дополняется системой уравнений, которые связывают характеристики уровней этой структуры. В основном эта система состоит из уравнений двух типов: уравнения уровней и уравнения темпов.

Построение уравнений уровней. Здесь временная ось разбивается на интервалы времени Δt_{ij} между i-м и j-м моментами времени.

Новые значения уровней рассчитываются на конец интервала, и по ним определяются новые темпы (решения) для следующего интервала $\Delta t_{i(j+1)}$.

Новый уровень = Текущий уровень + Входящий поток - Исходящий поток

$$z_i = z_{i-1} + \Delta t\left(\Delta x_i - \Delta y_i\right),\ i = 1, \ldots, n. \qquad (3.1)$$

В системной динамике такое равенство называется ф*ункциональным уравнением уровня*.

А для вычисления прироста Δx и сокращения Δy уровней z вводят *функциональные уравнения темпов*. Уравнение темпа на выходе можно записать в виде:

$$\Delta y_i / \Delta t = Z_{i-1} / T\ , \qquad (3.2)$$

где Z_{i-1} – величина уровня, отражающего запаздывание в момент времени i; T – константа (среднее время), необходимое для преодоления запаздывания.

К числу достоинств модели относятся: возможность отражать практически любую причинно-следственную связь; простая математическая форма; использование терминологии, синонимичной языку экономики и производства. Из уравнений уровней (3.1) для основных фазовых переменных (так называемых системных уровней) можно записать дифференциальные уравнения типа:

$$\frac{dz_i}{dt} = y_i^{+} - y_i^{-}, \qquad (3.3)$$

где $y^{+}{}_i$ – положительный темп скорости переменной y, включающий в себя все факторы, вызывающие рост переменной y, i - индекс резервуара (уровня);

$y^{-}{}_i$ – отрицательный темп скорости, включающий в себя все факторы, вызывающие убывание переменной y.

Предполагается, что эти темпы расщепляются на произведение функций, зависящих только от "факторов" – комбинаций основных переменных, т.е., в свою очередь, самих являющихся функциями системных уровней:

$$y^{\pm} = g(y_1, y_2, ..., y_n) = f(F_1, F_2, ..., F_k) = f_1(F_1) f_2(F_2) ... f_k(F_k),$$

где $F_j = g_j(y_{i_1},....,y_{i_m})$ — факторы, причем $m = m(j) < n$, $k < n$, n - число уровней, т.е. факторов меньше, чем основных переменных, и каждый фактор зависит не от всех системных уровней, а только от какой-то их части. Это позволяет упростить задачу моделирования.

Этапы построения модели Форрестера:

1. Построение базовой структуры модели в виде специализированного графа;
2. Параметризация графа и построение соответствующей системы уравнений;
3. Описание полученной модели на одном из языков и систем программирования и проведение экспериментов.

Рассмотрим задачу диффузии по Басу. Пусть имеется магазин, в который приходят покупатели (клиенты) и задача состоит в построении модели системной динамики как происходит процесс продажи, и за счет чего происходит изменение продажи.

На рис. 3.3 представлена модель системной динамики. Резервуарами являются потенциальные клиенты и клиенты, темпы прироста клиентов могут изменяться за счет рекламы и устной рекламы.

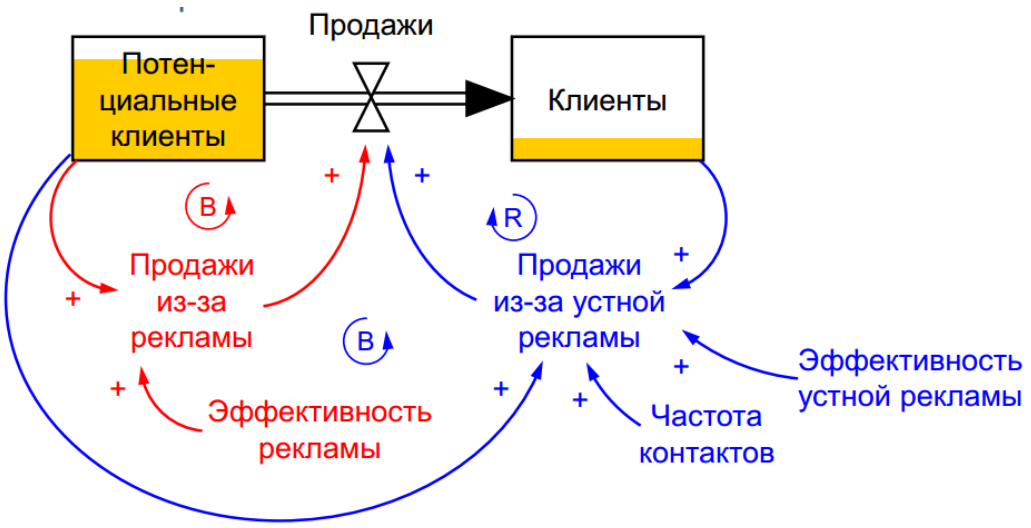


Рис. 3.3. Модель системной динамики

С другой стороны эта графовая модель представляет собой систему дифференциальных уравнений, которая может быть представлена как:

$$\frac{d(ПотенциальныеКлиенты)}{dt} = -(ПродажиИзЗаРекламы + ПродажиИзЗаУстнойРекламы)$$

$$\frac{d(Клиенты)}{dt} = (ПродажиИзЗаРекламы + ПродажиИзЗаУстнойРекламы)$$

$$ПродажиИзЗаРекламы = ПотенциальныеКлиенты \times ЭффективностьРекламы$$

$$ПродажиИзЗаУстнойРекламы = Клиенты \times ЧастотаКонтактов \times \frac{ПотенциальныеКлиенты}{ПотенциальныеКлиенты + Клиенты} \times ЭффективностьУстнойРекламы$$

Модель Форрестера – простая, ясная и полезная модель – иллюстрирует интересный графовый подход к моделированию сложных нелинейных систем. Задуманная как учебный пример применения метода системной динамики, она стала образцом для последующих работ, привлекла внимание к проблеме мировой динамики, дала толчок к проведению других исследований, что привело к появлению целого направления, получившего название глобального моделирования. Модели такого рода применяются для прогнозирования экологии, демографии, финансового анализа, экономики и социологии.

Программными средствами, поддерживающимися подход системной динамики (в частности модель Форрестера), являются: AnyLogic, Arena, SimBioSys, eM-Plant, Tecnomatix Plant Simulation, SimuLab, VenSim, PowerSim, Stella, Ithink, ModelMaker и др. Для построения моделей в них используются графическое представление зависимостей переменных, в виде так называемых «stock and flow diagrams».

Модели системной динамики хорошо реализуются в среде AnyLogic, так как AnyLogic является инструментом имитационного моделирования нового поколения, который основан на результатах, полученных в теории моделирования и в информационных технологиях за последнее десятилетие. AnyLogic поддерживает разработку и моделирование систем обратной связи (диаграммы потоков и накопителей , правила решений, включая массивы переменных). Итак, с помощью AnyLogic можно:

- определять потоковые переменные одну за другой
- использовать авто-заполнение при работе с формулами
- создавать копии переменных для лучшей читаемости модели
- использовать табличные функции со ступенчатой, линейной, сплайновой интерполяцией
- определять поведение функции за пределами допустимой области
- определять поддиапозоны и подразмерности
- объявлять переменные-массивы с заданной размерностью

- задать различные уравнения для различных наборов элементов массива
- использовать как специальные инструменты Системной динамики, так и возможности языка Java.

Для разработки сложных аналитических и имитационных моделей используют среды визуального программирования и моделирования MATLAB, MvStudium, Anylogic, Arena и др. Программный пакет AnyLogic позволяет анализировать и моделировать системы динамического типа. Разные средства анимации, спецификации и анализа результатов, имеющиеся в Anylogic, позволяют строить модели, имитирующие или анимирующие практически любой физический процесс (а также строить и многие другие модели), выполнять анализ моделей на компьютере без проведения реальных экспериментов и самостоятельных сложных вычислений.

Пакет AnyLogic является средой визуального программирования и моделирования, поэтому освоение методов моделирования в этой системе широко доступно непрофессионалам. Эта система может быть освоена также специалистами не математического профиля.

3.2. Моделирование задачи системной динамики «Ассимиляция этносов»

Рассмотрим три основных этноса, проживающих на территории Башкортостана (русские, башкиры, татары). Их расселение на территории РБ показано на рисунке 3.4:

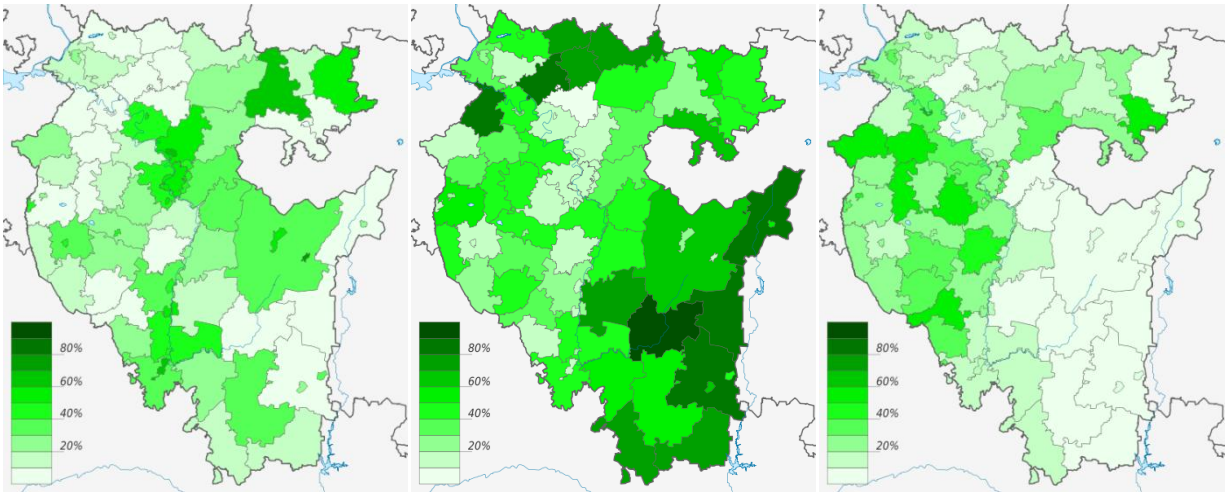


Рис.3.4. Национальный состав

Для исследования их развития используем математическую модель, которая описывает процесс развития численности трех этносов, проживающих на одной территории:

$$\frac{dN_R(t)}{dt} = [\alpha_R(t) - \beta_R(t)]N_R - \gamma_R(t)N_R{}^2 - \delta_{RB}(t)N_R N_B + \delta_{RT}(t)N_R N_T + \xi N_R$$

$$\frac{dN_B(t)}{dt} = [\alpha_B(t) - \beta_B(t)]N_B - \gamma_R(t)N_B{}^2 + \delta_{RB}(t)N_R N_B - \delta_{TB}(t)N_B N_T + \xi N_B \qquad (3.4)$$

$$\frac{dN_T(t)}{dt} = [\alpha_T(t) - \beta_T(t)]N_T - \gamma_T(t)N_T{}^2 + \delta_{TB}(t)N_T N_B - \delta_{RT}(t)N_R N_T + \xi N_T$$

N_i- количество этносов на единицу площади, $\alpha_i(t)$ - коэффициенты рождаемости, $\beta_i(t)$ - коэффициенты естественной смерти, $\gamma_i(t)$- коэффициенты, описывающие убыль этноса из-за агрессивности, ξ - коэффициент миграции этносов в изучаемую территорию, $\delta_{ij}(t)$ - коэффициенты ассимиляции, описывающие изменение численности этносов за счет смешанных браков ($i \neq j$). i=R,B,T ,где R- русские, B- башкиры, T- татары. Знаки перед слагаемыми определяют убыль или прибыль этноса за счет взаимодействия.

Представленная модель наиболее актуальна для исследования ассимиляций, развития этносов в больших мегаполисах, в частности в г. Уфе.

Построение модели. Для создания нашей модели воспользуемся библиотекой системной динамики.

Для создания нового проекта:

1) запускаем Anylogic;
2) создаем новый проект «Assim».

И увидим: в центре рабочей области– структурную диаграмму, слева – окно Проект, справа– Свойства.

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

Накопители представляют собой такие объекты реального мира, в которых сосредоточиваются некоторые ресурсы; их значения изменяются непрерывно. Потоки – это активные компоненты системы, они изменяют значения накопителей. В свою очередь, накопители системы определяют значения потоков. Вспомогательные переменные помогают преобразовывать одни числовые значения в другие.

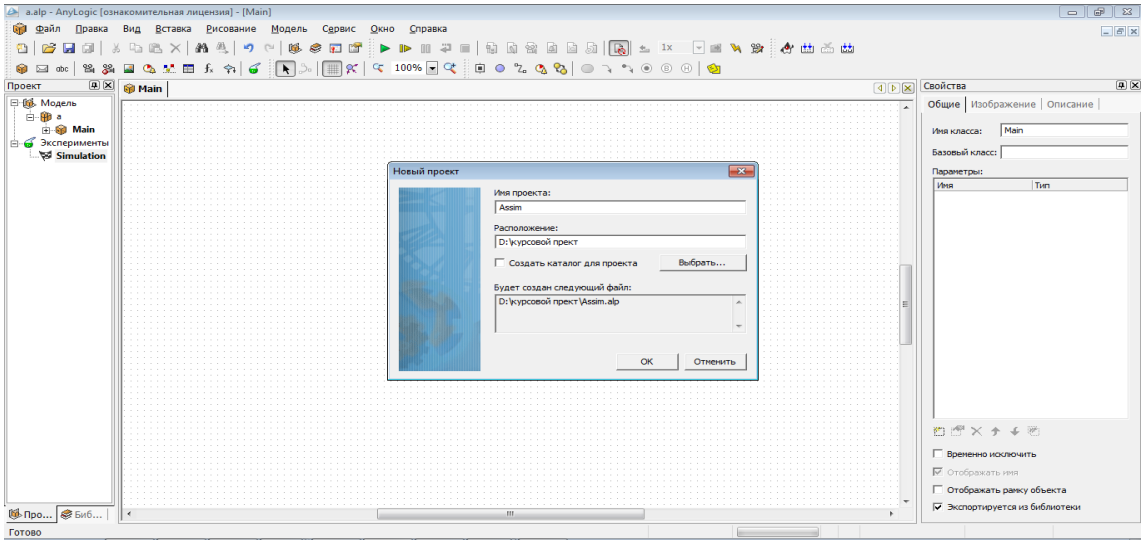


Рис.3.5. Создание проекта

При создании потоковой диаграммы выявим переменные, которые накапливают значения с течением времени. В нашей модели численности этносов продукта являются накопителями, а процесс ассимиляции – потоком.

Для начала определим константы. Константа в AnyLogic задается параметром , который находится в окне Свойств. Вводим все константы в открывающееся окно Параметры (рис.3.6).

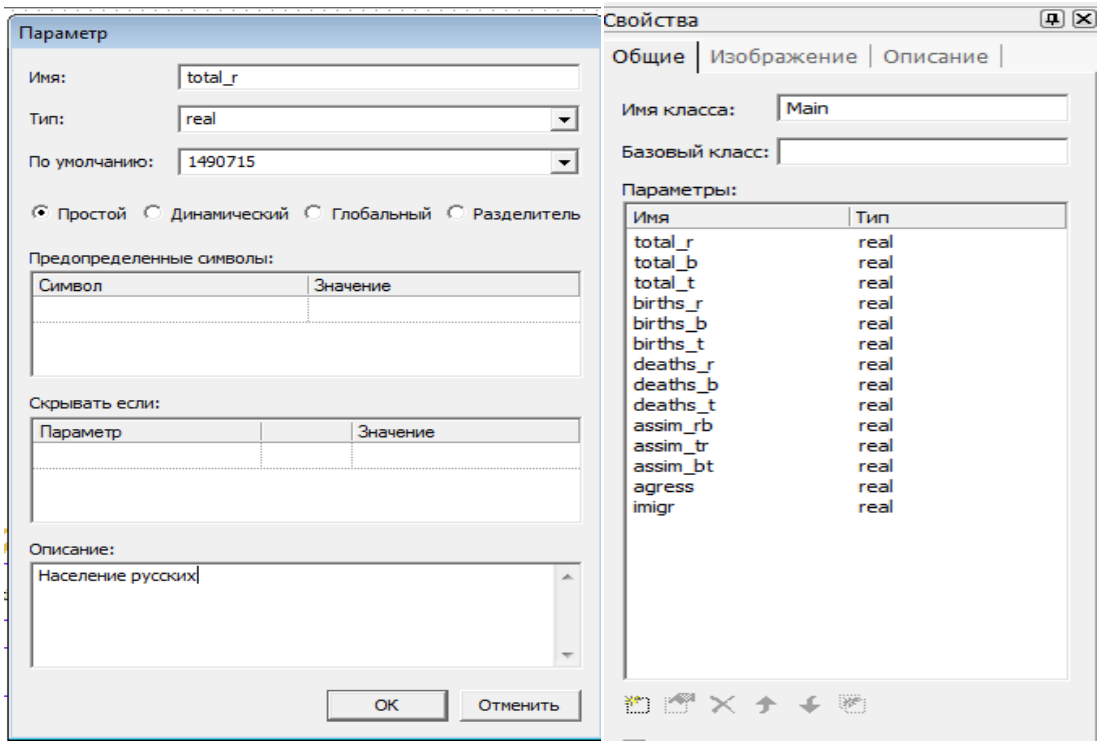


Рис.3.6. Ввод констант

Согласно уравнению (3.4) создаем такие константы: total_r, total_b, total_t – население русских, башкир, татар соответственно, births_r – коэффициент рождаемости русских, births_b – коэффициент рождаемости башкир, births_t – коэффициент рождаемости татар, deaths_r, deaths_b, deaths_t – коэффициенты смертности русских, башкир, assim_rb, assim_rt, assim_tb – коэффициенты ассимиляции трех этносов, agress – коэффициенты, описывающие убыль этноса из-за агрессивности, imigr – коэффициент иммиграции населения.

Теперь создадим переменные : total_russians, total_bashkir, total_tatar (рис. 3.7).

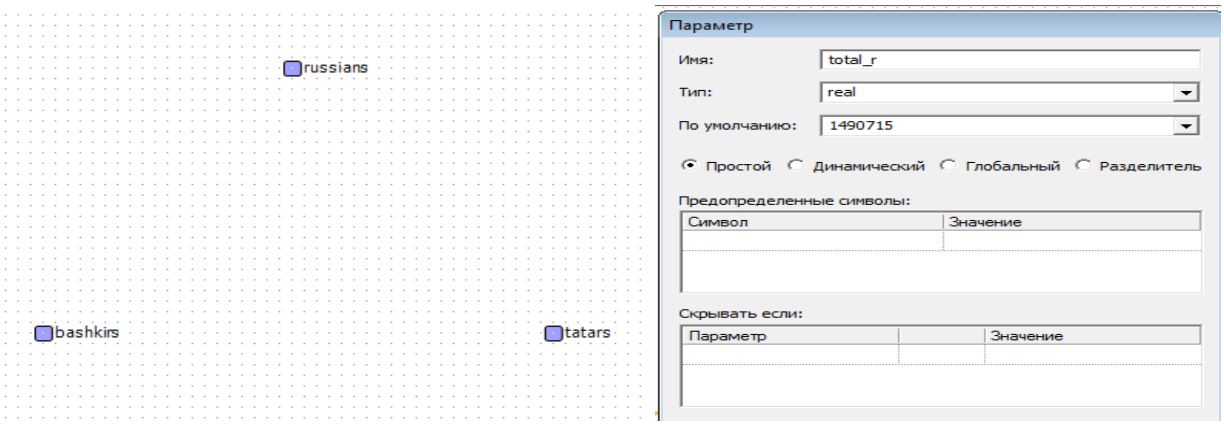


Рис. 3.7 Создание переменных

Зададим формулы для потоков . В окне свойств записываем формулу для каждого потока. Например, формулы потоков births_russians, deaths_russians agress_russians, ass_R_T показаны на рисунке 3.8.

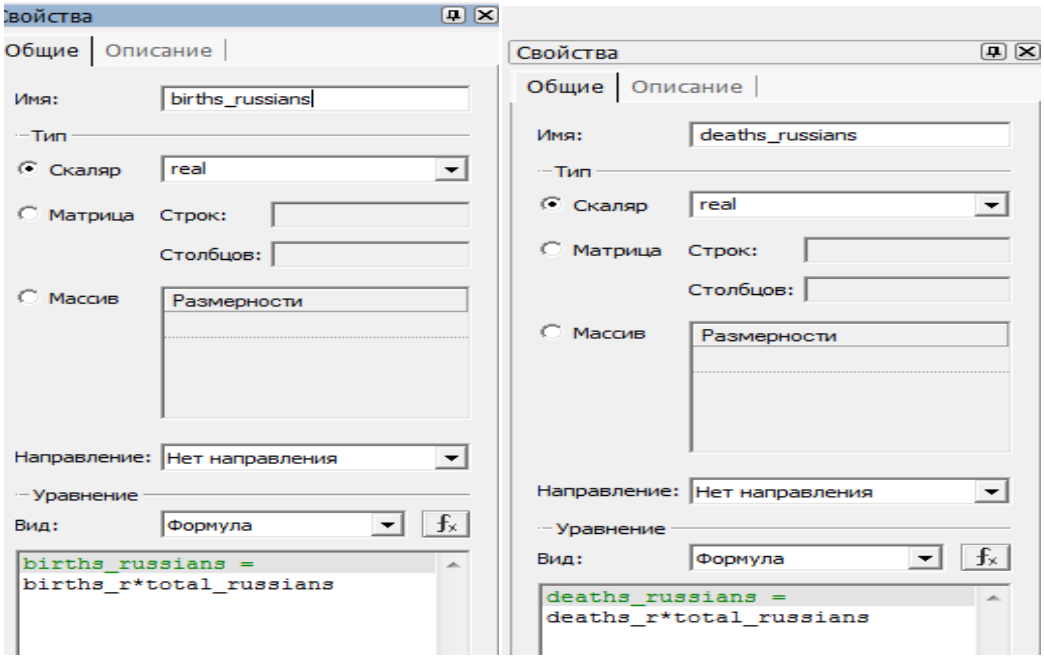


Рис.3.8

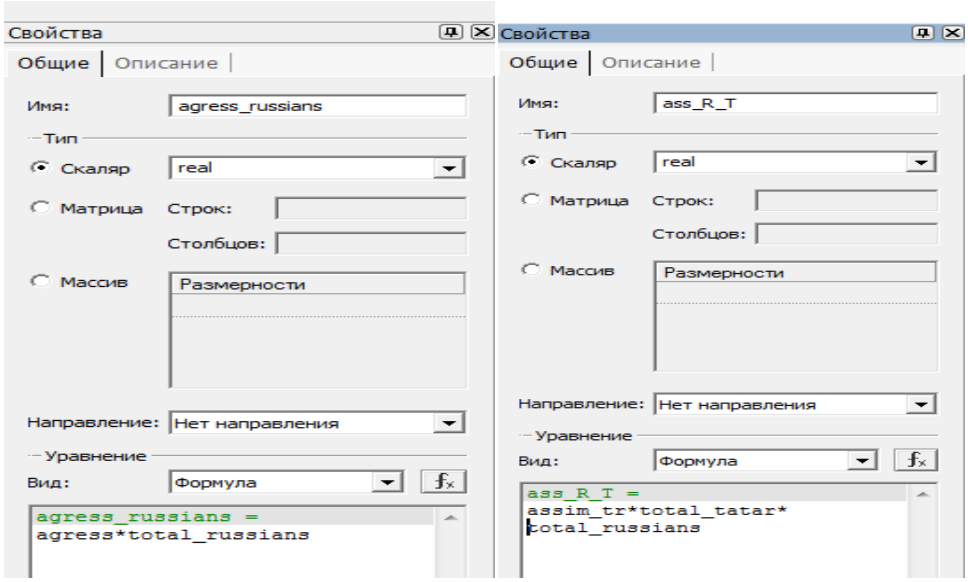


Рис. 3.9. Создание потоков

•

Для накопителей RUSSIANS, BASHKIRS, TATARS, задаем формулы аналогично переменным, только в вид уравнения выбираем Интеграл или накопитель. Формулы для этих накопителей показаны на рисунке 3.10.

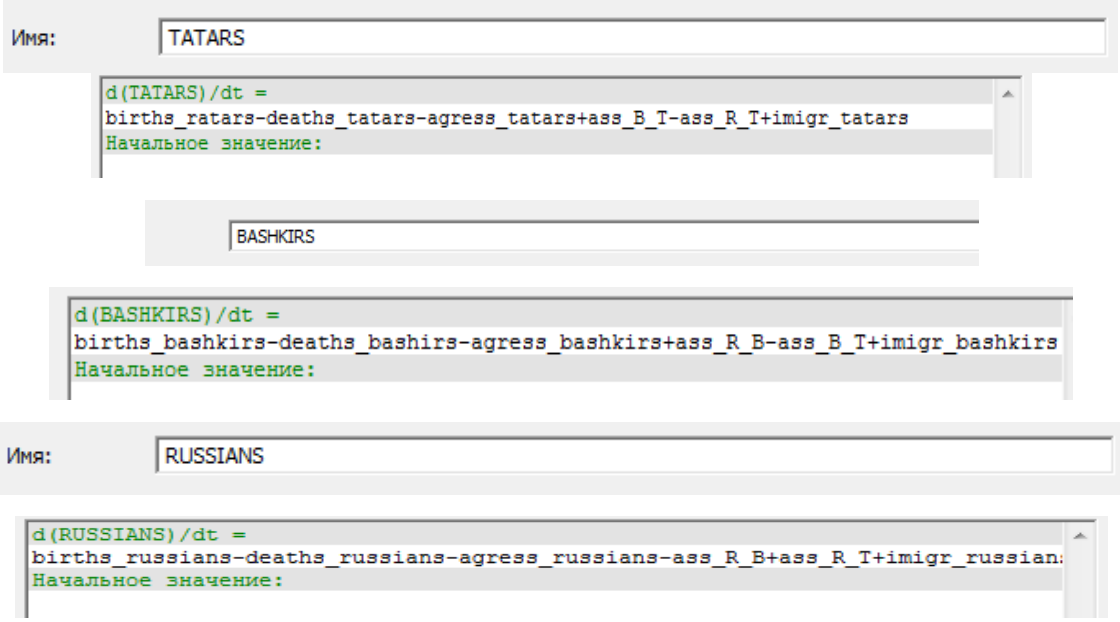


Рис. 3.10. Формулы накопителей

Для того чтобы увидеть взаимосвязи всех переменных, необходимо активировать панели инструментов кнопку Зависимости переменных .

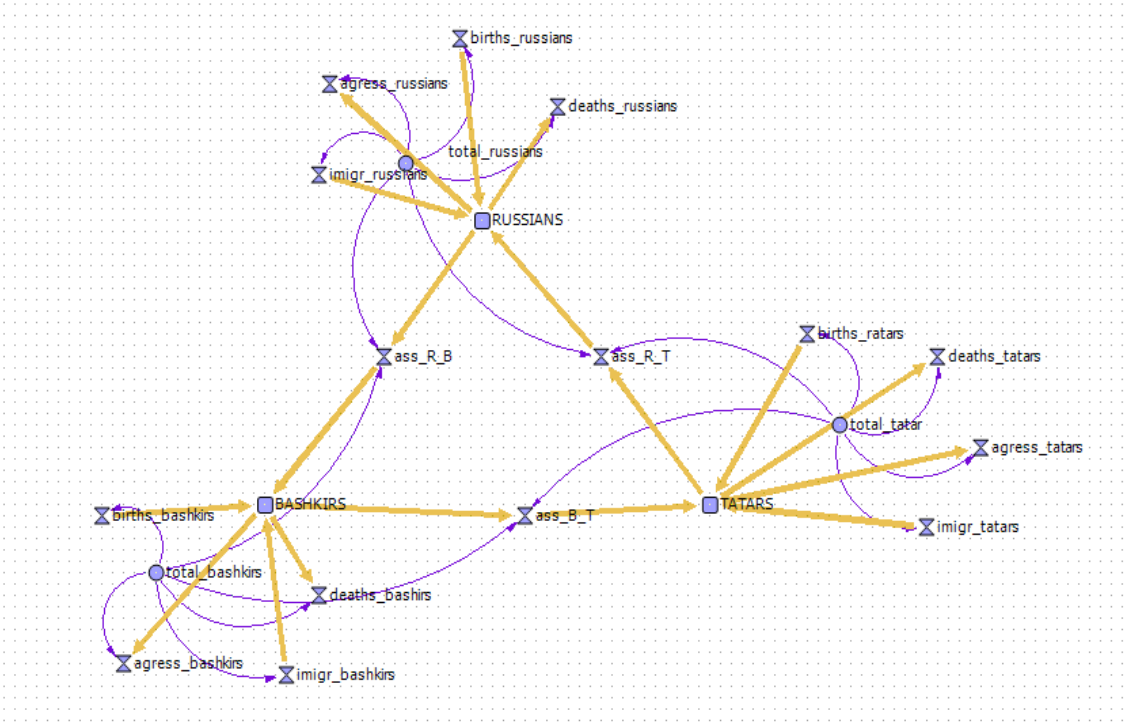


Рис. 3.11. Причинно-следственные зависимости модели

Для того чтобы внести и отредактировать уже имеющиеся уравнения, необходимо в окне Проект развернуть элемент Main и выбрать пункт Код. Все наши созданные уравнения показаны на рисунке 3.12.

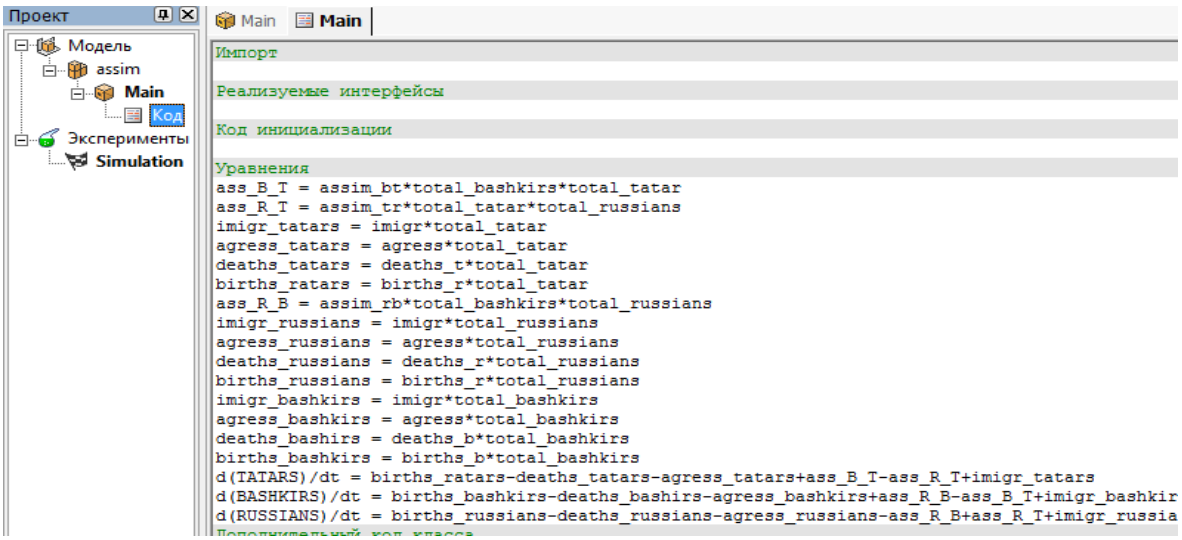


Рис.3.12. Все формулы

Зададим остановку модели после 100 единиц модельного времени. Для этого:

1) в окне Проект выбираем Simulation;
2) на вкладке Дополнительные окна Свойства, устанавливаем флажок Стоп по времени;
3) в поле редактирования справа вводим 100.

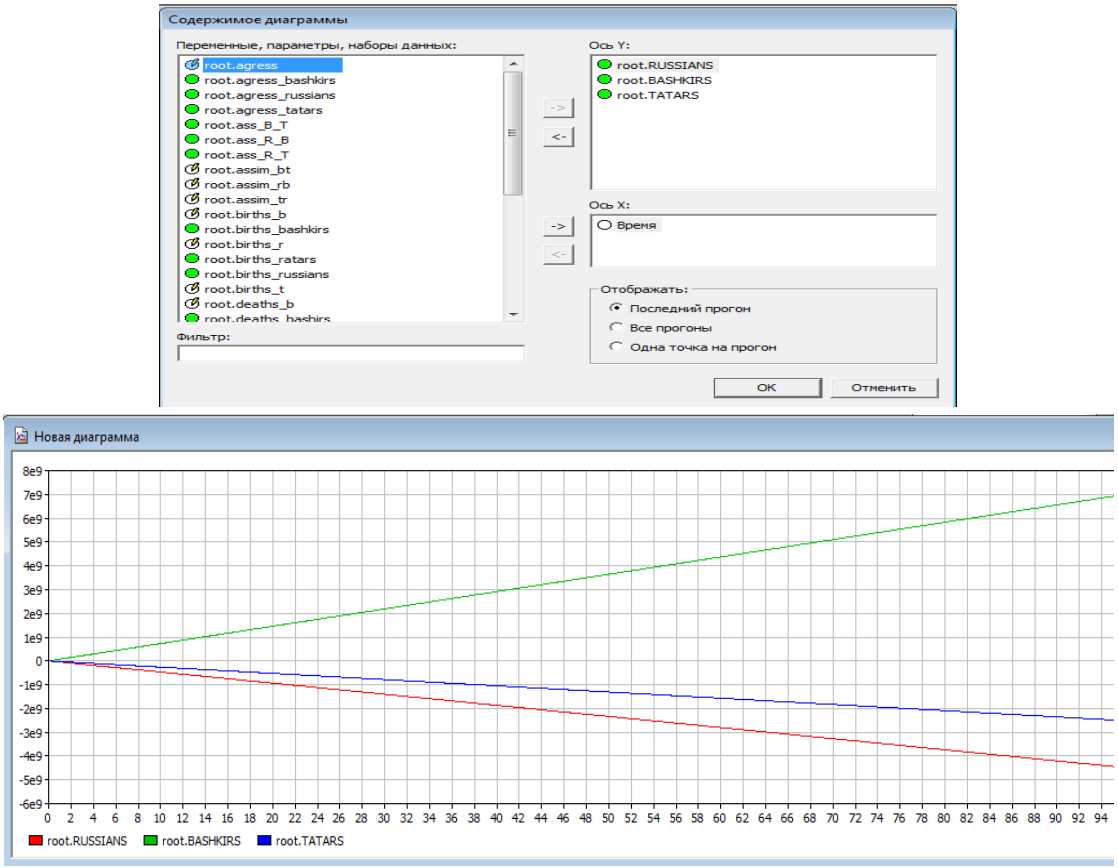
Запуск модели:

1) построим проект с помощью кнопки панели инструментов Построить ;
2) после того как проект успешно построен, запускаем модель. Для этого нажимаем кнопку Запустить .

Щелкнув по панели инструментов Дерево объектов запускаем его. В дереве отображаются переменные и текущие значения.

Можно исследовать модель,как изменятся рост населения этносов с помощью диаграмм. Для создания диаграммы:

1) в панели инструментов выбираем Новая Диаграмма;
2) выбираем переменные, которые должны быть отображены на диаграмме;
3) щелкнув правой кнопкой мыши по окну диаграммы, выбираем Содержимое диаграммы;
4) и получаем диаграмму, показанную на рисунке 3.13.



- Рис. 3.13. Диаграмма

Таким образом, с помощью диаграммы состояний сможем изучить динамику изменения этносов.

ГЛАВА 4

МОДЕЛИРОВАНИЕ ДИНАМИЧЕСКИХ СИСТЕМ

Под *динамической системой*, как мы уже рассматривали, понимается любой объект, процесс или явление, для которого однозначно определено понятие состояния как совокупности некоторых величин и задан закон, который описывает изменение начального состояния с течением времени, двигающуюся в пространстве и изменяющуюся во времени. Динамическими объектами могут быть механические, производственные, физические, химические, биологические объекты, вычислительные процессы, системы и др.

Исследование реальных систем сводится к изучению математических моделей, совершенствование и развитие которых определяются анализом экспериментальных и теоретических результатов при их сопоставлении. В связи с этим под динамической системой мы будем понимать ее математическую модель.

Математическая модель динамической системы считается заданной, если введены параметры (координаты) системы, определяющие однозначно ее состояние и указан закон эволюции.

Реальным физическим системам, моделируемым математическим понятием «динамической системы», приписывается важное свойство *детерминированности*: зная состояние системы в начальный момент времени, мы можем однозначно предсказать все ее дальнейшее поведение. Фазовым пространством динамической системы называется множество всех ее возможных состояний в фиксированный момент времени. Обычно состояние системы задается некоторым набором чисел (фазовых координат) и представляет собой область в многомерном пространстве.

Решение реальных физических задач методами имитационного моделирования с использованием ЭВМ исследователями, студентами, учащимися позволяет проводить требующие долгой подготовки и материальных ресурсов практически любые эксперименты, в том числе связанные с физическими процессами. Моделирование позволяет исследовать суть сложных физических процессов и явлений с помощью вычислительных экспериментов с моделью системы, а не с реальной системой.

Компьютерные модели физических процессов и систем чаще всего разрабатывают средствами процедурных языков программирования: Visual Basic, Turbo Pascal, Microsoft Fortran PowerStation, результаты моделирования визуализируются в графических пакетах Surfer, Microcal ORIGIN,

Grapher и др. Для разработки моделей несложных систем и объектов также используют системы компьютерной математики Maple, MathCAD, MATLAB и др., в которых результаты моделирования представляют в виде графического и частично анимационного изображения.

Разработать компьютерные модели физических процессов и систем можно с помощью систем компьютерной математики, технического моделирования[48-52, 89]. В данной главе мы рассмотрим технологии построения компьютерных моделей для изучения физических (динамических) процессов с помощью системы имитационного моделирования AnyLogic [66, 94].

4.1. Колебания маятника Фуко

Маятник Фуко представляет собой следующую систему: над центром вращающейся горизонтальной платформы подвешен маятник на длинном подвесе, такой маятник, отклонённый от равновесного положения, совершает колебания в плоскости, неподвижной в инерциальной системе отсчёта. Задача состоит в построении компьютерной и анимационной модели в среде моделирования.

Динамическая система с непрерывным временем, называемая маятником Фуко, записывается в виде:

$$\frac{d\upsilon_x}{dt} = 2\upsilon_y\omega + \omega^2 x - g\frac{x}{L},$$

$$\frac{d\upsilon_y}{dt} = -2\upsilon_x\omega + \omega^2 y - g\frac{y}{L},$$

$$\frac{dx}{dt} = \upsilon_x, \qquad \frac{dy}{dt} = \upsilon_y.$$

где ω – относительная частота, L – длина подвеса.

Система AnyLogic может быть использована для моделирования непрерывных динамических процессов и систем.

Приведем построение компьютерной модели маятника Фуко средствами системы AnyLogic. При построении модели воспользуемся подходом системной динамики, реализованной в данной среде.

Шаг 1. Создадим новый проект.

Щелкните мышью по кнопке панели инструментов “***Создать***”. Появится диалоговое окно “Новый Проект”. Щелкните мышью по кнопке “Выбрать…” и выберите директорию, в которой будем хранить файлы проекта. Укажем имя нового проекта. Например, Foucault's pendulum, в поле редактирования “Имя проекта”. Щелкните мышью по кнопке “OK”. Новый проект создан.

Шаг 2. Зададим данные внутри модели.

Нужно решить, что использовать – параметры или переменные.

Параметр обычно используется для задания статических характеристик модели и обычно хранит одно и то же значение в течение всего "прогона" модели; это значение изменяется пользователем только в какие-то определенные моменты времени при желании изменить характеристики модели. Параметры задаются на странице “Общие” панели “Свойства”.

Зададим величину W (относительная частота). Для этого присвоим параметру имя w, значение по умолчанию, например, 0.04. (рис.4.1)

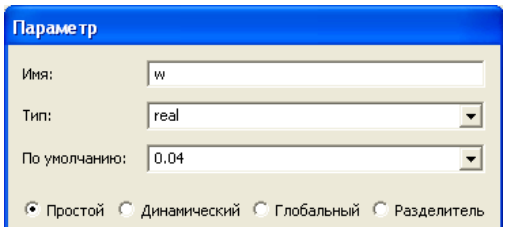


Рис.4.1.

Зададим величину L (длина подвеса). Для этого присвоим параметру имя L, значение по умолчанию - 50.

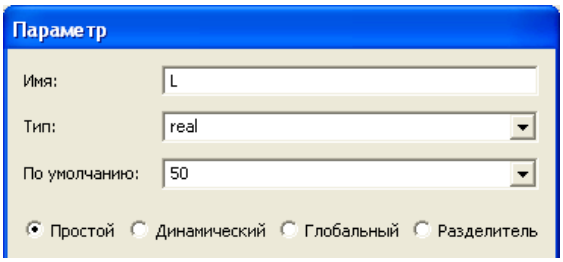


Рис.4.2.

Зададим величину g (гравитационная постоянная). Для этого присвоим параметру имя g, значение по умолчанию – 9.8.

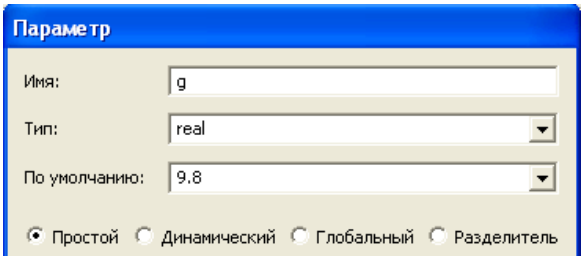


Рис.4.3.

Все заданные параметры можно увидеть в панели “Свойства”.

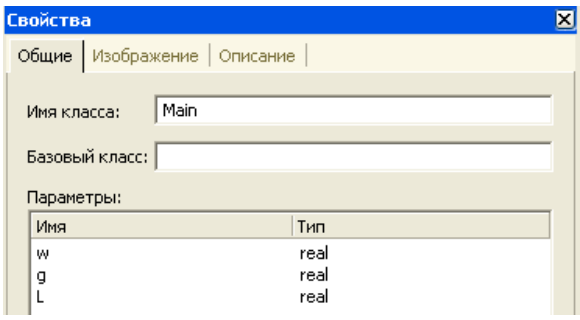


Рис.4.4.

Переменные обычно используются для моделирования изменяющихся характеристик объекта или для хранения результатов работы модели.

Зададим 4 переменных – x, y, vx и vy.

Чтобы задать переменным дифференциальные уравнения, необходимо выбрать на странице “Общие” панели “Свойства” в поле “вид” – “интеграл или накопитель”.

В AnyLogic уравнения записываются в строку.

Для переменной x: d(x)/dt = vx, начальное значение – 1.

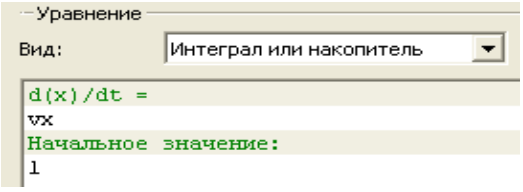


Рис.4.5.

Для переменной y: d(y)/dt = vy, начальное значение – 1.

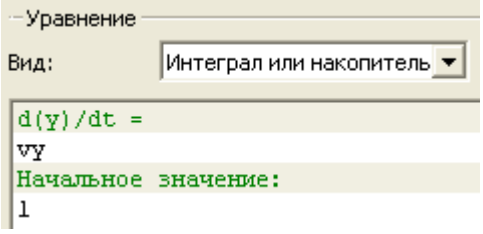


Рис.4.6.

Для переменной vx: d(vx)/dt = 2*vy*w+w^2*x-g*x/L, начальное значение – 0 (по умолчанию в AnyLogic, если оставить начальное значение пустым, это также будет 0).

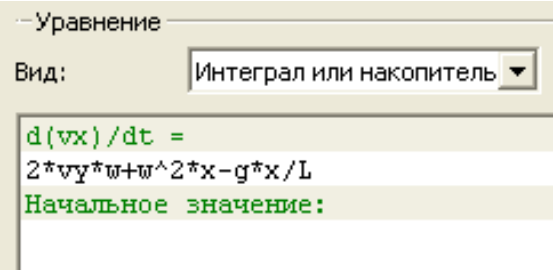


Рис.4.7.

Для переменной vy: d(vy)/dt = -2*vx*w+w^2*y-g*y/L, начальное значение – 0.

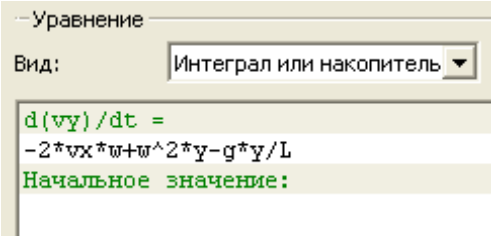


Рис.4.8.

Все параметры и переменные видны на презентации модели, и можно изменять их значения во время работы модели либо программно из кода модели, либо с помощью элементов управления.

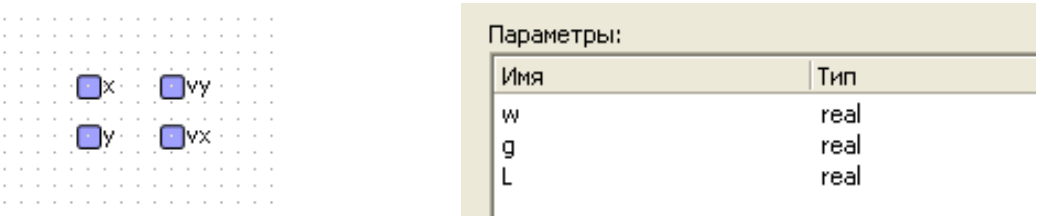


Рис.4.9.

Также добавим в модель график – xYChart из библиотеки Business Graphic Library, который понадобится для анимации движения конца маятника Фуко.

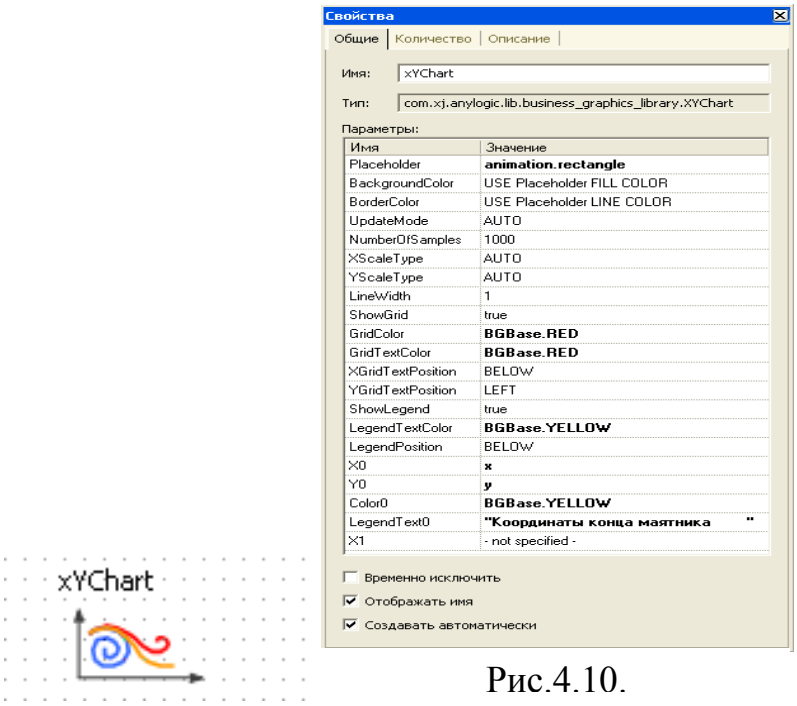


Рис.4.10.

Шаг. 3. Создание анимации.

Для создания анимации прежде всего в меню “Вставка” нужно выбрать пункт “Новая анимация”. Укажем размеры анимационной области, например, 1110x600. Это можно сделать в окне “Свойства” в полях “Ширина” и “Высота”.

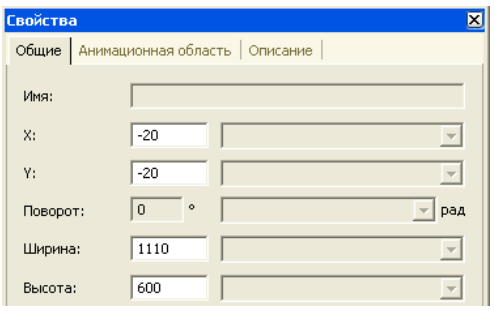


Рис.4.11.

Построим прямоугольник (rectangle) нужного размера, в котором будет показываться траектория движения конца маятника. В свойствах графика xYChart укажем следующее: Placeholder – animation.rectangle (это позволит увидеть в прямоугольнике строящуюся траекторию), GridColor и GridTextColor – Red (оси x и y, а также значения координат на осях будут подписаны красным цветом), LegendTextColor – Yellow (траектория будет показана желтым), XO и YO – x и y соответственно, Color0 – Yellow, LegendText0 – “Координаты конца маятника ” (ниже графика будут показаны текущие координаты).

Модель готова к запуску, но нужно сделать удобное изменение параметров во время ее выполнения.

Шаг. 4. Изменение параметров во время выполнения модели.

В анимации создадим 2 бегунка (sliders). Они будут отвечать за изменение параметров w и L во время выполнения модели.

Бегунок (slider) – элемент управления, позволяющий пользователю графически выбирать число из заданного диапазона значений путем перетаскивания рукоятки (рис.4.12).

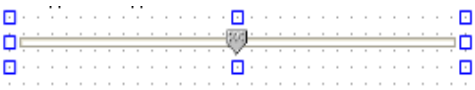


Рис.4.12.

Бегунки обычно используются для изменения значений численных переменных и параметров во время выполнения модели. Чтобы добавить бегунок, необходимо перетащить элемент “Бегунок” в нужное место графического редактора.

На странице “Бегунок” панели “Свойства” в поле “Переменная” введем численную переменную или параметр. В результате - присваивание этой переменной текущее значение этого бегунка.

Создадим бегунок для параметра L. В полях “минимум” и “максимум” введем минимальное и максимальное значение, которое может принять переменная при изменении бегунка. Например, 1-100.

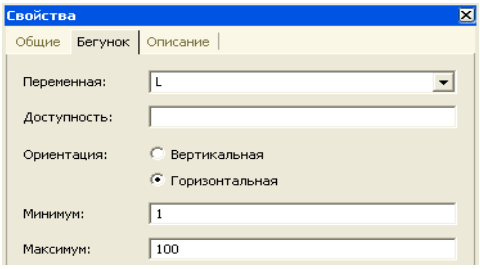


Рис.4.13.

Создадим бегунок для параметра w. В полях “минимум” и “максимум” введем минимальное и максимальное значение, которое может принять переменная при изменении бегунка. Например, 0-1 (включая сотые).

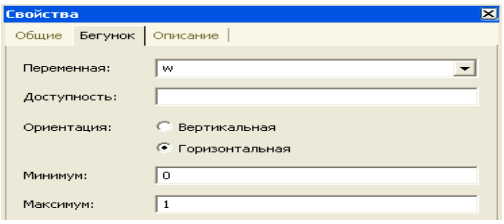
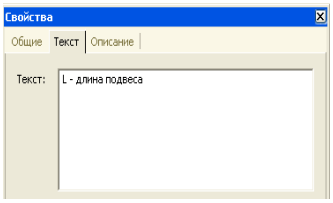


Рис.4.14.

Теперь, чтобы не путаться и знать, какое значение задано бегунком, создадим несколько текстовых полей.

Текстовое поле (иди поле ввода) является простейшим текстовым элементом управления, позволяющим пользователю вводить небольшие объемы текста. Чтобы добавить текстовое поле, необходимо перетащите элемент “Поле ввода” в то место графического редактора, где вы хотите его нарисовать.

На странице “Текст” панели “Свойства” в верхнем поле ввода “Текст” можно ввести любой текст, в нашем случае – подпишем бегунки (L и w).



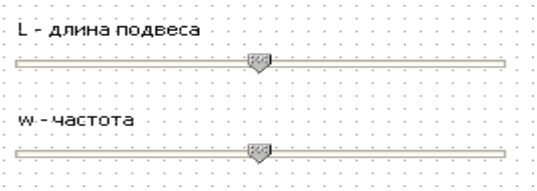


Рис.4.15.

В нижнем поле (“Динамический текст”) можно выбрать параметр или переменную, значение которой и будет показывать поле ввода. При изменении этого параметра (например, бегунком), значение текстового поля тоже изменится. Для одного бегунка выберем параметр L, для другого – w.

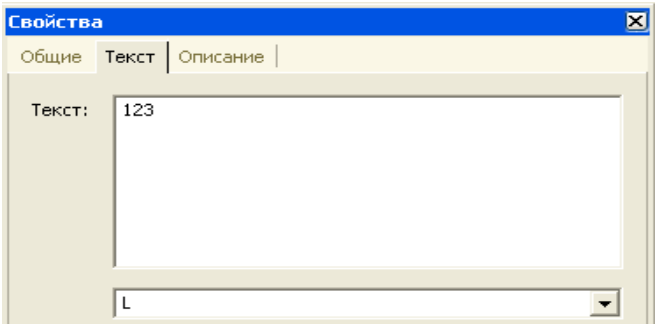


Рис.4.16

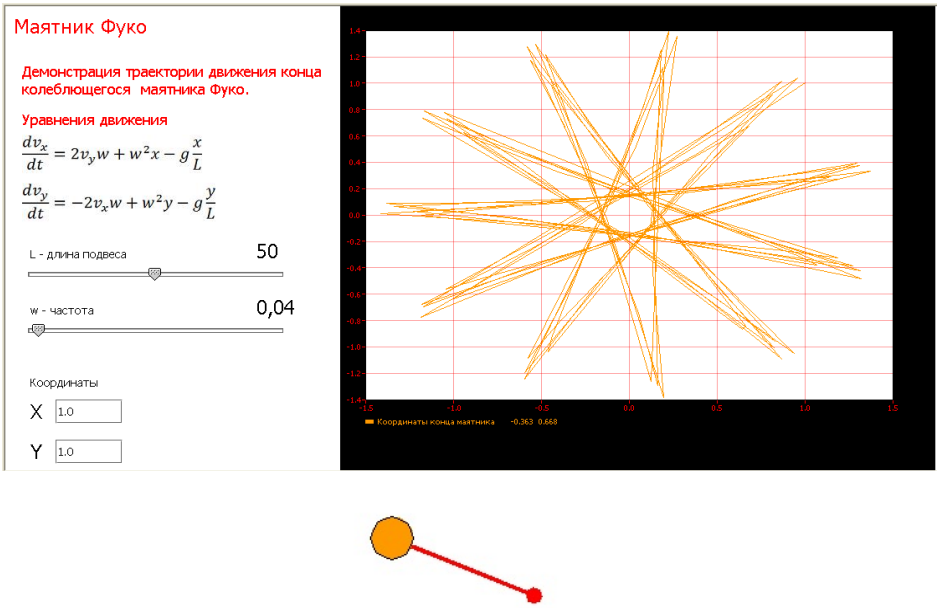


Рис.4.17. Интерфейс компьютерной установки «Маятник Фуко»

Можно также с помощью текстовых полей подписать модель (“Маятник Фуко”), указать используемые уравнения и т.д. В данной модели можно динамически изменять такие параметры, как длину подвеса и частоту, при этом график сразу же изменится. Также можно указать нужные координаты X и Y. Текущие координаты конца маятника указываются под графиком траектории движения.

4.2. Пространственный осциллятор

Постановка задачи. Рассмотрим динамическую систему в виде заряженного пространственного осциллятора находящегося в однородном магнитном поле. Задача состоит в создании компьютерной модели и изучении на ней траекторий движения осциллятора.

Дифференциальные уравнения .

$$\frac{dv_x}{dt} = -w_x^2 x + w_H v_y \qquad \frac{dx}{dt} = v_x$$

$$\frac{dv_y}{dt} = -w_y^2 y + w_H v_x \qquad \frac{dy}{dt} = v_z$$
$$\frac{dv_z}{dt} = -w_z^2 z \qquad \frac{dz}{dt} = v_y$$

Где $\omega_H = \dfrac{eH}{mc}$, m –масса, e – заряд движущегося осциллятора, H – напряженность магнитного поля.

Создание модели.

Для создания новой модели в меню “Файл” выберем “Создать - Модель” (рис.4.18).

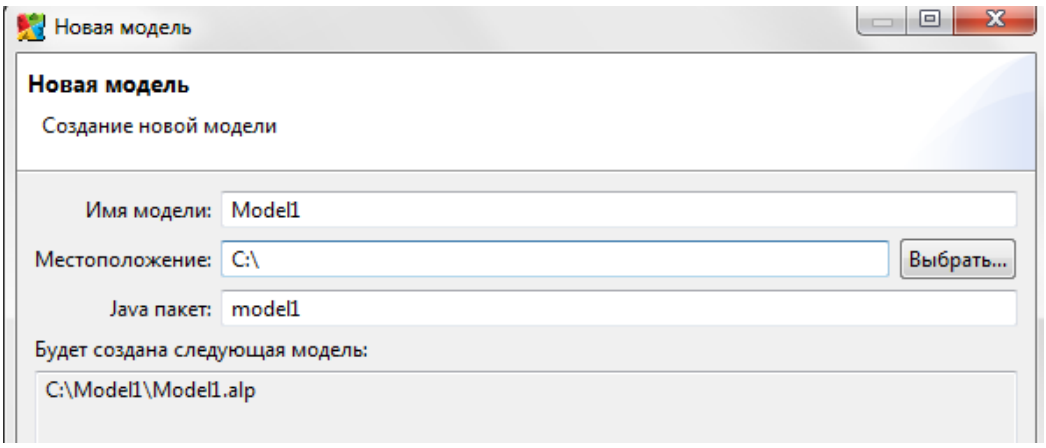


Рис. 4.18 Создание модели

В поле имя модели введем “Пространственный осциллятор”, укажем нужное вам местоположение для хранения модели.

Для решения задачи понадобятся элементы библиотеки “Системная динамика” – накопители и параметры. Перенесем 6 накопителей и 2 параметра на модель (рис.4.19).

Переименуем их, введем дифференциальные уравнения и значения.

Для первого накопителя (x): начальное значение 1, режим задания уравнения – произвольный, уравнение: vx (рис.4.20).

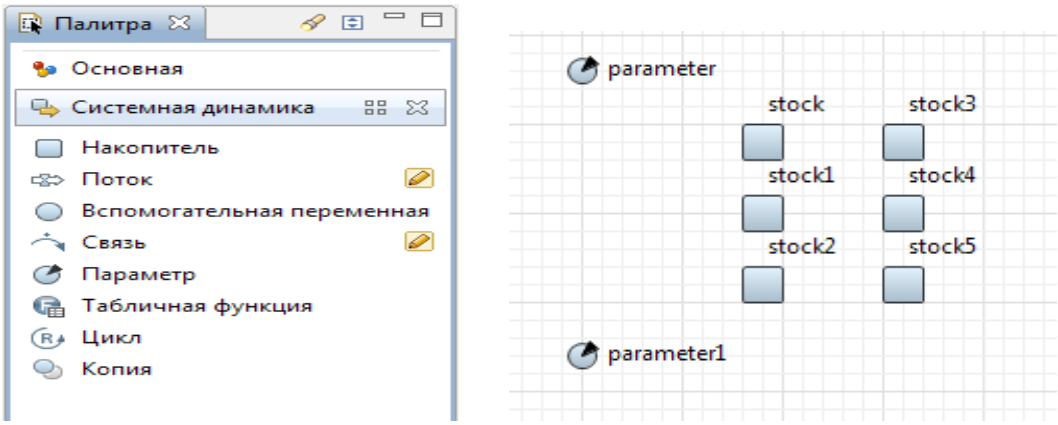


Рис.4.19. Накопители и параметры

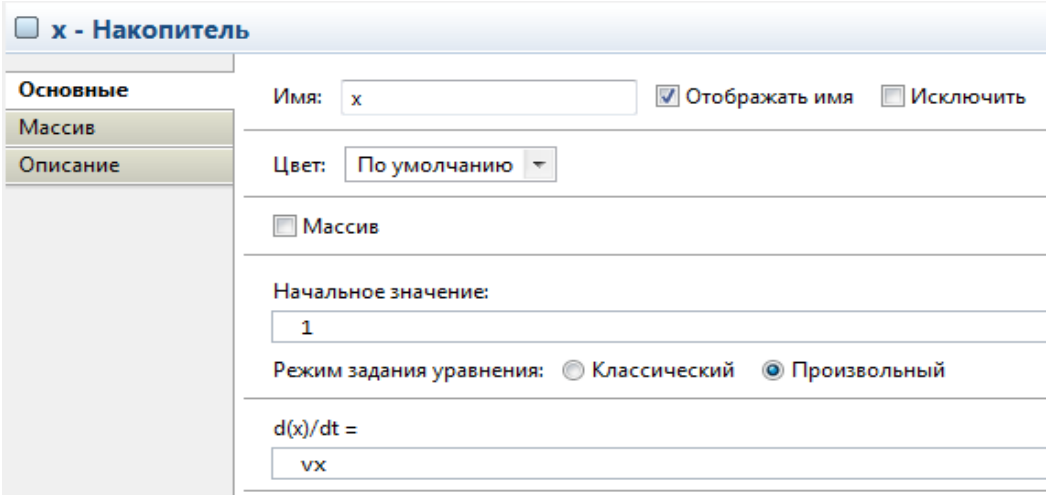


Рис.4.20. Накопитель x

Для второго накопителя (y): начальное значение 1, режим задания уравнения – произвольный, уравнение: vy (рис.4.21)

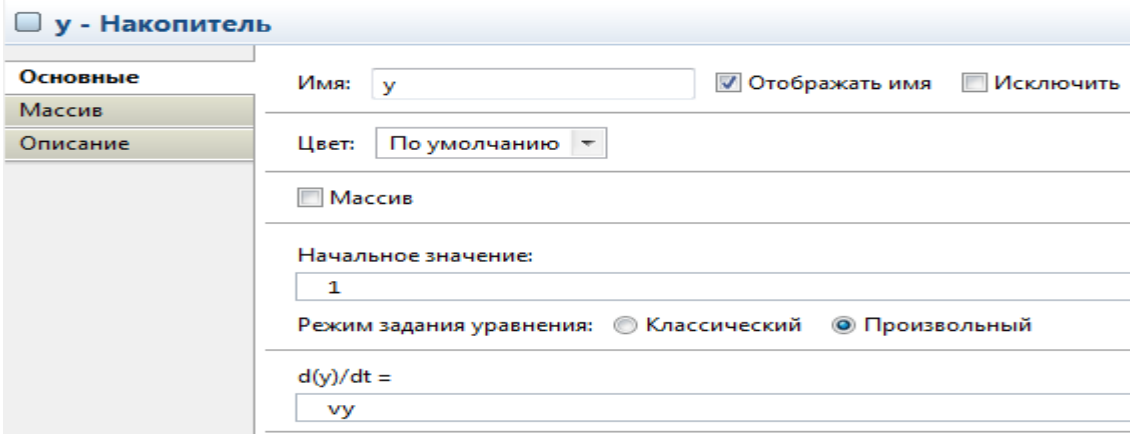


Рис.4.21. Накопитель y

Для третьего накопителя (z): начальное значение 1, режим задания уравнения – произвольный, уравнение: vz (рис.4.22)

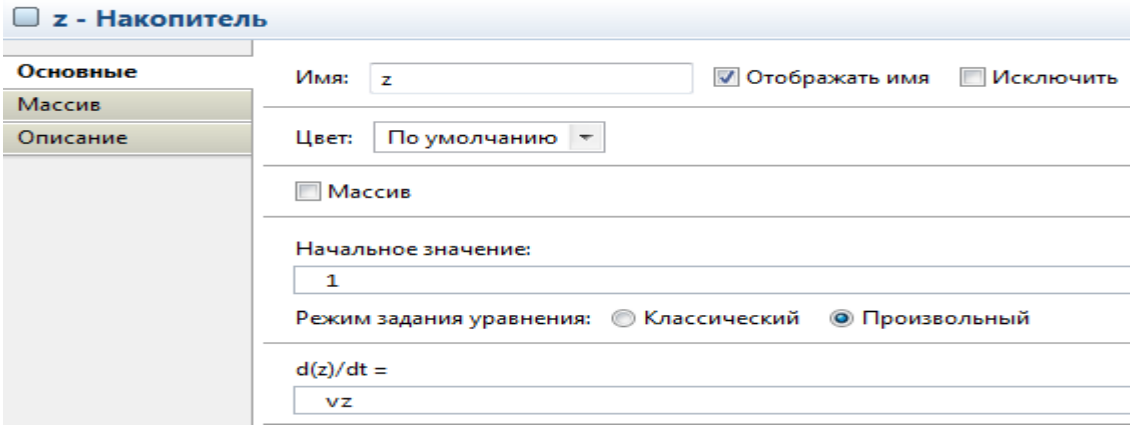


Рис.4.22. Накопитель z

Для четвертого накопителя (vx): начальное значение 0, режим задания уравнения – произвольный, уравнение: -w*w*x+wH*vy (рис.4.23).

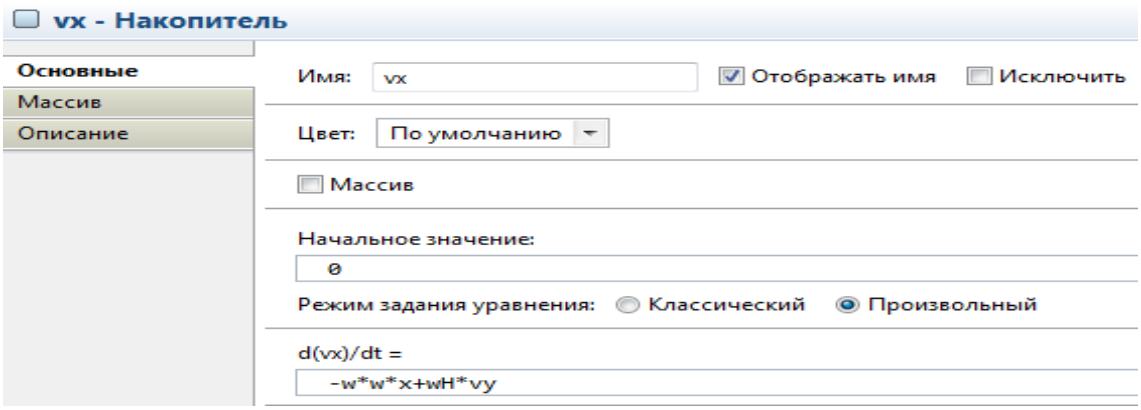


Рис.4.23. Накопитель vx

Для пятого накопителя (vy): начальное значение 0, режим задания равнения – произвольный, уравнение: -w*w*y+wH*vx (рис.4.24).

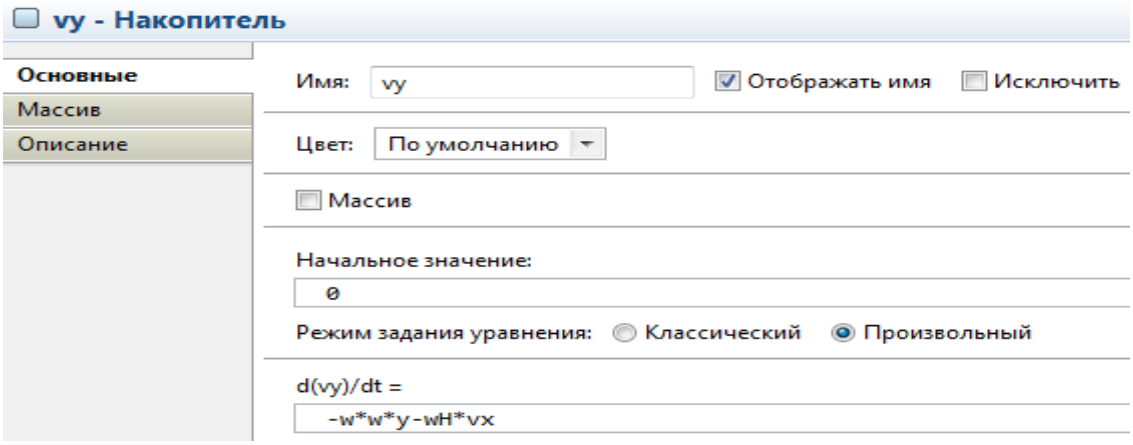


Рис.4.24. Накопитель vy

Для шестого накопителя (vz): начальное значение 0, режим задания уравнения – произвольный, уравнение: -w*w*z (рис.4.25).

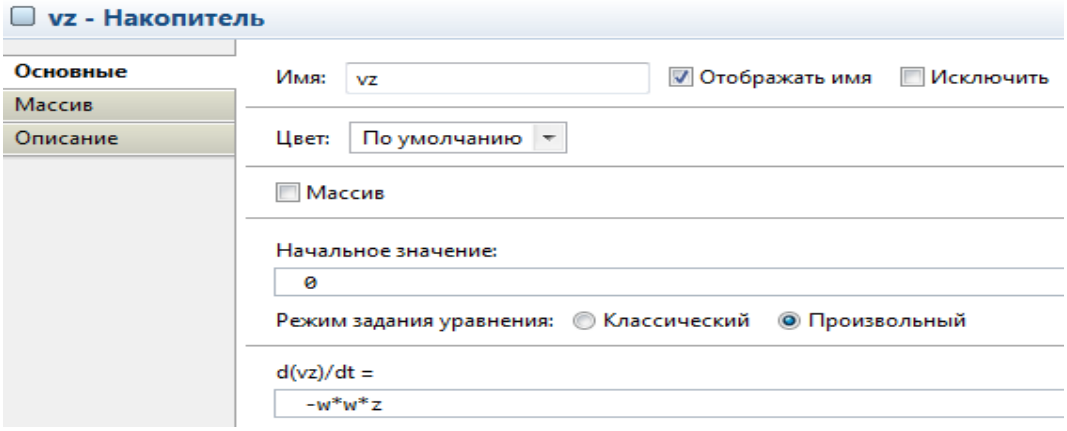


Рис.4.25. Накопитель xz

Для параметра wH заполнить поле “Значение по умолчанию”: 2 (рис.4.26).

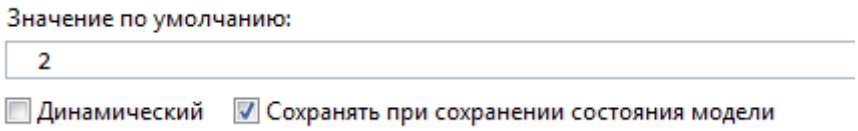


Рис.4.26. Параметр wH

Для параметра w заполнить поле “Значение по умолчанию”: 2 (рис.4.27).

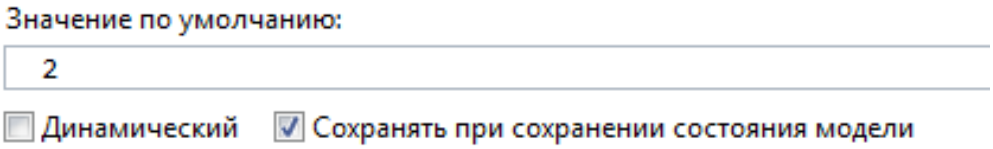


Рис.4.27. Параметр w

Теперь проведем связи между накопителями и параметрами (рис.4.28).

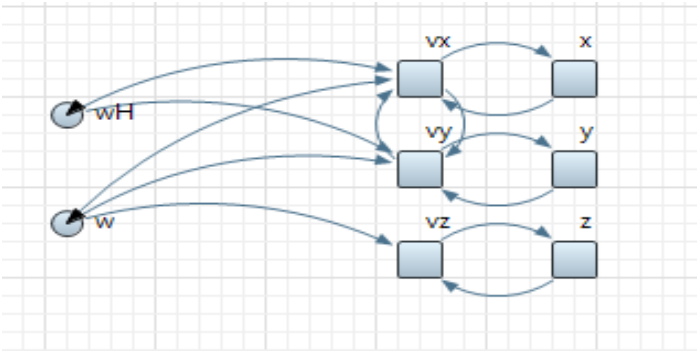


Рис.4.28. Связи

Перетащим график из палитры “Статистика” на модель. Выделим его, нажмем “Добавить элемент данных” и в поле “Значение по оси X” введем x, в поле “Значение по оси Y” введем y. В поле “Период” укажем 0.01, в поле “Отображать до” - 1000. Здесь также можно назначить толщину, цвет, стиль маркера, рисующего график (рис.4.29).

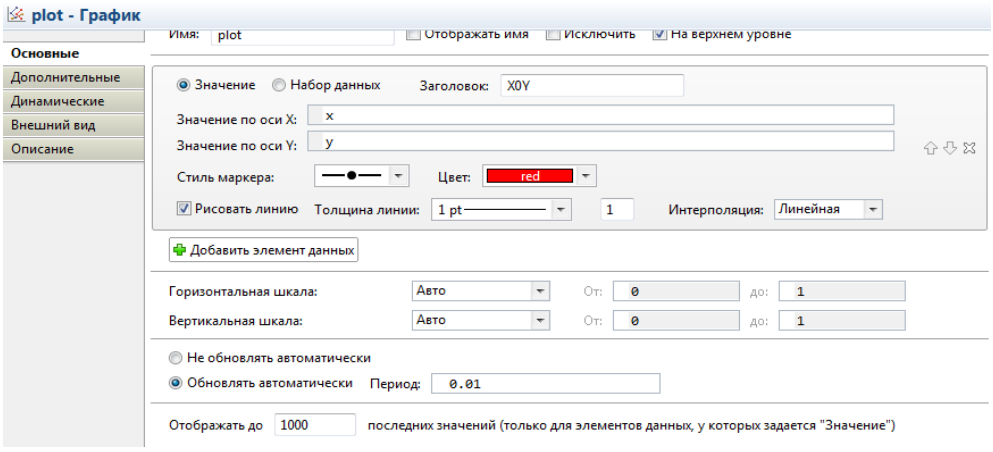


Рис. 4.29. График

Для измения параметров во время выполнения модели, перетащим 2 элемента “Бегунок” из библиотеки “Элементы управления” (рис. 4.30).

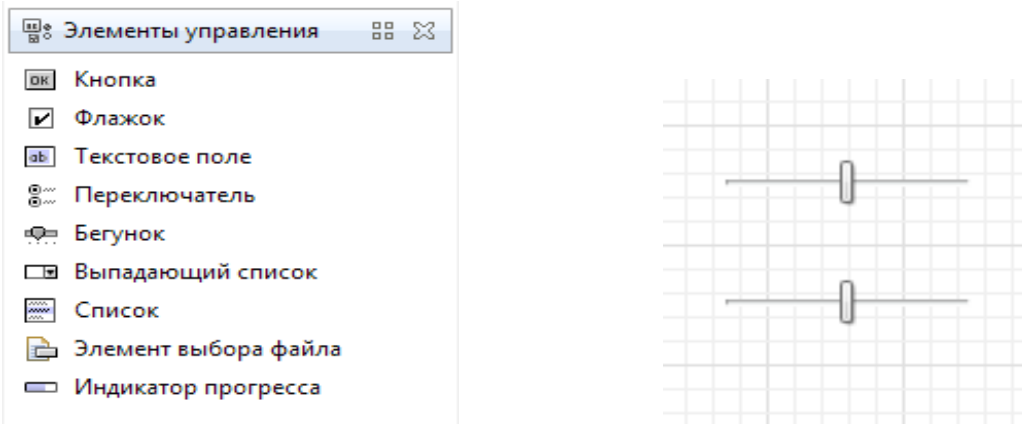


Рис.4.30. Бегунки

Для первого бегунка: свяжем его с параметром w, установим минимальное и максимальное значения (рис.4.31).

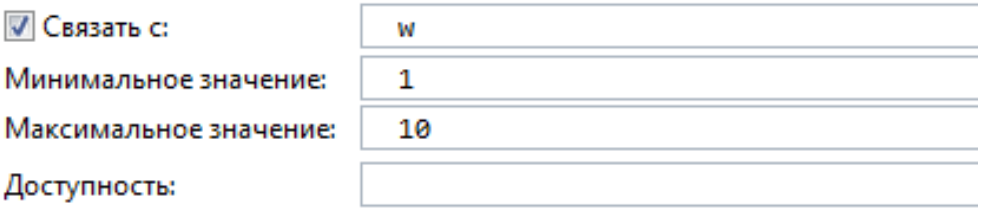


Рис.4.31. Бегунок для параметра w

Для второго бегунка: свяжем его с параметром wH, установим минимальное и максимальное значения (рис.4.32).

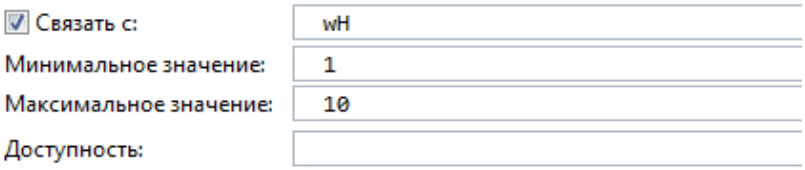


Рис.4.32. Бегунок для параметра wH

Работа модели (рис. 4.33).

Рис.4.33. Пример работы модели “Пространственный осциллятор”

4.3. Связанные маятники

Постановка задачи. Два связанных маятника представляют собой одну колебательную систему с упругой механической связью между ними в виде пружины; требуется воспроизвести колебания маятников.

Дифференциальные уравнения.

$$\frac{dv_1}{dt} = -w_1^2 x_1 - k_1(x_1 - x_2) - 2av_1 \qquad \frac{dx_1}{dt} = v_1$$

$$\frac{dv_2}{dt} = -w_2^2 x_2 - k_2(x_2 - x_1) \qquad \frac{dx_2}{dt} = v_2$$

где x_1 и x_2 - отклонения маятников от положения равновесия, ω_1 и ω_2 - частоты, определяемые параметрами маятников, k_1 и k_2 – коэффициенты, α - коэффициент, который учитывает наличие вязкого трения, препятствующего колебаниям первого маятника.

Создание модели.

Для создания новой модели в меню “Файл” выберем “Создать - Модель” (рис.4.34).

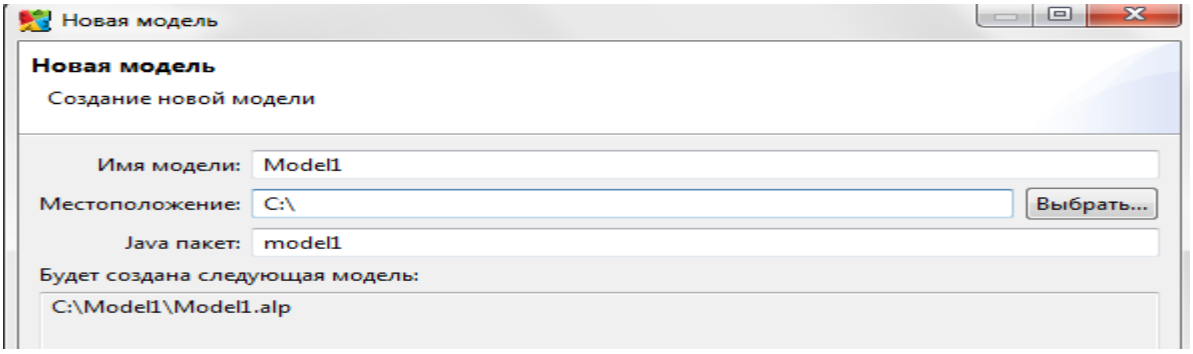


Рис.4.34. Создание модели

В поле имя модели введем “Связанные маятники”, укажем нужное вам местоположение для хранения модели.

Для решения задачи воспользуемся элементами библиотеки “Системная динамика” – накопители и параметры. Перенесем 4 накопителя и 5 параметров на модель (рис.4.35).

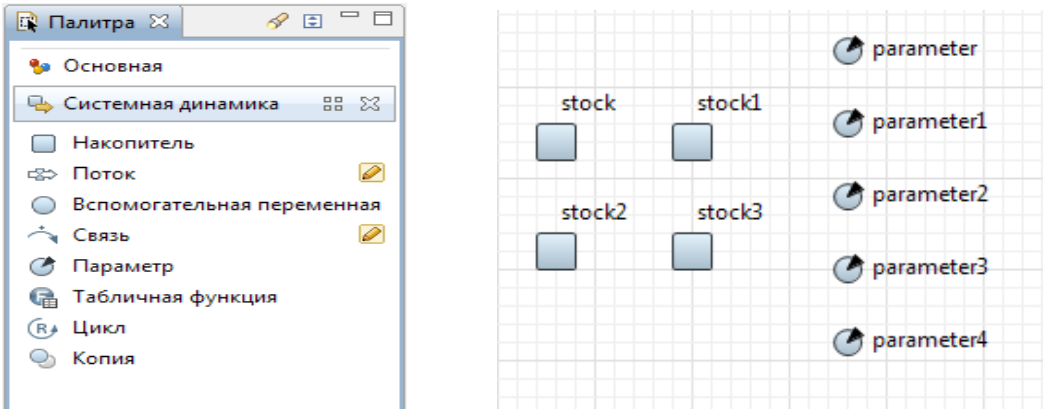


Рис.4.35. Параметры и накопители

Переименуем их, введем дифференциальные уравнения и значения.

Для первого накопителя (v1): начальное значение 0, режим задания уравнения – произвольный, уравнение: -w1*w1*x1-k1*(x1-x2)-2*a*v1 (рис.4.36).

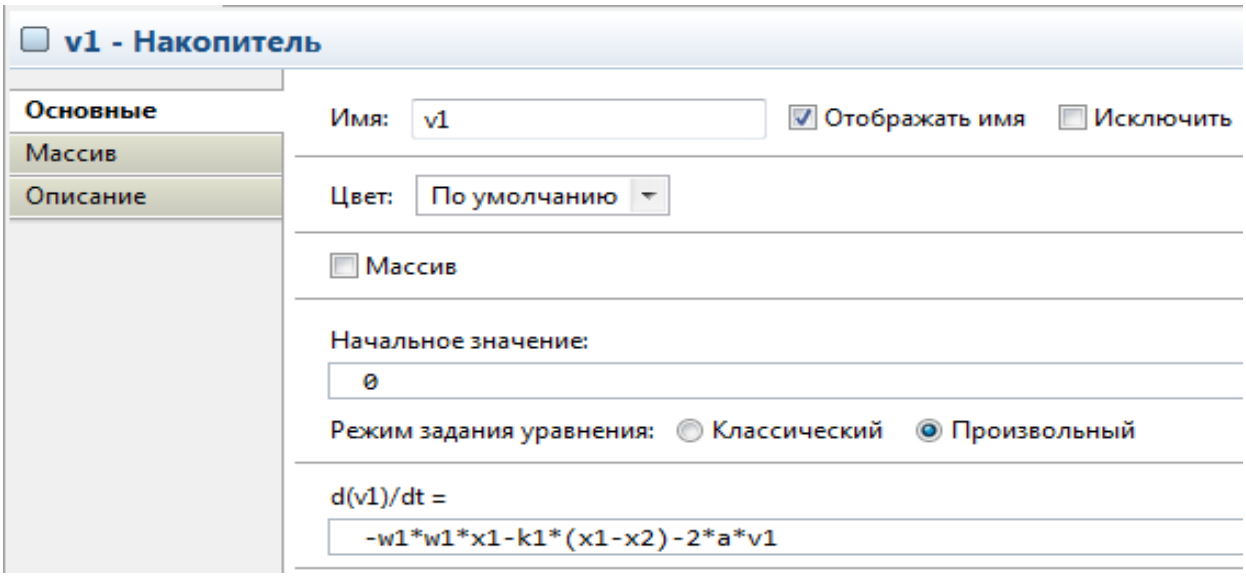


Рис.4.36. Накопитель v1

Для второго накопителя (v2): начальное значение 0, режим задания уравнения – произвольный, уравнение: -w2*w2*x2-k2*(x2-x1) (рис.4.37).

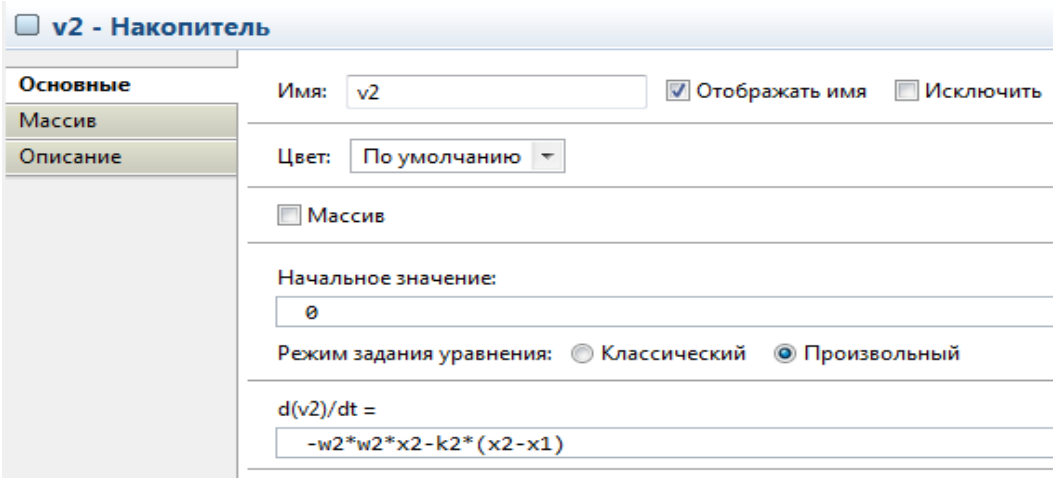


Рис.4.37. Накопитель v2

Для третьего накопителя (x1): начальное значение 0, режим задания уравнения – произвольный, уравнение: v1 (рис.4.38).

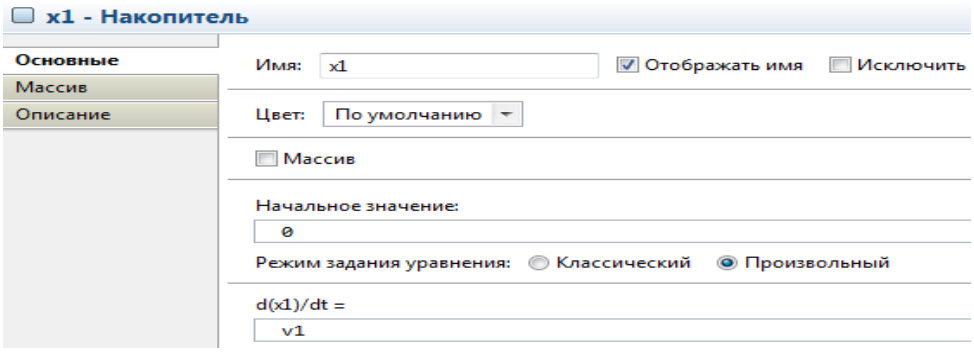


Рис.4.38. Накопитель x1

Для четвертого накопителя (x2): начальное значение 1.57, режим задания уравнения – произвольный, уравнение: v2 (рис.4.39).

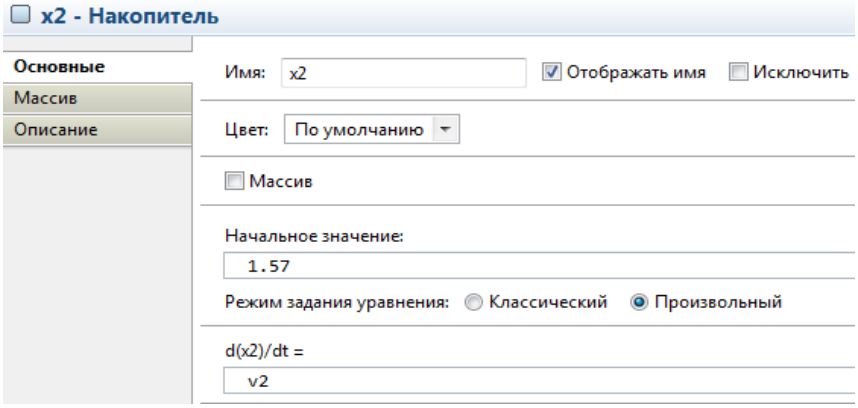


Рис.4.39. Накопитель x2

Для параметра w1 заполнить поле “Значение по умолчанию”: 1.1 (рис.4.40).

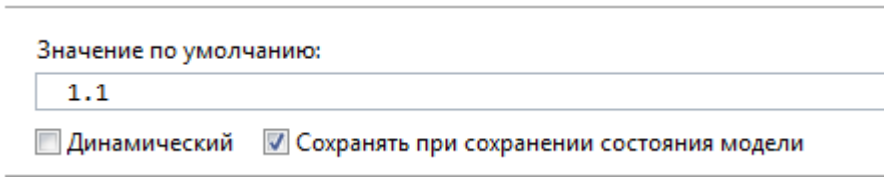


Рис.4.40 – Параметр w1

Для параметра w2 заполнить поле “Значение по умолчанию”: 1.09 (рис.4.41)

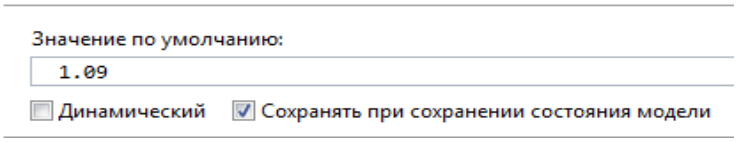


Рис.4.41. Параметр w2

Для параметра k1 заполнить поле “Значение по умолчанию”: 0.1 (рис.4.42).

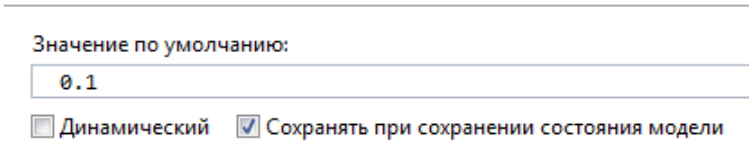


Рис.4.42. Параметр k1

Для параметра k2 заполнить поле “Значение по умолчанию”: 0.1 (рис.4.43).

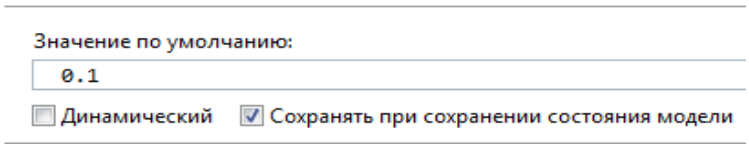


Рис.4.43. Параметр k2

Для параметра a заполнить поле “Значение по умолчанию”: 0 (рис.4.44).

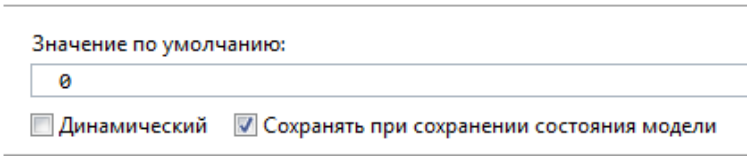


Рис.4.44. Параметр a

Теперь проведем связи между накопителями и параметрами (рис.4.45).

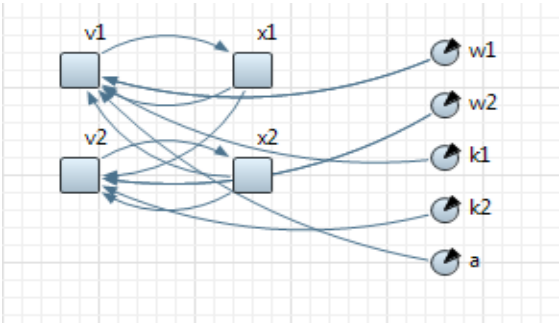


Рис.4.45. Связи

Следующий этап – создание графика. Перетащим график из палитры “Статистика” на модель. Выделим его, нажмем “Добавить элемент данных” и определим оси (значение по оси X: x2, значение по оси Y: x1).

Здесь также можно назначить толщину, цвет, стиль маркера, рисующего график.

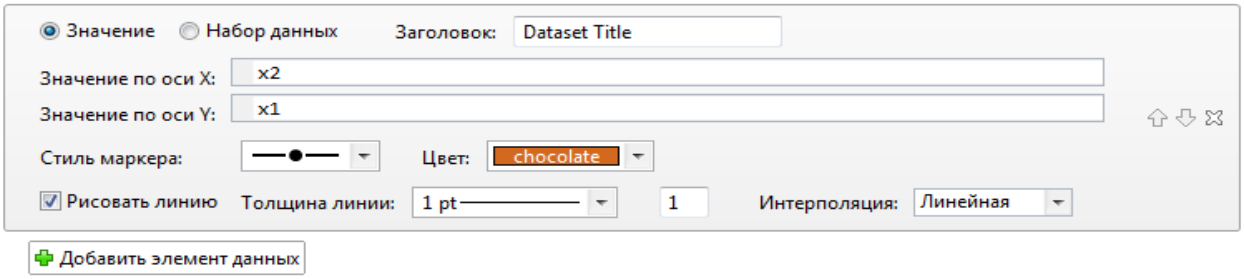


Рис.4.46. График

x1 и x2 – отклонения маятников от положения равновесия. Для того, чтобы показать колебательный процесс во времени, добавим еще 2 графика типа “Временной график” из палитры “Статистика”.

Выделим первый временной график, нажмем “Добавить элемент данных”, в поле “Значение” укажем x1 (рис.4.47).

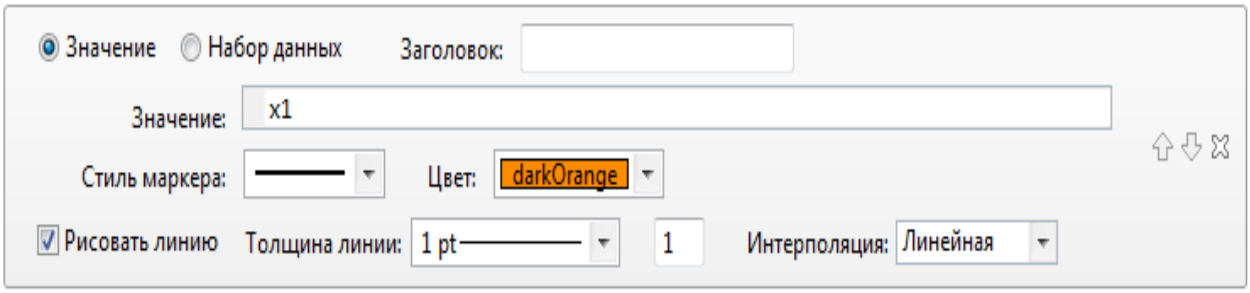


Рис.4.47. Временной график x1

Выделим второй временной график, нажмем “Добавить элемент данных”, в поле “Значение” укажем x2 (рис.4.48).

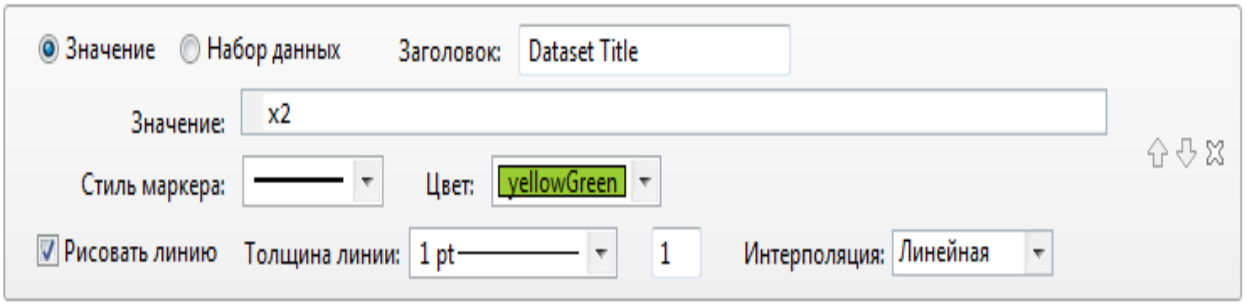


Рис.4.48. Временной график x2

Для измения параметров во время выполнения модели, перетащим 5 элементов “Бегунок” из библиотеки “Элементы управления” (рис.4.49).

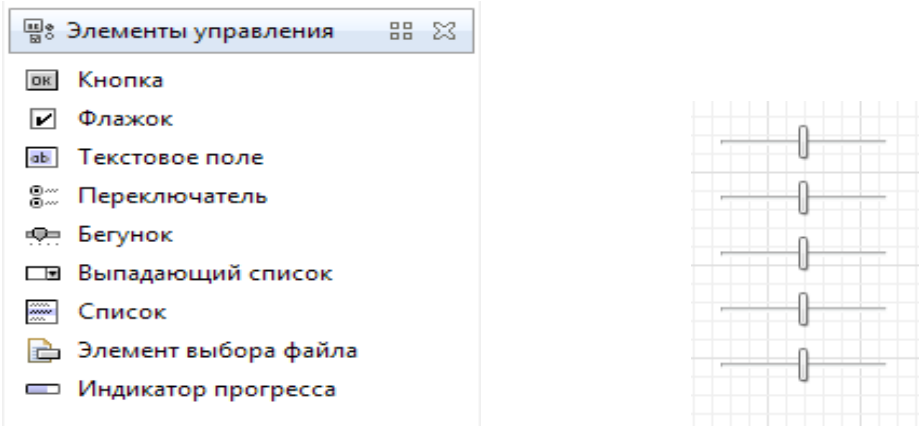


Рис.4.49. Бегунки

Для первого бегунка: свяжем его с параметром w1, установим минимальное и максимальное значения (рис.4.50).

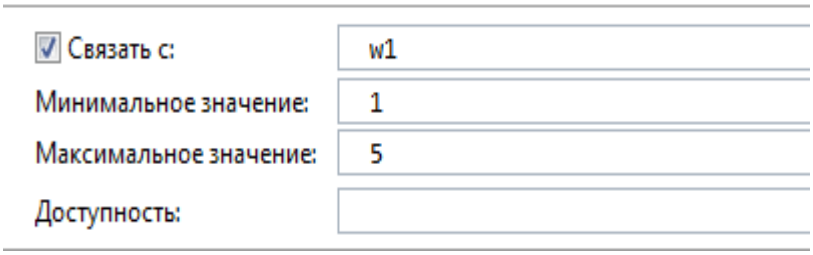


Рис.4.50. Бегунок для w1

Для второго бегунка: свяжем его с параметром w2, установим минимальное и максимальное значения (рис.4.51).

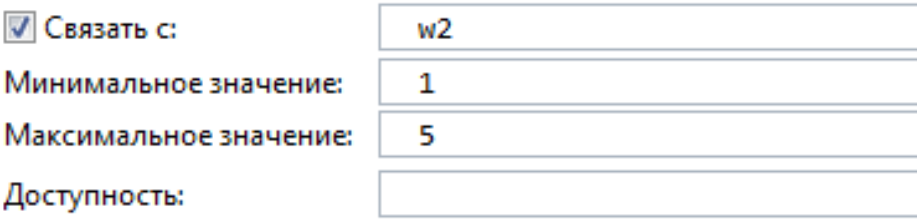


Рис.4.51. Бегунок для w2

Для третьего бегунка: свяжем его с параметром k1, установим минимальное и максимальное значения (рис.4.52).

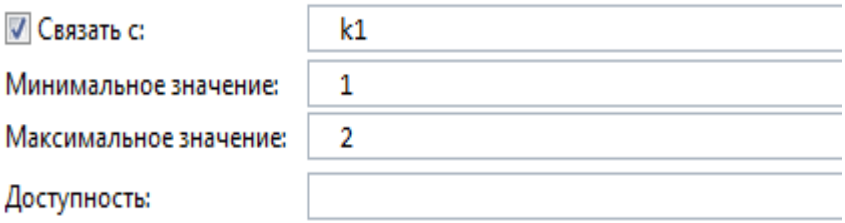


Рис.4.52. Бегунок для k1

Для четвертого бегунка: свяжем его с параметром k2, установим минимальное и максимальное значения (рис.4.53).

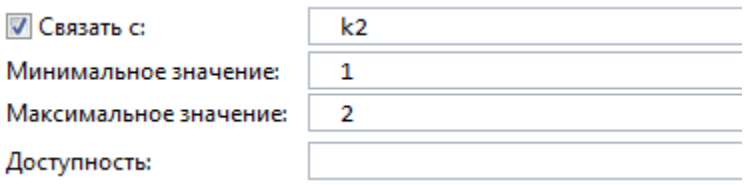


Рис.4.53. Бегунок для k2

Для пятого бегунка: свяжем его с параметром a, установим минимальное и максимальное значения (рис.4.54).

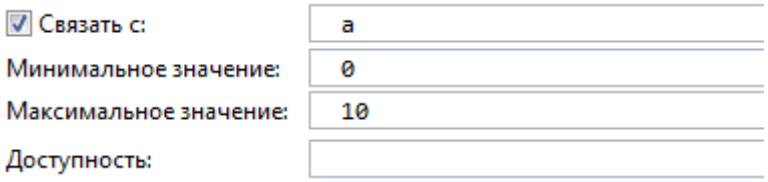


Рис.4.54. Бегунок для a

Работа модели (рис.4.55).

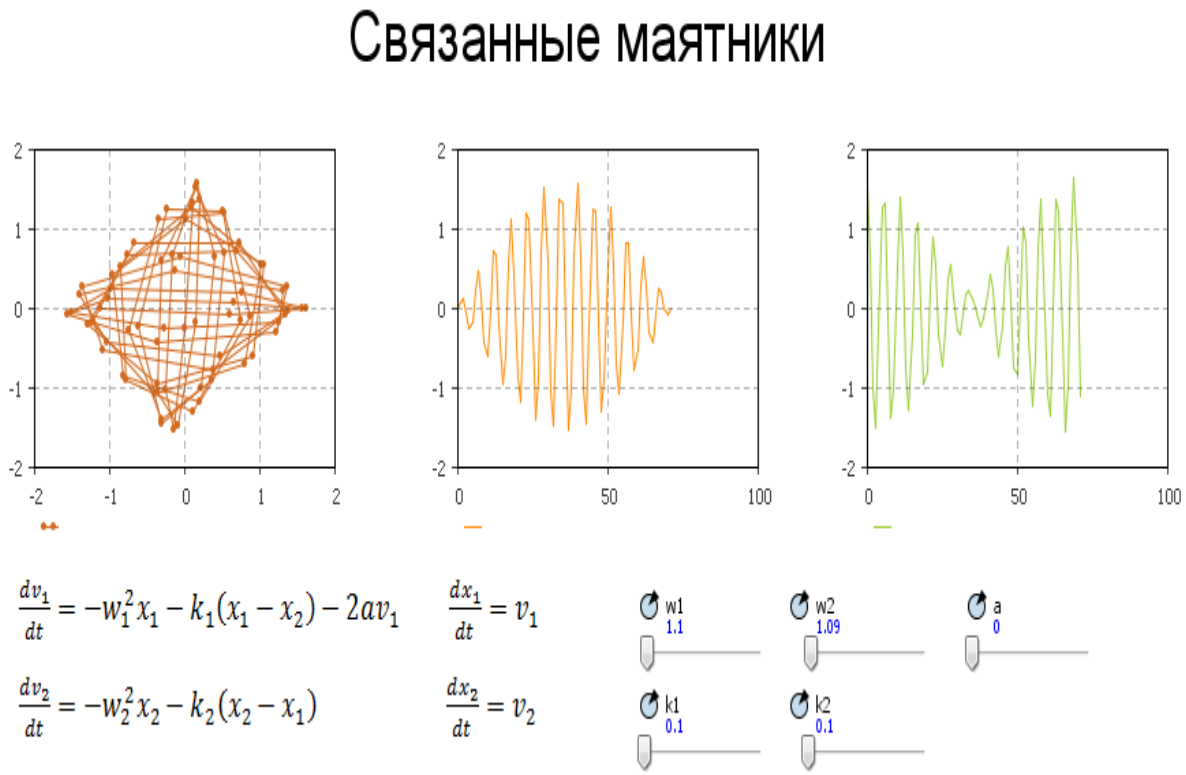


Рис. 4.55. Пример работы модели “Связанные маятники”

ГЛАВА 5

ДИСКРЕТНО-СОБЫТИЙНОЕ МОДЕЛИРОВАНИЕ ПРЕДПРИЯТИЙ ЗДРАВООХРАНЕНИЯ

5.1. Методология дискретно-событийного моделирования

Дискретно-событийное моделирование обязано своим рождением Дж. Гордону, который в начале 1960-х спроектировал и реализовал на IBM систему GPSS. Основной объект в этой системе — пассивный транзакт (заявка на обслуживание), который может определенным образом представлять собой работников, клиентов, покупателей, детали, сырье, документы, сигналы и т. п. «Перемещаясь» по модели, транзакты становятся в очереди к одноканальным и многоканальным устройствам, захватывают и освобождают эти устройства, расщепляются, уничтожаются и т. д. Таким образом, дискретно-событийную модель можно рассматривать как глобальную схему обслуживания заявок. Аналитические результаты для большого количества частных случаев таких моделей рассматриваются в теории массового обслуживания [37,63].

Этот подход используется для описания функционирования системы (процесса) из одного состояния в другое дискретным образом в виде события. Подход к построению имитационных моделей, предлагающий апроксимировать реальные процессы такими событиями и называется "дискретно-событийным" моделированием (discrete event modeling) и широко применяется в теории массового обслуживания, который изучает широкий класс случайных процессов в системах распространения информации, информационно-коммуникативных системах (компьютеры, интернет, связь и т.д.) и в различных отраслях массового обслуживания (железнодорожный, автомобильный транспорт, аэропорты, поликлиники, санаторные и лечебные учреждения, любые торговые предприятия, сферы обслуживания и др.) [1, 15, 16, 66, 69, 88, 94, 111, 127].

Процедура процесса массового обслуживания представляется следующим образом. Имеется очередь заявок (требований) на обслуживание. Поступив в обслуживающую систему, требование присоединяется к очереди других (ранее поступивших) требований. Канал обслуживания выбирает требование из находящихся в очереди, с тем чтобы приступить к его обслуживанию. После завершения процедуры обслуживания очередного требования канал обслуживания приступает к обслуживанию следующего требова-

ния, если таковое имеется в блоке ожидания. Цикл функционирования системы массового обслуживания такого рода повторяется многократно в течение всего периода работы обслуживающей системы.

Основными компонентами системы массового обслуживания являются:

- входной поток поступающих требований (или заявок на обслуживание), который задается на основе вероятностного закона распределения моментов поступления заявок требований;
- дисциплина очереди, принцип, на основе которого подключаются требования к очереди на обслуживание;
- механизм обслуживания определяется продолжительностью процедуры обслуживания и количеством требований, удовлетворяемых в результате.

В зависимости от каналов обслуживания и наличием очередей можно классифицировать системы массового обслуживания: как одноканальные или многоканальные системы, без очереди, с очередью или бесконечной очередью.

5.2. Дискретно-событийная модель стоматологической клиники

Рассмотрим систему, предоставляющую сервисное обслуживание пациентов в стоматологии. В стоматологии есть две регистратуры, отвечающие за два различных типа операций: выдачу амбулаторных карт и записью на прием к врачу. В регистратуру стоят посетители. После обслуживания каждый пациент идет на приём. Цель моделирования такой системы может быть разной. Однако наиболее типичной целью исследования в подобных задачах массового обслуживания является *оценка эффективности системы,* т. е. нахождение числовых значений характеристик, описывающих качество обслуживания системой потока посетителей. Такими характеристиками являются время, проведенное клиентом в стоматологии, длина очереди, которую он отстоял, процент времени занятости обслуживающего персонала. Для поддержки принятия управленческих решений важно также уметь решать обратные задачи анализа: например, определять минимальное количество обслуживающего персонала при ограниченной средней длине очереди клиентов.

Пациенты приходят в стоматологию обычно в случайные моменты времени. У каждого пациента разные причины обращения к врачу, поэтому время обслуживания тоже случайно. Посещение врача занимает случайное время. Стоматология является типичной *системой массового обслуживания*. Такую систему можно представить моделью с небольшим числом типов абстрактных объектов: пациенты представляются заявками на обслуживание, а объекты, выполняющие обслуживание (врачи-стоматологи), представляются приборами, обрабатывающими заявки. Объекты типа «*Очередь»* имитируют ожидание без обработки.

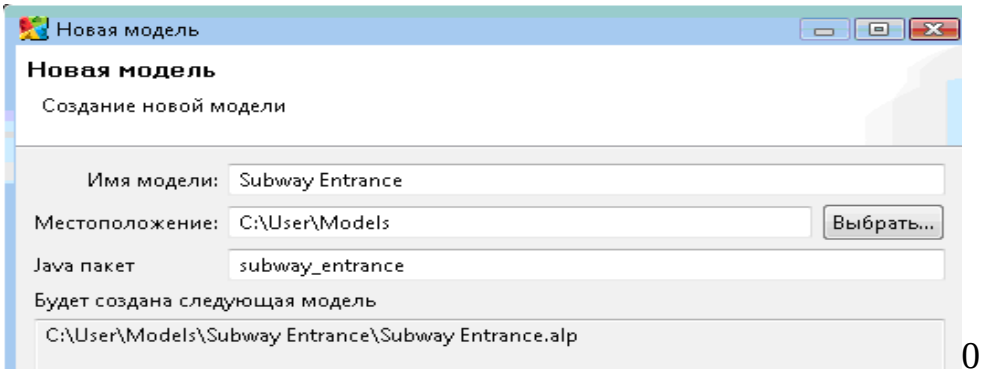
Структура имитационной модели должна отражать структуру реальной системы массового обслуживания: заявки (пациенты) генерируются (входят в систему), становятся в очереди к обслуживающим приборам, а после полного обслуживания покидают систему. Характерной особенностью системы массового обслуживания является стохастическая природа описывающих эти системы характеристик.

Шаг 1. Создание модели. Вначале создадим новую модель посещения больных стоматологический клиники. Для этого мы будем использовать объекты Пешеходной библиотеки.

Создайте новую модель

1. Щелкните мышью по кнопке панели инструментов *Создать*. Появится окно Мастера создания модели.

2. Задайте имя новой модели. В поле *Имя модели* введите Subway Entrance.



0

Рис. 5.1. Создание новой модели

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

4. Щелкните мышью по кнопке *Далее*. Откроется вторая страница Мастера создания модели.

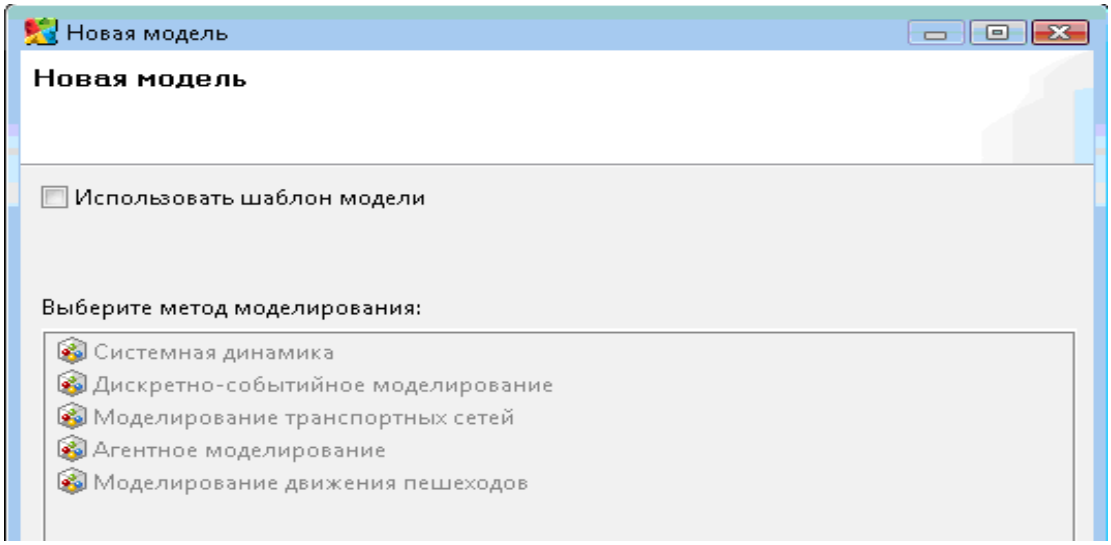


Рис. 5.2. Выбор метода моделирования

5. Здесь Вам будет предложено выбрать шаблон модели, на базе которого Вы будете разрабатывать Вашу модель. Поскольку мы хотим научить Вас процессу создания модели "с нуля", чтобы в дальнейшем Вы могли самостоятельно создавать аналогичные модели, не выбирайте шаблон модели, а просто закончите создание модели, щелкнув мышью по кнопке *Готово*.

Пользовательский интерфейс AnyLogic. В левой части рабочей области будет находиться панель *Проекты*. Панель *Проекты* обеспечивает легкую навигацию по элементам моделей, открытых в текущий момент времени. Поскольку модель организована иерархически, то она отображается в виде дерева: сама модель образует верхний уровень дерева; эксперименты, классы активных объектов и Java классы образуют следующий уровень; элементы, входящие в состав активных объектов, вложены в соответствующую подветвь дерева класса активного объекта и т.д.

В правой рабочей области будет отображаться панель *Палитра*, а внизу - панель *Свойства*. Панель *Палитра* содержит разделенные по категориям элементы, которые могут быть добавлены на диаграмму класса активного объекта или эксперимента. Панель *Свойства* используется для просмотра и изменения свойств выбранного в данный момент элемента (или элементов) модели.

В центре рабочей области AnyLogic Вы увидите графический редактор диаграммы класса активного объекта Main.

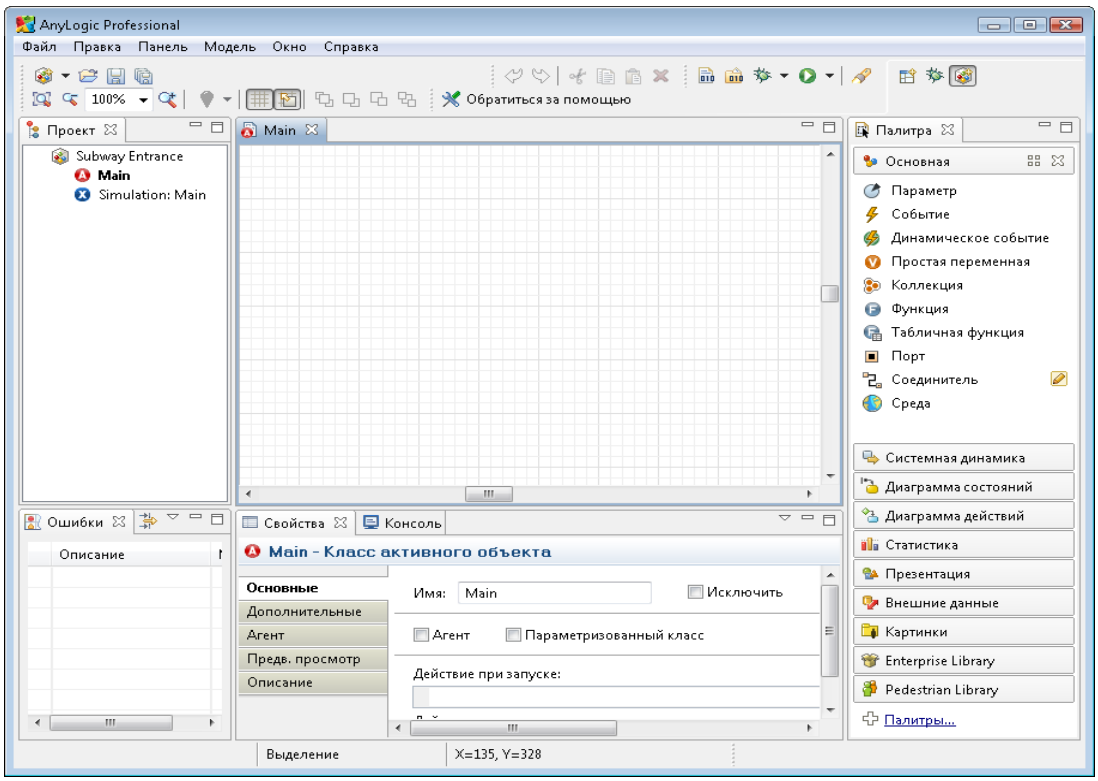


Рис. 5.3. Интерфейс среды

Шаг 2. Создание анимации. Теперь мы готовы к тому, чтобы начать разработку нашей модели.

Прежде всего, нам нужно создать анимацию модели. С помощью анимации Вы сможете визуально отображать поведение модели, кроме того, анимация нужна для того, чтобы графически задать объекты среды модели.

Чтобы облегчить рисование, мы вначале добавим изображение плана стоматологической клиники. Мы не будем рисовать план в графическом редакторе, а просто вставим уже готовое изображение.

Добавьте рисунок с изображением плана стоматологической клиники.

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

2. Палитра *Презентация* содержит элементы, используемые для рисования презентаций моделей: фигуры, с помощью которых Вы можете рисовать сложные презентации, а также элементы управления, с помощью которых Вы можете сделать Ваши презентации интерактивными.

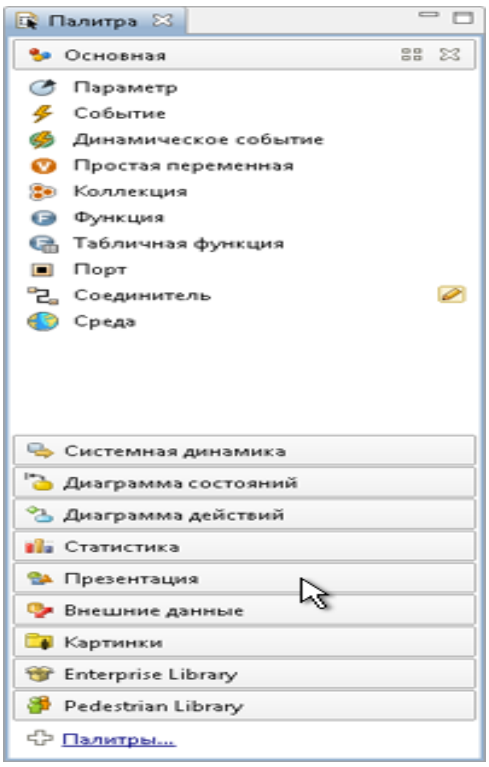


Рис. 5.4. Меню Палитр

3. Перетащите элемент *Изображение* из палитры *Презентация* на диаграмму класса активного объекта. Поместите его так, как показано на рисунке:

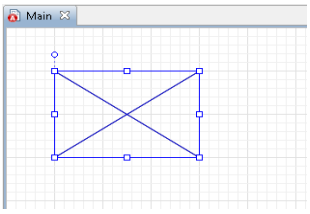


Рис. 5.5. Создание изображения

4. Задайте свойства изображения в панели *Свойства*. Щелкните мышью по кнопке *Добавить* и выберите файл изображения плана павильона. Вы увидите добавленный план в области изображения.

5. Чтобы сохранить исходный размер изображения, установите флажок Исходный размер.

6. Заблокируйте изображение, установив флажок Блокировать. Вы не сможете выбрать заблокированную фигуру в графическом редакторе до тех пор, пока не снимете с нее блокировку. Мы делаем так потому, что мы будем рисовать другие фигуры поверх этого изображения, и поэтому мы хотим исключить возможность случайного редактирования изображения при рисовании этих фигур.

7. Изображение должно будет выглядеть следующим образом:

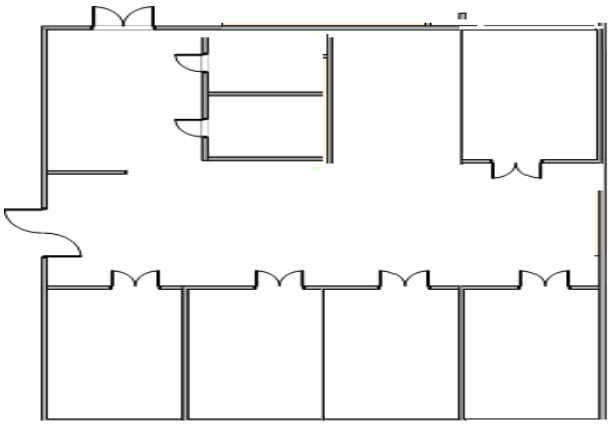


Рис. 5.6. Создание изображения клиники

Шаг 3. Нарисуйте узлы сети.

1. Нарисуйте каждую комнату отделения с помощью прямоугольника. Отделение офтальмологии включает в себя приемный покой, три процедурные, комнату хранения офтальмоскопов и комнату для персонала.

2. На приведенном ниже рисунке нарисованы и подписаны прямоугольники, которые Вам нужно будет нарисовать. Назовите прямоугольники именно так, как показано на рисунке. Это важно, потому что в дальнейшем мы будем ссылаться на ключевые области нашего отделения именно по именам соответствующих прямоугольников.

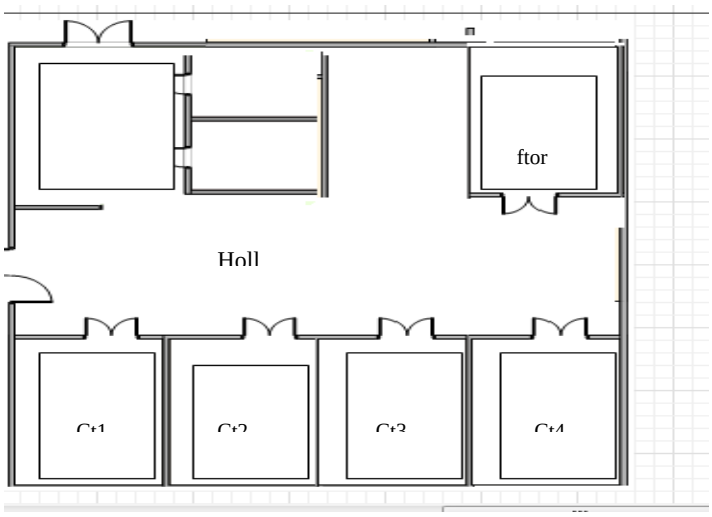


Рис. 5.7. Создание плана клиники

3. Чтобы нарисовать прямоугольник, сделайте двойной щелчок мышью по элементу **Прямоугольник** в палитре (при этом его значок должен поменяться на этот:). Затем щелкните мышью в том месте диаграммы, где Вы хотите нарисовать прямоугольник, и передвиньте ее, не отпуская

кнопки, пока контур создаваемого прямоугольника не примет желаемого размера, после чего отпустите кнопку мыши.

4. Рисуйте прямоугольники так, чтобы они помещались в соответствующие области на плане отделения, как показано на рисунке ниже. Чтобы облегчить рисование фигур нужного Вам размера, выключите привязку фигур к сетке диаграммы, щелкнув по кнопке панели инструментов **Включить/Отключить сетку**.

5. Чтобы сделать пути движения людей на анимации более реалистичными, добавьте дополнительные узлы сети, нарисовав еще несколько прямоугольников и поместив их так, как показано на приведенном ниже рисунке:

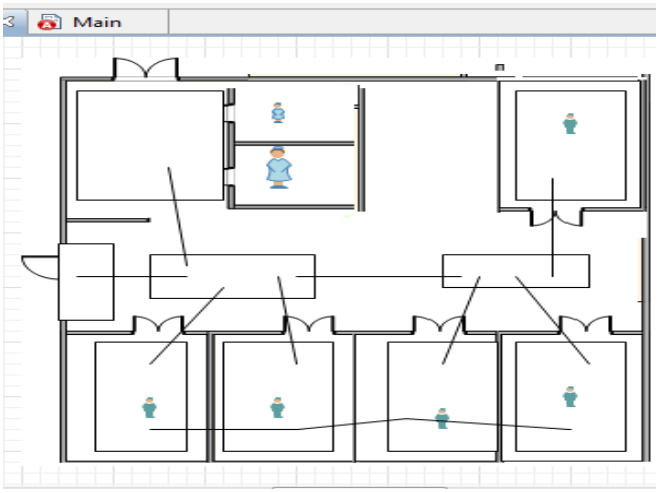


Рис. 5.5. Создание путей движения пациентов

Теперь мы нарисуем сегменты сети с помощью ломаных линий. Эти линии будут задавать пути движения пациентов и персонала больницы.

Шаг 4. Нарисуйте пути (сегменты сети).

1. С помощью инструмента рисования **Ломаная**, нарисуйте ломаные линии, как показано на рисунке ниже. Соедините соседние узлы сети - тем самым Вы зададите требуемую транспортную сеть модели:

2. Чтобы нарисовать ломаную, сделайте двойной щелчок мышью по элементу **Ломаная** в палитре (при этом его значок должен поменяться на этот:). Теперь Вы можете рисовать ломаную точка за точкой, последовательно щелкая мышью в тех точках диаграммы, куда Вы хотите поместить вершины ломаной.

3. Чтобы завершить рисование, добавьте последнюю точку ломаной двойным щелчком мыши.

4. Все начальные и конечные точки линий должны обязательно находиться внутри соединяемых прямоугольников (и в одном прямоугольнике не может находиться более одной точки одной и той же ломаной).

Теперь нам нужно будет добавить наши фигуры в группу. На базе элементов этой группы, которую мы потом укажем в соответствующем параметре конфигурационного объекта сети, будет сконструирована логическая структура сети.

Шаг 5. Добавьте фигуры в группу.

1. Добавьте все нарисованные фигуры в группу фигур. Вначале выберите эти фигуры. Лучше всего сделать это, нажав левую кнопку мыши сбоку от крайней из этих фигур, и, не отпуская кнопку перетащить мышь так, чтобы рамка выделения покрыла всю область, содержащую фигуры, которые Вы хотите выделить (как на рисунке ниже).

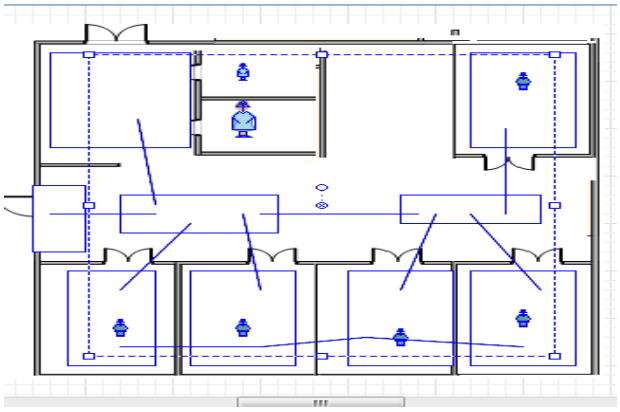


Рис. 5.9. Создание путей движения пациентов

2. Отпустите кнопку мыши. Выделенные таким образом фигуры будут подсвечены синим цветом. Если какая-то из фигур оказалась не выделенной, Вы можете добавить ее в группу выделенных фигур, нажав Ctrl, и, не отпуская ее, щелкнуть мышью по той фигуре, которую Вы хотите добавить. Фигура будет добавлена в выделение.

3. Когда Вы выделите все фигуры, сделайте щелчок правой кнопкой мыши по выделенным фигурам и выберите **Создать группу** из контекстного меню.

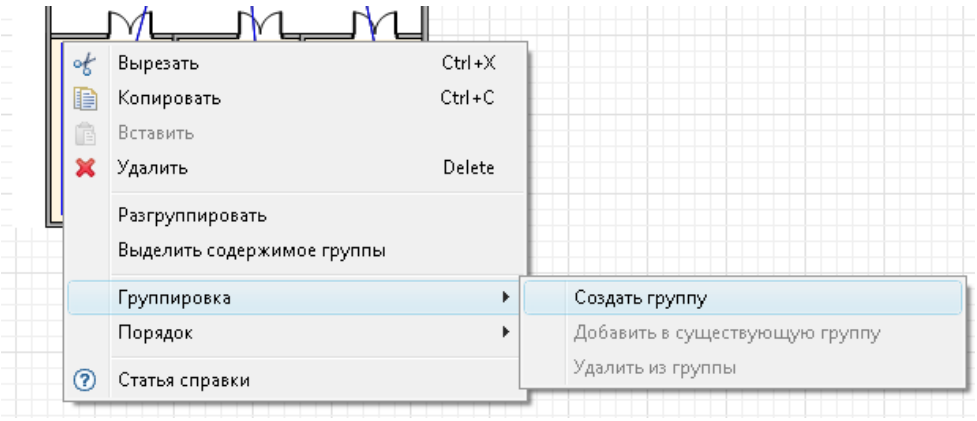


Рис. 5.10. Группировка объектов

Теперь мы нарисуем анимацию для нашего пациента и врача, чтобы мы могли различать их на анимации нашей модели.

Шаг 6. Создайте изображения врача и пациента.

1. Вы можете либо сами нарисовать фигурку врача, либо использовать одну из стандартных картинок, поставляемых вместе с AnyLogic. Мы выбираем второй подход как наиболее легкий. Откройте палитру **Картинки**. Эта палитра содержит набор картинок, которые наиболее часто используются пользователями AnyLogic при создании моделей.

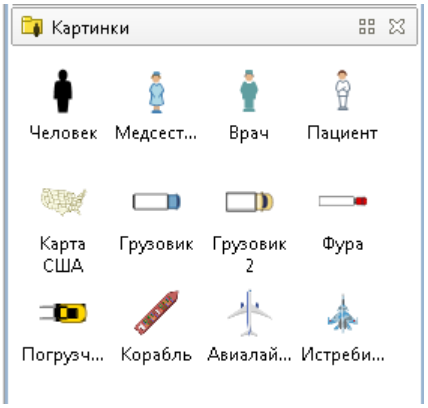


Рис. 5.11. Библиотека объектов

2. Перетащите элемент **Врач** из палитры на диаграмму графического редактора:

Рис. 5.12. Объект - доктор

3. По умолчанию эта картинка будет называться doctor. Оставьте ее название без изменений, поскольку в дальнейшем мы будем ссылаться на нее в блоке диаграммы нашего процесса именно по этому имени.

4. Перетащите элемент **Пациент** из палитры на диаграмму графического редактора:

Рис. 5.13. Объект - доктор и пациент

5. По умолчанию эта картинка будет называться patient. Не меняйте и это название по выше описанной причине.

Мы закончили создание презентации нашей модели. Теперь мы можем изменить настройки сети и задать процесс с помощью диаграммы процесса, составляемой из блоков моделирования транспортных сетей Основной библиотеки.

Теперь мы изменим свойства объекта, описывающего сеть, и добавим объекты, задающие имеющиеся в нашей модели ресурсы.

Шаг 7. Задание свойств сети.

1. Добавьте на диаграмму объект **Network**. Этот объект задает транспортную сеть модели и ее свойства. Детальное описание объекта, а также всех его параметров и функций Вы можете найти в Справочном руководстве по Основной библиотеке: Объекты Основной библиотеки.

2. Чтобы добавить на диаграмму объект Основной библиотеки, нужно открыть в панели **Палитра** палитру этой библиотеки, щелкнув мышью по панели с ее заголовком, а затем перетащить нужный Вам объект из палитры на диаграмму класса.

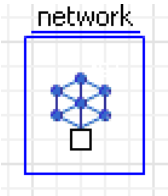


Рис. 5.14. Объект – транспортная сеть

3. В параметре **Группа фигур сети** введите имя группы фигур анимации: group.

Этот параметр позволяет указать этому объекту **Network**, какие именно фигуры анимации составляют логическую структуру задаваемой этим объектом сети. Проверьте, называется ли группа Ваших фигур (в которую Вы добавили нарисованные ранее прямоугольники и соединяющие их ломаные, представляющие узлы и сегменты сети) group, и если нет – то введите здесь ее действительное имя.

Чтобы проверить имя группы фигур, щелкните по любой фигуре этой группы, и имя группы будет отображено в панели **Свойства**.

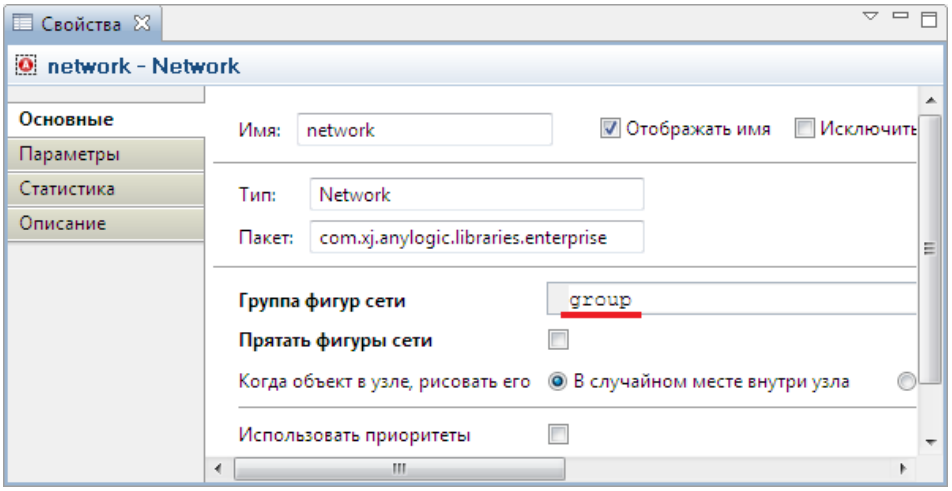


Рис. 5.15. Формирование группы объект

Теперь мы добавим объект, задающий сетевые ресурсы. Сетевые ресурсы могут быть трех видов: движущиеся, переносные и статические. В нашем случае процедурные комнаты – статические.

Шаг 8. Задание ресурсов типа "процедурная комната".

Добавьте еще один объект **NetworkResourcePool**. Этот объект будет задавать свойства ресурсов, представляющих в нашей модели процедурные комнаты.

Задайте следующие свойства объекта:

1. Назовите объект Room.
2. Измените **Тип ресурса** на **Статический**.

3. Задайте местоположение комнат. Статические ресурсы всегда находятся в месте, указанном как базовое. Вы можете задать несколько таких мест, для этого нужно нарисовать ломаную линию с точками, лежащими в соответствующих прямоугольниках, и указать ее в свойствах объекта. Поскольку в нашей модели три таких комнаты, то мы должны воспользоваться именно этим способом. Для этого выберите **Путь через узлы** из выпадающего списка **Базовое местоположение задается как** и введите в поле **Путь через узлы** имя созданной ранее именно для этой цели ломаной: polyline5.

4. Укажите, что количество ресурсов, задаваемое этим объектом, равно количеству точек указанной ломаной. Для этого выберите из группы кнопок **Количество задано** опцию **Фигурой базового местоположения**.

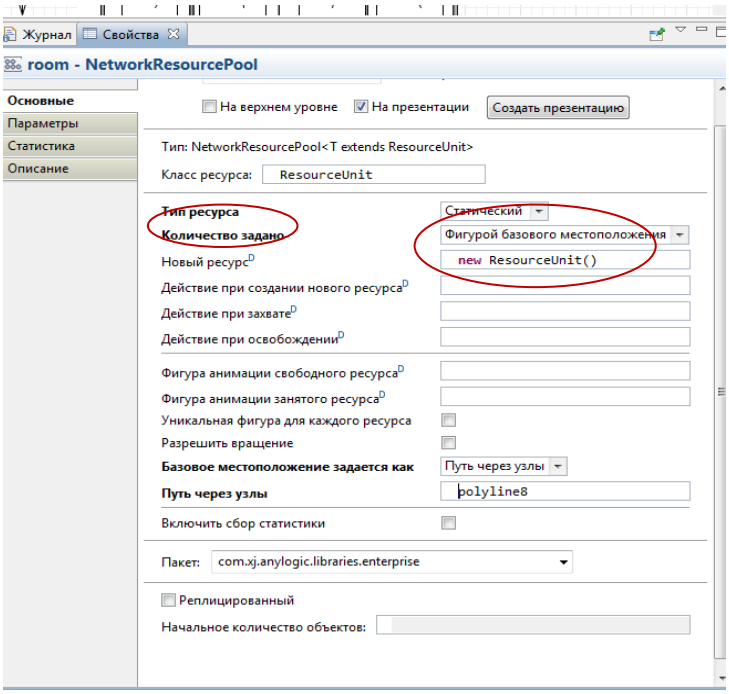


Рис. 5.16. Задание свойств ресурсов

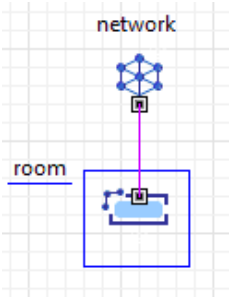


Рис. 5.17. Подключение к сети

Шаг 9. Диаграмма процесса.

Теперь мы закончим создание модели, в которой пациенты, прибывающие в отделение, вначале направляются в регистратуру, а затем следуют в кабинет стоматолога, где проводится процедура. После ее проведения они покидают отделение.

1. Создайте следующую диаграмму процесса, добавив на диаграмму класса Main новые блоки Основной библиотеки и соединив их так, как показано на рисунке ниже:

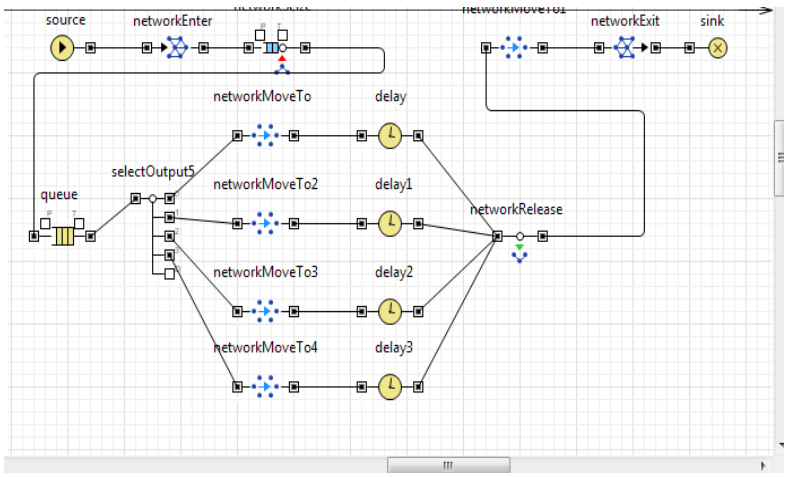


Рис. 5.15. Дискретно-событийная модель стоматологической клиники

Эта диаграмма будет описывать моделируемый нами процесс. Давайте бегло ознакомимся с тем, из каких блоков она состоит. Попутно мы будем изменять параметры блоков в соответствии с моделируемой нами задачей.

Первым следует объект **Source**. Объект **Source** создает заявки. Обычно он используется в качестве начальной точки потока заявок. В нашем случае заявки будут представлять собой пациентов, и этот блок будет моделировать поступление пациентов в отделение.

Измените свойства объекта source

В поле **Интенсивность прибытия** задайте интенсивность поступления пациентов на отделение: 0.05.

В поле **Фигура анимации заявки** введите имя ранее добавленной нами фигурки пациента: patient.

За этим объектом следует блок **NetworkEnter**. Этот объект добавляет заявки в заданное место сети, в нашем случае мы хотим, чтобы он помещал пациентов в приемный покой.

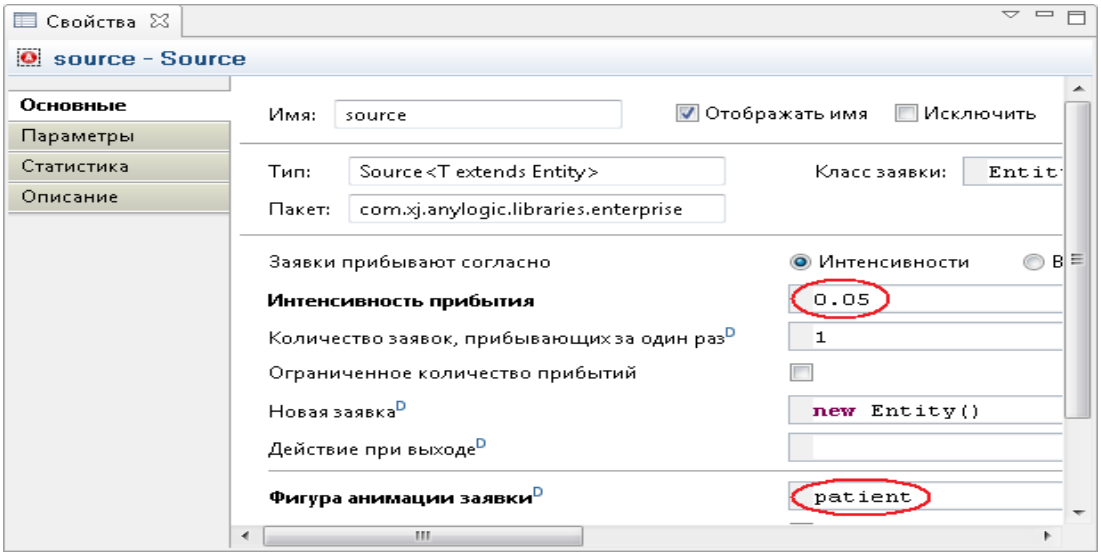


Рис. 5.19. Задание интенсивности поступления пациентов

Измените свойства объекта networkEnter

1. Введите Holl в поле **Узел входа**. Здесь Holl – это имя ранее нарисованного нами прямоугольника, который располагается на плане отделения прямо поверх приемного покоя. Заявки-пациенты будут прибывать в указанный узел сети, задающий в нашей модели приемный покой.

2. В поле **Сеть** Вы можете увидеть введенное **Мастером создания модели** имя объекта **Network** (network). Именно в сеть, заданную этим объектом, и будут помещаться заявки этим блоком.

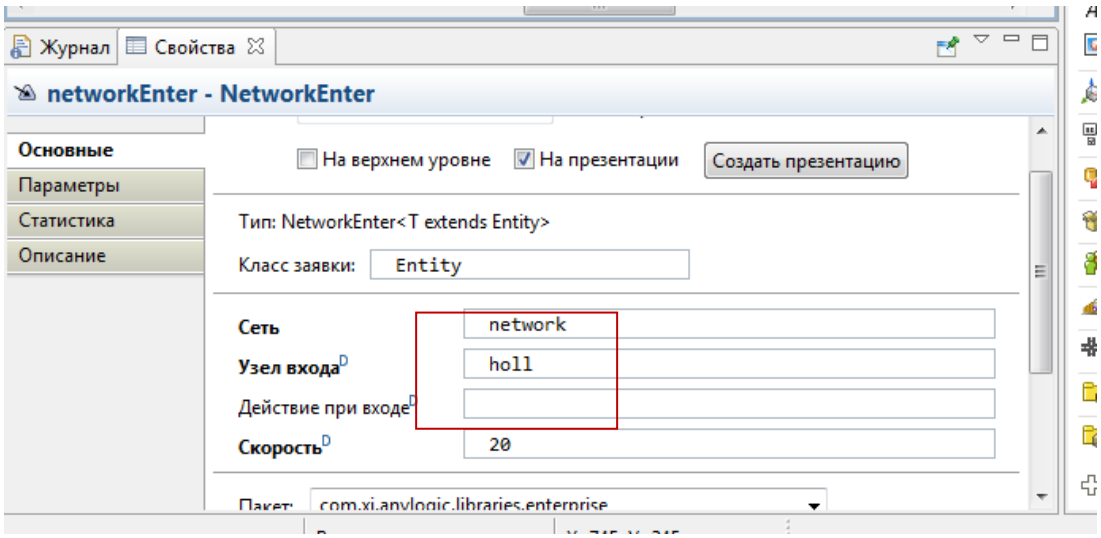


Рис. 5.20. Поступление пациентов в приемный покой

Следующий блок в диаграмме процесса – **NetworkMoveTo**. Объект **NetworkMoveTo** перемещает заявку в новое место сети. С помощью этого объекта мы хотим промоделировать, как пациенты переходят из приемного покоя в процедурную комнату.

Задайте следующие свойства объекта**NetworkResourcePool**:

В поле **Список ресурсов {pool1, ...}** введите список ресурсов, которые мы хотим занять с помощью этого блока. Список составляется следующим образом: Вы перечисляете имена объектов типа **NetworkResourcePool**, которые задают те ресурсы, которые будут захватываться данным блоком. Имена объектов пишутся через запятую, а весь список заключается в фигурные скобки. Поскольку нам нужно, чтобы этот блок занимал одну процедурную комнату, напишите здесь {room}. Если Вам необходимо захватить несколько ресурсов одного типа, то Вам нужно будет написать имя задающего эти ресурсы объекта столько раз, сколько ресурсов Вы хотите захватить.

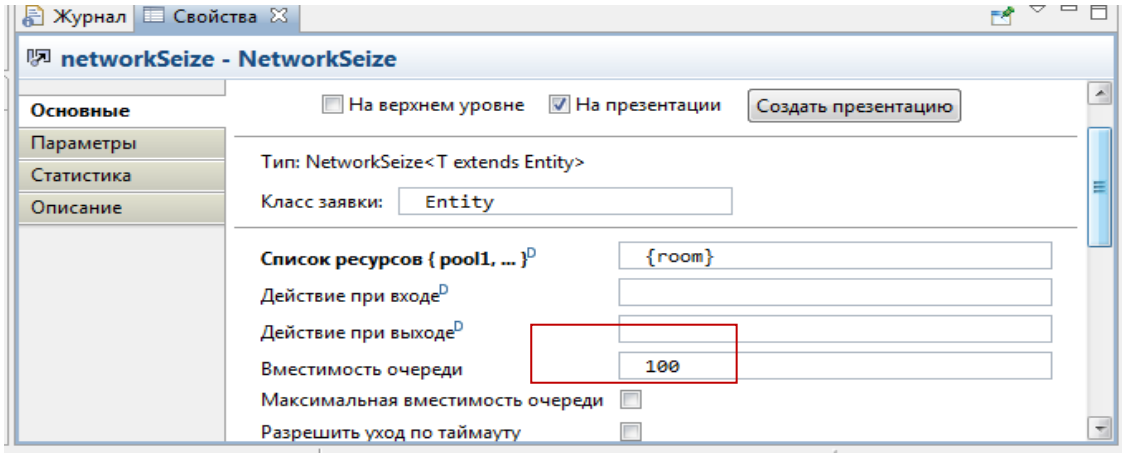


Рис. 5.21. Определение процедурной комнаты

Далее наши пациенты направляются каждый к своему врачу. В нашей модели их 4.

Измените свойства объекта networkMoveTo

1. Введите в поле **Узел** 1 (имя прямоугольника, который задает одну из процедурных комнат). Пациенты будут перемещаться в указанный Вами узел сети. На данный момент всех пациентов будут осматривать в первой процедурной комнате.

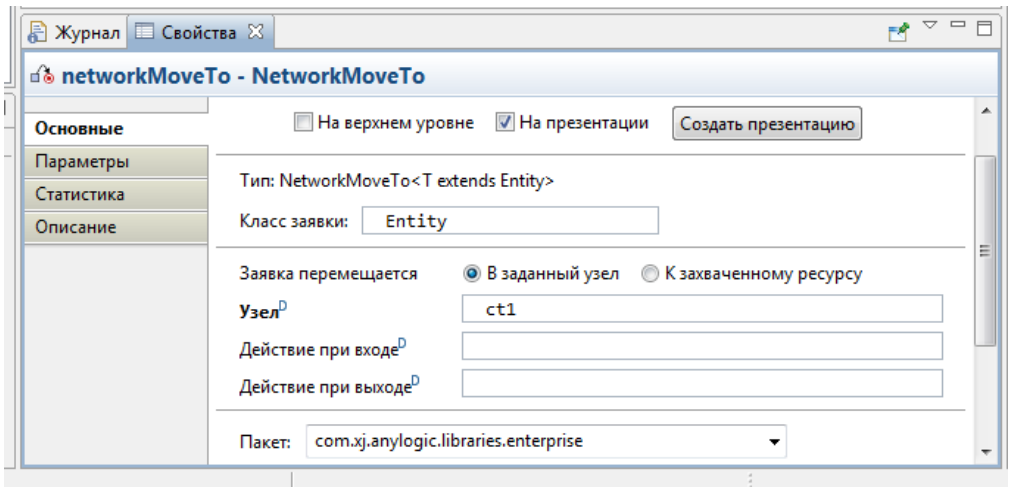


Рис. 5.22. Перемещение пациентов

Объект **Delay** задерживает заявку на заданное время. В нашей модели этот объект моделирует задержку, связанную с проведением процедуры.

Присутствие "обычного" несетевого блока **Delay** в диаграмме сетевого процесса наглядно подтверждает тот факт, что Вы можете использовать в диаграмме любые объекты Основной библиотеки, такие как **Queue**, **Delay**, **Service** и т.д.

Еще один блок **NetworkMoveTo** будет моделировать то, как пациенты направляются к выходу после проведения процедуры.

Измените свойства объекта networkMoveTo

1. Мы хотим, чтобы этот объект перемещал пациентов в указанный нами узел сети.
2. Введите в поле **Узел** exit (имя прямоугольника, который задает выход из отделения).

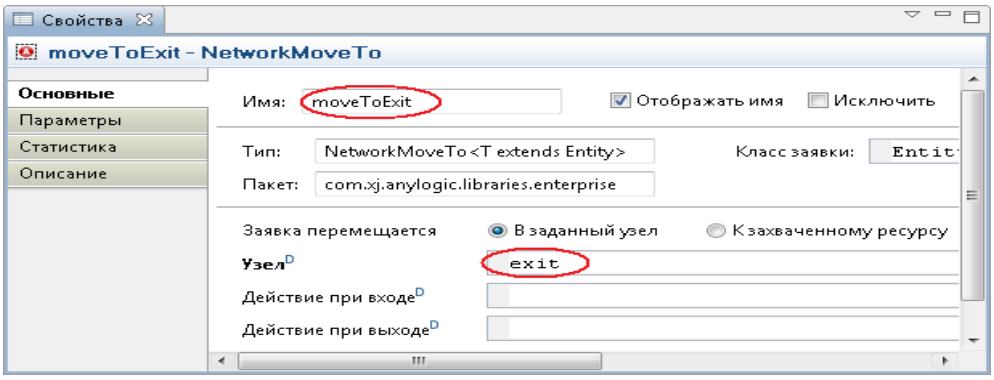


Рис. 5.23. Перемещение пациентов

Объект **NetworkExit** удаляет заявку из сети. Заявка при этом перестает отображаться на анимации сети. В нашем случае он моделирует уход пациентов из отделения. Оставьте свойства объекта **NetworkExit** без изменений. Объект **Sink** уничтожает поступившие заявки. Обычно он используется в качестве конечной точки диаграммы процесса. Оставьте свойства объекта **Sink** без изменений.

Шаг 10. Запуск модели.

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

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

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

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

Запустите модель

1. Щелкните мышью по кнопке панели инструментов **Запустить** и выберите из открывшегося списка эксперимент, который Вы хотите запустить.

В дальнейшем по нажатию на кнопку **Запустить** (или по нажатию F5) будет запускаться тот эксперимент, который запускался Вами в послед-

ний раз. Чтобы выбрать какой-то другой эксперимент, Вам будет нужно щелкнуть мышью по стрелке, находящейся в правой части кнопки **Запустить** и выбрать нужный Вам эксперимент из открывшегося списка (или щелкнуть правой кнопкой мыши по этому эксперименту в панели **Проекты** и выбрать **Запустить** из контекстного меню).

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

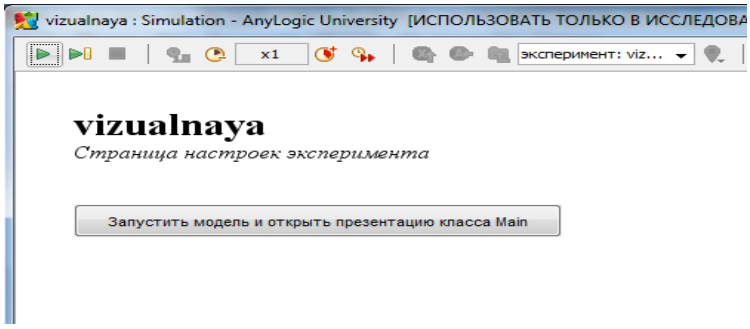
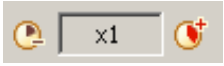


Рис. 5.24. Интерфейс модели

Щелкните по этой кнопке. Тем самым Вы запустите модель и перейдете к презентации корневого класса активного объекта запущенного эксперимента.

Вы увидите, что поведение модели совпадает с ожидаемым: пациенты прибывают в приемный покой, затем проходят в процедурную, которую они покидают после прохождения процедуры.

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

.

Шаг 11. Сбор статистических данных.

AnyLogic предоставляет пользователю удобные средства для сбора статистики по работе блоков диаграммы процесса. Цель моделирования такой системы может быть разной. Однако наиболее типичной целью исследования в подобных задачах массового обслуживания является оценка эффек-

тивности системы, т. е. нахождение числовых значений характеристик, описывающих качество обслуживания системой потока посетителей.

Source. Объект Основной библиотеки. Создает заявки. Обычно используется в качестве начальной точки потока заявок. Заявки могут быть либо базового для заявок класса Entity, либо любого класса пользователя, унаследованного от этого базового класса. Вы можете сконфигурировать объект так, чтобы он создавал заявки других типов, указав конструктор нужного класса в параметре Новая заявка, а также задать действие, которое должно выполняться перед тем, как новая заявка покинет объект, и связать с заявкой определенную фигуру анимации.

Сбор статистики использования ресурсов.

Queue. Объект Основной библиотеки. Объект Queue моделирует очередь заявок, ожидающих приема объектами, следующими за данным в потоковой диаграмме, или же хранилище заявок общего назначения. При необходимости Вы можете задать максимальное время ожидания заявки в очереди. Вы также можете программно извлекать заявки из любых позиций в очереди.

Delay. Объект Основной библиотеки. Задерживает заявки на заданный период времени. Время задержки вычисляется динамически, может быть случайным, зависеть от текущей заявки или от каких-то других условий. Это время может, в частности, вычисляться как длина фигуры, заданной в качестве фигуры анимации этого объекта, поделенной на "скорость" заявки.

Одновременно могут быть задержаны сразу несколько заявок (не более заданной вместимости объекта capacity). Заявки задерживаются независимо друг от друга – время задержки вычисляется отдельно для каждой заявки.

Sink. Объект Основной библиотеки. Уничтожает поступившие заявки. Обычно используется в качестве конечной точки потока заявок. Для того, чтобы заявки удалялись из модели и уничтожались, нужно соединить выходной порт последнего блока процессной диаграммы с портом объекта Sink или Exit.

Для успешного уничтожения заявки необходимо выполнение трех условий:

- если заявка находится в сети, то она должна быть удалена из этой сети с помощью объекта NetworkExit;
- заявка не должна обладать ни одним ресурсом или сетевым ресурсом;
- если заявка содержит другие заявки, то они тоже должны удовлетворять вышеуказанным условиям;

- если какое-то из этих условий не выполняется, объект Sink выдает ошибку.

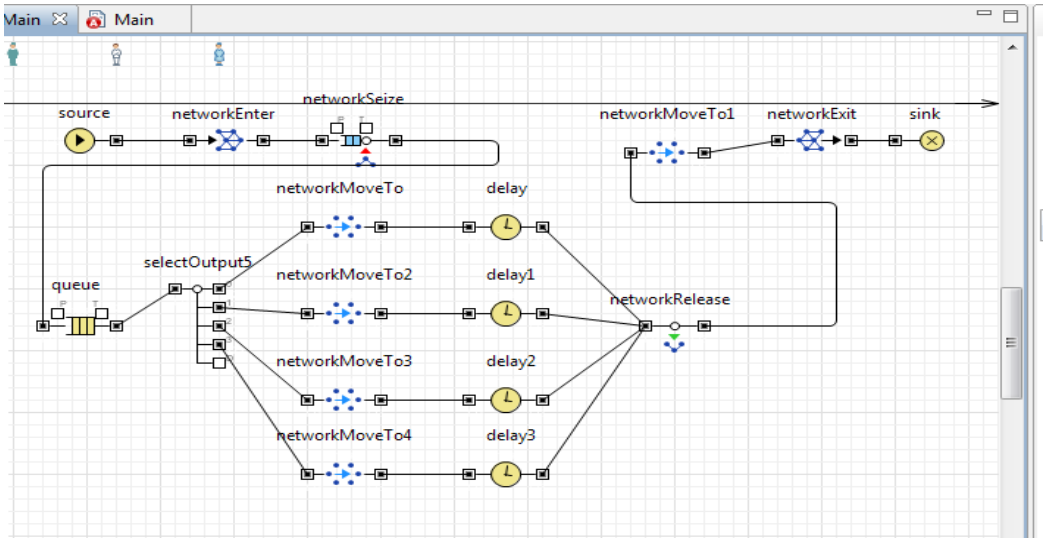


Рис. 5.25. Модель клиники

AnyLogic предоставляет пользователю удобные средства для сбора статистики по работе блоков диаграммы процесса. Объекты Основной библиотеки самостоятельно проводят сбор основной статистики. Все, что Вам нужно сделать – это включить сбор статистики для объекта.

Поскольку мы уже сделали это для объектов delay, то теперь мы можем, например, просмотреть интересующую нас статистику (скажем, статистику занятости стоматолога и длины очереди) с помощью диаграмм.

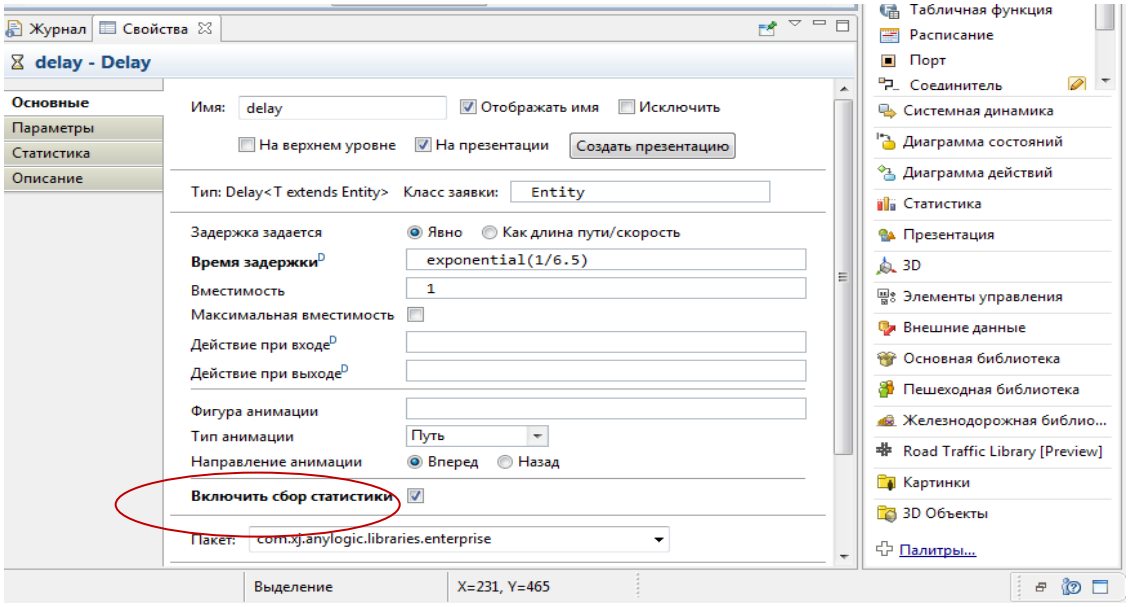


Рис. 5.25. Формирование занятости стоматолога

Добавьте диаграмму для отображения средней занятости кассира.

Откройте палитру Статистика. Эта палитра содержит элементы сбора данных и статистики, а также диаграммы для визуализации данных и результатов моделирования.

Перетащите элемент Столбиковая диаграмма из палитры Статистика на диаграмму класса и измените ее размер.

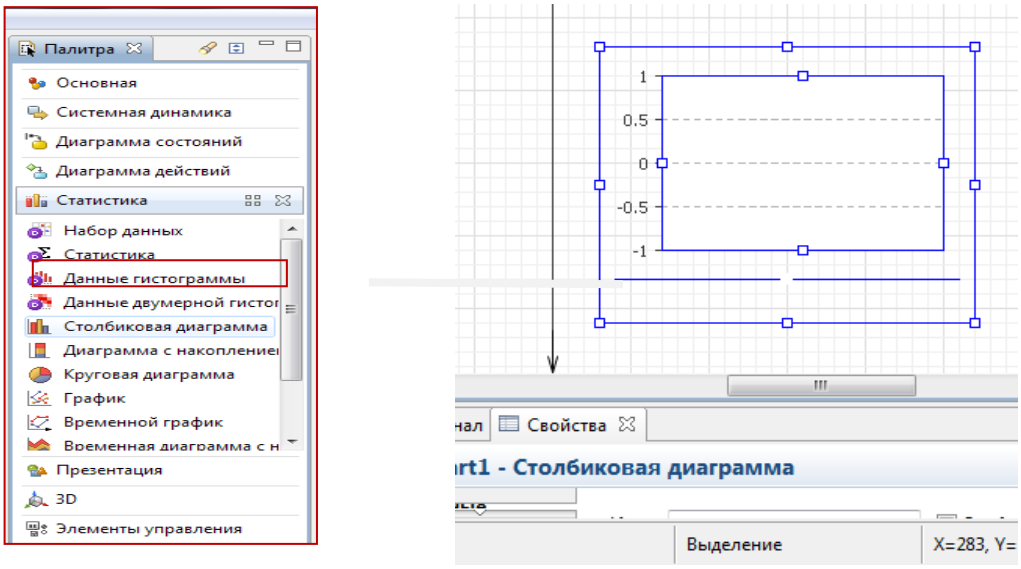


Рис. 5.26. Формирование столбиковой диаграммы

Перейдите на страницу Основные панели Свойства. Щелкните мышью по кнопке Добавить элемент данных. При этом появится секция свойств того элемента данных, который будет отображаться на этой диаграмме.

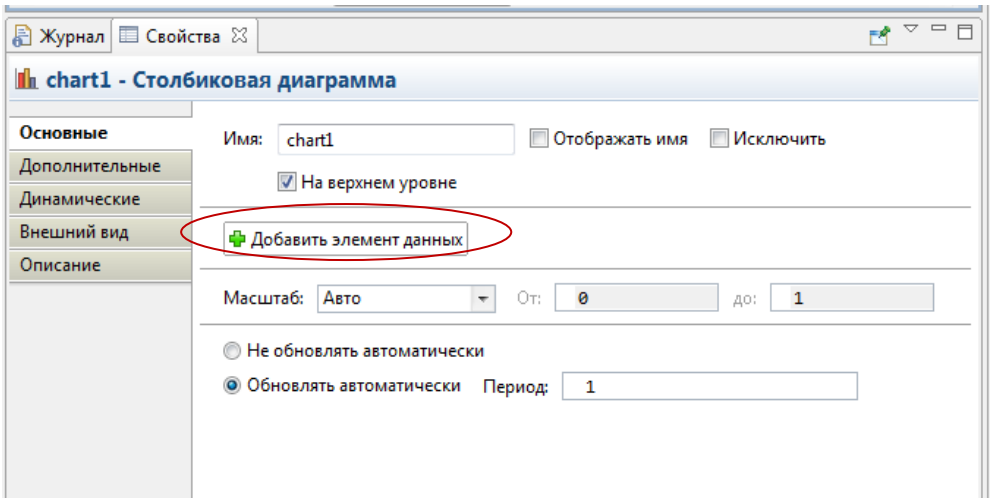


Рис. 5.27. Формирование столбиковой диаграммы

Введите delay.statsUtilization.mean() в поле Значение. Здесь delay – это имя нашего объекта Delay. У каждого объекта Delay есть встроенный набор данных statsUtilization, занимающийся сбором статистики использования этого объекта. Функция mean() возвращает среднее из всех измеренных этим набором данных значений. Вы можете использовать и другие методы сбора

статистики, такие, как min() или max(). Полный список методов можно найти на странице документации этого класса набора данных: StatisticsContinuous .

Перейдите на страницу Внешний вид панели Свойства. Выберите первую опцию из набора кнопок Расположение, чтобы изменить расположение легенды относительно диаграммы (мы хотим, чтобы она отображалась справа).

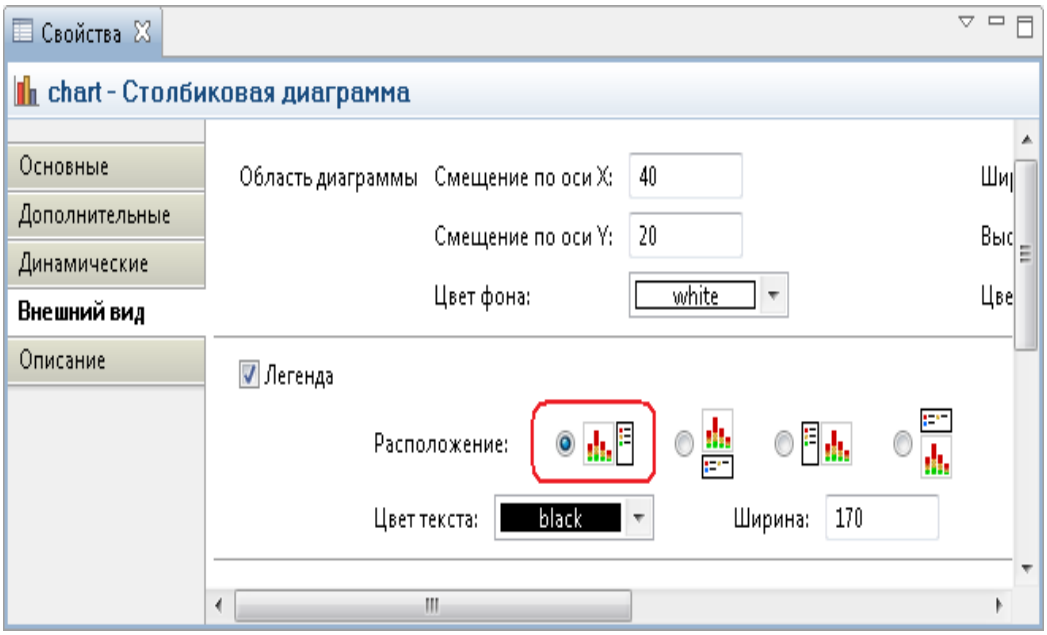


Рис. 5.25. Формирование столбиковой диаграммы

После чего добавляем еще 3 объекта delay1, delay2, delay3. Получаем вид

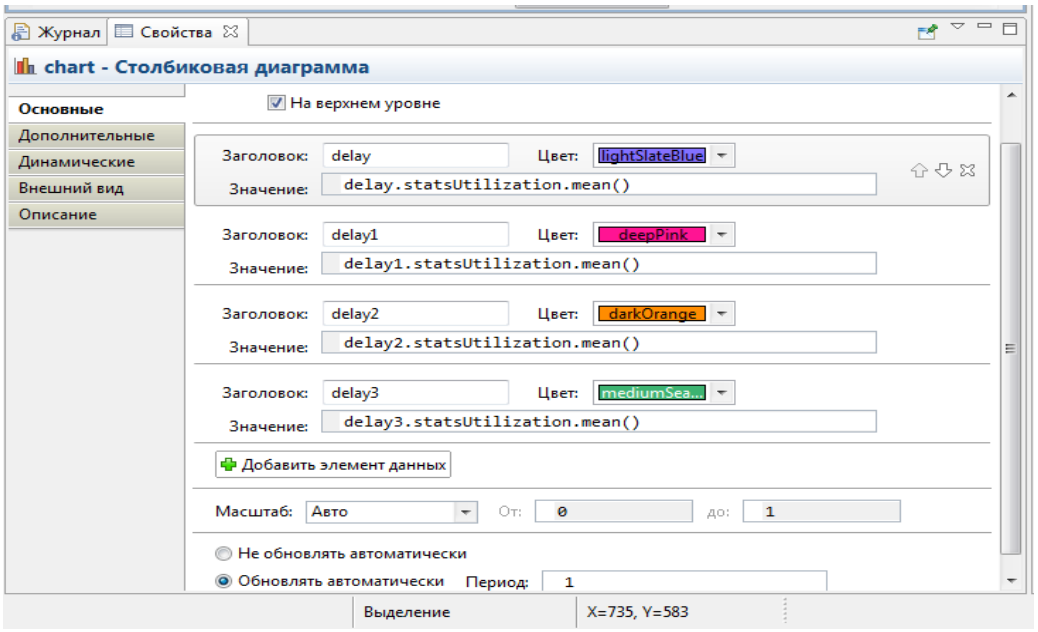


Рис. 5.29. Формирование столбиковой диаграммы

При запуске модели мы видим интересующую нас статистику (скажем, статистику занятости стоматологов и длины очереди) с помощью диаграммы.

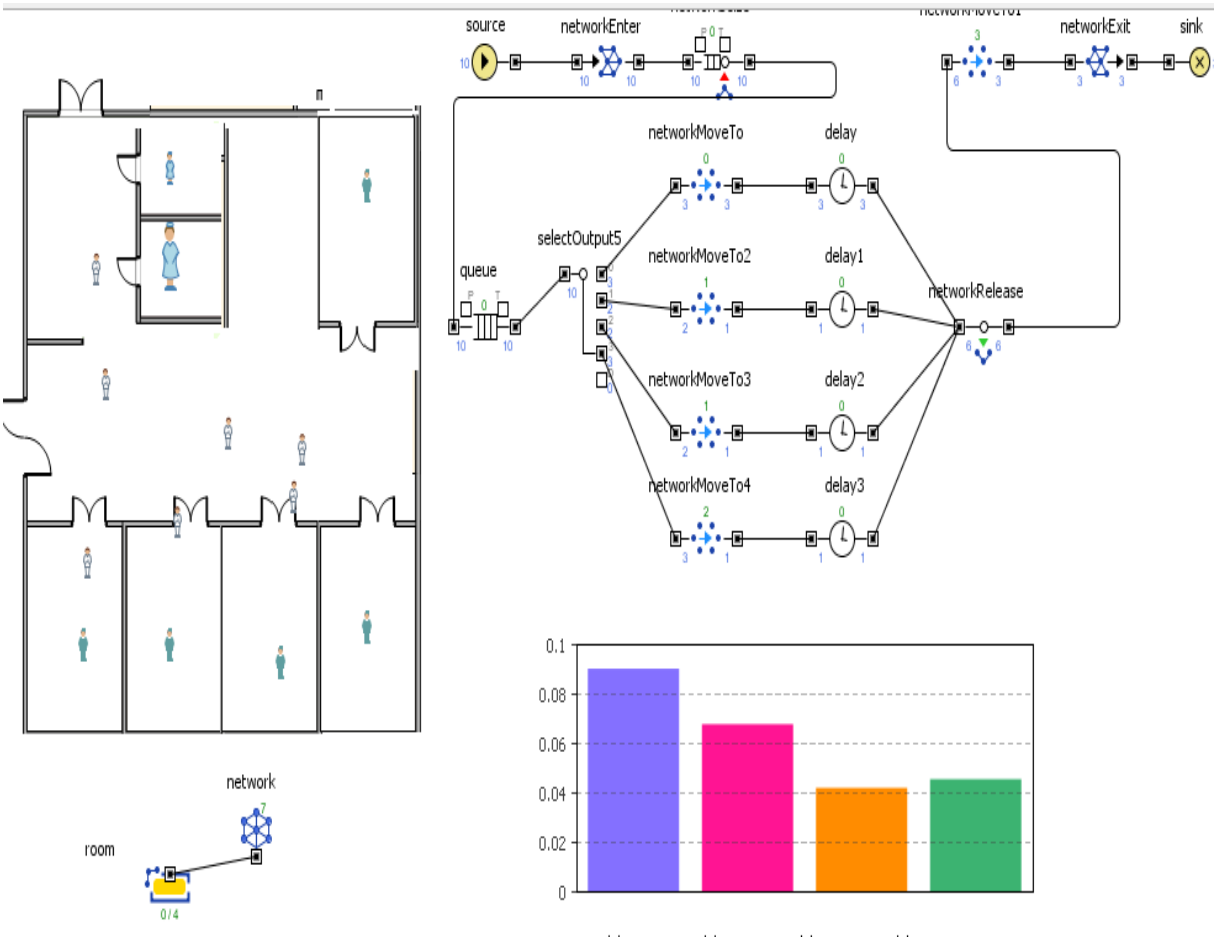


Рис. 5.30. Анимационная модель, структурная диаграмма и статистика клиники

ГЛАВА 6

ДИСКРЕТНО-СОБЫТИЙНОЕ МОДЕЛИРОВАНИЕ ТРАНСПОРТНЫХ ПОТОКОВ И СЕТЕЙ

Актуальность изучения транспортных потоков и сетей обусловлена тем, что в современном мире автомобиль и любой другой транспорт стал неотъемлемой частью жизни человека, количество транспорта увеличивается с каждым днем. В связи с этим остро встает проблема регулирования перекрестков, заторов и пробок на дорогах на различных типах транспортных развязок, т.е. появляются проблемы, связанные с дорожным движением. Проблема заторов и пробок, к сожалению, знакома уже не только жителям мегаполисов. Пробки и заторы возникают из-за разных причин: аварии на полосе дороги, в результате которой уменьшается пропускная способность участка дороги; неправильная развязка перекрестков; плохая организация или неэффективное регулирование движения на перекрестках [27, 96, 128].

Типовой пример такой неэффективности – плохо настроенные светофорные фазы. Часто бывает так, что драгоценные «зеленые» секунды включаются тогда, когда они никому не нужны. И наоборот, красный свет заставляет резко тормозить большое количество приближающихся к перекрестку машин. Такие светофоры легко могут превратить езду по прямой и широкой улице в мучение, заставляя то и дело тормозить, каждый раз сокрушаясь о потерянных секундах.

Такой режим езды ударяет не только по нервам водителей, но и по их кошельку, ведь при разгоне тратится в 2-3 раза больше топлива, чем при езде с постоянной скоростью.

Для устранения этих проблем расширяют дороги, модернизируют методы регулирования перекрестков с помощью светофоров, но ситуация все равно не меняется. Поэтому для решения задач такого типа нужно использовать средства имитационного моделирования. Имитационное моделирование применяется к процессам, в ход которых может время от времени вмешиваться человек. Человек, руководящий операцией, может в зависимости от сложившейся обстановки принимать те или иные решения. Затем приводится в действие математическая модель, которая показывает, какое ожидается изменение обстановки в ответ на это решение и к каким последствиям оно приведет спустя некоторое время. Следующее «текущее решение» принимается уже с учетом реальной новой обстановки и т.д. В результате многократного повторения такой процедуры руководитель как бы «набирает опыт»,

учится на своих и чужих ошибках и постепенно приобретает опыт принимать правильные решения. Разработка различных имитационных моделей транспортных развязок, максимально приближенных к реальным ситуациям, позволит в той или иной степени решить некоторые проблемы дорожных развязок и транспортных потоков. Более того, в настоящее время задача моделирования транспортных потоков и дорожных развязок является все более актуальной и эта проблема обсуждается в ряде конференций по моделированию сложных систем (ИММОД, КОММОД, ПИТ и др.). Для моделирования транспортных сетей и потоков используются различные среды имитационного моделирования:VISUM, GPSS, AnyLogic и др. Для моделирования транспортных сетей г.Казани при подготовке к Универсиаде -2012 разработчики компании «Элина-Компьютер» (www.elina-computer.ru) использовали собственную разработку - расширенный редактор на базе среды объектно-ориентированного программирования GPSS World, в который входит текстовый редактор GPSS, графический редактор моделей, редактор форм и генератор отчетов [46]. Для решения задач транспортных развязок и потоков в них наиболее удобной системой моделирования является среда AnyLogic. В этой главе мы покажем некоторые возможности этой среды для потоков машин на транспортных развязках.

6.1. Модель дорожного перекрестка

Рассмотрим определенную транспортную развязку, например, перекресток. Через перекресток движутся потоки автомобилей. Для упорядочивания движения служит автоматический светофор или регулировщик.

Создадим модель Дорога на основе шаблона Дискретно-событийное моделирование. AnyLogic создаст потоковую диаграмму, состоящую из 4-х элементов: source, queue, delay и sink.

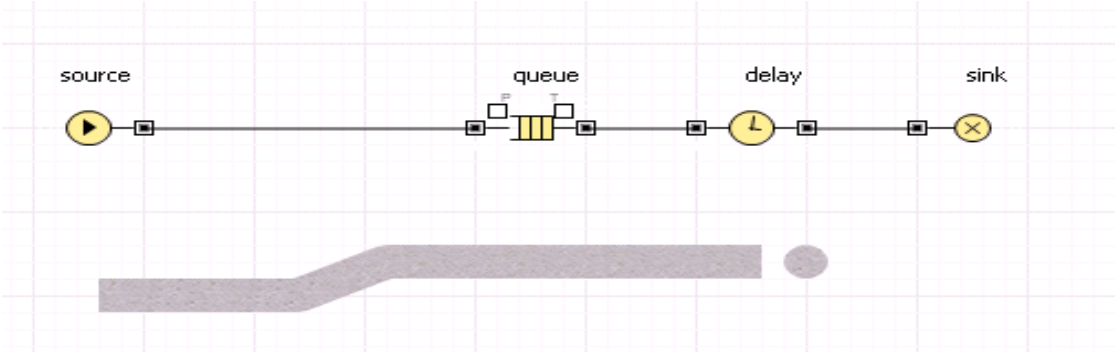


Рис. 6.1. Потоковая диаграмма

Source – создает заявки в настраиваемые моменты времени. Объект Source не разрешает заявкам храниться в буфере выходного порта, если они

не могут покинуть объект. Поэтому в случае, если за объектом source расположен объект, который по той или иной причине не может принять новую заявку, нужно между ними поставить специальный объект буферизации, например, queue.

Queue – хранит заявки в определенном порядке. Моделирует очередь заявок, ожидающих приема объектами, следующими за ним в потоковой диаграмме.

Delay – задерживает заявки на заданный период времени. Время задержки вычисляется динамически, может быть случайным или зависеть от каких-то других условий. Это время может вычисляться как длина фигуры анимации этого объекта, поделенной на "скорость" заявки.

Sink – уничтожает поступившие заявки. Обычно используется в качестве конечной точки потока заявок.

Автомобили подъезжают к перекрестку в произвольные моменты времени, поэтому объект source также должен создавать заявки в случайные моменты времени. Для этого в строке свойств объекта source **Заявки прибывают согласно** нужно выбрать **Время между прибытиями**. В появившемся поле ввода времени между прибытиями запишите exponential(0.1). Функция exponential генерирует реализацию случайной величины с экспоненциальным законом распределения.

В свойствах объекта queue следует удалить связь с анимацией: в поле Фигура анимации панели свойств, удалите ссылку на polyline. Заявки, находящиеся в очереди, анимироваться не будут. Этот объект будет выполнять буферную функцию для предотвращения ошибки в момент, когда source создал заявку, а delay еще не готов ее принять.

В свойствах объекта delay следует сделать следующие изменения:

- задержка задается – укажите: как длина пути/скорость;
- время задержки – исправьте на triangular(6., 10., 16.);
- вместимость – исправьте на 10;
- фигура анимации – исправьте на polyline;
- тип анимации – исправьте на Путь.

Двигаясь в 3 потока, автомобили должны останавливаться перед стоплинией на красный сигнал светофора. Для моделирования движения с остановкой подходит объект Conveyor, который перемещает заявки по пути заданной длины с заданной скоростью (одинаковой для всех заявок), сохраняя их порядок и оставляя заданные промежутки между ними. Когда заявка достигает конца конвейера, но не может его покинуть, то она там и останется.

Если конвейер – накапливающий, то он продолжит двигать заявки, которые имеют достаточно свободного места перед собой.

Перенесите из палитры Enterprise Library на диаграмму класса Main объект conveyor и поместите его между queue и delay.

В свойствах объектов conveyor следует сделать следующие изменения:

- длина задается – укажите: согласно пути,
- расстояние между заявками – укажите: 55,
- фигура анимации – укажите на соответствующую ломаную.

Для моделирования остановки автомобилей на красный свет необходимо в диаграмму установить элемент, останавливающий движение заявок – hold. Этот объект блокирует/разблокирует поток заявок на определенном участке блок-схемы. Если объект находится в заблокированном состоянии, то заявки не будут поступать на его входной порт и будут ждать, пока объект не будет разблокирован. Поставьте объект hold после conveyor.

Для определенного порядка передвижения заявок после светофора нужно поставить еще один conveyor после объекта hold. Тогда наша диаграмма примет следующий вид:

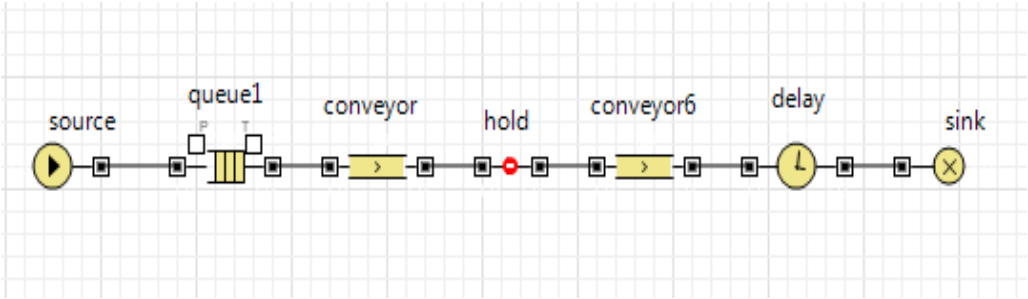


Рис. 6.2. Потоковая диаграмма

Для анимации движения заявок (автомобилей) по конвейеру (зоне остановки перед светофором) нужно нарисовать ломаные линии и связать их с соответствующим объектом conveyor.

Каждая линия движения у нас будет состоять из двух ломаных линий. Точка соединения ломаных будет находиться в месте остановки перед светофором. Для каждой линии движения у нас будет своя потоковая диаграмма. Тогда в первой диаграмме в conveyor перед объектом hold нужно указать фигура анимации polyline1, а для conveyor после объекта hold polyline11.

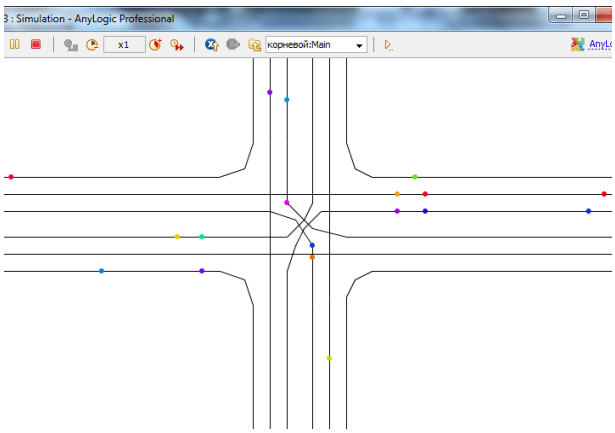


Рис. 6.3. Запуск модели

Чтобы вместо точек в модели были машины, нужно из палитры 3D Объекты перетащить элемент «Автомобиль». А в элементе Source прописать фигуру анимации car.

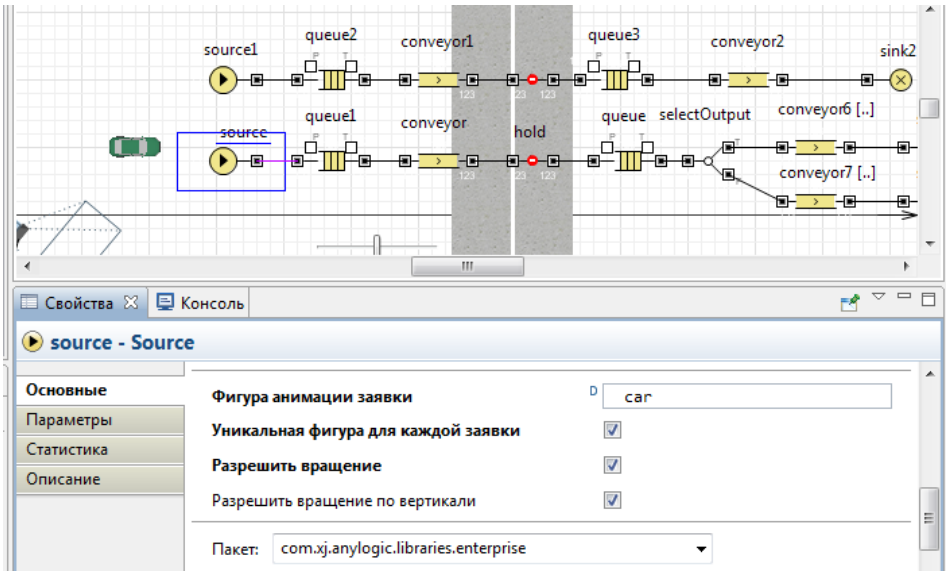


Рис. 6.4. Элемент Source

Также нужно изменить линии, чтобы они были больше похожи на дорогу. Нужно изменить цвет линии и толщину. Цвету зададим текстуру, а толщину зададим 30.

Построение модели светофора. Построим модель регулируемого дорожного движения со светофором, разрешающим или запрещающим движение транспорта.

Светофор, регулирующий движение автомобилей на пешеходном переходе, может находиться в следующих состояниях: движение транспорта разрешено (зеленый), приготовиться к запрещающему сигналу (мигающий зеленый), приготовиться к остановке (желтый), движение запрещено (красный) и приготовиться к движению (красный и желтый).

Светофор работает в автоматическом режиме. В каждом состоянии светофор находится определенный постоянный период времени.

На диаграмму класса активного объекта Model поместим Начало диаграммы состояний из панели Диаграмма состояний и назовем ее Для_автомобилей.

Для того чтобы построить стейтчарт, следует использовать элементы из палитры Диаграмма состояний.

Состояние внимание представим гиперсостоянием с парой переключающихся элементарных состояний: в одном из них зеленый горит (состояние А), в другом – нет (состояние В). Постройте эти состояния и соедините их переходами, как показано на рисунке 6.5.

В соответствии с алгоритмом работы светофора помимо начального состояния в модель нужно ввести дополнительные состояния (рис. 6.6). Начальное состояние назовите движение – движение автомобилям разрешено (зеленый свет), затем светофор переходит в состояние внимание – внимание (мигающий зеленый), медленно – приготовиться к остановке (желтый свет), остановка транспорта stop – запрет движения (красный свет) и приготовиться – приготовиться к движению (красный и желтый свет горят одновременно).

В каждом состоянии светофора должен гореть определенный сигнал: в состоянии движение должен гореть зеленый, в состоянии приготовиться должны гореть красный и желтый одновременно и т. п. Создайте три параметра логического типа: красный, желтый и зеленый, которые будут принимать истинное значение тогда, когда у светофора горит соответствующий сигнал: красный, желтый или зеленый (рис. 6.6). Начальные значения этих булевых параметров можно не задавать: по умолчанию они будут равны false. Мы создали стейтчарт именно для управления значениями этих параметров, каждое состояние отвечает за зажигание своего света или их комбинации. Например, в состоянии медленно должен гореть желтый, в состоянии stop должен загореться красный свет, а в состоянии приготовиться должны гореть красный и желтый одновременно.

1. В свойствах состояния движение в поле Действие при входе запишите зеленый = true, а в поле Действие при выходе запишите зеленый = false;

2. То же самое запишите для состояния В гиперсостояния внимание, а у состояния А эти поля нужно оставить пустыми – когда светофор находится в этом состоянии, он не горит.

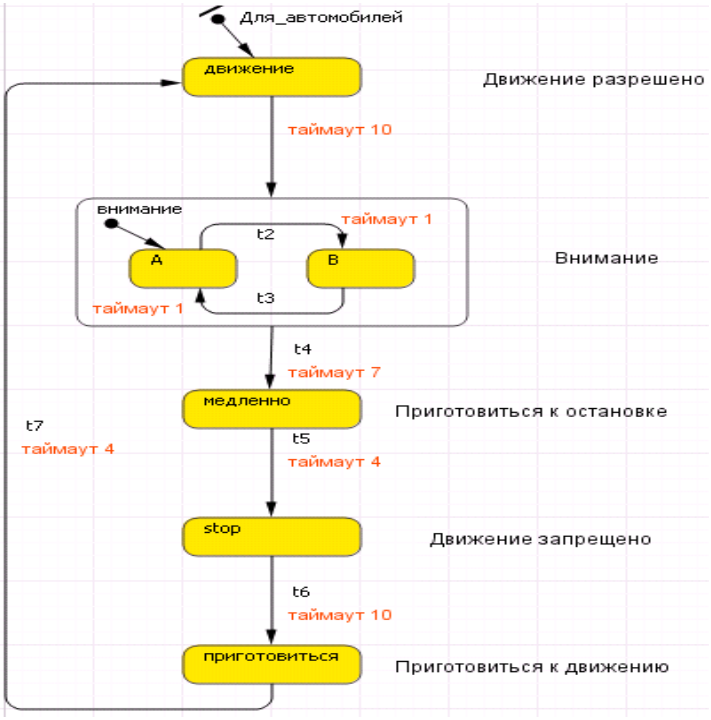


Рис. 6.5. Модель светофора

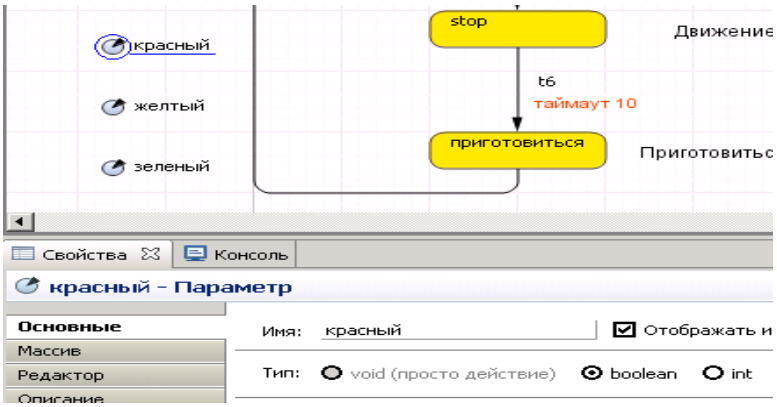


Рис. 6.6. Параметры

3. Аналогично, в состоянии медленно нужно включить желтый сигнал, т. е. при входе в это состояние установить параметр желтый в true, а при выходе из этого состояния установить его в false.

4. Для состояния stop аналогично опишите состояния параметра красный, а для состояния приготовиться оба параметра красный и желтый нужно установить в true при входе, и в false при выходе.

Презентация модели рисуется в той же диаграмме (в графическом редакторе), в которой задается и диаграмма моделируемого процесса. Графические объекты цвета сигналов светофора в презентации имеют динамические параметры, все остальные – статические. Светофор строится из трех овалов. Установим динамическое значение цвета верхнего сигнала светофора: если переменная красный истинна, то цвет должен быть red (красный), в

противном случае его цвет нужно установить gray (серый). Это записывается следующим условным выражением на языке Java:

красный? red: gray

Цвет среднего и нижнего овалов, следует установить в поле их динамических значений соответственно так:

желтый? yellow: gray

зеленый? green: gray

red, yellow, green и gray – предопределенные константы, обозначающие стандартные цвета.

Для состояний Движение и stop пропишем команды для активации объектов hold.

Изменение интенсивности потока. Для того чтобы было возможно изменять интенсивность потока, необходимо добавить элемент «бегунок» и параметр «par» (рис. 6.7).

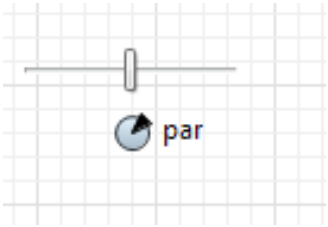


Рис. 6.7. Элемент бегунок и параметр

Элементу «бегунок» необходимо изменить свойства (рис. 6.8). Его нужно связать с параметром «par», а также задать максимальное и минимальное значение.

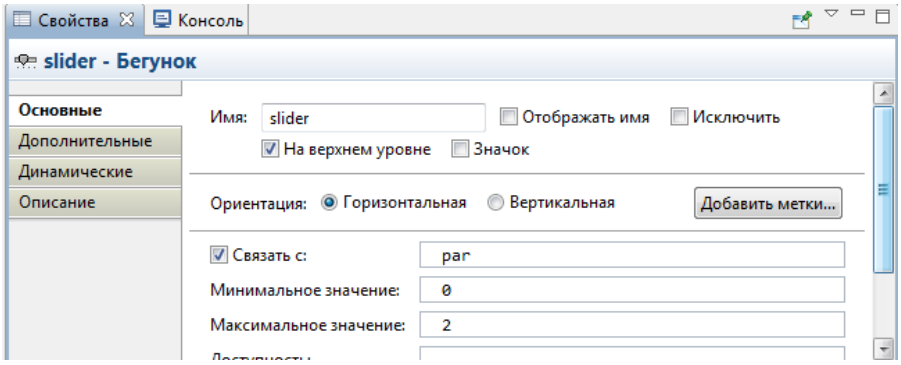


Рис. 6.8. Свойства бегунка

Параметру «par» нужно назначит тип double. Используя элемент «Текст» палитры «Презентация», задайте подпись к ползунку, это позволит отображать текущее значение.

Размещения графиков. Чтобы представить процесс очереди машин и время пребывания, разместим два графика. Первый график отображает среднее значение машин в очереди. Для размещения этого графика нужно ис-

пользовать палитру «Статистика» и элемент «Столбиковая диаграмма». (рис. 6.9).

Рис. 6.9. Столбиковая диаграмма

- Добавьте элемент данных. Задайте подпись «Длина очереди». Выберите цвет, а затем в качестве значения задайте выражение:
- queue.statsSize.mean()
- Здесь метод mean() – возвращает среднюю длину очереди.

При вводе выражения можно использовать помощник AnyLogic. Для этого следует нажать комбинацию клавиш CTRL + SPACE. После ввода выражения нужно сменить ориентацию графика. Нужно открыть вкладку «Внешний вид» и изменить направление столбцов.

- Для второго графика нам необходимо создать класс заявки. Класс содержит два глобальных атрибута:
- enteredSystem – время появления клиента в системе.

startWaiting – время начала ожидания обслуживания.

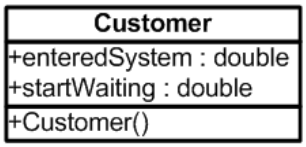


Рис. 6.10. Класс для учета времени.

Размещение класса выполняется путем вызова диалога, показанного на рисунке. Для этого следует получить контекстное меню для корневого класса на дереве проекта и выполнить команду «Создать > Java класс».

В результате открывается диалог создания класса. На первом шаге нужно задать имя класса и имя базового класса, который будет расширять новый класс. При работе с заявками в качестве такого класса выступает системный класс com.xj.anylogic.libraries.enterprise.Entity. Он представляет собой абстрактную заявку (рис. 6.11).

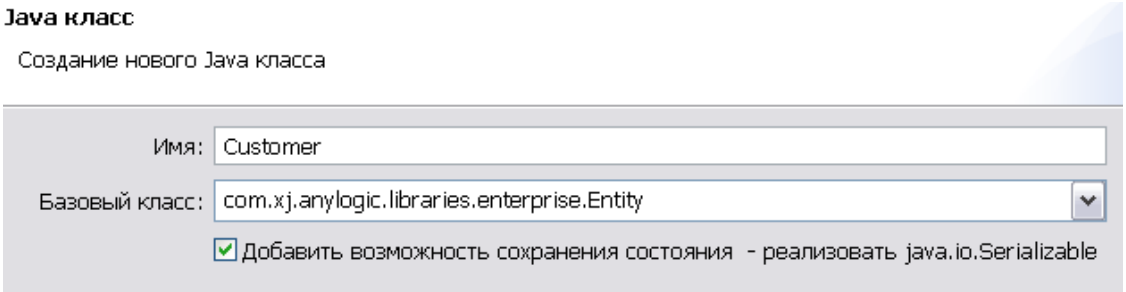


Рис. 6.11. Создание класса, первая фаза

Затем следует перейти ко второму шагу и задать поля класса – атрибуты, так как это показано на рисунке 6.12.

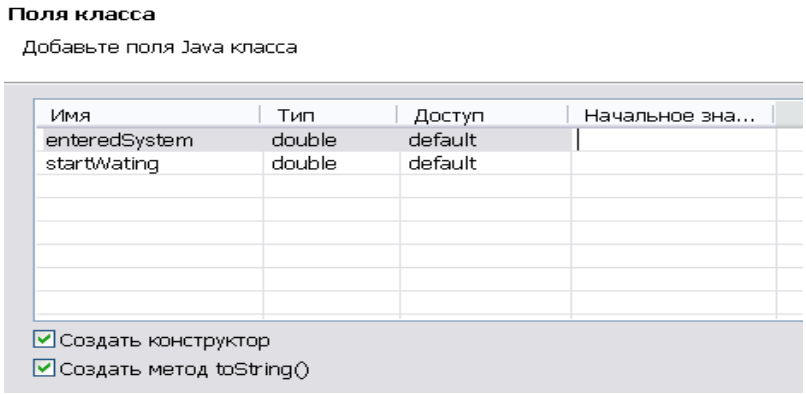


Рис. 6.12. Создание класса, завершение

- В результате будет создан класс Java, код которого можно открыть и отредактировать при необходимости.

Затем разместите в модель из палитры «Статистика» элемент «Гистограмма» и элемент «Данные гистограммы». Гистограмма будет отображать распределение времени в системе на основе данных, собранных в элементе «Данные гистограммы». Назовите этот элемент timeInSystemDistr (распределение времени пребывания в системе).

- Откройте гистограмму, нажмите кнопку «Добавить данные». Задайте заголовок «Время в системе», в поле «Данные» задайте timeInSystemDistr, подберите цвет гистограммы. При настройке гистограммы должен быть активен флажок «Отображать плотность вероятности».

Откройте элемент Source и измените его настройки. На вкладке «Основные» установите свойства:

- класс заявки: Customer
- новая заявка: new Customer()
- действие при выходе: entity.enteredSystem=time().
- с помощью оператора new получают новый экземпляр заявки. Для обращения к атрибутам экземпляра служит указатель entity. При покидании заявки источника фиксируется время выхода.

Откройте элемент Conveyor и установите параметры.

Класс заявки: Customer.

Действие при входе: timeInSystemDistr.add(time()-entity.enteredSystem)

Данные настройки позволяют зафиксировать время пребывания заявки в системе от входа до ее обработки. Элемент «Данные гистограммы» содержит метод add, с помощью которого добавляют данные.

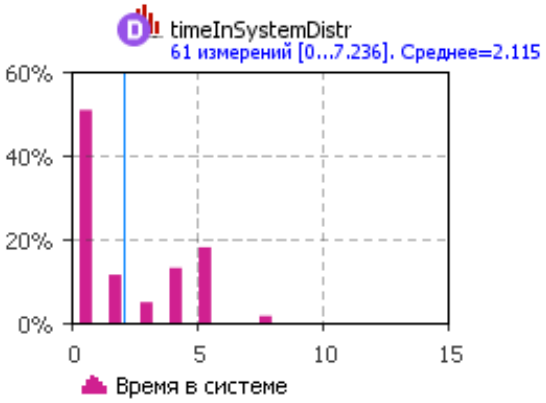


Рис. 6.13. Гистограмма параметров системы

Вертикальная линия отображает среднее значение распределения. Для отображения среднего значения времени откройте последовательно гистограммы и на вкладке «Основные» установите флажок «Отображать среднее», подберите цвет линии.

Время переключения светофора. Для того чтобы было возможно изменять время переключения светофора, необходимо разместить три параметра и три текстовых поля.

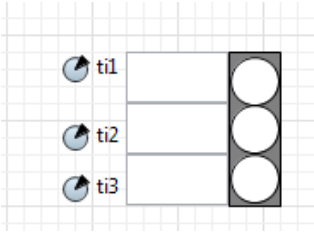


Рис. 6.14. Параметры и тексовые поля

Параметрам нужно задать время и тип int. А текстовое поле связать с соответствующей переменной. Также нужно изменить свойства перехода. Вместо конкретного времени нужно указать соответствующий параметр.

Модернизирую модель, получаем анимационную модель перекрестка (рис.6.15).

С помощью среды AnyLogic можно смоделировать различные перекрестки. Можно построить разное количество линий движений, разное количество дорог, которые пересекаются. Можно также построить целую часть дороги. А с помощью стейтчартов можно задавать различное время для работы определенного цвета светофора. Тем самым можно добиться оптимальной работы светофора, что позволит устранить проблему пробок и облегчит жизнь всем автомобилистам.

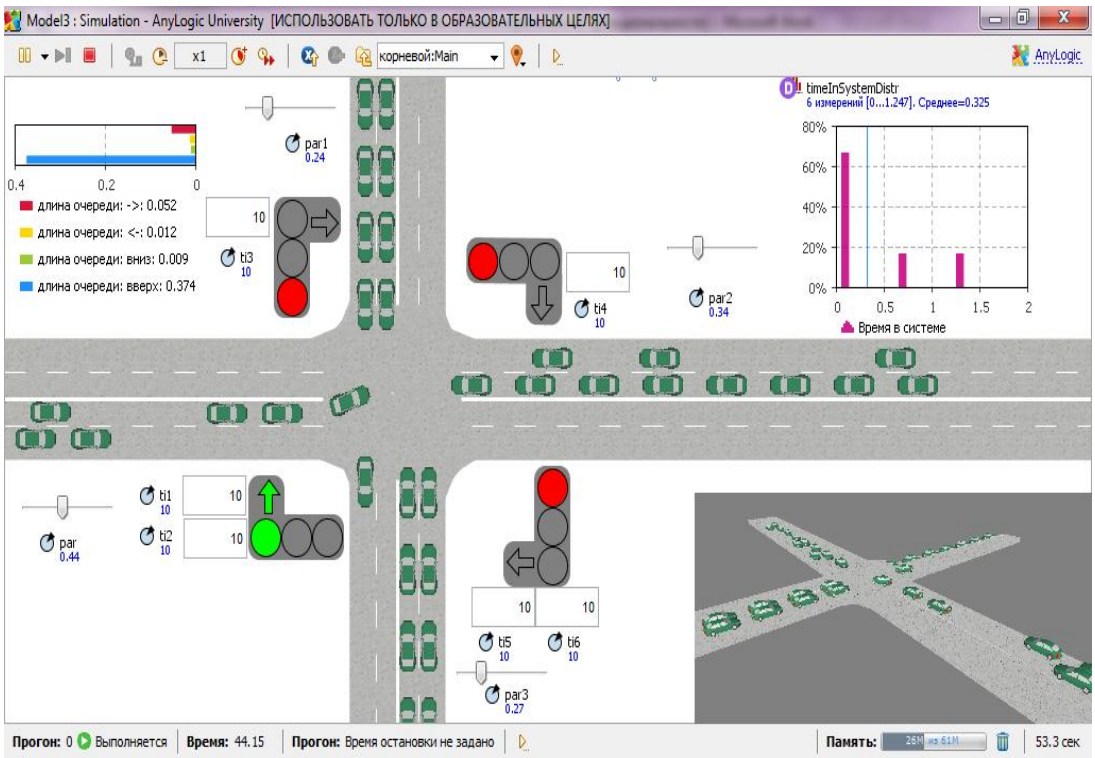


Рис. 6. 15. Интерфейс имитационной модели «Перекресток»

6.2. Модель дорожного движения на трех перекрестках

В качестве реального объекта имитационного моделирования можно выбрать улицу, имеющую три пересечения с другими улицами. По улице и перекресткам движутся потоки автомобилей. Для упорядочивания движения служит автоматический светофор или регулировщик.

Шаг 1. Построение модели

Создадим модель Дорога на основе шаблона Дискретно-событийное моделирование. AnyLogic создаст потоковую диаграмму, состоящую из 4-х элементов: source, queue, hold и sink.

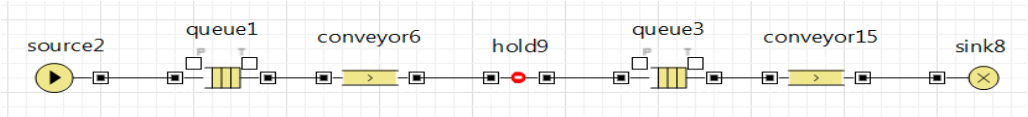


Рис.6.16. Потоковая диаграмма

Source – создает заявки в настраиваемые моменты времени. Объект Source не разрешает заявкам храниться в буфере выходного порта, если они не могут покинуть объект. Поэтому в случае, если за объектом source расположен объект, который по той или иной причине не может принять новую заявку, нужно между ними поставить специальный объект буферизации, например, queue.

Автомобили подъезжают к перекрестку в произвольные моменты времени, поэтому объект source также должен создавать заявки в случайные моменты времени.

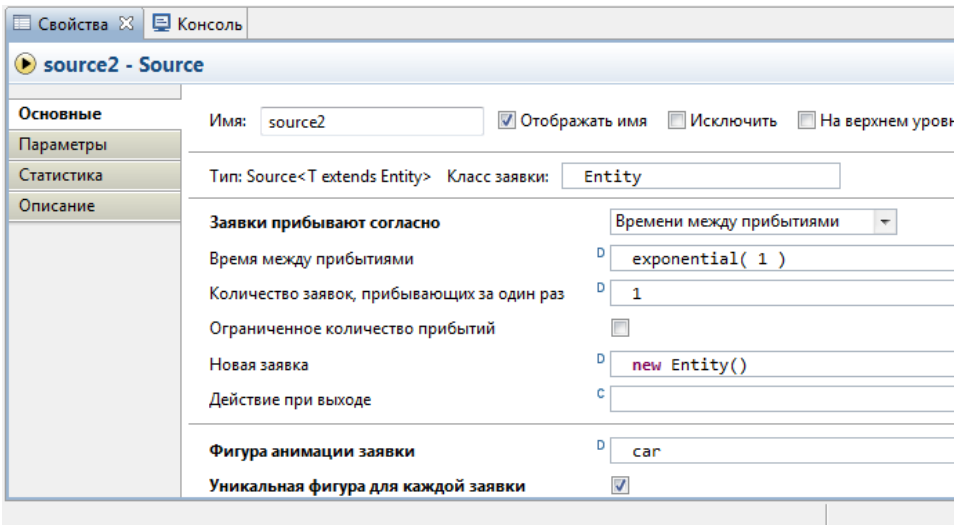


Рис.6.17. Выбор заявок

Шаг 2. Для этого в строке свойств объекта source “Заявки прибывают согласно” нужно выбирать Времени между прибытиями. В появившемся поле ввода времени между прибытиями запишите exponential(1). Функция exponential генерирует реализацию случайной величины с экспоненциальным законом распределения.

Шаг 3. В свойствах объекта queue следует удалить связь с анимацией: в поле Фигура анимации панели свойств, удалите ссылку на polyline. Заявки, находящиеся в очереди, анимироваться не будут. Этот объект будет выполнять буферную функцию для предотвращения ошибки в момент, когда source создал заявку, а delay еще не готов ее принять.

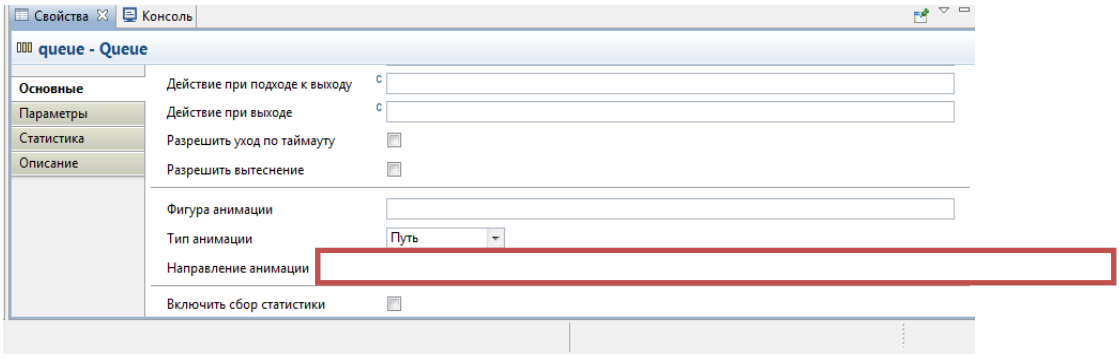


Рис.6.18. Определение очереди

Двигаясь в потоке, автомобили должны останавливаться перед стоп-линией на красный сигнал светофора.

Шаг 4. Для моделирования движения с остановкой подходит объект Conveyor, который перемещает заявки по пути заданной длины с заданной скоростью (одинаковой для всех заявок), сохраняя их порядок и оставляя за-

данные промежутки между ними. Когда заявка достигает конца конвейера, но не может его покинуть, то она там и останется. Если конвейер – накапливающий, то он продолжит двигать заявки, которые имеют достаточно свободного места перед собой.

Шаг 5. Перенесите из палитры Enterprise Library на диаграмму класса Main объект conveyor и поместите его между queue и delay.

В свойствах объектов conveyor следует сделать следующие изменения:

• длина задается – укажите: согласно пути,

• расстояние между заявками – укажите: 55,

• фигура анимации – укажите на соответствующую ломаную.

Шаг 6. Для моделирования остановки автомобилей на красный свет необходимо в диаграмму установить элемент, останавливающий движение заявок – hold. Этот объект блокирует/разблокирует поток заявок на определенном участке блок-схемы. Если объект находится в заблокированном состоянии, то заявки не будут поступать на его входной порт, и будут ждать, пока объект не будет разблокирован. Поставьте объект hold после conveyor.

Шаг 7. Для определенного порядка передвижения заявок после светофора нужно поставить еще один conveyor после объекта hold. Тогда наша диаграмма примет следующий вид:

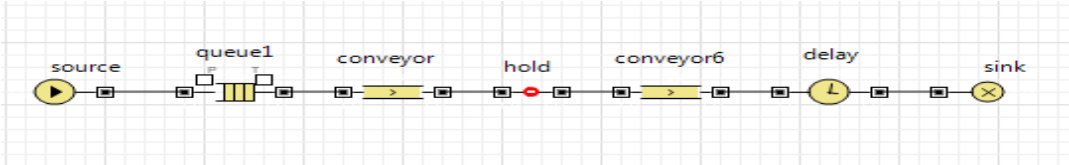


Рис.6.19. Потоковая диаграмма

Каждая линия движения у нас будет состоять из двух ломаных линий. Точка соединения ломаных линий будет находиться в месте остановки перед светофором. Для каждой линии движения будет своя потоковая диаграмма. Для анимации движения заявок (автомобилей) по конвейеру (зоне остановки перед светофором) нужно нарисовать ломаные линии и связать их с соответствующим объектом conveyor.

Как видно на рисунке, не всем линиям подходит наша диаграмма. Линии, которые не поворачивают, нужно разделить, для каждой части линии нужно сделать первоначальную диаграмму. Получаем следующую диаграмму (рис.6.21).

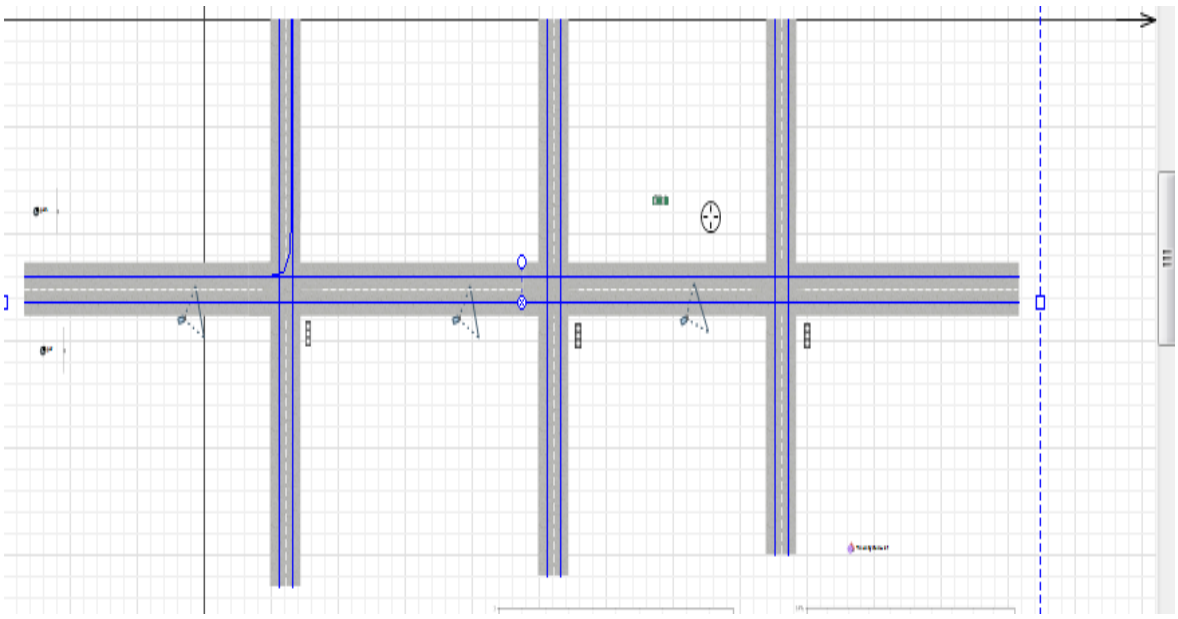


Рис.6.20

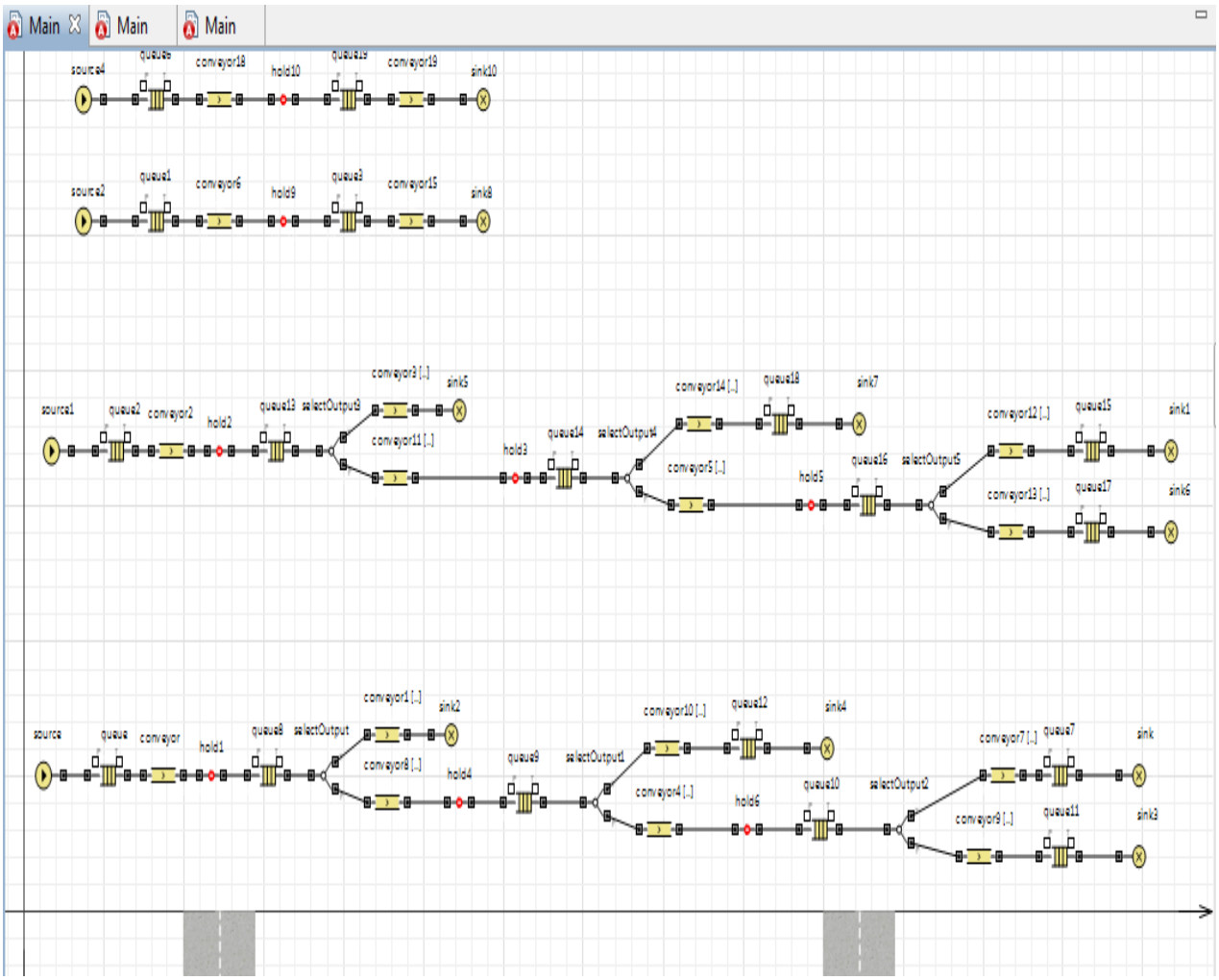


Рис.6.21. Разветвленная потоковая диаграмма

Заметьте, что для любого выделенного объекта внизу появляется панель его свойств, в котором можно изменить параметры и, в частности, имя объекта, если это необходимо. Структурные ошибки при рисовании стейтчарта – повисшие переходы, дублированные указатели начального состояния и т.

п. – выделяются в панели Проекты значком X красного цвета и записью в панели Ошибки. Выделенные переходы должны иметь на концах зеленые точки, если точки белые, это значит, что переход не соединен с состоянием – висит.

Шаг 8. Построение модели светофора

Построим модель регулируемого дорожного движения со светофором, разрешающим или запрещающим движение транспорта. Светофор работает в автоматическом режиме. В каждом состоянии светофор находится определенный постоянный период времени.

На диаграмму класса активного объекта Model поместим Начало диаграммы состояний из панели Диаграмма состояний и назовите ее Для автомобилей.

Для того чтобы построить стейтчарт, следует использовать элементы из палитры Диаграмма состояний.

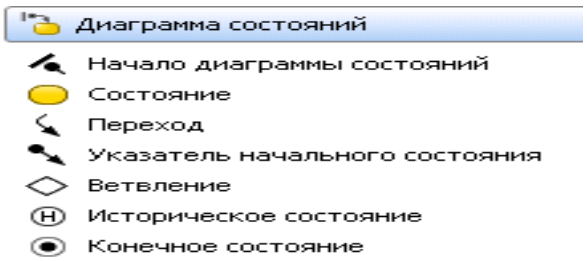


Рис.6.22. Диаграмма состояний

В соответствии с алгоритмом работы светофора помимо начального состояния в модель нужно ввести дополнительные состояния (рис. 6.23.). Начальное состояние назовите движение – движение автомобилям разрешено (зеленый свет), затем светофор переходит в состояние внимание – внимание (мигающий зеленый), медленно – приготовиться к остановке (желтый свет), остановка транспорта stop – запрет движения (красный свет) и приготовиться – приготовиться к движению (красный и желтый свет горят одновременно).

Шаг 9. Состояние внимание представим гиперсостоянием с парой переключающихся элементарных состояний: в одном из них зеленый горит (состояние А), в другом – нет (состояние В). Постройте эти состояния и соедините их переходами, как показано на рисунке ниже. Зададим условия срабатывания переходов. Переходы в нашем автоматическом светофоре выполняются по таймауту, т. е. по истечении интервала времени, который прошел с момента прихода системы в данное состояние.

В состоянии движение светофор находится 10 секунд, 6. затем 7 секунд зеленый сигнал мигает, 3. в состоянии медленно 4 секунды горит желтый, 4.

в течение 10 секунд движение запрещено и 6. 4 секунды светофор находится в состоянии приготовиться.

Рис.6.23. Модель светофора

В каждом состоянии светофора должен гореть определенный сигнал: в состоянии движение должен гореть зеленый, в состоянии приготовиться должны гореть красный и желтый одновременно и т. п. Создайте три параметра логического типа: красный, желтый и зеленый, которые будут принимать истинное значение тогда, когда у светофора горит соответствующий сигнал: красный, желтый или зеленый (рис. 6.24).

Начальные значения этих булевых параметров можно не задавать: по умолчанию они будут равны false. Мы создали стейтчарт именно для управления значениями этих параметров, каждое состояние отвечает за зажигание своего света или их комбинации. Например, в состоянии медленно должен гореть желтый, в состоянии stop должен загореться красный свет, а в состоянии приготовиться должны гореть красный и желтый одновременно.

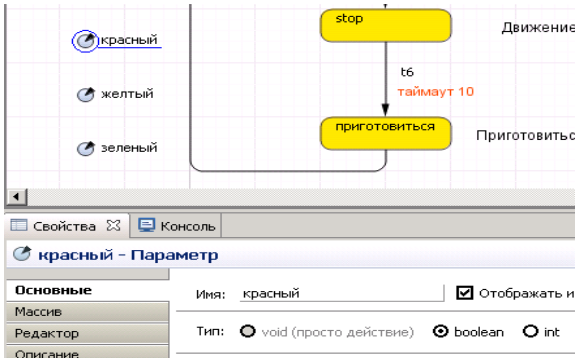


Рис.6.24. Параметры

Шаг 10.

1. В свойствах состояния движение в поле Действие при входе запишите зеленый=true;, а в поле Действие при выходе запишите зеленый = false; (рис.6.25).

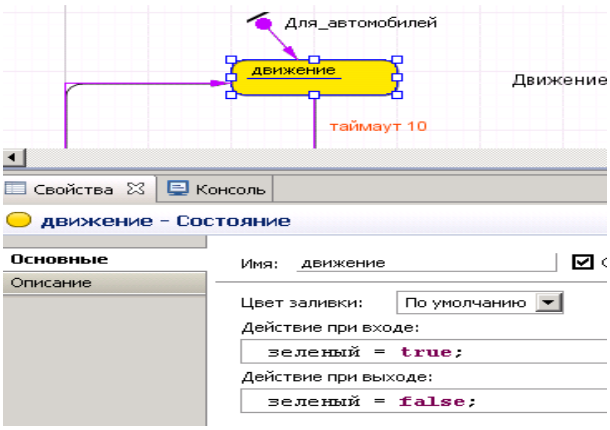


Рис.6.25. Свойства состояния движения

2. То же самое запишите для состояния В гиперсостояния внимание, а у состояния А эти поля нужно оставить пустыми – когда светофор находится в этом состоянии, он не горит.

3. Аналогично, в состоянии медленно нужно включить желтый сигнал, т. е. при входе в это состояние установить параметр желтый в true, а при выходе из этого состояния установить его в false.

4. Для состояния stop аналогично опишите состояния параметра красный, а для состояния приготовиться оба параметра красный и желтый нужно установить в true при входе, и в false при выходе.

Шаг 11. Презентация модели рисуется в той же диаграмме (в графическом редакторе), в которой задается и диаграмма моделируемого процесса. Графические объекты цвета сигналов светофора в презентации имеют динамические параметры, все остальные – статические. Светофор строится из трех овалов. Установим динамическое значение цвета верхнего сигнала светофора: если переменная красный истинна, то цвет должен быть red (красный), в противном случае его цвет нужно установить gray (серый). Это записывается следующим условным выражением на языке Java:

красный? red: gray

Цвет среднего и нижнего овалов следует установить в поле их динамических значений соответственно так:

желтый? yellow: gray

зеленый? green: gray

red, yellow, green и gray – предопределенные константы, обозначающие стандартные цвета.

Для состояний go и stop пропишем команды для активации объектов hold.

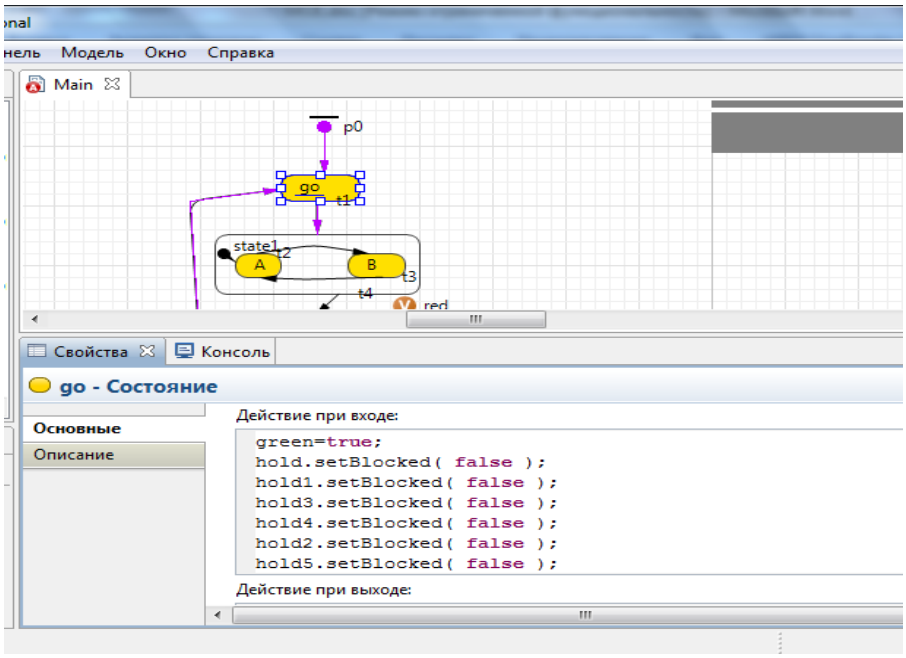


Рис.6.26. Состояние go

Шаг 12. Размещения графиков. Чтобы представить процесс очереди машин и время пребывания, разместим два графика. Первый график отображает среднее значение машин в очереди. Для размещения этого графика нужно использовать палитру «Статистика» и элемент «Столбиковая диаграмма» .

Рис.6.27. Столбиковая диаграмма

Шаг 13. Добавьте элемент данных. Задайте подпись «Длина очереди». Выберите цвет, а затем в качестве значения задайте выражение:

- queue.statsSize.mean()
- здесь метод mean() – возвращает среднюю длину очереди.

При вводе выражения можно использовать помощник AnyLogic. Для этого следует нажать комбинацию клавиш CTRL + SPACE. После ввода выражения нужно сменить ориентацию графика. Нужно открыть вкладку «Внешний вид» и изменить направление столбцов.

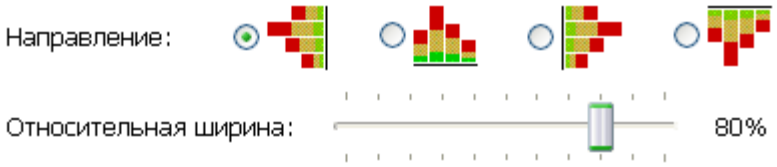


Рис.6.28. Направление столбцов

Шаг 14. Для второго графика нам необходимо создать класс заявки. Класс содержит два глобальных атрибута:

- enteredSystem – время появления клиента в системе;
- startWaiting – время начала ожидания обслуживания.

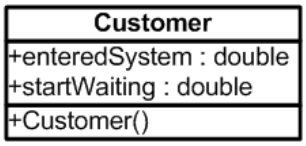


Рис.6.29. Класс для учета времени

Размещение класса выполняется путем вызова диалога, показанного на рисунке. Для этого следует получить контекстное меню для корневого класса на дереве проекта и выполнить команду «Создать > Java класс» (рис. 6.30).

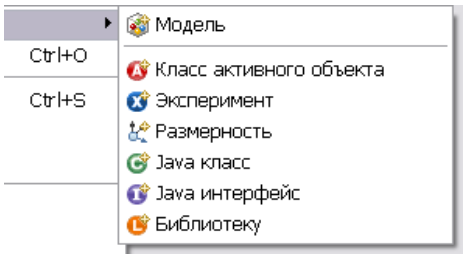


Рис.6.30. Создание класса

В результате открывается диалог создания класса. На первом шаге нужно задать имя класса и имя базового класса, который будет расширять новый класс. При работе с заявками в качестве такого класса выступает системный класс com.xj.anylogic.libraries.enterprise.Entity. Он представляет собой абстрактную заявку.

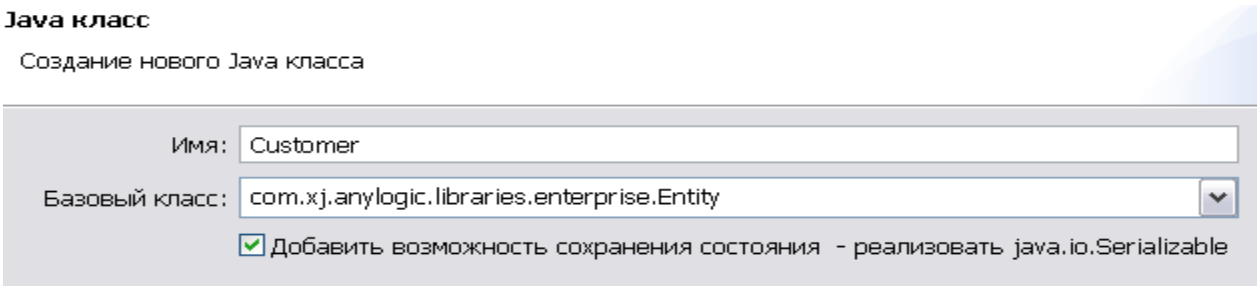


Рис.6.31. Создание класса, первая фаза

Шаг 15. Затем следует перейти ко второму шагу и задать поля класса – атрибуты, так как это показано на рисунке.

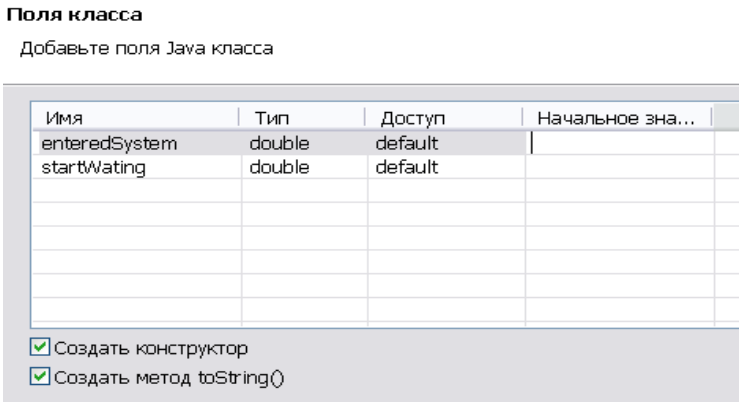


Рис.6.32. Создание класса, завершение

В результате будет создан класс Java, код которого можно открыть и отредактировать при необходимости.

Шаг 16. Затем разместите в модель из палитры «Статистика» элемент «Гистограмма» и элемента «Данные гистограммы». Гистограмма будет отображать распределение времени в системе на основе данных, собранных в элементе «Данные гистограммы». Назовите этот элемент назовите timeInSystemDistr (распределение времени пребывания в системе).

Откройте гистограмму, нажмите кнопку «Добавить данные». Задайте заголовок «Время в системе», в поле «Данные» задайте timeInSystemDistr, подберите цвет гистограммы. При настройке гистограммы должен быть активен флажок «Отображать плотность вероятности».

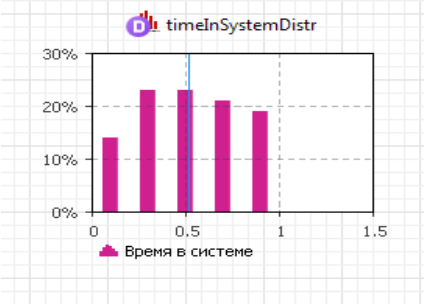


Рис.6.33. Гистограммы

Шаг 17. Откройте элемент Source и измените его настройки. На вкладке «Основные» установите свойства:

- класс заявки: Customer
- новая заявка: new Customer()
- действие при выходе: entity.enteredSystem=time().
- с помощью оператора new получают новый экземпляр заявки. Для обращения к атрибутам экземпляра служит указатель entity. При покидании заявки источника фиксируется время выхода.

Откройте элемент Conveyor и установите параметры.

Класс заявки: Customer

Действие при входе: timeInSystemDistr.add(time()-entity.enteredSystem)

Данные настройки позволяют зафиксировать время пребывания заявки в системе от входа до ее обработки. Элемент «Данные гистограммы» содержит метод add, с помощью которого добавляют данные.

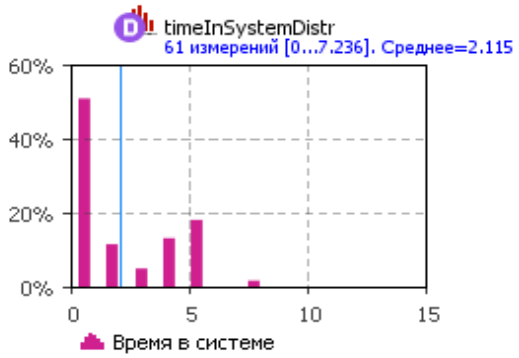


Рис.6.34. Гистограмма параметров системы

Вертикальная линия отображает среднее значение распределения. Для отображения среднего значения времени откройте последовательно гистограммы и на вкладке «Основные» установите флажок «Отображать среднее», подберите цвет линии.

Шаг 18. Также модернизируем внешний вид нашего перекрестка.

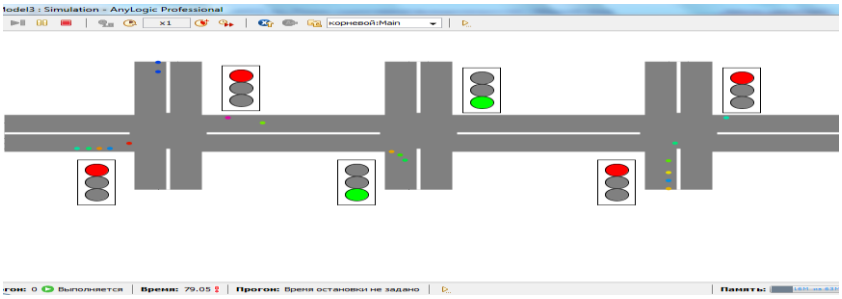


Рис.6.35. Дорога

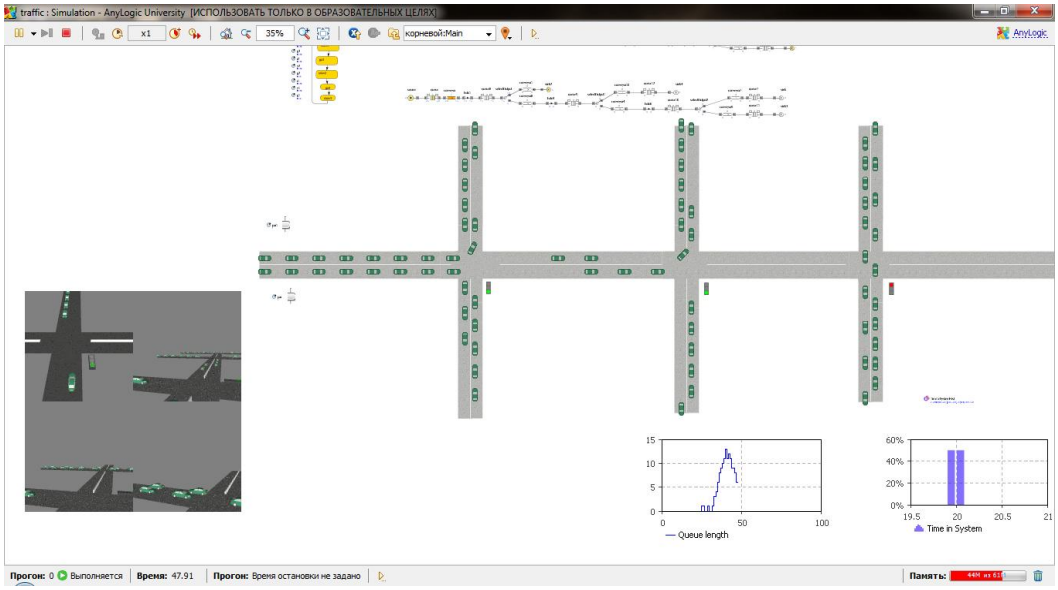


Рис.6.36. Интерфейс модели

С помощью среды AnyLogic можно смоделировать различные перекрестки. Можно построить разное количество линий движений, разное количество дорог, которые пересекаются. Можно также построить целую часть дороги. А с помощью стейтчартов можно задавать различное время для работы определенного цвета светофора. Тем самым можно добиться оптимальной работы светофора, что позволит устранить проблему пробок и облегчит жизнь всем автомобилистам.

6.3. Модель дорожно-транспортной развязки с железнодорожным переездом

Рассмотрим библиотеку дорожного движения среды Anylogic. Библиотека дорожного движения позволяет моделировать и визуализировать движение потоков машин. Библиотека поддерживает детализированное, но в то же время высокоэффективное моделирование движения машин на физическом уровне. С ее помощью можно промоделировать как движение машин на автомагистрали, так и уличный трафик машин, транспортировку на производстве, парковки и любые другие системы с машинами, дорогами и дорожными полосами. С помощью этой библиотеки можно моделировать и крупномасштабные системы дорожного трафика, поскольку какие-то части системы можно будет задать на более высоком уровне абстракции с помощью системной динамики или дискретно-событийного подхода моделирования, что потребует куда меньших затрат на вычисления.

Для начала надо добавить на рабочее поле такие объекты, как CarSource, RoadNetwork, CarDispose. Это основные элементы для задания транспортной сети. Также нам понадобятся такие элементы, как: Selectoutput, Carmoveto, и Delay.

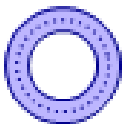


Рис.6.37. Объект RoadNetwork

Объект RoadNetwork (показан на рис. 6.37) задает сеть дорог, на основе нарисованного пользователем графика, проверяет правильность сети и отображает сеть дорог на анимации во время выполнения модели. Этот объект должен присутствовать в любой модели дорожного трафика. В модели может быть несколько независимых (не соединенных) сетей дорог – в этом

случае должно быть несколько объектов RoadNetwork- по одному на каждую сеть.

CarSource (показан на рис.6.38) создает машины, помещает их на одну из дорог и вставляет заявку типа Car в диаграмму процесса, задающую автотрафик.

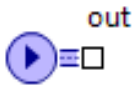


Рис. 6.38. Объект CarSource

С объекта CarSource обычно начинается диаграмма процесса дорожного трафика. CarDispose (показан на рис.6.37) удаляет машины из модели. Есть два способа удаления машины: во-первых, машина может выехать за пределы дорожной сети по незамкнутому пути, в этом случае объект CarDispose должен следовать за последним объектом TrainMoveTo, управлявшем машиной; кроме того, машина может "исчезнуть" из любого места дорожной сети (при условии, что она не движется). Удалять любые машины нужно с помощью объекта CarDispose (а не объектов Sink или Exit).

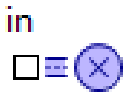


Рис. 6.39 Объект CarDispose

CarMoveTo (показан на рис.6.40) объект, который управляет движением машины. Для каждой машины-заявки объект CarMoveTo высчитывает маршрут от текущего местоположения машины до заданной точки назначения и направляет машину по этому маршруту. При достижении машиной точки назначения, она может как остановиться, так и продолжить движение без заданного маршрута.

Рис.6.40. Объект CarMoveTo

SelectOutput (рис.6.41) объект направляет входящие заявки в один из двух выходных портов в зависимости от выполнения заданного (детермини-

стического или заданного с помощью вероятностей) условия. Условие может зависеть как от заявки, так и от каких-то внешних факторов. Поступившая заявка покидает объект в тот же момент времени.

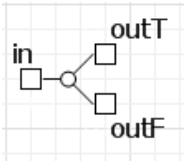


Рис. 6.41. Объект SelectOutput

Delay (рис.6.42) задерживает заявки на заданный период времени. Время задержки вычисляется динамически, может быть случайным, зависеть от текущей заявки или от каких-то других условий. Это время может, в частности, вычисляться как длина фигуры, заданной в качестве фигуры анимации этого объекта, поделенной на "скорость" заявки.

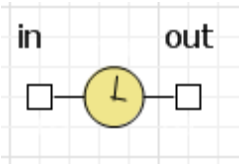


Рис. 6.42. Объект Delay

С помощью этих элементов построим первую сеть дорог, на которой будет осуществляться движение потока машин, а также заезд на АЗС. Построенная схема показана на рис. 6.43.

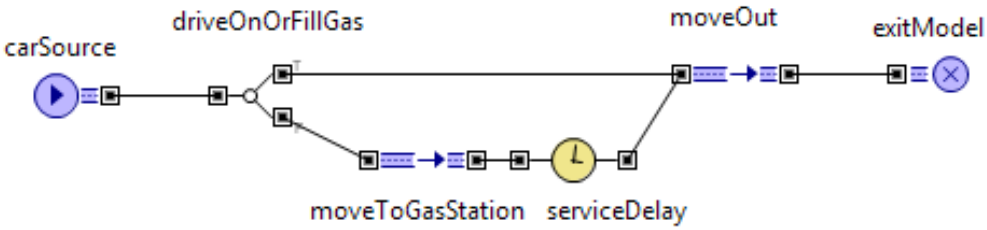


Рис. 6.43

По такому же принципу строим и вторую ветвь дороги. Машины, движущиеся по данной дороге, будут пересекать железнодорожный переезд, после чего разъезжаться в две стороны. Получаем результат (рис. 6.44).

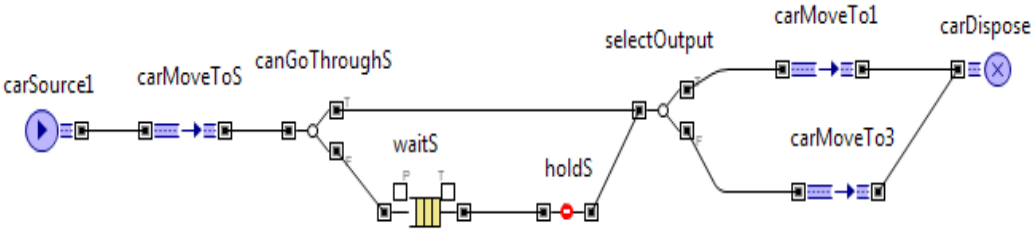


Рис. 6.44

Так как нам необходимо движение Железнодорожного состава через переезд необходимо построить схему и для него. Строится она также как и предыдущие схемы.

Чтобы заставить автомобили останавливаться на переезде во время движения состава, через него следует построить блок-схему, описывающую действие шлагбаума. При появлении состава посылается сообщение о закрытии шлагбаума, после чего перекрывается дорога. После его проезда отправляется сообщение от открытие переезда для автомобилей. Готовая блок-схема показана на рис. 6.45.

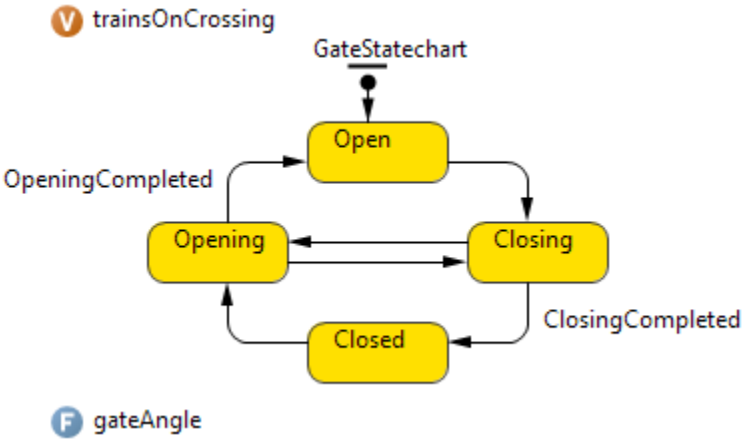


Рис. 6.45

Вся дорожная сеть будет выглядеть следующим образом, как показано на рис. 6.46.

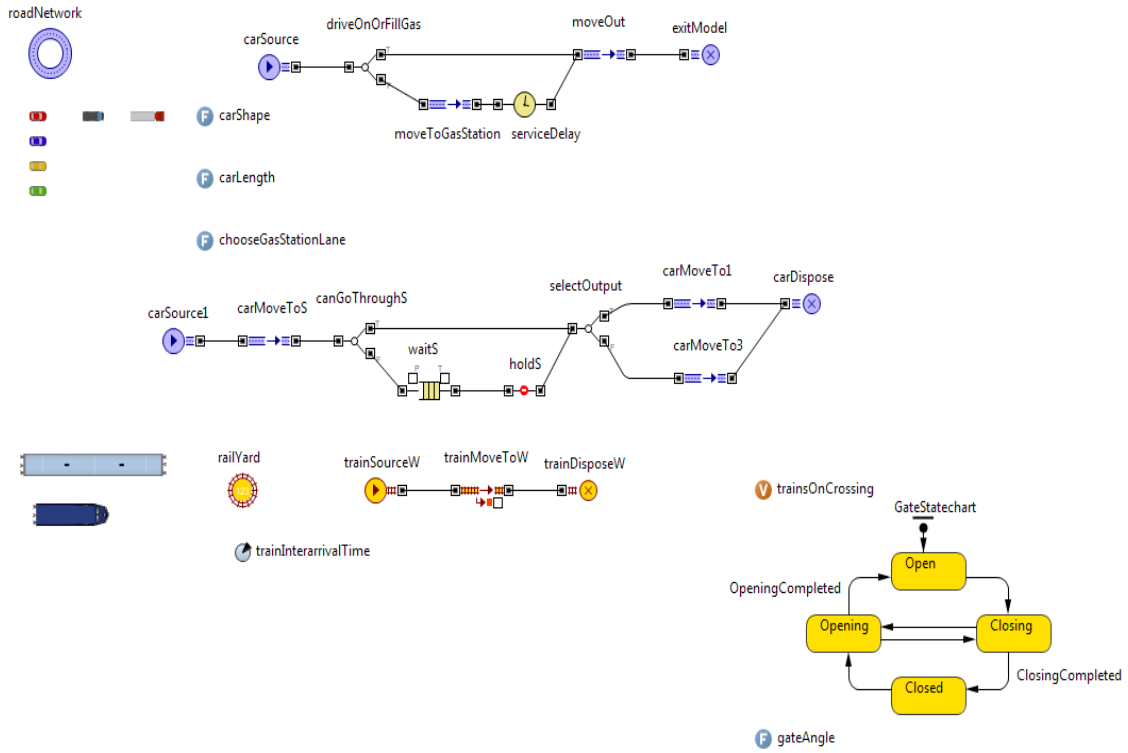


Рис. 6.46

Теперь остается только настроить все объекты сети, чтобы задать все параметры, касающиеся движения автомобилей. Ниже на рис.6.47 представлена дорожная сеть в 2D, нарисованная для закрепления за ней схем управления движением. Именно данная дорога будет отображаться в 3D окне.

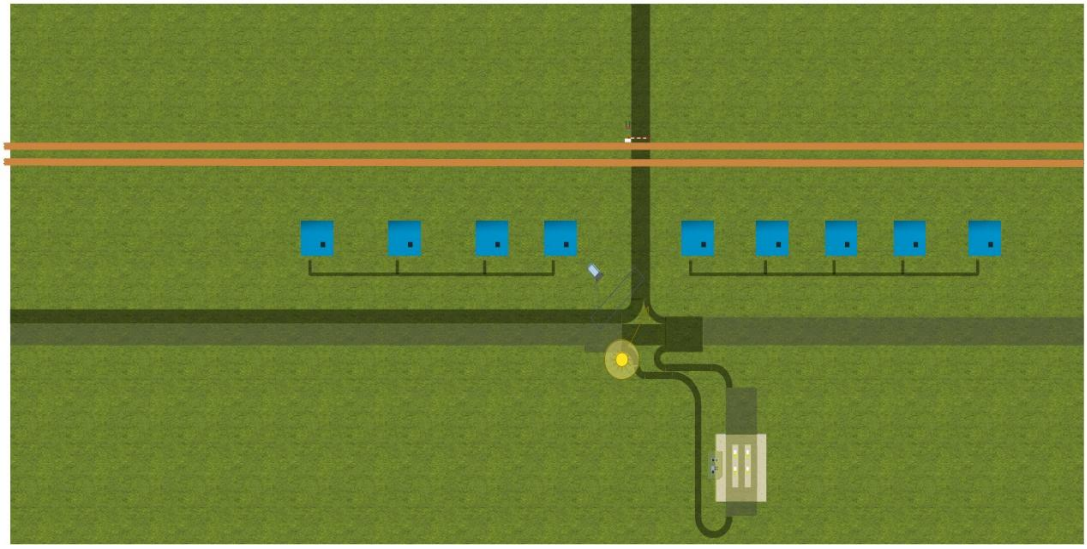


Рис. 6.47. Вид дороги в 2D

Также для отображения в 3D автомобильного потока нужно использовать объект, называемый «3D окно». Добавляем его на рабочее поле Main и выставляем нужное нам расширение окна.

Конечный вариант работы в 3D представлен ниже на рис.6.48

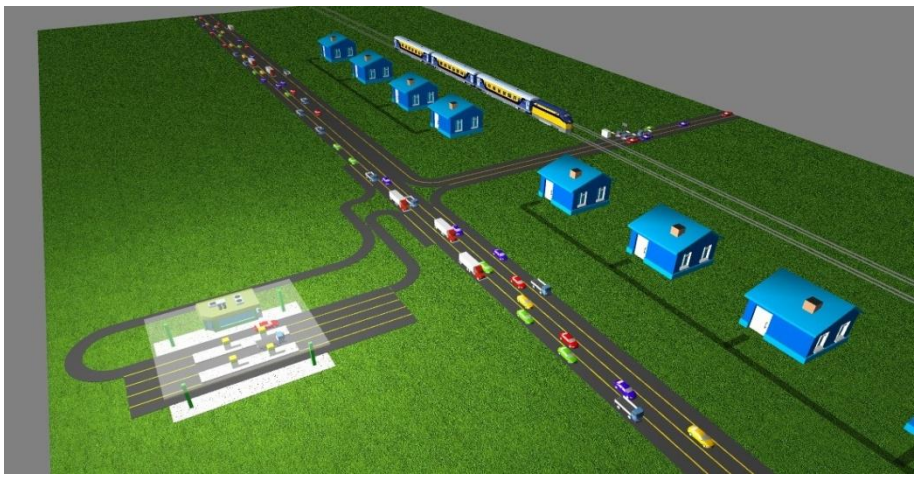


Рис. 6.48. Конечный вариант дорожной сети в 3D

6.4. Модель трубовидной транспортной развязки

Существуют большое количество различных видов транспортных развязок. Рассмотрим трубовидную развязку, такой вид реализован на развязке Аэропорт - Уфа.

Шаг 1. Создание новой модели

1. Нажмем **Создать** . Появится диалоговое окно **Новая модель**.

2. Зададим имя новой модели. В поле **Имя модели** введем «Развязка»

3. Выберим каталог, в котором будут сохранены файлы модели.

4. Нажмем **Далее**. Откроется вторая страница **Мастера создания модели**.

5. Здесь Вам будет предложено выбрать шаблон модели, на базе которого Вы будете разрабатывать Вашу модель. Поскольку мы хотим создать новую модель "с нуля", подробно объяснив Вам все шаги создания модели системной динамики в AnyLogic, оставьте флажок **Использовать шаблон модели** сброшенным и завершите процесс создания модели, щелкнув мышью по кнопке **Готово**.

Пользовательский интерфейс AnyLogic.

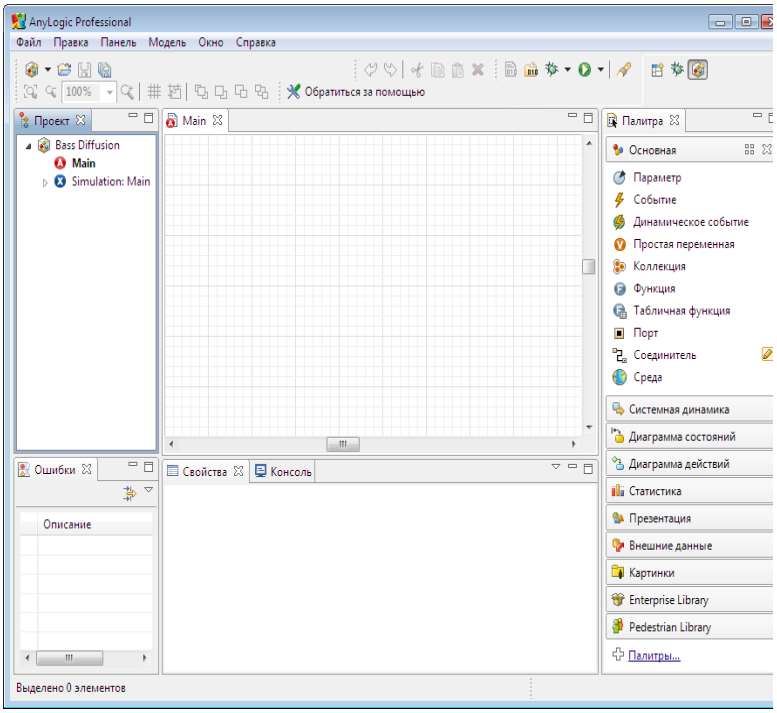


Рис.6.49

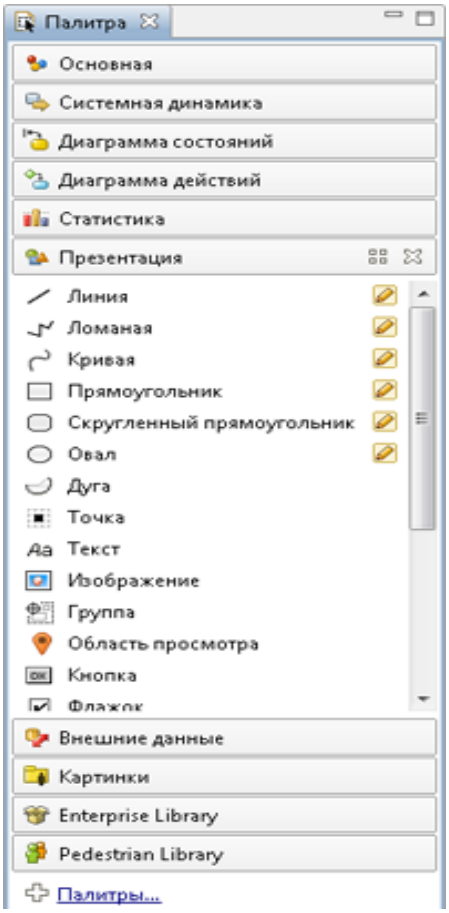


Рис. 6.50. Палитра Презентация

В левой части рабочей области находится панель Проекты. Панель **Проекты** обеспечивает легкую навигацию по элементам моделей, открытым в текущий момент времени. Поскольку модель органи-

зована иерархически, то она отображается в виде дерева: сама модель образует верхний уровень дерева; эксперименты, классы активных объектов и Java классы образуют следующий уровень; элементы, входящие в состав активных объектов, вложены в соответствующую подветвь дерева класса активного объекта и т.д.

В правой части рабочей области отображается панель **Палитра**, а внизу панель Свойства. Панель **Палитра** содержит разделенные по категориям элементы, которые могут быть добавлены на графическую диаграмму класса активного объекта или эксперимента. Панель **Свойства** используется для просмотра и изменения свойств выбранного в данный момент элемента (или элементов) модели.

В центре рабочей области AnyLogic находится графический редактор диаграммы класса активного объекта Main. При работе с моделью не забывайте сохранять изменения с помощью кнопки панели инструментов **Сохранить** .

Палитра Презентация содержит элементы, используемые для рисования презентаций моделей: фигуры, с помощью которых Вы можете рисовать сложные презентации, а также элементы управления для интерактивных презентаций.

Шаг 6. Задание дорожной сети.

Топология сети дорог задается с помощью обычных фигур презентации AnyLogic - линий и дуг. Ширина фигуры (точнее, ее **Толщина линии**) определяет количество полос на соответствующем участке дороги. Например, если толщина линии 60 пикселей, масштаб дорожной сети равен 10 пикселям в метре, а ширина полосы равна 3 метрам, то на соответствующем участке дороги будет две полосы. Сеть создается путем проверки соединения полос концов участков дорог. Если концы двух полос находятся друг от друга не дальше, чем на заданном параметром **Допуск при соединении дорог, пикселей**, и образуют тупой угол, то они считаются соединенными, так что машина, выезжающая с одной полосы, может продолжать движение по другой. Полосы могут сливаться и разветвляться.

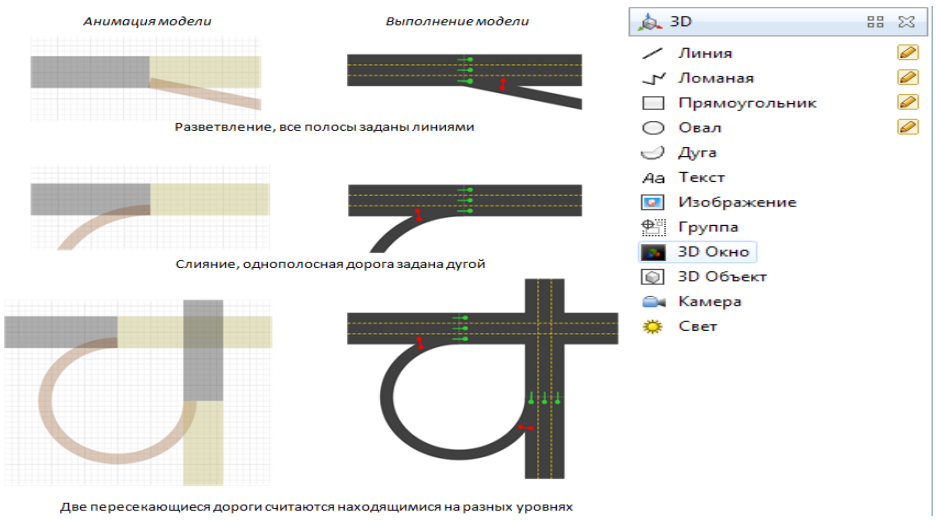


Рис. 6.51. Пример соединения дорог

Если фигуры двух дорог накладываются на анимации друг на друга, но соединения полос нет (см., например, пересечение вертикальной и горизонтальной дорог на самом нижнем рисунке), они считаются находящимися на разных уровнях, и трафики машин, проходящие по ним, не пересекаются.

Все фигуры, задающие дороги сети, должны быть помещены в группу, и группа должна быть указана в соответствующем параметре объекта RoadNetwork, который создаст на базе этой группы фигур дорожную сеть (набор объектов класса Road и Lane).

Если фигуры отображаются на 3D сцене, то дороги и машины будут показываться в 3D анимации как трехмерные объекты. Вы можете помещать дороги на разные уровни, задавать как прямые, так и наклонные участки дорог.

Дороги не должны накладываться друг на друга.

В блоке "roadNetwork" есть опция "Draw debug info". Рекомендуется ее включать во время отладки модели. Места правильного соединения дорог отмечаются зелеными значками. Необходимо убрать галочку "Показывать на 3D сцене" у группы фигур, задающих сеть, чтобы видеть эту служебную информацию. Также в большинстве случаев требуется указать несколько дорог, которые будут задавать направление движения (см. свойство "Direction hint road shapes" в блоке "roadNetwork").

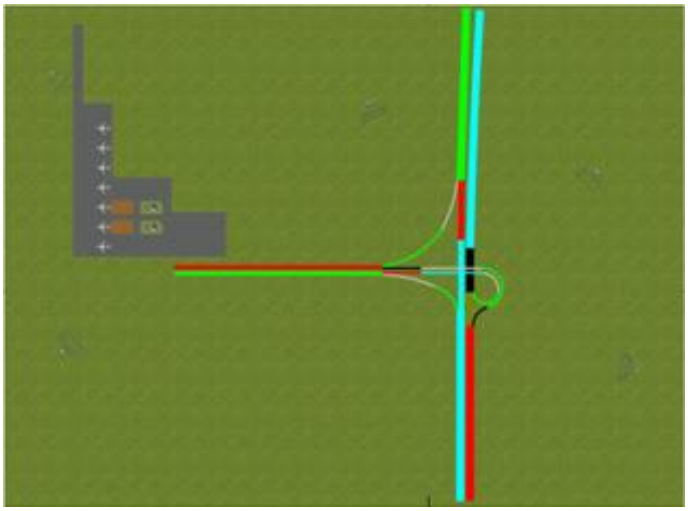


Рис. 6.52. Готовая дорожная сеть

Шаг 3. Моделирование дорожного движения

С помощью диаграммы процесса в моделях дорожного движения задается поведение автомобилей (так же, как в Основной библиотеке с их помощью задается поведение заявок). Диаграмма процесса в AnyLogic создается путем добавления объектов библиотеки из палитры на диаграмму класса активного объекта, соединения их портов и изменения значений свойств объектов в соответствии с требованиями Вашей модели.

Попадающие в моделируемую систему автомобили будут последовательно проходить по блокам созданной Вами диаграммы процесса.

Создадим диаграмму процесса

1. Добавим на диаграмму класса Main объекты библиотеки дорожного движения (имена их типов показаны на приведенном ниже рисунке).
2. Чтобы добавить на диаграмму объект библиотеки дорожного движения , нужно открыть в панели **Палитра** палитру этой библиотеки, щелкнув мышью по панели с ее заголовком, а затем перетащить нужный Вам объект из палитры на диаграмму класса.
3. Разместите объекты так, как показано на рисунке.

В свойствах **CarSource** в графе Дорога (Фигура) задаем начало движения. В свойствах **carMoveTo** в графе Доехать до дороги задаем конец маршрута.

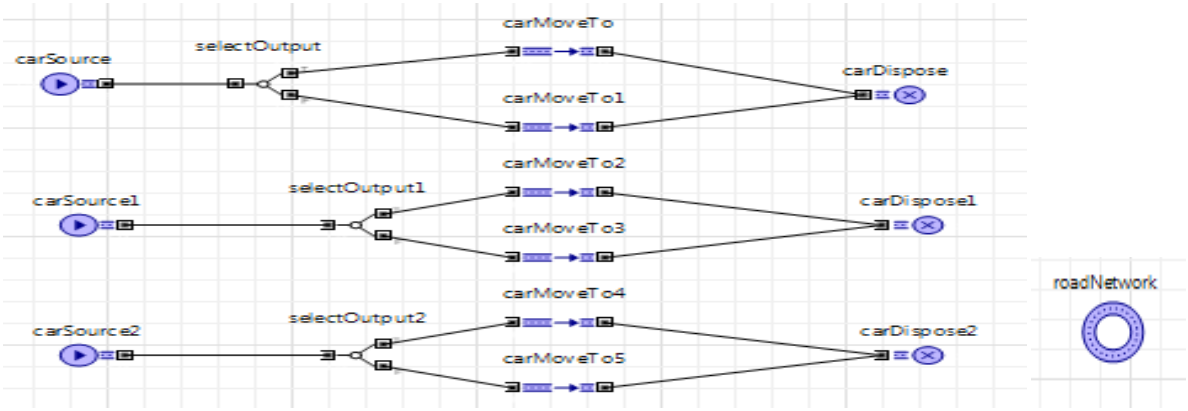


Рис. 6.53. Диаграмма процесса

1. Road Network

Объект **RoadNetwork** задает сеть дорог, на основе нарисованного пользователем графика, проверяет правильность сети и отображает сеть дорог на анимации во время выполнения модели. Этот объект должен присутствовать в любой модели дорожного трафика. В модели может быть несколько независимых (не соединенных) сетей дорог – в этом случае должно быть несколько объектов **RoadNetwork** – по одному на каждую сеть.

Сеть дорог создается путем тщательного поиска соединенных участков дорог (заданных линиями и дугами). Две полосы считаются соединенными, если их конечные точки находятся на расстоянии, не превосходящем заданный в объекте **Допуск при соединении дорог, пикселей**. Направление первой дороги определяется направлением соответствующей фигуры. Чтобы помочь объекту **RoadNetwork** присвоить правильные направления, Вам нужно указать список фигур с правильным направлением в параметре **Фигуры направления движения (необязательно)**. Количество полос на дороге вычисляется как ширина дороги (в метрах), поделенная на ширину полосы. Для этого Вы должны задать параметры **Масштаб, пикселей в метре**, который преобразует пиксели в метры, и **Ширина полосы, м.**. Вы можете изменить цвет дорожного покрытия и разметки дороги.

Если группа фигур сети содержит 3D фигуры, то объект **RoadNetwork** создаст как 3D, так и 2D анимацию сети дорог, иначе – только 2D. Обратите внимание, что при выборе у группы фигур опции **Отображать на 3D сцене**, Z-высота всех ее фигур устанавливается равной 10, что вряд ли может считаться желаемой Z-высотой для дорог. Проще всего изменить Z-высоту, щелкнув правой кнопкой мыши по группе, выбрав из меню пункт **Выделить**

содержимое группы и изменить значение в поле **Z-Высота** на странице свойств **Дополнительные**.

Обратите внимание, что объект **RoadNetwork** не будет автоматически отслеживать пересечение нарисованных Вами дорог (т.е. мест, в которых две дороги пересекают друг друга без соединения их полос), так что, если пересечение дорог находится на одном уровне, Вам придется самим следить за тем, чтобы на этом участке не было столкновений машин. Отдельная группа параметров задает типичное поведение водителей с учетом скорости и расстояний между машинами. Если быть точнее, есть зависимости минимального и максимального расстояний до находящейся впереди машины, выражаемые как функции от скорости, также две соответствующие обратные зависимости. Эти функции могут зависеть от типа машины.

Параметр **Дистанция для предупреждения о перестроении, м.** задает расстояние до предстоящего перестроения машины, движущейся без заданного маршрута, когда машина будет предупреждена о нем.

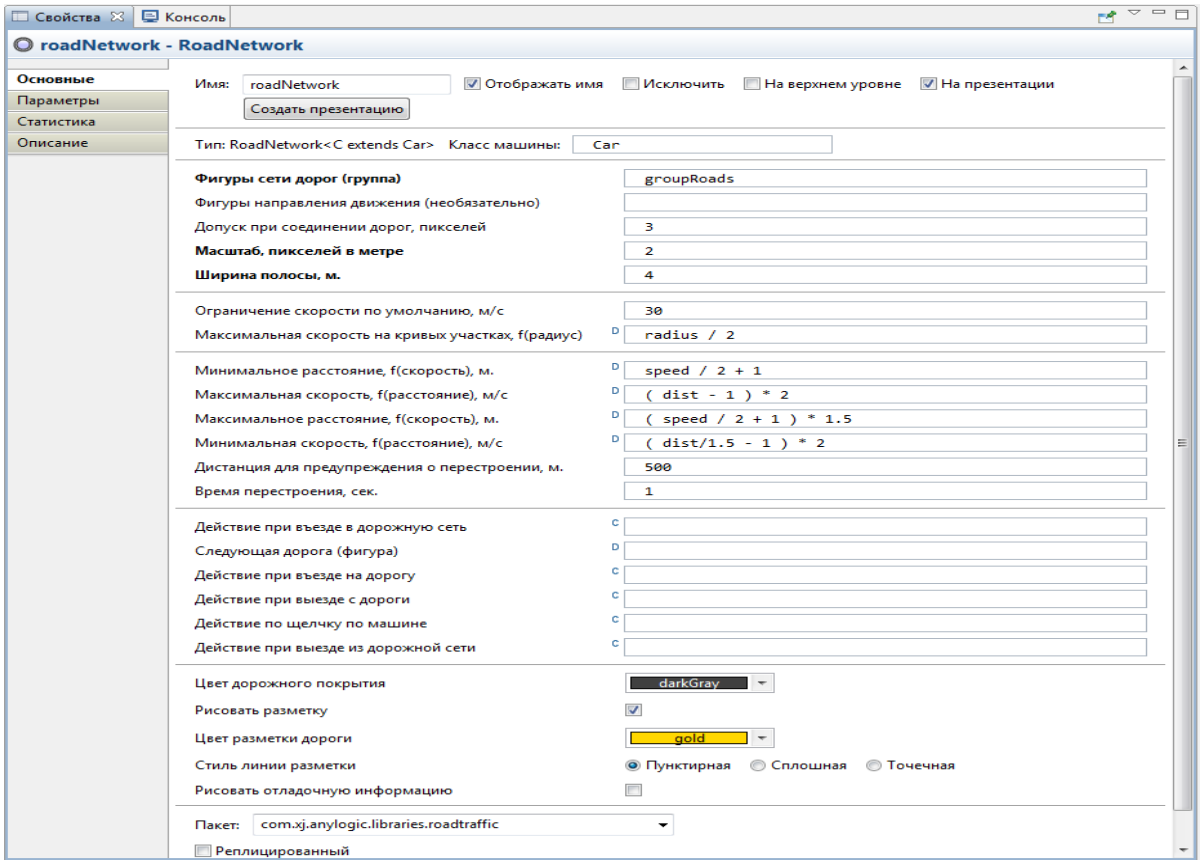


Рис. 6.54.Свойства объекта RoadNetwork

Есть также несколько параметров, в которых Вы можете задать какие-то специфические действия, которые будут выполняться в определенные ключевые моменты - при помещении машины в дорожную сеть, при въезде на новую дорогу и т.д.

2.

CarSource

Создает машины, помещает их на одну из дорог и вставляет заявку типа Car в диаграмму процесса, задающую автотрафик. С объекта **CarSource** обычно начинается диаграмма процесса дорожного трафика.

"Времена прибытий" машин, т.е. времена возникновения машин в модели, задаются аналогично временам появления заявок в объекте Source Основной библиотеки согласно заданной интенсивности, либо с помощью расписания прибытий или расписания интенсивностей, либо же путем задания времен между прибытиями. Кроме того, есть "ручной режим" создания, при котором объект **CarSource** будет создавать машины только в моменты вызова его функции inject(). Как и в объекте **Source**, Вы можете ограничить количество прибытий.

По умолчанию создаются машины класса Car, но Вы можете создавать машины и других классов, унаследованных от этого, базового. Вы можете создавать машины нестандартной длины и присваивать им какие-то свои 2D или 3D фигуры анимации. Обратите внимание, что размер нестандартной фигуры анимации машины не будет автоматически изменяться в соответствии с заданной длиной машины, поэтому Вам нужно будет выбрать соответствующий масштаб фигуры анимации так, чтобы ее визуальная длина соответствовала логической.

Новую машину помещают на полосу заданной дороги с заданным смещением от ее начала. Машина будет помещена на дорогу только в том случае, если на этой дороге перед ней будет требуемое расстояние, свободное от машин. Иначе же машина будет храниться во внутреннем буфере объекта **CarSource**, пока такое расстояние не появится. Если количество машин в буфере достигнет максимальной вместимости, возникнет ошибка. По умолчанию машиной управляет стандартный водитель (мы не советуем задавать своих собственных водителей до выхода окончательной версии библиотеки). У водителя есть один параметр – коэффициент превышения скорости, который означает отношение комфортной скорости водителя к ограничению на скорость.

Свойства | Консоль

carSource - CarSource

Основные | Параметры | Статистика | Описание

Имя: carSource | Отображать имя | Исключить | На верхнем уровне | На презентации | Создать презентацию

Тип: CarSource<C extends Car> Класс машины: Car

Объект RoadNetwork roadNetwork

Машины прибывают согласно Интенсивности

Интенсивность появления, машин в минуту 100

Ограниченное количество прибытий

Вместимость буфера 1000

Дорога (фигура) line19

Полоса: Выбирается случайно | Задана

Смещение заднего торца, м. 0

Расстояние перед машиной Безопасное расстояние для ограничения скорости | Другое значение

Новая машина new Car()

Длина, м. carLength()

Ширина, м. 2

Цвет spectrumColor(uniform_discr(0, 19), 20)

Фигура анимации машины carShape(car)

Ускорение, м/с2 10

Торможение (<0), м/с2 -50

Машиной управляет Стандартный водитель | Нестандартный водитель (должен реализовывать интерфей

Коэффициент превышения скорости (1 = заданное ограничение) uniform(0.7, 1.2)

Действие при выходе

Пакет: com.xj.anylogic.libraries.roadtraffic

Реплицированный

Начальное количество объектов:

Рис. 6.55. Свойства объекта CarSource.

Вы также можете задавать в объекте **CarSource** ускорение и торможение машины.

Свойства | Консоль

carMoveTo - CarMoveTo

Основные | Параметры | Статистика | Описание

Имя: carMoveTo | Отображать имя | Исключить | На верхнем уровне | На презентации | Создать презентацию

Тип: CarMoveTo<C extends Car> Класс машины: Car

Доехать до дороги arc5

Прибыть на полосу: Любая полоса | Самая правая | Полоса с заданным индексом

Смещение на дороге, м. roadlength-1

У цели Остановиться (водитель прекращает управление) | Продолжать движение

Действие при входе

Действие при въезде на дорогу

Действие при выезде с дороги

Действие при выходе

Пакет: com.xj.anylogic.libraries.roadtraffic

Реплицированный

Начальное количество объектов:

Рис. 6.56. Свойства объекта CarMoveTo

3.

CarDispose

Удаляет машины из модели. Есть два способа удаления машины: во--первых, машина может выехать за пределы дорожной сети по незамкнутому пути, в этом случае объект **CarDispose** должен следовать за последним объектом TrainMoveTo, управлявшим машиной; кроме того, машина может "исчезнуть" из любого места дорожной сети (при условии, что она не движется). Удалять любые машины нужно с помощью объекта **CarDispose** (а не объектов Sink или Exit).

4. Функции

AnyLogic позволяет пользователям создавать свои собственные функции. С помощью функций Вы можете единожды задать определенную последовательность действий (обычно – вычислений, возвращающих результат), которую нужно будет выполнять из разных мест (или в разные моменты жизни) модели. Функции пишут 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.

AnyLogic поддерживает диаграммы действий – структурированные блок-схемы, позволяющие задавать алгоритмы графически в стиле структурированного программирования. Диаграммы действий облегчают задание алгоритмов, делая необязательным знание синтаксиса Java операторов. Использование диаграмм действий дает еще одно преимущество: с их помощью Вы можете визуализировать алгоритмы, делая их более понятными для других пользователей модели.

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.

Чтобы задать функцию carLength

1. Перетащите элемент **Функция** из палитры **Основная** на диаграмму класса активного объекта (или эксперимента).
2. Перейдите на страницу **Основные** панели **Свойства** и задайте свойства функции.
3. Введите имя функции **carLength** в поле **Имя**.
4. Нужно указать тип возвращаемого значения **double**.
5. В свойствах аргументы функции ввести имя:car тип: Car

Перейдите на страницу **Код** панели **Свойства** и введите тело функции в поле **Тело функции**:

```
double rnd = uniform();
if( rnd < 0.1 ) return 10;
if( rnd < 0.3 ) return 6;
return 5;
```

Чтобы задать функцию carShape

1. Перетащите элемент **Функция** из палитры **Основная** на диаграмму класса активного объекта (или эксперимента).
2. Перейдите на страницу **Основные** панели **Свойства** и задайте свойства функции.
3. Введите имя функции **carShape** в поле **Имя**.
4. Нужно указать тип возвращаемого значения Другой и ввести в поле **Shape**.

Перейдите на страницу **Код** панели **Свойства** и введите тело функции в поле **Тело функции**:

```
double l = car.getLength();
if( l == 10 ) return truck3D;
if( l == 6 ) return lorry3D;
int rnd = uniform_discr( 1, 4 );
switch( rnd ) {
case 1: return car3DBlue;
case 2: return car3DRed;
case 3: return car3DYellow;
case 4: return car3DGreen;
}
return null;
```

Чтобы задать функцию carSpeeding

1. Перетащите элемент **Функция** из палитры **Основная** на диаграмму класса активного объекта (или эксперимента).
2. Перейдите на страницу **Основные** панели **Свойства** и задайте свойства функции.
3. Введите имя функции **carSpeeding** в поле **Имя**.

4. Нужно указать тип возвращаемого значения **double**.
5. В свойствах аргументы функции ввести Имя: car тип: Car

Перейдите на страницу **Код** панели **Свойства** и введите тело функции в поле **Тело функции**:

```
double l = car.getLength();
if( l == 10 ) return uniform( 0.4, 0.9 );
if( l == 6 ) return uniform( 0.6, 1.0 );
return uniform( 0.7, 1.2 );
```

Шаг 4. Создание 3D анимации

Начиная с версии 6.5 AnyLogic позволяет пользователям создавать трехмерные (3D) анимации моделей. Трехмерная анимация является самым наглядным и реалистичным способом визуализации моделируемого процесса.

Вы можете создавать трехмерные анимации из стандартных фигур презентации AnyLogic: прямоугольник, овал, линия, ломаная, текст, изображение, группа. Теперь у каждой из этих фигур есть возможность добавления третьей размерности – Z-высоты, и они появляются как в двухмерной анимации, так и в трехмерной: тем самым, Вы разом получаете два представления Вашей модели.

Примитивные фигуры обычно используются для того, чтобы нарисовать объекты невысокого уровня сложности, например, коробки, дороги, стены помещений. Более сложные трехмерные объекты (люди, автопогрузчики, грузовики, самолеты, и т.д.) обычно не рисуют в AnyLogic, а импортируют извне. Для этого используется элемент 3D фигура, позволяющий добавлять на анимацию изображения трехмерных фигур любой сложности из файлов форматов X3D и VRML.

- Если у Вас есть уже готовые модели с двухмерной анимацией, то Вам не нужно заново рисовать фигуры для создания аналогичной трехмерной анимации. Все, что Вам нужно – это просто указать в свойствах этих фигур, что они будут отображаться и в трехмерной анимации (установив флажок Отображать на 3D сцене), добавить на диаграмму окно трехмерной анимации (3D окно) и все!
- Еще одним преимуществом нашей реализации трехмерной анимации является то, что Вы добавляете окно трехмерной анимации на привычную Вам графическую диаграмму аквтиного объекта. Это дает возможность создания своего рода стендов управления моделью с элементами управления, графиками, двухмерной и трехмерной анимацией бок о бок, на панели общего для всех окна презентации. Таким образом, в

отличие от большинства инструментов моделирования, мы избавляем пользователей от необходимости переключения между несколькими окнами во время выполнения модели.

- Не стоит забывать о том, что трехмерная анимация – всего лишь еще один способ представления результатов моделирования. Он идеально подходит в тех случаях, если Вам нужно провести презентации Ваших имитационных проектов. Но создание трехмерной анимации зачастую требует существенного времени на разработку. И в том случае, если у Вас нет такой необходимости, Вы можете обойтись и другими, более простыми средствами визуализации результатов работы модели.

3D окно представляет собой элемент, задающий на диаграмме класса активного объекта область, в которой во время запуска модели будет отображаться трехмерная анимация этого объекта.

Если Вы хотите, чтобы у Вашего объекта была трехмерная анимация, то этот элемент будет необходимо добавить на диаграмму этого объекта.

На диаграмме одного объекта может присутствовать сразу несколько окон трехмерной анимации, каждое из которых может показывать какой-то конкретный участок общей трехмерной сцены (аналогично связанным с разными камерами видеонаблюдения дисплеям на пульте оператора).

Вы можете настроить окно трехмерной анимации на то, чтобы показывать по умолчанию какой-то определенный участок трехмерной сцены, выбрав у него в свойствах камеру, которая и будет "снимать" то, что будет "показывать" окно трехмерной анимации. Направьте камеру в графическом редакторе на те фигуры презентации, которые Вы хотите увидеть в окне при запуске модели. Эта возможность очень часто используется, поскольку не всегда окно 3D анимации будет по умолчанию позиционироваться так, что будет отображать именно нужные Вам трехмерные объекты, и настроить один раз камеру будет куда проще, чем постоянно навигироваться к требуемым объектам во время работы модели.

Чтобы добавить на презентацию окно трехмерной анимации (3D окно)

1. Перетащите элемент **3D Окно** из палитры **3D** на диаграмму активного объекта.
2. Вы увидите в графическом редакторе закрашенную черным область.

3. Перейдите на страницу **Основные** панели **Свойства**.
4. Выберите подходящий **Тип навигации.**

Шаг 5. Добавление Камеры

Камера используется для определения того, какой именно участок презентации будет отображаться в окне трехмерной анимации. Она как бы "снимает" то, что "показывает" окно трехмерной анимации (3D окно) (естественная аналогия – камера видеонаблюдения и дисплей на пульте оператора).

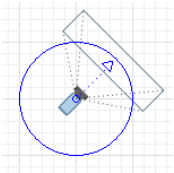
Если вы хотите, чтобы при запуске модели в окне трехмерной анимации отображались именно нужные вам объекты (и под оптимальным углом), то лучше один раз задать нужное вам расположение смотрящего на сцену (и направление его взгляда) с помощью камеры, чем постоянно перемещаться к требуемым объектам по ходу выполнения модели.

Вы можете задать сразу несколько камер, направленных как на разные участки трехмерной сцены, так и на одни и те же объекты, но из разных точек и/или под разными углами, и переключаться между ними во время работы модели.

Камера может перемещаться по ходу выполнения модели. Вы можете сделать так, чтобы в окне 3D анимации всегда отображалась картинка, попадающая в объектив движущейся камеры. Эта возможность может быть полезна в том случае, если вы хотите постоянно иметь в поле зрения объект, который может двигаться по ходу выполнения модели.

Чтобы добавить камеру

1. Перетащите элемент **Камера** из закладки **3D** панели **Палитра** в то место диаграммы вашего класса активного объекта, где вы хотите ее поместить. Вы увидите значок камеры:



2. Поверните камеру так, чтобы она была направлена на нужные вам фигуры презентации.
3. Теперь вы можете выбрать эту камеру в качестве камеры окна трехмерной анимации. Выделите 3D окно и выберите имя этой камеры в его свойстве **Камера**. Если вы предполагаете, что ваша камера будет двигаться во время работы модели, и вы хотите, чтобы окно следовало за этой камерой, то выберите опцию **Следовать за камерой**.

Шаг 6. Запуск модели

Чтобы запустить модель, щелкаем по кнопке со стрелкой справа от кнопки панели инструментов Запуск и выбираем эксперимент, который мы хотим запустить, из выпадающего списка.

В окне презентации нажимаем на кнопку Запустить модель.

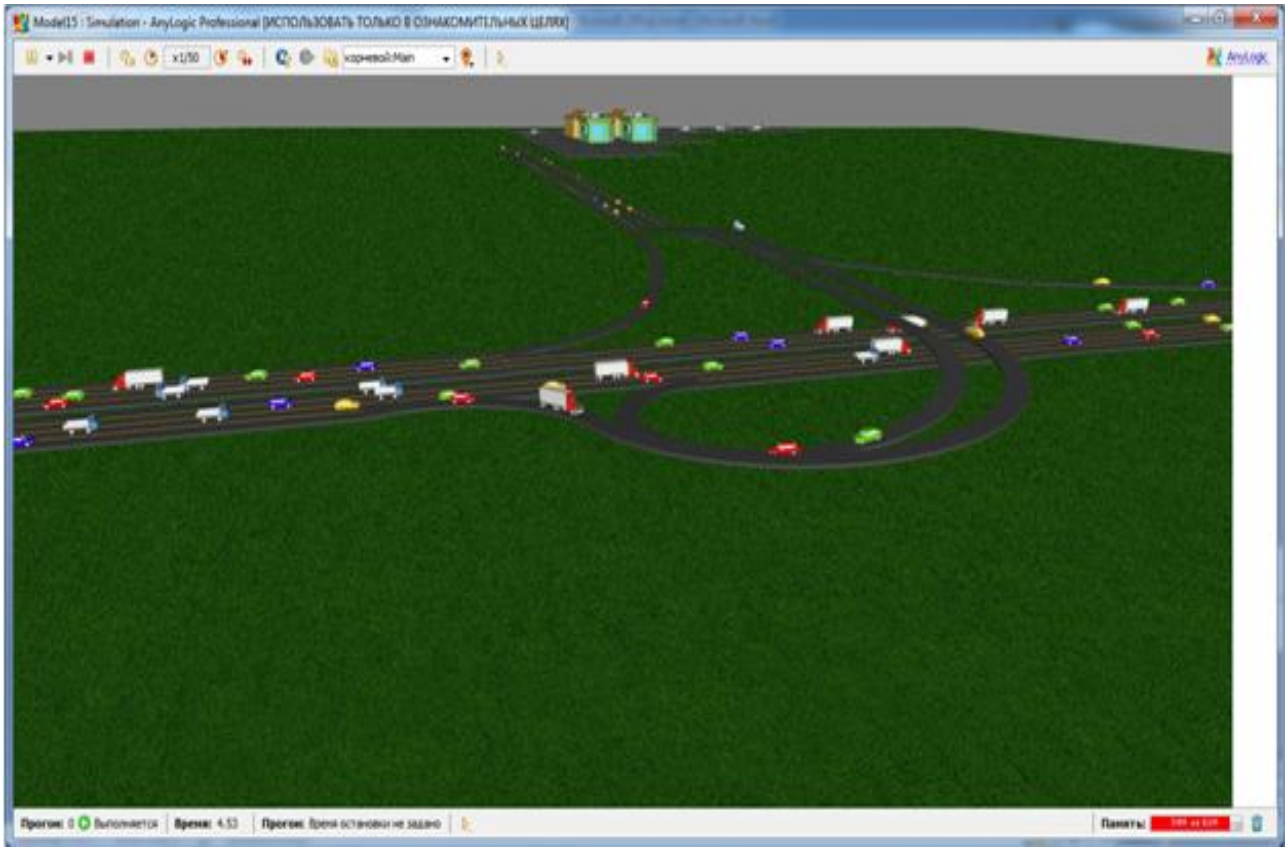


Рис.6.57. Вид транспортной развязки в анимационной модели

Рис. 6.58. Фрагмент анимационная модель

ГЛАВА 7

ДИСКРЕТНО-СОБЫТИЙНОЕ МОДЕЛИРОВАНИЕ ПОЛИГРАФИЧЕСКИХ ПРОЦЕССОВ

Производственные процессы в полиграфическом производстве подразделяются в основном на несколько этапов: допечатные процессы, печатный процесс и послепечатная обработка. На всех этапах полиграфического производства возникают свои проблемы, требующие решения. Появляется необходимость спроектировать технологию «от и до», рационализирующую процесс изготовления издания, и делающую издание экономически выгодным[31, 34, 76, 109].

Технологический процесс представляет собой определенную последовательность операций. Порядок прохождение заказов по цехам, участкам, операциям, машинам не может быть изменен (могут только отсутствовать некоторые операции). Однако при выполнении пакета заказов есть возможность оптимизировать производственный процесс по времени, путем изменения очередности выполнения заказов.

Возможность применения теории массового обслуживания (ТМО) для исследования полиграфического производства определяется следующими факторами:

1. Количество заявок в системе (которая рассматривается как СМО) должно быть достаточно велико (массово).

2. Все заявки, поступающие на вход СМО, должны быть однотипными.

3. Для расчетов по формулам необходимо знать законы, определяющие поступление заявок и интенсивность их обработки. Более того, потоки заявок должны быть пуассоновскими.

4. Структура СМО, т.е. набор обслуживающих аппаратов (ОА) и последовательность обработки заявки, должна быть жестко зафиксирована.

5. Необходимо исключить из системы субъектов или описывать их как ОА с постоянной интенсивностью обработки.

К перечисленным выше ограничениям можно добавить еще одно, оказывающее сильное влияние на размерность и сложность математической модели.

7. Количество используемых приоритетов должно быть минимальным. Приоритеты заявок должны быть постоянными, т.е. они не могут меняться в процессе обработки внутри СМО.

Массовость заявок. Анализ полиграфического производства показывает, что теория массового обслуживания может быть использована для анализа различных производственных ситуаций. Первой из них является задача анализа функционирования издательств и типографий как систем массового обслуживания при поступлении в них случайных заявок на издание полиграфической продукции. Заявки могут поступать в случайные моменты времени и могут обладать различной степенью сложности и различным объемом, что приводит к случайному времени исполнения заявок. Применение теории массового обслуживания позволяет оценить пропускную способность издательства или типографии, количество заказов, находящихся в производстве, время пребывания заявок и т.д.

Если рассматривать заказ как заявку в СМО, то количество заказов должно быть массовым. В средних допечатных полиграфических фирмах в среднем обрабатывается около 20 заказов в день. Ежедневный приток новых заказов в среднем составляет не более 25%. Одна печатная машина редко за один день печатает более трех заказов. Аналогичная ситуация возникает и в фирмах, занимающихся послепечатной обработкой.

Из вышесказанного следует, что о массовости заказов можно говорить только при рассмотрении продолжительных интервалов времени (недели, месяцы). Для оценки производительности полиграфического предприятия чаще всего достаточно оценить небольшой промежуток времени. В этом случае использование аналитического подхода не оправдано. Для крупных полиграфических корпораций и для фирм, ориентированных на долгосрочные заказы, использование ТМО может оказаться более эффективным.

Преодоление ограничения массовости возможно за счет рассмотрения в качестве заявок не заказов, а его составляющих частей. Например, можно представлять заказ как множество страниц.

Однотипность заявок. Проблема однотипности заявок является одним из основных препятствий для использования ТМО в полиграфической предметной области. Многие полиграфисты считают, что каждый заказ является уникальным, поэтому его нельзя типизировать. Если считать, что заказ имеет более 100 характеристик, каждая их которых может принимать несколько значений, то нельзя не согласиться с этим утверждением.

Преодоление этой проблемы заключается в разбиении заказа на составные части (страницы, технологические операции и др.) и пренебрежении некоторыми характеристиками. Например, задание плотности бумаги, типа и цвета кожи обложки не оказывает серьезного влияния на процесс печати.

Параметры заказа, определяющие количество цветов (красок) и формат издания, наоборот, прямо определяют и возможность, и время печати.

В общем случае неучтенные характеристики заказа могут рассматриваться как необходимые ресурсы. Наличие или отсутствие ресурсов может оказывать серьезное влияние на последовательность и время решения задачи. Например, заказы, для которых имеется достаточное количество ресурсов, могут иметь более высокий приоритет. Сложностью такого подхода является то, что приоритеты могут меняться динамически в процессе работы.

Если рассматривать заказ как совокупность технологических операций, на каждую из которых существует норма, то разнородность заявок можно представить в виде нескольких простейших заявок. При этом количество простейших заявок зависит от технологической операции, объема материала и коэффициента сложности.

Еще одной сложностью, вызванной разнородностью, является наличие ОА, которые могут обслуживать только определенные типы заявок. Решение этой проблемы заключается в разделении соответствующих потоков заявок и рассмотрении системы в виде множества независимых последовательно или параллельно связанных СМО. Недостатками этого подхода являются: невозможность точного определения процесса окончания обработки заказа, так как заявки привязаны только к технологическим операциям; необходимость генерации и уничтожения заявок на стыках отдельных систем; использование ОА только для обработки одного типа заявок.

Законы поступления заявок. В ТМО, в основном, используются пуассоновские потоки заявок (заказов, технологических операций, компонент заказов). Для расширения области применения ТМО используют различные методы сведения произвольных потоков к пуассоновскому (дифференциальный и интегральный методы, метод Кендалла, асимптотические методы). Часто, не зная законы распределения, аналитики рассматривают самые “худшие” случаи, когда число поступающих заявок экспоненциально растет. Однако такой подход позволяет дать только грубую оценку системы.

Для того чтобы определить потоки заявок и интенсивность их обработки, необходимо иметь статистические данные. В настоящее время количество исследований в полиграфии на данную тему крайне мало. Единственным источником для анализа полиграфической фирмы может служить собственная статистика, набранная за значительный период времени. Большинство малых и средних фирм оформляют и рассчитывают технологические карты только в бумажном виде. Кроме того, не документируется информация об

интенсивности обработки, количестве и причине возникающих ошибок, сложности работы и т.д. Все это служит серьезным препятствиям для использования ТМО.

Фиксированная структура. Для описания системы с помощью ТМО необходимо, чтобы она имела фиксированную структуру и последовательность обработки заявок. Необходимо отметить, что фиксированная последовательность движения заявок не означает полную детерминированность всех процессов обработки. Например, если заявка находится в очереди, которую обслуживают несколько устройств (ОА), то процесс выбора ОА в общем случае является стохастическим. Однако количество ОА для одной очереди, последовательность и направление движение заявок не может меняться.

На практике последовательность выполнения заказа может сильно изменяться. Например, если невозможно осуществить сканирование материалов (зарезервировать ОА1) или распечатку исходных данных для вычитки (использовать ОА2), то можно осуществить ввод данных (ОА3), верстку (ОА4) и т.д. Часто один ПК (ОА) используется для выполнения различных технологических операций. При переполнении одной из очередей к ОА, ее необходимо сделать приоритетной, а заявки из других очередей перенаправить к другим ОА. Более того, в классическом представлении к ОА должна иметься только одна очередь. Тогда заявки, размещенные в очередь, должны определенным образом сортироваться либо с помощью динамического изменения приоритетов, либо с помощью других механизмов.

Решением этой проблемы является создание множества математических моделей, фиксация структуры в системе, рассмотрение приближенных случаев. Если количество возможных состояний системы велико, то размерность и сложность математической модели сильно возрастает, и использование ТМО не оправдано.

Представление субъектов. Субъекты (персонал) можно представлять в виде ОА с определенной интенсивностью обработки. Их производительность следует определять из статистики, накопленной за продолжительный промежуток времени. Сложностью этого подхода является наличие в производстве новых субъектов (с неизвестной производительностью); постоянное, но неравномерное влияние опыта работника на его производительность; уникальность каждого субъекта.

Альтернативным вариантом задания интенсивности обработки заявки субъектом является использование нормативов. В этом случае математическая модель будет соответствовать “идеальному” производству. Такой под-

ход можно применять для анализа крупных предприятий, где каждый работник выполняет (должен выполнять) один вид работы за фиксированное время. Результатами анализа может являться проверка адекватности нормативов и возможность их изменения.

Часто моделируемую систему исследуют на работоспособность в критических ситуациях: при наличии большого числа заказов, при возникновении большой срочности, при отсутствии или незначительном количестве работы и т.д. В этих случаях поведение субъектов крайне трудно спрогнозировать.

Представление субъектов в полиграфии, особенно в допечатном производстве, имеет специфику. Один сотрудник может выполнять различные виды работы с различным качеством. Отсюда следует, что субъект может обладать вектором специальностей, и для передачи ему работы необходимо оценивать его профессиональные качества и загрузку. Таким образом, переход заявки от одного ОА к другому не является случайным и зависит от многих параметров. Представление зависимостей такого рода в СМО крайне затруднительно.

Приоритеты. Использование приоритетов в полиграфической предметной области необходимо для задания срочности заказов (заявок), для формирования общей очереди к устройствам, выполняющим различные виды работ и т.д. Особенностью использования приоритетов является то, что они могут изменяться динамически в процессе работы. Это связано с тем, что заказы должны выполняться в течение определенного времени, и система должна работать без отказов заявок после их принятия к обработке.

На рис.7.1представлена IDEF- модель полиграфического производства. В модели показан процесс производства полиграфических изделий в три стадии.

На первой стадии изделия проходят допечатную подготовку, потом направляются в печатный отдел, где в зависимости от типа изделия печатаются четырьмя разными видами печати. Третья стадия производства – это послепечатная обработка изделий, после которого готовая продукция доставляется в места реализации. Этапы полиграфического производства представлены на рис.7.2.

Рис 7.1. Упрощенная IDEF модель полиграфического производства

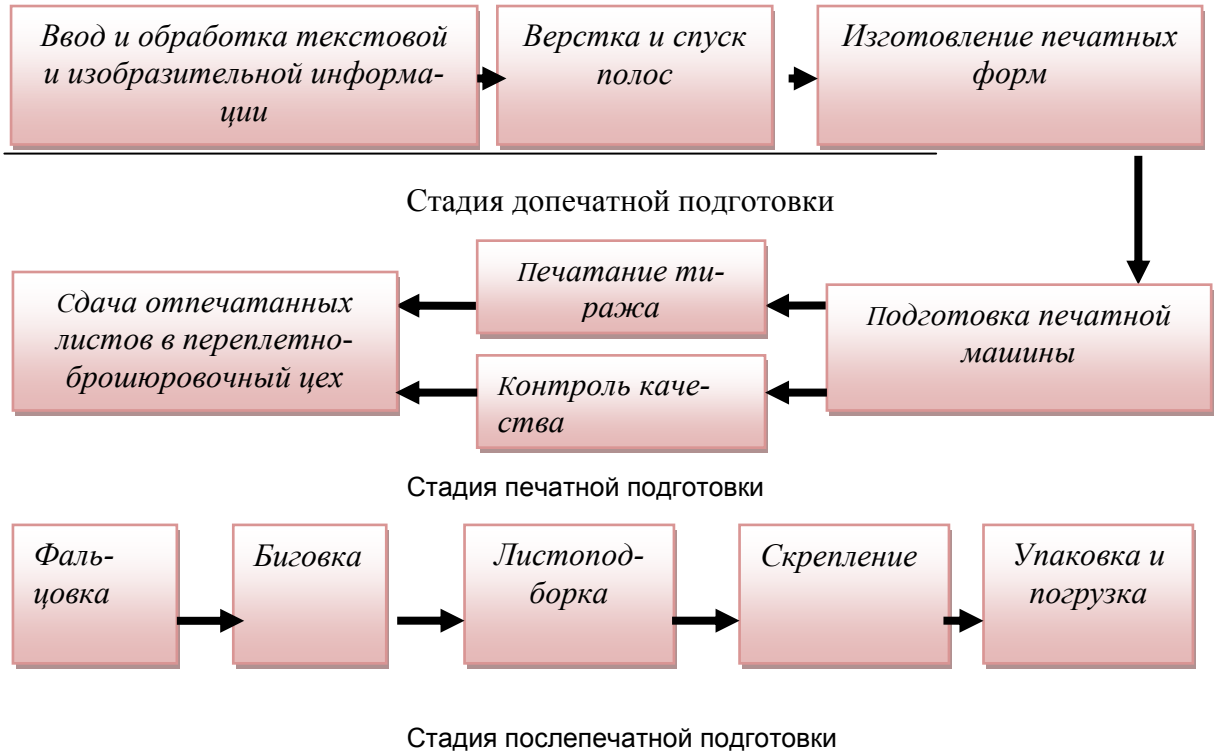


Рис. 7.2. Этапы полиграфического производства

Здесь мы представили тринадцать этапов. На первом этапе ввод и обработка текстовой и изобразительной информации. На втором этапе проходит

верстка и спуск полос. На третьем этапе выполняется изготовление печатных форм. На четвертом этапе подготовка печатной машины, на пятом проходят два параллельных процесса, такие как: печатание тиража и контроль качества. В шестой этап входит сдача отпечатанных листов в переплетно-брошюровочный цех. Седьмой этап – это фальцовка. Восьмой этап – это биговка, девятый этап – это листоподборка, десятый и одиннадцатый этапы – это скрепление и последнее что происходит в производстве, это упаковка и погрузка.

7.1. Имитационная модель подготовки макета издания (допечатная подготовка)

Проблемы редакторской подготовки изданий, моделирования допечатных и других полиграфических процессов изложены в ряде книг [5, 7, 13, 31, 35, 47, 56, 61, 76, 109, 118, 137, 141].

В данной работе мы рассмотрим технологии допечатных процессов с помощью среды имитационного моделирования AnyLogic. Создадим модель работы редакционно-издательского отдела в среде AnyLogic, в котором проводится допечатная подготовка макета издания Будем считать, что система поступления заявки на издание и прохождение их в отделе допечатной подготовки до стадии макета издания является системой массового обслуживания. Рассмотрим один из алгоритмов поступления заявки на издание рукописи автора.

1. Автор, прибывающий в отдел, вначале направляется в кабинет главного редактора.

2. Главный редактор передает рукопись одному из редакторов. Если все редакторы оказываются занятыми, то рукопись становится в очередь, пока один из редакторов не освободится.

3. После обработки рукописи редактор передает ее корректору.

4. После корректорской обработки рукопись передается автору для последней проверки и правки текста, после которого рукопись отдается техническому редактору для подготовки оригинал-макета, на этом же этапе проводится верстка.

5. Заключительным этапом является подпись в печать главным редактором и передача оригинал-макета в печатный цех.

На основе этого алгоритма создадим имитационную модель допечатной подготовки с помощью объектов Основной библиотеки для моделирования транспортных сетей (названия этих объектов начинаются с Network) в среде AnyLogic. Эти объекты используются в том случае, если для модели-

руемых процессов имеет значение физическое расположение объектов в пространстве. В таких моделях заявки движутся по заданным путям и используют ресурсы, находящиеся в сети.

Целью нашей модели будет анализ работы отдела и оптимальности имеющих в нем место процессов обслуживания – с учетом точного времени, как на проведение основных этапов, так и передвижение рукописи по отделу на основе теории массового обслуживания. Чтобы время передвижения рукописи по отделу соответствовали реальному, мы добавим в модель план отдела (физического расположения комнат), нарисуем пути движения людей между кабинетами, зададим им скорости передвижения и соответственно время, затрачиваемое на достижение того или иного кабинета, будет зависеть от реальной длины пути, который нужно будет пройти.

Реализация проекта допечатные процессы полиграфического производства в среде AnyLogic.

Шаг 1. Создание новой модели. Вначале мы создали новую модель.

1. Щелкнули мышью по кнопке панели инструментов Создать . Появилось окно Мастера создания модели.

2. Задали имя новой модели. В поле Имя модели ввели Pre – press.

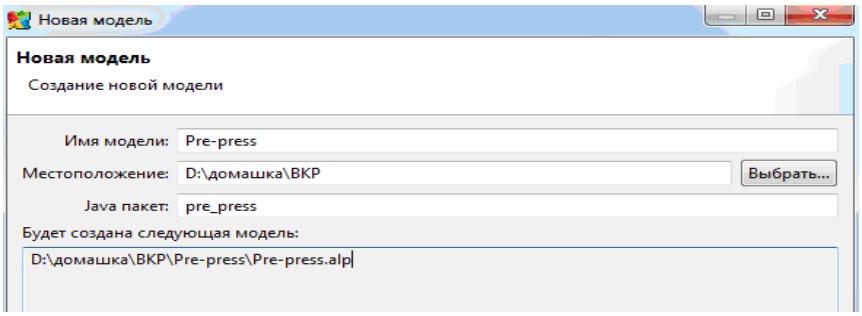


Рис.7.3 Создание новой модели

3. Выбрали каталог, в котором будут сохранены файлы модели.

4. Щелкнули мышью по кнопке Далее. Открылась вторая страница Мастера создания модели. Выбираем пункт «Начать создание модели «с нуля». Подтверждаем операцию, нажав кнопку Готово.

5. Перед нами открылась рабочая область.

Пользовательский интерфейс AnyLogic

В левой части рабочей области находится панель Проекты. Панель Проекты обеспечивает легкую навигацию по элементам моделей, открытых в текущий момент времени. Поскольку модель организована иерархически, то она отображается в виде дерева: сама модель образует верхний уровень дерева; эксперименты, классы активных объектов и Java классы образуют

следующий уровень; элементы, входящие в состав активных объектов, вложены в соответствующую подветвь дерева класса активного объекта и т.д.

В правой рабочей области отображается панель Палитра, а внизу – панель Свойства. Панель Палитра содержит разделенные по категориям элементы, которые могут быть добавлены на диаграмму класса активного объекта или эксперимента. Панель Свойства используется для просмотра и изменения свойств выбранного в данный момент элемента (или элементов) модели.

В центре рабочей области AnyLogic находится графический редактор диаграммы класса активного объекта Main.

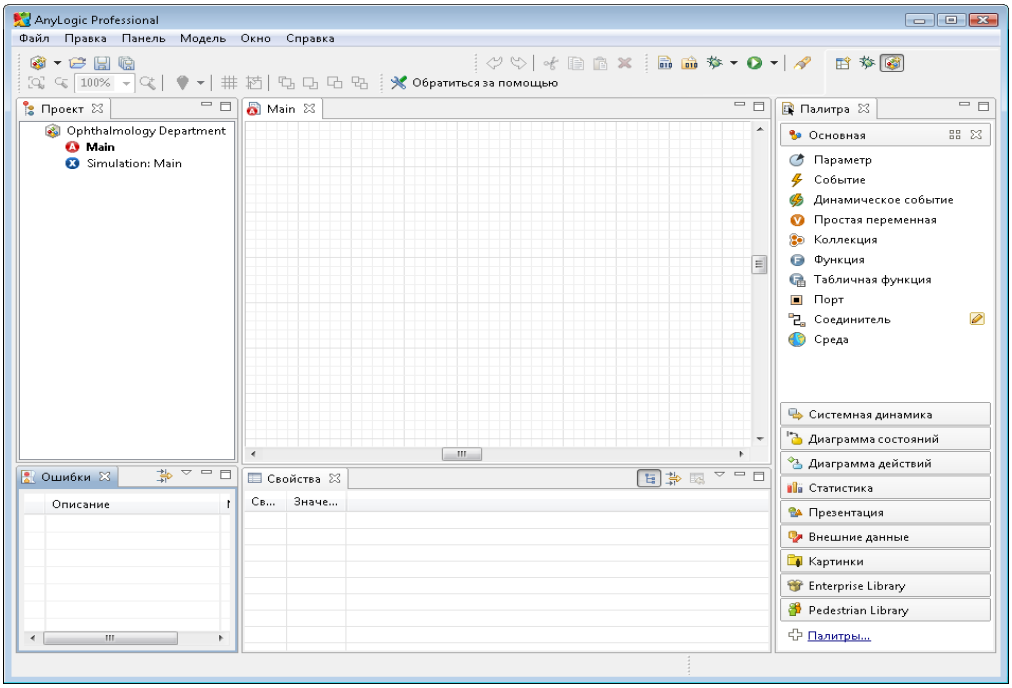


Рис.7.4. Рабочая область AnyLogic

При работе с моделью необходимо сохранять производимые изменения с помощью кнопки панели инструментов Сохранить .

Шаг 2. Создание анимации модели. Нарисовали анимацию нашей модели, поскольку именно она будет определять структуру нашей модели – ее транспортную сеть.

Вначале добавили изображение плана отдела, чтобы потом рисовать узлы транспортной сети нашей модели (соответствующие кабинетам отдела) и сегменты транспортной сети (то есть, пути движения людей по отделу) уже в дополнение к этому плану.

1. Добавляем на презентацию план отдела.

2. На закладке Презентация панели Палитра выбираем Изображение. Палитра Презентация содержит элементы, используемые для рисования

презентаций моделей: фигуры, с помощью которых можно рисовать сложные презентации, а также элементы управления, с помощью которых можно сделать презентации интерактивными.

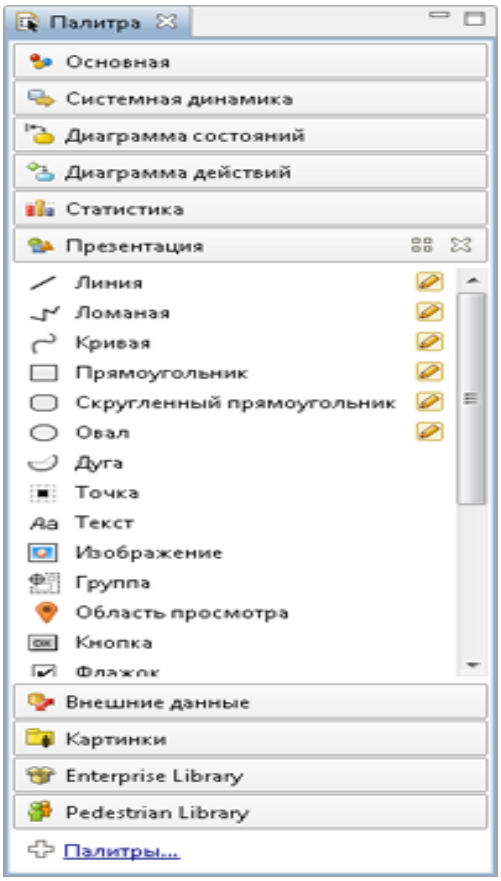


Рис.7.5. Панель палитра

3. Перетаскиваем элемент Изображение из палитры Презентация на диаграмму класса активного объекта. Поместили его так, как показано на рисунке:

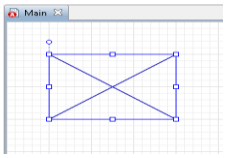


Рис.7.6. Объект Изображение

4. Задали свойства изображения в панели Свойства. Щелкнули мышью по кнопке Добавить и выбрали файл изображения плана отдела. Файл находится в каталоге Program Files \ AnyLogic 6 \ resources \ tutorials \ Ophthalmology Department \ layout.png. В области появилось добавленное изображение.

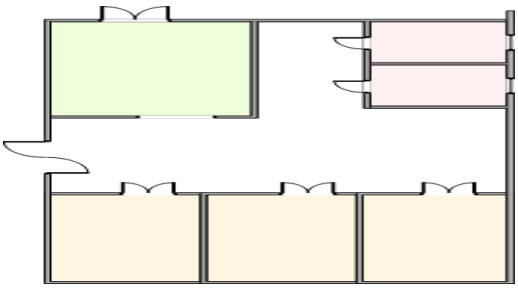


Рис.7.7. Добавленное изображение

5. Чтобы сохранить исходный размер изображения, установили флажок Исходный размер. Заблокировали изображение, установив флажок Блокировать, для того чтобы исключить возможность случайного редактирования изображения при рисовании этих фигур. Невозможно выбрать заблокированную фигуру в графическом редакторе до тех пор, пока не снята блокировка.

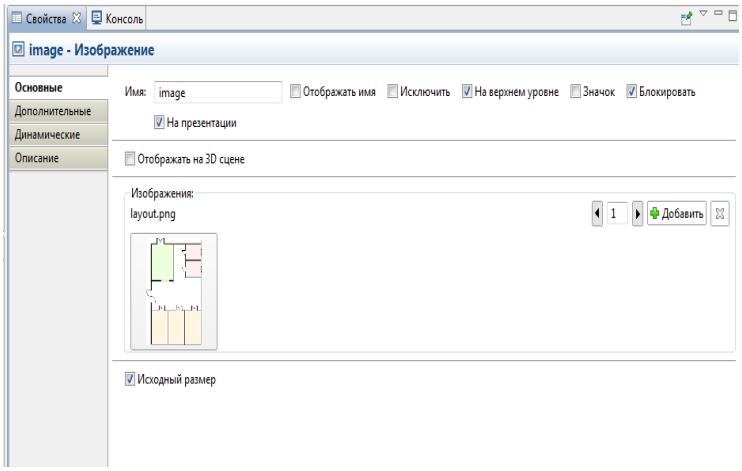


Рис.7.8. Свойства Изображения

6. Изображение выглядит следующим образом (рис.7.8):

7. Далее нарисовали анимацию модели. На основе анимации строится транспортная сеть модели: прямоугольники соответствуют узлам сети, а ломаные линии – связям между ними, играющим роль путей движения заявок и ресурсов в моделируемом пространстве.

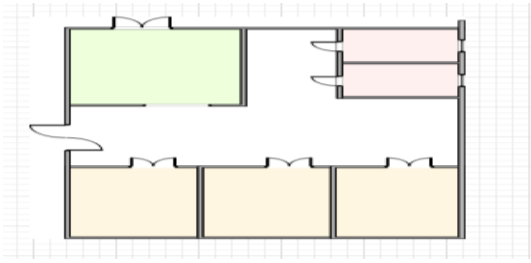


Рис.7.9. Заблокированное изображение

Поэтому, чтобы создать требуемую транспортную сеть, нарисовали на анимации помещения нашего отдела с помощью прямоугольников и соединили их ломаными линиями.

Рисуем узлы сети.

1. Нарисовали каждый кабинет редакционно-издательского отдела с помощью прямоугольника. Отдел включает в себя кабинет главного редактора, два кабинета редакторов, кабинет технического редактора, кабинет корректора и кабинет верстальщика.

2. Далее рисуем прямоугольники и подписываем их как на приведенном ниже рис.7.10.

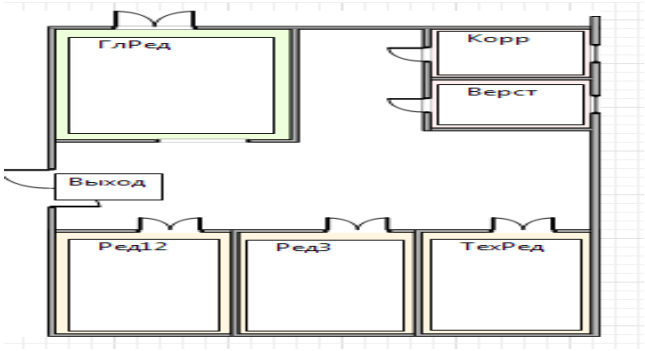


Рис.7.10. Добавленные прямоуголиники с подписями

3. Чтобы нарисовать прямоугольник, сделали двойной щелчок мышью по элементу Прямоугольник ▭ в палитре (при этом его значок поменялся на этот: ✎).

4. Чтобы сделать пути движения людей на анимации более реалистичными, добавили дополнительные узлы сети, нарисовав еще несколько прямоугольников и поместив их так, как показано на приведенном ниже рисунке:

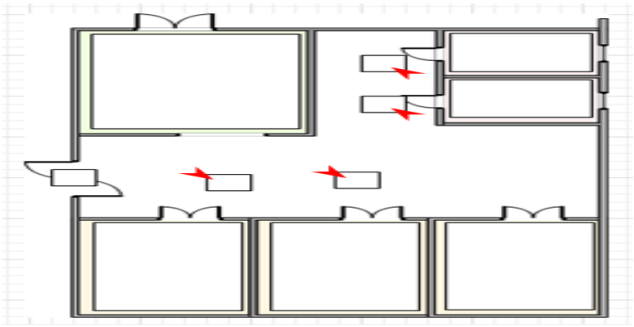


Рис.7.11. Дополнительные узлы сети

Теперь рисуем сегменты сети с помощью ломаных линий. Эти линии будут задавать пути движения рукописи.

Рисуем пути (сегменты сети).

1. С помощью инструмента рисования Ломаная , нарисовали ломаные линии, как показано на рисунке ниже. Соединили соседние узлы сети – тем самым задали требуемую транспортную сеть модели:

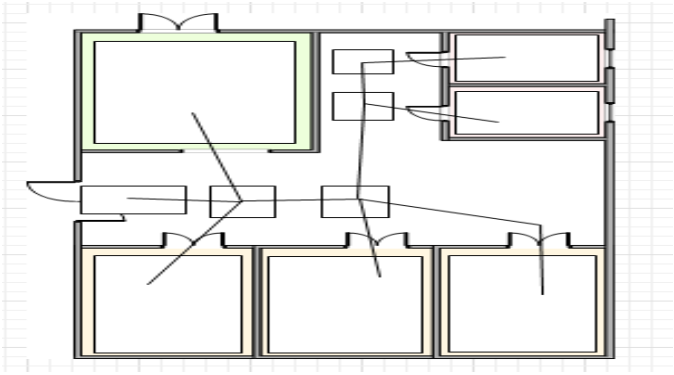


Рис.7.12. Сегменты сети

2. Чтобы нарисовать ломаную, сделали двойной щелчок мышью по элементу Ломаная в палитре (при этом его значок поменялся на этот:). Все начальные и конечные точки линий должны обязательно находиться внутри соединяемых прямоугольников (и в одном прямоугольнике не может находиться более одной точки одной и той же ломаной).

Теперь необходимо добавить наши фигуры в группу. На базе элементов этой группы, которую мы потом укажем в соответствующем параметре конфигурационного объекта сети, будет сконструирована логическая структура сети.

Добавление фигур в группу.

1. Добавили все нарисованные фигуры в группу фигур. Вначале выбрали эти фигуры. Лучше всего сделать это, нажав левую кнопку мыши сбоку от крайней из этих фигур, и не отпуская кнопку перетащить мышь так, чтобы рамка выделения покрыла всю область, содержащую фигуры, которые необходимо выделить (как на рис. ниже).

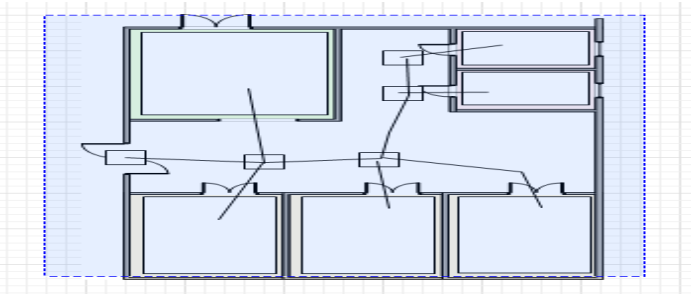


Рис.7.13. Выделение

2. Отпустили кнопку мыши. Выделенные таким образом фигуры подсвечены, синим цветом. Если какая-то из фигур оказалась не выделенной, можно добавить ее в группу выделенных фигур, нажав Ctrl, и не отпуская ее, щелкнуть мышью по той фигуре, которую необходимо добавить. Фигура будет добавлена в выделение.

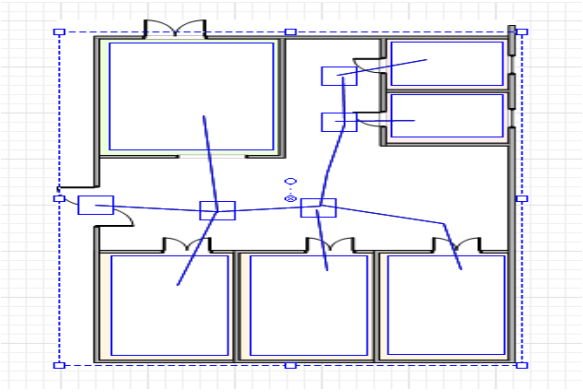


Рис.7.14. Выделенные объекты

3. Когда все фигуры выделены, сделали щелчок правой кнопкой мыши по выделенным фигурам и выбрали Создать группу из контекстного меню.

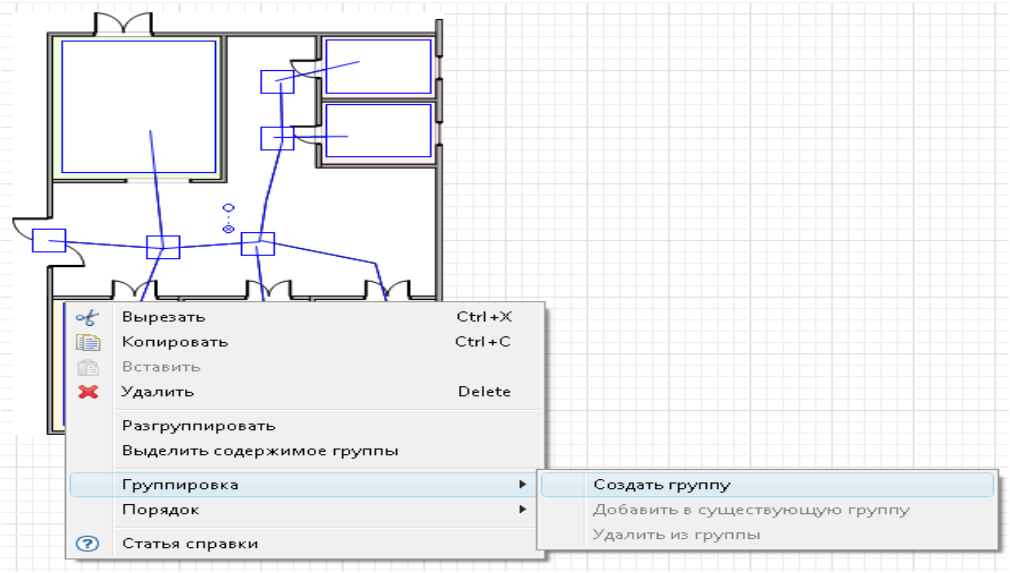


Рис.7.15. Группировка

Задание местоположения кабинетов редакторов.

1. Нарисовали ломаную линию, соединяющую прямоугольники, представляющие собой кабинеты редакторов. Она понадобится нам чуть позже. Поместили точки ломаной точно внутрь прямоугольников Ред1, Ред2, Ред3.

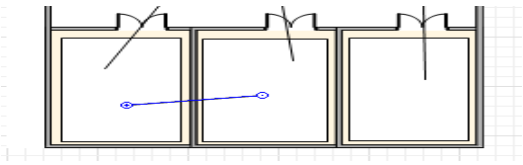


Рис.7.16. Ломаная

2. Назвали ее Линия и выбрали для нее другой цвет.

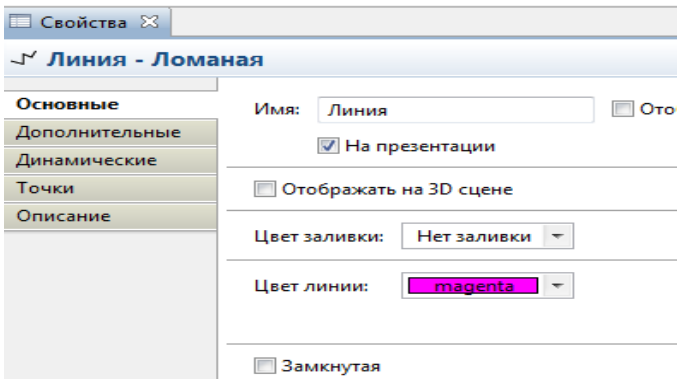


Рис.7.17. Свойства ломаной

3. Сделали ее невидимой во время выполнения модели. Перешли на страницу Динамические панели свойств ломаной и ввели false в поле Видимость.

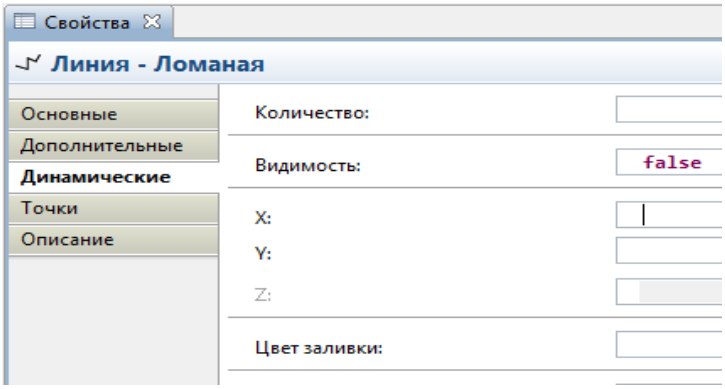


Рис.7.18. Свойства Ломаной

Шаг 3. Анимация ресурсов

Далее нарисуем анимацию для нашего автора, рукописи и работников отдела, чтобы мы могли отличать их на анимации нашей модели.

Создание изображений автора и работников.

1. Открыли палитру Картинки. Эта палитра содержит набор картинок, которые наиболее часто используются пользователями AnyLogic при создании моделей.

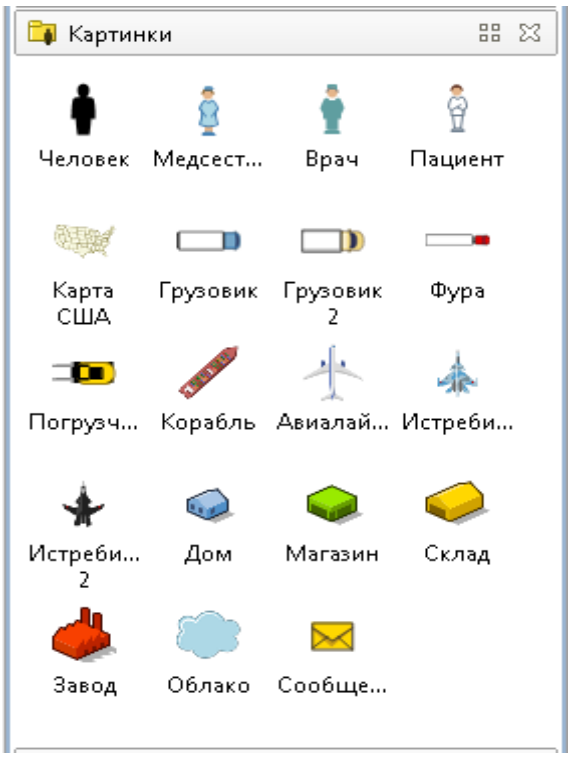


Рис.7.19. Картинки

2. Перетащили элемент Человек из палитры на диаграмму графического редактора:

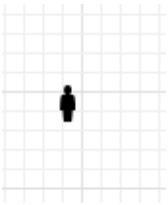


Рис.7.20. Добавление картинки Человек

3. По умолчанию эта картинка будет называться person. Переименуем ее в название Автор, в дальнейшем мы будем ссылаться на нее в блоке диаграммы нашего процесса именно по этому имени.

4. Перетащили еще один элемент Человек из палитры на диаграмму графического редактора:

5. Переименуем ее в ГлРедактор.

6. Добавим необходимое количество работников со следующими названиями картинок: Редактор1, Редактор2, Редактор3, Корректор, ТехРедактор, Верстальщик.

7. Добавляем картинку рукописи, как показано на рис.7.21.

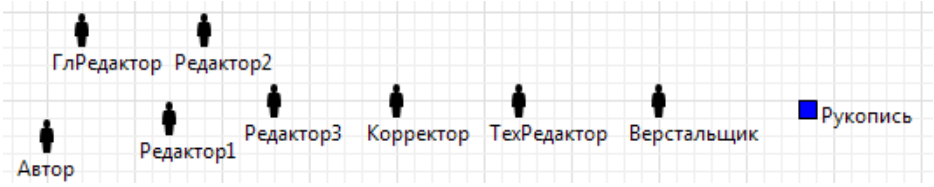


Рис.7.21. Добавленные картинки

Чтобы картинки ресурсов не попали в область презентации, которая будет видна после запуска модели, передвинули их за границы области презентации, которая будет отображаться в окне презентации запущенной модели.

Помещение картинок ресурсов за границу видимой области презентации

1. Передвинули холст диаграммы немного вниз, нажав правую кнопку мыши в графическом редакторе и перетащив мышь, не отпуская кнопки, вниз.

2. Выделили мышью картинки всех ресурсов и перетащите их чуть выше границы видимой области диаграммы.

3. Передвинули холст диаграммы обратно, чтобы картинки ресурсов оказались за областью видимости диаграммы.

Рис.7.22. Картинки за областью видимости диаграммы

Выделили фигуры автора и рукописи и сгруппировали их.

Закончили создание презентации нашей модели. Теперь мы можем изменить настройки сети и задать процесс с помощью диаграммы процесса, составляемой из блоков моделирования транспортных сетей Основной библиотеки.

Далее изменим свойства объекта, описывающего сеть, и добавим объекты, задающие имеющиеся в нашей модели ресурсы.

Шаг 4. Задание сети и ресурсов

Задание свойств сети.

1. Добавили на диаграмму объект Network. Этот объект задает транспортную сеть модели и ее свойства.

2. Чтобы добавить на диаграмму объект Основной библиотеки, открыли в панели Палитра палитру этой библиотеки, щелкнув мышью по панели с ее заголовком, а затем перетащили нужный объект из палитры на диаграмму класса.

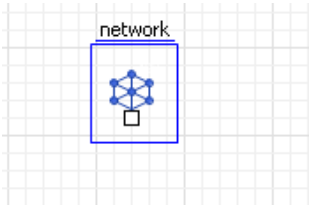


Рис.7.23. Добавление объекта Network

3. В параметре Группа фигур сети ввели имя группы фигур анимации: group. Этот параметр позволяет указать этому объекту Network, какие именно фигуры анимации задают логическую структуру задаваемой этим объектом сети.

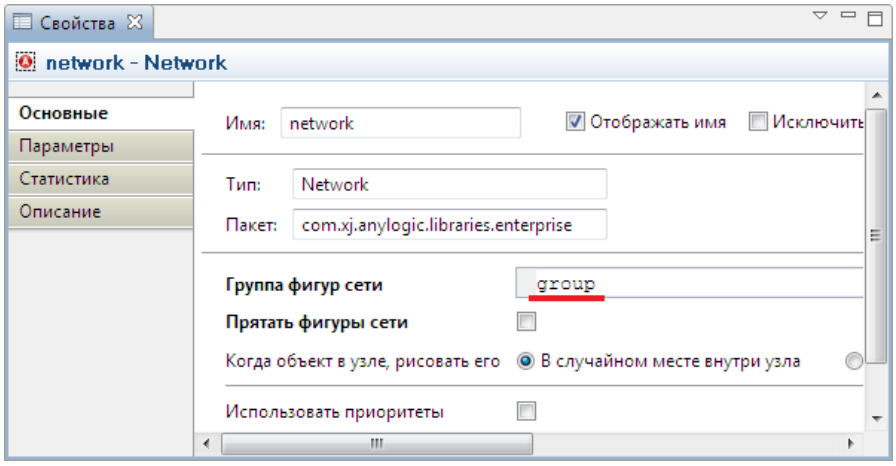


Рис.7.24. Свойства объекта Network

Далее добавили объекты, задающие сетевые ресурсы. Сетевые ресурсы могут быть трех видов: движущиеся, переносные и статические. В нашем случае люди будут заданы движущимися ресурсами, рукопись – портативными ресурсами, а кабинеты – статическими.

Задание ресурса типа «ГлавРедактор».

1. Добавили на диаграмму класса Main объект Основной библиотеки NetworkResourcePool. Объект NetworkResourcePool описывает ресурсы определенного типа. Этот объект будет задавать свойства ресурсов, представляющих в нашей модели авторов и работников. Задали следующие свойства объекта:

- Назвали объект ГлавРедактор.
- Указали, сколько работников будет присутствовать в модели. Ввели в поле Количество ресурсов: 1.

- Задали базовое местонахождение ресурсов. Движущиеся ресурсы возвращаются в заданное здесь место, когда они становятся свободными. В нашей модели все работники возвращаются в свои кабинеты, поэтому оставьте выбранным в поле Базовое местоположение задается как опцию Один узел и ввели в поле Базовый узел имя прямоугольника, который представляет на анимации кабинет главного редактора: ГлРед.
- Указали, какой фигуркой хотим отображать главного редактора на анимации. Ввели имя добавленной нами ранее картинки ГлРедактор в полях Фигура анимации свободного ресурса и Фигура анимации занятого ресурса.

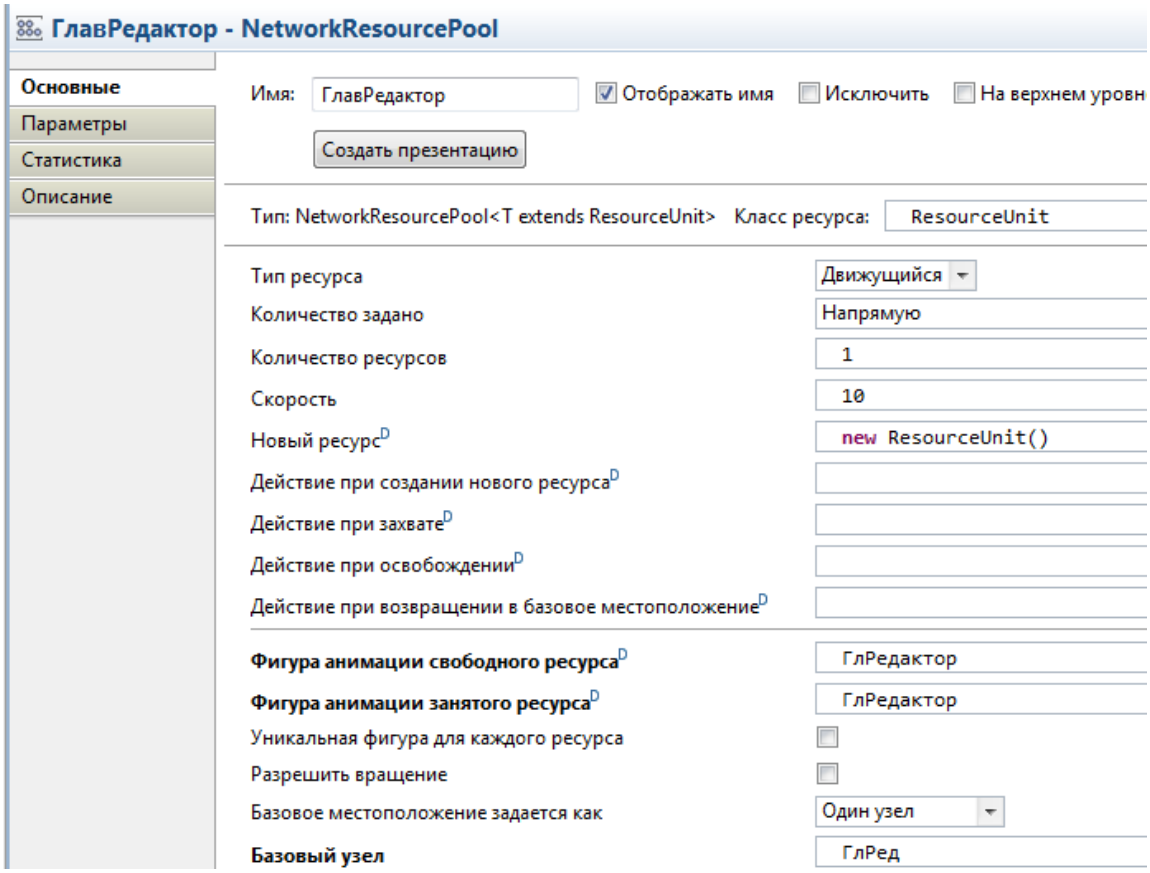


Рис.7.25. Свойства объекта NetworkResourcePool

Таким же образом задаем остальные ресурсы. На приведенных ниже рисунках показаны свойства всех добавленных объектов.

Свойства | Консоль

Редактор_1 - NetworkResourcePool

Основные
Параметры
Статистика
Описание

Имя: Редактор_1 ☑ Отображать имя ☐ Исключить ☐ На верхнем уровне

Создать презентацию

Тип: NetworkResourcePool<T extends ResourceUnit> Класс ресурса: ResourceUnit

Тип ресурса	Движущийся
Количество задано	Напрямую
Количество ресурсов	1
Скорость	10
Новый ресурс[D]	new ResourceUnit()
Действие при создании нового ресурса[D]	
Действие при захвате[D]	
Действие при освобождении[D]	
Действие при возвращении в базовое местоположение[D]	
Фигура анимации свободного ресурса[D]	Редактор1
Фигура анимации занятого ресурса[D]	Редактор1
Уникальная фигура для каждого ресурса	☐
Разрешить вращение	☐
Базовое местоположение задается как	Один узел
Базовый узел	Ред12

Рис.7.26. Изменение свойств обекта NetworkResourcePool

Свойства | Консоль

Редактор_2 - NetworkResourcePool

Основные
Параметры
Статистика
Описание

Имя: Редактор_2 ☑ Отображать имя ☐ Исключить ☐ На верхнем уровне

Создать презентацию

Тип: NetworkResourcePool<T extends ResourceUnit> Класс ресурса: ResourceUnit

Тип ресурса	Движущийся
Количество задано	Напрямую
Количество ресурсов	1
Скорость	10
Новый ресурс[D]	new ResourceUnit()
Действие при создании нового ресурса[D]	
Действие при захвате[D]	
Действие при освобождении[D]	
Действие при возвращении в базовое местоположение[D]	
Фигура анимации свободного ресурса[D]	Редактор2
Фигура анимации занятого ресурса[D]	Редактор2
Уникальная фигура для каждого ресурса	☐
Разрешить вращение	☐
Базовое местоположение задается как	Один узел
Базовый узел	Ред12

Рис.7.27. Изменение свойств обекта NetworkResourcePool

Свойства | Консоль

Редактор_3 - NetworkResourcePool

Основные | Параметры | Статистика | Описание

Имя: Редактор_3 | Отображать имя | Исключить | На верхнем уровне

Создать презентацию

Тип: NetworkResourcePool<T extends ResourceUnit> Класс ресурса: ResourceUnit

Тип ресурса	Движущийся
Количество задано	Напрямую
Количество ресурсов	1
Скорость	10
Новый ресурс	new ResourceUnit()
Действие при создании нового ресурса	
Действие при захвате	
Действие при освобождении	
Действие при возвращении в базовое местоположение	
Фигура анимации свободного ресурса	Редактор_3
Фигура анимации занятого ресурса	Редактор_3
Уникальная фигура для каждого ресурса	
Разрешить вращение	
Базовое местоположение задается как	Один узел
Базовый узел	Ред3

Рис.7.28. Изменение свойств обекта NetworkResourcePool

Свойства | Консоль

Корректор_ - NetworkResourcePool

Основные | Параметры | Статистика | Описание

Имя: Корректор_ | Отображать имя | Исключить | На верхнем уровне

Создать презентацию

Тип: NetworkResourcePool<T extends ResourceUnit> Класс ресурса: ResourceUnit

Тип ресурса	Движущийся
Количество задано	Напрямую
Количество ресурсов	1
Скорость	10
Новый ресурс	new ResourceUnit()
Действие при создании нового ресурса	
Действие при захвате	
Действие при освобождении	
Действие при возвращении в базовое местоположение	
Фигура анимации свободного ресурса	Корректор
Фигура анимации занятого ресурса	Корректор
Уникальная фигура для каждого ресурса	
Разрешить вращение	
Базовое местоположение задается как	Один узел
Базовый узел	Корр

Рис.7.29. Изменение свойств обекта NetworkResourcePool

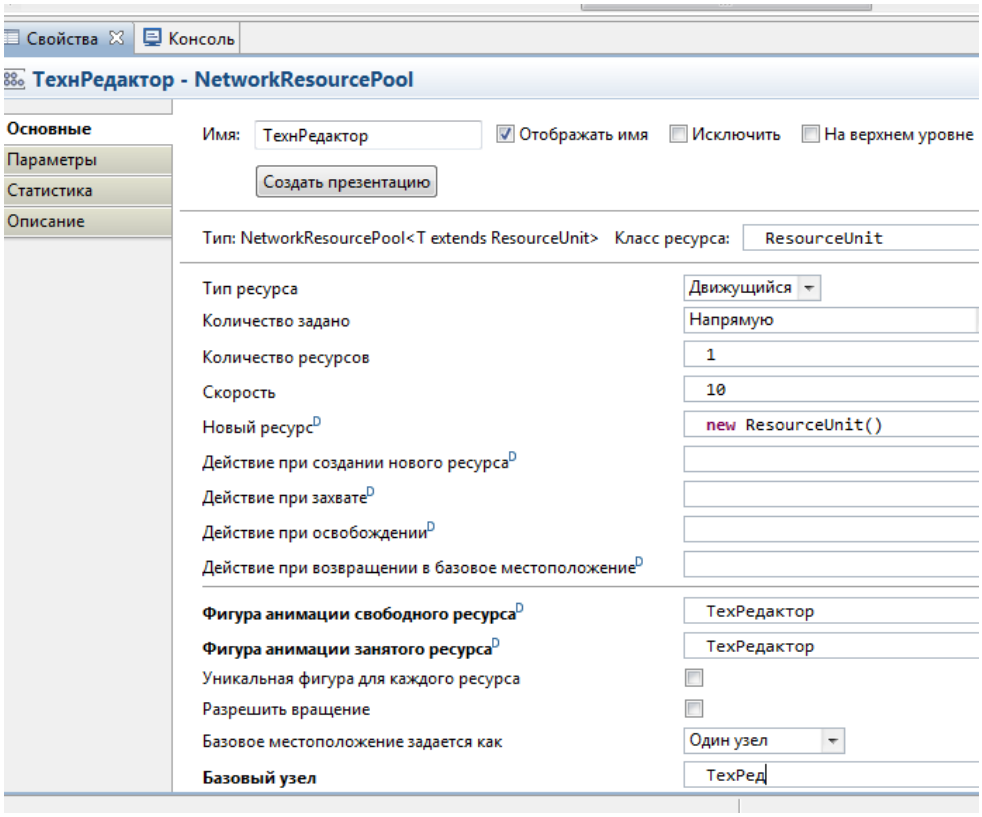


Рис.7.30. Изменение свойств обекта NetworkResourcePool

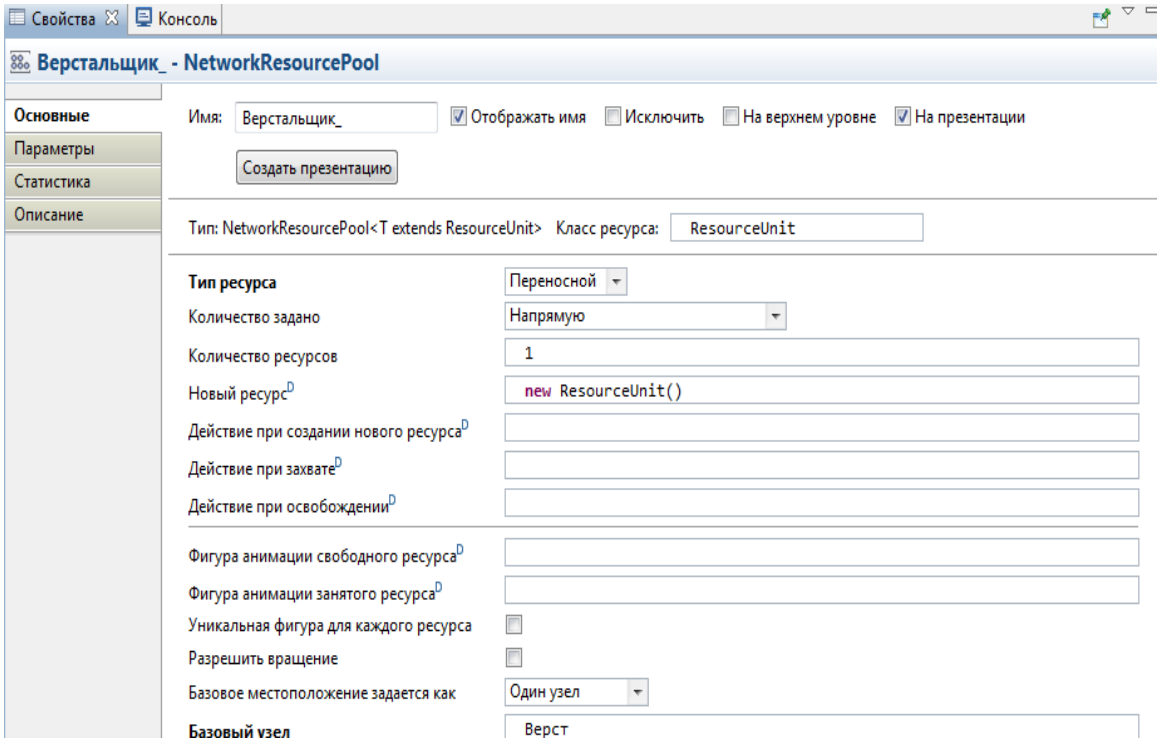


Рис.7.31. Изменение свойств обекта NetworkResourcePool

Задаем ресурсы типа «кабинеты редакторов».

1. Добавляем еще один объект NetworkResourcePool. Этот объект будет задавать свойства ресурсов, представляющих в нашей модели эти кабинеты. Задаем следующие свойства объекта:

- называем объект РедКаб.
- изменяем Тип ресурса на Статический.
- задаем местоположение кабинетов. Статические ресурсы всегда находятся в месте, указанном как базовое. Можно задать несколько таких мест, для этого нужно нарисовать ломаную линию с точками, лежащими в соответствующих прямоугольниках и указать ее в свойствах объекта. Поскольку в нашей модели две таких комнаты, то мы должны воспользоваться именно этим способом. Для этого выбрали Путь через узлы из выпадающего списка Базовое местоположение задается как, и ввели в поле Путь через узлы имя созданной ранее именно для этой цели ломаной: Линия.
- Указали, что количество ресурсов, задаваемое этим объектом, равно количеству точек указанной ломаной. Для этого выбираем из группы кнопок Количество задано опцию Фигурой базового местоположения.

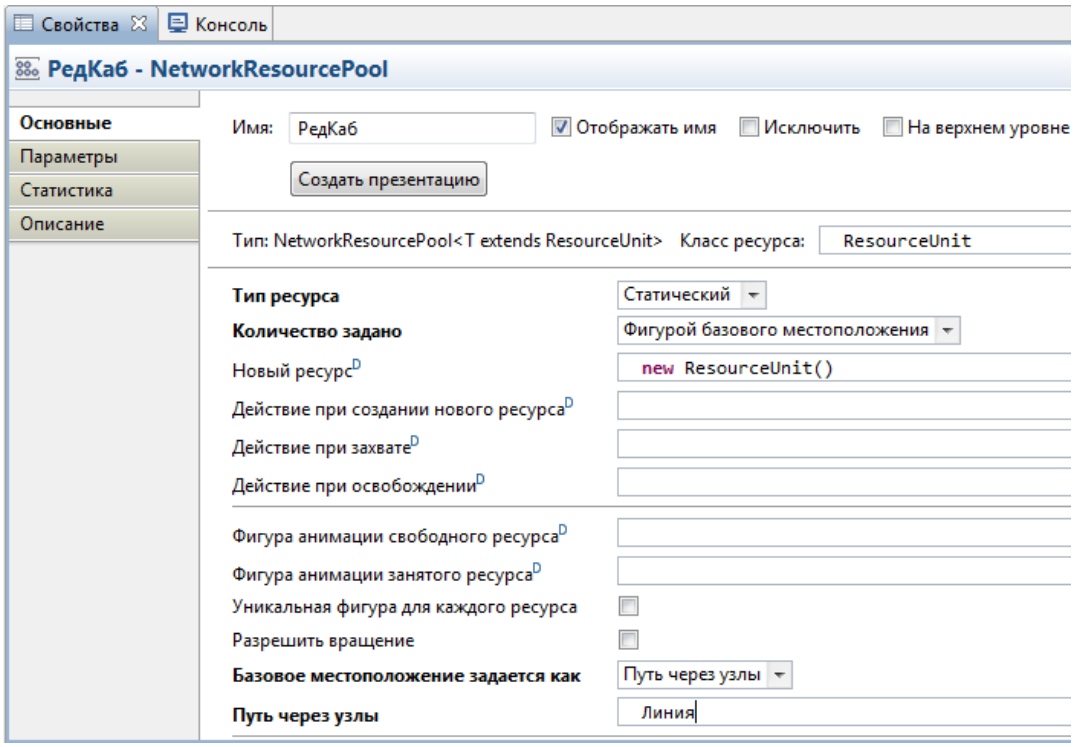


Рис.7.32. Изменение свойств обекта NetworkResourcePool

Соединение объектов NetworkResourcePool с объектом Network.

1. Чтобы добавить заданные ресурсы в нашу сеть, нужно соединить порты объектов NetworkResourcePool с портом объекта Network. Соединили их, как показано на рисунке ниже:

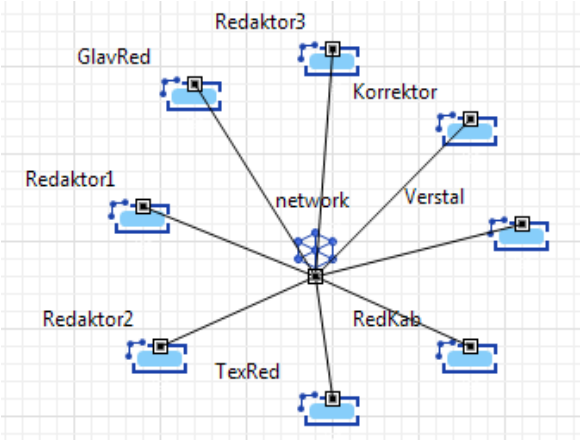


Рис.7.33. Соединенные объекты

Шаг 5. Диаграмма процесса

Создание диаграммы процесса.

1. Создали следующую диаграмму процесса, добавив на диаграмму класса Main новые блоки Основной библиотеки и соединив их так, как показано на рисунке ниже:

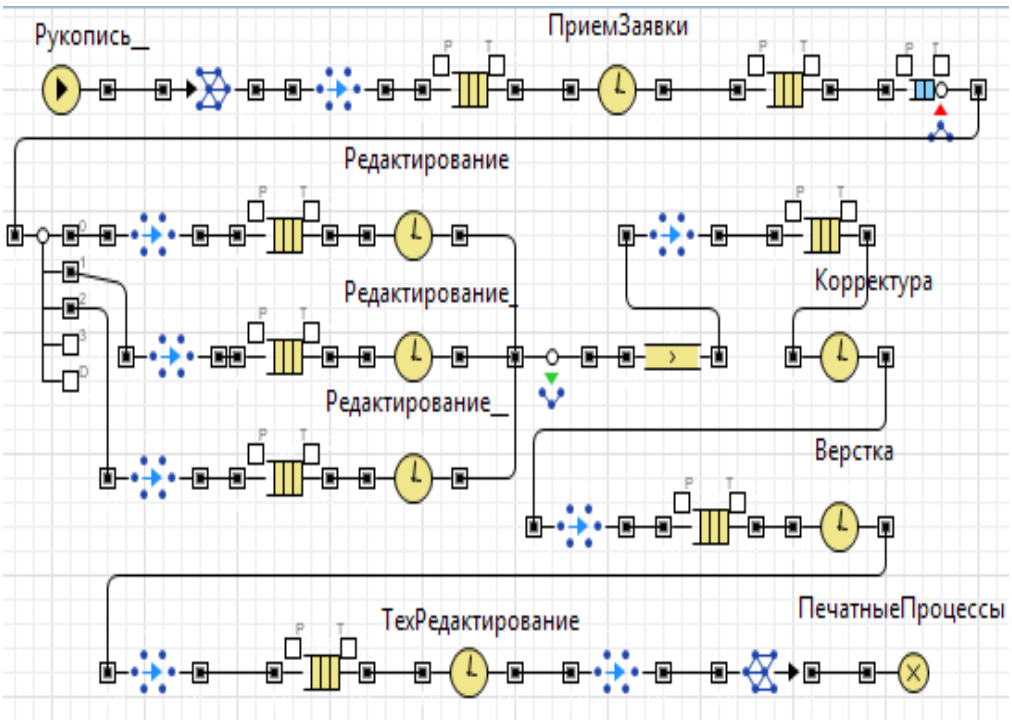


Рис.7.34. Диаграмма процесса

Эта диаграмма будет описывать моделируемый нами процесс.

Первым следует объект Source. Объект Source создает заявки. Обычно он используется в качестве начальной точки потока заявок. В нашем случае заявки будут представлять собой авторов, и этот блок будет моделировать поступление авторов в отдел.

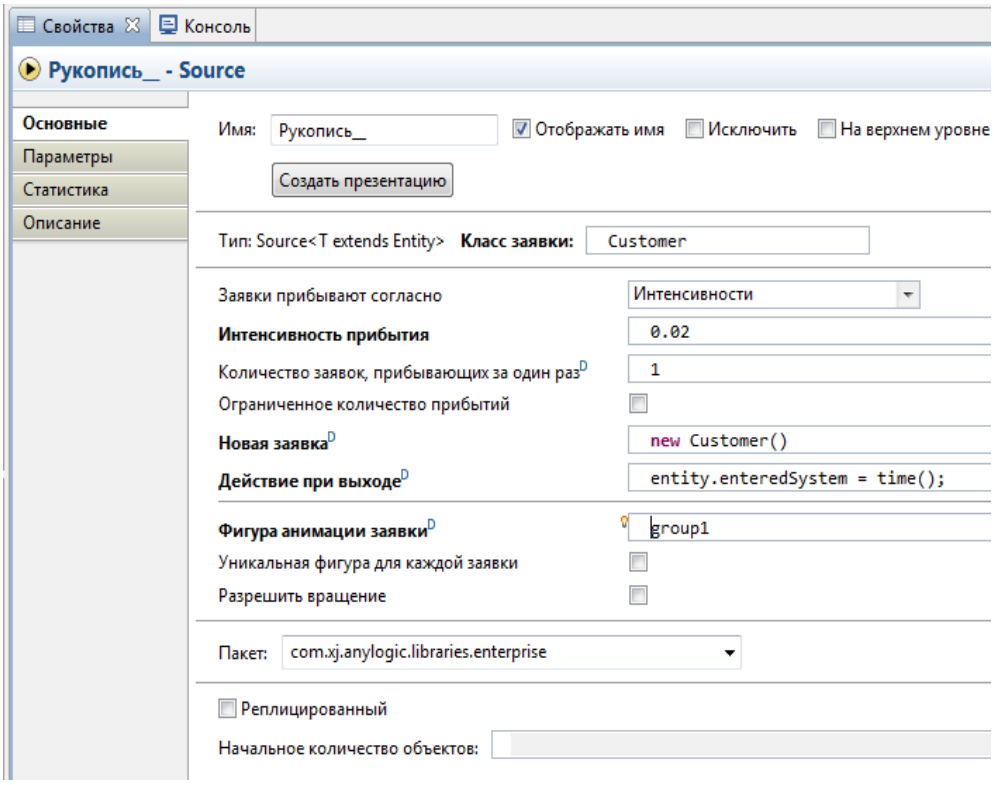


Рис.7.35. Изменение свойств объекта Source

Изменение свойств объекта Source.

1. В поле Интенсивность прибытия задаем интенсивность поступления авторов в отдел: 0.02.

2. В поле Фигура анимации заявки вводим имя ранее сгруппированных фигур: group1.

За этим объектом следует блок NetworkEnter. Этот объект добавляет заявки в заданное место сети, в нашем случае он помещает авторов с рукописями в кабинет главного редактора.

Изменение свойств объекта networkEnter.

1. Вводим Выход в поле Узел входа. Здесь Выход – это имя ранее нарисованного нами прямоугольника, который располагается на плане отдела. Заявки-авторы будут прибывать в указанный узел сети.

2. В поле Сеть можно увидеть введенное Мастером создания модели имя объекта Network (network). Именно в сеть, заданную этим объектом, и будут помещаться заявки этим блоком.

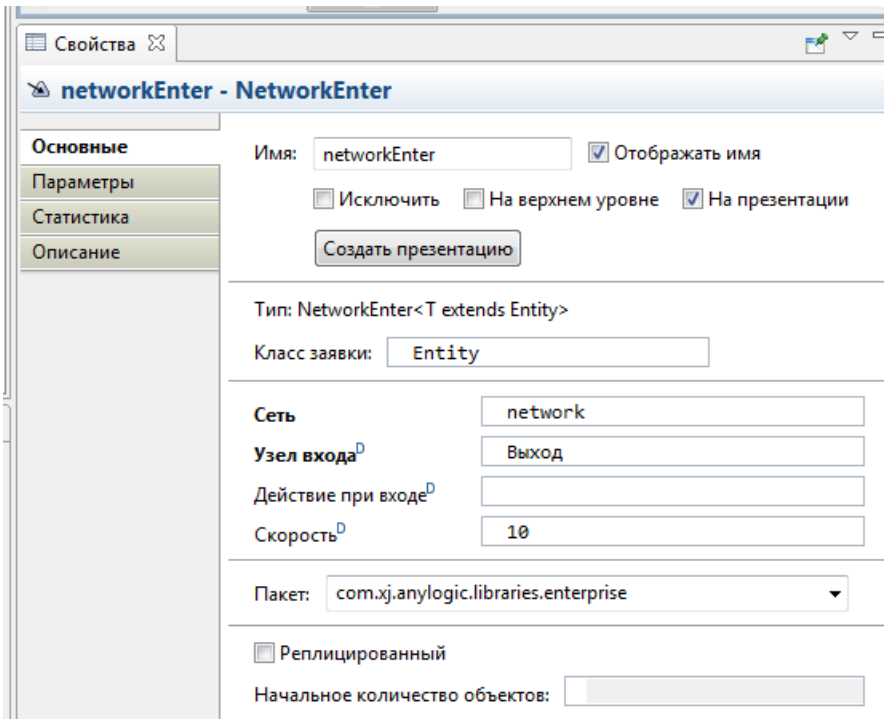


Рис.7.36 Изменение свойств NetworkEnter

Изменение свойств объектов networkMoveTo.

Следующий блок в диаграмме процесса – NetworkMoveTo. Объект NetworkMoveTo перемещает заявку в новое место сети. С помощью этого объекта моделируем, как авторы переходят из кабинета главного редактора в кабинет редактора. Задаем следующие свойства объектов:

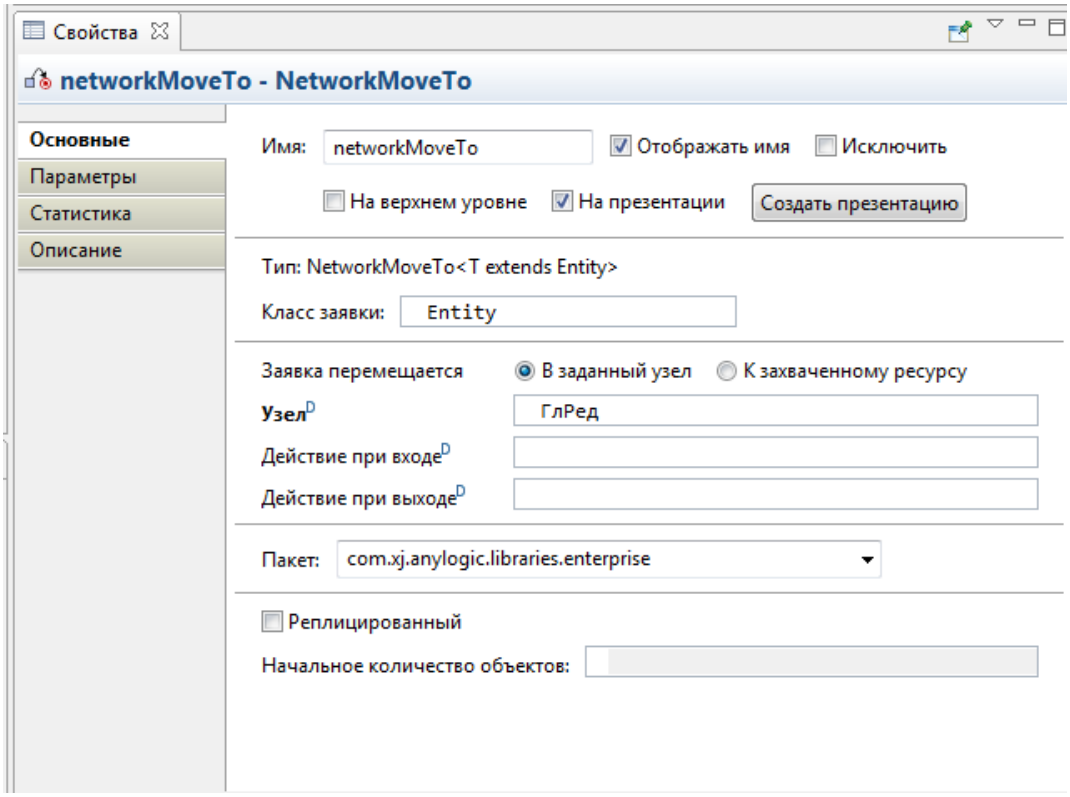


Рис.7.37. Изменение свойств объекта NetworkMoveTo

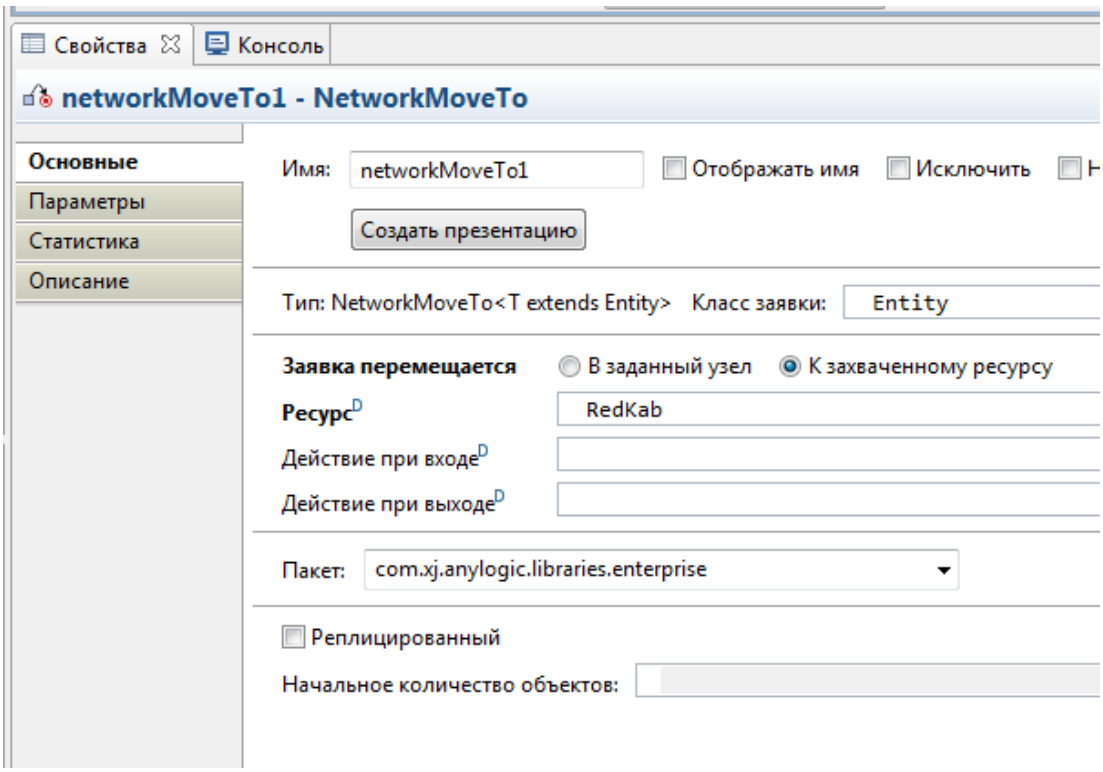


Рис.7.38. Изменение свойств объекта NetworkMoveTo

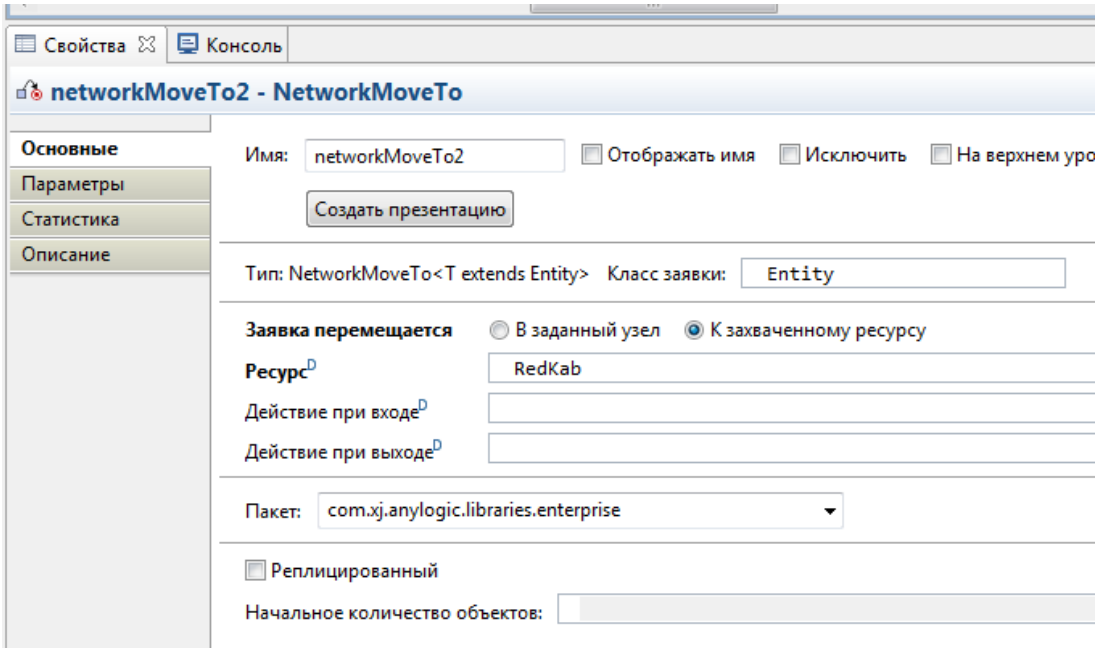


Рис.7.39. Изменение свойств объекта NetworkMoveTo

Еще один объект NetworkMoveTo моделирует переход авторов в кабинет корректора.

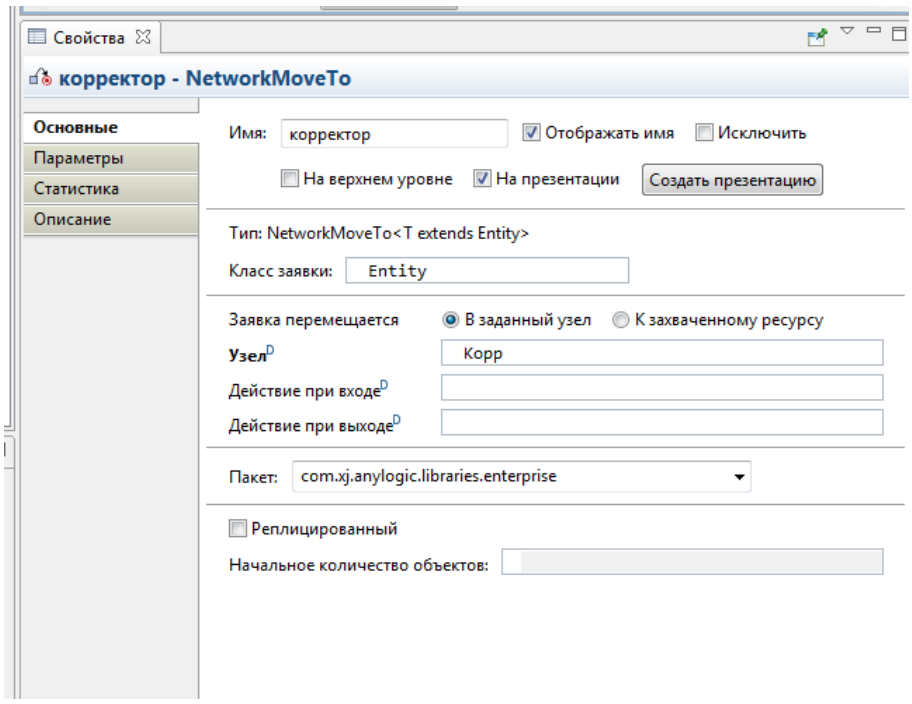


Рис.7.40. Изменение свойств объекта NetworkMoveTo

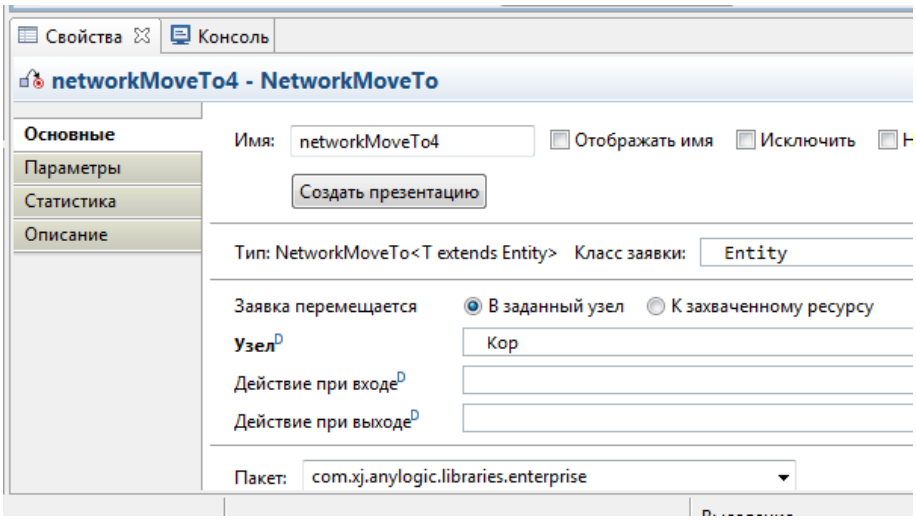


Рис.7.41. Изменение свойств объекта NetworkMoveTo

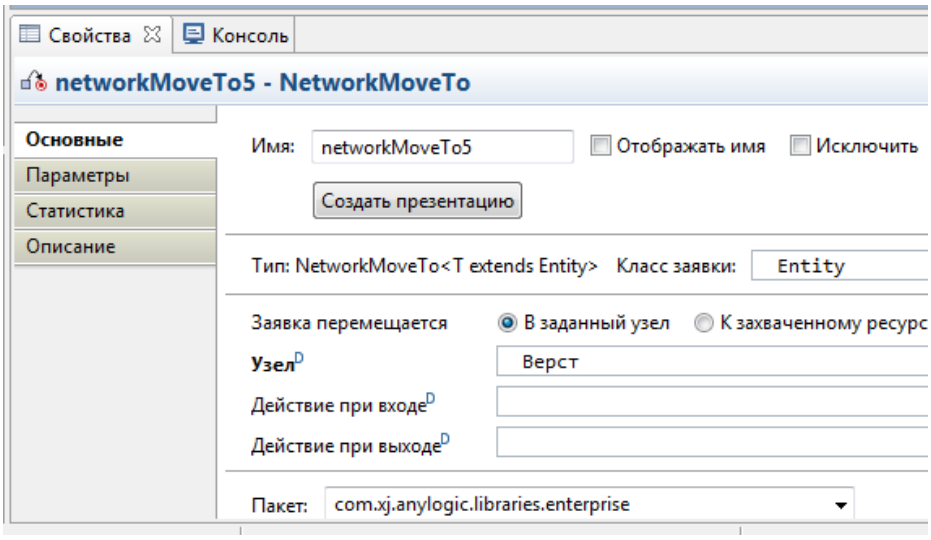


Рис.7.41. Изменение свойств объекта NetworkMoveTo

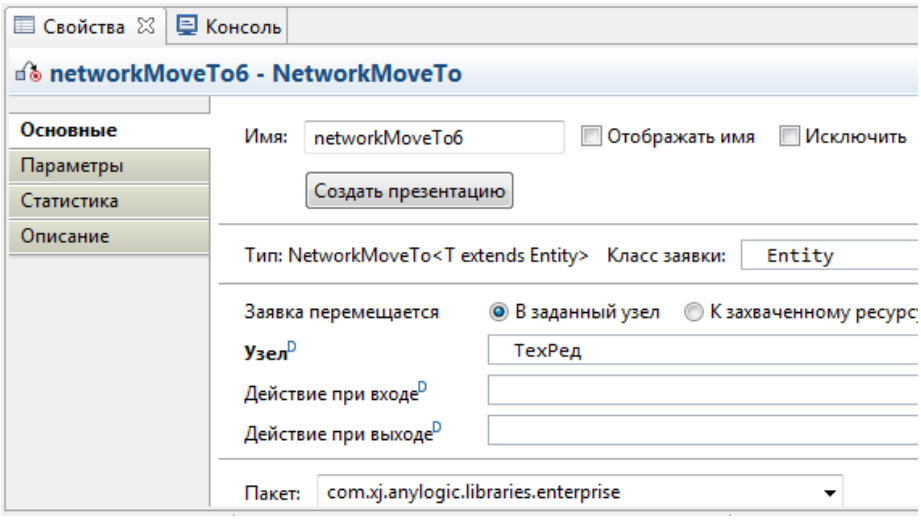


Рис.7.42. Свойства объекта NetworkMoveTo

Еще один объект NetworkMoveTo моделирует переход авторов в печатные процессы.

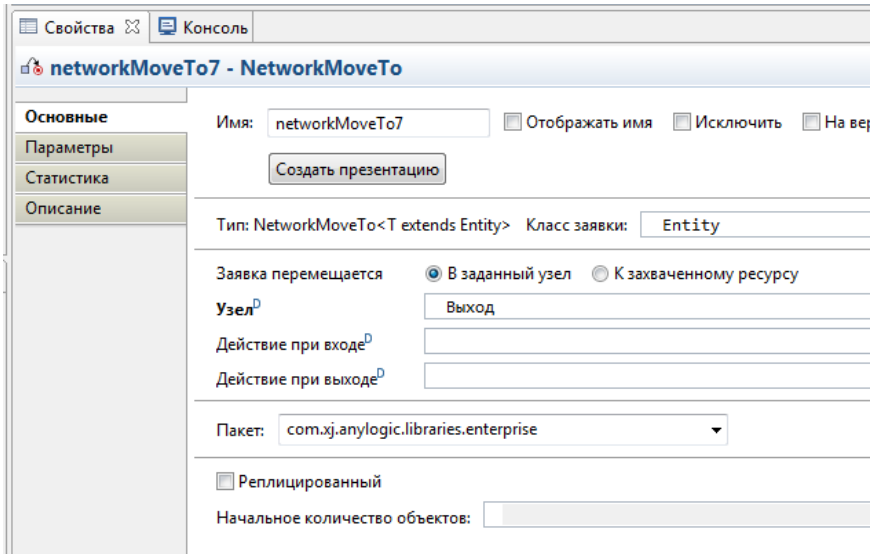


Рис.7.43. Свойства объекта NetworkMoveTo

Изменение свойств объектов Queue.

Следующий объект Queue хранит заявки в определенном порядке. Моделирует очередь заявок, ожидающих приема объектами, следующими за данным в потоковой диаграмме.

Изменяем только поле вместимость ставим 15, таким образом, в очереди будет находиться не более 15 рукописей.

Аналогично изменяем свойства всех объектов Queue.

Изменение свойств объектов Delay.

Объект Delay задерживает заявку на заданное время. В нашей модели этот объект моделирует задержку, связанную с проведением приема заявки. Задаем следующие свойства:

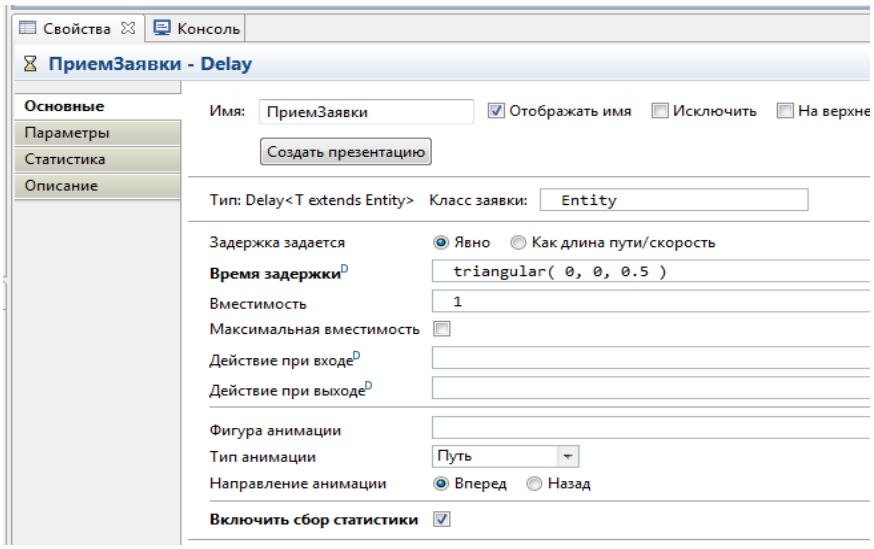


Рис.7.44. Изменение свойств объекта Delay

Аналогично изменяем свойства всех объектов Delay, изменяя лишь временные значения.

Изменение свойств объекта Network Seize.

Объект захватывает для заявки заданное количество сетевых ресурсов. При необходимости может пересылать захваченные ресурсы в заданное место сети и (опять же, при необходимости) присоединять их к заявке. Задаем следующие свойства:

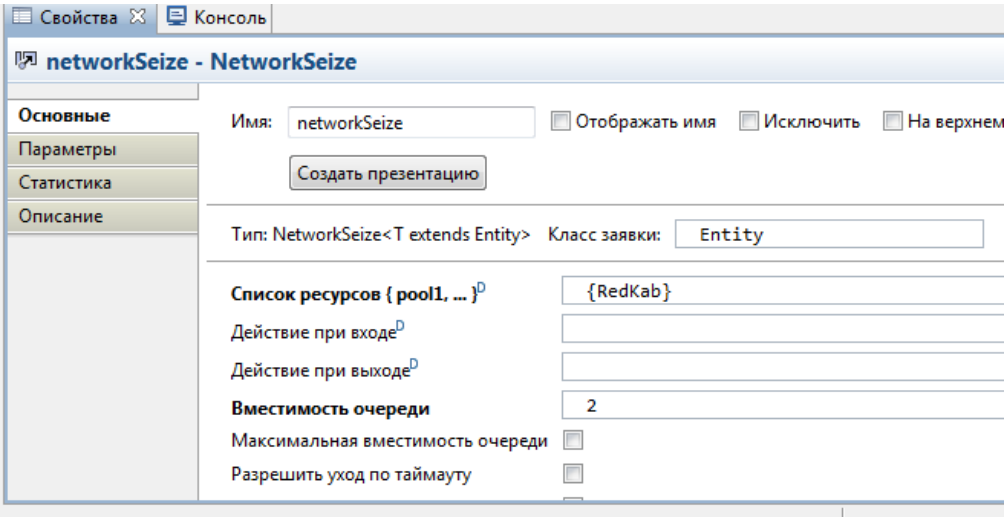


Рис.7.45. Изменение свойств объекта Network Seize

Изменение свойств объекта Network Release.

Освобождает ранее захваченные заявкой сетевые ресурсы. Задаем следующие свойства:

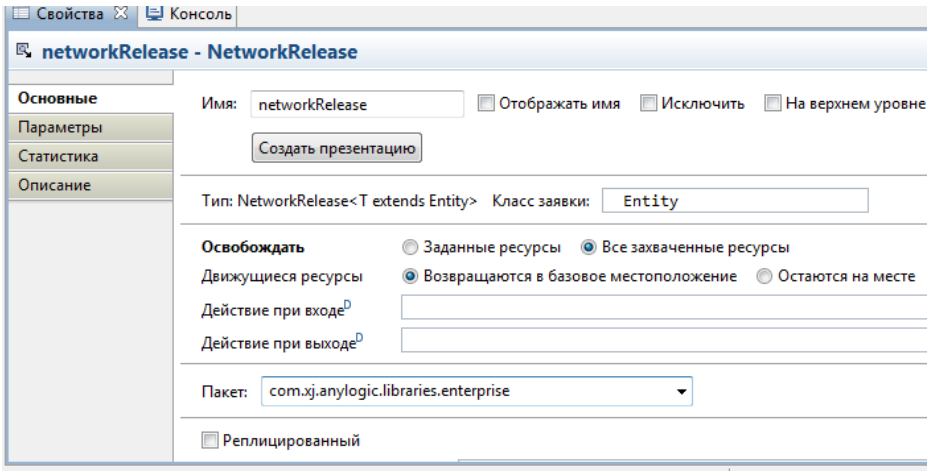


Рис.7.46. Изменение свойств объекта Network Release

Изменение свойств объекта Conveyor.

Моделирует конвейер. Перемещает заявки по пути заданной длины с заданной скоростью (одинаковой для всех заявок), сохраняя их порядок и оставляя заданные промежутки между ними.

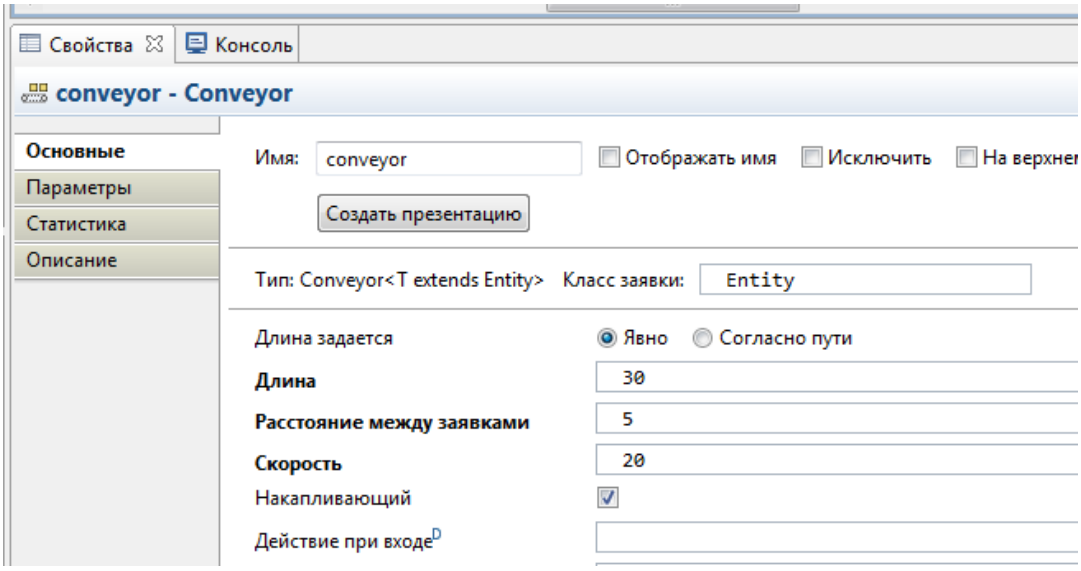


Рис.7.47. Изменение свойств объекта Conveyor

Объект NetworkExit удаляет заявку из сети. Заявка при этом перестает отображаться на анимации сети. В нашем случае он моделирует переход авторов в печатные процессы. Оставляем свойства объекта NetworkExit без изменений.

Объект Sink уничтожает поступившие заявки. Обычно он используется в качестве конечной точки диаграммы процесса. Оставляем свойства объекта Sink без изменений.

Шаг 6. Запуск модели

Щелкнули мышью по кнопке панели инструментов Запустить и выбрали из открывшегося списка наш эксперимент. Эксперимент модели называется Pre-press.

После запуска открылось окно презентации нашей модели. В нем отображена презентация запущенного эксперимента. AnyLogic автоматически помещает на презентацию каждого простого эксперимента заголовок и кнопку, позволяющую запустить модель и перейти на презентацию, нарисованную для главного класса активного объекта этого эксперимента (Main).

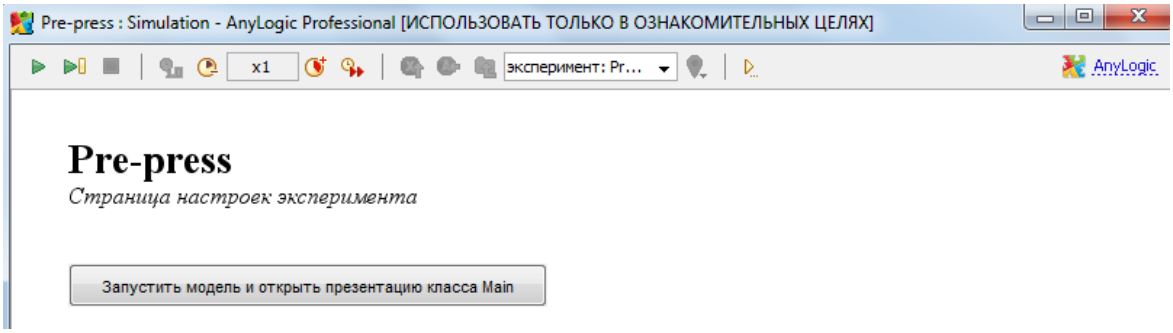


Рис.7.48. Запуск модели

Щелкнули по этой кнопке. Тем самым запустили модель, и перешли к презентации корневого класса активного объекта запущенного эксперимента.

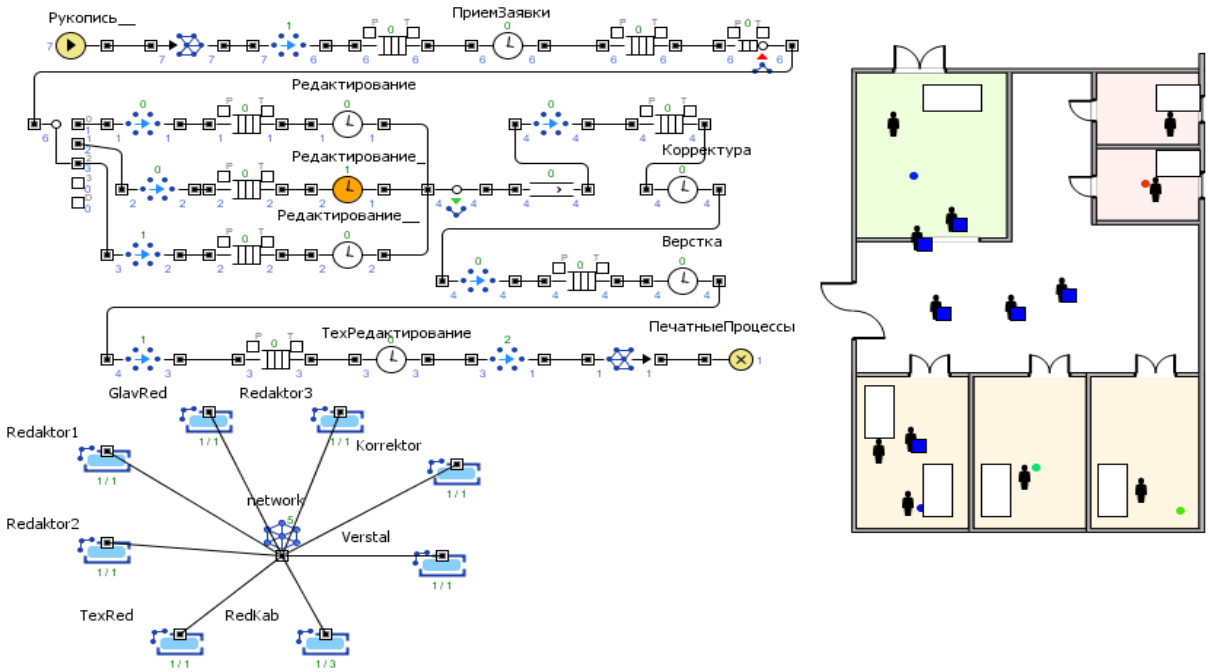


Рис.7.49. Полученная модель допечатных процессов

Шаг 7. Сбор статистики использования ресурсов.

Нужно включить сбор статистики для объектов delay и queue. Просмотрим интересующую нас статистику занятости редакционного отдела и длины очереди с помощью диаграммы.

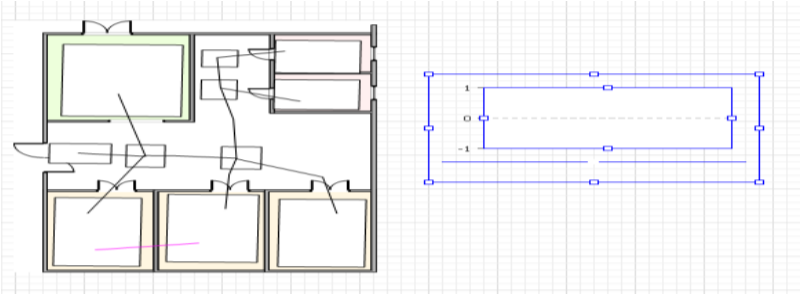


Рис.7.50. Вставка диаграммы

Добавим диаграмму для отображения средней занятости отдела

1. Открываем палитру Статистика. Эта палитра содержит элементы сбора данных и статистики, а также диаграммы для визуализации данных и результатов моделирования.
2. Перетащим элемент Столбиковая диаграмма из палитры Статистика на диаграмму класса и изменим ее размер, как показано на рисунке ниже:
3. Перейдем на страницу Основные панели Свойства. Щелкнем мышью по кнопке Добавить элемент данных. При этом появилась секция свойств того элемента данных, который будет отображаться на этой диаграмме.
4. Измените Заголовок на Прием заявки.
5. Введите ПриемЗаявки.statsUtilization.mean() в поле Значение. Здесь ПриемЗаявки - это имя нашего объекта Delay. У каждого объекта Delay есть встроенный набор данных statsUtilization, занимающийся сбором статистики использования этого объекта. Функция mean() возвращает среднее из всех измеренных этим набором данных значений. Аналогично добавляем все необходимые элементы данных.

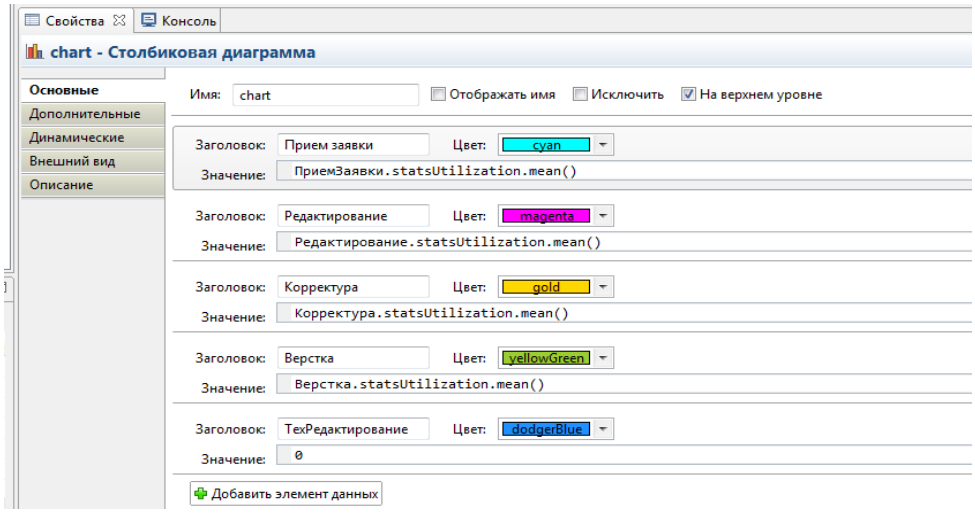


Рис.7.51. Свойства объекта chart

6. На страницу Внешний вид панели Свойства выбрали первую опцию из набора кнопок Расположение, чтобы изменить расположение легенды относительно диаграммы (мы хотим, чтобы она отображалась снизу).

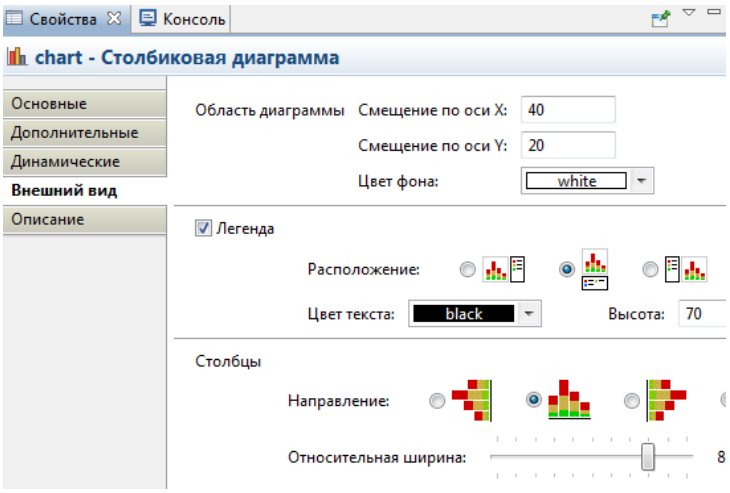


Рис.7.52. Свойства объекта chart

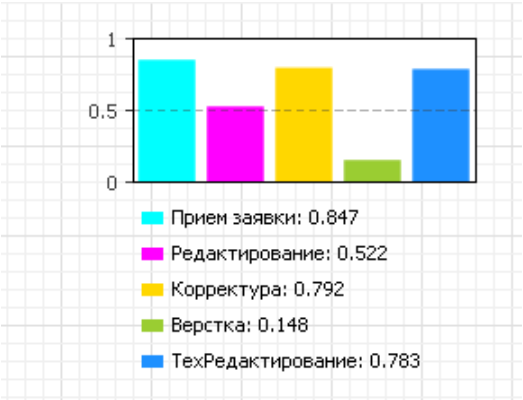


Рис.7.53. Диаграмма отображает среднюю занятость отдела

Сбор статистики по времени обслуживания

Определим время прохождения рукописи по этапам допечатной подготовки с помощью специальных объектов сбора данных и отобразим собранную статистику распределения времени обслуживания клиентов с помощью гистограммы.

Для этого создадим Java класс Customer. Экземпляры этого класса будут использоваться в нашей модели в качестве заявок, и будут представлять авторов. Мы создадим в этом классе специальные поля для запоминания необходимой нам информации о проведенном автором времени.

Создание Java класса Customer

1. В панели Проекты щелкнули правой кнопкой мыши по элементу модели и выбрали Создать|Java класс из контекстного меню.

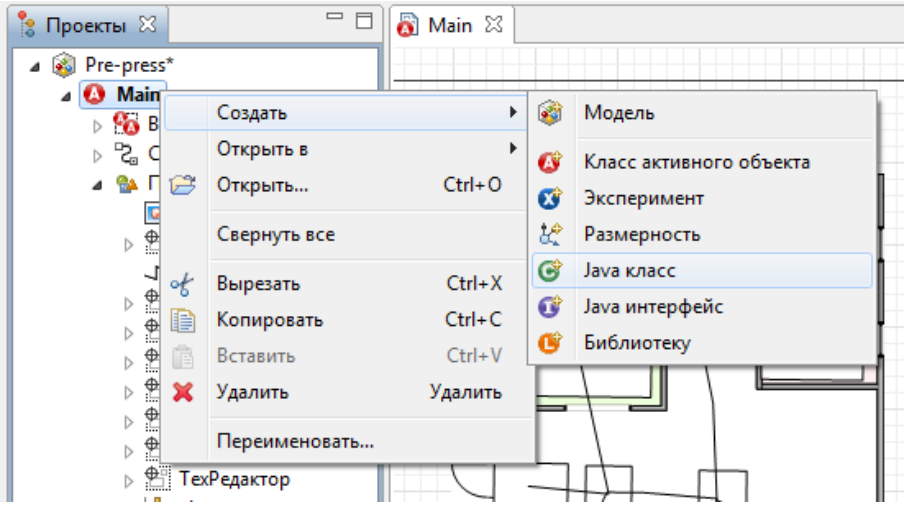


Рис.7.54. Создание Java класса

2. Появилось диалоговое окно Новый Java класс. В поле Имя ввели имя нового класса: Customer.

3. Сделали так, что этот класс будет наследоваться от базового класса заявки Entity. Для этого выбрали из выпадающего списка Базовый класс полное имя этого класса: com.xj.anylogic.libraries.enterprise.Entity.

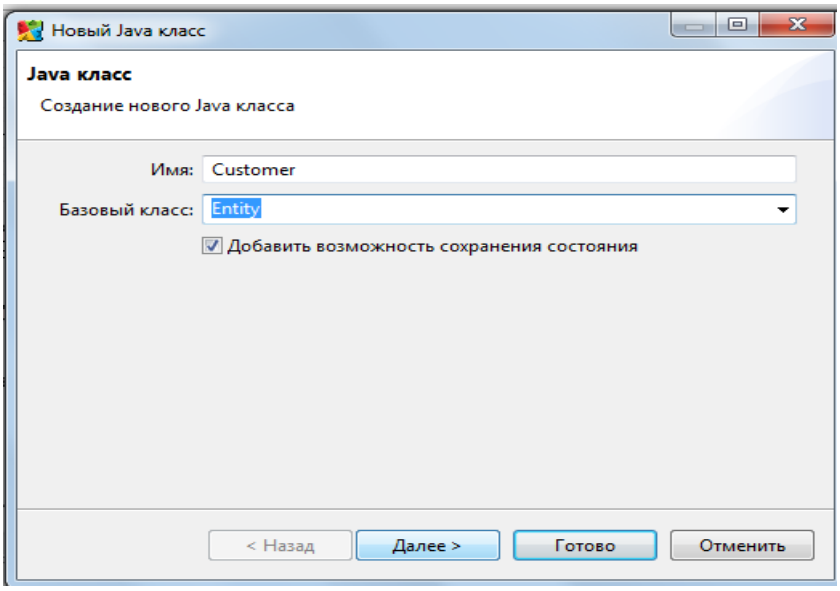


Рис.7.55. Окно Мастера создания Java класса

3. Щелкнули мышью по кнопке Далее, чтобы перейти к следующей странице Мастера создания Java класса. На второй странице Мастера задали параметры создаваемого Java класса:

- enteredSystem типа double для сохранения момента времени, когда автор пришел в отдел;
- startWaiting типа double для сохранения момента времени, когда автор встал в очередь.

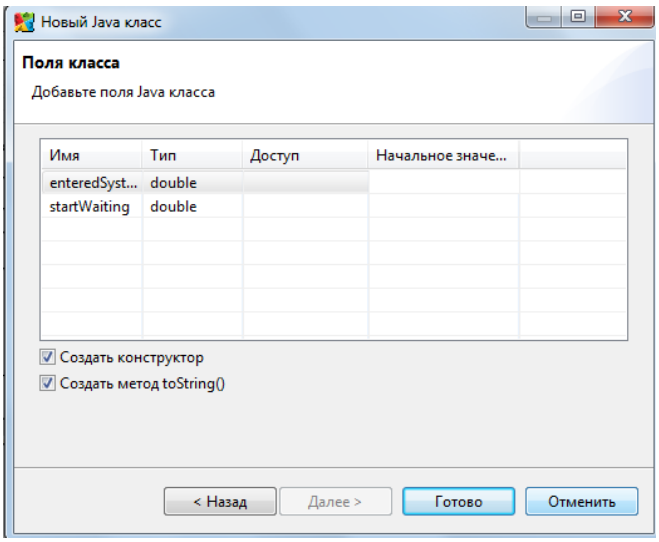


Рис.7.56. Окно Мастера создания Java класса

4. Щелкнули мышью по кнопке Готово. Появился редактор кода созданного класса. Закрыть его, щелкнув мышью по крестику в закладке с его названием.

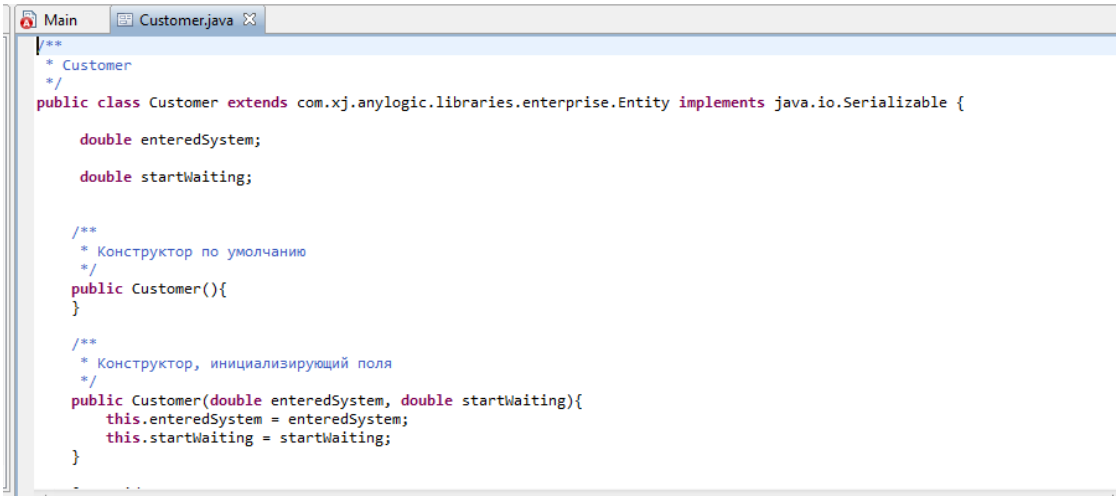


Рис.7.57. Окно Java класса

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

Добавление элементов сбора данных

1. Чтобы добавить объект сбора данных гистограммы на диаграмму, перетащили элемент Данные гистограммы с палитры Статистика на диаграмму активного класса.

2. Задали свойства элемента:

- изменили Имя на waitTimeDistr,
- задали Кол-во интервалов равным 50,
- задали Начальный размер интервала: 0.01.

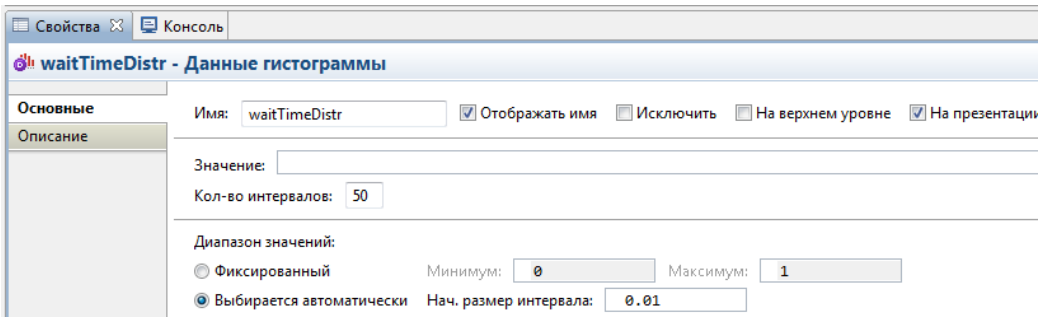


Рис.7.58. Свойства объекта data

3. Создали еще один элемент сбора данных гистограммы. Ctrl+перетащили только что созданный объект данных гистограммы, чтобы создать его копию.

4. Изменили Имя этого элемента на timeInSystemDistr.

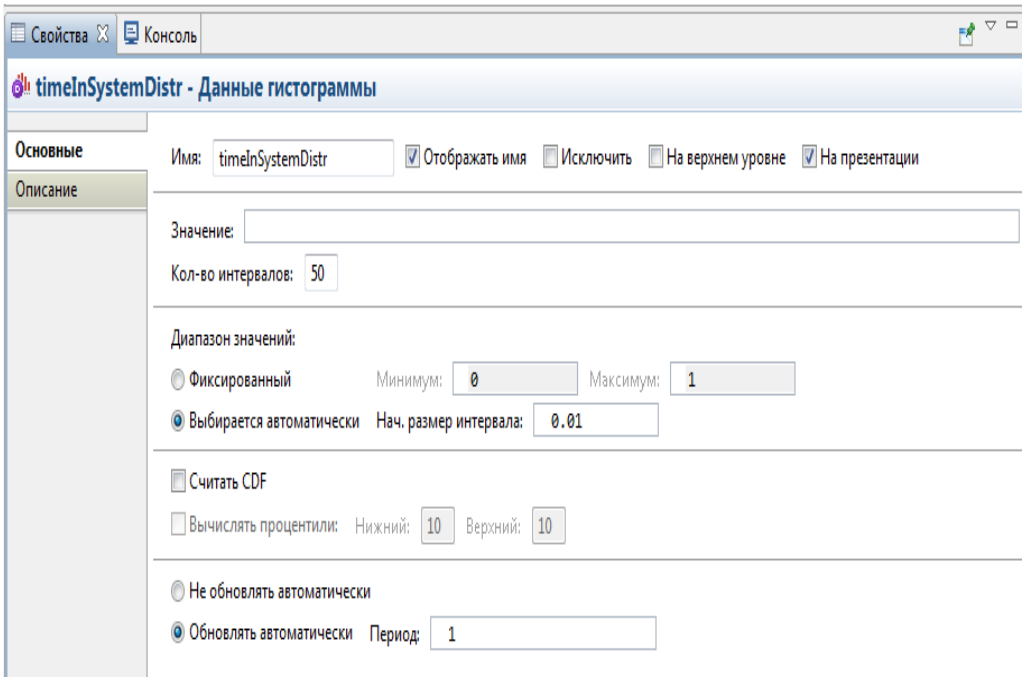


Рис.7.59. Свойства объекта data

Теперь нам нужно изменить свойства блоков нашей диаграммы процесса.

Изменение свойств блоков диаграммы процесса

1. Изменили свойства объекта source:

- ввели new Customer() в поле Новая заявка. Теперь этот объект будет создавать заявки нашего типа Статистика.

- ввели Customer в поле Класс заявки. Это позволит напрямую обращаться к полям класса заявки Customer в коде динамических параметров этого объекта.
- ввели entity.enteredSystem = time(); в поле Действие при выходе. Этот код будет сохранять время создания заявки-автора в переменной enteredSystem нашего класса заявки Customer.

Функция time() возвращает текущее значение модельного времени.

2. Изменили свойства объекта queue:

- ввели Customer в поле Класс заявки;
- ввели entity.startWaiting = time(); в поле Действие при входе. Этот код запоминает время начала ожидания клиентом его очереди на обслуживание в переменной startWaiting нашего класса заявки Customer;

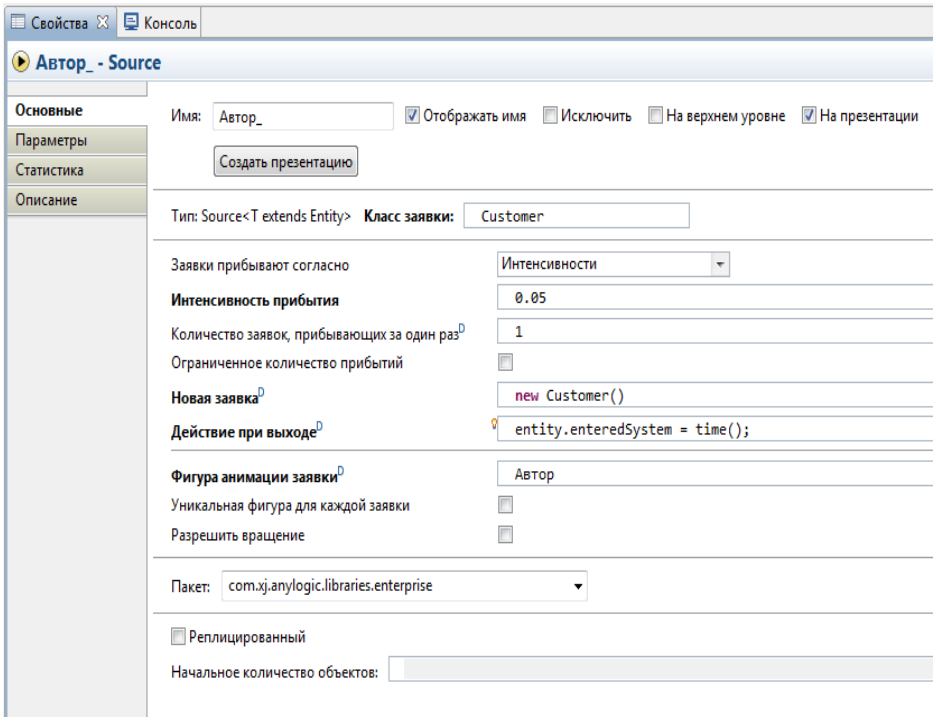


Рис.7.60. Изменение свойств объекта Source

- ввели waitTimeDistr.add(time() - entity.startWaiting); в поле Действие при выходе. Этот код добавляет время, в течение которого автор ожидал обслуживания, в объект сбора данных waitTimeDistr.

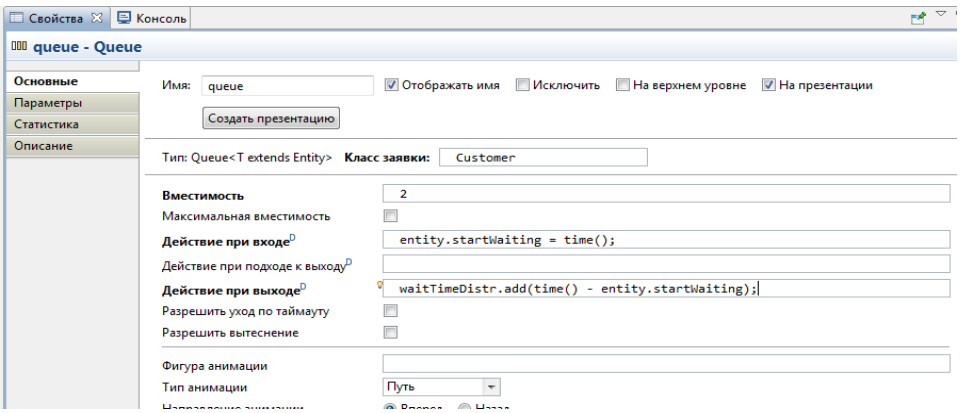


Рис.7.61. Изменение свойств объекта Queue

3. Изменили свойства объекта sink:

- ввели Customer в поле Класс заявки;
- ввели timeInSystemDistr.add(time()-entity.enteredSystem); в поле Действие при входе. Этот код добавляет полное время пребывания клиента в банковском отделении в объект сбора данных гистограммы timeInSystemDistr.

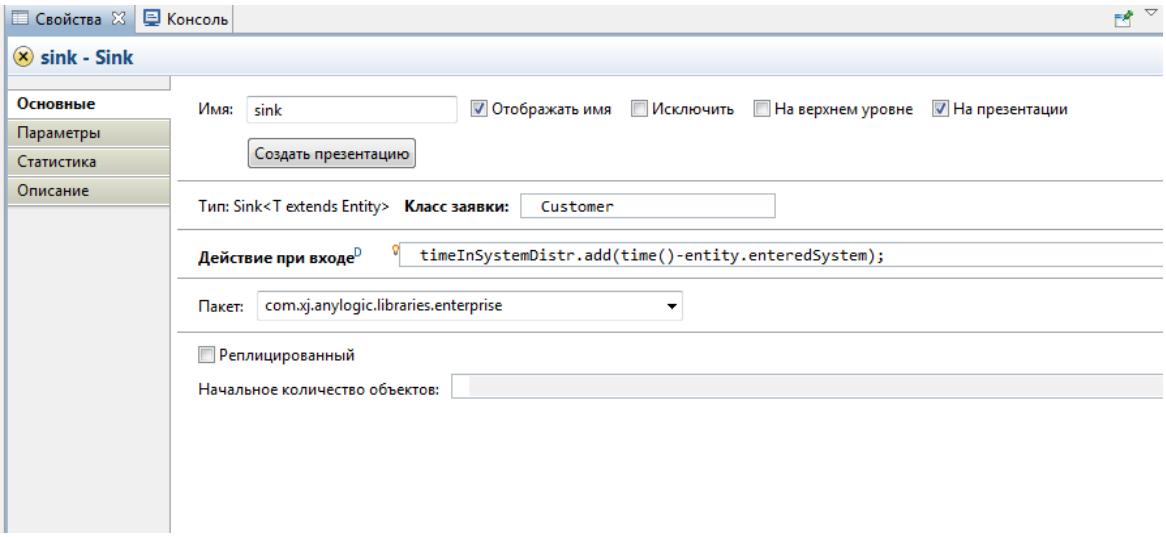


Рис.7.62. Изменение свойств объекта Sink

Запустили модель и просмотрели статистику с помощью окон инспекта. Открыть окно инспекта можно, щелкнув мышью по значку объекта сбора данных. Здесь увидели стандартные для статистического анализа данные, приведенные для значений, собранных в данном объекте сбора статистики (см. рисунок):

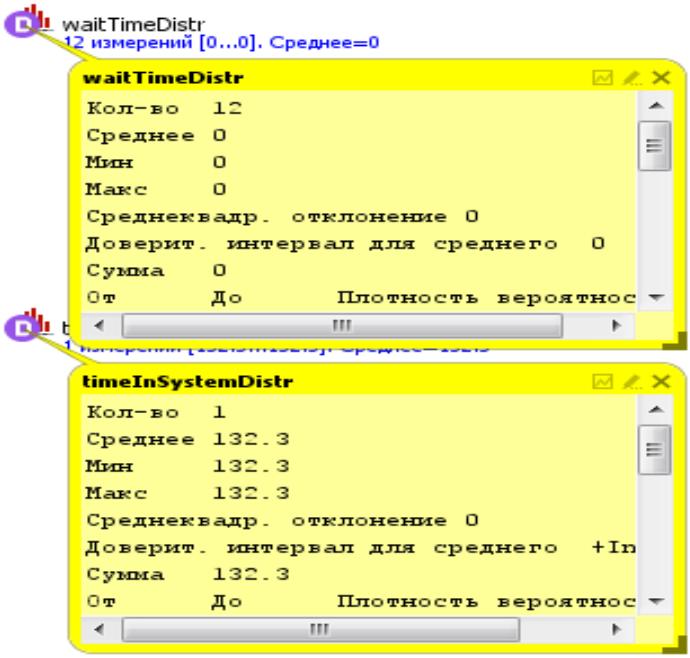


Рис.7.63. Окна инспекта с данными

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

Добавление двух гистограмм для отображения распределений времени ожидания авторов и пребывания авторов в отделе.

1. Чтобы добавить гистограмму на диаграмму класса активного объекта, перетащили элемент Гистограмма из палитры Статистика на диаграмму активного класса.

2. Указали, какой элемент сбора данных хранит данные, которые необходимо отображать на гистограмме: щелкнули мышью по кнопке Добавить данные и ввели в поле Данные имя соответствующего элемента: waitTimeDistr. Изменили Заголовок отображаемых данных на Распределение времени ожидания.

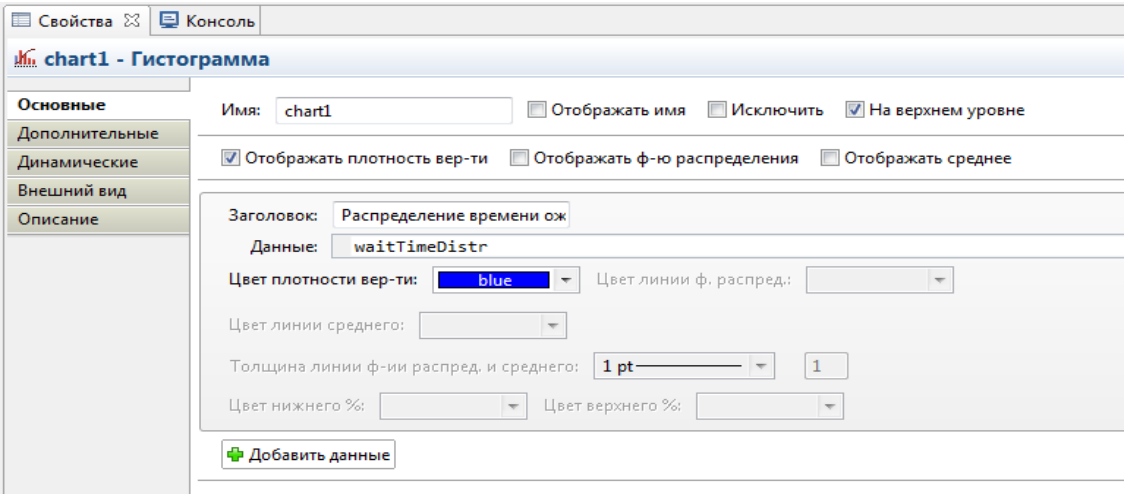


Рис.7.64. Свойства объекта Гистограмма

3. Добавили еще одну гистограмму и расположили ее под ранее добавленной.

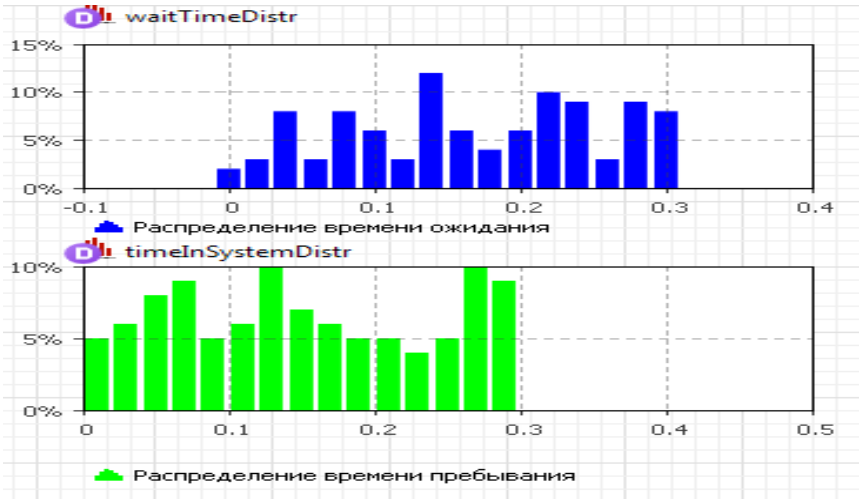


Рис.7.65. Добавленные гистограммы

4. В поле Данные ввели имя элемента, хранящего данные, которые будут отображаться на гистограмме: timeInSystemDistr. Изменили, Заголовок отображаемых данных на Распределение времени пребывания в системе.

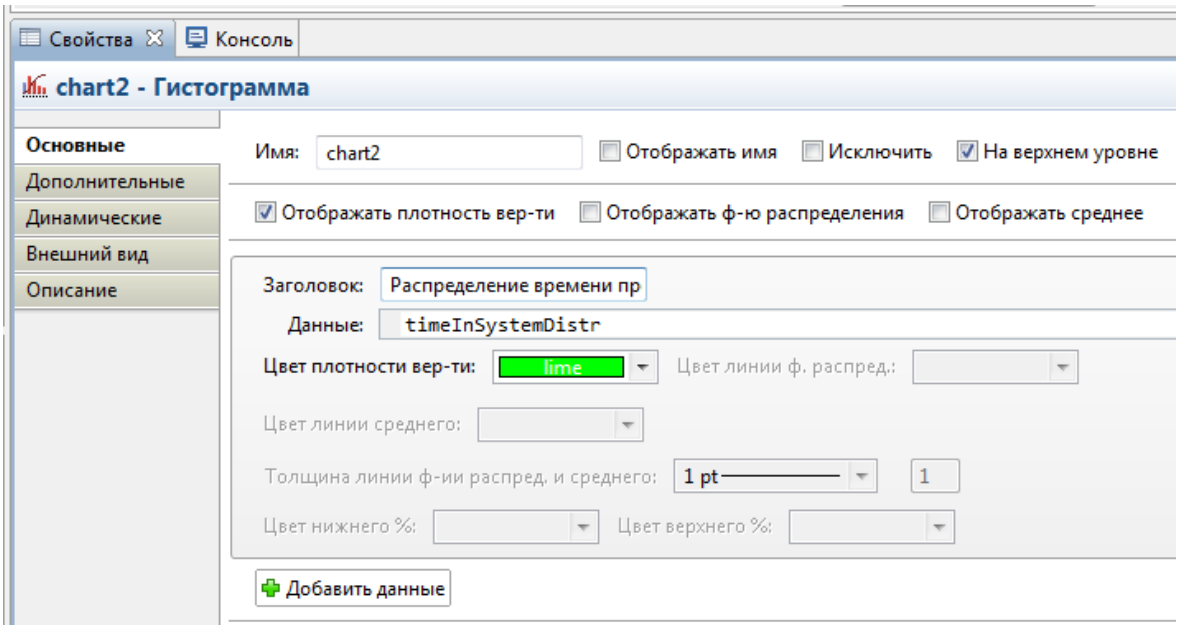


Рис.7.66. Свойства объекта Гистограмма

Запустили модель. Включили режим виртуального времени и понаблюдали за тем, какой вид примет распределение времени ожидания и пребывания авторов в системе.

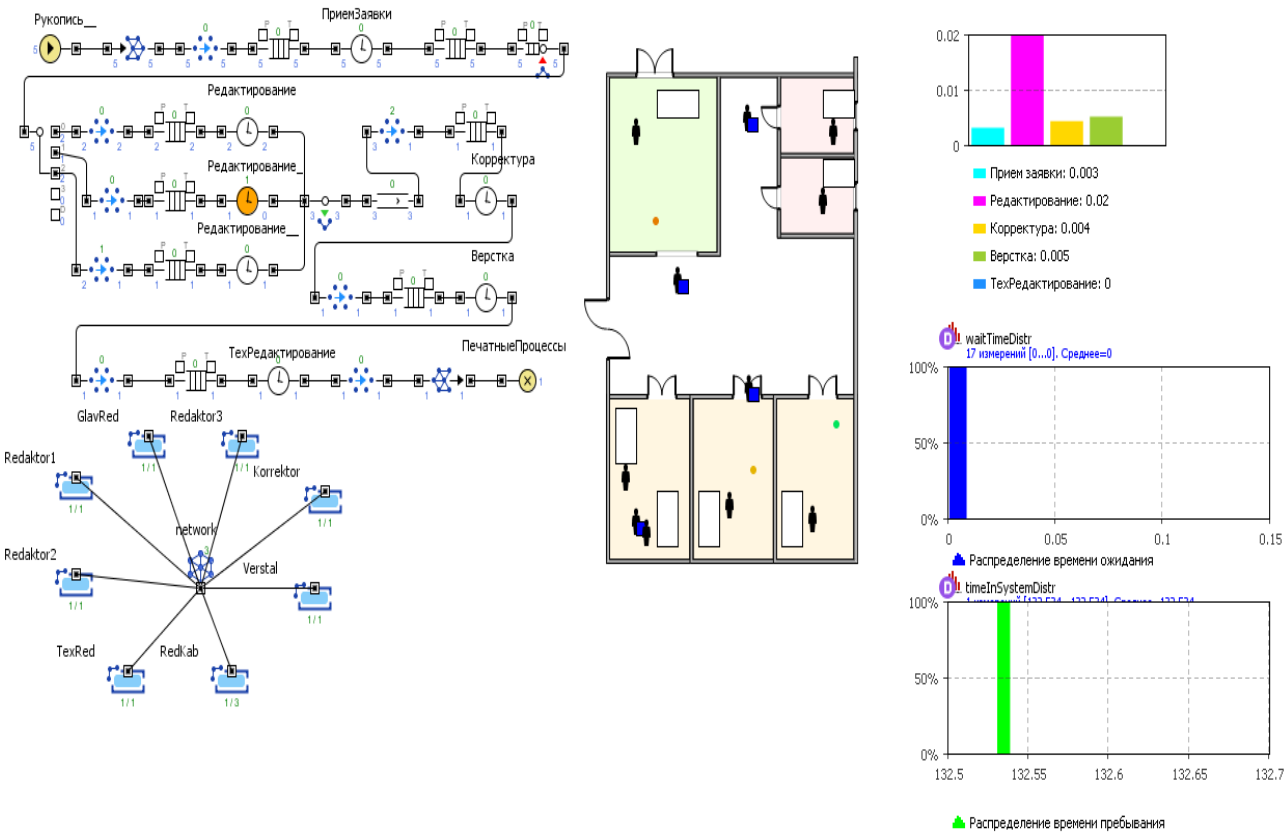


Рис.7.67. Интерфейс модели допечатной подготовки издания

Запустив еще раз модель, можно понаблюдать за тем, какой вид примет распределение времени ожидания и пребывания рукописи в системе.

7.2. Моделирование печатных процессов

Процесс производства полиграфических изделий состоит из пяти этапов [27, 41, 48, 92]. На первом этапе происходит подготовка печатной машины, далее идет печатание контрольных оттисков, печатание тиража, контроль качества и сдача отпечатанных листов в переплетно-брошюровочный цех. IDEF - модель печатного процесса показано на рис. 7.68

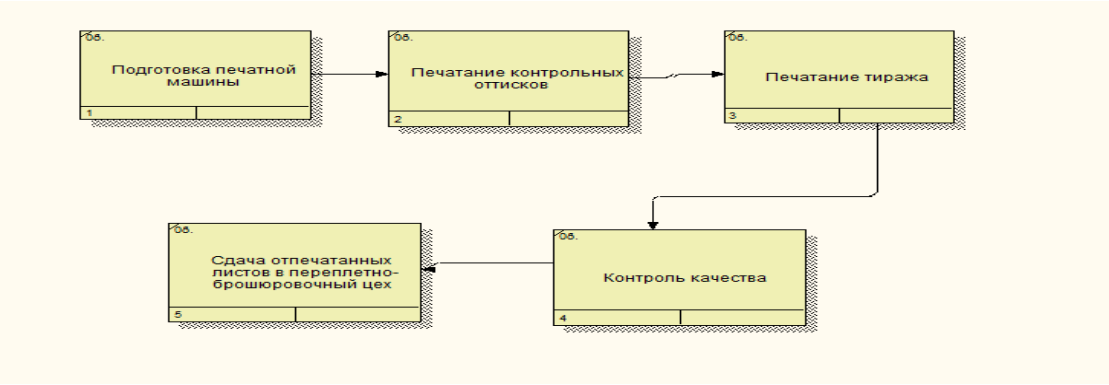


Рис. 7.68. Модель печатного процесса

7.2.1. Анимационная модель печатного процесса

Построим имитационную модель печатного процесса, будем предполагать, что в цехе имеется одна офсетная машина. Для построения такой модели воспользуемся средой AnyLogic.

Шаг 1. Создание нового проекта

1. Щелкнем мышью по панели инструментов "Создать", появится диалоговое окно "Новый Проект".
2. Щелкнем мышью по кнопке "Выбрать"… и выберем директорию, хранящей файлы проекта.
3. Укажем имя нового проекта "Poligraf" в поле редактирования "Имя проекта".
4. В окне новая модель выбираем пункт «использовать шаблон модели».
5. Выбираем дискретно-событийное моделирование.

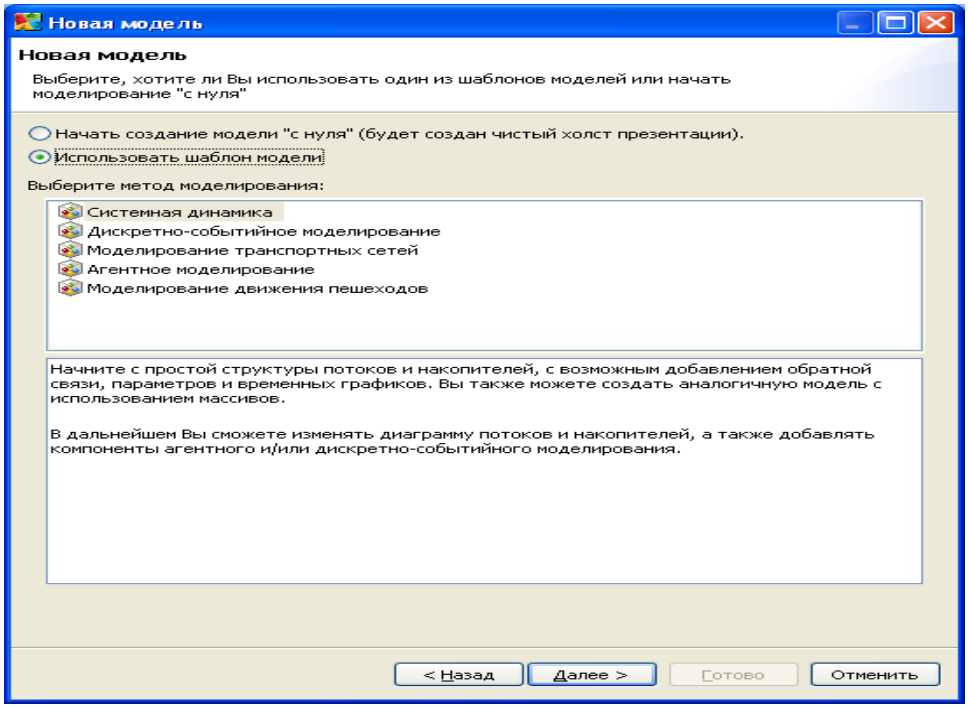


Рис. 7.69. Создание новой модели

6. Подтверждаем операцию, нажав кнопку "Готово". Создали новый проект. В центре появилась структурная диаграмма в центре рабочей области AnyLogic, окно "Проект" – в левой панели, и окно "Свойства" в правой.

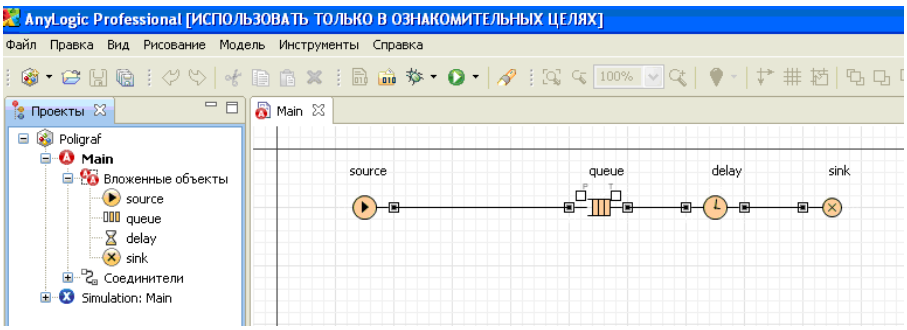


Рис. 7.70. Схема дискретно-событийного моделирования

В данном шаблоне уже имеются основные объекты, которые нам понадобятся в данной работе.

Source – создает заявки, настраиваемые в моменты времени.

Delay – задерживает заявки на заданный период времени, время задержки вычисляется случайно, динамически, так же может зависеть от текущей заявки или каких - либо других условий. Время может вычисляться как длина фигуры, заданной в качестве анимации этого объекта, деленной на "скорость" заявки.

Queue – хранит заявки в определенном порядке. Моделирует очередь заявок, ожидающих приема объектами, следующими за ним в потоковой диаграмме.

Sink – уничтожает поступившую заявку, используется в качестве конечной точки потока заявок.

7. Из основной библиотеки на форму добавим объекты sources, delay, sink, connector, queue и используя эти объекты соберем схему как показано на рисунке 7.

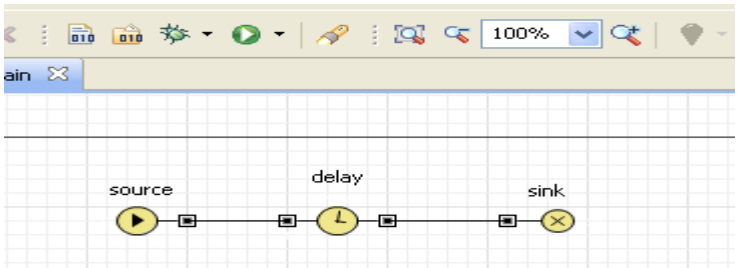


Рис. 7.71. Схема

8. Выделяем схему и копируем его несколько раз

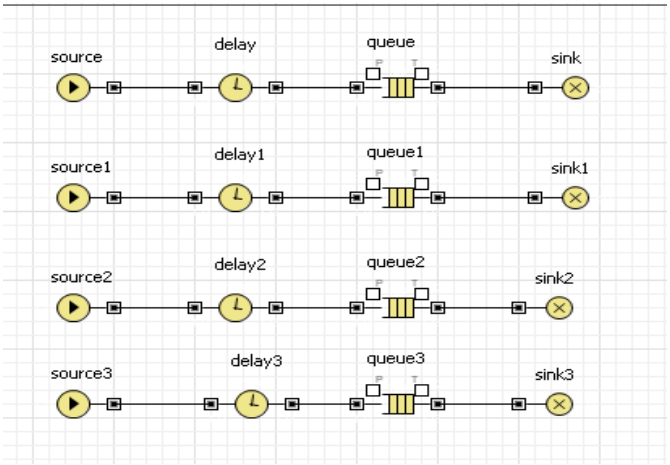


Рис. 7.72. Скопированные схемы

Затем соединяем их, как это указано на рис. 7.73.

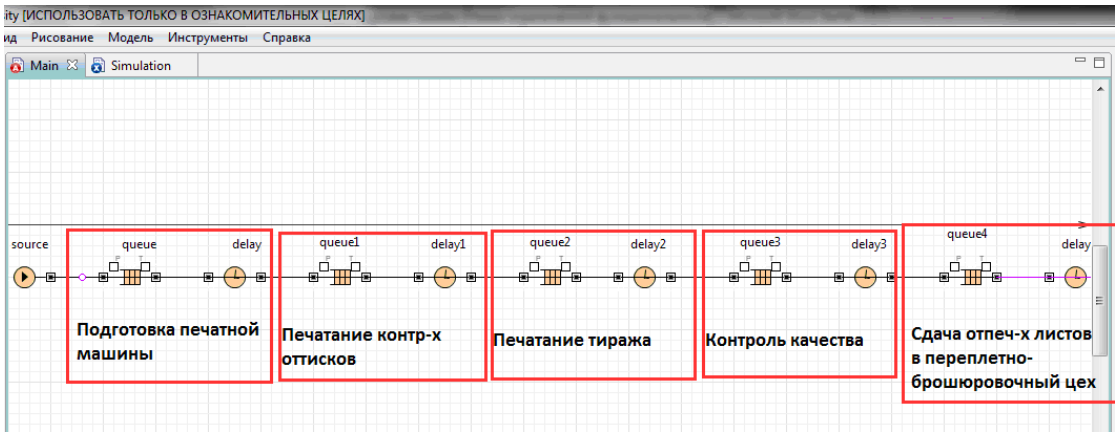


Рис. 7.73.

Шаг 2. Создание анимационной модели

9.Для создания анимационной модели из 3D палитры на форму добавляем объект window3d, 3D окно

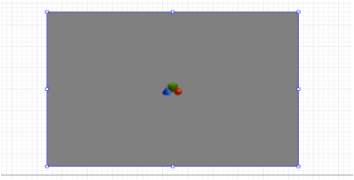


Рис. 7.74. 3D окно

10. Из 3D палитры на форму добавляем объект camera

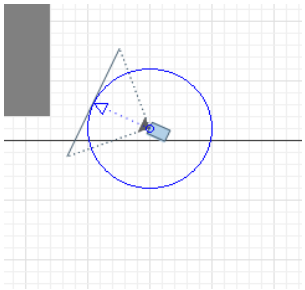


Рис. 7.75. Камера

11. Используя объекты 3D палитры, создаем облик печатной машины

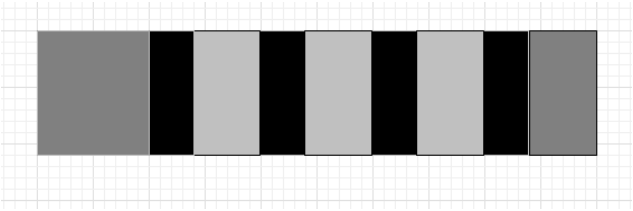


Рис. 7.76. Печатная машина

12. Добавим объект polyline, и размещаем как показано на рис. 7.77

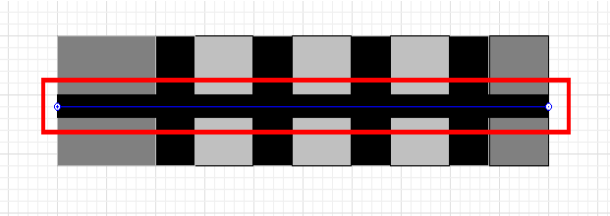


Рис. 7.77. Размещение объекта polyline

13. Добавим объект rectangle, и ставим его фигурой анимации объекта source

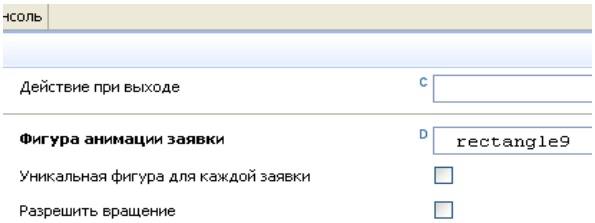


Рис 7.78. Фигура анимации объекта source

14. Добавим объект image, и ставим его фигурой анимации объекта source2

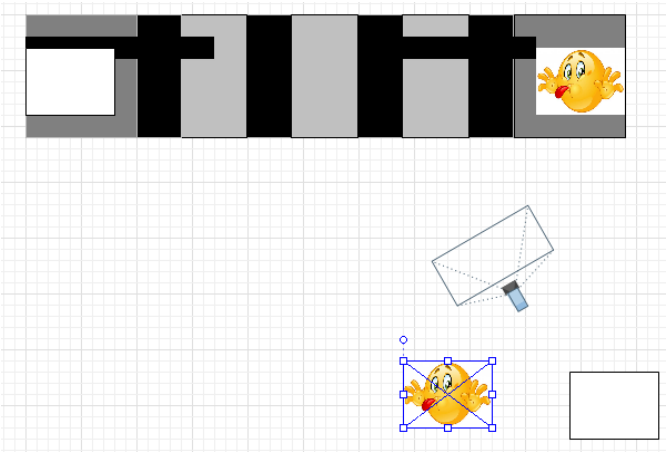


Рис. 7.79. Объект image

15. С помощью 3D палитры создаем стол для контроля качества изделий и сгруппируем их в один объект

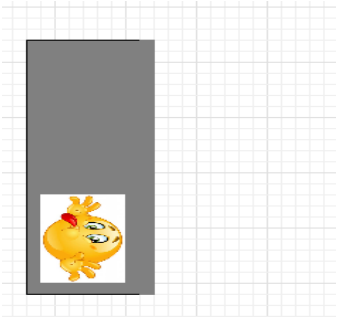


Рис. 7.80. Группировка объектов

16. Создаем модели транспортеров для перемещения листов из одного полиграфического процесса в другую

Рис. 7.81. Транспортеры

17. Добавим объекты worker4

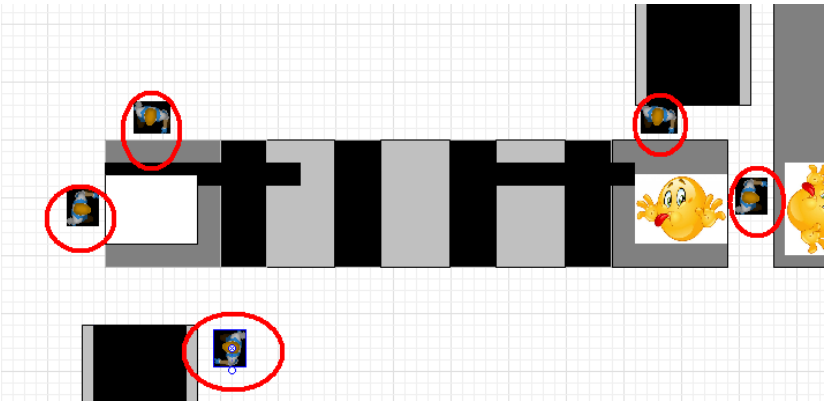


Рис. 7.82. Объекты worker4

18. Добавим объект rectangle20 и сделаем из него пол

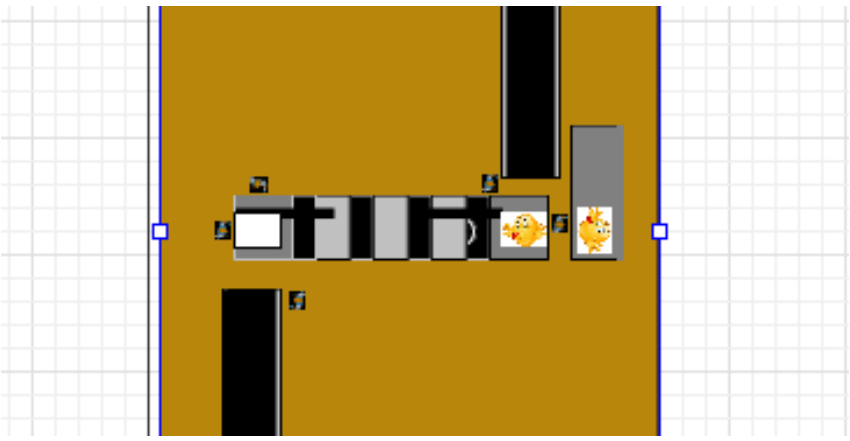


Рис. 7.83. Создание пола

19. Создадим еще несколько объектов, что бы придать модели более реальный вид

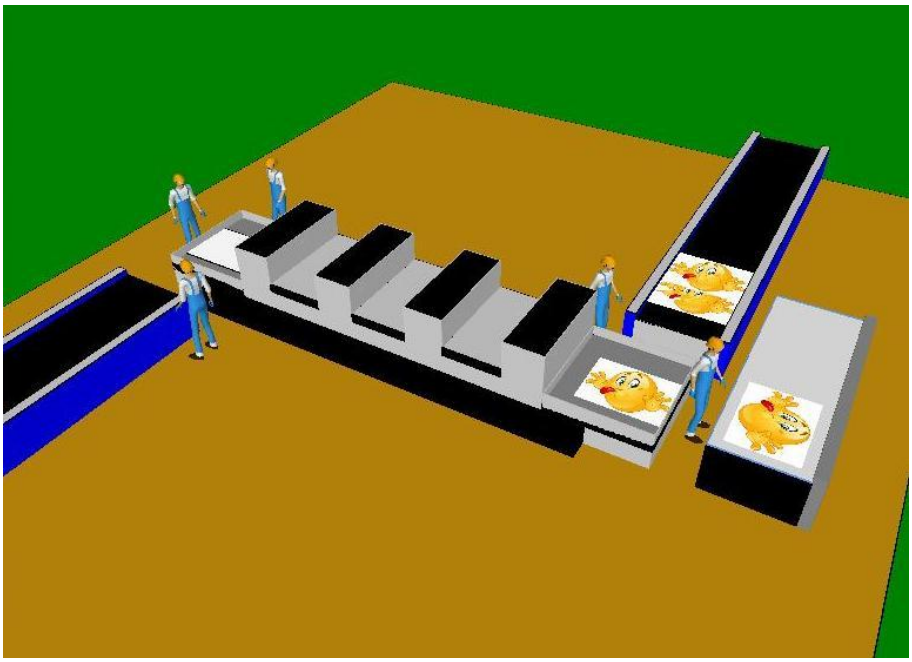


Рис. 7.84. 3D модель печатного процесса

7.2.2. Модель печатного цеха при наличии трех офсетных машин

Построим модель полиграфического производства при наличии в печатном цехе трех машин: однокрасочной, двухкрасочной и четырехкрасочной.

Шаг. 1. Создание проекта печатного цеха при наличии трех офсетных машин.

1. Щелкнем мышью по кнопке панели инструментов Создать появится диалоговое окно Новый Проект.

2. Щелкнем мышью по кнопке Выбрать… и выберите директорию, в которой вы будете хранить файлы проекта.

3. Укажем имя нового проекта Poligraf в поле редактирования Имя проекта.

4. В окне новая модель выбираем пункт «использовать шаблон модели».

5. Выбираем дискретно-событийное моделирование.

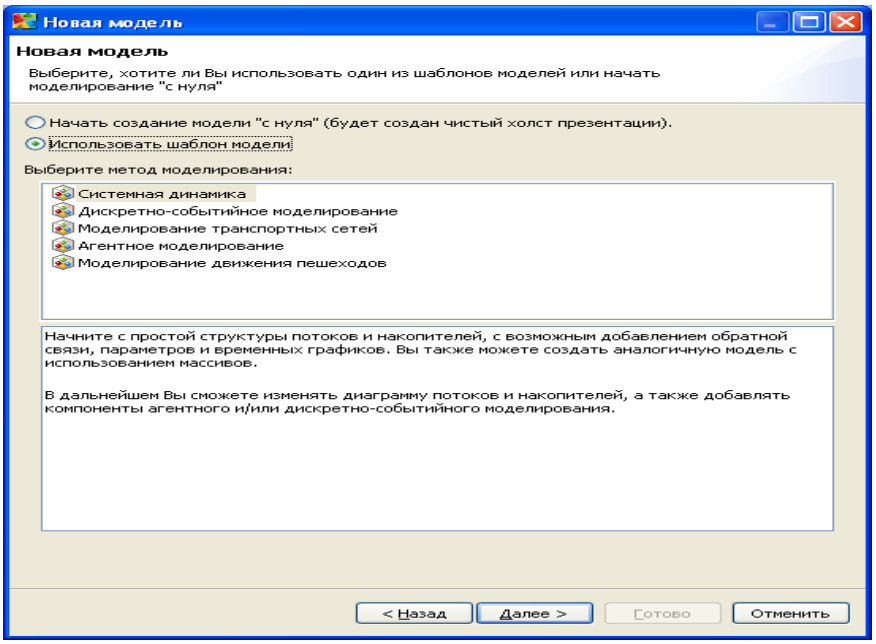


Рис. 7.85. Создание новой модели

6. Подтверждаем операцию, нажав кнопку Готово.

Создали новый проект. В центре появилась структурная диаграмма в центре рабочей области AnyLogic, окно Проект – в левой панели, и окно Свойства в правой.

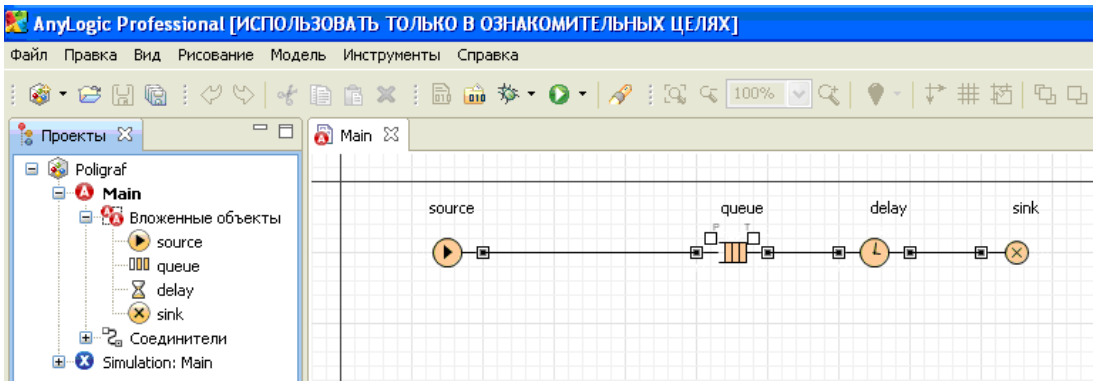


Рис. 7.86. Схема дискретно-событийного моделирования

В данном шаблоне уже имеются основные объекты которые нам понадобятся в данной работе.

Source – создает заявки, в настраиваемые моменты времени.

Delay – задерживает заявки на заданный период времени. Время задержки вычисляется динамически, может быть случайным, зависеть от текущей заявки или от каких-то других условий. Это время может, в частности, вычисляться как длина фигуры, заданной в качестве фигуры анимации этого объекта, поделенной на "скорость" заявки.

Queue – хранит заявки в определенном порядке. Моделирует очередь заявок, ожидающих приема объектами, следующими за ним в потоковой диаграмме.

Sink – уничтожает поступившие заявки. Обычно используется в качестве конечной точки потока заявок.

7. Из основной библиотеки на форму добавим объекты sources, delay, sink, connector, queue и используя эти объекты соберем схему как показано на рисунке 7.

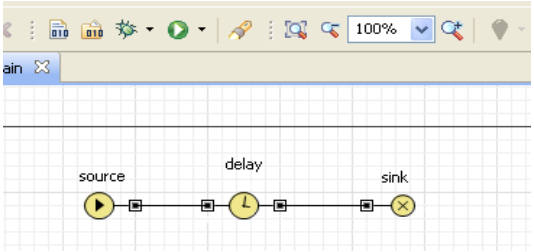


Рис. 7.87. Схема

8. Выделяем схему и копируем его несколько раз

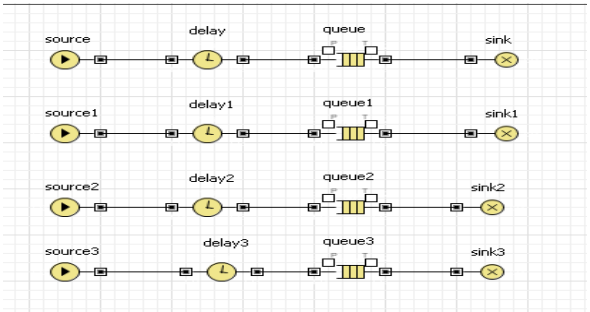


Рис. 7.88. Скопированные схемы

Шаг 2. Создание анимационной модели печатного цеха

9. Для создания анимационной модели из 3D палитры на форму добавляем объект window3d, 3D окно

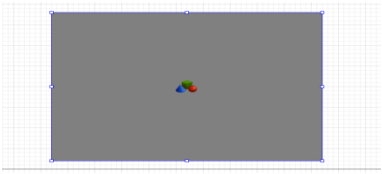


Рис. 7.89. 3D окно

10. Выделяем объект «sourct» кликом левой кнопки мыши, в свойствах объекта ставим интенсивность прибытия 0,2.

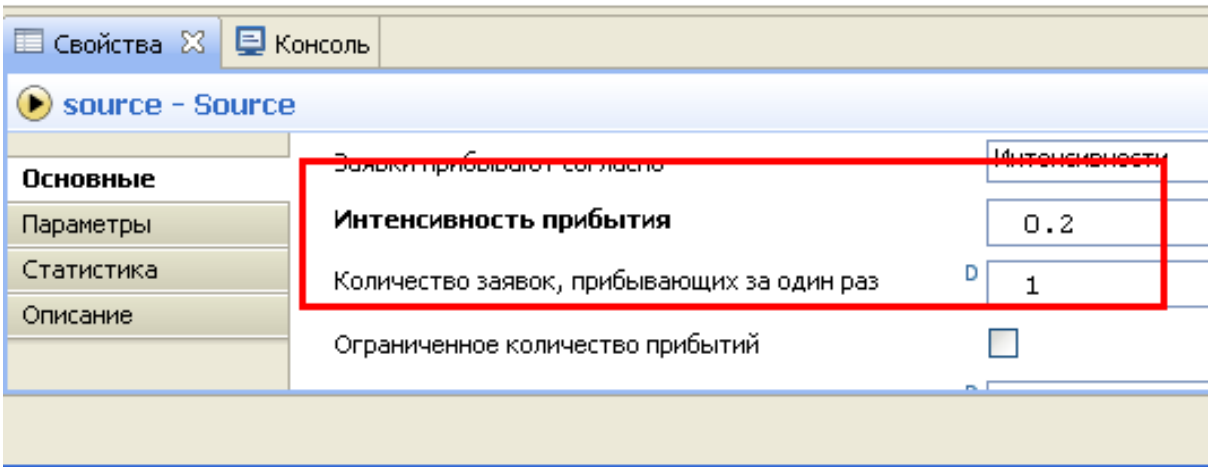


Рис. 7.90. Настройка свойств «sourct» объекта

11. Объекты «queue» оставляем без изменений.

12. Настройка объектов «delay» производиться исходя от того какую роль он выполняет в данной схеме, выделяем левой кнопкой мыши и в свойствах устанавливаем время задержки.

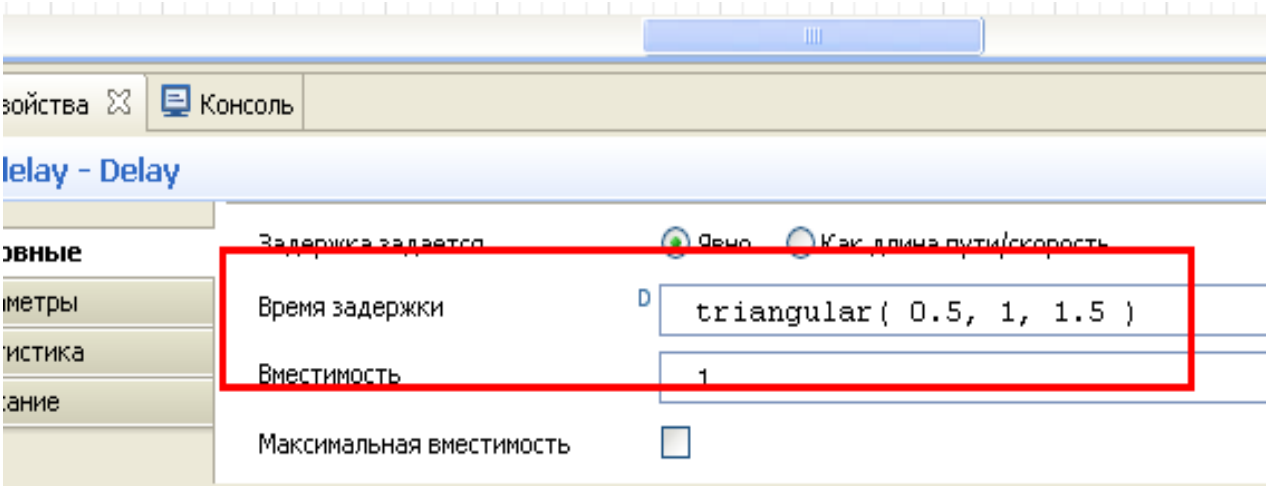


Рис. 7.91. Настройка объектов «delay»

13. Выделяем объект «selectOutput5», в свойствах ставим галочку Использовать: вероятности и для каждого выхода устанавливаем определенное значение, выход «0»-0,2, «1»-0,2, «2»-0,1 и «3»-0,1 на выходе «4»-0

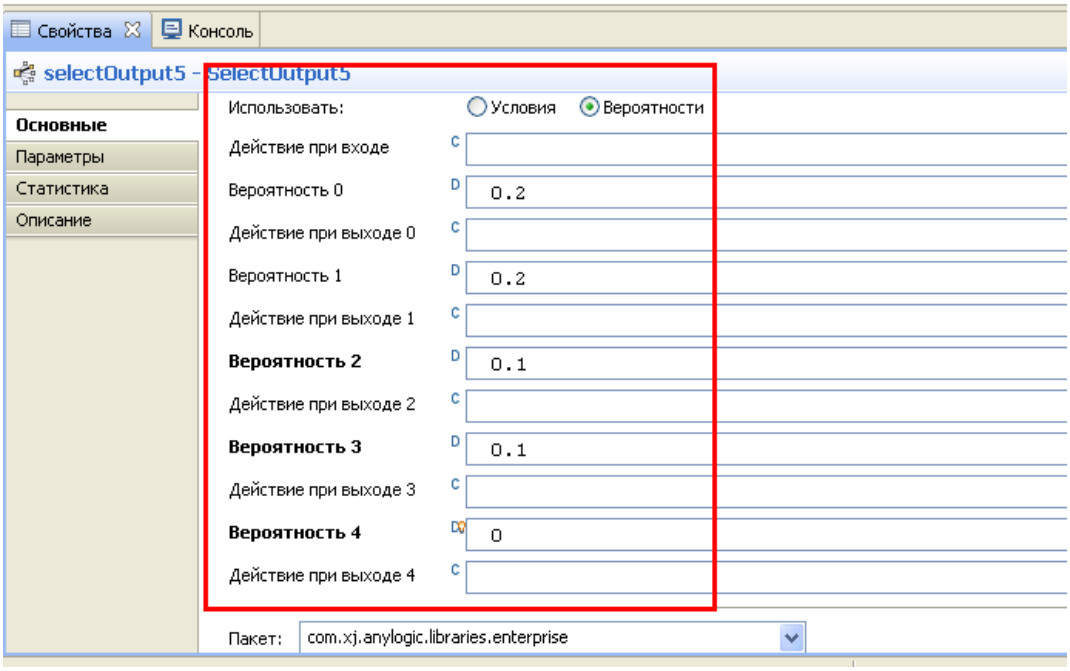


Рис. 7.92. Настройка свойств объекта «selectOutput5»

14. Выделяем объект «conveyor». Устанавливаем расстояние между заявками 1 и скорость 100.

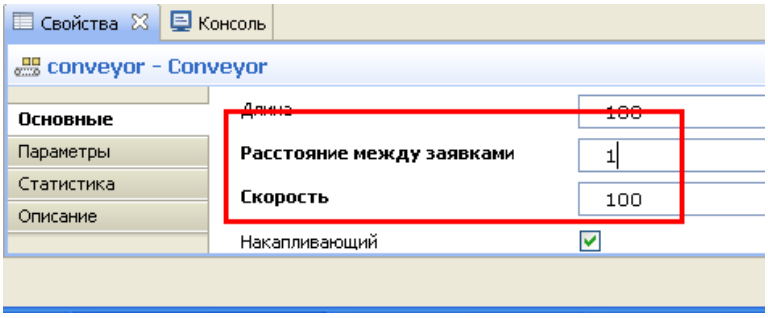


Рис. 7.93. Настройка свойств объекта «conveyor»

15. Проверяем схему, запускаем созданную модель. Вкладка Модель-запустить и выбираем созданную модель.

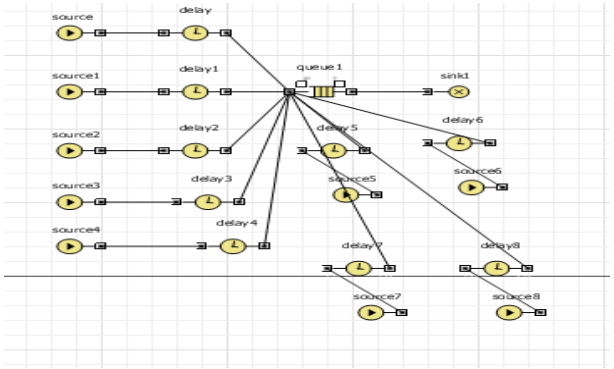


Рис. 7.94. Схема созданной модели

16. Теперь начинаем строить 3D модель данной схемы, для этого из 3D палитры добавляем в созданный класс 3D окшко «window3d».

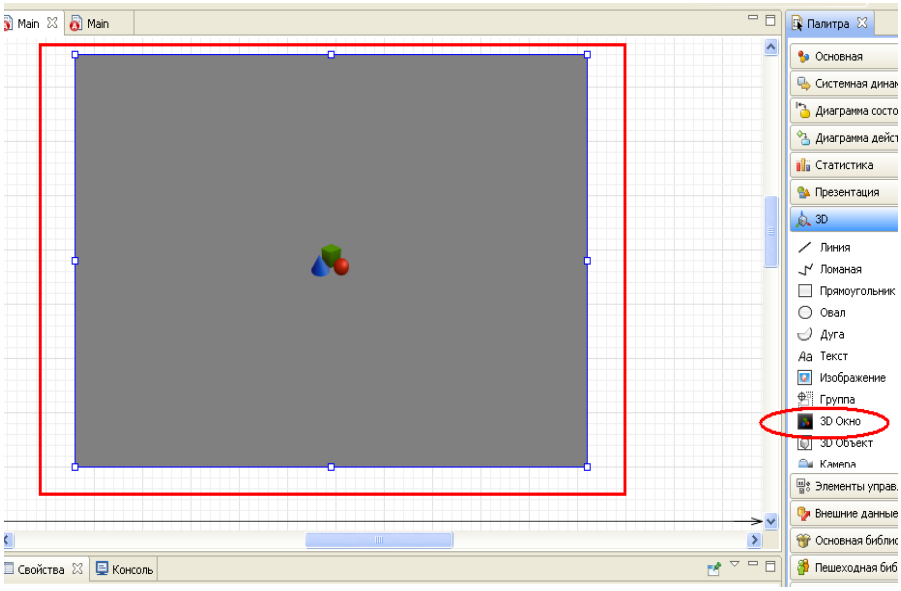


Рис. 7.95. 3D окно

17. Теперь для каждого объекта добавляем 3D рисунок и закрепляем его для конкретного объекта.

18. Перетаскиваем из палитры 3D объекты на форму «container», и закрепляем его с объектом «source», для этого выделяем объект «source» и в свойствах фигура анимации заявки ставим «container», в свойствах «container» устанавливаем масштаб 300 % .

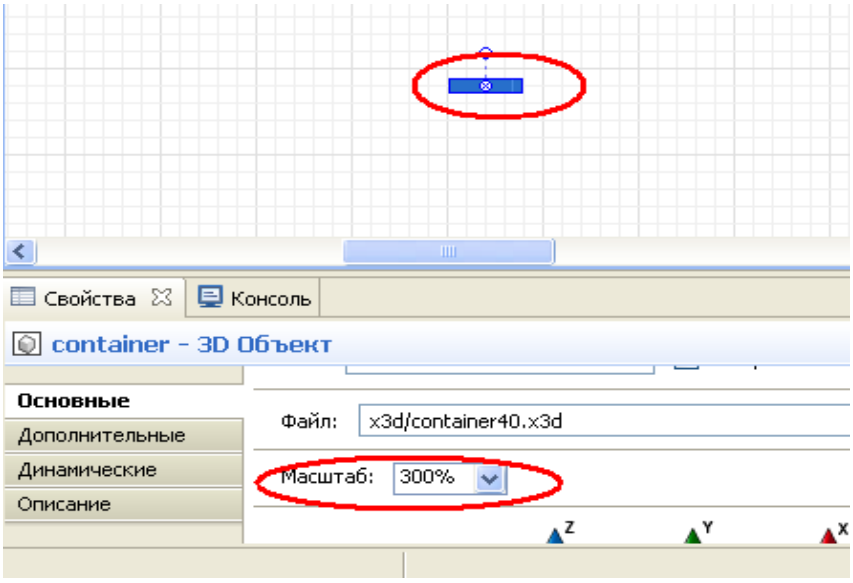


Рис. 7.96. Масштаб объекта «container»

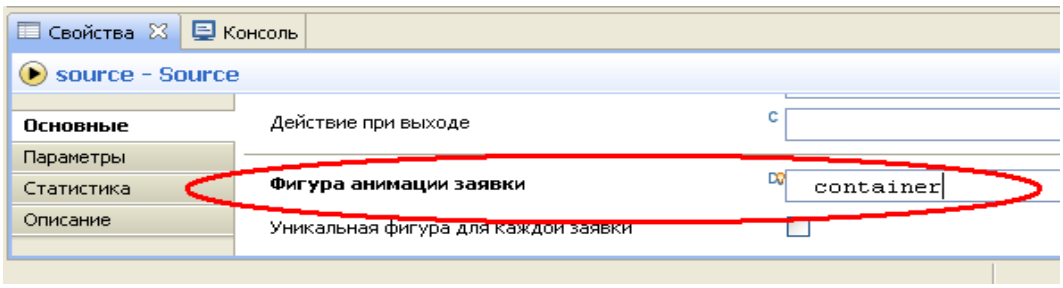


Рис. 7.97. Фигура анимации заявки

19. Добавляем фигуру анимации для «delay», из шаблона имеющегося на программном продукте копируем элемент «polylineConveyorIn» и вставляем его в созданный проект. В свойствах «delay» фигуру анимации заявки ставим «polylineConveyorIn», тип анимации путь.

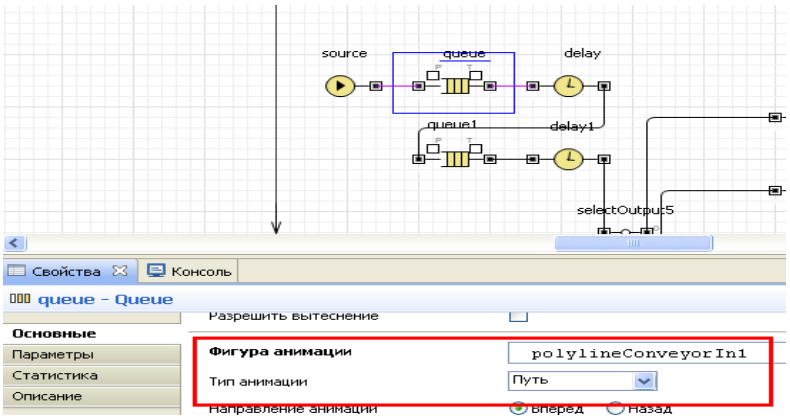


Рис. 7.98. Фигура анимации

20. Таким же способом прикрепляем фигуры анимации для остальных объектов «delay», «delay1», «delay6» и «delay7».

21. Для объектов «delay2», «delay3», «delay4» и «delay5» прикрепляем фигуру «polylineConveyorIn5», «polylineConveyorIn6», «polylineConveyorIn7» и «polylineConveyorIn8» соответственно, тип анимации ставим путь.

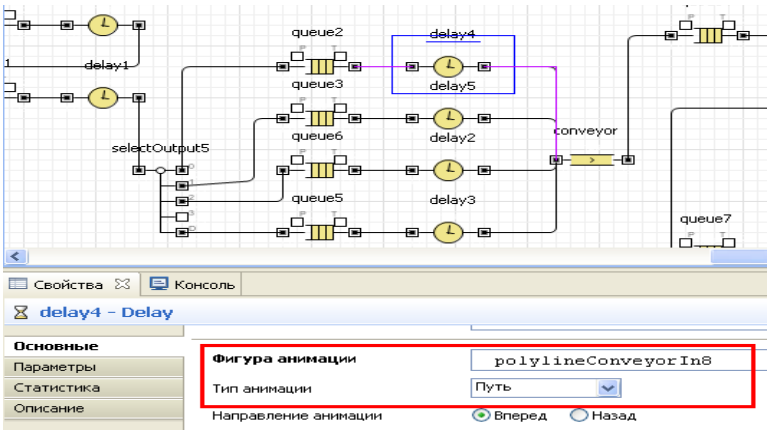


Рис. 7.99. Фигура анимации

22. Копируем объект «polylineConveyorIn» и скопированный объект устанавливаем фигурой анимации для «conveyor».

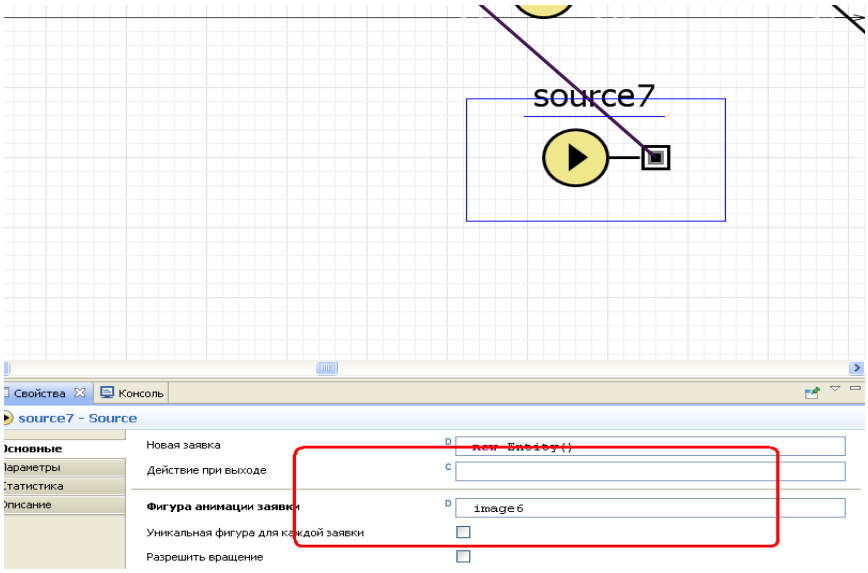


Рис. 7.100. Фигура анимации

23. Из 3D палитры добавляем «camera», в свойствах устанавливаем поворот по X=20 градусов, поворот по У=45 градусов и устанавливаем на против 3D объектов.

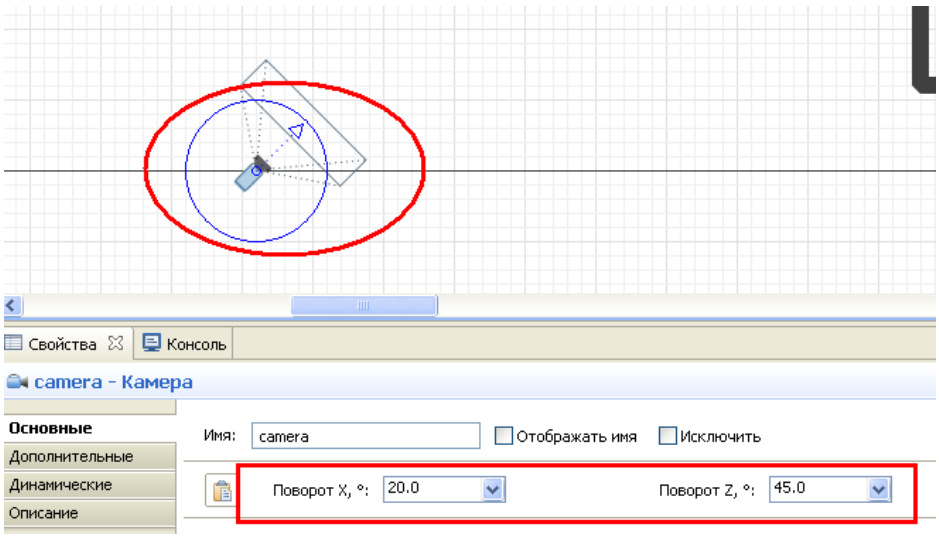


Рис. 7.101. Добавление камеры

24. Выделяем объект «window3d» в свойствах камера из выпадающего списка выбираем «camera».

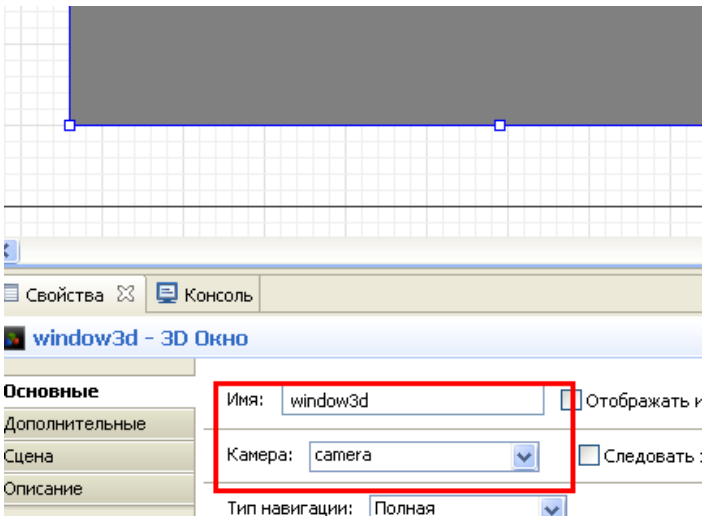


Рис. 7.102. Свойства окна «window3d»

25. Запускаем модель и смотрим, что получилось в 3D окошке, для этого после запуска модели в пункте показать область из выпадающего списка выбираем «window3d»

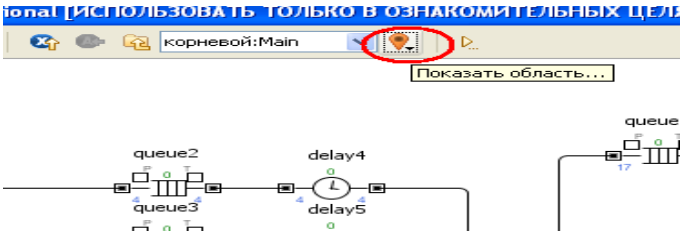


Рис. 7.103. Кнопка выбора окна

26. Из 3D палитры на форму добавляем объект camera

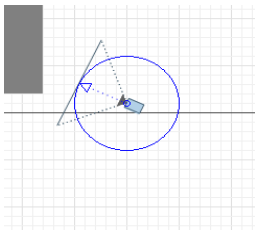


Рис. 7.104. Камера

27. Используя объекты 3D палитры, создаем облик печатной машины

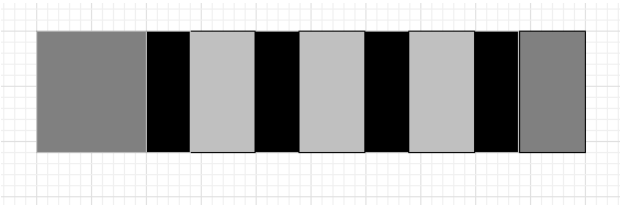


Рис. 7.105. Печатная машина

28. Добавим объект polyline, и размещаем как показано на рисунке

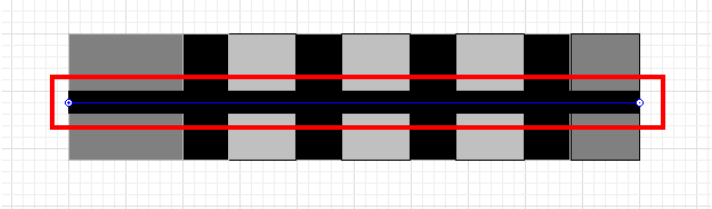


Рис. 7.106. Размещение объекта polyline

29. Добавим объект rectangle, и ставим его фигурой анимации объекта source

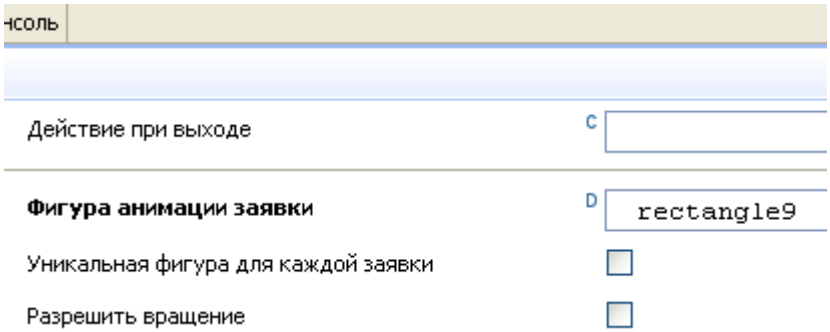


Рис. 7.107. Фигура анимации объекта source

30. Добавим объект image, и ставим его фигурой анимации объекта source2

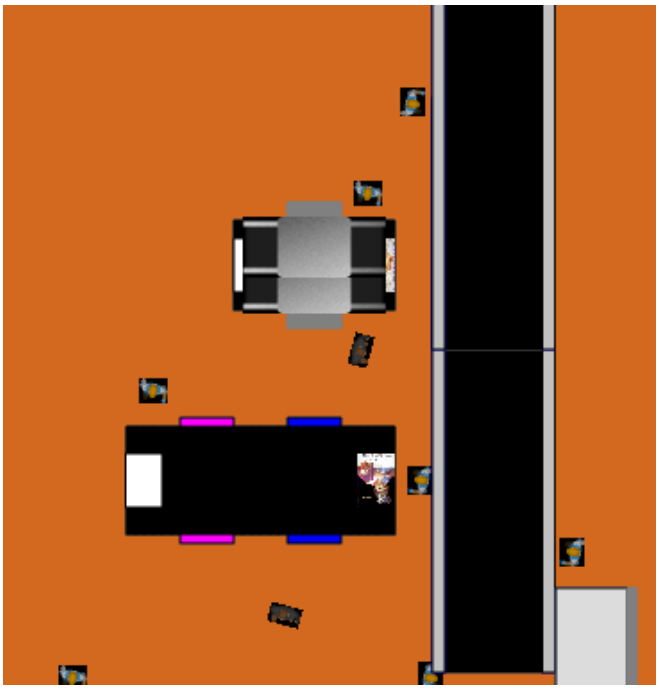


Рис. 7.108. Объект image

31. С помощью 3D палитры создаем стол для контроля качества изделий и сгруппируем их в один объект

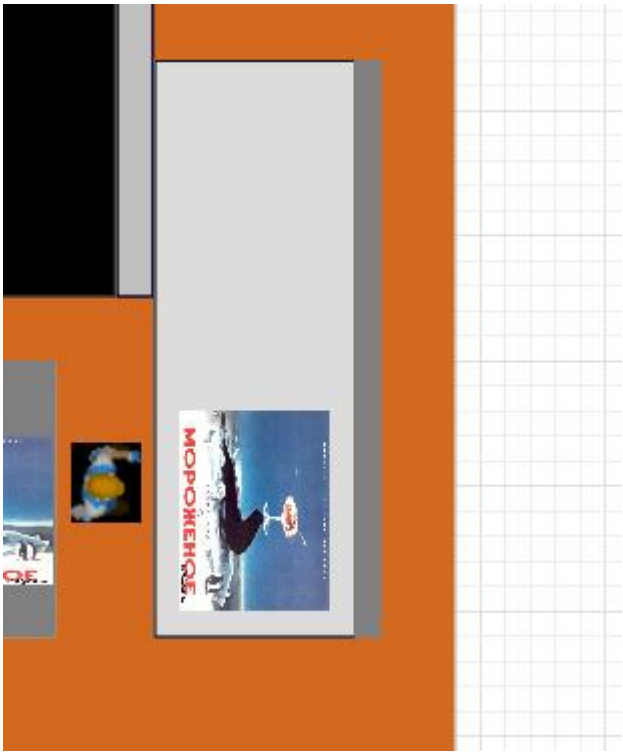


Рис. 7.109. Группировка объектов

32. Создаем модели транспортеров для перемещения листов из одного полиграфического процесса в другую

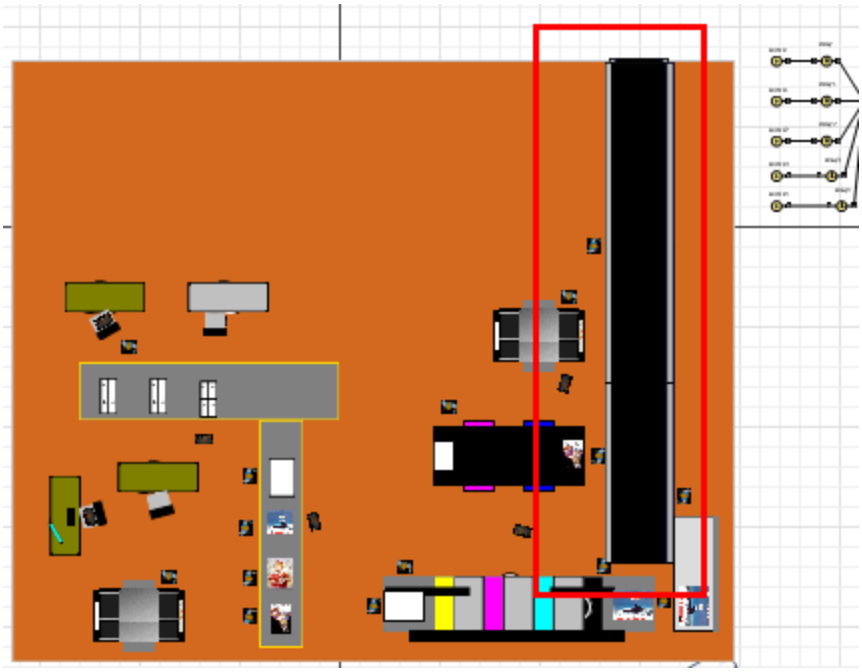


Рис. 7.110. Транспортеры

33. Добавим объекты worker4

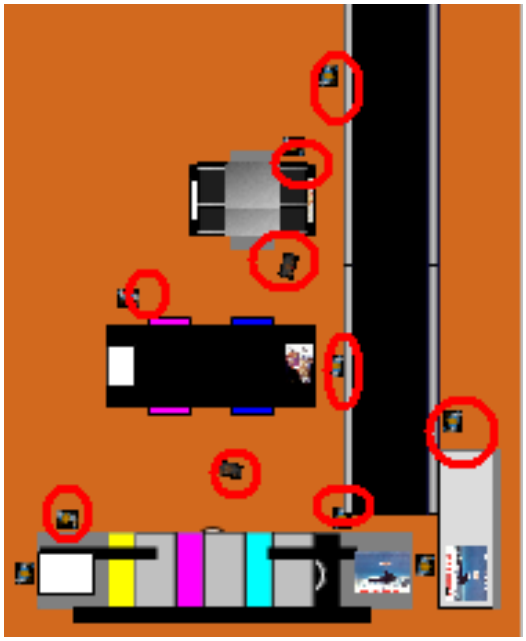


Рис. 7.111. Объекты worker4

34. Добавим объект rectangle20 и сделаем из него пол

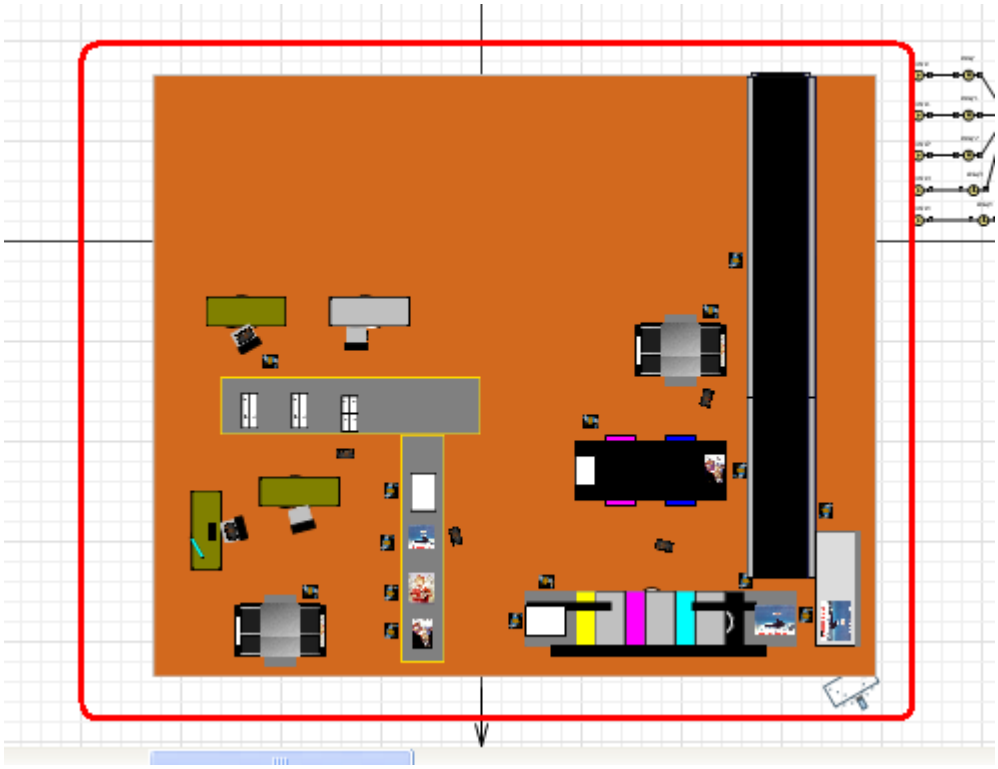


Рис. 7.112. Создание пола

35. Создадим еще несколько объектов, что бы придать модели более реальный вид

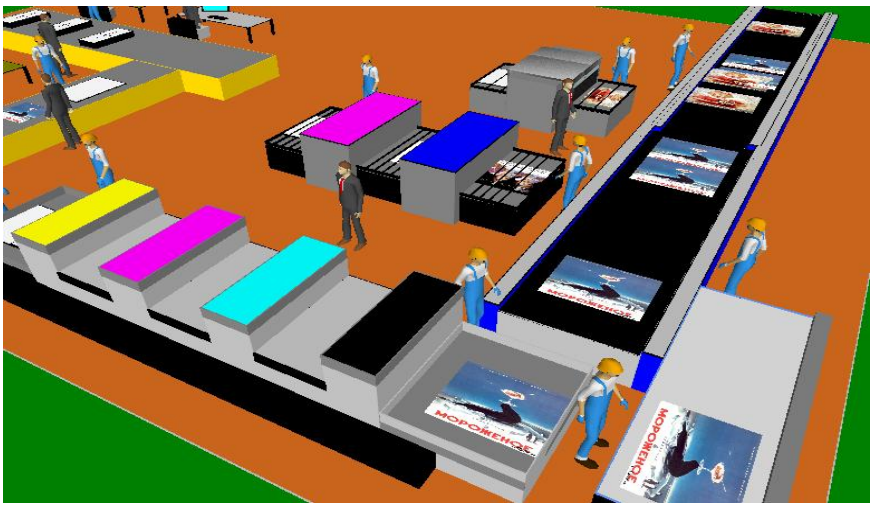


Рис. 7.113. 3D модель печатного процесса

7.3. Моделирование послепечатных процессов

Существует два типа книжных переплетов. К первому методу относится производство книг в твердом переплете, которая имеет высокую технологичность, функциональность и износоустойчивость.

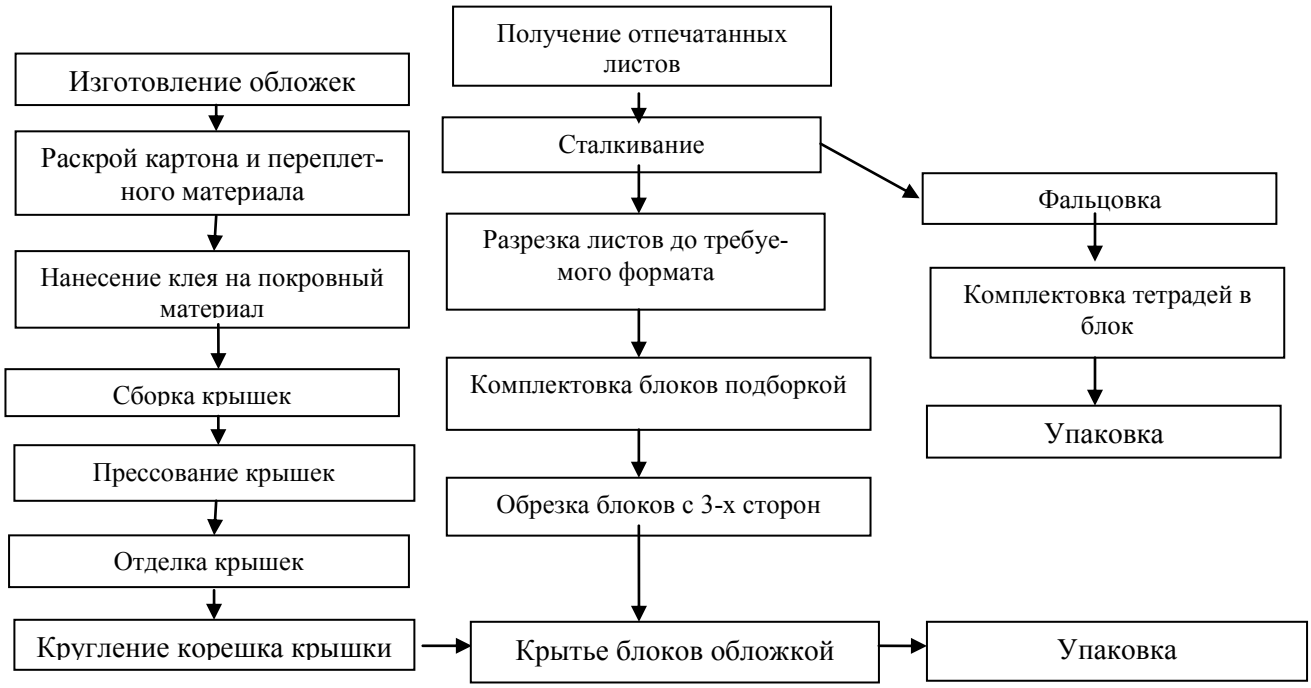


Рис.7.114. Схема технологии изготовления книги в твердом переплете

На рис.7.114 представлена схема технологии изготовления книги в твердом переплете. Важной составляющей при создании твердой брошюровки является крышка. Одним из самых практичных и недорогих материалов является цельнокрытая переплетная крышка. Она легка и дешева при изготовлении, но сохраняет свою прочность в течение долгого периода времени.

Изготовление твердого переплета подразумевает использование материалов на бумажной основе, таких как бумвинил или ни тропалиадные покрытия.

Ко второму методу производства книг относятся книги в мягкой обложке. Схема изготовления представлена на рис. 7.115.

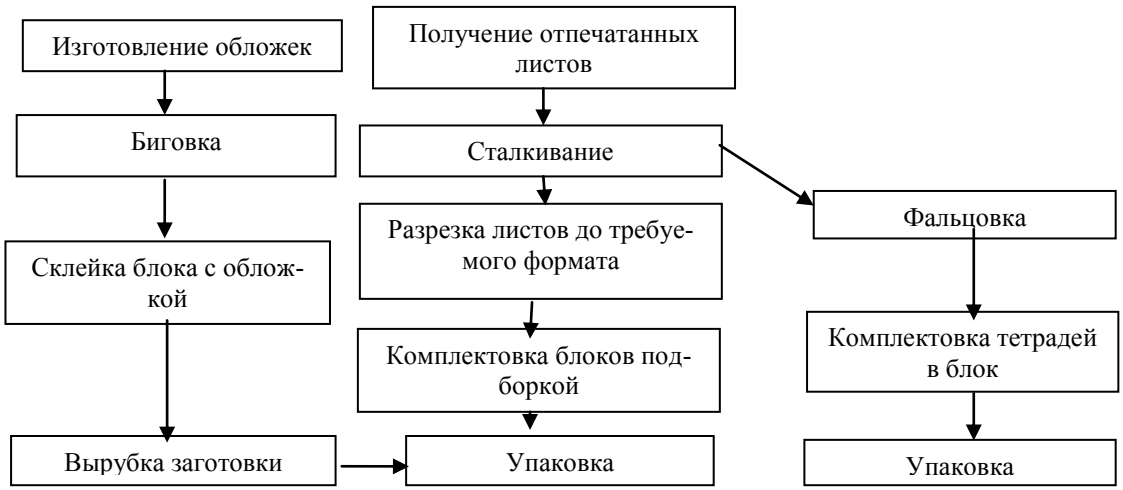


Рис. 7.115. Схема технологии изготовления книги в мягком переплете

На рис. 7.116 показана упрощенная IDEF модель послепечатного процесса. В модели показан процесс производства полиграфических изделий послепечатного процесса, который состоит из пяти этапов.

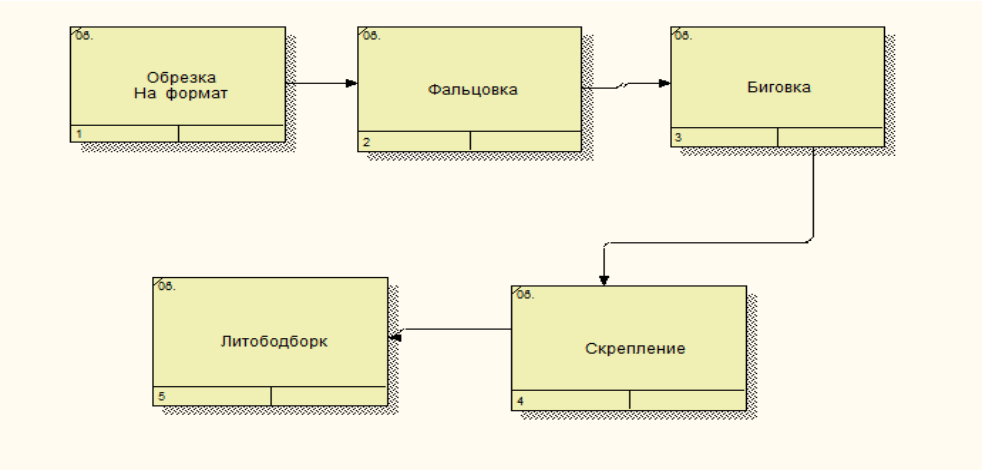


Рис. 7.116. IDEF модель послепечатного процесса

На первом этапе происходит обрезка листа на формат, после которого изделия проходят этапы фальцовки и биговки изделия. На третьем этапе выполняется листоподборка, после которого идет крепление листов различными способами.

Рассмотрим различные модели полиграфического производства, построенные с помощью среды имитационного моделирования AnyLogic. Модели построены на основе шаблона Дискретно-событийного моделирования. При создании модели использовалась основная библиотека.

7.3.1. Первая имитационная модель послепечатного цеха

Построим анимационную модель работы послепечатного цеха.

Начинаем с создания нового проекта.

1. Щелкнем мышью по кнопке панели инструментов Создать появится диалоговое окно Новый Проект.

2. Щелкнем мышью по кнопке Выбрать… и выберите директорию, в которой вы будете хранить файлы проекта.

3. Укажем имя нового проекта Poligraf в поле редактирования Имя проекта.

4. В окне новая модель выбираем пункт «использовать шаблон модели».

5. Выбираем дискретно-событийное моделирование.

Подтверждаем операцию, нажав кнопку Готово. Создали новый проект. В центре появилась структурная диаграмма в центре рабочей области AnyLogic, окно Проект – в левой панели, и окно Свойства в правой.

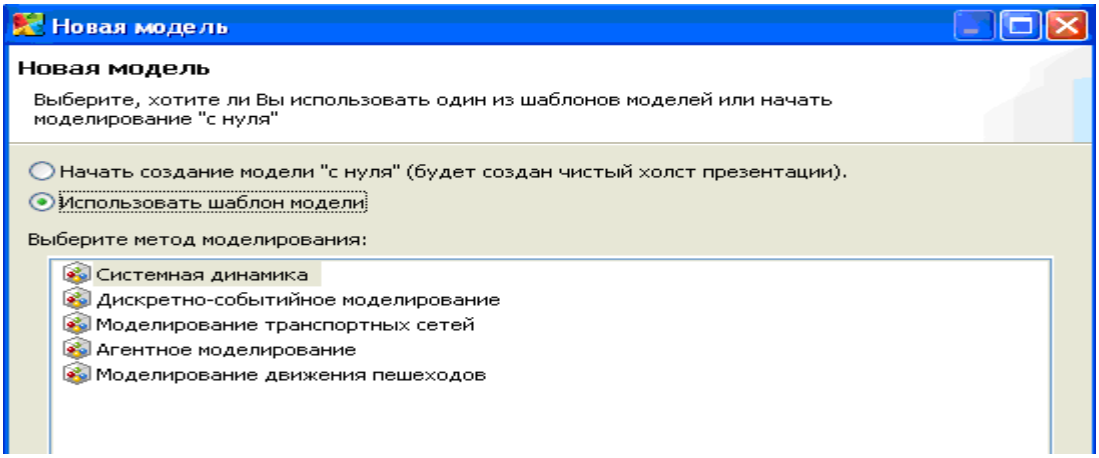


Рис. 7.117.Создание новой модели

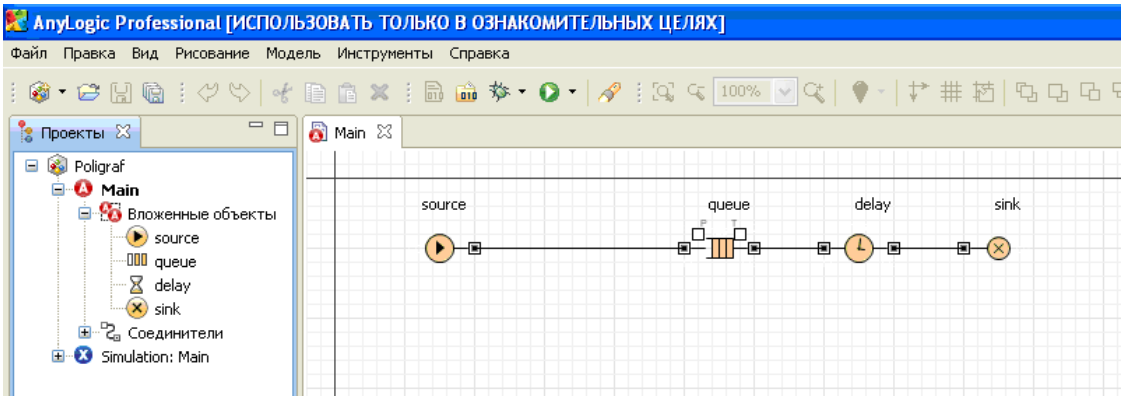


Рис. 7.118. Схема дискретно-событийного моделирования

В данном шаблоне уже имеются основные объекты, которые мы использовали в данной работе: Source, Delay, Queue, Sink.

6. Из основной библиотеки на форму добавим объект «sources», в свойствах объекта настройку производим согласно рисунку 4.

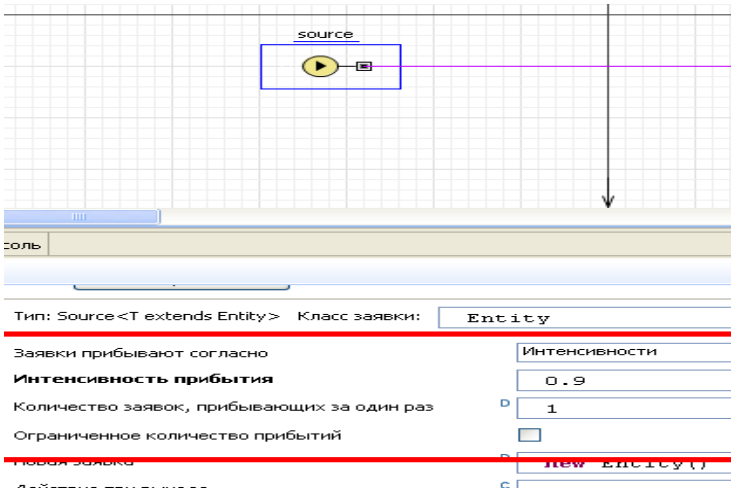


Рис. 7.119. Настройка объекта «sources»

7. Из основной библиотеки добавляем на форму объекты «queue» и «delay». В свойствах «delay» настраиваем значения по рисунку 7.

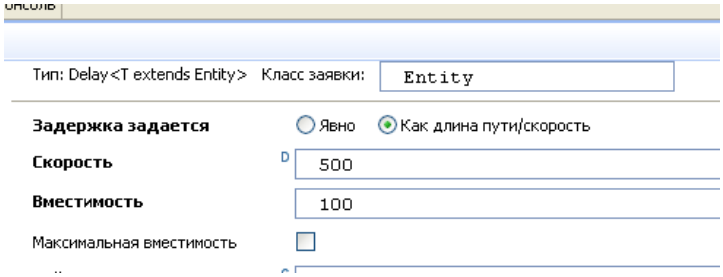


Рис. 7.120. Свойста объекта «delay»

8. Копируем «queue», «delay» четыре раза и собираем из них схему.

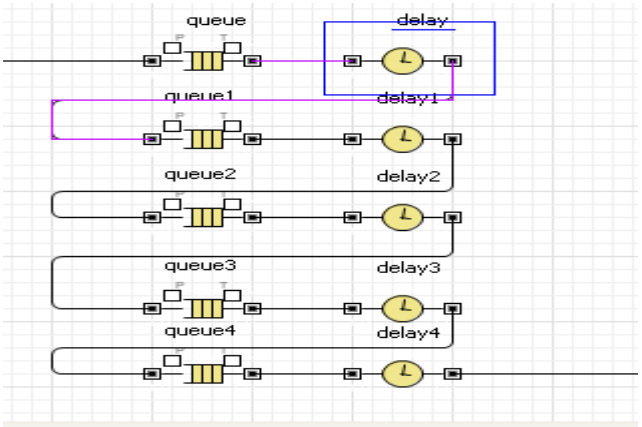


Рис. 7.121. Схема

9. Из основной библиотеки на форму добавим объект «sink».

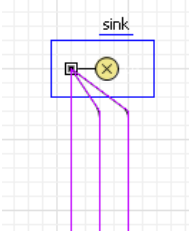


Рис. 7.122. Объект «sink».

10. Также из основной библиотеки добавим объект «source1».

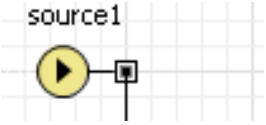


Рис. 7.123. Объект «source1»

11. Добавим объекты «queue5», «delay5» и настраиваем их свойства

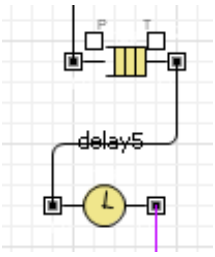


Рис. 7.124. Объекты «queue5», «delay5»

12. Настраиваем свойства объекта «queue5», «delay5».

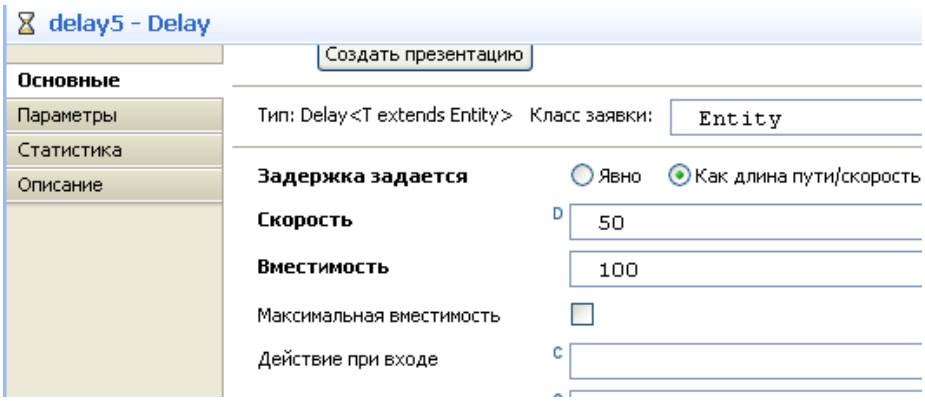


Рис. 7.125. Настройка объектов «queue5», «delay5»

13. Добавив из основной библиотеки Объекты «source2», «queue6», «delay6».

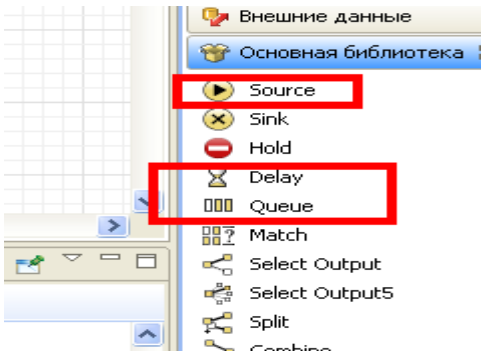


Рис. 7.126. Объекты «source2», «queue6», «delay6»

14. Настройка объектов «queue6», «delay6».

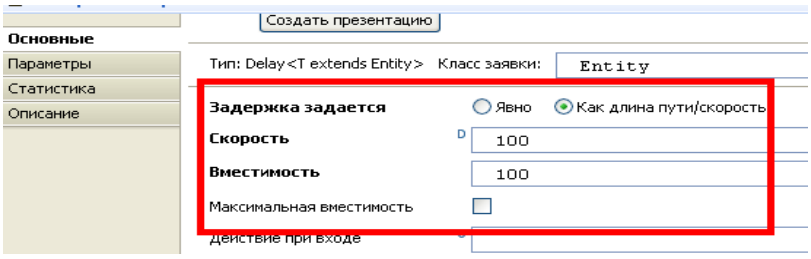


Рис. 7.127. Настройка объектов «queue6», «delay6»

15. Из 3D библиотеки в форму добавляем объект «window3d».

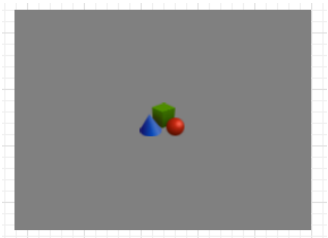


Рис. 7.128. Объект «window3d»

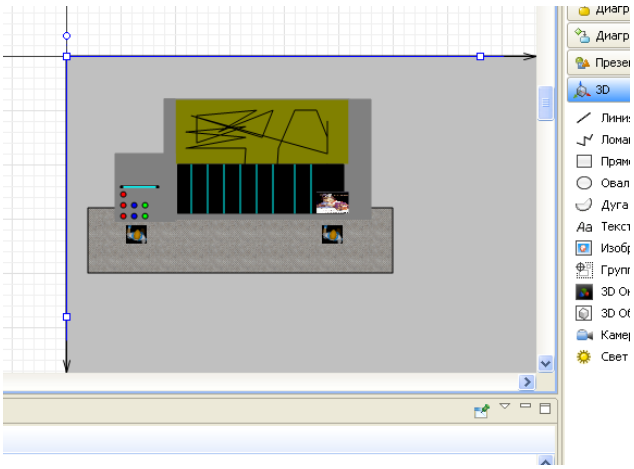


Рис. 7.129. 3D модель резальной машины

16. Из палитра 3D объекты и 3D добавляем на форму объекты, после чего располагаем их как на рис. 7.130 и сгруппируем.

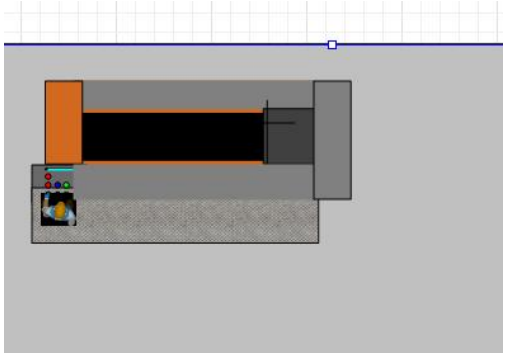


Рис. 7.130. 3D модель фальцовочной машины

17. Из палитра 3D объекты и 3D добавляем на форму объекты, после чего располагаем их как на рис. 7.131 и сгруппируем.

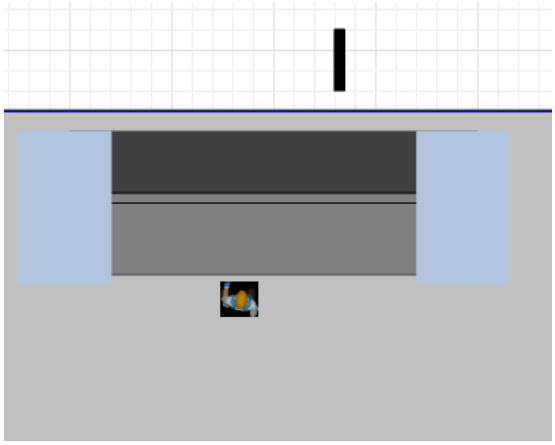


Рис. 7.131. 3D модель биговочной машины

18. Из палитра 3D объекты и 3D добавляем на форму объекты, после чего располагаем их как на рис.7.132 и сгруппируем.

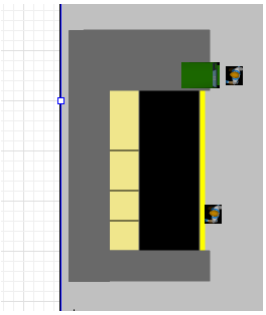


Рис. 7.132. 3D модель листоподборной машины

19. Из палитра 3D объекты и 3D добавляем на форму объекты, после чего располагаем их как на рис. 7.133 и сгруппируем.

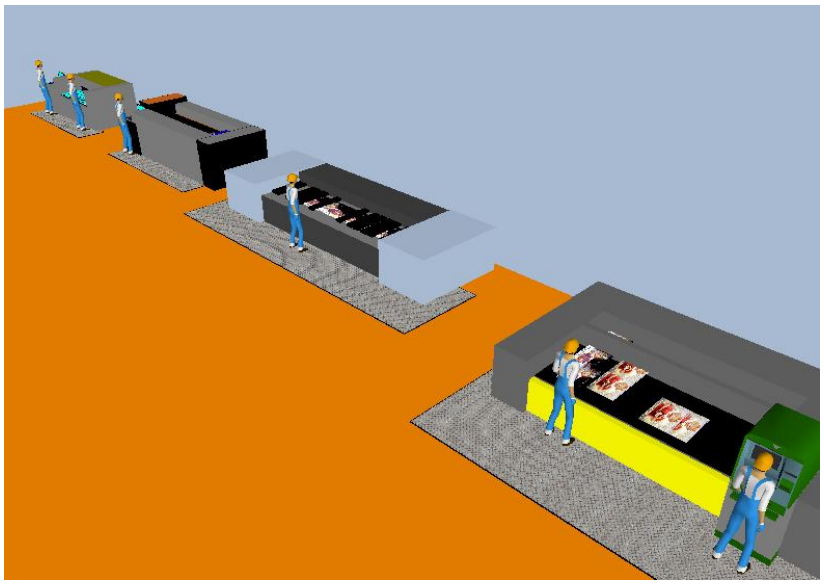


Рис. 7.133. 3D модель скрепляющей машины

7.3.2. Вторая имитационная модель работы послепечатного цеха

Рассмотрим теперь другой проект для послепечатного цеха.

1. Щелкнем мышью по кнопке панели инструментов Создать, появится диалоговое окно Новый Проект.
2. Щелкнем мышью по кнопке Выбрать… и выберите директорию, в которой вы будете хранить файлы проекта.
3. Укажем имя нового проекта Poligraf в поле редактирования Имя проекта.
4. В окне новая модель выбираем пункт «использовать шаблон модели».
5. Выбираем дискретно-событийное моделирование.

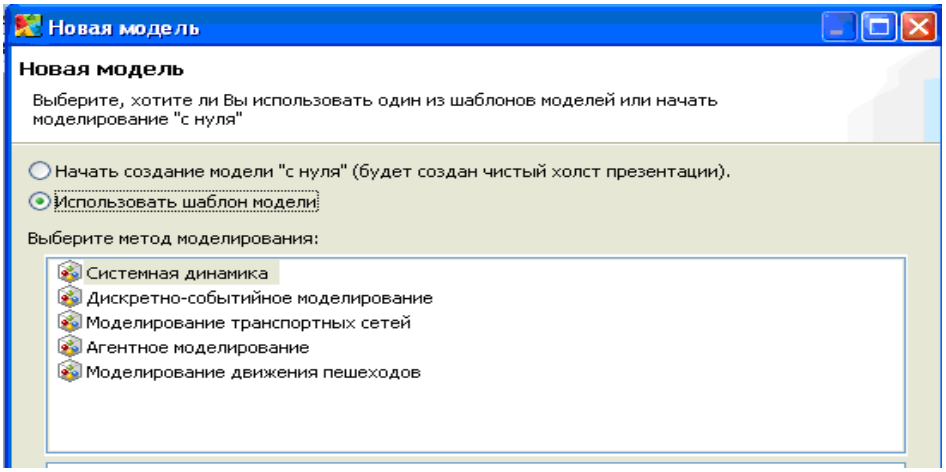


Рис. 7.134. Создание новой модели

6. Подтверждаем операцию, нажав кнопку Готово.

Создали новый проект. В центре появилась структурная диаграмма в центре рабочей области AnyLogic, окно Проект – в левой панели, и окно Свойства в правой.

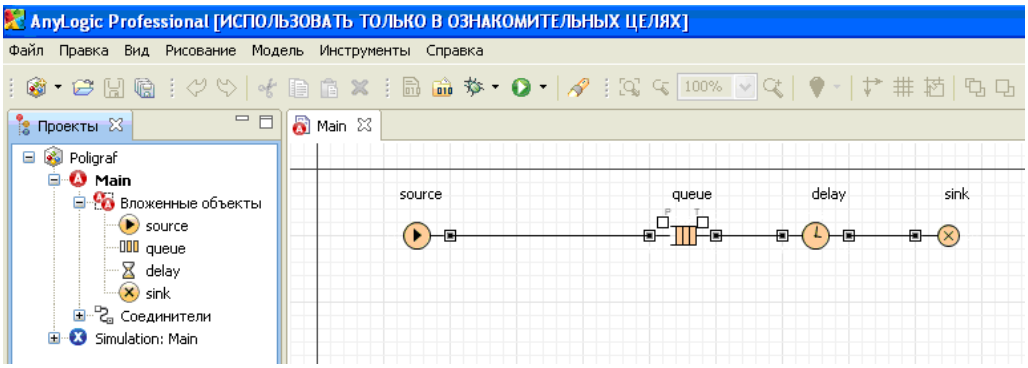


Рис. 7.135. Схема дискретно-событийного моделирования

В данном шаблоне уже имеются основные объекты, которые мы использовали в данной работе: Source, Delay, Queue, Sink.

7. Из основной библиотеки на форму добавим объект «sources», в свойствах объекта настройку производим согласно рис. 7.136.

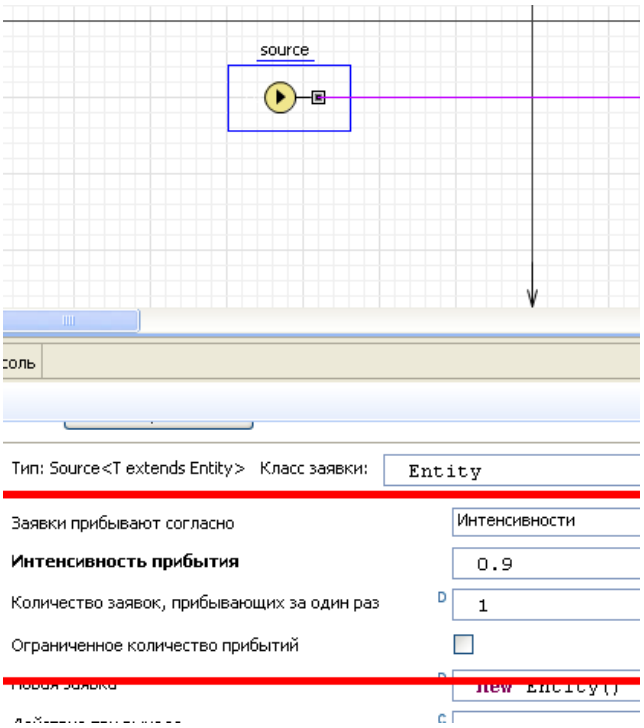


Рис. 7.136. Настройка объекта «sources»

8. Из основной библиотеки добавляем на форму объекты «queue» и «delay». В свойствах «delay» настраиваем значения по рис. 7.137.

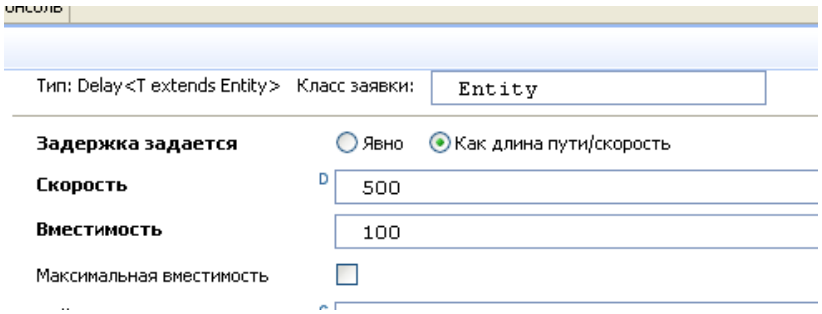


Рис. 7.137. Свойста объекта «delay»

9. Копируем «queue», «delay» четыре раза и собираем из них схему.

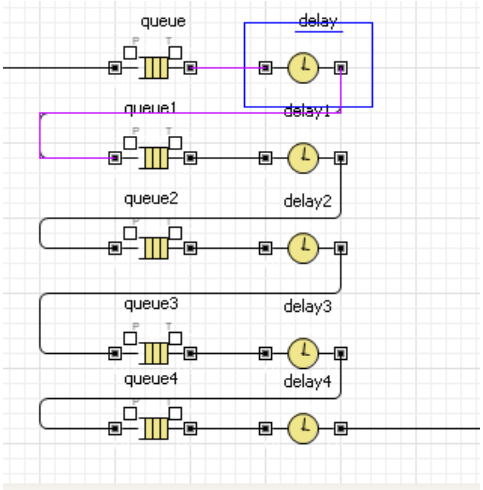


Рис. 7.138. Схема

10. Из основной библиотеки на форму добавим объект «sink».

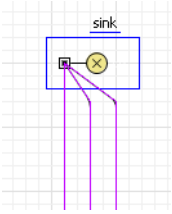


Рис. 7.139. Объект «sink».

11. Также из основной библиотеки добавим объект «source1».

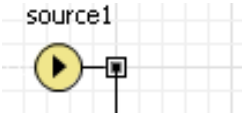


Рис. 7.140. Объект «source1»

12. Добавим объекты «queue5», «delay5» и настраиваем их свойства.

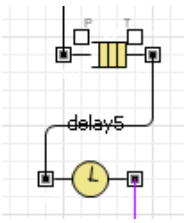


Рис. 7.141. Объекты «queue5», «delay5»

13. Настраиваем свойства объекта «queue5», «delay5».

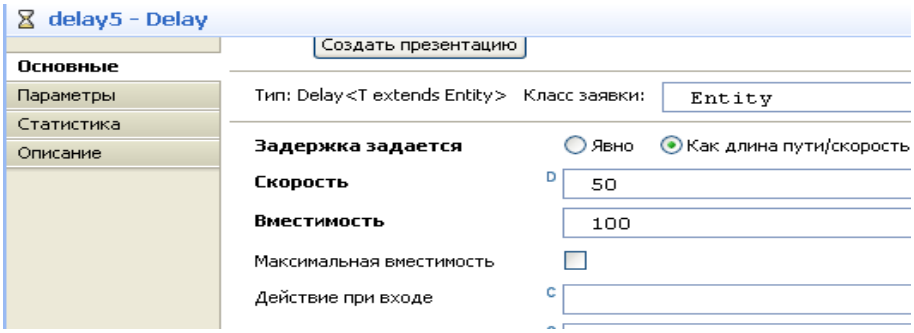


Рис. 7.142. Настройка объектов «queue5», «delay5»

14. Добавив из основной библиотеки Объекты «source2», «queue6», «delay6».

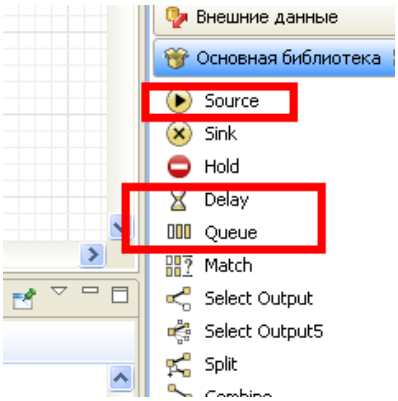


Рис. 7.143. Объекты «source2», «queue6», «delay6»

15. Настройка объектов «queue6», «delay6».

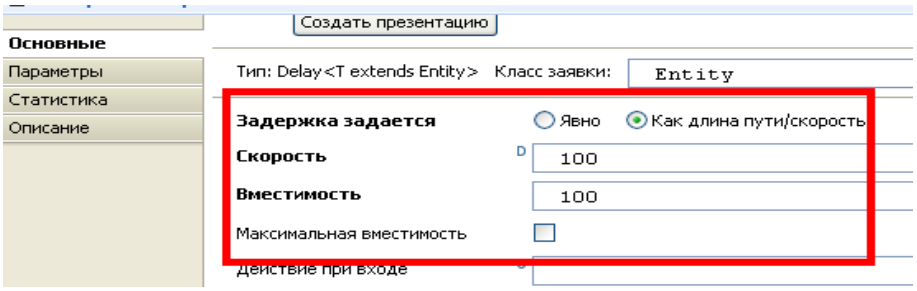


Рис. 7.144. Настройка объектов «queue6», «delay6»

16. Из 3D библиотеки в форму добавляем объект «window3d».

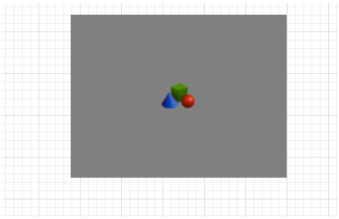


Рис. 7.145. Объект «window3d»

17. Из палитра 3D объекты и 3D добавляем на форму объекты, после чего располагаем их как на рис.7.146 и сгруппируем.

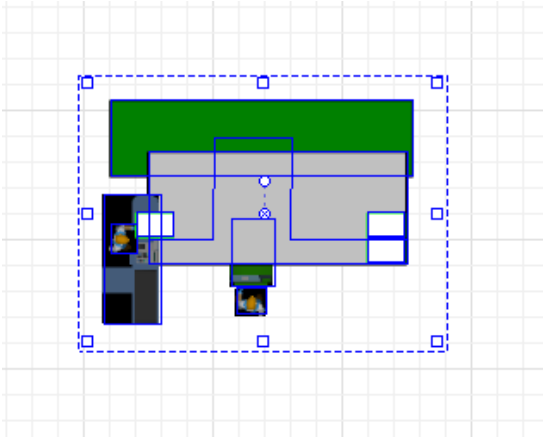


Рис. 7.146. 3D модель резальной машины

18. Из палитра 3D объекты и 3D добавляем на форму объекты, после чего располагаем их как на рис. 7.147 и сгруппируем.

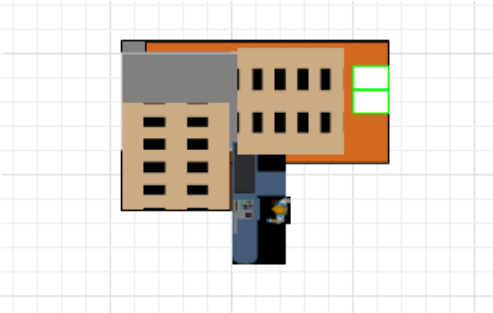


Рис.7.147. 3D модель фольцовочной машины

19. Из палитра 3D объекты и 3D добавляем на форму объекты, после чего располагаем их как на рис.7.149 и сгруппируем.

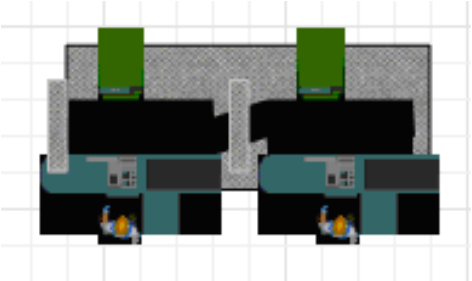


Рис.7.148. 3D модель биговочной машины

20. Из палитра 3D объекты и 3D добавляем на форму объекты, после чего располагаем их как на рис.7.149 и сгруппируем.

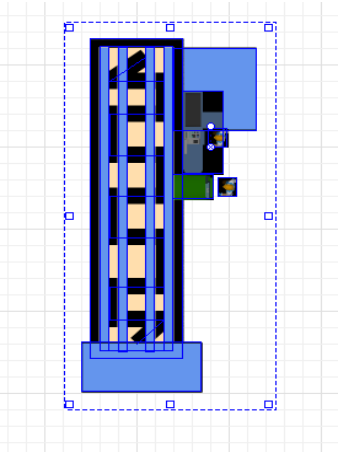


Рис.7.149. 3D модель листободборной машины

21. Из палитра 3D объекты и 3D добавляем на форму объекты, после чего располагаем их как на рис.7.150 и сгруппируем.

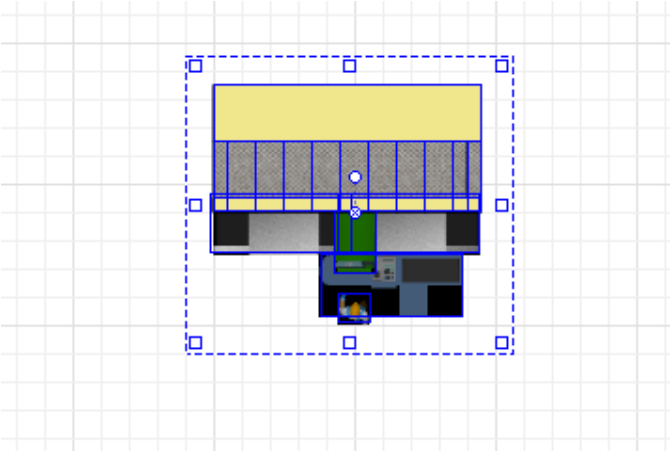


Рис.7.150. 3D модель скрепляющей машины

22. Из палитра 3D добавим объект «rectangle»

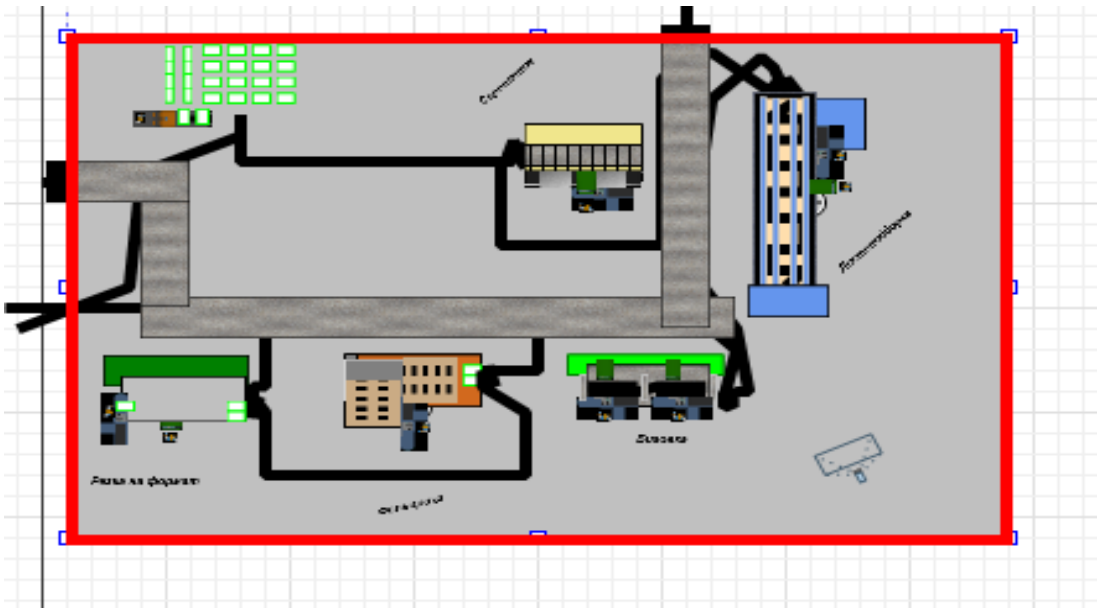


Рис.7.151. Объект «rectangle»

23. Добавим объекты «rectangle1» сделаем несколько копий и утановим их как показано на рис.7.152.

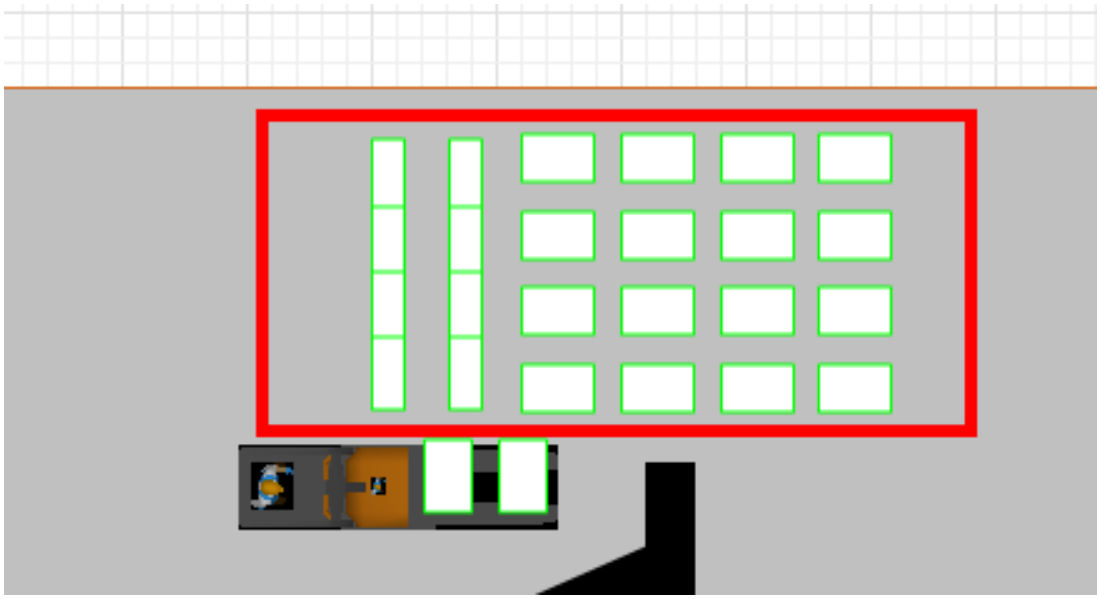


Рис.7.152. Объект «rectangle1»

24. Добавим объекты «rectangle2», «rectangle3», «rectangle4», «rectangle5» и расположим их как на рис.7.153.

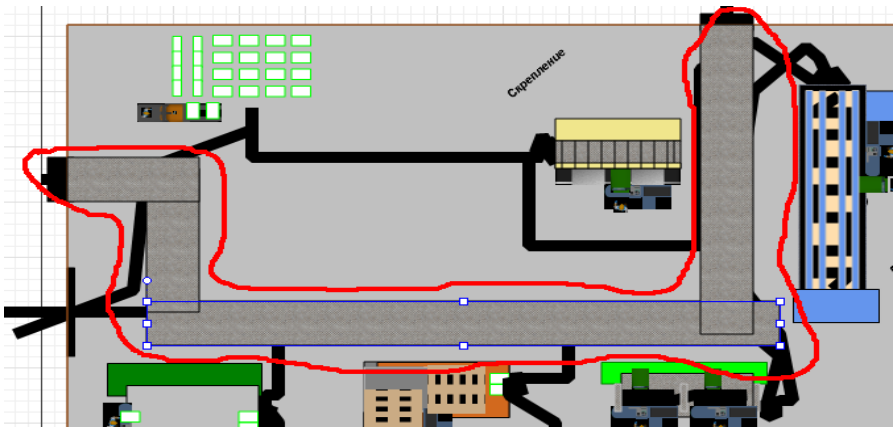


Рис.7.153. Объекты «rectangle2», «rectangle3», «rectangle4», «rectangle5»

25. Для объекта «source» фигуру анимации ставим «rectangle2»

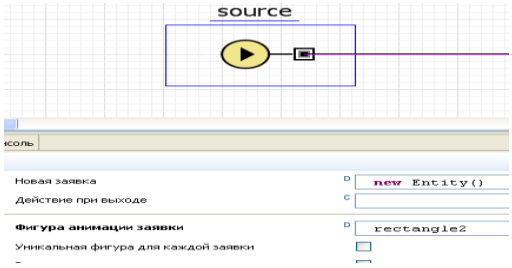


Рис.7.154. Объект «source».

26. Для объекта «delay» фигуру анимации ставим «polyline»
27. Для объекта «delay1» фигуру анимации ставим «polyline1»
28. Для объекта «delay2» фигуру анимации ставим «polyline2»
29. Для объекта «delay3» фигуру анимации ставим «polyline3»
30. Для объекта «delay4» фигуру анимации ставим «polyline5»
31. Для объекта «delay5» фигуру анимации ставим «polyline5»
32. Для объекта «delay6» фигуру анимации ставим «polyline6»
33. Для объекта «source1» фигуру анимации ставим «group11»
34. Для объекта «source2» фигуру анимации ставим «group12»
35. Сделаем стены, для этого из палитры палитры 3D на форму добавим ломанную и в свойствах объекта поставим значения как на рис.155

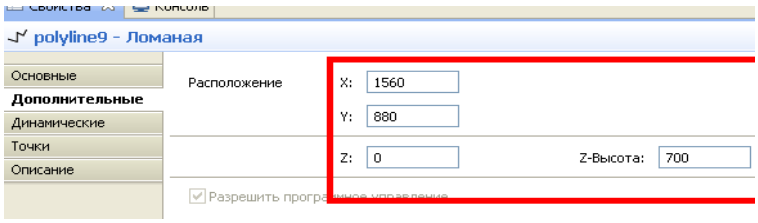


Рис.7.155. Объект «Ломаная»

36. Добавим камеру и настроем его свойства

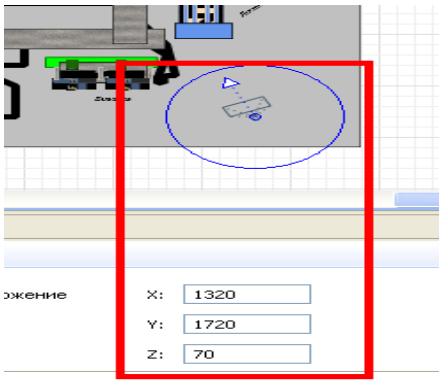


Рис.7.156. Объект «Камера »

37. Запускаем модель нажатием на кнопку F7.

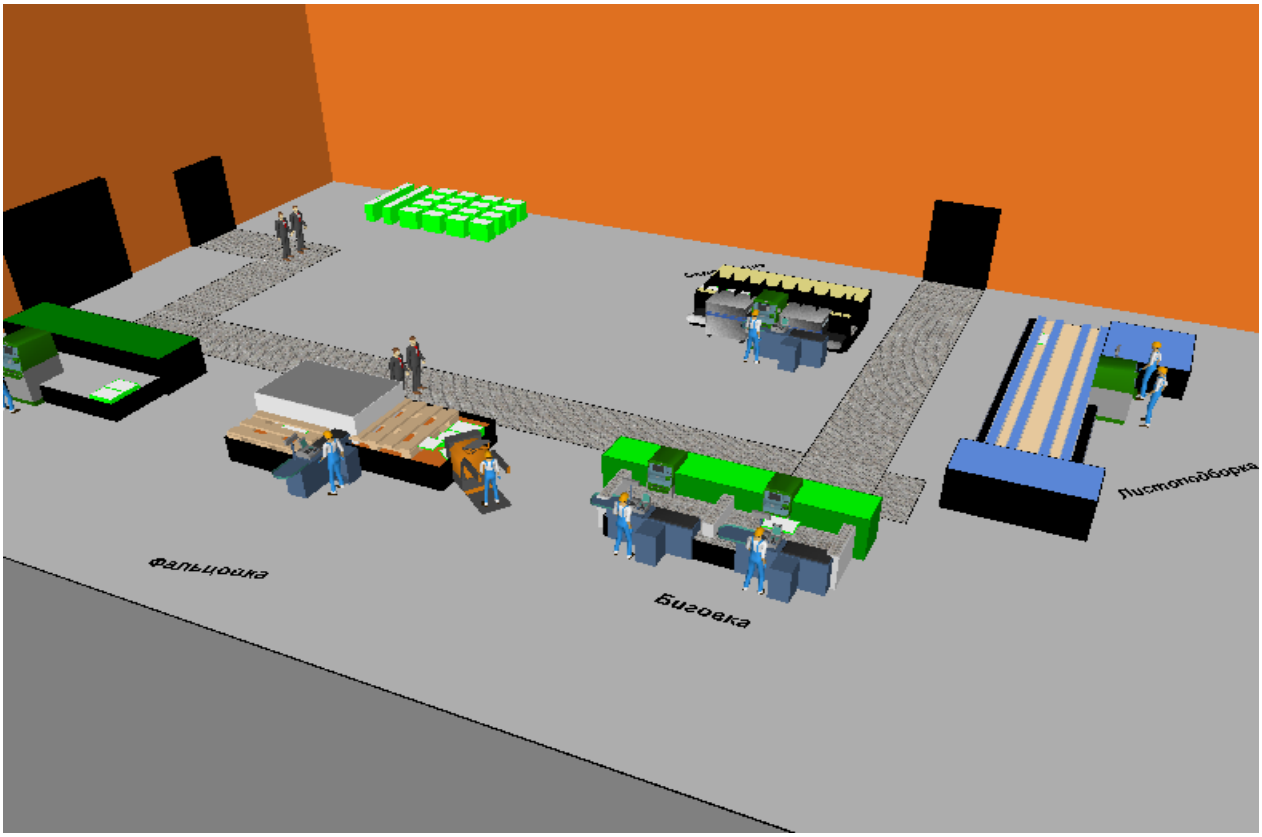


Рис.7.157. 3D модель послепечатного процесса

ГЛАВА 8

МОДЕЛИРОВАНИЕ ДВИЖЕНИЯ ПЕШЕХОДОВ

При моделировании пешеходных потоков решаются следующие задачи:

1) Расчет пропускной способности помещений. Допустим, необходимо построить гипермаркет, станцию метро, железнодорожный или аэровокзал. В таком случае появляется задача: как сконфигурировать помещение таким образом, чтобы пешеходные потоки не мешали друг другу, сервисы справлялись с нагрузкой, а люди чувствовали себя комфортно;

2) Организация пешеходного движения. При строительстве парков развлечений, музеев, стадионов возникают вопросы организации движения людей, например: «Где поставить киоск или рекламный щит?», «Как организовать процесс, чтобы люди, стоящие в очередях за билетами, не мешали проходящей толпе?». Чем больше размер помещения и количество посетителей, тем актуальнее данные вопросы;

3) Анализ вариантов эвакуации людей. При эвакуации люди ведут себя агрессивно, стараясь как можно быстрее покинуть зону опасности. Встает вопрос организации пешеходных потоков в нештатных ситуациях. Для этого применяются соответствующие знаки, указывающие на аварийные выходы, кроме того, часто за эвакуацию отвечают специальные люди. Моделирование чрезвычайных происшествий позволяет заранее предвидеть проблемы, возникающие при эвакуации людей, и в конечном счете спасти человеческие жизни.

8.1. Пешеходная динамика покупателей в магазине

Постановка задачи. Построить имитационную модель посещения небольшого магазина покупателями. Для моделирование выбираем среду имитационного моделирования AnyLogic. Для построения модели мы будем использовать объекты Пешеходной библиотеки:

Шаг.1. Создание новой модели

1. Щелкните мышью по кнопке панели инструментов Создать . Появится окно Мастера создания модели.

2. Задайте имя новой модели. В поле Имя модели введите Subway Entrance.

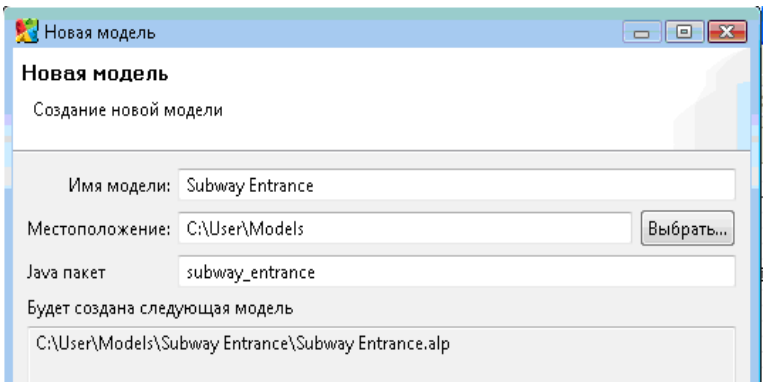


Рис.8.1. Создание новой модели

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

4. Щелкните мышью по кнопке Далее. Откроется вторая страница Мастера создания модели.

5. Здесь Вам будет предложено выбрать шаблон модели, на базе которого Вы будете разрабатывать модель. Поскольку мы хотим научить Вас процессу создания модели "с нуля", чтобы в дальнейшем Вы могли самостоятельно создавать аналогичные модели, не выбирайте шаблон модели, а просто закончите создание модели, щелкнув мышью по кнопке Готово.

Пользовательский интерфейс AnyLogic. В левой части рабочей области будет находиться панель Проекты. Панель Проекты обеспечивает легкую навигацию по элементам моделей, открытых в текущий момент времени. Поскольку модель организована иерархически, то она отображается в виде дерева: сама модель образует верхний уровень дерева; эксперименты, классы активных объектов и Java классы образуют следующий уровень; элементы, входящие в состав активных объектов, вложены в соответствующую подветвь дерева класса активного объекта и т.д.

В правой рабочей области будет отображаться панель Палитра, а внизу – панель Свойства. Панель Палитра содержит разделенные по категориям элементы, которые могут быть добавлены на диаграмму класса активного объекта или эксперимента. Панель Свойства используется для просмотра и изменения свойств выбранного в данный момент элемента (или элементов)

модели. В центре рабочей области AnyLogic Вы увидите графический редактор диаграммы класса активного объекта Main.

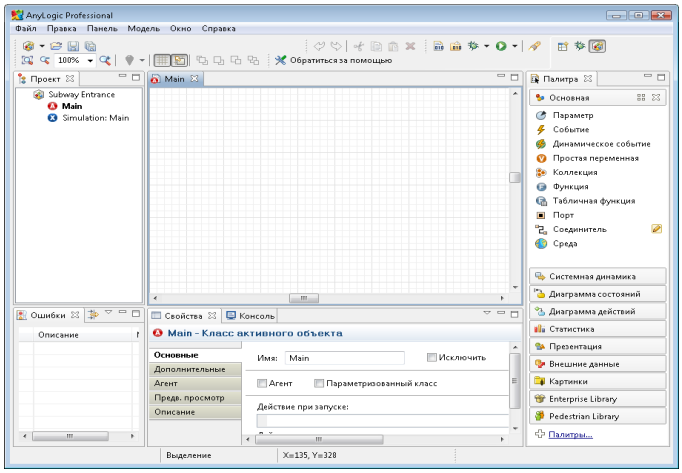


Рис.8.2. Интерфейс среды AnyLogic

Создание анимации. Теперь мы готовы к тому, чтобы начать разработку нашей модели. Прежде всего, нам нужно создать анимацию модели. С помощью анимации Вы сможете визуально отображать поведение модели, кроме того, анимация нужна для того, чтобы графически задать объекты среды модели. Чтобы облегчить рисование, мы вначале добавим изображение плана павильона метро. Мы не будем рисовать план в графическом редакторе, а просто вставим уже готовое изображение. Рассмотрим процедуру как добавить в модель рисунок с изображением плана павильона.

Шаг 2. Построение схемы магазина

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

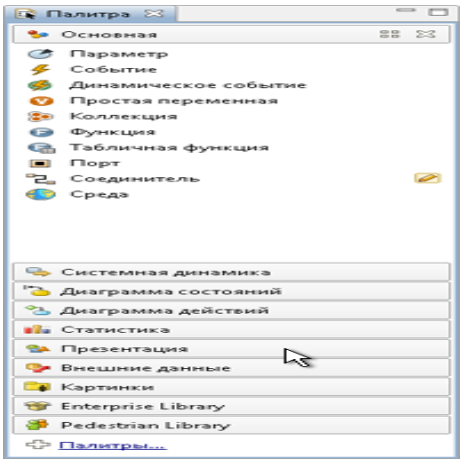


Рис.8.3. Палитра Презентация

2. Палитра Презентация содержит элементы, используемые для рисования презентаций моделей: фигуры, с помощью которых Вы можете рисовать сложные презентации, а также элементы управления, с помощью которых Вы можете сделать презентации интерактивными.

3. Перетащите элемент Изображение из палитры Презентация на диаграмму класса активного объекта. Поместите его так, как показано на рисунке:

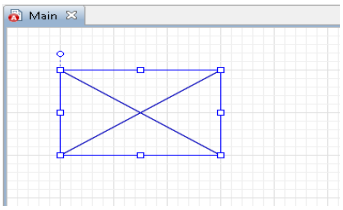


Рис.8.4.Размещение изображения

4. Задайте свойства изображения в панели Свойства. Щелкните мышью по кнопке Добавить и выберите файл изображения плана павильона. Вы увидите добавленное изображение в области предварительного просмотра на панели Свойства:

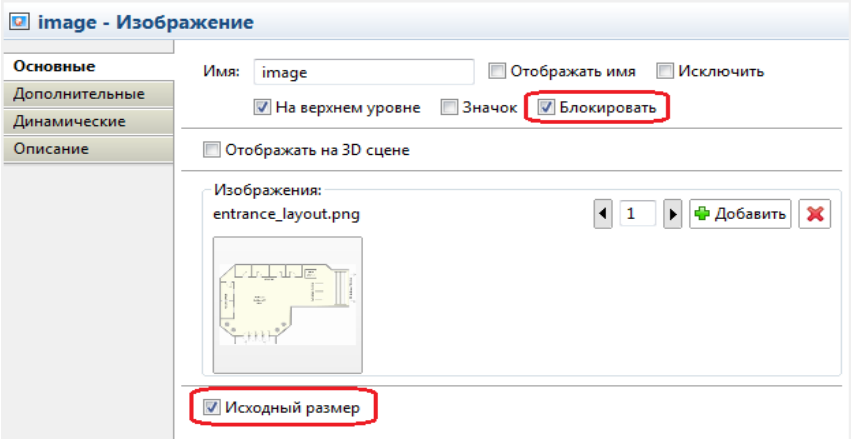


Рис.8.5. Подготовка изображения схемы магазина

5. Чтобы сохранить исходный размер изображения, установите флажок Исходный размер.

6. Заблокируйте изображение, установив флажок Блокировать. Вы не сможете выбрать заблокированную фигуру в графическом редакторе до тех пор, пока не снимете с нее блокировку. Мы делаем так потому, что мы будем рисовать другие фигуры поверх этого изображения, и поэтому мы хотим исключить возможность случайного редактирования изображения при рисовании этих фигур.

7. Изображение должно будет выглядеть следующим образом:

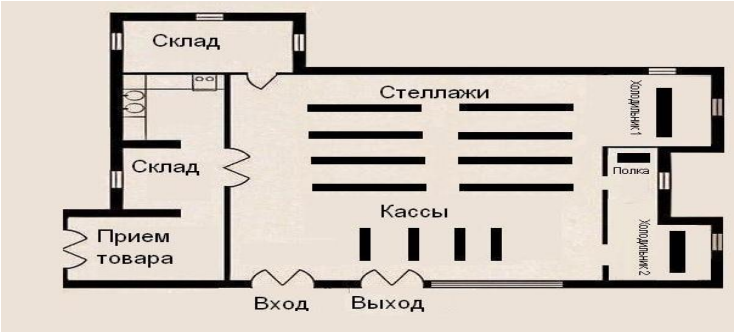


Рис.8.6. Изображение схемы магазина

Теперь мы нарисуем на анимации объекты моделируемой среды. Вначале мы нарисуем границу моделируемого нами пространства, играющей роль стен здания.

Рассмотрим, как нарисовать границы здания.

1. Чтобы было легче рисовать, лучше отключить сетку и увеличить масштаб анимации с помощью соответствующих кнопок панели инструментов:

Рис.8.7

2. Нарисуйте ломаную, как показано на рисунке ниже. Чтобы нарисовать ломаную, сделайте двойной щелчок мышью по элементу Ломаная в палитре (при этом его значок должен поменяться на этот:). Теперь Вы можете рисовать ломаную точка за точкой, последовательно щелкая мышью в тех точках диаграммы, куда Вы хотите поместить вершины ломаной. Чтобы завершить рисование, добавьте последнюю точку ломаной двойным щелчком мыши.

Измените свойства только что нарисованной ломаной:

Назовите ломаную walls.

Измените цвет и толщину линии, чтобы она была более заметна на презентации. Сделайте ломаную линию замкнутой. Установите флажок Замкнутая. Тем самым, Вы сделаете ломаную замкнутой, соединив ее первую и последнюю точки.

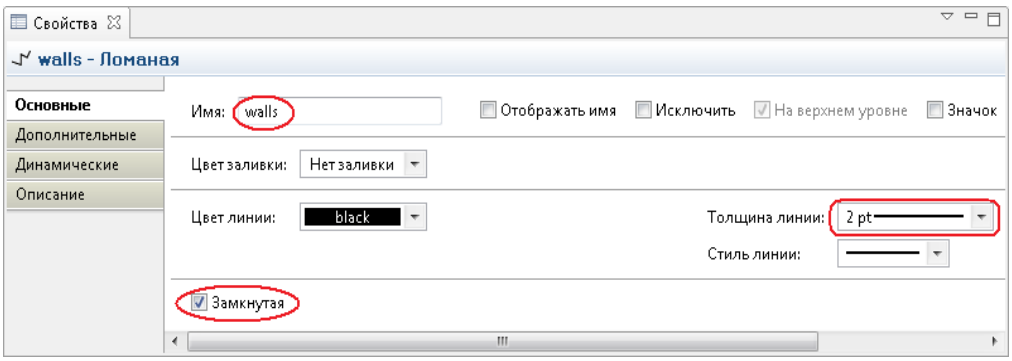


Рис.8.8. Построение границ магазина

Зачастую границы моделируемого пространства рисуются с помощью нескольких фигур, поэтому есть требование, чтобы все такие фигуры были добавлены в одну группу, а эта группа была указана в параметре Стены соответствующего библиотечного объекта. Поэтому сейчас мы добавим нарисованную нами ломаную линию walls в группу.

Добавьте только что нарисованную ломаную линию в группу. Щелкните по ней правой кнопкой мыши (при этом она должна подсветиться синим цветом) и выберите Группировка|Создать группу из контекстного меню.

Теперь нужно задать области входа и выхода покупателей.

Вначале мы нарисуем область входа – линию, в которой покупатели будут появляться. Область входа может быть задана линией или ломаной.

Нарисуйте вход. Перетащите элемент Линия из палитры Презентация в графический редактор.

Перетащите точки линии так, чтобы они были расположены примерно в тех же местах диаграммы, что и точки линии на рисунке.

Назовите линию entry. Позднее мы будем ссылаться на эту линию в блоках диаграммы процесса именно по этому имени.

Нарисуйте выход.

Аналогично нарисуйте линию выхода покупателей из моделируемого пространства. При достижении линии выхода покупатели будут удаляться из моделируемой среды.

Назовите линию exit.

Обе линии должны находиться полностью внутри фигуры, задающей границу моделируемой среды – в нашем случае – внутри ломаной линии walls.

Шаг 3. Моделирование потока покупателей

Теперь мы закончим создание модели, моделирующей простейший поток покупателей, создав диаграмму моделируемого нами процесса.

С помощью диаграммы процесса в моделях аналогичной динамики задается поведение покупателей (так же, как в Основной библиотеке с их помощью задается поведение заявок). Диаграмма процесса в AnyLogic создается путем добавления объектов библиотеки из палитры на диаграмму класса активного объекта, соединения их портов и изменения значений свойств объектов в соответствии с требованиями Вашей модели.

Попадающие в моделируемую систему люди будут последовательно проходить по блокам созданной Вами диаграммы процесса.

Помимо объектов, составляющих диаграмму процесса, модели библиотеки состоят из объектов, моделирующих объекты среды (стены, различные области, сервисы, очереди и т.д.). Чтобы задать объект среды, Вы должны вначале графически нарисовать его на анимации, а затем добавить соответствующий объект библиотеки на структурную диаграмму активного класса модели и задать необходимые свойства этого объекта.

Создание диаграммы процесса. Для этого необходимо добавить на диаграмму класса Main объекты.

1. Чтобы добавить на диаграмму объект Пешеходной библиотеки, нужно открыть в панели Палитра палитру этой библиотеки, щелкнув мышью по панели с ее заголовком, а затем перетащить нужный Вам объект из палитры на диаграмму класса.
2. Разместите объекты так, как показано на рисунке.

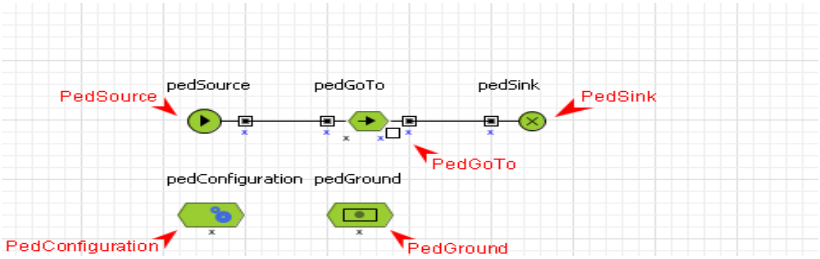


Рис.8.9. Размещение объектов

Соедините порты добавленных на диаграмму объектов, как показано на рисунке:

1. Чтобы соединить порты объектов, сделайте двойной щелчок мышью по одному порту, затем при желании щелкните в тех местах диаграммы, где Вы хотите добавить точку изгиба соединителя, и завершите создание, сделав двойной щелчок мышью по второму порту. При этом между портами появится соединитель заданной Вами формы.

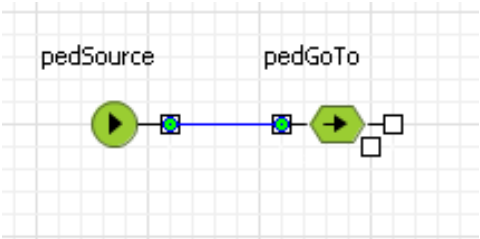


Рис.8.10. Соединение объектов

2. Если Вы выделите соединитель щелчком мыши, и его конечные точки в портах будут подсвечиваться светло-зеленым цветом, то это будет означать, что Вы успешно соединили порты. Иначе же Вам придется проверить, точно ли в порт была добавлена одна из двух конечных точек соединителя, и если нужно, то передвинуть ее туда.

Согласно принятым стандартам, блоки в диаграмме процесса обычно располагаются цепочкой слева направо, представляя собой последовательную очередность операций, которые будут производиться над пешеходом.

Теперь, когда мы создали диаграмму моделируемого процесса, нам осталось только немного подкорректировать значения объектов диаграммы в соответствии с нашими требованиями. Давайте последовательно сконфигурируем каждый добавленный нами библиотечный объект (на приведенном выше рисунке мы отметили тип каждого блока, чтобы Вам было легче привыкнуть к пока незнакомым пиктограммам этих объектов).

Объект PedGround позволяет задавать двухмерное пространство в моделируемой среде, представляющее собой «этаж», т.е. поверхность, по которой будут перемещаться покупатели. Этажи могут быть ограничены какой-то стеной или быть неограниченными. Стены – это объекты, которые люди не могут пересекать. Стены являются частью этажа, то есть одна стена не может быть использована несколькими этажами.

Измените свойства блока pedGround:

1. Свойства объекта (как и любого другого элемента AnyLogic) можно изменить в панели Свойства.

Обратите внимание, что панель Свойства является контекстно-зависимой – она отображает свойства выделенного в текущий момент элемента. Поэтому для изменения свойств элемента нужно будет предварительно щелчком мыши выделить его в графическом редакторе или в панели Проекты. Чтобы у Вас всегда была уверенность в том, что в текущий момент в рабочем пространстве выбран именно нужный Вам элемент, и именно его свойства Вы редактируете в панели Свойства, обращайте внимание на первую строку, показываемую в панели Свойства – в ней отображается имя выбранного в текущий момент времени элемента и его тип.

2. Задайте стены моделируемого этажа. Введите в поле Стены (группа, необязательный) имя нашей группы: group, созданной ранее именно для этой цели.

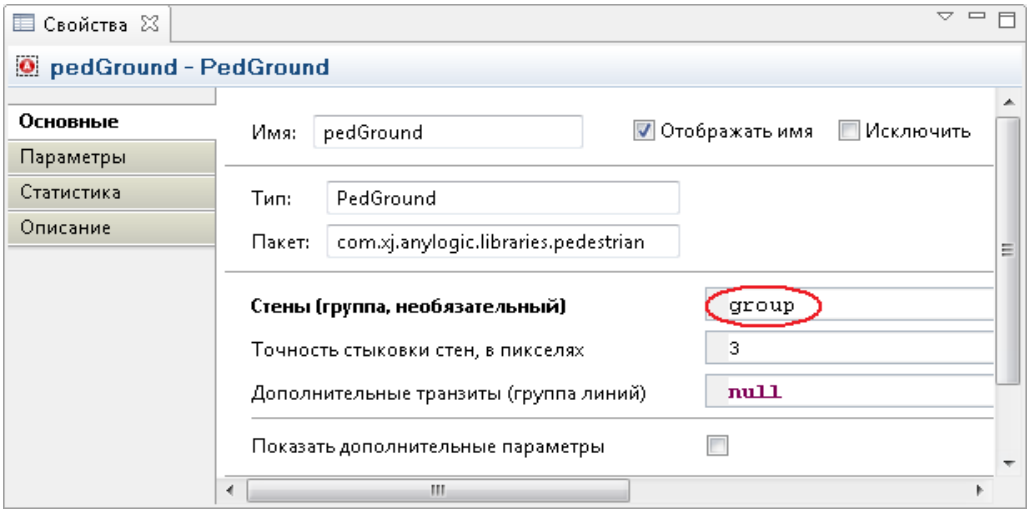


Рис.8.11. Создание покупателей

Объект PedSource создает покупателей. Обычно он используется в качестве начальной точки диаграммы процесса, формализующей поток людей. В нашем примере он моделирует приход покупателей в магазин.

Измените свойства блока pedSource:

1. Укажите имя объекта PedGround, задающего этаж, на который будут прибывать пассажиры. В поле Этаж (PedGround) введите имя добавленного Вами ранее объекта PedGround: pedGround.

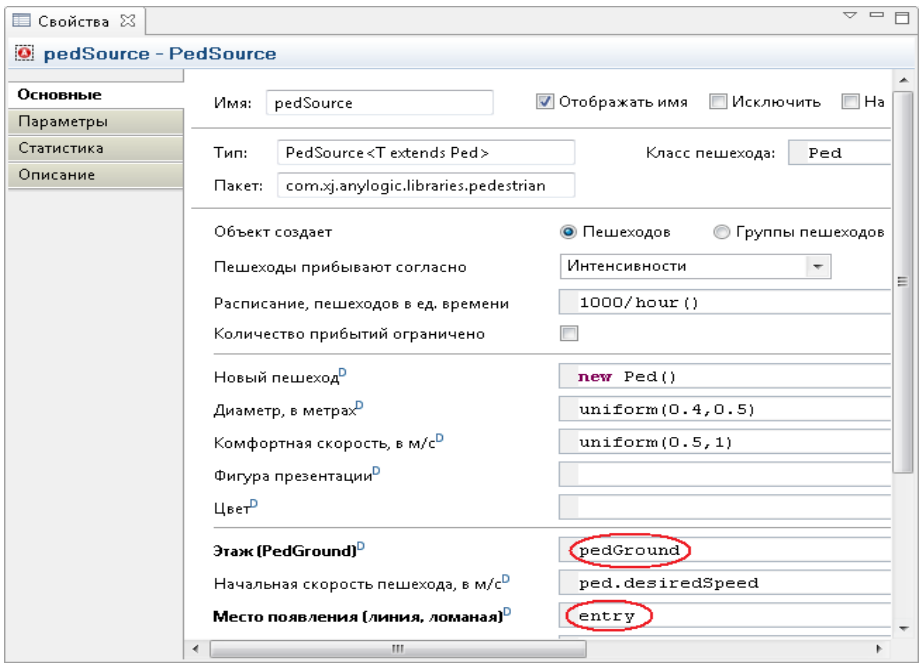


Рис.8.12. Подготовка точки прихода покупателей в магазин

2. Задайте место на этаже, где будут появляться покупатели. Место появления покупателей на этаже может быть задано линией или ломаной. Введите entry (имя ломаной, нарисованной нами ранее для этой цели) в поле Место появления (линия, ломаная). Теперь наши покупатели будут появляться в случайно выбранной точке линии entry.

Следующий объект в созданной нами диаграмме процесса - PedGoTo. Этот объект моделирует перемещение пешеходов из текущего местоположения в другое (заданное параметром этого объекта). С помощью этого объекта мы будем моделировать то, как покупатели перемещаются от входа в павильон к выходу.

Измените свойства блока pedGoTo:

1. Задайте то место, куда будут перемещаться покупатели, достигшие этого блока в диаграмме процесса. Такое место может быть задано линией или точкой, нарисованной на презентации. На данный момент мы хотим, чтобы покупатели, вошедшие в магазин, сразу двигались к выходу. Поэтому введите exit (имя линии, нарисованной нами на плане у выхода из магазина) в поле Цель (точка,линия).Теперь для того, чтобы покинуть моделируемую среду, покупатели должны будут вначале дойти до заданной области выхода.

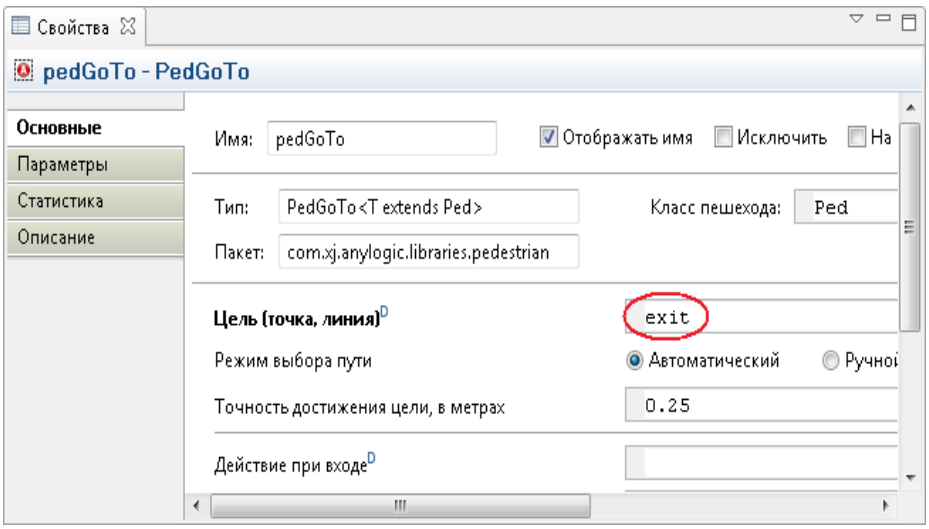


Рис.8.13. Организация перемещения покупателей

Оставьте заданные по умолчанию свойства объекта PedSink без изменений. Объект PedSink удаляет поступивших в объект покупателей из моделируемой среды. Обычно объект используется в качестве конечной точки диаграммы процесса.

Объект PedConfiguration позволяет изменять общие настройки библиотеки, настроив ее под конкретную задачу таким образом, чтобы повысить производительность модели.

Измените свойства блока pedConfiguration:

1. Сбросьте флажок Скрыть фигуры среды, чтобы не отображать на анимации фигуры, которые использовались для задания моделируемой среды (стен, различных областей, сервисов и т.д.).

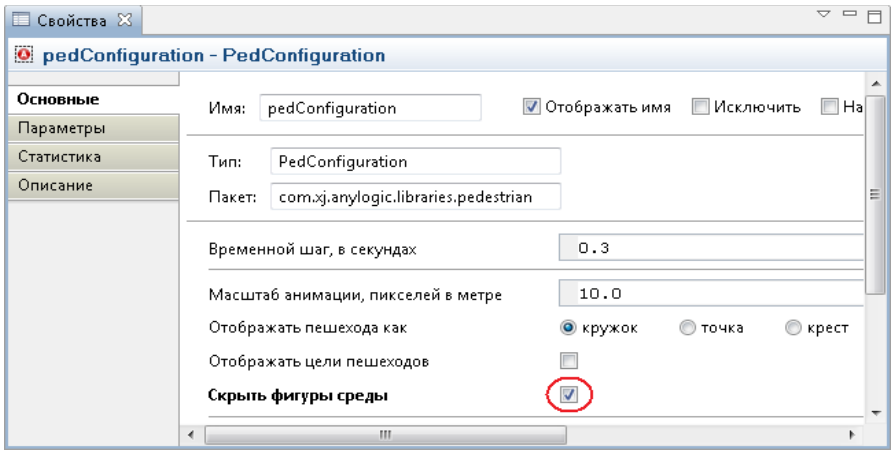


Рис.8.14. Определение временного шага

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

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

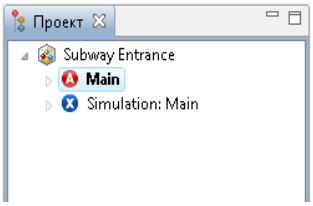


Рис.8.15.

Если мы сейчас запустим модель, то будут моделироваться 100 единиц модельного времени, после чего выполнение модели будет остановлено. Поскольку Вы можете захотеть наблюдать поведение модели в течение длительного периода (до того момента, пока Вы сами не остановите выполнение модели), нужно изменить соответствующие настройки эксперимента.

Уберите условие остановки модели по прошествии заданного времени:

1. В панели Проекты, выделите эксперимент Simulation:Main щелчком мыши. Перейдите на страницу Модельное время панели Свойства. Здесь задаются установки модельного времени. Вы можете увидеть, что по умолчанию в качестве единиц модельного времени в моделях изучения движения пешеходов используются секунды.

2. Чтобы убрать условие остановки модели, выберите Нет из выпадающего списка Остановить.

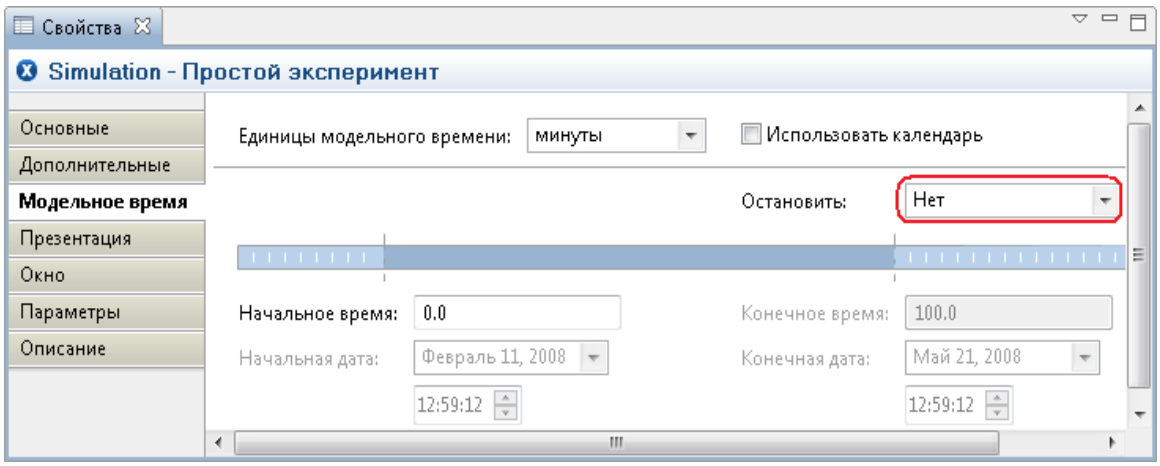


Рис.8.16. Определение длительности эксперимента

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

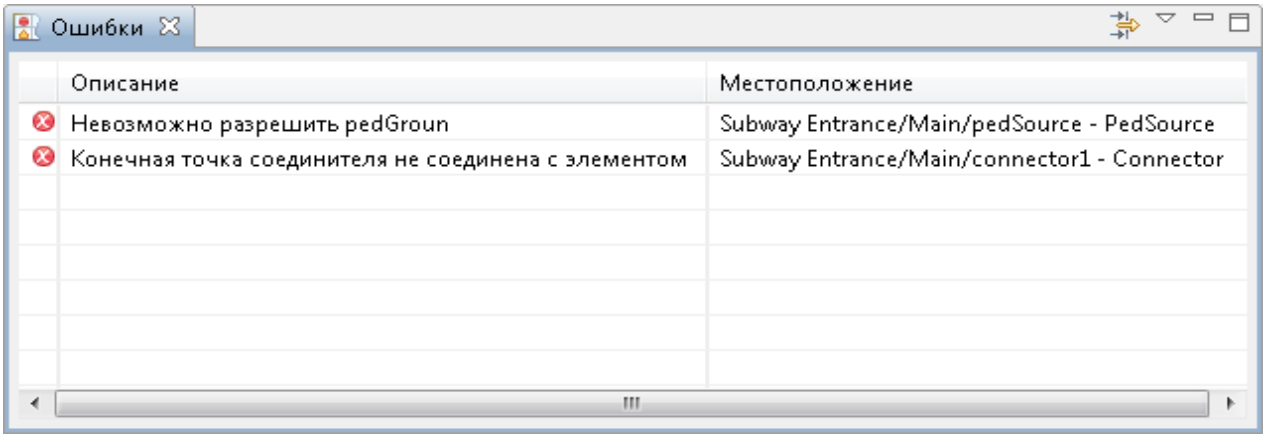


Рис.8.17. Панель Ошибки

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

Запустите модель:

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

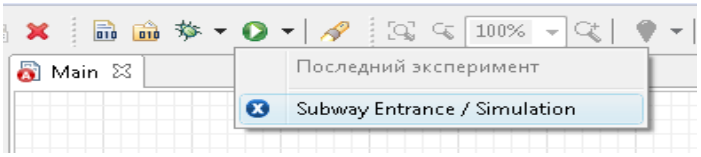


Рис.8.18. Запуск модели

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

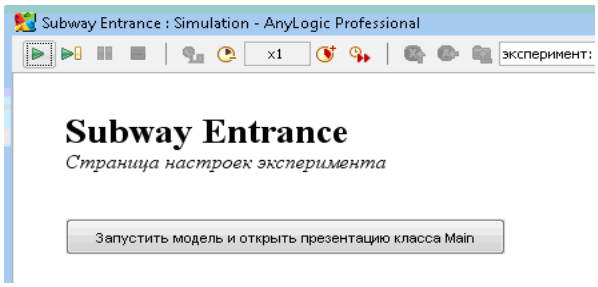


Рис.8.19. Интерфейс оболочки модели

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

Щелкните по кнопке Запустить модель и открыть презентацию класса Main. Модель запустится и Вы сможете пронаблюдать за динамикой моделируемого процесса с помощью нарисованной Вами в классе Main презентации.

Можно увидеть, что покупатели входят в магазин и движутся к выходу.

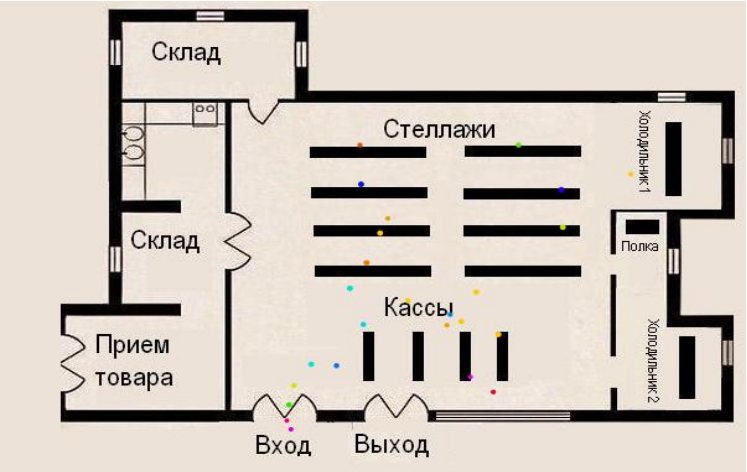
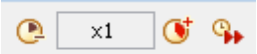


Рис.8.20. Работа модели

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

.

Вы можете переключиться в режим виртуального времени, тогда модель будет выполняться на максимально возможной скорости и Вы сможете быстро промоделировать работу системы за долгий период времени. Чтобы переключиться в режим виртуального времени, нужно щелкнуть мышью по кнопке панели инструментов Виртуальное время Реальное/виртуальное время
.

Когда Вы захотите остановить выполнение модели, щелкните мышью по кнопке панели управления окна презентации Прекратить выполнение
.

Моделирование касс. Теперь мы добавим в нашу модель кассы для оплаты товара. Таким образом, мы покажем, как моделируются сервисы.

Вначале мы изменим анимацию модели, а затем – диаграмму процесса.

Изменение анимации модели

1. Нарисуйте четыре линии, которые будут графически задавать турникеты в нашей модели.
2. Чтобы нарисовать линию, сделайте двойной щелчок мышью по элементу Линия в палитре Презентация (при этом его значок должен поменяться на этот:). Затем рисуйте линию согласно данным на рисунке ниже инструкциям:

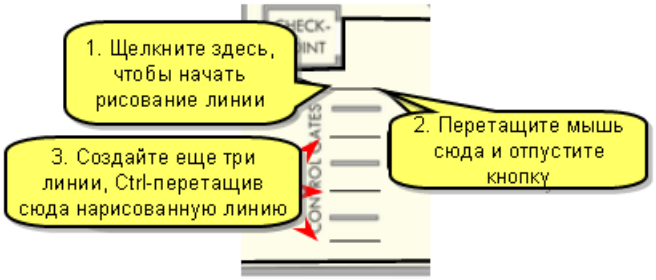


Рис.8.21. Изменение анимации

3. Очень важно, где Вы поместите начальную точку линии. Покупатели будут двигаться от точки, которую Вы нарисуете первой, к точке, которую Вы нарисуете последней.

4. Создайте группу фигур. Группа фигур нужна нам для того, чтобы сгруппировать нарисованные линии касс, поскольку мы хотим задать одним объектом сразу несколько идентичных сервисов. Выделите все линии (проще всего сделать это, последовательно щелкая по ним мышью с нажатой клавишей Ctrl), щелкните правой кнопкой мыши по выделенным фигурам и выберите Группировка/Создать группу из контекстного меню. Вы увидите в панели Свойства свойства только что созданной группы. Переименуйте группу в gatesGroup.

5. Нарисуйте линии, которые будут графически задавать очереди к кассам. Нарисуйте четыре ломаные линии так, как показано на рисунке ниже.

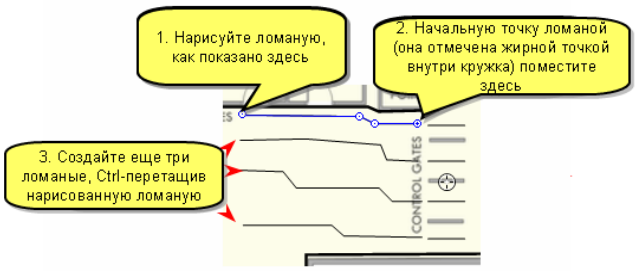


Рис.8.22. Подготовка линий подхода к кассам

6. Очень важно, где Вы поместите начальную точку ломаной. Эта точка в нашем случае будет соответствовать началу очереди. Поэтому ее нужно будет поместить рядом с соответствующей кассой.

7. Добавьте только что нарисованные ломаные в группу фигур (как это сделать, мы объяснили на примере ранее нарисованных линий). Назовите эту группу gatesQueuesGroup.

Теперь мы внесем небольшие изменения в диаграмму процесса.

Изменение диаграммы процесса:

1. Добавьте новые объекты и соедините их, как показано ниже:

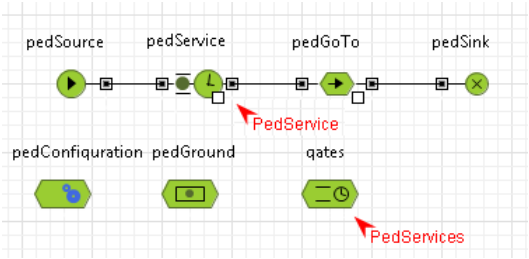


Рис.8.23. Добавление касс

Первым добавленным нами объектом является объект PedServices. Объект PedServices задает группу одинаковых физических объектов обслуживания (например, несколько касс). Объект позволяет задавать очереди и сервисы в любой комбинации и задавать правила выбора сервисов – какую очередь выбрать, какой сервис выбрать, к какой очереди должен обращаться сервис, может ли сервис обслуживать несколько очередей и т.д.).

Задав такой объект, Вы можете ссылаться на него в диаграмме моделируемого Вами процесса с помощью блока PedService.

Измените свойства объекта pedServices:

1. Назовите объект gates.
2. Задайте группу фигур, задающую линии точек сервисов (в нашем случае это кассы). Введите gatesGroup в поле Сервисы (группа линий).
3. Введите имя группы, задающей очереди к сервисам, в поле Очереди (группа линий, ломаных): gatesQueuesGroup. Мы задаем в качестве фигуры группу фигур, содержащую несколько ломаных, поскольку мы хотим задать несколько очередей (по одной на каждую кассу).
4. Оставьте остальные свойства без изменений. Вы увидите, что заданное Время задержки распределено по нормальному закону с минимальным значением, равным 2 секундам, и максимальным, равным 3 секундам. Функция uniform() является стандартной функцией генератора случайных чисел AnyLogic. AnyLogic предоставляет функции и других случайных распределений, таких как нормальное, экспоненциальное, треугольное, и т.д.

5. Оставьте выбранным Тип сервиса: Протяженный.

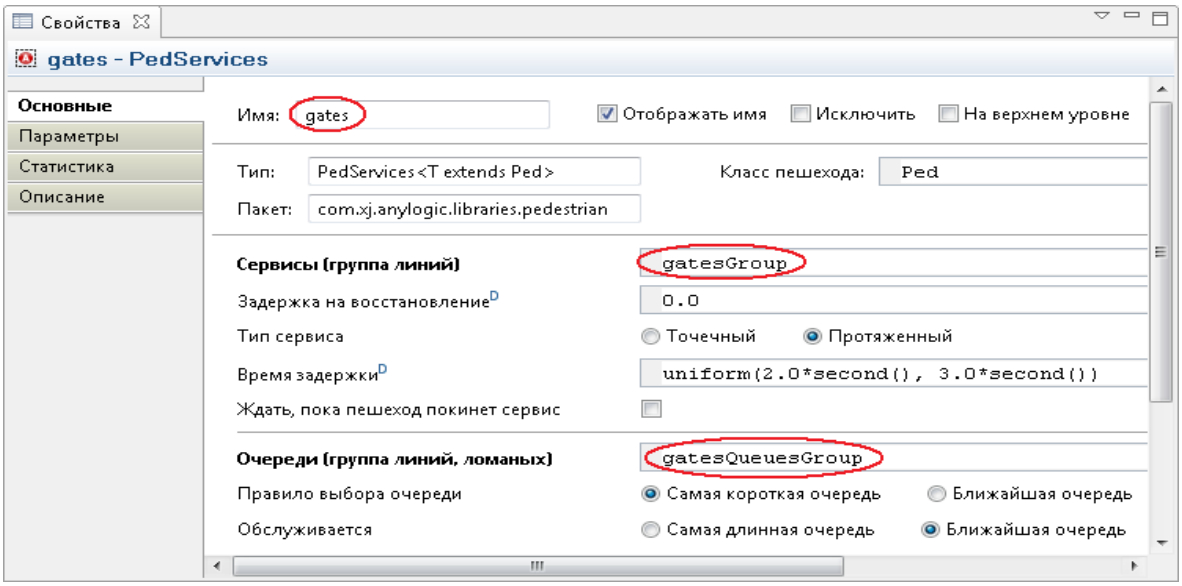


Рис.8.24. Модернизация системы СМО

Измените свойства объекта PedService. Объект PedService добавляется в диаграмму процесса, чтобы промоделировать, как покупатели обслуживаются в сервисе, заданном объектом PedServices.

Измените свойства объекта pedService:

1. В поле Сервис (PedServices) укажите имя объекта PedServices, задающего сервис, через который должны пройти покупатель (кассы). Введите здесь gates (имя нашего объекта PedServices).

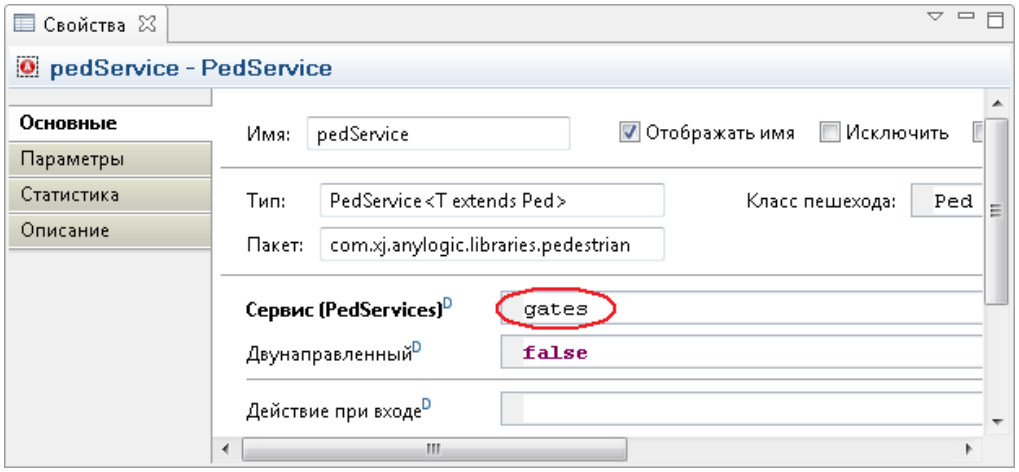


Рис.8.25. Модернизация системы прохода покупателей через кассы

Запустите модель и проследите за поведением модели с помощью анимации. Вы можете увидеть, что теперь покупатели проходят через кассы. Иногда перед кассами образуются небольшие очереди.

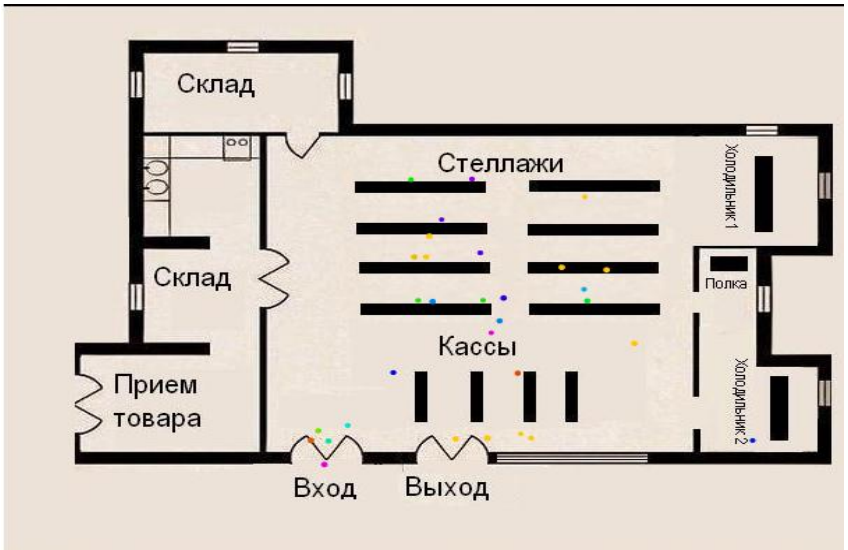


Рис.8.26. Динамика движения покупателей в магазине с учетом касс

Изменение интенсивности потока покупателей. Теперь покажем как можно динамически изменять параметры объектов Pedestrian Library. В текущей модели интенсивность потока покупателей задана фиксированной величиной. Мы же хотим иметь возможность интерактивно изменять интенсивность во время выполнения модели, чтобы проверить, насколько модель устойчива к возможным увеличениям нагрузки. Для этого мы добавим специальный элемент управления.

Добавьте бегунок, чтобы менять интенсивность потока людей:

1. Откройте палитру Элементы управления и перетащите элемент Бегунок из палитры на диаграмму класса Main (справа от плана магазина).

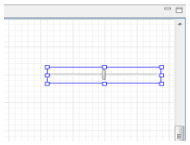


Рис.8.27. Добавление бегунка

2. Измените свойства бегунка. Мы хотим изменять с помощью этого бегунка значение параметра rate объекта pedSource. Для этого установите флажок Связать с и введите имя параметра, который мы хотим варьировать, в расположенном справа поле: pedSource.rate

3. Мы хотим дать возможность пользователю варьировать количество приходящих в минуту покупателей от 0 до 100. Введите 100/minute() в поле Максимальное значение.

4. Щелкните по кнопке Добавить метки... на странице свойств бегунка. Тем самым, мы создадим рядом с бегунком три текстовые метки, которые будут отображать его минимальное, текущее и максимальное значения.

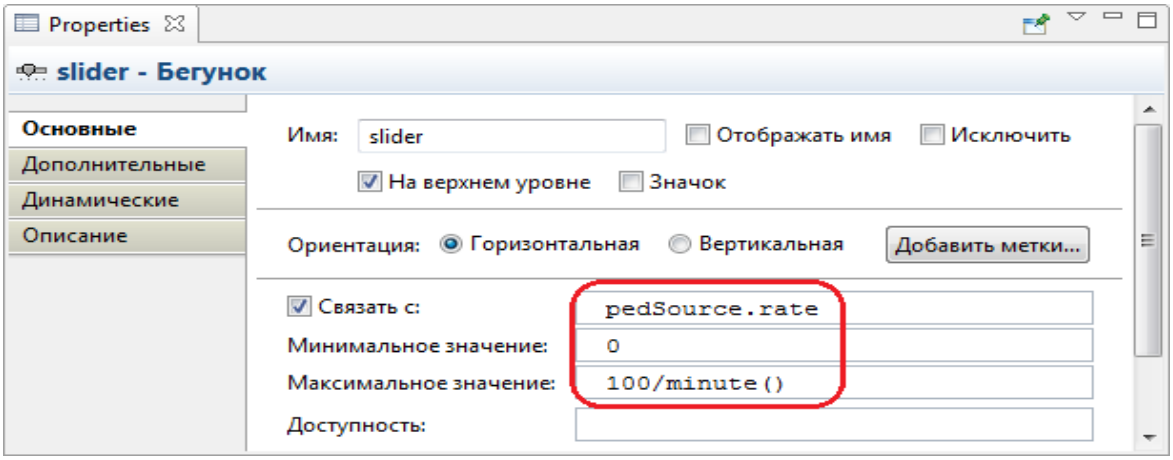


Рис.8.28. Изменение интенсивности покупателей

Чтобы другим пользователям модели было понятнее, какой именно параметр варьирует добавленный нами бегунок, добавьте рядом с ним поясняющую текстовую метку.

Добавьте текстовую метку для бегунка:

1. Добавьте к бегунку текстовую метку с заголовком. Перетащите элемент Текст Aaиз палитры Презентация на диаграмму (выше бегунка).

2. Введите текст, который Вы хотите показать с помощью этой метки, в поле Текст.

Запустите модель. Теперь Вы можете интерактивно изменять интенсивность потока покупателей с помощью созданного элемента управления. Таким образом, Вы можете проанализировать, как хорошо система справляется с текущей нагрузкой.

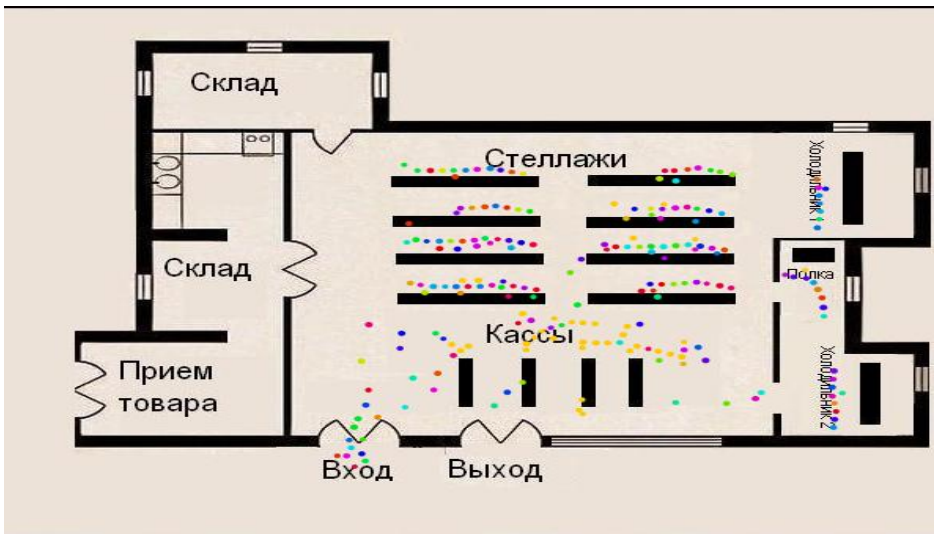


Рис.8.29. Динамика движения покупателей

Моделирование стеллажей с товарами. Теперь мы добавим в нашу модель стеллажи с товарами, холодильник.

Изменить анимацию модели:

1. Нарисуйте линии, которые будут графически задавать стеллажи в нашей модели. Нарисуйте линии так, как показано на рисунке ниже.

Рис.8.30.

2. Добавьте эти линии в новую группу фигур. Назовите ее windowsGroup.

3. Нарисуйте ломаные линии, которые будут графически задавать очереди к полкам. Нарисуйте ломаные так, как показано на рисунке ниже.

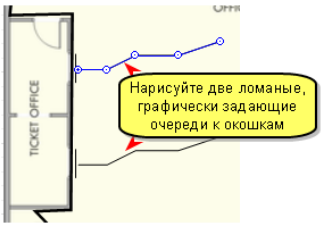


Рис.8.31

4. Первую точку ломаной рисуйте рядом с линией, задающей сервис. Именно эта точка будет соответствовать началу очереди

5. Добавьте эти ломаные в новую группу. Назовите группу windowsQueuesGroup.

Измените диаграмму процесса:

1. Добавьте новые объекты и соедините их, как показано ниже:

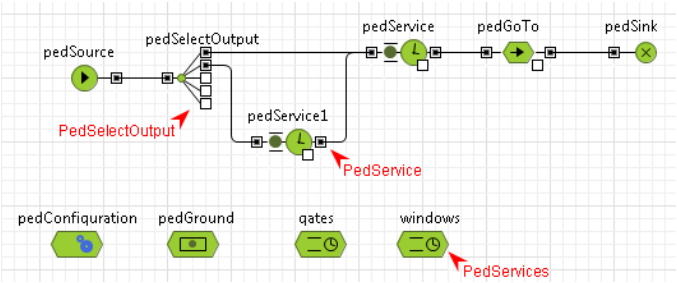


Рис.8.32. Модификация модели

Измените свойства объекта PedServices. Этот объект будет задавать параметры сервиса стеллажей.

Измените свойства только что добавленного объекта pedServices:

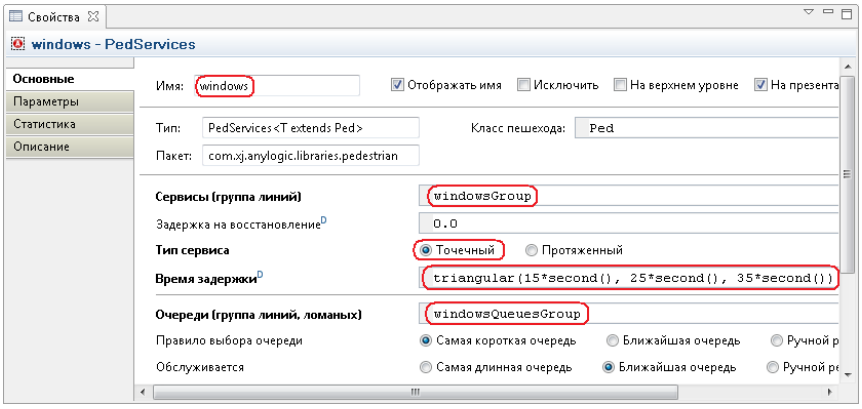


Рис.8.33. Покупка товара

1. Назовите этот объект windows.

2. Задайте группу фигур, задающую линии сервисов. Введите windowsGroup в поле Сервисы (группа линий).

3. Введите имя группы, задающей очереди к сервисам, в поле Очереди (группа линий, ломаных): windowsQueuesGroup.

4. Задайте время выбора товаров. Предположим, что время выбора распределено по треугольному закону с минимальным значением, равным 15

секундам, средним – равным 25 секундам, и максимальным - равным 35 секундам. Введите в поле Время задержки triangular(15*second(), 25*second(), 35*second())

5. Измените Тип сервиса на Точечный. Точечные сервисы используются тогда, когда для того, чтобы быть обслуженным, покупатель должен просто подойти к любой точке фигуры, задающей соответствующий сервис и провести там время, заданное как Время обслуживания этого сервиса. В нашем примере стеллажи представляют собой именно точечный сервис.

Объект PedSelectOutput является блоком принятия решения Пешеходной библиотеки. Покупатель, вошедший в блок PedSelectOutput, будет перенаправляться в один из пяти выходных портов в зависимости от заданных для этих портов коэффициентов предпочтения.

Объект PedSelectOutput будет перенаправлять покупателей без товара к выходу, а покупателей с товарами – к кассам.

Измените свойства объекта pedSelectOutput:

1. Задайте коэффициенты предпочтения для потоков покупателей, следующих к кассам (Коэфф. предпочтения 1) и к стеллажам (Коэфф. предпочтения 2). В этой модели мы полагаем, что доля покупателей, взявших товары, значительно выше:

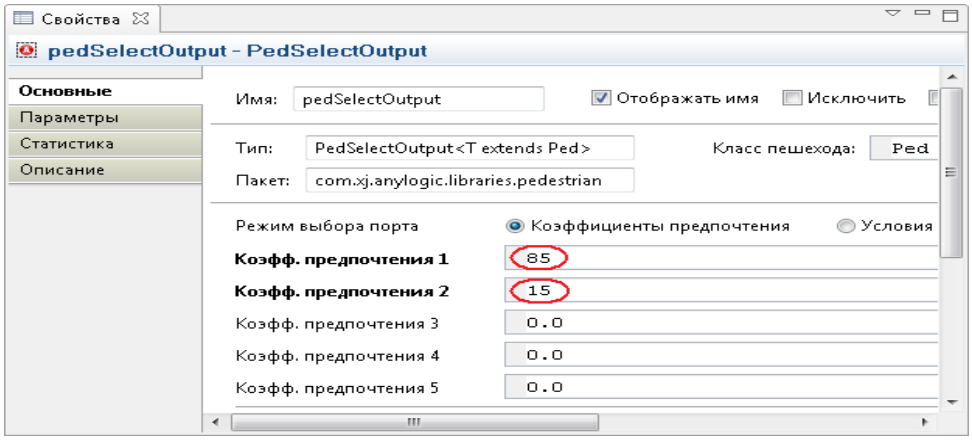


Рис.8.34. Организация выхода покупателей

Еще один объект PedService добавляется нами для того, чтобы направить проходящий через него поток покупателей на обслуживание в сервис – к стеллажам.

Измените свойства только что добавленного объекта pedService:

1. В поле Сервис (PedServices) укажите имя объекта PedServices, задающего сервис (полки с товарами), где должны быть обслужены проходящие через этот блок покупатели. Введите windows (имя соответствующего объекта PedServices) в поле Сервис (PedServices).

2. Введите ped.setColor(Color.orange) в качестве действия при выходе покупателя из объекта. Те покупатели, которые приобретут товар, будут отображаться на анимации желтым цветом.

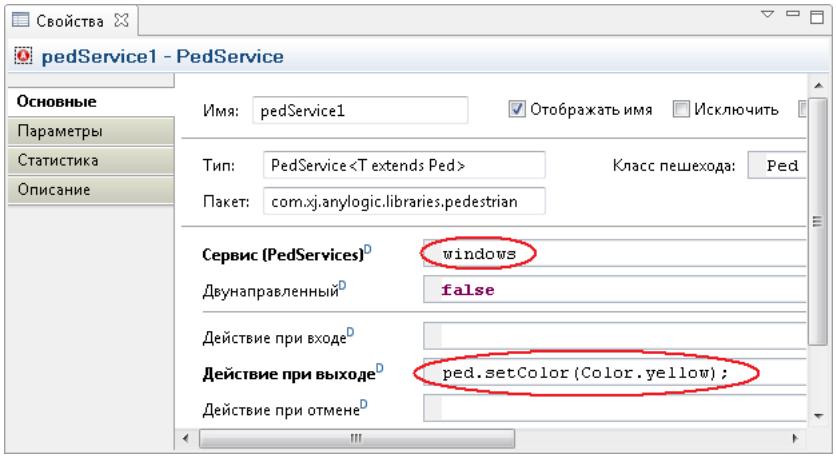


Рис.8.35. Организация приобретения товара

Запустите модель. Вы увидите, что некоторые покупатели теперь перед тем, как пройти к кассам, вначале подходят к стеллажам, чтобы приобрести товары.

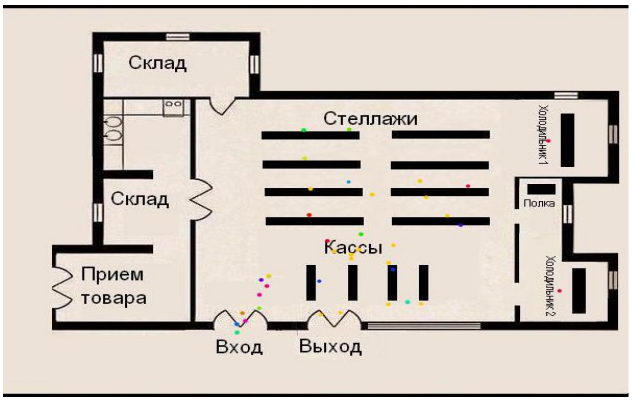


Рис.8.36. Динамика движения покупателей в магазине
с учетом касс и покупки товара

Таким образом, в результате разработки имитационной модели магазина, была создана модель посещаемости магазина. При создании модели использовалась пешеходная библиотека. В модели есть возможность регулирования потока покупателей. Посетители, совершившие покупку, отмече-

ны желтым цветом, тем самым можно наблюдать количество потенциальных покупателей.

Цель моделирования и сбор статистических данных. AnyLogic предоставляет пользователю удобные средства для сбора статистики по работе блоков диаграммы процесса Цель моделирования такой системы может быть разной. Однако наиболее типичной целью исследования в подобных задачах массового обслуживания является оценка эффективности системы, т. е. нахождение числовых значений характеристик, описывающих качество обслуживания системой потока посетителей.

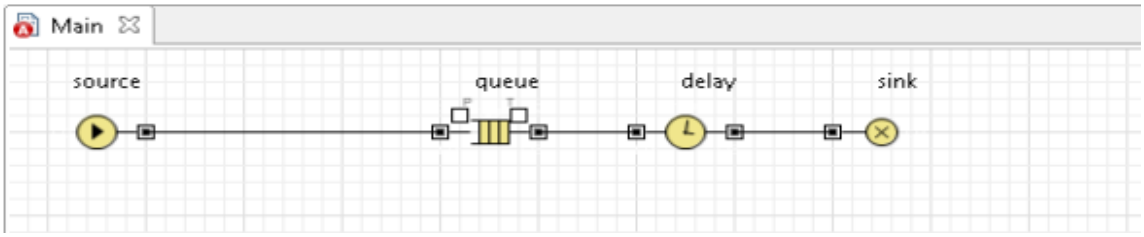


Рис.8.37. Стандартный шаблон дискретно-событийного моделирования

Source. Объект Основной библиотеки.

Создает заявки. Обычно используется в качестве начальной точки потока заявок.

Заявки могут быть либо базового для заявок класса Entity, либо любого класса пользователя, унаследованного от этого базового класса. Вы можете сконфигурировать объект так, чтобы он создавал заявки других типов, указав конструктор нужного класса в параметре Новая заявка, а также задать действие, которое должно выполняться перед тем, как новая заявка покинет объект, и связать с заявкой определенную фигуру анимации.

Сбор статистики использования ресурсов

Queue. Объект Основной библиотеки.

Объект Queue моделирует очередь заявок, ожидающих приема объектами, следующими за данным в потоковой диаграмме, или же хранилище заявок общего назначения. При необходимости Вы можете задать максимальное время ожидания заявки в очереди. Вы также можете программно извлекать заявки из любых позиций в очереди.

Delay. Объект Основной библиотеки.

Задерживает заявки на заданный период времени. Время задержки вычисляется динамически, может быть случайным, зависеть от текущей заявки или от каких-то других условий. Это время может, в частности, вычисляться

как длина фигуры, заданной в качестве фигуры анимации этого объекта, поделенной на "скорость" заявки.

Одновременно могут быть задержаны сразу несколько заявок (не более заданной вместимости объекта capacity). Заявки задерживаются независимо друг от друга – время задержки вычисляется отдельно для каждой заявки.

Sink. Объект Основной библиотеки.

Уничтожает поступившие заявки. Обычно используется в качестве конечной точки потока заявок. Для того, чтобы заявки удалялись из модели и уничтожались, нужно соединить выходной порт последнего блока процессной диаграммы с портом объекта Sink или Exit.

Для успешного уничтожения заявки необходимо выполнение трех условий:

Если заявка находится в сети, то она должна быть удалена из этой сети с помощью объекта NetworkExit.

Заявка не должна обладать ни одним ресурсом или сетевым ресурсом.

Если заявка содержит другие заявки, то они тоже должны удовлетворять вышеуказанным условиям.

Если какое-то из этих условий не выполняется, объект Sink выдает ошибку.

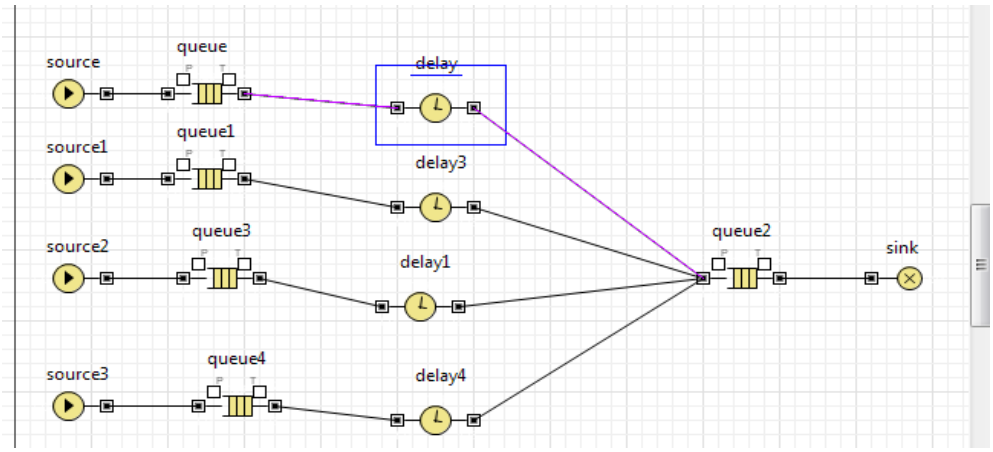


Рис.8.38. Модель многоканальной системы «Магазин»

AnyLogic предоставляет пользователю удобные средства для сбора статистики по работе блоков диаграммы процесса. Объекты Основной библиотеки самостоятельно осуществляют сбор основной статистики. Все, что Вам нужно сделать – это включить сбор статистики для объекта.

Поскольку мы уже сделали это для объектов delay, то теперь мы можем, например, просмотреть интересующую нас статистику (скажем, статистику занятости кассира и длины очереди) с помощью диаграмм.

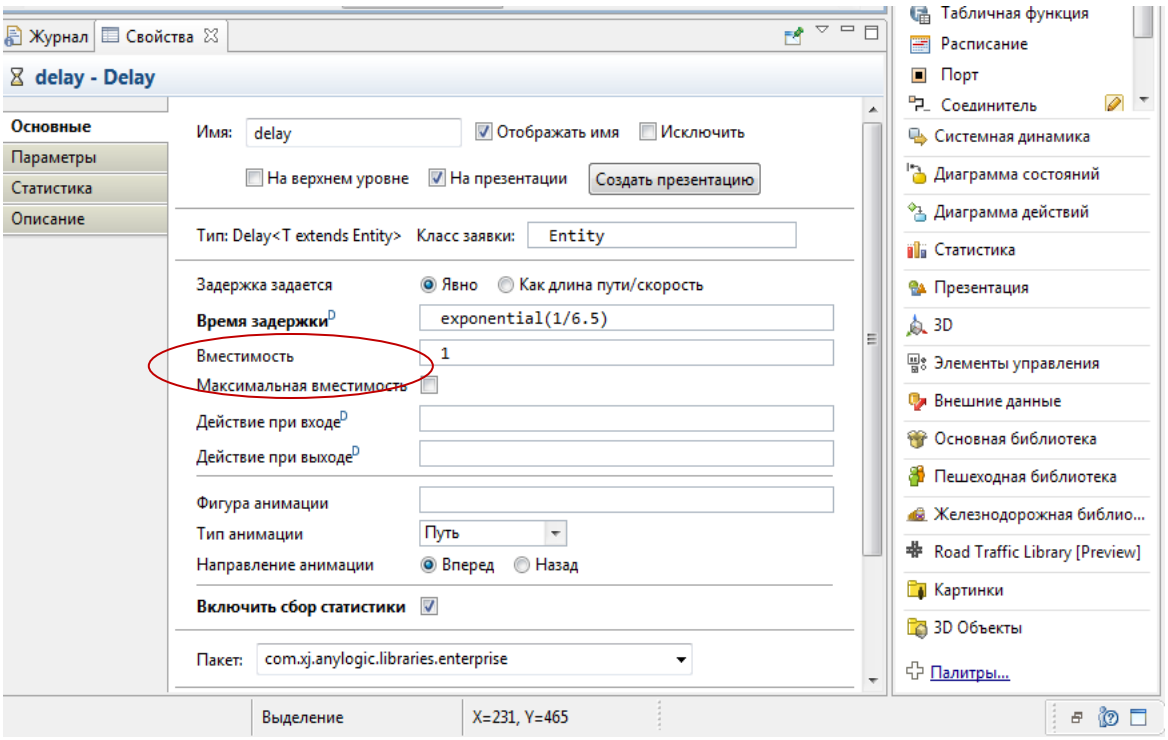


Рис.8.39. Организация работы кассира

Добавьте диаграмму для отображения средней занятости кассира

Откройте палитру Статистика. Эта палитра содержит элементы сбора данных и статистики, а также диаграммы для визуализации данных и результатов моделирования.

Перетащите элемент Столбиковая диаграмма из палитры Статистика на диаграмму класса и измените ее размер.

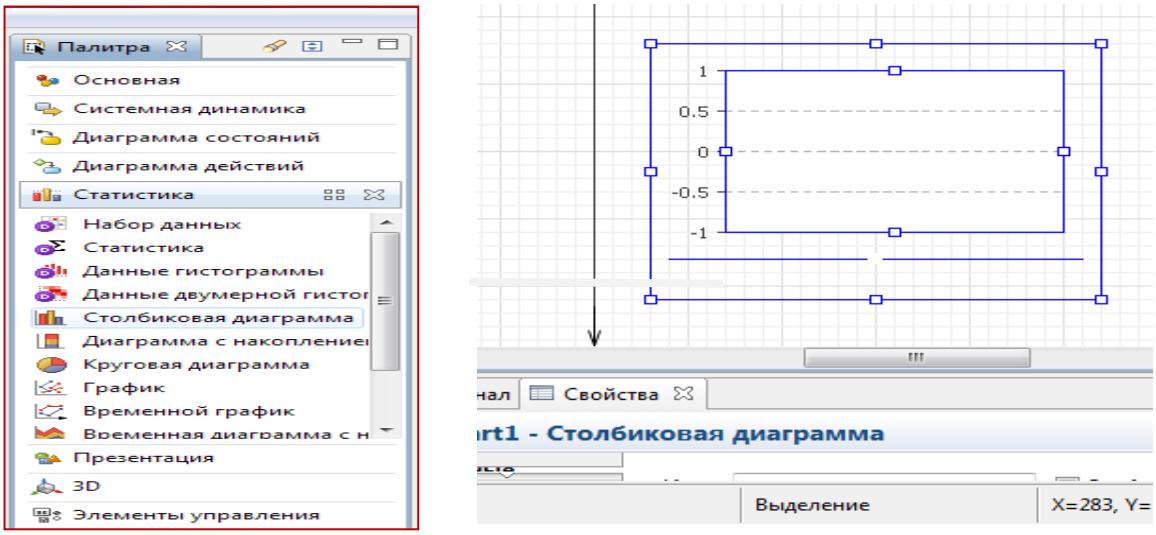


Рис.8.40. Подготовка диаграмм статистики работы магазина

Перейдите на страницу Основные панели Свойства. Щелкните мышью по кнопке Добавить элемент данных. При этом появится секция свойств того элемента данных, который будет отображаться на этой диаграмме.

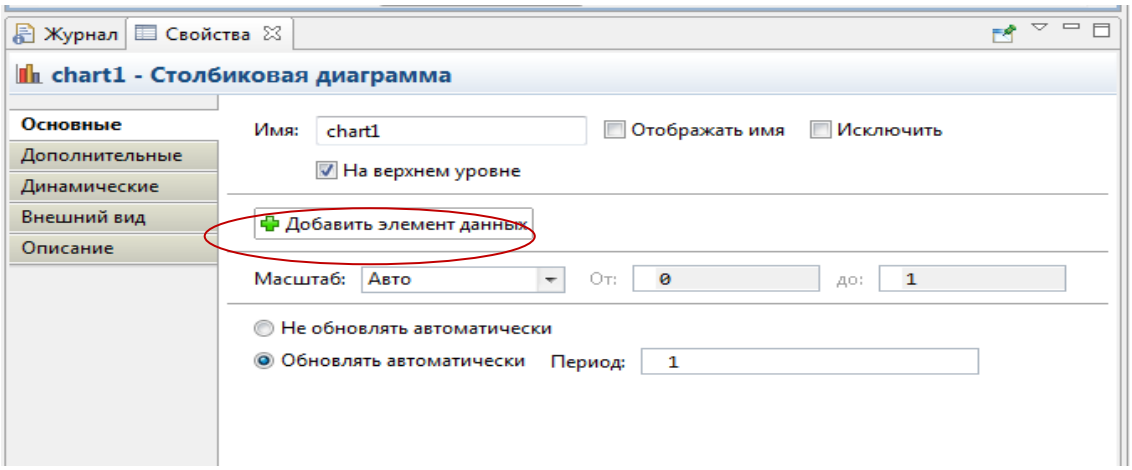


Рис.8.41. Организация подсчета покупателей

Введите delay.statsUtilization.mean() в поле Значение. Здесь delay – это имя нашего объекта Delay. У каждого объекта Delay есть встроенный набор данных statsUtilization, занимающийся сбором статистики использования этого объекта. Функция mean() возвращает среднее из всех измеренных этим набором данных значений. Вы можете использовать и другие методы сбора статистики, такие, как min() или max(). Полный список методов можно найти на странице документации этого класса набора данных: StatisticsContinuous .

Перейдите на страницу Внешний вид панели Свойства. Выберите первую опцию из набора кнопок Расположение, чтобы изменить расположение легенды относительно диаграммы (мы хотим, чтобы она отображалась справа).

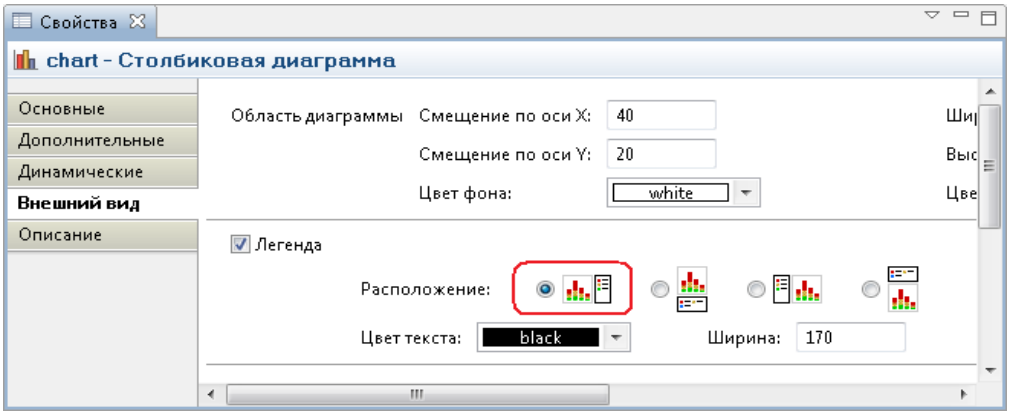


Рис.8.42. Выбор диаграммы

После чего добавляем еще 3 объекта delay1, delay2, delay3. Получаем вид

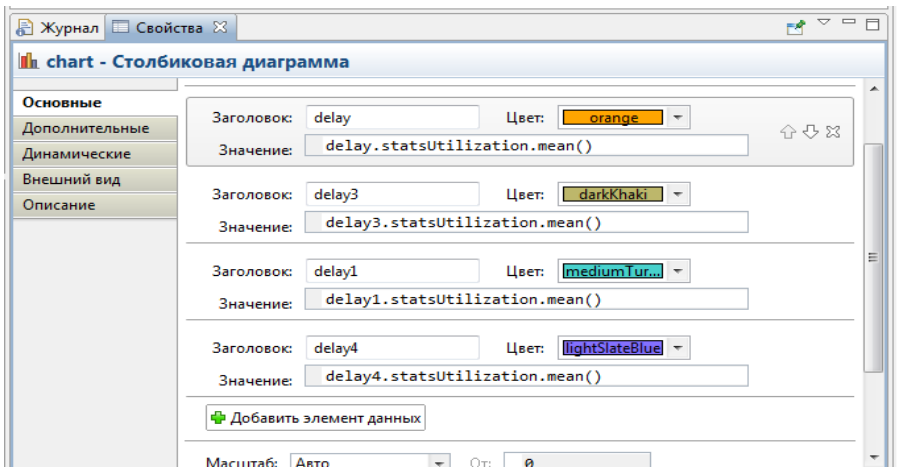


Рис.8.43. Выбор цвета

При запуске модели мы видим, как меняются столбиковые диаграммы.

Функция mean() возвращает среднее из всех измеренных этим набором данных значений. Теперь мы можем просмотреть интересующую нас статистику – статистику занятости кассира и длины очереди с помощью диаграммы.

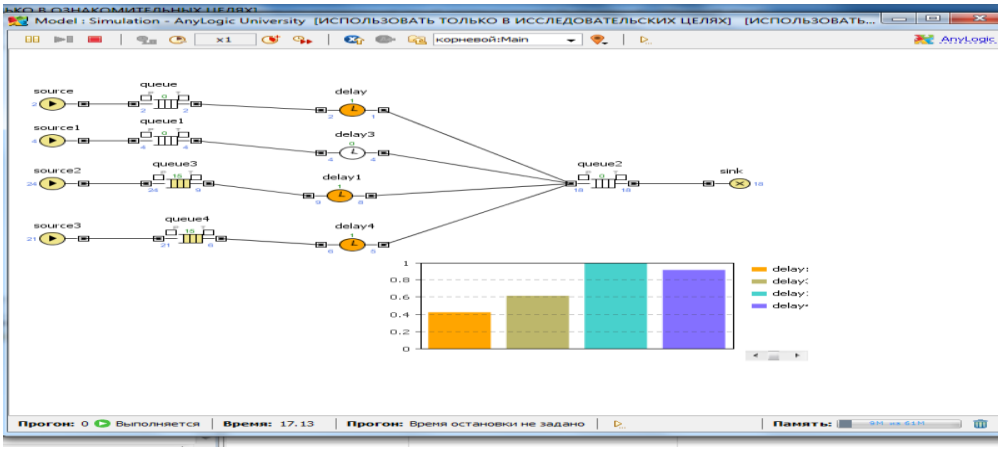


Рис.8.44. Модель и статистика работы магазина

8.2. Пешеходная динамика зрителей в кинотеатре

При эвакуации люди ведут себя очень сложно, стараясь как можно быстрее покинуть зону опасности. Встает вопрос организации пешеходных потоков в нештатных ситуациях. Для этого применяются соответствующие знаки, указывающие на аварийные выходы, кроме того, часто за эвакуацию отвечаю специальные люди. Моделирование чрезвычайных происшествий позволяет заранее предвидеть проблемы, возникающие при эвакуации людей, и в конечном счете спасти человеческие жизни. В данных целях построим имитационную модель посещения зрителей в кинотеатре с течки зрения их распределения согласно купленным билетам и их выходу по окончании представления.

Создадим имитационную модель под названием Кинотеатр в среде AnyLogic.

Далее добавим изображение плана кинотеатра. Для этого вначале откройте закладку Презентация панели Палитра. Чтобы открыть какую-либо закладку панели Палитра (именуемую в дальнейшем палитрой), нужно щелкнуть мышью по заголовку этой палитры.

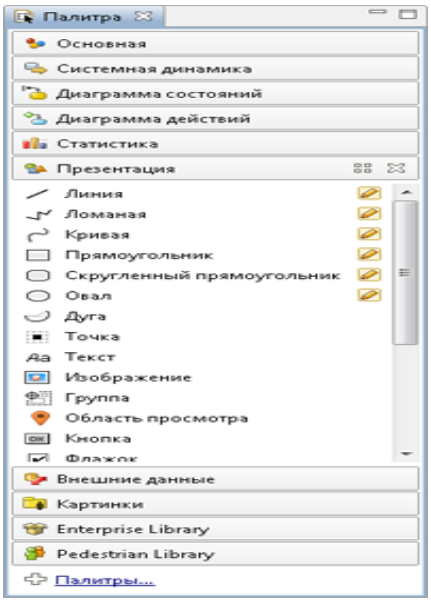


Рис. 8.45. Палитра Презентация

Палитра Презентация содержит элементы, используемые для рисования презентаций моделей: фигуры, с помощью которых Вы можете рисовать сложные презентации, а также элементы управления, с помощью которых Вы можете сделать презентации интерактивными.

Перетащите элемент Изображение из палитры Презентация на диаграмму класса активного объекта.

Задайте свойства изображения в панели Свойства. Щелкните мышью по кнопке Добавить и выберите файл изображения плана кинотеатра Вы увидите добавленное изображение в области предварительного просмотра на панели Свойства изображения (Рис. 8.46).

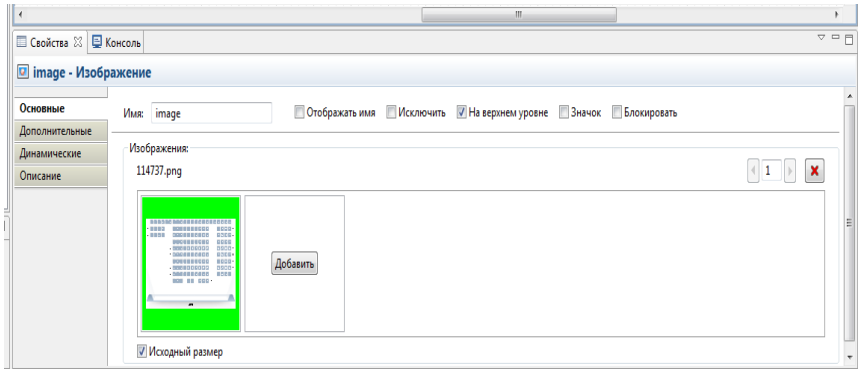


Рис. 8.46. Свойства изображения

Чтобы сохранить исходный размер изображения, установите флажок Исходный размер.

Далее мы нарисуем границу моделируемого пространства, играющую роль стен здания. Чтобы было легче рисовать, лучше отключить сетку и увеличить масштаб анимации с помощью соответствующих кнопок панели инструментов. С помощью инструмента Ломанная или Линия, находящиеся в палитре Презентация, нарисуйте границы, как показано на рисунке 8.47.

Теперь сгруппируем нарисованные линии. Для этого выделите нарисованные границы, нажмите ЛКМ и выберите Группировка – Создать группу. Переименуйте эту группу в свойствах группы на walls.

Группа фигур используется для группировки фигур презентации. С помощью динамических свойств группы фигур (X, Y, Поворот, и т.п.) можно передвигать группу фигур и поворачивать ее вокруг опорной точки. Сама опорная точка группы фигур на анимации не отображается.

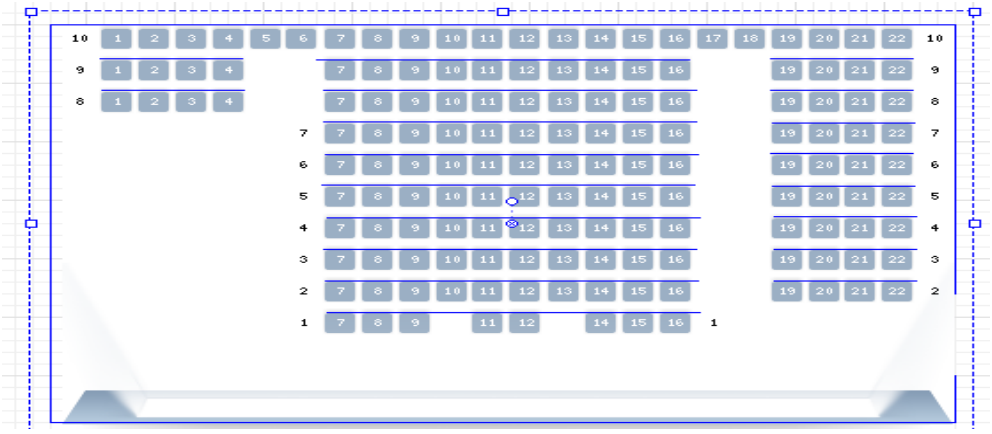


Рис. 8.47. Нарисованные границы

Чтобы добавить группу фигур на презентацию:

- перетащите элемент Группа из палитры Презентация в то место графического редактора, где Вы хотите поместить опорную точку группы.

Чтобы добавить фигуры в группу:

- выделите в графическом редакторе те фигуры, которые Вы хотите добавить в группу;
- если Вы хотите добавить фигуры в уже существующую группу, щелкните правой кнопкой мыши по выделенным фигурам, выберите Группировка – Добавить в существующую группу из контекстного меню и затем щелкните мышью по тому значку группы , в которую Вы хотите добавить выделенные фигуры;

- если же Вы хотите добавить фигуры в новую группу, то тогда щелкните правой кнопкой мыши по выделенным фигурам и выберите Группировка – Создать группу из контекстного меню.

Чтобы удалить фигуры из группы:

- выделите фигуры, которые Вы хотите удалить из группы;
- щелкните правой кнопкой мыши по выделенным фигурам и выберите Группировка – Удалить из группы из контекстного меню;
- также добавляем линию в оставшемся проеме и называем ее exit.

Разместим в рабочей области объекты из библиотеки Pedestrian Library:

- pedSource
- pedService
- pedGoTo
- pedSink
- pedConfiguration
- pedGround
- pedServices

Расположите и соедините их с помощью соединителя как показано на рисунке 8.48.

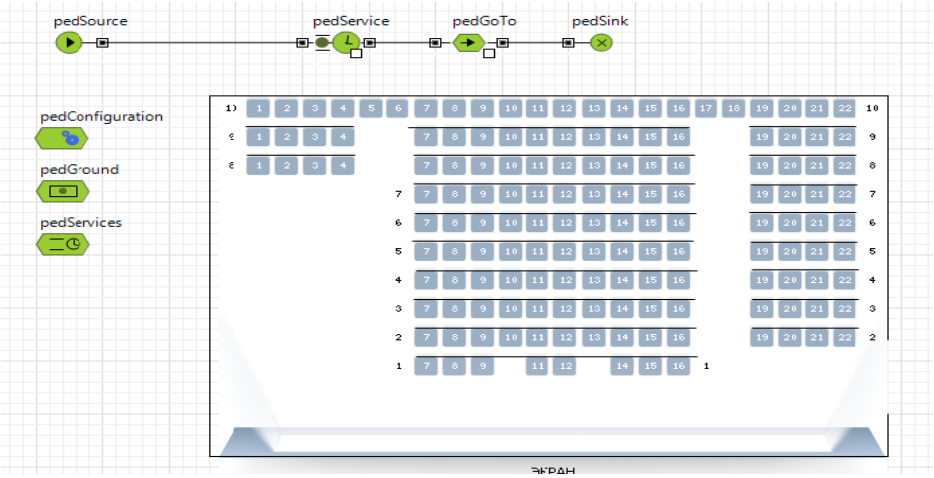


Рис. 8.48. Расположение и соединение объектов

PedSource – создает пешеходов. Обычно используется в качестве начальной точки блок-схемы, формализующей поток пешеходов. Создает пешеходов любых подклассов базового класса Ped через случайные промежутки времени.

PedServices – задает группу одинаковых физических объектов обслуживания (например, несколько турникетов или автоматов по продаже билетов). Объект позволяет задавать очереди и сервисы в любой комбинации и зада-

вать правила выбора сервисов: какую очередь выбрать, какой сервис выбрать, к какой очереди должен обращаться сервис, может ли сервис обслуживать несколько очередей и т.д. Используйте этот объект, чтобы создавать ряды сервисов и задавать их параметры по умолчанию.

PedGoTo – заставляет пешеходов перейти в заданное место моделируемого пространства, которое может быть задано линией или точкой. Переход будет считаться выполненным, когда пешеход пересечет заданную линию или достигнет заданной точки. Пешеходы будут искать путь к заданному транзиту в пределах текущего этажа.

PedSink – удаляет поступивших в объект пешеходов из моделируемой среды. Обычно объект используется в качестве конечной точки блок-схемы, формализующей поток пешеходов. PedSink автоматически ведет подсчет пешеходов. Также объект можно использовать для удаления любого пешехода. Если выходной порт объекта библиотеки Pedestrian Library ни к чему не подсоединен, то он не сможет произвести вывод пешеходов, поэтому, если Вам нужно удалять пешеходов, то обязательно поместите объект PedSink.

PedConfiguration – это главный объект Pedestrian Library. Объект PedConfiguration должен присутствовать на диаграмме в каждой модели AnyLogic, использующей Pedestrian Library, поскольку он поддерживает перемещение пешеходов и их анимацию. Он позволяет задавать общие параметры, относящиеся ко всем объектам Pedestrian Library, и настраивать модель для конкретной задачи с целью получения максимальной производительности.

PedGround – позволяет задавать двумерное пространство в моделируемой среде, представляющее собой «этаж», т.е. поверхность, по которой будут перемещаться пешеходы. Этажи могут быть ограничены какой-то стеной или быть неограниченными. Стены – это объекты, которые пешеходы не могут пересекать. Стены являются частью этажа, то есть одна стена не может быть использована несколькими этажами.

PedServices – задает группу одинаковых физических объектов обслуживания (например, несколько турникетов или автоматов по продаже билетов). Позволяет задавать очереди и сервисы в любой комбинации и задавать правила выбора сервисов: какую очередь выбрать, какой сервис выбрать, к какой очереди должен обращаться сервис, может ли сервис обслуживать несколько очередей и т.д. Используйте этот объект, чтобы создавать ряды сервисов и задавать их параметры по умолчанию. [7]

Соединитель – линия, соединяющая два порта или две переменные.

В свойствах объекта pedSource следует сделать следующие изменения:

- пешеходы прибывают согласно: Интенсивности;
- расписание, пешеходов в ед. времени: 150 / minute();
- этаж (PedGround): PedGround;
- место появления (линия, ломаная): exit.

В свойствах объектов pedService следует сделать следующие изменения:

- сервис (PedServices): PedServices

В свойствах объектов PedGoTo следует сделать следующие изменения:

- цель (точка, линия): exit

В свойствах объектов PedGround следует сделать следующие изменения:

- стены (группа, необязательный): walls

Теперь создадим сервисы обслуживания и очереди. Их можно сделать с помощью инструмента Ломаная или Линия. Сервисами будут места в кинотеатре, а очередями промежутки между рядами. Сделаем их как показано на рисунке 8.49.

Розовые линии – это сервисы, а черные между рядами – очереди.

Нужно сгруппировать сервисы и переименовать в mesta, затем сгруппировать очереди и переименовать в groupqueue. В их свойствах – Динамические следует установить видимость false.

В свойствах объектов PedServices следует сделать следующие изменения:

- сервисы (группа линий): mesta,
- время задержки: uniform(5.0*minute(), 5*minute()),
- правило выбора очереди: самая короткая очередь,
- обслуживается: самая длинная очередь.

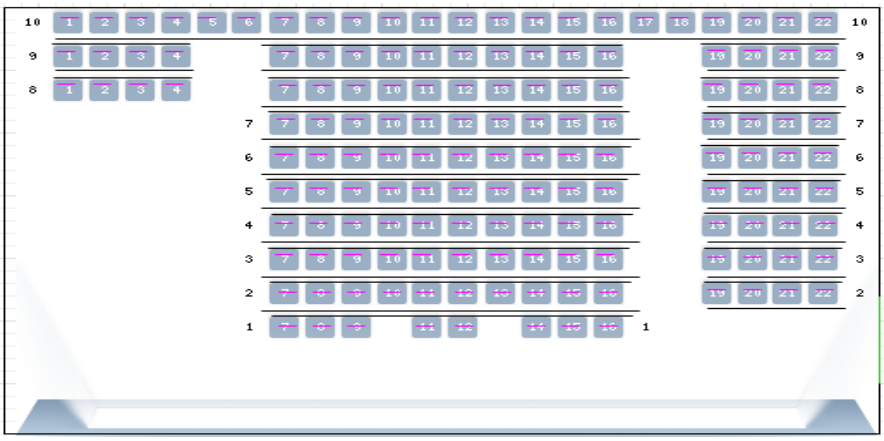


Рис. 8.49. Сервисы и очереди

Шаг 18. Изменение интенсивности потока зрителей.

В текущей модели интенсивность потока зрителей задана фиксированной величиной. Мы же хотим иметь возможность интерактивно изменять интенсивность во время выполнения модели, чтобы на тот или иной сеанс было разное количество зрителей. Для этого мы добавим специальный элемент управления.

Откроем палитру Элементы управления и перетащим элемент Бегунок из палитры на диаграмму класса Main (справа от плана кинотеатра).

В свойствах объекта Бегунок следует сделать следующие изменения:

- связать с: pedSource.rate,
- минимальное значение: 0,
- максимальное: 150 / minute().

Добавим текстовую метку для бегунка. Для этого добавим к бегунку текстовую метку с заголовком. Перетащим элемент Текст из палитры Презентация на диаграмму (выше бегунка). Введем текст, который мы хотим показать с помощью этой метки, в поле Текст. Например: Изменение интенсивности зрителей.

Шаг 13. Изменение окна презентации

Окно презентации автоматически отображается при запуске эксперимента модели. Изначально оно показывает презентацию запущенного эксперимента. В этот момент модель еще не создана и создан только сам эксперимент.

Панель управления окна презентации содержит секции с различными командами, позволяющими управлять выполнением модели, отрисовкой презентации, сохранять и загружать состояние модели из файла, осуществлять быструю навигацию по объектам модели, показывая в окне презентацию нужного нам объекта и т.д.

Изменим фон презентации путем добавления элемента изображения из палитры Презентация. Добавляем в свойствах изображения любую картинку по своему вкусу подходящего размера и задаем нужный размер.

Также можно изменить свойства кнопки запуска. Изменяя размер, цвет шрифта и т.п.

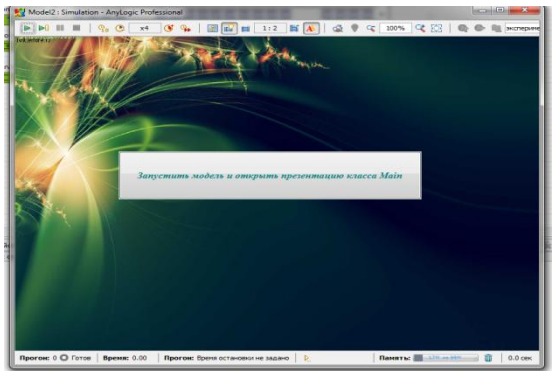


Рис. 8.50. Окно презентации

Запуск модели. Чтобы запустить модель, щелкаем по кнопке со стрелкой справа от кнопки панели инструментов Запуск и выбираем эксперимент, который мы хотим запустить, из выпадающего списка.

В окне презентации нажимаем на кнопку Запустить модель.

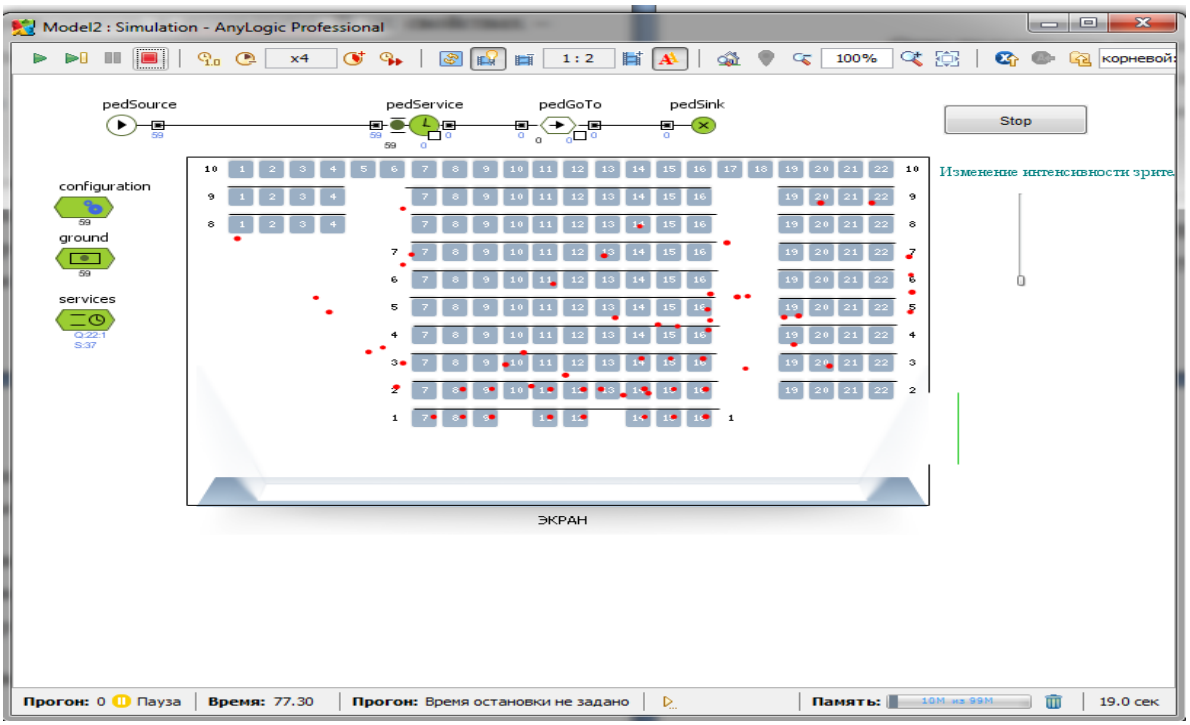


Рис. 8.51. Зрители занимают свои места

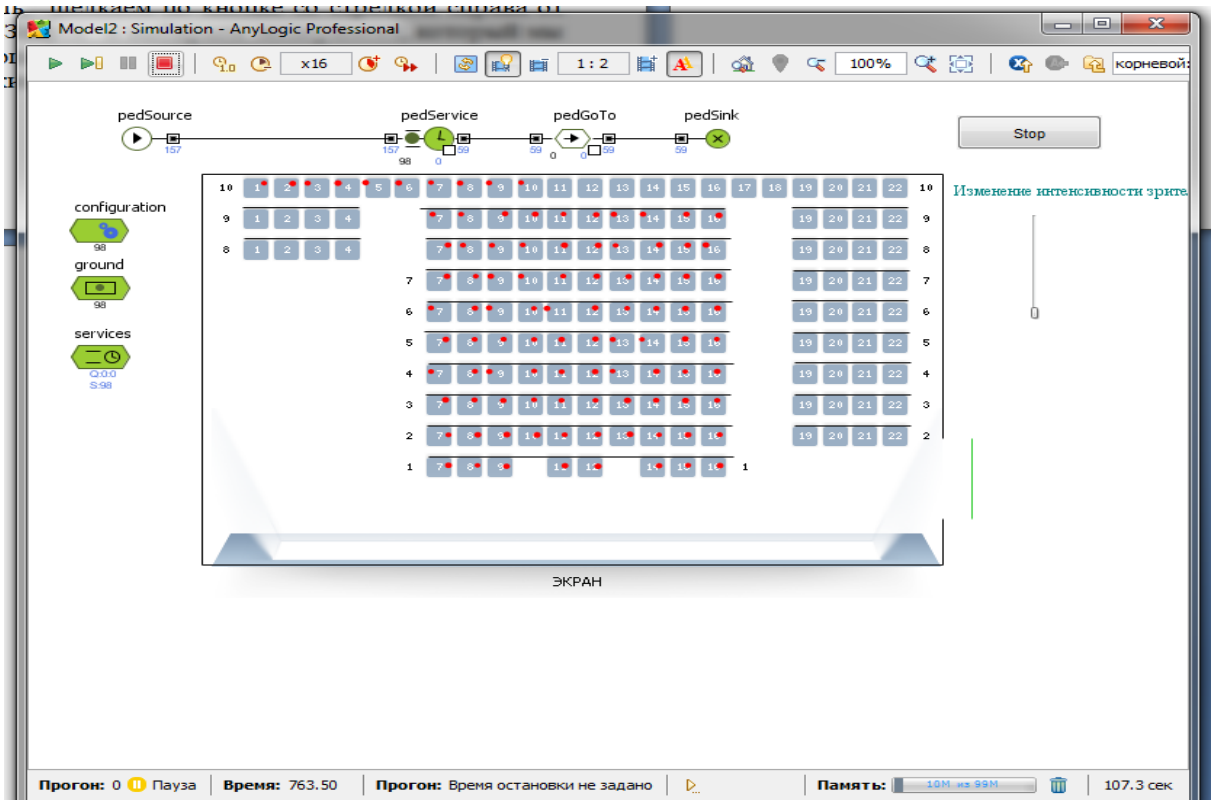


Рис. 8.52. Зрители заняли свои места

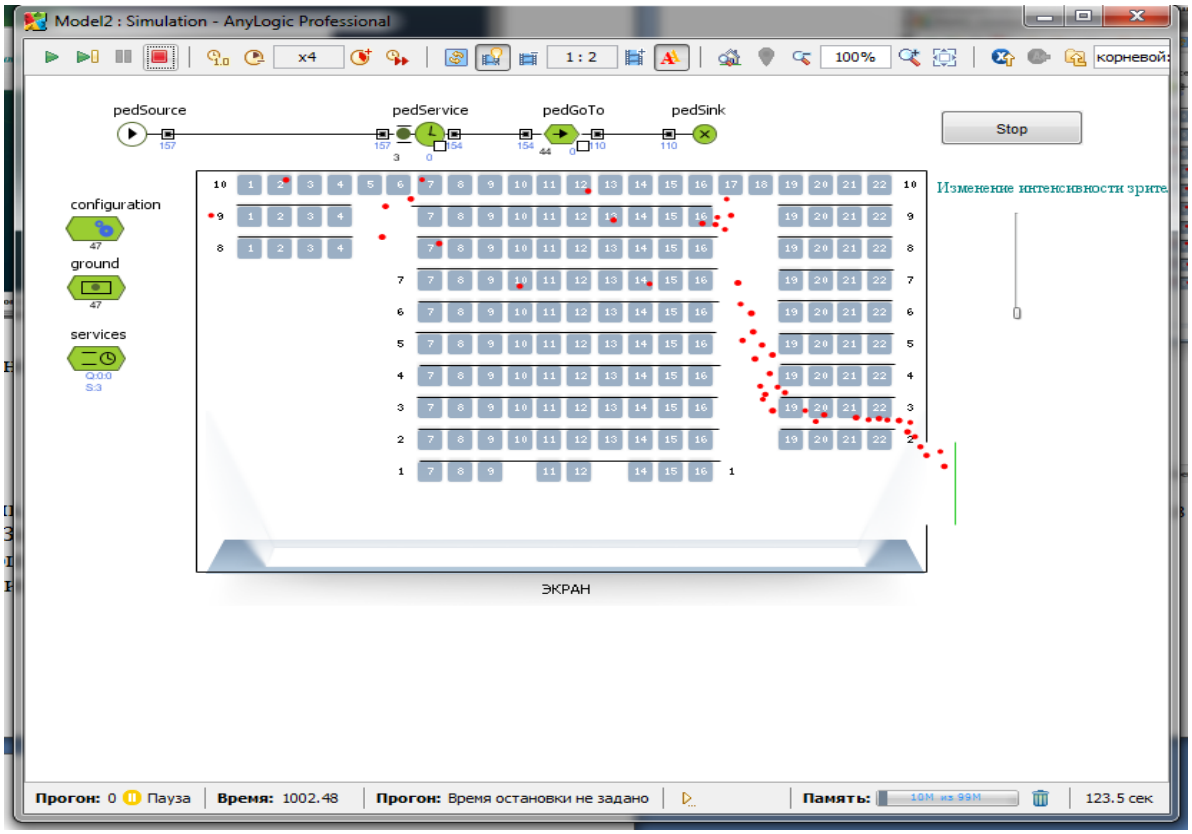


Рис. 8.53. Зрители идут на выход

Литература

1. AnyLogic Руководство пользователя [Электронный ресурс] – Режим доступа: http:// www.xjtek.com/ products / anylogic5 / usersmanual. pdf, свободный.
2. AnyLogic. Учебное пособие по системной динамике [Электронный ресурс] – Режим доступа: http:// www.xjtek.com , свободный.
3. Абрамова, Н.А.. Когнитивный анализ и управление развитием ситуаций: проблемы методологии, теории и практики [Текст] / Н.А.Абрамова, З.К.Авдеева // Проблемы управления. – 2008. – № 3. – С. 85-87.
4. Аверченков, В.И. Основы математического моделирования технических систем [Текст]: учеб. пособие / В.И. Аверченков, В.П. Федоров, М.Л. Хейфец. – М: Изд-во «Флинта», 2011. – 271с.
5. Акопов, А.И. Общий курс издательского дела [Текст]: учеб. пособие / А.И.Акопов. Под ред. проф. В. В. Тулупова. – Воронеж: Изд-во ВГУ, 2004. – 218с.
6. Алиев, Т.И. Основы моделирования дискретных систем [Текст] /Т.И. Алиев. – СПб: СПбГУИТМО, 2009. – 363 с.
7. Антонова, С. Г. Редакторская подготовка изданий [Текст] / С. Г. Антонова. – М.: Логос, 2004. – 261 с.
8. Арнольд, В.И. “Жесткие” и “мягкие” математические модели // Математическое моделирование социальных процессов. М., МГУ, 1998. С. 29-51.
9. Асанов, А.З. Введение в математическое моделирование динамических систем [Текст] / А.З. Асанов. – Казань: Изд-во КГУ. – 2007. – 205 с.
10. Бабенко, Л.П. Основы программной инженерии [Текст] / Л.П.Бабенко, Е.М. Лаврищева. – М.: Знание, 2001. – 269с.
11. Басс, Л. Архитектура программного обеспечения на практике [Текст] / Л.Басс, П. Клементс, Р.Кацман, 2-е изд. – СПб.: Питер, 2006. – 575 с.
12. Блок, А.А. Курс лекций по дисциплине «Моделирование систем» [Текст]: учеб. пособие / А.А. Блок. – Архангельск: Изд-во Арханг. гос. техн. ун-та, 2009. – 96 с.
13. Бобров, В.И. Принципы построения цифровой имитационной многофакторной модели процесса функционирования полиграфических автоматизированных систем и машин [Текст] / В. И. Бобров // Известия ВУЗов. Проблемы полиграфии и издательского дела. – 2000. – № 1-2.

14. Боголюбов, А.Н. Основы математического моделирования [Текст]: учеб. пособие/ А.Н.Боголюбов. - М.: МГУ, 2003. – 137с.
15. Боев, В.Д. Компьютерное моделирование [Текст]: пособие для курсового и дипломного проектирования / В.Д.Боев, Д.И.Кирик, Р.П. Сыпченко. – СПб.: ВАС, 2011. — 348 с.
16. Боев, В.Д. Компьютерное моделирование [Текст] / В.Д.Боев, Р.П. Сыпченко. – М.: ИНТУИТ.РУ, 2010. – 349 с.
17. Боев, В.Д. Моделирование систем. Инструментальные средства GPSS World [Текст]: учеб. пособие / В.Д.Боев. – СПб.: BHV-Петербург, 2004.– 368 с.
18. Болотова, Л.С. Системы искусственного интеллекта: модели и технологии, основанные на знаниях [Текст]: учебник. – М.: Финансы и статистика, 2012. – 664 с.
19. Бордовский, Г.А. Физические основы математического моделирования. [Текст] / Г.А.Бордовский, А.С.Кондратьев, А.Д.Р.Чоудори. – М.: Издательский центр «Академия», 2005. – 320с.
20. Борщев, А. Как строить красивые и полезные модели сложных систем [Текст]: материалы конф. «Имитационное Моделирование. Теория и Практика» ИММОД- 2013. – Казань: Изд-во «Фэн» АН РТ, 2013. – Т.1. – с.21-34.
21. Брауде, Э. Технология разработки программного обеспечения [Текст] / Э Брауде. – СПб.: Питер, 2004. – 655 с.
22. Бусленко, Н.П. Метод статистического моделирования [Текст] / Н.П. Бусленко. – М.: Статистика, 1970. – 112 с.
23. Бусленко, Н.П. Моделирование сложных систем [Текст] / Н.П Бусленко. – М.: Наука, 1978. – 400с.
24. Буч, Г. Объектно-ориентированное проектирование с примерами применения [Текст] / Г.Буч. Пер. с англ. – М.: Конкорд, 1992. – 519 с.
25. Васильев В.И. Интеллектуальные системы управления. Теория и практика [Текст] / В.И Васильев, Б.Г.Ильясов. – М.: Изд-во УРСС, 2009. – 392 с.
26. Введение в математическое моделирование [Текст]: учеб. пособие / Под ред. П.В.Трусова. – Логос, 2004. – 440с.
27. Введение в математическое моделирование транспортных потоков [Текст] / А.Гасников, С.Кленов, Е.Нурминский, Я.Холодов, Н.Шамрай. –М.: Изд-во МЦНМО, 2013. - 427 с..

28. Введение в математическое моделирование. Интернет–Университет Информационных Технологий [Электронный ресурс] / Режим доступа: http://www.intuit.ru/studies/courses/1020/188/info, (2014, 21фев.).
29. Вендров, А.М. CASE-технологии. Современные методы и средства проектирования информационных систем [Текст] / А.М. Вендров. – М.: Финансы и статистика, 1998. – 176 с.
30. Венчковский, Л.Б. Разработка сложных программных изделий [Текст]: учеб. пособие / Л.Б.Венчковский. – М.: ЗАО "Финстатинформ", 1999. – 109с.
31. Винокурова, О.А. Методы и средства переработки информации в допечатных системах [Текст]: монография / О.А.Винокурова, М.В.Ефимов, Ю.Н.Самарин, М.А. Синяк. Моск. гос. ун-т печати. – М.: МГУП.
32. Влацкая, И.В. Моделирование систем массового обслуживания [Текст]: методические указания к расчетно-графическим работам / И.В.Влацкая, О.А.Татжибаева. – Оренбург: Изд-во ОГУ, 2005. – 20 с.
33. Волкова, В.Н. Основы теории систем и системного анализа [Текст] / В.Н.Волкова, А.А.Денисов. – СПб.: Санкт-Петербургский гос.тех. ун-т, 1997. – 510с.
34. Ганиева, Н.М. Проектирование полиграфического производства [Текст]: учеб. пособие / Н.М.Ганиева. – Омск : Изд-во ОмГТУ, 2010. – 116 с.
35. Гасов, В.М. Программные средства допечатных процессов [Текст]: учеб. пособие / В.М. Гасов, А.М.Цыганенко. В 3 кн. – М.: МГУП, 2000.
36. Гейн, К. Системный структурный анализ: средства и методы [Текст] / К.Гейн, Т.Сарсон. – М.: Эйтекс, 1992.
37. Гнеденко, Б.Б. Введение в теорию массового обслуживания [Текст] / Б.Б.Гнеденко, И.Н.Коваленко. – М., Наука, 1966. – 255 с.
38. Горбачев, А. А. Определение затрат времени на цифровую обработку изображений [Текст] / А. А. Горбачев, Ю. Н. Самарин // Управление и информатика в полиграфических системах: Межвед. сб. научн. тр. — М.: МГУП, 2003.
39. Горбачев, А.А. Методика определения производительности системы допечатной подготовки. [Текст]: монография / А.А.Горбачев, Ю.Н.Самарин. – М.: Август-Принт, 2006.
40. Горелова, Г.В. Исследование слабоструктурированных проблем социально-экономических систем: когнитивный подход. [Текст]/

Г.В.Горелова, Е.Н. Захарова, С.А.Радченко. – Ростов-на-Дону: Изд-во РГУ, 2006. – 332 с.
41. Горелова, Г.В. Когнитивный анализ и моделирование устойчивого развития социально-экономических систем [Текст]/ Г.В.Горелова, Е.Н.Захарова, Л.А Гинис. – Ростов на Дону: Изд-во РГУ, 2005. – 288 с.;
42. Горелова, Г.В. Региональная система образования, методология комплексных исследований [Текст] / Г.В.Горелова, Н.Х.Джаримов. – Краснодар: Изд-во Печатный двор, 2002. - 360с.
43. ГОСТ 34.602–89. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы.
44. Гультяев, А.К. Имитационное моделирование в среде Windows. MATLAB 5.3 [Текст]: практическое пособие / А.К.Гультяев. – СПб.: КОРОНА - ПРИНТ, 2001. – 400с.
45. Гуц, А.К. Глобальная этносоциология [Текст]: учеб. пособие. – Омск: ОмГУ, 1997. – 212 с.
46. Девятков, В.В. Методология и технология имитационных исследований сложных систем: современное состояние и перспективы развития [Текст]: монография / В.В.Девятков. – М.: Вузовский учебник: ИНФРА–М, 2013. – 448с.
47. Долгова, Т.А. Моделирование технологических процессов полиграфического производства [Текст]: лаб. практикум по одноименному курсу для студентов специальности 1-47 02 01 «Технология полиграфических производств» / сост. Т.А.Долгова, Т.В.Анкуд. – Минск: БГТУ, 2005.— 58 с.
48. Дьяконов, В.П. Maple 6 [Текст]: учеб. курс / В.П.Дьяконов. – СПб.: Питер, 2001. – 608с.
49. Дьяконов, В.П. Mathematica 4 [Текст]: учеб. курс / В.П.Дьяконов. – СПб.: Питер, 2001. – 654с.
50. Дьяконов, В.П. MATLAB 6/6.1/6.1. Simulink 6.5 в математике и моделировании. Полное руководство пользователя [Текст] / В.П.Дьяконов. – М.: Солон-Пресс, 2003.
51. Дьяконов, В.П. MATLAB 6: [Текст]: учеб. курс / В.П.Дьяконов. – СПб.: Питер, 2001.
52. Дьяконов, В.П. VisSim + MathCAD + MATLAB. Визуальное математическое моделирование [Текст] / В.П. Дьяконов. – М: СОЛОН-Пресс, 2004. – 384с.

53. Емельянов, А.А. Имитационное моделирование экономических процессов [Текст] / А.А.Емельянов, Е.А.Власова, Е.А. Дума. – М.: Финансы и статистика, 2002. – 364 с.
54. Ермаков, С.М. Методы Монте-Карло и смежные вопросы [Текст] / С.М. Ермаков. – М.: Наука, 1971.
55. Ермаков, С.М. Статистическое моделирование [Текст] / С. М. Ермаков, Г. А. Михайлов. – 2-е изд., доп. – М.: Наука, 1982. – 296 с.
56. Ефимов, М.В. Автоматизированное управление полиграфическим производством [Текст] / М. В. Ефимов. – М.: Изд-во МГУП "Мир Книги", 1998. – 416 с.
57. Задорожный, В.Н. Имитационное моделирование [Текст]:: учеб. пособие/ В. Н. Задорожный. – Омск: Изд-во ОмГТУ, 1999. – 151 с.
58. Задорожный, В.Н. Имитационное и статистическое моделирование [Текст]: учеб. пособие / В.Н.Задорожный. – Омск: Изд-во ОмГТУ, 2013. – 136 с.
59. Замятина, О.М. Моделирование систем [Текст]: учеб. пособие / О.М Замятина. – Томск: Изд-во ТПУ, 2009. – 204 с.
60. Замятина, Е.Б. Современные теории имитационного моделирования (Специальный курс для магистров второго курса). Пермь: ПГУ, 2007, 119 с.
61. Иванова, А. Е. Автоматизация учета и прохождения заказов в цехе допечатной подготовки издания [Текст] / А.Е.Иванова, Т.В.Козачук // Вестник МГУП. – 2005. – № 12.
62. Ивашкин, Ю.А. Агентные технологии и мультиагентное моделирование систем [Текст]: учеб. пособие / Ю.А.Ивашкин. – М.: МФТИ, 2013. – 268с.
63. Ивченко, Г.И. Теория массового обслуживания [Текст] / Г. И. Ивченко, В. А. Каштанов, И. Н. Коваленко. – М.: Высш. школа, 1982. – 256 с.
64. Имитационное моделирование для науки и бизнеса [Электронный ресурс] // Применение имитационного моделирования. – Режим доступа: http://www.xjtek.ru/anylogic/articles/4/, свободный.
65. Имитационное моделирование систем бизнеса [Электронный ресурс] // Применение имитационного моделирования. – Режим доступа: http://www.gpss.ru/, свободный.
66. Карпов, Ю.Г. Имитационное моделирование систем. Введение в моделирование с AnyLogic 5 [Текст] / Ю.Г.Карпов. – СПб.: БХВ-Петербург, 2005. – 400 с.

67. Кельтон, В. Имитационное моделирование. Классика CS [Текст] / В. Кельтон, А. Лоу. 3-е изд. Пер. с англ. – СПб.: Издательская группа BNV, 2004. – 847 с.
68. Киндлер, Е. Языки моделирования [Текст] / Е. Киндлер. – М. : Энергия, 1985. – 288 с.
69. Киселева, М.В. Имитационное моделирование систем в среде AnyLogic [Текст]: учеб.-метод. пособие / М.В.Киселёва. – Екатеринбург: УГТУ-УПИ, 2009. – 88 с.
70. Клыков, Ю.И. Ситуационное управление большими системами [Текст] / Ю.И Клыков. – М.: Энергия, 1974. – 135 с.
71. Кобелев, Н.Б. Имитационное моделирование [Текст]: учеб. пособие / Н.Б.Кобелев, В.В.Девятков, В.А.Половников. – М.: КУРС: ИНФРА-М, 2013. – 368с.
72. Кобелев, Н.Б. Основы имитационного моделирования сложных экономических систем [Текст]: учеб. пособие/ Н.Б.Кобелев. – М.: Дело, 2003. – 336 с.
73. Колесов, Ю.Б. Моделирование систем. Практикум по компьютерному моделированию [Текст] / Ю.Б.Колесов, Ю.Б.Сенченков. – СПб.: БХВ - Петербург, 2007. – 352 с.
74. Колесов, Ю.Б. Моделирование систем. Динамические и гибридные системы [Текст]: учеб. пособие / Ю.Б.Колесов, Ю.Б.Сенченков. — СПб.: БХВ-Петербург, 2012. – 224с.
75. Колесов, Ю.Б. Моделирование систем. Объектно-ориентированный подход [Текст]: учеб. пособие / Ю.Б.Колесов, Ю.Б.Сенченков. — СПб.: БХВ-Петербург, 2006. –192с.
76. Кондрашова, В. К. Экономика полиграфического предприятия [Текст] : учеб. пособие для вузов / В. К. Кондрашова, О. Г. Исаева. – М.: Изд-во МГУП, 2005. – 317 с.
77. Константайн, Л. Разработка программного обеспечения [Текст] / Л. Константайн, Л. Локвуд. – СПб.: Питер, 2004. – 592 с.
78. Кофман, А. Массовое обслуживание. Теория и приложения [Текст] / А. Кофман, Р. Крюон. – М.: Мир, 1965. – 304 с.
79. Кудрявцев, Е.М. GPSS World. Основы имитационного моделирования различных систем [Текст] / Е. М. Кудрявцев. – М.: ДМК Пресс, 2004. – 320 с.
80. Лаврушина Е.Г., Слугина Н.Л. Теория систем и системный анализ: учебный комплекс. – Владивосток: Изд-во ВГУЭС, 2007. – 168 с.

81. Лазарев, Ю.Ф. Моделирование процессов и систем в MATLAB [Текст]: учеб. курс / Ю.Ф. Лазарев. – Питер, БХВ - Петербург, 2005. – 512 с.
82. Леоненко, А. В. Самоучитель UML [Текст] / А. В. Леоненко – 2-е изд, перераб. и доп. – СПб.: БХВ-Петербург, 2004. – 432с.
83. Липаев, В.В. Отладка сложных программ [Текст] / В.В.Липаев. – М.: Энергоатомиздат, 1993. – 296 с.
84. Лычкина, Н.Н. Имитационное моделирование экономических процессов [Текст]: учеб. пособие / Н.Н. Лычкина. – М.:, ИНФРА-М, 2012. – 254 с.
85. Максимей, И.В. Имитационное моделирование на ЭВМ [Текст] / И.В. Максимей. – М.: Радио и Связь, 1988. – 255 с.
86. Максимов В.И. Когнитивные технологии – от незнания к пониманию / Сб. трудов 1-й Международной конференции «Когнитивный анализ и управление развитием ситуаций», (CASC'2001). – М.: ИПУ РАН, 2001. – Т.1. – С. 4-18.
87. Маликов, Р.Ф. Основы математического моделирования [Текст]: учеб. пособие / Р.Ф.Маликов. – М: Изд-во «Горячая линия – Телеком», 2010. – 348с.
88. Маликов, Р.Ф. Основы разработки компьютерных моделей сложных систем [Текст]: учеб. пособие / Р.Ф.Маликов. – Уфа: Изд-во БГПУ, 2012. – 256 с.
89. Маликов, Р.Ф. Основы систем компьютерного моделирования [Текст]: учеб. пособие / Р.Ф.Маликов. – Уфа: Изд-во БГПУ, 2008. – 279с.
90. Масалович, А. И. Моделирование и анализ поведения бизнес-процессов [Текст]: конспект лекций / А. И. Масалович, Ю. А. Шебеко — М.: Тора-Инфо-Центр, 2002. – 219с.
91. Математические модели социальных систем [Текст] / А.К.Гуц, В.В. Коробицын, А.А.Лаптев, Л.А.Паутова, Ю.В.Фролова. – Омск: ОмГУ, 2000. – 256с.
92. Математическое моделирование [Текст] / под ред. Дж. Эндрюса, Р. Мак-Лоуна, пер. с англ. 2006. – с.416.
93. Межотраслевые нормы времени и выработки на процессы полиграфического производства. – М.: ГП НИЦ Экономика, 1997. – 447 с.
94. Мезенцев, К.Н. Моделирование систем в среде AnyLogic 6.4.1 [Текст]: учеб. пособие / К.Н.Мезенцев. Под редакцией Заслуженного деятеля науки РФ, д.т.н., профессора А.Б.Николаева. Часть 1, 2. – М.: МАДИ. – 2011. – 103с.

95. Месарович, М. Общая теория систем: математические основы [Текст] / М. Месарович, Я. Такахара. – М.: Мир, 1978. – 311с.
96. Метсон, Т.М. Организация движения [Текст] / Т.М.Метсон, У.С.Смит, Ф. В.Хард, пер. с англ. – М.: 1960.
97. Микони, С.В. Основы системного анализа [Текст]: учеб. пособие / С.В. Микони, В.А.Ходаковский. – СПб.: Петербургский государственный университет путей сообщения, 2011. – 143с.
98. Михайлов, Г.А. Численное статистическое моделирование. Методы Монте-Карло [Текст]: учеб. Пособие / Г.А.Михайлов, А.В.Войтишек. – М.: Издательский центр «Академия», 2006. – 368с.
99. Модели систем автоматического управления и их элементов [Текст]: учеб. пособие / С.Т.Кусимов, Б.Г.Ильясов, В.И.Васильев и др. – М.: Машиностроение, 2003. – 214с.
100. Моделирование и прогнозирование мировой динамики [Текст] / В.А.Садовничий, А.А.Акаев, А.В.Коротаев, С.Ю.Малков. Научный совет по Программе фунд. исслед. Президиума Российской академии наук «Экономика и социология знания». – М.: ИСПИ РАН, 2012. – (Экономика и социология знания). – 359 с.
101. Моделирование социальных процессов [Текст]: учеб. пособие / Н.П. Тихомиров, В.Я.Райцин, Ю.Н. Гаврилец, Ю.Д.Спиридонов. – М.: Изд-во Рос. экон. акад., 1993. – 304 с.
102. Осоргин, А. Е. AnyLogic 6 [Текст]: лабораторный практикум / А.Е. Осоргин. – Самара: ПГК, 2011. – 100 с.
103. Павловский, Ю. Н. Имитационное моделирование [Текст] / Ю.Н. Павловский, Н.В. Белотелов, Ю.И.Бродский. – М.: Изд-во «Академия», 2008. – 236 с.
104. Павловский, Ю. Н. Имитационные модели и системы [Текст] / Ю.Н. Павловский. – М.: ФАЗИС: ВЦ РАН, 2000. – 166 с.
105. Петров, А.Е. Справочное руководство по Enterprise Library [Текст]: учеб. пособие / А.Е.Петров. – Москва: Изд-во «Юнити», 2005. – 120с.
106. Петухов, О.А. Моделирование: системное, имитационное, аналитическое [Текст]: учеб. пособие / О. А. Петухов, А. В. Морозов, Е. О. Петухова. – 2-е изд., испр. и доп. – СПб.: Изд-во СЗТУ, 2008. – 288 с.
107. Плотинский, Ю.М. Модели социальных процессов [Текст]: учеб. пособие / Ю.М.Плотинский. – Изд. 2-е, перераб. и доп. – М.: Логос, 2001. – 296 с.
108. Плотников, А.М. Анализ современного состояния и тенденции развития имитационного моделирования в Российской Федерации (по материа-

лам конференции «Имитационное моделирование. Теория и практика» (ИММОД)) [Текст] / А. М. Плотников, Ю. И. Рыжиков, Б. В. Соколов, Р.М.Юсупов // Труды СПИИРАН, вып. № 2(25). – Санкт-Петербург, 2013. - С. 42 -112.

109. Полянский, Н. Н. Основы полиграфического производства [Текст] / Н.Н. Полянский. – М.: Книга, 1991. – С. 352.
110. Поспелов, Д.А. Ситуационное управление: теория и практика [Текст] / Д.А.Поспелов. – М.: Наука, 1986. – 284 с.
111. Примеры имитационных моделей [Электронный ресурс] // примеры имитационных моделей, построенных в среде AnyLogic. – Режим доступа: http://headwire.narod.ru/, www.runthemodel.com, свободный.
112. Прицкер А. Введение в имитационное моделирование и язык СДАМ II [Текст] / А.Прицкер. – М.: Мир, 1987.
113. Робертс, Ф.С. Дискретные математические модели с приложениями к социальным, биологическим и экологическим задачам [Текст] / Ф.С. Робертс. – М.: Наука, 1986. – 496с.
114. Рожков, М.И. Разработка имитационных моделей управления запасами в цепях поставок [Текст] / М.И. Рожков. – М.:, 2011.
115. Рыжиков, Ю. И. Имитационное моделирование. Теория и технологии [Текст] / Ю. И. Рыжиков. – СПб. : КОРОНАпринт; М. : Альтекс-А, 2004. – 384 с.
116. Саати Т.Л. Математические модели конфликтных ситуаций [Текст] / Т. Л. Саати. – М.: Сов. радио, 1977. – 306с.
117. Саати, Т. Л. Элементы теории массового обслуживания и ее приложения [Текст] / Т. Л. Саати. – М.: Сов. радио, 1966. – 365 с.
118. Самарин, Ю. Н. Автоматизация учета затрат времени и управление обработкой информации в системах допечатной подготовки [Текст] / Ю. Н. Самарин, А. А. Горбачев // КомпьюАрт-2006.
119. Самарский, А.А. Математическое моделирование: Идеи. Методы. Примеры [Текст] / А.А.Самарский, А.П. Михайлов. – М: Наука, 1999. – 320 с.
120. Семененко, М.Г. Введение в математическое моделирование [Текст] / М.Г. Семененко. – М: Солон-Р, 2002. – 112с.
121. Сидоренко, В.Н. Системная динамика. – М.: Эконом. факультет МГУ, ТЕИС, 1998. – 200 с.
122. Смирнов, Э. А. Основы теории организации [Текст] / Э.А. Смирнов. – М.: Изд-во ЮНИТИ, 2008. –463 с.

123. Соболь, И. М. Численные методы Монте-Карло [Текст] / И. М. Соболь. – М.: Наука, 1973. – 212 с.
124. Советов, Б.Я. Моделирование систем [Текст]: учебник для бакалавров / Б.Я. Советов, С. А. Яковлев. – М.: Изд-во Юрайт, 2012. – 343с.
125. Современное состояние полиграфической промышленности России: Информационное совещание в МИТР РФ [Текст] // Полиграфия, 2001.- №6.
126. Современные задачи управления производственным предприятием - как их решать? [Электронный ресурс]. Режим доступа http://petrovmv. narod.ru /Articles/problems.htm , свободный.
127. Справочная система Anylogic "Presentation and Animation: Working with Shapes, Groups, Colors" [Электронный ресурс]. Режим доступа: http:// www. xjtek. com/ files/ book/ Presentation _and _animation-working _with _shapes _groups _colors.pdf.
128. Страментов, А. Е. Городское движение [Текст] / А.Е.Страментов, М.С. Фишельсон , 2 изд. – М.: 1965.
129. Строгалев, В.П. Имитационное моделирование [Текст]: учеб. пособие / В.П.Строгалев, И.О.Толкачева. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2008. – 280 с.
130. Сценарный анализ динамики поведения социально-экономических систем [Текст]: научное издание / В.В. Кульба, Д.А. Кононов, С.С. Ковалевский и др. – М.:ИПУРАН, 2002. – 122 с.
131. Тарасевич, Ю.Ю. Математическое и компьютерное моделирование. Вводный курс [Текст]: учеб. пособие / Ю.Ю.Тарасевич. – Изд.4-е, испр. – М.: Едиториал УРСС, 2004. – 152с.
132. Тарасик, В.П. Математическое моделирование технических систем [Текст]: учеб. пособие / В.П.Тарасик. – 2-е изд. – Мн.: ДизайнПРО, 2004. – 640с.
133. Толуев, Ю.И. Моделирование и симуляция логистических систем [Текст]: курс лекций для высших технических учебных заведений / Ю.И.Толуев, С.И.Планковский. – Киев: «Миллениум», 2009. – 85с.
134. Томашевский, В. Н. Имитационное моделирование в среде GPSS [Текст] / В. Н. Томашевский, Е. Г. Жданова. – М.: Бестселлер, 2003. – 416 с.
135. Устенко, А.С. Основы математического моделирования и алгоритмизации процессов функционирования сложных систем [Электронный ре-

сурс]. Режим доступа: http://ustenko.fromru.com/index.html (2014, 20 фев.)
136. Федоров, Н.Ю. Справочник инженера по АСУТП. Проектирование и разработка [Текст]: учебно-практическое пособие / Н. Ю. Федоров. – М.: Инфра-Инженерия, 2008. – 928 с.,12 ил.
137. Филиппович, А. Ю. Система имитационного моделирования допечатных процессов [Текст] / А.Ю.Филиппович, И.С.Шапиро // Известия ВУЗов. Проблемы полиграфии и издательского дела, 2002. - № 3.
138. Форрестер, Дж. Основы кибернетики предприятия (индустриальная динамика) [Текст]: пер. с англ. / Дж. Форрестер; под общ. ред. Д. М. Гвишиани. – М.: Прогресс, 1971. – 340 с.
139. Форрестер, Дж. Динамика развития города [Текст] / Дж. Форрестер. – М.: Прогресс, 1974. – 287 с.
140. Форрестер, Дж. Мировая динамика [Текст] / Дж. Форрестер. – М.: АСТ, 2003. – 384с.
141. Ходеев, А.Ю. Принятие решения в допечатном производстве с помощью экспертных систем и систем имитационного моделирования [Текст] / А. Ю. Ходеев, А.Ю. Филиппович // Интеллектуальные технологии и системы. Сб. ст. Вып. 3. – М.: Изд-во МГУП, 2001.
142. Черемных, С.В. Структурный анализ систем: IDEF – технологии [Текст] / С. В. Черемных, И. О. Семенов, В. С. Ручкин. – М.: Финансы и статистика, 2003. – 208 с.
143. Черненький, В.М. Разработка САПР. Книга 9. Имитационное моделирование [Текст]: практическое пособие / Под ред. А. В. Петрова. – М.: Высш. школа., 1990. – 112 с.
144. Чернышов, В.Н. Теория систем и системный анализ : учеб. пособие / В.Н. Чернышов, А.В. Чернышов. – Тамбов : Изд-во Тамб. гос. техн. ун-та, 2008. – 96 с.
145. Чикуров, Н.Г. Моделирование систем [Текст]: учеб. пособие / Н.Г. Чикуров. – М.: РИОР: ИНФРА-М, 2013. – 398с.
146. Шеннон, Р. Имитационное моделирование систем: искусство и наука [Текст] / Р. Шеннон. – М.: Мир, 1978. – 297с.
147. Яблочников, Е.И. Моделирование приборов, систем и производственных процессов [Текст]: учеб. пособие / Е.И. Яблочников, Д.Д. Куликов, В.И. Молочник. – СПб: СПбГУИТМО, 2008. – 156 с.

148. Якимов, А.И. Технология имитационного моделирования систем управления промышленных предприятий [Текст]: монография / А.И.Якимов. – Могилев: Белорус.- Рос. ун-т, 2010. – 304 с.
149. Metropolis N., Ulam S. The Monte-Carlo method. –J. Amer. Statistical assoc., 1949, v.44, N247, p335-341.

Приложение

Пример 1. *Техническое задание на разработку имитационной модели дорожного движения по принципу «Зеленой волны».*

1. Общие сведения

Настоящее Техническое задание (ТЗ) составлено в соответствии с

ГОСТ 34.602-89 «Техническое задание на создание автоматизированной системы».

1.1. Полное наименование системы и ее условное обозначение

1.1.1. Полное наименование системы – Имитационная модель дорожного движения по принципу «Зеленой волны».

1.1.2. Краткое наименование системы – ИМДД.

1.2. Наименования предприятий разработчиков и заказчика

1.2.1. Заказчик – Институт профессионального образования и информационных технологий БГПУ им.М.Акмуллы.

1.2.2. Разработчиком является студент 4 курса ИПОИТ Аглиуллин Азамат Азатович., д.ф.-м.н., проф. Маликов Р.Ф

1.3. Основание для разработки

Междисциплинарный курсовой проект, направленный на проектирование и разработку имитационной модели дорожного движения.

1.4. Плановые сроки работ

1.4.1. Начало работ по созданию системы – 20 января 2013 г.

1.4.2. Первый этап работ – 20 января 2013 г. – 20 февраля 2013 г.

1.4.3. Второй этап работ – 21 февраля 2013 г. – 20 апреля 2013 г.

1.4.4. Третий этап работ – 20 апреля 2013 г. – 1 мая 2013 г.

1.4.5. Выполнение работ по развитию имитационной модели и поддержке ее функционирования планируется на протяжении всего жизненного цикла и сроками не ограничивается.

1.5. Нормативно-правовая база

Вопросы разработки, внедрения и использования ИМДД регулируются совокупностью актов, образующих нормативно-правовую базу ИМДД.

2. Назначение и цели создания системы

2.1. Назначение системы

2.1.1. ИМДД предназначена для комплексной автоматизации транспортного движения автомобилей на дорогах города.

2.1.2. Автоматизации подлежит регулирование дорожного движения по принципу «Зеленой волны».

2.2. Цель создания системы

Целью создания ИМДД является улучшение транспортного движения.

ИМДД предполагает разработку имитационной модели транспортного потока, которая будет обеспечивать наилучшую работу светофора на дорогах.

3. Характеристика объекта информатизации

3.1. Объект информатизации: транспортная система города.

3.2. Характеристика текущего уровня информатизации

3.2.1. Транспортная система обладает значительными и постоянно возрастающими информационными и вычислительными ресурсами. Информационные ресурсы слабо систематизированы, существенно разрознены, как логически, так и физически.

3.2.2. Информация о транспортной системе слабо представлена для доступа по телекоммуникационным каналам.

4. Требования к системе

4.1. Основные задачи, подлежащие реализации в ИМДД

4.1.1. Задачи, подлежащие реализации в ИМДД, разделяются на две категории: задачи обеспечения функционирования и информационные задачи. Решение задач обеспечения функционирования носит первостепенный характер и должно обеспечить базу для решения информационных задач.

Примечание. В процессе решения задач обеспечения функционирования осуществляется выбор и закупка ряда программных средств. При этом требуется обеспечить максимальную эффективность финансовых вложений и лицензионную чистоту всего используемого программного обеспечения.

4.1.2. К задачам обеспечения функционирования относятся:

Определение и анализ требований к базовой сетевой операционной системе, выбор операционной системы и приобретение в случае необходимости достаточного количества лицензий.

Примечание. Предварительный анализ позволил рекомендовать для использования в ИМДД в качестве платформы Windows XP/7.

Выбор среды программирования для разработки имитационной модели.

Примечание. Предварительный анализ позволил рекомендовать для разработки ИМДД среду AnyLogic.

Установка и настройка базового программного обеспечения.

Обучение ИМДД

Примечание. Для работы с ИМДД будет представлено руководство пользователя.

4.1.3. К задачам информационного обеспечения относятся:

Моделирование дорожного движения с заданным значением машин

Моделирование дорожного движения с возможностью переключения перекрестка

Вывод статистики о длине очереди

4.2. Требования к информационному обеспечению

4.2.1. Информационное обеспечение ИМДД составляют:

данные об интенсивности машин;

время переключения светофора;

данные о статистики.

5. Состав работ по созданию системы

5.1. Работы по созданию ИМДД проводятся в три этапа (см. раздел 1.4 настоящего технического задания):

–на первом этапе разрабатывается структура модели,

–на втором этапе выполняется программная разработка модели,

–на третьем этапе производится тестирование и отладка.

6. Порядок контроля и приемки системы

6.1. Приемка и оценка работ, выполненных в соответствии с настоящим техническим заданием осуществляется комиссией, включающей представителей Заказчика и Разработчика.

6.2. По результатам приемо-сдаточных работ оформляется Акт приемки-сдачи.

7. Требования к документированию

7.1. В состав принимаемых в эксплуатацию ИМДД включается комплект документации, подготовленный в соответствии с требованиями ЕСКД и ЕСПД.

Учебное издание

Рамиль Фарукович Маликов

Практикум по имитационному моделированию сложных систем в среде AnyLogic 6

Редактор Т.В. Подкопаева
Технический редактор И.В. Пономарев

Лиц. на издат. деят. Б848421 от 03.11.2000 г. Подписано в печать 17.12.2013.
Формат 60Х84/16. Компьютерный набор. Гарнитура Times New Roman.
Отпечатано на ризографе. Усл. печ. л. – 18,5. Уч.-изд. л. – 18,3.
Тираж 100 экз. Заказ №

ИПК БГПУ 450000, г.Уфа, ул. Октябрьской революции, 3а