621.4
Б759

ВОЕННАЯ АКАДЕМИЯ СВЯЗИ

В. Д. Боев, Д. И. Кирик, Р. П. Сыпченко

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ

Пособие для курсового и дипломного проектирования

Санкт-Петербург
2011

УДК 681.142.1.001.57
681.142.33
62-506/507

Боев В. Д., Кирик Д. И., Сыпченко Р. П.

Компьютерное моделирование: Пособие для курсового и дипломного проектирования. — СПб.: ВАС, 2011. — 348 с.

Излагаются сведения, необходимые при выполнении курсовых и дипломных проектов по дисциплинам, связанным с компьютерным моделированием. Даны указания по составу и оформлению пояснительной записки и рекомендации по содержанию разделов. Курсовое и дипломное проектирование выполняются в системах имитационного моделирования GPSS World и AnyLogic. В пособии приведены необходимые сведения по этим системам моделирования и примеры разработки моделей. Даны темы и основные направления проектирования.

Для студентов, курсантов, слушателей, аспирантов, преподавателей и научных работников.

Технический редактор Г. Н. Кузей

Подписано к печати 25.11.2010 г. Объем 21,75 печ. л. Зак. 66
Бесплатно

Типография ВАС

СОДЕРЖАНИЕ

Введение

Учебное пособие предназначено для обучаемых, изучающих курс имитационного моделирования в рамках дисциплин под разными названиями: «Компьютерное моделирование», «Моделирование», «Моделирование систем», «Имитационное моделирование систем управления» и т.п.

Целью проектирования является освоение практических приёмов имитационного моделирования, планирования, проведения и обработки данных компьютерного эксперимента. Имитационное моделирование выбрано потому как мировая практика научных исследований свидетельствует о том, что методы имитационного моделирования занимают около 70 % в общем объёме исследовательского инструментария.

В настоящее время в имитационном моделировании выделяют три подхода: системной динамики, дискретно-событийный и агентный. Из этих подходов в рамках многих дисциплин изучается дискретно-событийный подход, обеспечивающий универсальность и эффективность имитационного моделирования. Он ориентирован на исследование широкого класса сложных систем, представимых в виде систем массового обслуживания. Развитию и широкому распространению данного подхода в значительной степени способствовало наличие у разработчиков имитационных моделей специализированных систем имитационного моделирования, основанных на языке GPSS. Непрерывная эволюция этих систем, берущая начало от системы GPSS 1, созданной Джеффри Гордоном в 1961 г., позволяет в течение уже более пятидесяти лет использовать язык GPSS на новых типах ЭВМ, в новых операционных системах, расширяя возможности самого языка. Доступность учебных версий таких наиболее известных систем моделирования из этой серии, как GPSS/PC и GPSS World, а также учебно-методического обеспечения [1, 2, 3], создаёт условия для эффективной организации учебного процесса по имитационному моделированию сложных систем.

Система AnyLogic разработана компанией XJTechnologies в 1991 г. Это объектно-ориентированная система. Она для построения моделей позволяет использовать системную динамику, агентный и дискретно-событийный подходы. Также, наряду с GPSS World, получила признание и широкое распространение.

В зависимости от заданий на курсовые и дипломные проекты эксперименты на имитационной модели могут выполняться с использованием автоматически создаваемых генераторов экспериментов GPSS World и AnyLogic.

Помимо жестких требований по оформлению пояснительной записки, в пособии даны рекомендации по раскрытию содержания разделов с приведением справочных материалов. В некоторых случаях авторы сочли необходимым давать ссылки на источники, где обучаемые могут получить более подробные сведения.

Пособие составлено с учетом того, что исполнители проектов усвоили соответствующие дисциплины в объеме основных учебных пособий, например [1, 7].

Для обучаемых предусмотрена возможность самостоятельного выбора темы курсового или дипломного проектирования. В пособии также приведён ряд заданий на проектирование, темы которых наиболее актуальны для профессиональной ориентации обучаемых.

Новые направления в применении имитационного моделирования связаны с их использованием для решения задач прогнозирования и принятия решений в процессе управления сложными системами.

В пособии учтен опыт курсового проектирования в Военной академии связи и других высших учебных заведениях. Использованы материалы, опубликованные на сайте www.xjtek.ru [9]. На этом сайте также доступна ознакомительная версия AnyLogic 6.

Авторы благодарны доктору технических наук, профессору И. Б. Саенко за рецензирование пособия, сотрудникам компании XJTechnologies П. А. Лебедеву и С. А. Суслову за рекомендации по построению моделей, а также Д. В. Боеву за допечатную подготовку.

Авторы

Глава 2. Система моделирования AnyLogic

2.1. Общие сведения о системе моделирования 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. Использование AnyLogic дает возможность оценить эффект конструкторских решений в сложных системах реального мира.

Отечественный профессиональный инструмент имитационного моделирования AnyLogic нового поколения, который 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.

Моделирование систем с дискретными событиями. На рынке существует множество пакетов, облегчающих разработку дискретно-событийных моделей и проведение экспериментов с моделями в этой традиционной области моделирования. В первую очередь это GPSS, который стал революцией более 50 лет назад, задав парадигму моделирования в этой области в виде блоков и транзактов. Транзакты отображают динамические объекты моделирования (заявки), а блоки — объекты, обрабатывающие эти заявки. Большинство других инструментов моделирования (Arena, Extend, ProModel, SimProcess и др.) также используют эту парадигму.

Системная динамика — это методология изучения и моделирования систем, характеризующихся циклами обратных связей в сложных взаимных причинных зависимостях их параметров. Математически эти системы описываются системами дифференциальных уравнений, приведенных к форме Коши. Эти модели применяются для корпоративного планирования и анализа политик управления корпорацией, политик управления социальными и экономическими системами, в экологии и т.п. В 1961 г. Джей Фор-

рестер заложил методологические принципы системной динамики с использованием графического представления причинных зависимостей переменных, в виде так называемых «stock and flow diagrams», которые и сейчас являются основой инструментов моделирования, конкурирующих на рынке: VenSim, PowerSim, Stella, ModelMaker и др.

Динамические системы, область моделирования систем управления, физических, механических систем, систем обработки сигналов и т.п. Широкое применение в этой области имеет пакет моделирования Simulink, являющийся составной частью пакета Matlab. Пакет имеет библиотеку предопределенных блоков, из которых можно методом «drug-and-drop» строить блочную структуру, аналогичную блочной структуре моделей, когда-то давно, лет 40 назад набираемых для их решения на аналоговых вычислительных машинах из интеграторов, усилителей, сумматоров, источников сигналов и т. п.

Агентное моделирование. Под агентом понимается активный объект, обладающий поведением и имеющий возможность взаимодействия с другими агентами и со средой. Многоагентное моделирование позволяет вывести характеристики целого (множества агентов) из совокупности локальных поведений и характеристик отдельных активных элементов целого, распределенных в среде. Моделирование многоагентных систем используется в анализе социальных процессов, процессов урбанизации и даже при исследовании рынка в анализе предпочтений различных социальных групп или корпораций, выступающих как агенты со своим поведением. Существует несколько экспериментальных инструментов, поддерживающих моделирование в этой области; наиболее известные из них — это Swarm, разработанный в Университете Санта Фе и RePast, разработанный в Чикагском Университете. Эти академические инструменты предоставляют пользователю библиотеки программных модулей, несколько упрощающих разработку агентных моделей по сравнению с разработкой их «с нуля» на языке программирования, но они никак не могут претендовать на роль продукта для профессионалов-симуляционистов.

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 кон-

цепция 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 со-

бытия. Например, анализ IT-инфраструктуры компании (анализ производительности серверов, узких мест локальной сети и т. п.). Он легко производится с помощью методов дискретного событийного моделирования, имеет немного пользы, если в модели не отражено влияние возможных изменений параметров этой инфраструктуры на бизнес-процессы и, в конечном счете, на прибыль компании, а такая связь в модели не может быть реализована только средствами дискретно-событийного моделирования. 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.

Удобный интерфейс и многочисленные средства поддержки разработки моделей в AnyLogic делают не только использование, но и создание компьютерных имитационных моделей в этой среде моделирования доступным даже для начинающих.

2.2. Этапы имитационного моделирования в AnyLogic

Имитационное моделирование состоит из двух больших этапов: создания модели и анализа полученных с помощью модели результатов с целью принятия решения.

При детальном рассмотрении, построение действительно полезной имитационной модели требует большой работы. Сначала разработчик модели должен определить, какие задачи будут решаться с ее помощью, т. е. моделированию в любой его форме должна предшествовать формулировка цели моделирования. От цели зависит то, какие процессы в реальной системе следует выделить и отразить в модели, а от каких процессов абстрагироваться, какие характеристики этих процессов учитывать, а какие – нет, какие соотношения между переменными и параметрами модели должны быть отражены в модели. Данный этап можно охарактеризовать как создание концептуальной (содержательной) модели. На нем происходит структуризация модели, т.е. выделение отдельных подсистем, определение элементарных компонентов модели и их связей на каждом уровне иерархии.

В имитационном моделировании структура модели отражает структуру реального объекта моделирования на некотором уровне

абстракции, а связи между компонентами модели являются отражением реальных связей. Элементы системы, их связи, параметры и переменные, а также их соотношения и законы их изменения должны быть выражены средствами среды моделирования, т.е. в этой среде должны быть определены переменные и параметры модели, построены процедуры вычисления изменения переменных и характеристик модели во времени.

При необходимости для большего понимания процессов, протекающих в модели, должно быть разработано анимационное представление этих процессов.

Затем построенная модель должна быть проверена с точки зрения корректности ее реализации.

Следующий этап — это калибровка или идентификация модели, т.е. сбор данных и проведение измерений тех характеристик в реальной системе, которые должны быть введены в модель в виде значений параметров и распределений случайных величин.

Далее, необходимо выполнить проверку правильности модели (ее валидацию), которая состоит в том, что выход модели проверяется на нескольких тестовых режимах, в которых, характеристики поведения реальной системы известны либо очевидны. Последним этапом работы с моделью является компьютерный эксперимент, т. е. собственно то, ради чего и создавалась модель.

В простейшем случае эксперимент — это выполнение модели при различных значениях ее существующих параметров (факторов) и наблюдение ее поведения с регистрацией характеристик поведения. Этот вид использования модели называется прогнозом, или экспериментом типа «что будет, если…». Компьютерное моделирование позволяет не только получить прогноз, но и определить, какие управляющие воздействия на систему приведут к благоприятному развитию событий. Более сложные эксперименты позволяют выполнить анализ чувствительности модели, оценку рисков различных вариантов управляющих решений, а также оптимизацию для определения параметров и условий рационального функционирования модели.

Один из важных вопросов — представление и анализ результатов моделирования. Для этого в инструментальной среде могут быть использованы специальные средства для обработки статистической информации, для представления в структурированном или графическом виде получения данных, интеграция с внешними базами данных и т.п.

Таблица 2.1

Этапы компьютерного моделирования

№ п./п.	Название этапа	Результат
1	Анализ системы	Понимание того, что происходит в системе, подлежащей анализу, какова ее структура, какие процессы в ней протекают
2	Формулировка цели моделирования системы	Список задач, которые предполагается решить с помощью будущей модели. Список входных и выходных параметров модели, список исходных данных, критерии завершенности будущего исследования
3	Разработка концептуальной структуры модели	Структура модели, состав существенных процессов, подлежащих отображению в модели, зафиксированный уровень абстракции для каждой подсистемы модели (список допущений), описание управляющей логики для подсистем
4	Реализация модели в среде моделирования	Реализованные подсистемы, их параметры и переменные, их поведение, реализованная логика и связи подсистем
5	Реализация анимационного представления модели	Анимационное представление модели, интерфейс пользователя
6	Проверка корректности реализации модели	Убеждение в том, что модель корректно отражает те процессы реальной системы, которые требуется анализировать
7	Калибровка модели	Фиксация значений параметров, коэффициентов уравнений и распределений случайных величин, отражающих те ситуации, для анализа которых модель будет использоваться

8	Планирование и проведение компьютерного эксперимента	Результаты моделирования — графики, таблицы и т.п., дающие ответы на поставленные вопросы

Часто имитационная модель используется в качестве модуля большей системы принятия решения, получающей в режиме реального времени данные мониторинга состояния управляемой системы, оценивающей, к каким последствиям может привести текущая ситуация, и предлагающей оптимальное (или просто рациональное) управляющее решение для минимизации отрицательных последствий развития системы в будущем. Для этого обычно требуется интеграция модели с другими информационными системами и разработка специального интерфейса пользователя.

2.3. Основные концепции, реализуемые AnyLogic

Две фазы имитационного моделирования. В AnyLogic две фазы имитационного моделирования — разработка модели и ее анализ — явно разделены. Разработка модели выполняется в среде редактора AnyLogic, анализ модели происходит в среде исполнения. В каждой фазе существуют свои средства управления. Переход из одной фазы в другую производится очень легко. Можно многократно использовать переход между фазами редактирования и исполнения модели при разработке модели.

Активные объекты, классы и экземпляры активных объектов. Класс в программировании является мощным средством, позволяющим структурировать сложную систему. Класс определяет шаблон, в соответствии с которым строятся отдельные экземпляры класса. Эти экземпляры могут быть определены как объекты других активных объектов.

В AnyLogic основным структурным блоком при создании моделей являются классы активных объектов. Использование активных объектов является естественным средством структуризации модели сложных систем: мир состоит из множества параллельно функционирующих и взаимодействующих между собой сущностей. Различные типы этих сущностей и представляют разные активные объекты.

Чтобы создать модель AnyLogic, нужно создать классы активных объектов (или использовать объекты библиотек AnyLogic). Определение активного объекта задает шаблон, и отдельные объекты, построенные в соответствии с этим шаблоном (экземпляры активного объекта), могут использоваться затем как элементы дру-

гих активных объектов. В любой класс могут быть включены несколько экземпляров других классов, которые могут различаться своими параметрами. Всегда один класс в модели является корневым. Для него в модели AnyLogic порождается один экземпляр с предопределенным именем root, он и запускается исполнительной системой AnyLogic на выполнение. Имя класса корневого активного объекта можно менять в окне его свойств.

Каждый активный объект имеет структуру (совокупность включенных в него активных объектов и их связи), а также поведение, определяемое совокупностью переменных, параметров, стейтчартов и т. п. Каждый экземпляр активного объекта в работающей модели имеет свое собственное поведение. Он может иметь свои значения параметров, он функционирует независимо от других объектов, взаимодействуя с ними и с внешней средой.

Объектно-ориентированный подход. AnyLogic использует объектно-ориентированный подход к представлению сложных систем. Этот подход позволяет простым и естественным образом организовать и представить структуру сложной системы с помощью иерархии абстракций. Например, на некотором уровне абстракции автомобиль можно считать неким единым объектом. Но более детально его можно представить как совокупность взаимодействующих подсистем: двигателя, рулевого управления, тормозной системы и т. п. Каждая из этих подсистем может быть представлена, если это необходимо, своей структурой взаимодействующих подсистем.

Именно такую иерархию абстракций позволяет создать AnyLogic при разработке моделей. Всю модель можно рассматривать как единый объект root. При детальном рассмотрении видно, что этот объект может содержать, к примеру, два экземпляра класса MyClass с именами myClass и myClass1. Сам класс MyClass может иметь свое сложное строение, которое будет скрыто в классе Root. Такая иерархия структуры может быть произвольной глубины.

Визуальная разработка модели. При построении модели в среде AnyLogic не предусмотрена возможность использования никаких других средств, кроме средств визуальной разработки (введения состояний и переходов стейтчарта, введения пиктограмм переменных и т. п.), задания численных значений параметров, аналитических записей соотношений переменных и аналитических записей условий наступления событий. Основной парадиг-

мой, принятой в AnyLogic при разработке моделей, является визуальное проектирование — построение с помощью графических объектов и пиктограмм иерархий структуры и поведения активных объектов.

Встроенный язык Java. AnyLogic является надстройкой над языком Java — одним из самых мощных и в то же время простых современных объектно-ориентированных языков программирования. Все объекты, определенные пользователем при разработке модели на AnyLogic с помощью его графического редактора, транслируются в конструкции языка Java, а затем происходит компиляция всей собранной программы на Java, задающей модель, в исполняемый код.

Хотя при построении модели на AnyLogic разработчик использует конструкции языка Java в большей или меньшей степени, в действительности он никогда не разрабатывает полные программы, он не программирует, а лишь вставляет фрагменты кода в специально предусмотренные для этого поля окна **Код** и окон свойств объектов модели. Эти фрагменты выражают логику конкретных шагов или действий в модели. Как и все другие включенные в модель программные фрагменты, эти выражения должны быть синтаксически правильными конструкциями Java, потому разработчик моделей AnyLogic должен иметь некоторое представление об этом языке.

Средства описания поведения объектов. Основным средством спецификации поведения объектов в AnyLogic являются переменные, таймеры и стейтчарты. Переменные отражают изменяющиеся характеристики объекта. Таймеры можно взводить на определенный интервал времени и по окончании этого интервала выполнять заданное действие. Стейтчарты позволяют визуально представить поведение объекта во времени под воздействием событий или условий, они состоят из графического изображения состояний и переходов между ними. Любая сложная логика поведения объектов модели в AnyLogic может быть выражена с помощью комбинации стейтчартов, дифференциальных и алгебраических уравнений, переменных, таймеров и программного кода на Java. Алгебраические и дифференциальные уравнения, как и логические выражения, записываются в AnyLogic аналитически.

Имитация нескольких параллельно протекающих процессов. Интерпретация любого числа параллельно протекающих про-

цессов в модели AnyLogic скрыта от пользователя. Никаких календарей событий разработчик модели на AnyLogic не ведет, отслеживание событий во всех процессах, определенных в модели, выполняется системой автоматически. Никаких усилий разработчика для организации квазипараллелизма интерпретации при этом не требуется.

Модельное и реальное время. Понятие модельного времени является базовым в системах имитационного моделирования. Модельное время — это условное логическое время, в единицах которого определено поведение всех объектов модели.

В моделях AnyLogic модельное время может изменяться либо непрерывно, если поведение объектов описывается дифференциальными уравнениями, либо дискретно, переключаясь от момента наступления одного события к моменту наступления следующего события, если в модели присутствуют только дискретные события. Моменты наступления всех планируемых событий в дискретной модели исполнительная система хранит в так называемом календаре событий, выбирая оттуда наиболее раннее событие для выполнения связанных с ним действий. Значение текущего времени в моделях AnyLogic может быть получено обращением к функции getTime ().

Единицу модельного времени разработчик модели может интерпретировать как любой отрезок времени: секунду, минуту, час или год. Важно только, чтобы все процессы, зависящие от времени, были выражены в одних и тех же единицах. При моделировании физических процессов все параметры и уравнения должны быть выражены в одной и той же системе физических величин. Например, все физические величины выражаются в системе Си, в которой единица времени — секунда, а единица длины — метр.

Интерпретация модели выполняется на компьютере. Физическое время, затрачиваемое процессором на имитацию действий, которые должны выполняться в модели в течение одной единицы модельного времени, зависит от многих факторов. Поэтому, конечно, единица физического и единица модельного времени не совпадают.

В AnyLogic приняты два режима выполнения моделей: режим виртуального времени и режим реального времени. В режиме виртуального времени процессор работает с максимальной скоростью без привязки к физическому времени. Этот режим используется

для факторного анализа модели, набора статистики, оптимизации параметров модели и т. п. Поскольку анимация и другие окна наблюдения за поведением модели обычно существенно снижают производительность компьютера, для повышения скорости выполнения модели эти окна нужно закрыть.

В режиме реального времени пользователь задает связь модельного времени с физическим временем, то есть устанавливается ограничение на скорость работы процессора при выполнении модели. В этом режиме задается количество единиц модельного времени, которые должны выполняться процессором в одну секунду. Обычно данный режим включается для того, чтобы визуально представить функционирование системы в реальном темпе наступления событий, проникнуть в суть процессов, происходящих в модели.

Соотношение физического и модельного времени при работе модели в режиме реального времени можно понять на следующем примере. При коэффициенте ускорения 4, если процессор успевает выполнить менее чем за 1 с все операции, которые в модели определены в течение четырех единиц модельного времени, он будет ждать до конца секунды. Если же процессор не успевает сделать это, то у него не будет интервала ожидания, и коэффициент ускорения будет меньше того, который установлен пользователем.

Анимация поведения модели. Удобные средства разработки анимационного представления модели в AnyLogic позволяют представить функционирование моделируемой системы в живой форме динамической анимации, что позволяет «увидеть» поведение сложной системы.

Средства анимации позволяют пользователю легко создать виртуальный мир (совокупность графических образов, ожившую мнемосхему и т. п.), управляемый динамическими параметрами модели по законам, определенным пользователем с помощью уравнений и логики моделируемых объектов. Визуальное представление поведения системы помогает пользователю проникнуть в суть процессов, происходящих в системе.

Интерактивный анализ модели. Многие системы моделирования позволяют менять параметры модели только до запуска модели на выполнение. Система AnyLogic предоставляет пользователю возможность изменять параметры модели в ходе ее функционирования. Поэтому окно анимации можно назвать «стендом» для проведения компьютерного эксперимента с моделью. Основным

примером средств, изменяющих параметры модели, являются слайдеры (бегунки).

2.4. Объекты Enterprise Library

Библиотека AnyLogic Enterprise Library поддерживает дискретно-событийный подход моделирования. С помощью объектов Enterprise Library можно моделировать системы реального мира, динамика которых представляется как последовательность операций (прибытие, задержка, захват или недостаток ресурса, разделение, ...) над некими сущностями (entities, по-русски — транзакты, заявки).

Сущностями могут быть, сообщения, запросы, документы, звонки, пакеты данных, транспортные средства, технические устройства и т. п. Эти сущности пассивны и сами не контролируют свою динамику. Но они могут обладать определёнными атрибутами, влияющими на процесс их обработки (например, тип звонка, сложность работы, длина сообщения) или накапливающими статистику (общее время ожидания, стоимость, время обработки).

Процессы задаются в форме потоковых диаграмм (блок-схем). То есть в графическом представлении, принятом во многих областях (производстве, бизнес-процессах, центрах обработки звонков, коммутации сообщений, логистике, здравоохранении и т. д.). Потоковые диаграммы AnyLogic иерархичны, масштабируемы. Они также расширяемы и объектно-ориентированы, что позволяет пользователю моделировать сложные системы любого уровня детальности. Другой важной особенностью Enterprise Library является возможность создания достаточно сложной анимации моделей.

В Enterprise Library также входят объекты, разработанные для моделирования процессов, происходящих в пространстве. То есть таких процессов, где объекты-заявки и ресурсы перемещаются в некой сети. Это подмножество объектов значительно упрощает моделирование некоторых типов систем, например, производства, внутризаводской логистики, супермаркета, склада, госпиталя, мастерской.

Для использования этого подхода, называемого сетевым моделированием, необходимо определить топологию сети (например, используя векторную графику AnyLogic поверх плана или чертежа здания или сооружения), множество ресурсов (статических, движущихся или перемещаемых), и собственно процесс. Процесс

в данном случае — это комбинация объектов типа «переместиться туда-то» или «присоединить к себе ресурс» (таких, как NetworkMoveTo, NetworkSeize, NetworkSendTo) и обычных объектов Enterprise Library (Queue, Delay, Service). Заявки и ресурсы автоматически анимируются движущимися объектами по сегментам сети или находящимися в её узлах. Эта анимация может также комбинироваться с обычной анимацией.

Иерархические модели и повторно используемые модельные компоненты. Если моделируемая система сложна, имеет смысл разбить её модель на компоненты (подпроцессы) и поместить каждый из них в отдельный активный объект. Вы можете определить входы и выходы из подпроцесса, поместить их на внешний интерфейс активного объекта и скрыть его реализацию. На верхнем уровне вы будете оперировать такими объектами как блоками, соединяя их входы и выходы. Вы можете создать несколько экземпляров активного объекта с разными параметрами, в том числе и в других проектах.

Пример: вам понадобилось создать в модели аэропорта такие компоненты, как `Регистрация`, `ПаспортныйКонтроль`, `Посадка` и т. д. В блок, моделирующий зону регистрации, вы можете лишь единожды добавить объект `СтойкаРегистрации` и сделать его реплицированным. Ресурсы модели могут разделяться между различными активными объектами, так что вы можете, например, использовать общий компонент `ИнформационнаяСистема`, предоставляющий ресурсы для этих подпроцессов.

Расширение процессных моделей. Как и все в AnyLogic, процессная модель может быть расширена до любого требуемого уровня детальности с необходимой нестандартной функциональностью.

Во-первых, базовые классы Entity и ResourceUnit могут быть расширены вашими собственными классами с любыми дополнительными полями и методами. Указав эти классы в параметрах классов объектов, вы позволите более простой доступ к вашим полям (без явного приведения типов, как это было в AnyLogic 5). Например, в модели аэропорта вам может понадобиться создать специальные классы, моделирующие пассажиров, багаж, офицеров безопасности и т. д.

Во-вторых, у каждого объекта Enterprise Library есть специальные «точки расширения» — места, где вы можете задать какие-то

действия или выражения. Такими точками расширения являются динамические параметры (помеченные в описаниях объектов как [динамические]), вычисляемые во время выполнения модели при прохождении заявок через процессную диаграмму. Например, в объекте Delay, моделирующем распечатку посадочных талонов в объекте `Регистрация`, вы можете присвоить пассажиру выход (гейт), написав `entity.gate=main.gateof(entity.flightno)` в параметре `Действие` при выходе. Здесь `gate` (гейт) и `flightno` (номер полета) — поля класса Passenger, моделирующего пассажиров, а `gateof` — функция, заданная в корневом объекте модели Main.

Обратите внимание, что у вас есть доступ к любой части модели из любой другой части модели, и более того, поскольку вы пишете Java код, вы можете взаимодействовать с любым внешним программным обеспечением и/или использовать любой из огромного количества Java пакетов.

Создание анимации для процессных моделей. Библиотека Enterprise Library тесно интегрирована с анимационными средствами AnyLogic и позволяет создавать анимации процессов любой степени сложности, в том числе иерархические и с несколькими различными графическими представлениями процесса. Например, вы можете определить глобальный взгляд на процесс производства с несколькими агрегированными индикаторами, а также детальные анимации конкретных операций — и переключаться между ними.

Для большинства объектов Enterprise Library анимация задается следующим образом: вы рисуете фигуру, скажем, ломаную, указываете ее в параметре Фигура анимации, скажем, объекта Delay и определяете этому объекту, что вы хотите отображать заявки, задерживающиеся в этом объекте, движущимися по этой фигуре анимации (см. Анимация объектов Enterprise Library). Затем вы рисуете другую фигуру (или группу фигур), и делаете ее Фигурой анимации заявки, например, в объекте Source. Тогда пока заявки будут находиться в объекте Delay, анимации этих заявок будут отображаться в соответствующих точках этой ломаной.

В сетевом моделировании вы рисуете сеть узлов и сегментов, и заявки автоматически отображаются движущимися вдоль сегментов или остающимися в узлах (более подробную информацию вы можете получить здесь [9]).

Комбинирование процессных моделей с моделями и конструкциями других типов. Одним из главных преимуществ AnyLogic является возможность комбинирования различных стилей моделирования, позволяющая отражать комплексность и неоднородность систем реального мира. Вы можете комбинировать ваши процессные модели, построенные с помощью Enterprise Library, с моделями системной динамики или агентными моделями, или же просто создавать ваши собственные объекты с помощью базовых элементов AnyLogic и включать их в диаграмму, описывающую ваш процесс.

Есть множество способов комбинирования подходов моделирования, мы приведем лишь некоторые примеры:

Влияние системно-динамической диаграммы потоков и накопителей на процессную модель может быть реализовано, например, как объект Source, создающий заявки с интенсивностью, управляемой динамической переменной.

Обратная связь может быть представлена, например, системно-динамическим накопителем, увеличивающим свое значение при каждом прибытии заявки в объект процессной диаграммы Sink.

В определенный момент своего жизненного цикла агент может становиться заявкой и вставляться в процесс с помощью блока Enter (и наоборот), или же агент может существовать параллельно с заявкой и взаимодействовать с ней.

Если вам нужен объект со специфической функциональностью, отличной от функциональности, предоставляемой объектами Enterprise Library (и недостижимой путем комбинирования этих объектов), вы можете создать ваш собственный класс активного объекта и использовать объекты Exit и Enter в качестве интерфейсных элементов. Тогда вы сможете вставлять экземпляры данного класса активного объекта в вашу процессную диаграмму. В таком активном объекте можно использовать диаграммы состояний, события, переменные и т. д.

Классы активных объектов библиотеки AnyLogic Enterprise Library являются блоками, с помощью которых строятся блок-схемы, моделирующие процессы, которые будут происходить с заявками.

Библиотечные классы Entity и ResourceUnit являются базовыми классами для заявок и ресурсов соответственно. Объекты услов-

но делятся по своей функциональности на несколько категорий, их краткое описание дано ниже.

2.4.1. Поток заявок

 **Source**	Создает заявки.
 **Sink**	Уничтожает поступающие заявки.
 **Enter**	Вставляет уже существующие заявки в определенное место внутри процесса, заданного потоковой диаграммой.
 **Exit**	Извлекает поступающие в объект заявки из процесса, заданного потоковой диаграммой, предоставляя пользователю решать, что делать с этими заявками.
 **Hold**	Блокирует/разблокировывает поток заявок на определенном участке блок-схемы.
 **Split**	Для каждой поступающей заявки объект создает заданное число новых заявок и пересылает их дальше.
 **Combine**	Дожидается поступления двух заявок в порты *in1* и *in2* (в произвольном порядке), а затем создает новую заявку и направляет ее на выходной порт.
 **SelectOutput**	Направляет входящие заявки в один из двух выходных портов в зависимости от выполнения заданного условия.
 **Queue**	Хранит заявки в определенном порядке. Моделирует очередь заявок, ожидающих приема объектами, следующими за данным в потоковой диаграмме.
 **Match**	Синхронизирует два потока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия.
 **RestrictedAreaStart**	Обозначает вход в область процесса, в которой одновременно может находиться ограниченное количество заявок.

Restricted AreaEnd Обозначает выход из области процесса, в которой может находиться только ограниченное количество заявок.

2.4.2. Работа с содержимым заявки

Batch Преобразует заданное количество поступающих в объект заявок в одну заявку-партию.

Unbatch Извлекает все заявки, содержащиеся в поступающей заявке-партии, и пересылает их далее. Сама заявка-партия при этом уничтожается.

Pickup Удаляет заявки из заданного объекта **Queue** и добавляет их к содержимому поступающей заявки-контейнера.

Dropoff Удаляет избранные заявки из поступающей заявки-контейнера и пересылает их далее.

2.4.3. Обработка

Delay Задерживает заявки на заданный период времени.

2.4.4. Работа с ресурсами

ResourcePool Задает набор ресурсов, которые могут захватываться и освобождаться заявками.

Seize Захватывает для заявки заданное количество ресурсов определенного типа.

Release Освобождает ранее захваченные заявкой ресурсы.

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

2.4.5. Транспортировка

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.

2.4.6. Моделирование транспортных сетей

Network	Задает топологию сети и управляет сетевыми ресурсами.
NetworkEnter	Регистрирует заявку в сети и помещает ее в заданный узел сети.
NetworkExit	Удаляет заявку из сети.
NetworkMoveTo	Перемещает заявку в новое место сети.
NetworkResource Pool	Задает набор сетевых ресурсов, которые могут захватываться и освобождаться заявками.
NetworkRelease	Освобождает ранее захваченные заявкой сетевые ресурсы.
NetworkSeize	Захватывает для заявки заданное количество сетевых ресурсов.
NetworkSendTo	Посылает (перемещает) указанные движущиеся/переносные сетевые ресурсы из их текущего местоположения в заданный узел сети.
NetworkAttach	Присоединяет к заявке указанные движущиеся/переносные сетевые ресурсы.
NetworkDetach	Отсоединяет от заявки ранее присоединенные ресурсы.

2.4.7. Моделирование зон хранения в сети

 **NetworkStorage**	Моделирует два стоящих друг напротив друга стеллажа и проход между ними.
 **NetworkStorage Zone**	Моделирует зону хранения, состоящую из набора стеллажей и проходов между ними (моделируемыми с помощью объектов **NetworkStorage**), предоставляющий централизованный доступ и управление этими стеллажами.
 **NetworkStorage Pick**	Извлекает заявку из ячейки стеллажа или зоны хранения и перемещает ее в заданный узел сети.
 **NetworkStorage Put**	Помещает заявку в ячейку заданного стеллажа **NetworkStorage** или зоны хранения **NetworkStorageZone**.

2.4.8. Вспомогательные объекты

 **Clock**	Отображает текущее модельное время и дату в виде часов.

Глава 3. Разработка моделей в AnyLogic

3.1. Создание модели с использованием шаблона

Постановка задачи. Сервер обрабатывает запросы, поступающие с автоматизированных рабочих мест с интервалами, распределенными по показательному закону со средним значением 2 мин. Время обработки сервером одного запроса распределено по экспоненциальному закону со средним значением 3 мин. Сервер имеет входной буфер емкостью 5 запросов.

Построить имитационную модель для определения оценки математического ожидания вероятности обработки запросов.

Сервер представляет собой однофазную систему массового обслуживания разомкнутого типа с ограниченной входной емкостью.

Версия 6 AnyLogic при создании подобных простейших моделей предоставляет возможность использования шаблонов моделей. То есть выполнение первых одних и тех же шагов можно перепоручить **Мастеру создания модели**. Все, что нужно пользователю, это указать, какой метод моделирования будете применять и выбрать те опции, которые нужны в модели. После этого **Мастер** автоматически создаст простейшую модель. Далее при необходимости можно продолжить ее разработку, изменяя детали.

Поскольку мы только приступаем к разработке моделей в AnyLogic, то воспользуемся шаблоном. В дальнейшем, при изменении и дополнении модели согласно постановке задачи, вам станет понятно, как начинать разработку не только простейшей, но и более сложной модели «с нуля».

3.1.1. Создание диаграммы процесса

1. Выполните команду **Файл/Создать/Модель**  на панели инструментов. Появится диалоговое окно **Новая модель** (рис. 3.1).

2. Задайте имя новой модели. В поле **Имя модели** введите `Server`.

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

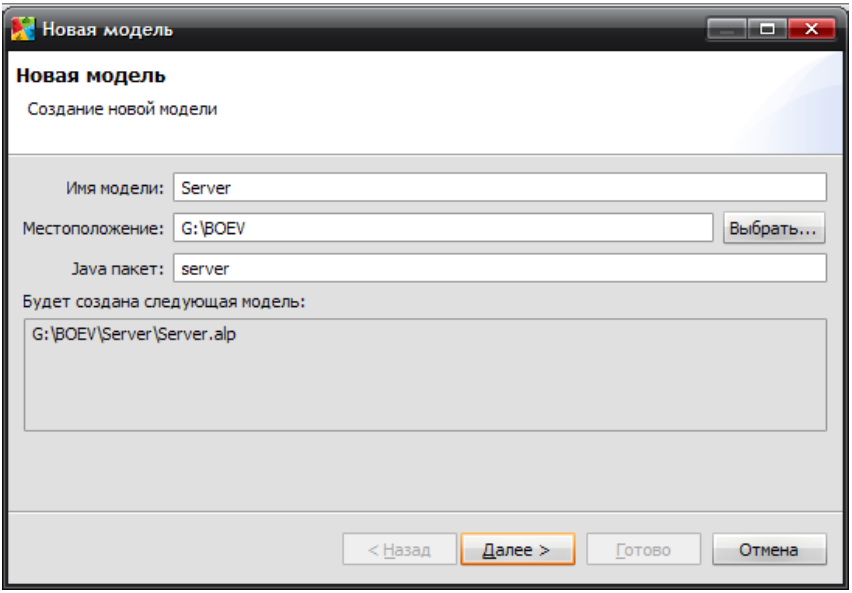


Рис. 3.1. Диалоговое окно **Новая модель**

4. Щелкните кнопку **Далее**. Откроется вторая страница **Мастера создания модели** (рис. 3.2).

5. Здесь будет предложено выбрать шаблон, на базе которого будете разрабатывать модель. Поскольку мы хотим создать новую дискретно-событийную модель не «с нуля», установите флажок **Использовать шаблон модели** и выберите **Дискретно-событийное моделирование** в расположенном ниже списке.

6. Щелкните кнопку **Далее**. На следующей странице **Мастера** будет предложено выбрать: хотите ли вы сразу же добавить в создаваемую модель ресурсы, график, отображающий длину очереди к сервису, анимацию обслуживающихся и ожидающих обслуживания заявок или гистограмму, отображающую распределение времени пребывания заявок в моделируемой системе. Поскольку мы хотим лишь создать с помощью **Мастера** простейшую диаграмму процесса, а остальные шаги выполнять совместно по шагам, чтобы вы знали, как добавлять ресурсы, создавать анимацию модели и собирать статистику и могли в дальнейшем делать это самостоятельно, то оставьте флажок **Использовать ресурсы** сброшенным и закончите создание модели, щелкнув мышью кнопку **Готово**.

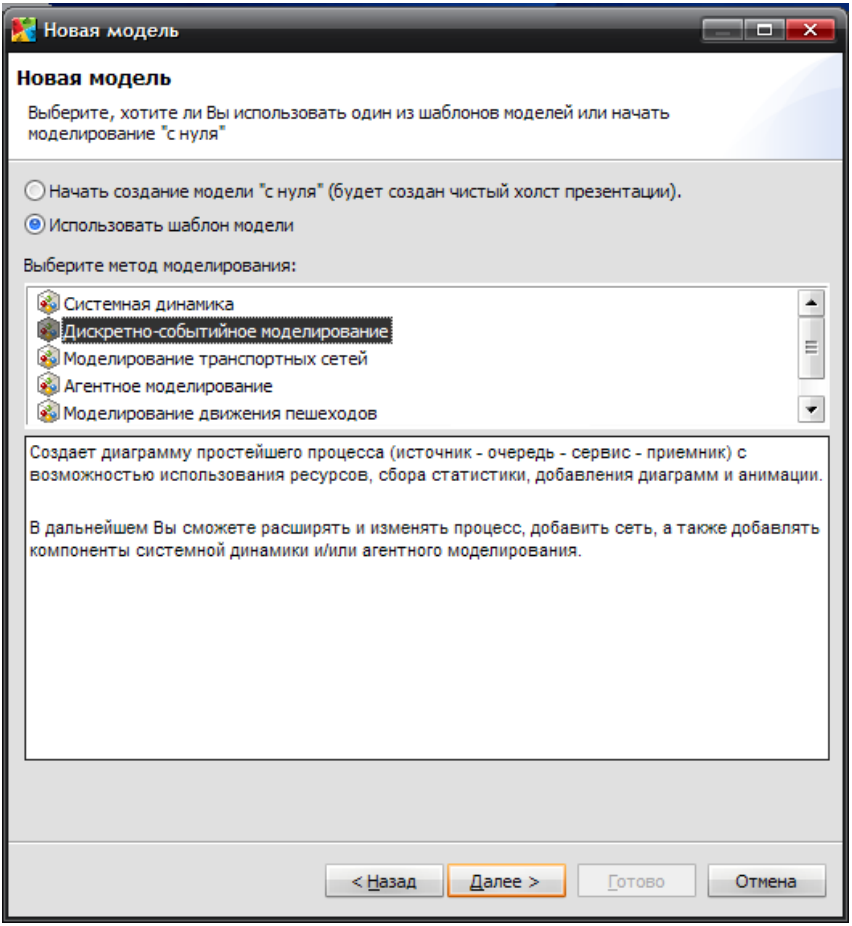


Рис. 3.2. Вторая страница диалогового окна **Новая модель**

7. Вы создали новую модель. Далее познакомимся с пользовательским интерфейсом AnyLogic (рис. 3.3).

8. В левой части рабочей области находится панель **Проект**. Панель **Проект** обеспечивает 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.

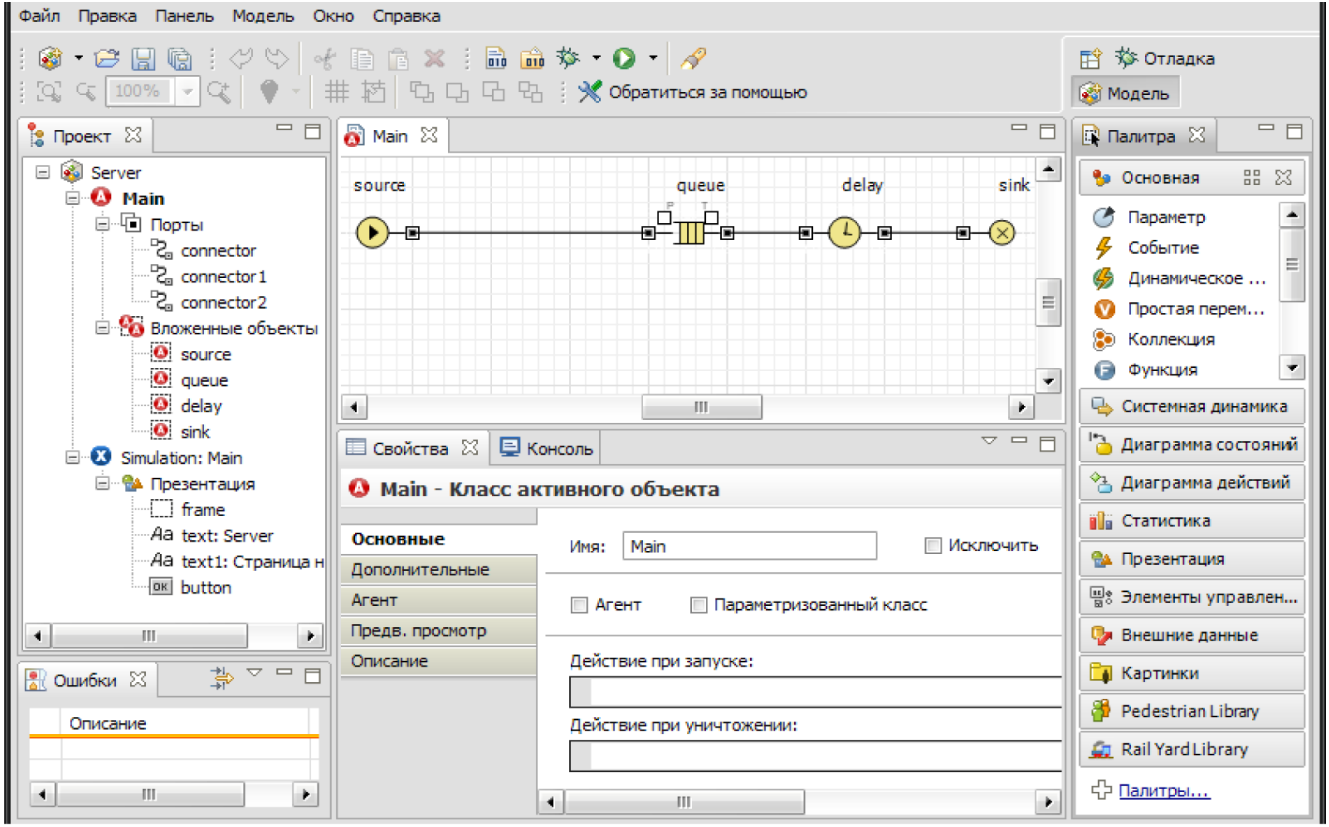


Рис. 3.3. Пользовательский интерфейс AnyLogic

9. В правой рабочей области будет отображаться панель **Палитра**, а внизу в средней части интерфейса — 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.

10.В центре рабочей области AnyLogic находится графический редактор диаграммы класса активного объекта **Main**.

11.В левой нижней части расположена панель **Ошибки**. Она отображает ошибки в модели и помогает их локализовать.

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

3.1.2. Изменение свойств блоков модели, ее настройка и запуск

Помним, что мы хотим сначала создать простейшую модель, в которой будем рассматривать только обработку запросов сервером.

В основе каждой дискретно-событийной модели лежит диаграмма процесса — последовательность соединенных между собой блоков (в AnyLogic это блоки библиотеки Enterprise Library), задающих последовательность операций, которые будут производиться над проходящими по диаграмме процесса заявками.

Обратите внимание на диаграмму класса Main. Вы увидите, что диаграмма нашего простейшего процесса (рис. 3.4) была автоматически создана **Мастером создания модели**, поскольку такая модель является ничем иным, как простейшей системой массового обслуживания, наиболее часто используемой в качестве отправной точки создания дискретно-событийных моделей. Поэтому она и выбрана в качестве базового шаблона при разработке дискретно-событийных моделей.

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

Все, что нам нужно, чтобы сделать созданный шаблон модели адекватным постановке задачи — это изменить некоторые свойства объектов.

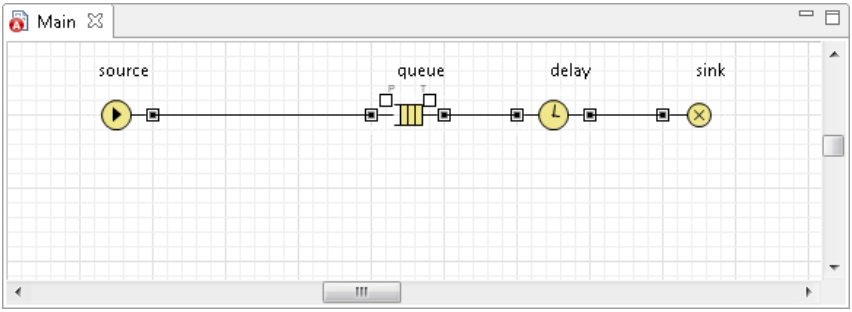


Рис. 3.4. Диаграмма простейшей системы массового обслуживания

3.1.2.1. Изменение свойств блоков диаграммы процесса

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 **Проект**.

Чтобы всегда была уверенность в том, что в текущий момент в рабочем пространстве выбран именно нужный 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.

Согласно принятым стандартам, блоки в диаграмме процесса обычно располагаются цепочкой слева направо, представляя собой последовательную очередность операций, которые будут производиться над заявкой.

Первым объектом в диаграмме процесса является объект типа **Source**. Объект **Source** генерирует заявки определенного типа. Заявки представляют собой объекты, которые производятся, обрабатываются, обслуживаются, или еще каким-нибудь образом подвергаются действию моделируемого процесса: это могут быть клиенты в системе обслуживания, детали в модели производства, транспортные средства в модели перевозок, документы в модели документооборота и т.д. В нашем примере заявками будут запросы на обработку данных, а объект Source будет моделировать поступление запросов на сервер.

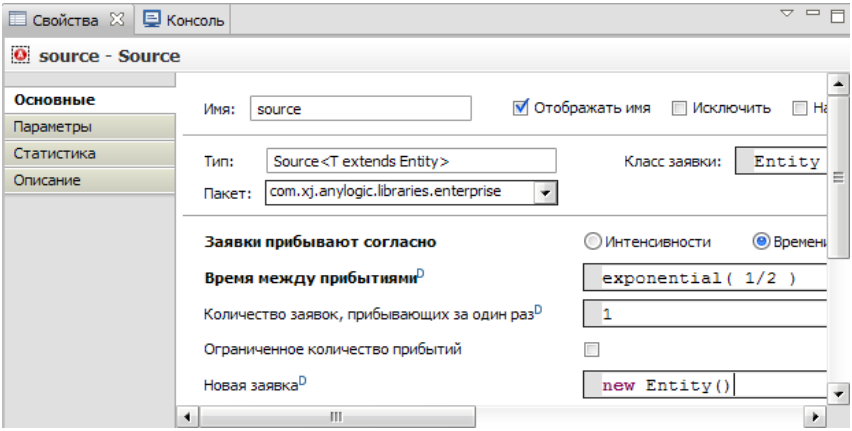


Рис. 3.5. Свойства объекта **Source**

В нашем случае объект создает заявки через временной интервал, распределенный по показательному (экспоненциальному) закону со средним значением 2 мин. Выделите объект source. Установите, что запросы поступают согласно **Времени между прибытиями** (рис. 3.5). Установите согласно постановке задачи среднее значение интервалов времени поступления запросов на сервер, изменив свойства объекта **source**. Для этого введите в поле **Время между прибытиями** `exponential(1/2)` вместо exponential(1).

Следующий объект — **Queue**. Выделите его. Он моделирует очередь заявок, ожидающих приема объектами, следующими за данным объектом в диаграмме процесса. В нашем случае он будет моделировать очередь запросов, ждущих освобождения сервера.

Измените свойства объекта **queue** (рис. 3.6).

1. Задайте максимальную длину очереди. Введите в поле **Вместимость** 5. В очереди будут находиться не более 5 запросов.

2. Установите флажок **Включить сбор статистики**, чтобы включить сбор статистики для этого объекта. В этом случае по ходу моделирования будет собираться статистика по количеству запросов в очереди. Если же вы не установите этот флажок, то данная функциональность будет недоступна, поскольку по умолчанию она отключена для повышения скорости выполнения модели.

Следующим в нашей диаграмме процесса расположен объект **Delay**. Он задерживает заявки на заданный период времени, представляя в нашей модели непосредственно сервер, на котором обрабатываются запросы.

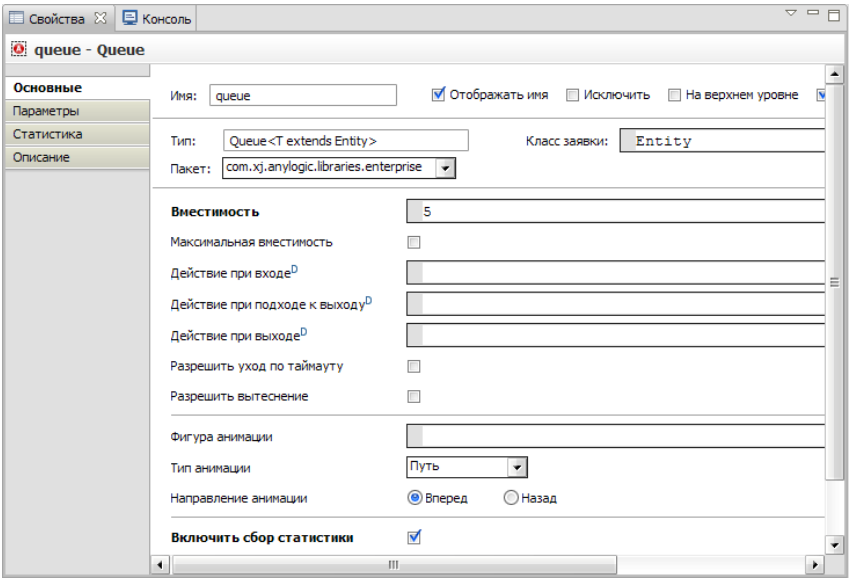


Рис. 3.6. Свойства объекта **Queue**

Измените свойства объекта **delay** (рис. 3.7).

1. Обработка одного запроса занимает примерно 3 минуты. Задайте время обслуживания, распределенное по экспоненциальному закону со средним значением, равным 3 мин. Для этого введите в поле **Время задержки** `exponential(1/3)`. Функция exponential() является стандартной функцией генератора случайных чисел AnyLogic. AnyLogic предоставляет функции и других случайных распределений, таких как нормальное, равномерное, треугольное, и т. д. Детальное описание всех функций вероятностных распределений дано здесь [9]. Мы установили среднее время поступления запросов и среднее время обработки запросов в минутах. Однако имеется возможность установить время в часах, секундах, днях, неделях, в чем вы убедитесь несколько позднее (п. 3.1.2.2), когда будете устанавливать модельное время.
2. Установите флажок **Включить сбор статистики**.

Последним в диаграмме нашей дискретно-событийной модели находится объект **Sink**. Этот объект уничтожает поступившие заявки. Обычно он используется в качестве конечной точки потока заявок (и диаграммы процесса соответственно). В нашем случае он выводит из модели обработанные сервером запросы.

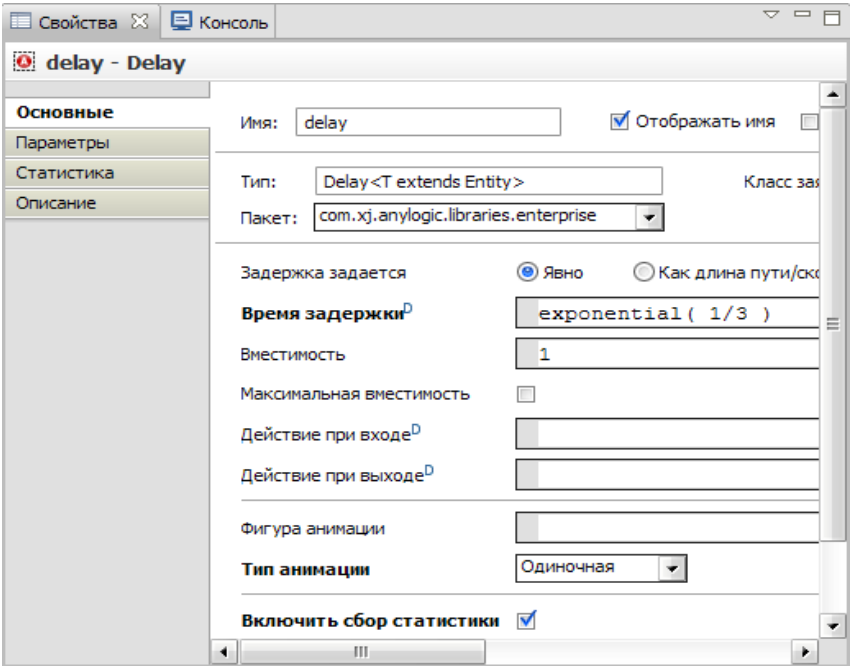
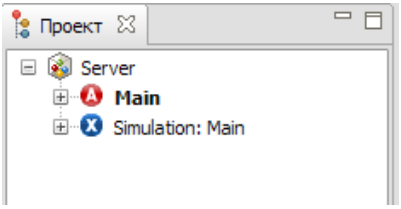


Рис. 3.7. Свойства объекта **Delay**

3.1.2.2. 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 **Проект** эксперименты отображаются в нижней части дерева модели. Один эксперимент, названный **Simulation**, создается по умолчанию (см. справа). Это простой эксперимент, позволяющий запускать модель с заданными значениями параметров, поддерживающий режимы виртуального и реального времени, анимацию и отладку модели.



Если мы запустим модель, то моделирование будет продолжаться 100 единиц модельного времени (установлены по умолча-

нию), после чего будет остановлено. Если вы хотите 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 истечении заданного времени (рис. 3.8). Для этого:

1. В панели **Проект**, выделите эксперимент **Simulation:Main**.
2. На странице **Модельное время** панели **Свойства** выберите `Нет` из выпадающего списка **Остановить:**.
3. Установите **Единицы модельного времени:** `минуты`.

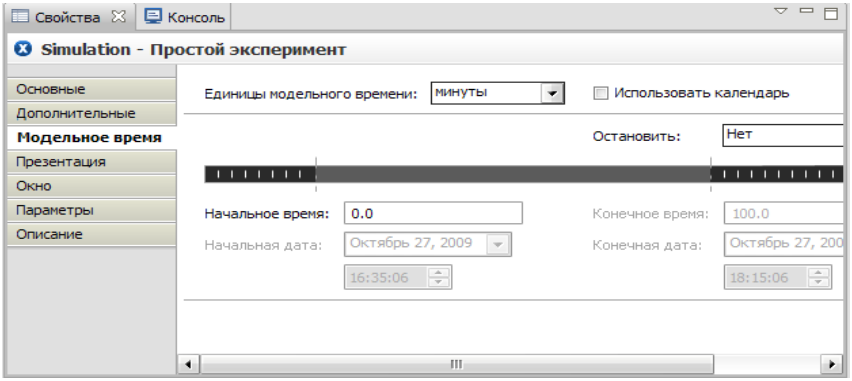


Рис. 3.8. Панель **Свойства**

3.1.2.3. Запуск модели

Постройте вашу модель с помощью кнопки панели инструментов (F7) **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 построите модель, вы можете ее запустить. Запустите модель:

1. Щелкните мышью кнопку панели инструментов **Запустить** (или нажмите F5) и выберите из открывшегося списка эксперимент, который вы хотите запустить. Эксперимент этой модели будет называться **Server/Simulation**.

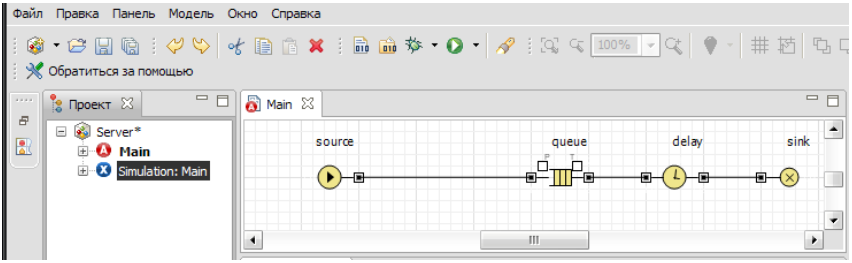


Рис. 3.9. Выбор эксперимента

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

3. После запуска модели вы увидите окно презентации этой модели (рис. 3. 10). 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).

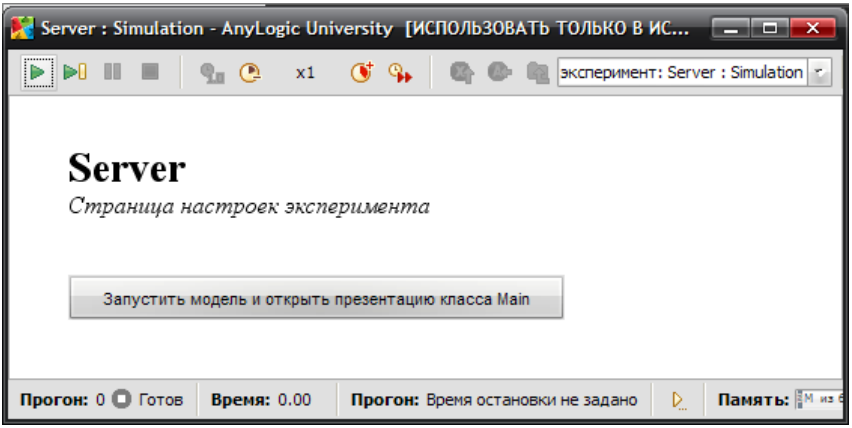


Рис. 3.10. Окно презентации модели

Рис. 3.11. Автоматически созданная блок-схема процесса

4. Щелкните данную кнопку. Этим щелчком вы запустите модель и перейдете к презентации корневого класса активного объекта запущенного эксперимента. Для каждой модели, созданной в Enterprise Library, автоматически создается блок-схема с наглядной визуализацией процесса, с помощью которой вы можете изучать текущее состояние модели, например, длину очереди, количество обработанных запросов и так далее (рис. 3.11).

5. Для каждого объекта определены правила, при каких условиях принимать заявки. Некоторые объекты задерживают заявки внутри себя, некоторые — нет. Для объектов также определены правила: может ли заявка, которая должна покинуть объект, ожидать на выходе, если следующий объект не готов ее принять. Если заявка должна покинуть объект, а следующий объект не готов ее принять, и заявка не может ждать, то модель останавливается с ошибкой (рис. 3.12). Ошибка означает, что запрос не может войти в блок queue, так как его емкость, равная 5, заполнена.

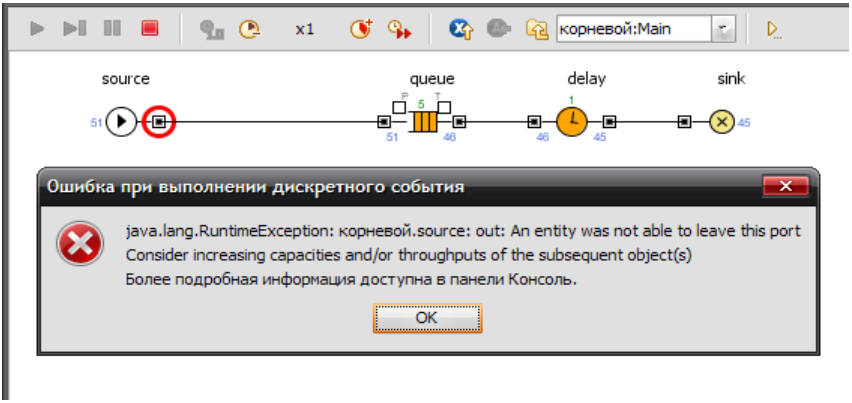


Рис. 3.12. Модель остановилась с ошибкой

6. Нажмите OK. Далее измените, свойства объекта **queue**, т. е. увеличьте длину очереди (см. рис. 3.6). Для этого введите в поле **Вместимость** 12. Можете убедиться, что при увеличении емкости в пределах 6 … 11 модель по-прежнему останавливается с этой же ошибкой. Момент появления ошибки зависит от длительности времени моделирования.

7. Снова запустите модель.

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

9. Вы можете следить за состоянием любого блока диаграммы процесса во время выполнения модели с помощью окна инспекта этого объекта. Чтобы открыть окно инспекта, щелкните мышью по значку нужного блока. Окно инспекта, подведя курсор, можно свободно перемещать в нужное вам место. Также, подведя курсор к правому нижнему углу окна инспекта, можно изменять его размеры. В окне инспекта будет отображена базовая информация по выделенному блоку: например, для блока Queue будет отображена вместимость очереди, количество заявок, прошедшее через каждый порт объекта, и т. д. (рис. 3.13).

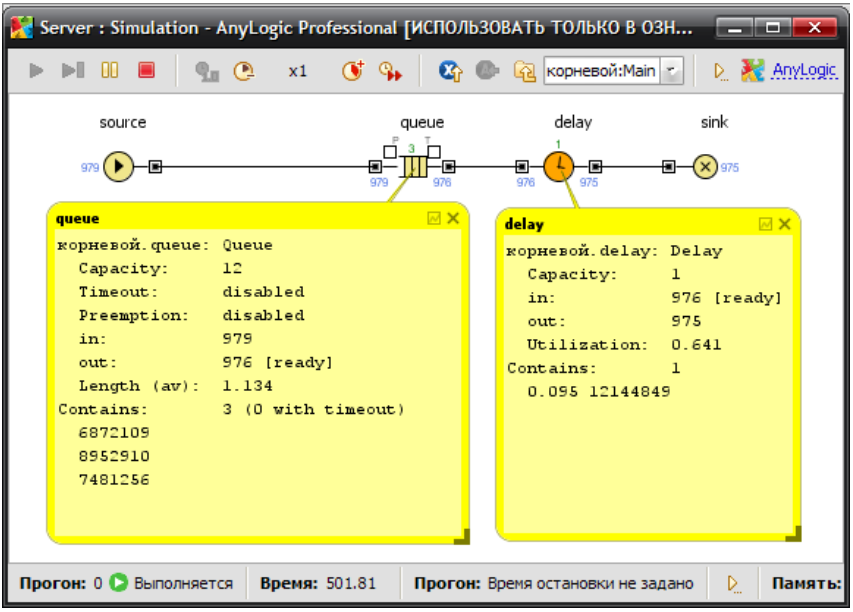


Рис. 3.13. Окно инспекта

10. Когда вы захотите остановить выполнение модели, щелкните мышью кнопку **Прекратить выполнение** панели управления окна презентации.

11. Для предотвращения остановок модели по ранее указанной ошибке мы увеличили емкость блока queue. Однако можно было бы изменять среднее время имитации запросов блоком source и среднее время обработки запросов сервером, т. е. среднее время задержки блока delay, оставляя неизменной длину очереди и добиваясь безошибочной работы модели. Конечно, при изменении свойств блоков модели нужно обязательно исходить из целей ее построения. Мы же не выполнили условий, указанных в постановке задачи, поэтому к выполнению их вернемся позже.

3.1.3. Создание анимации модели

Можно было наблюдать, анализировать и интерпретировать работу запущенной модели с помощью визуализированной диаграммы процесса (см. рис. 3.11, 3.13). Однако удобнее иметь более наглядную визуализацию с помощью анимации. В этом примере мы хотим создать визуализированный процесс поступления запросов на сервер и обработки запросов сервером.

Так как в данном случае нас не интересует конкретное расположение объектов в пространстве, то мы можем просто добавить схематическую анимацию интересующих нас объектов — сервер и очередь запросов к нему.

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

Нарисуйте прямоугольник, который будет обозначать на анимации сервер.

1. Вначале откройте закладку **Презентация** панели **Палитра** (рис. 3.14). Чтобы открыть какую-либо закладку панели **Палитра** (в дальнейшем палитра), нужно щелкнуть мышью по заголовку этой палитры.

2. Палитра **Презентация** (рис. 3.15) содержит в качестве элементов примитивные фигуры, используемые для рисования презентаций моделей. Это линия, ломаная, кривая, овал, дуга, точка, изображение, кнопка, флажок и др. Имеются также элементы управления, с помощью которых вы можете сделать ваши презентации интерактивными.

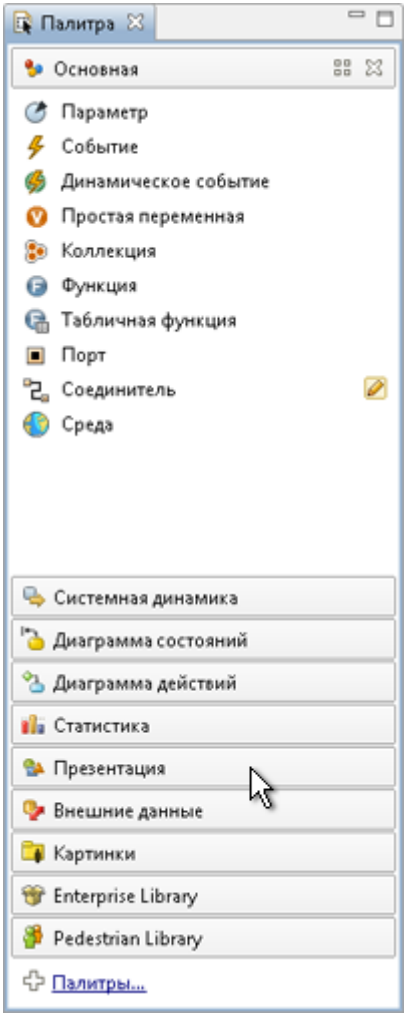


Рис. 3.14. Панель **Палитра**

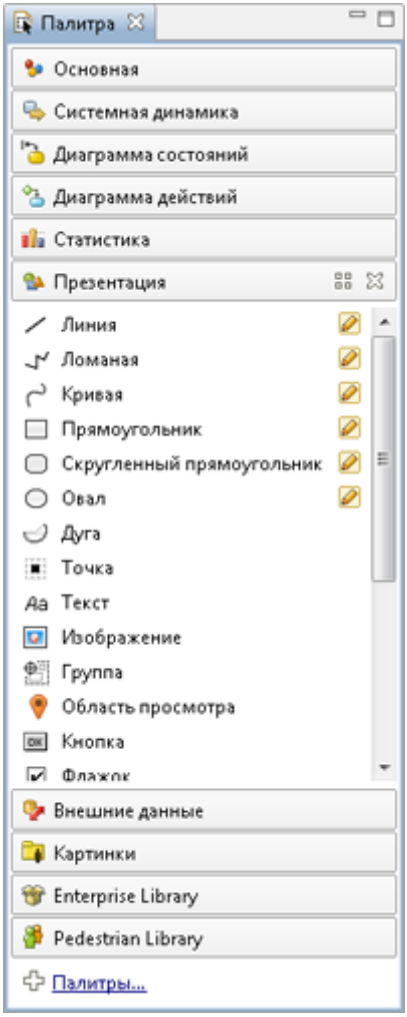


Рис. 3.15. Палитра **Презентация**

3. Выделите щелчком мыши элемент **Прямоугольник** на палитре **Презентация** и перетащите его на диаграмму класса активного объекта. Поместите элемент **Прямоугольник** так, как показано на рис. 3.16.

4. Давайте сделаем так, что цвет этого прямоугольника будет меняться в зависимости от того, обрабатывает ли сервер в данный момент времени запрос или нет.

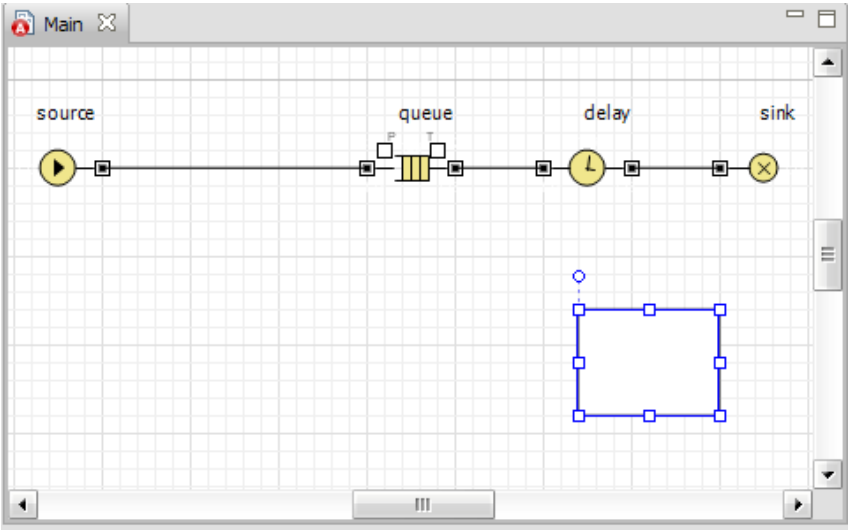


Рис. 3.16. Элемент **Прямоугольник** на диаграмме

5. Для этого выделите нарисованную нами фигуру на диаграмме и перейдите на страницу **Динамические** панели свойств (рис. 3.17). Здесь вы увидите список полей, в которых задаются значения динамических свойств фигуры.

Если нужно, чтобы по ходу моделирования то или иное свойство фигуры меняло свое значение в зависимости от каких-то условий, то можете ввести в поле соответствующего динамического свойства выражение, которое будет постоянно вычисляться заново при выполнении модели.

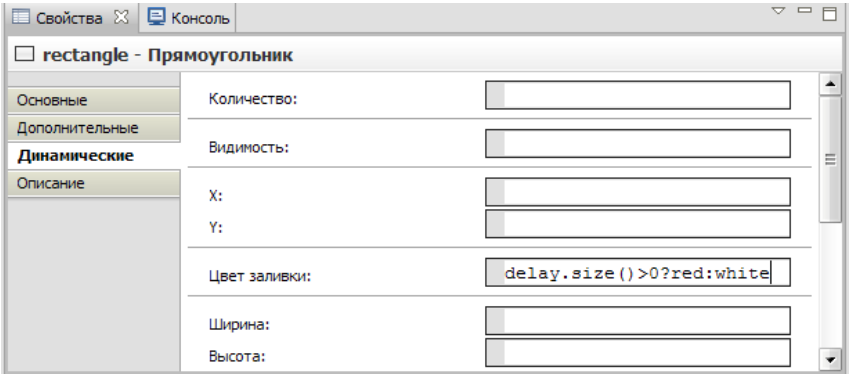


Рис. 3.17. Страница **Динамические** панели свойств

Возвращаемый результат вычисления будет присваиваться текущему значению этого свойства. Мы хотим, чтобы во время моделирования менялся цвет нашей фигуры, поэтому перейдите в поле **Цвет заливки** и введите там следующую строку: `delay.size()>0?red:white`

Здесь delay — это имя нашего объекта **Delay**. Функция size() возвращает число запросов, обслуживаемых в данный момент времени. Если сервер занят, то цвет кружка будет красным, в противном случае — зеленым.

6. Нарисуйте ломаную, которая будет обозначать на анимации очередь к серверу (рис. 3.18). 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.

Очень важно, какую точку ломаной вы создаете первой. Заявки будут располагаться вдоль нарисованной вами ломаной в направлении от конечной точки к начальной точке. Поэтому начните рисование ломаной слева и поместите рядом с сервером конечную точку ломаной, которая будет соответствовать в этом случае началу очереди.

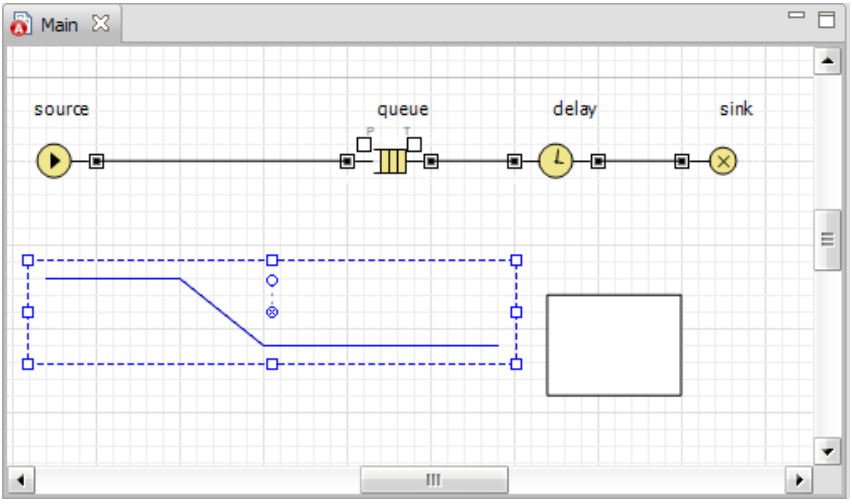


Рис. 3.18. Ломаная

7. Теперь мы должны задать созданные анимационные объекты в качестве анимационных фигур блоков диаграммы нашего процесса. Задайте ломаную линию в качестве фигуры анимации очереди. На странице свойств объекта queue, введите `polyline` в поле **Фигура анимации** (рис. 3.19).

8. Задайте прямоугольник в качестве фигуры анимации сервера. Введите в поле **Фигура анимации** имя нашего прямоугольника: `rectangle` (рис. 3.20). Выберите из выпадающего списка **Тип анимации** *Одиночная*. Большинство объектов Enterprise Library поддерживает несколько анимационных стилей. Например, очередь может отображаться в виде линии, упорядоченного или неупорядоченного набора элементов. В нашем случае, если сервер будет занят, то мы будем показывать в фигуре сервера обрабатываемого в нем запроса, а поскольку единовременно наш сервер не обрабатывает больше одного запроса, то мы и выбираем тип анимации *Одиночная*.

Теперь вы можете запустить модель. Для ускорения работы модели переключитесь в режим виртуального времени, щелкнув мышью кнопку **Реальное/виртуальное время** панели инструментов. В режиме виртуального времени модель будет выполняться быстро, без привязки модельного времени к реальному.

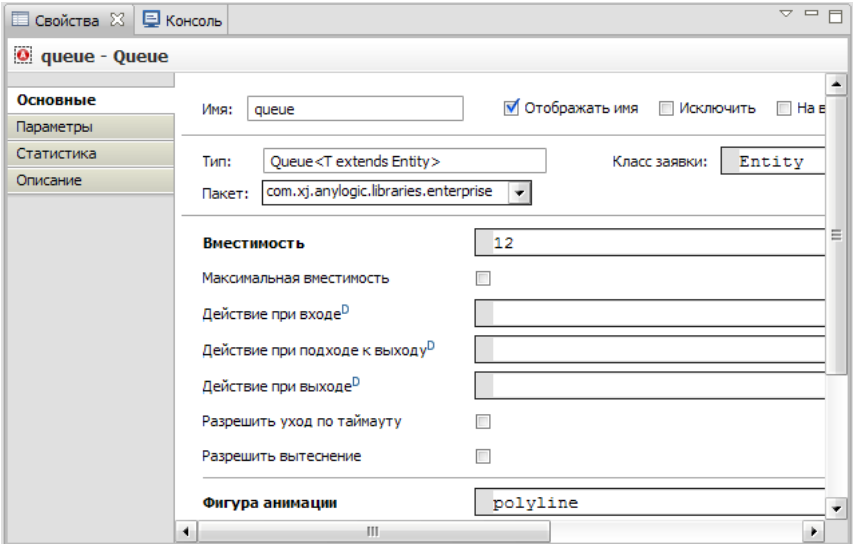


Рис. 3.19. Задание ломаной в качестве фигуры анимации очереди

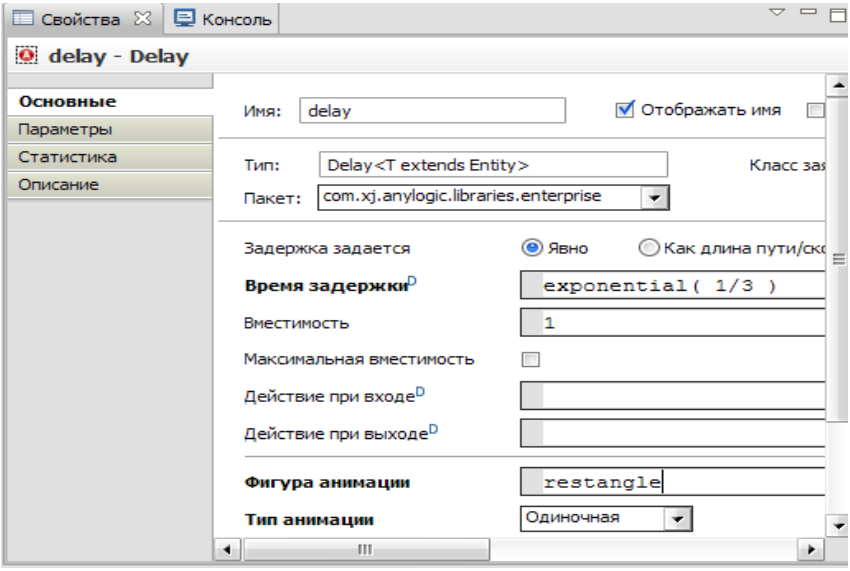


Рис. 3.20. Задание прямоугольника в качестве фигуры анимации сервера

9. Запустите модель. вы увидите, что у модели теперь есть простейшая анимация — сервер и очередь запросов к нему (рис. 3.21). Цвет фигуры сервера будет меняться в зависимости от того, обрабатывается ли запрос в данный момент времени или нет.

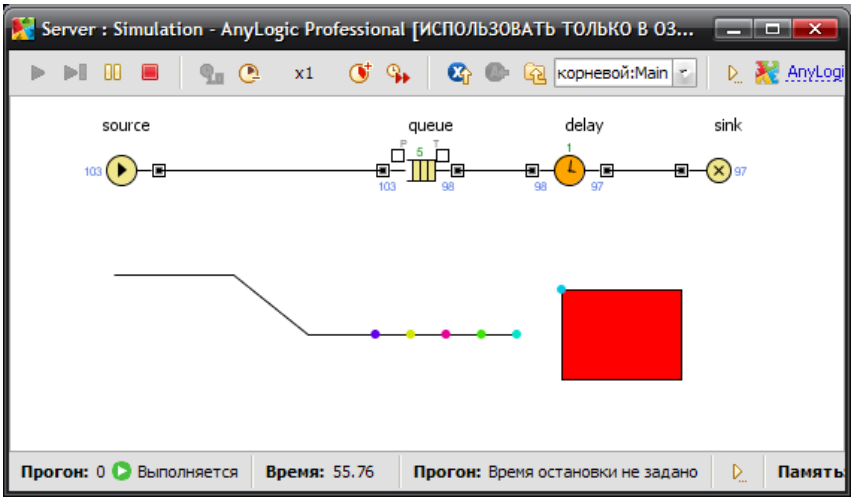


Рис. 3.21. Анимация модели

3.1.4. Сбор статистики использования ресурсов

AnyLogic предоставляет пользователю удобные средства для сбора статистики по работе блоков диаграммы процесса. Объекты Enterprise Library самостоятельно производят сбор основной статистики. Все, что вам нужно сделать — это включить сбор статистики для объекта.

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

Добавьте диаграмму для отображения средней занятости сервера:

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

2. Перетащите элемент **Столбиковая диаграмма** из палитры **Статистика** на диаграмму класса и измените ее размер, как показано на рис. 3.22.

3. Перейдите на страницу **Основные** панели **Свойства**. Щелкните мышью кнопку **Добавить элемент данных**. После щелчка появится секция свойств того элемента данных (**chart – Столбиковая диаграмма**), который будет отображаться на этой диаграмме (рис. 3.23).

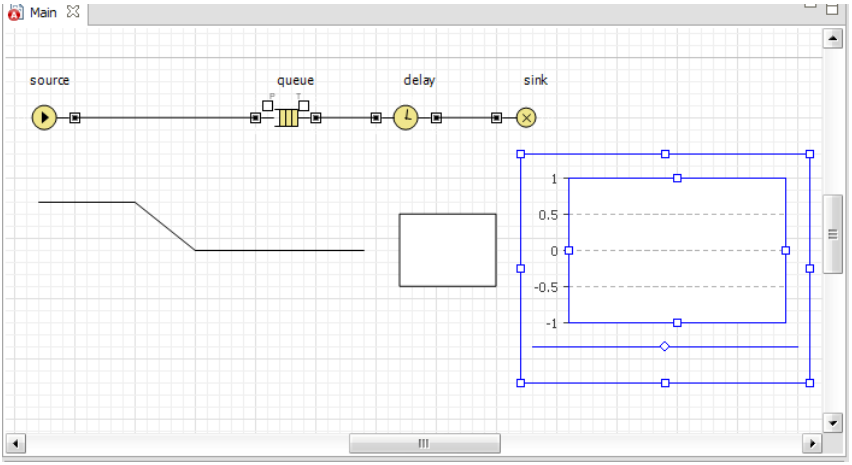


Рис. 3.22. Элемент **Столбиковая диаграмма** на диаграмме класса

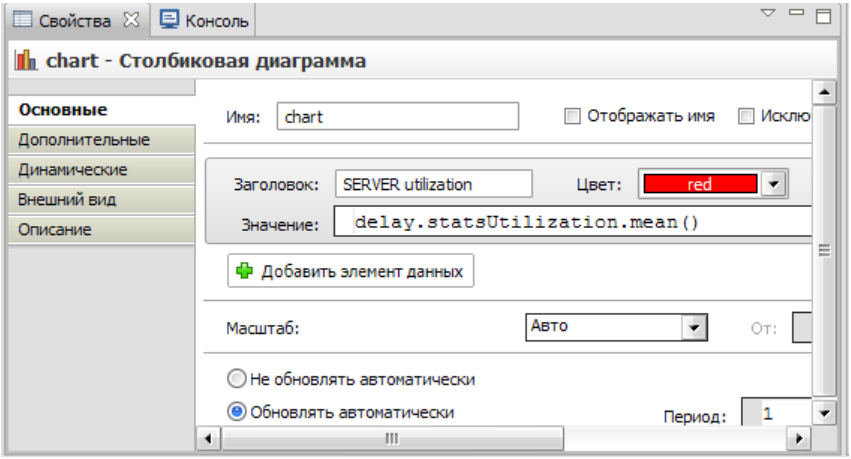


Рис. 3.23. Страница **Основные** панели **Свойства**

4. Измените **Заголовок** на `SERVER utilization`.
5. 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** (на английском языке).
6. Перейдите на страницу **Внешний вид** панели **Свойства** (рис. 3.24).

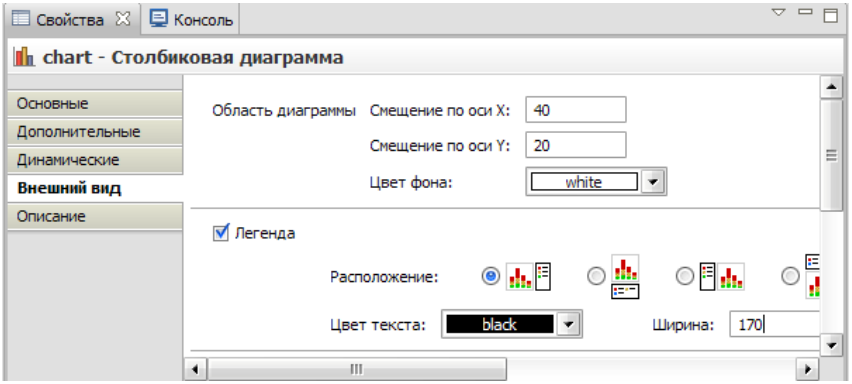


Рис. 3.24. Страница **Внешний вид** панели **Свойства**

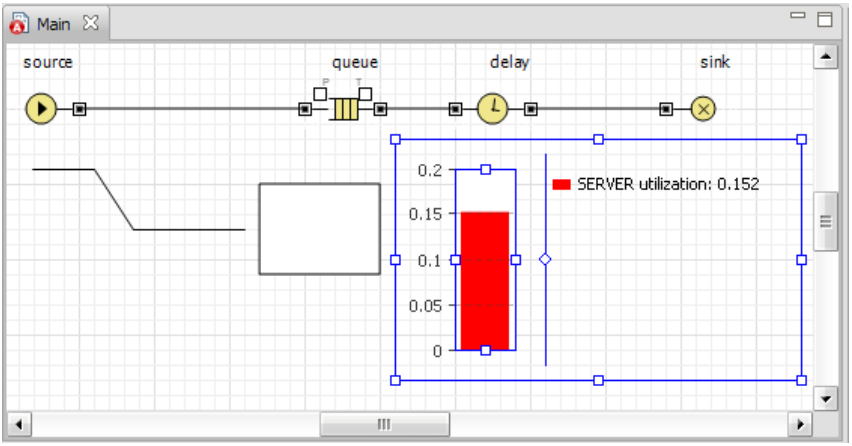


Рис. 3.25. Измененный вид столбиковой диаграммы

7. Выберите первую опцию из набора кнопок **Расположение**, чтобы изменить расположение легенды относительно диаграммы (мы хотим, чтобы она отображалась справа). Размер диаграммы в графическом редакторе измените так, чтобы она приняла вид, показанный на рис. 3.25.

8. Аналогичным образом добавьте еще одну столбиковую диаграмму для отображения средней длины очереди. **Заголовок** и **Значение** измените так, как показано на рис. 3.26.

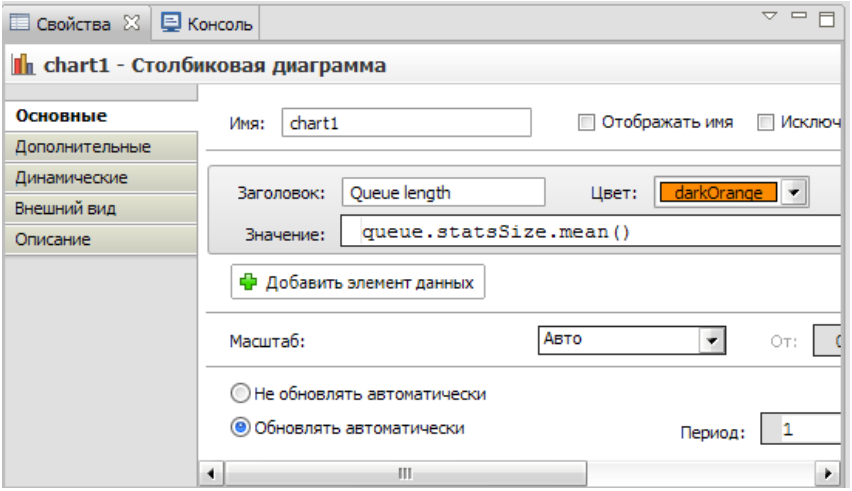


Рис. 3.26. Страница **Основные** панели **Свойства**

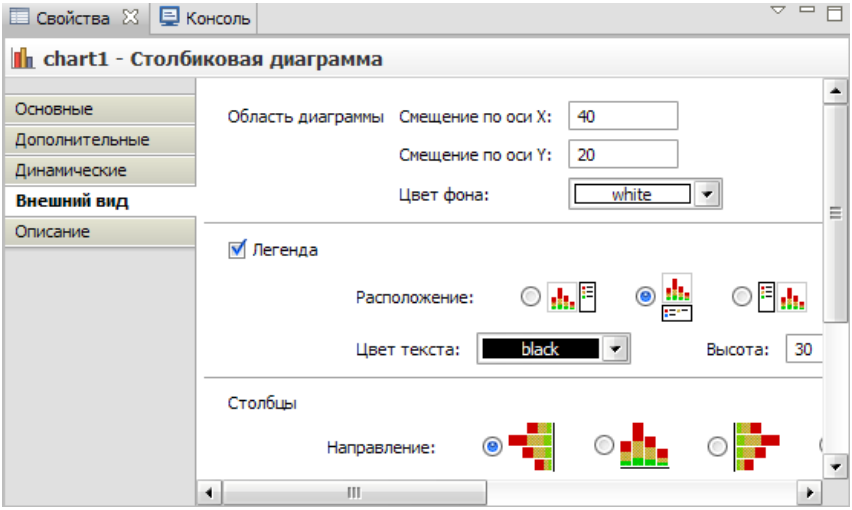


Рис. 3.27. Страница **Внешний вид** панели **Свойства**

9. Здесь queue — это имя нашего объекта **Queue**. У каждого объекта **Queue**, как и объекта **Delay**, также есть встроенный набор данных statsSize, занимающийся сбором статистики использования этого объекта. Функция mean() также возвращает среднее из всех измеренных этим набором данных значений.

10. Перейдите на страницу **Внешний вид** панели **Свойства** и выберите в секции свойств **Направление** первую опцию (рис. 3.27), чтобы столбцы во второй столбиковой диаграмме, расположенной горизонтально, росли влево (рис. 3.28).

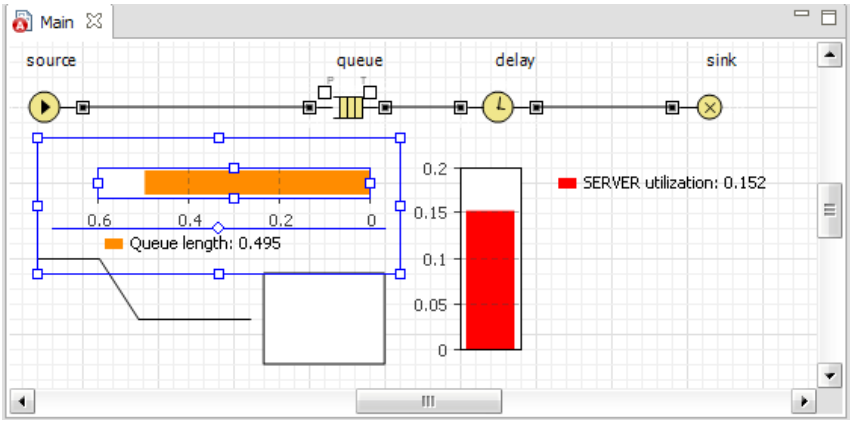


Рис. 3.28. Добавлена вторая столбиковая диаграмма

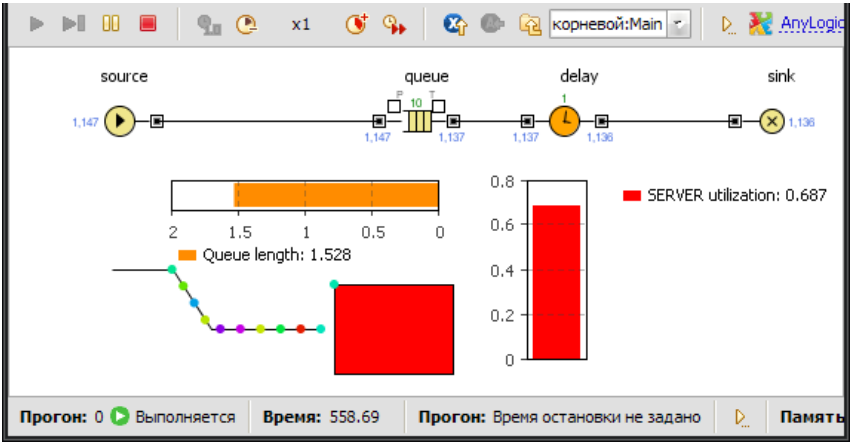


Рис. 3.29. Наблюдение за моделью с двумя столбиковыми диаграммами

11. Запустите модель с двумя столбиковыми диаграммами и понаблюдайте за ее работой (рис. 3.29).

3.1.5. Уточнение модели согласно емкости входного буфера

На рис. 3.29 (снимок сделан, можно сказать, в случайный момент времени, равный 558,69 единицам) видно, что длина очереди равна 10 запросам при установленной максимальной длине 12. Но ведь в постановке задачи емкость буфера была определена в 5 запросов. Нам не удалось до этого построить модель с такой емкостью из-за ошибки (см. рис. 3.12) — невозможности очередного запроса покинуть блок source, так как длина очереди уже была равна 5 запросам. Нам пришлось во избежание этой ошибки увеличить емкость буфера до 12 запросов.

А возможно ли выполнить данное условие постановки задачи средствами AnyLogic? Оказывается, что можно. Причем, различными способами. Уточним модель согласно постановке задачи одним из этих способов.

Объект Queue моделирует очередь заявок, ожидающих приема объектами, следующими за ним в потоковой диаграмме, или же моделирует хранилище заявок общего назначения. При необходимости вы можете задать максимальное время ожидания заявки в очереди. Вы также можете с помощью написанной вами программы извлекать заявки из любых позиций в очереди.

Заявка может покинуть объект Queue различными способами:

обычным способом через порт *out*, когда объект, следующий в блок-схеме за этим объектом, готов принять заявку;

через порт *outTimeout*, если заявка проведет в очереди заданное количество времени (если включен режим таймаута);

через порт *outPreempted*, будучи вытесненной другой поступившей заявкой при заполненной очереди (если включен режим вытеснения);

«вручную», путем вызова функции remove() или removeFirst().

В первом случае объект Queue покидает заявка, находящаяся в самом начале очереди (в нулевой позиции). Если заявка направлена в порт outTimeout или outPreempted, то она должна покинуть объект мгновенно. Если включена опция вытеснения, то объект Queue всегда готов принять новую заявку, в противном случае при заполненной очереди заявка принята не будет.

Поступающие заявки помещаются в очередь в определенном порядке: либо согласно правилу FIFO (в порядке поступления в очередь), либо согласно приоритетам заявок. Приоритет может быть либо явно храниться в заявке, либо вычисляться согласно свойствам заявки и каким-то внешним условиям. Очередь с приоритетами всегда примет новую входящую заявку, вычислит ее приоритет и поместит ее в очередь в позицию, соответствующую ее приоритету. Если очередь будет заполнена, то приход новой заявки вынудит последнюю хранящуюся в очереди заявку покинуть объект через порт outPreempted. *Но если приоритет новой заявки не будет превышать приоритет последней заявки, то тогда вместо нее будет вытеснена именно эта новая заявка*.

Для выполнения условия постановки задачи воспользуемся последним способом вытеснения. Все запросы, вырабатываемые блоком source, имеют один и тот же приоритет. Поэтому при полном заполнении накопителя (5 запросов) теряться будет последний запрос.

Уточните модель:

1. Выделите блок queue. На странице **Основные** панели **Свойства** измените **Вместимость** с 12 на 5 запросов.
2. На этой же странице установите **Разрешить вытеснение**.
3. Для уничтожения потерянных запросов вследствие полного заполнения накопителя необходимо добавить второй блок sink. Откройте в **Палитре** библиотеку Enterprise Library и перетащите блок sink на диаграмму (рис. 3.30).

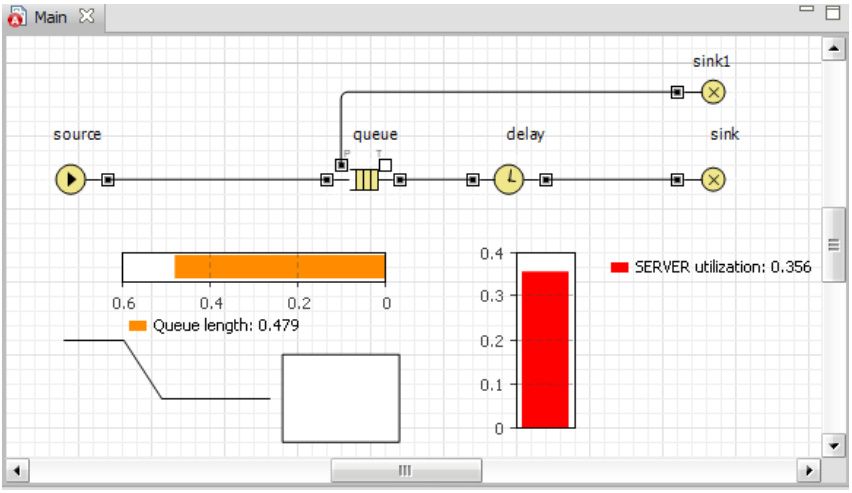


Рис. 3.30. Уточненная модель

4. Соедините порт outPreempted блока queue с входным портом InPort блока sink1. Чтобы соединить порты блоков друг с другом, сделайте двойной щелчок мышью по одному порту, например, outPreempted, затем последовательно щелкните в тех местах диаграммы, где вы хотите поместить точки изгиба соединителя. Завершите процесс соединения двойным щелчком мыши по второму порту.

5. После двойного щелчка по второму порту вы увидите, что появится соединитель. Если выделить его мышью, то в том случае, если вы правильно соедините порты, конечные точки соединителя должны будут подсветиться зелеными точками. Если вы не увидите этого, то значит, что точки не были помещены точно внутрь портов, и их нужно будет туда передвинуть.

6. Запустите уточненную модель и понаблюдайте за ее работой. Сравните рис. 3.31 с рис. 3.12. На рис. 3.31 видно, что запросы при длине очереди в 5 запросов теряются, и ошибки при этом не возникает. Таким образом, модель по условию ограничения емкости входного буфера и значениям других параметров соответствует постановке задачи.

Однако согласно постановке задачи требуется определить математическое ожидание времени обработки одного запроса и математическое ожидание вероятности обработки запросов.

Приступим к реализации этих требований.

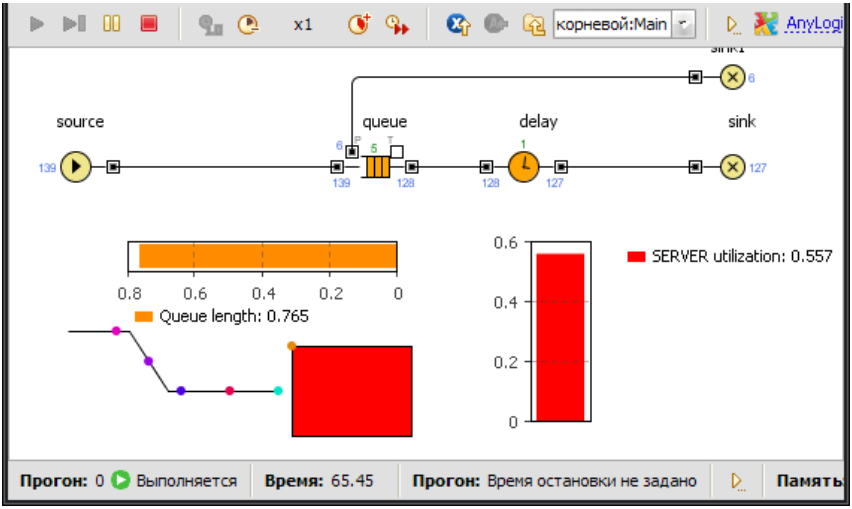


Рис. 3.31. Работа модели согласно постановке задачи

3.1.6. Сбор статистики по показателям обработки запросов

Entity (заявка) являются базовым классом для всех заявок, которые создаются и работают с ресурсами в процессе, описанном вами с помощью диаграммы из объектов библиотеки AnyLogic Enterprise Library. Entity по существу является обычным Java классом с теми функциональными возможностями, которые необходимы и достаточны для обработки и отображения анимации заявки объектами Enterprise Library. Эти функциональные возможности можно расширить добавлением дополнительных полей и методов и работой с ними из объектов диаграммы, описывающей моделируемый процесс.

Согласно постановке задачи, как уже отмечалось, нужно определять математическое ожидание времени и вероятности обработки запросов сервером.

Математическое ожидание или среднее время обработки одного запроса определяется как отношение суммарного времени обработки n запросов к их количеству, т. е. к n. Для определения суммарного времени нужно знать время обработки i-го запроса. Для этого введем дополнительные поля: `time_vxod` – время входа запроса в буфер сервера, `time_vixod` – время выхода запроса с сервера (входа в блок sink). Тогда

```
time_obrabotki=time_vixod-time_vxod
```

Вероятность обработки запросов сервером определяется как отношение количества обработанных запросов к количеству всех поступивших запросов. Значит, нужно вести счет запросов на выходе источника запросов и на выходе с сервера (входе в блок sink). Для этого также введем дополнительные поля: `col_vxod` — количество поступивших всего запросов, `col_vixod` — количество обработанных сервером запросов. Тогда

```
ver_obrabotki=col_vixod/col_vxod
```

Замечание. Ничего необычного во введенных дополнительных полях нет. Это параметры реальных элементов потоков, в данном случае запросов. AnyLogic предоставляет возможность создавать запросы с теми параметрами, которые необходимы в модели.

3.1.6.1. Создание нестандартного класса заявок

Для ввода в запросы дополнительных полей необходимо создать нестандартный класс заявки.

Создайте класс заявок `Inquiry`.

1. В панели **Проект** щелкните правой кнопкой мыши элемент модели верхнего уровня дерева и выберите **Создать/Java класс** из контекстного меню.
2. Появится диалоговое окно **Новый Java класс** (рис. 3.32). В поле **Имя:** введите имя нового класса: `Inquiry`.
3. В поле **Базовый класс:** выберите из выпадающего списка `com.xj.anylogic.libraries.enterprise.Entity` в качестве базового класса. Щелкните кнопку **Далее**.

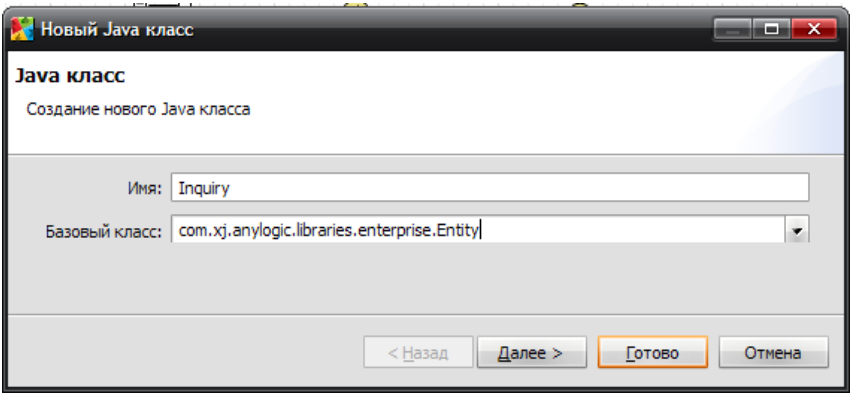


Рис. 3.32. Диалоговое окно **Мастера создания Новый Java класс**

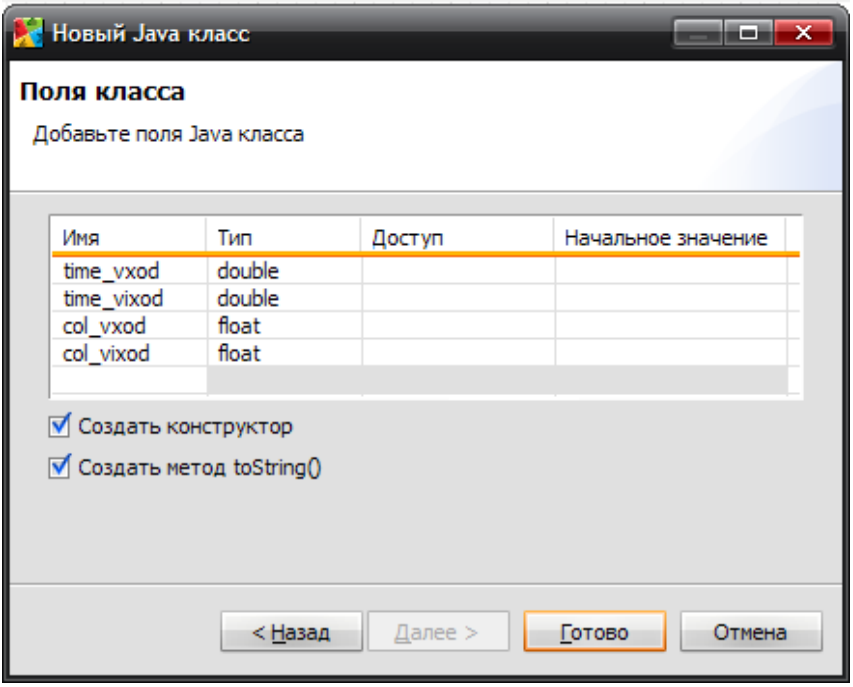


Рис. 3.33. Вторая страница **Мастера создания Новый Java класс**

4. Появится вторая страница **Мастера создания Java класса** (рис. 3.33). Добавьте поля Java класса: `time_vxod` типа `double`, `time_vixod` типа `double`, `col_vxod` типа `float`, `col_vixod` типа `float`. Типы полей выбираются из выпадающего списка. Начальные значения всех параметров, поскольку не указаны, по умолчанию будут установлены равными нулю.

5. Оставьте выбранными флажки **Создать конструктор** и **Создать метод toString ()**. Тогда у класса будут созданы сразу два конструктора: один, по умолчанию, без параметров, и второй, с параметрами, инициализирующими поля класса. Эти конструкторы используются объектами, создающими новые заявки, такие, как Source.

6. Щелкните кнопку **Готово**. Вы увидите редактор кода, в котором будет показан автоматически созданный код вашего Java класса (рис. 3.34). Изучите этот код и выясните, как можно самостоятельно добавлять в класс заявки новые поля и методы. Закройте редактор, щелкнув крестик в закладке рядом с его названием.

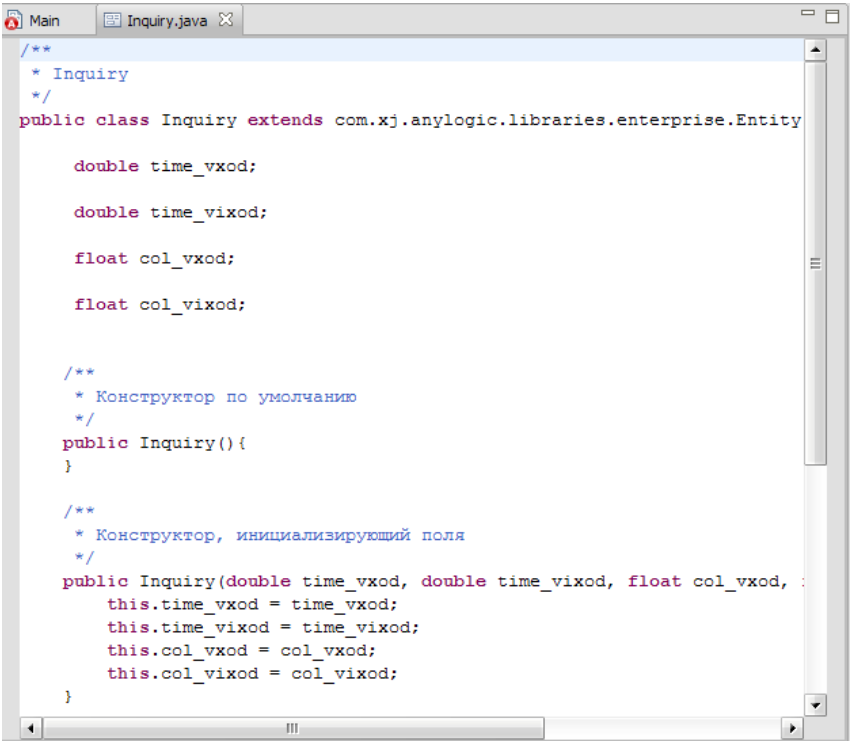


Рис. 3.34. Окно редактора кода созданного нестандартного класса

3.1.6.2. Добавление элементов статистики

Для сбора статистических данных о времени поступления запросов и времени завершения обработки сервером необходимо добавить элемент статистики. Этот элемент будет запоминать соответствующие значения времен для каждого запроса. На основе этого он предоставит пользователю стандартную статистическую информацию (среднее, минимальное, максимальное из измеренных значений, среднеквадратичное отклонение, доверительный интервал для среднего и т.д.). Добавьте элемент сбора статистики.

1. Чтобы добавить объект сбора данных гистограммы на диаграмму, перетащите элемент **Данные гистограммы** с палитры **Статистика** на диаграмму активного класса.
2. Задайте свойства элемента (рис. 3.35):

измените **Имя:** на `time_obrabotki`;

сделайте **Кол-во интервалов:** равным `50`;

задайте **Нач. размер интервала:** `0.01`.

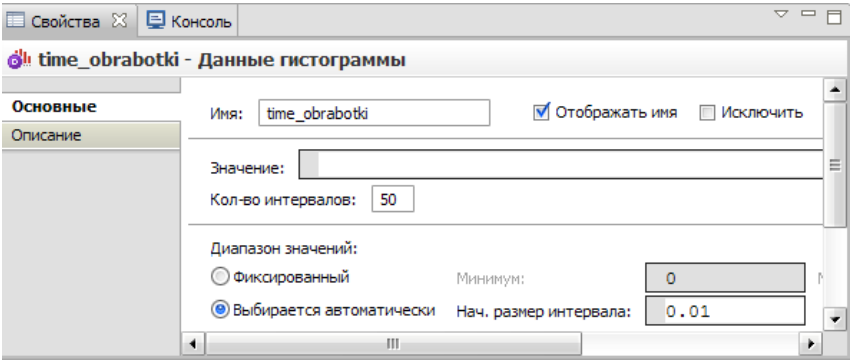


Рис. 3.35. Элемент сбора статистики о времени обработки запросов

Добавьте еще элемент сбора статистики для определения вероятности обработанных запросов.

1. Перетащите элемент **Данные гистограммы** с палитры **Статистика** на диаграмму активного класса.
2. Задайте свойства элемента (рис. 3.36):

измените **Имя:** на `ver_obrabotki`;
сделайте **Кол-во интервалов:** равным `50`;
задайте **Нач. размер интервала:** `0.01`.

3.1.6.3. Изменение свойств объектов диаграммы

Чтобы создавать заявки нестандартного типа, как в нашем случае `Inquiry`, вам нужно поместить вызов конструктора этого типа в поле **Новая заявка** объекта Source. Но, несмотря на то, что заявки в потоке теперь и будут типа `Inquiry`, остальные объекты диаграммы будут продолжать их считать заявками типа `Entity`.

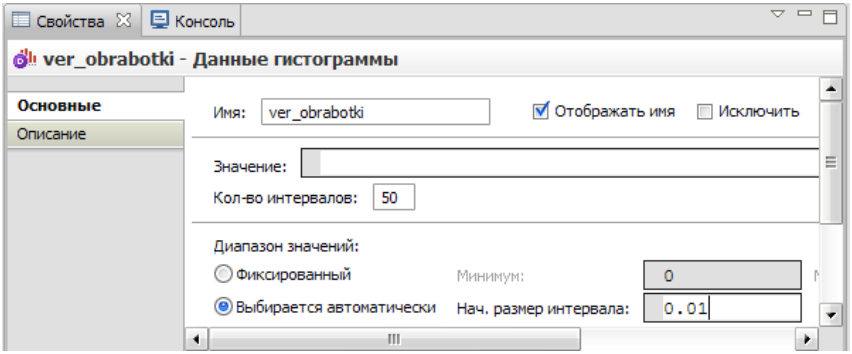


Рис. 3.36. Элемент сбора статистики о вероятности обработки запросов

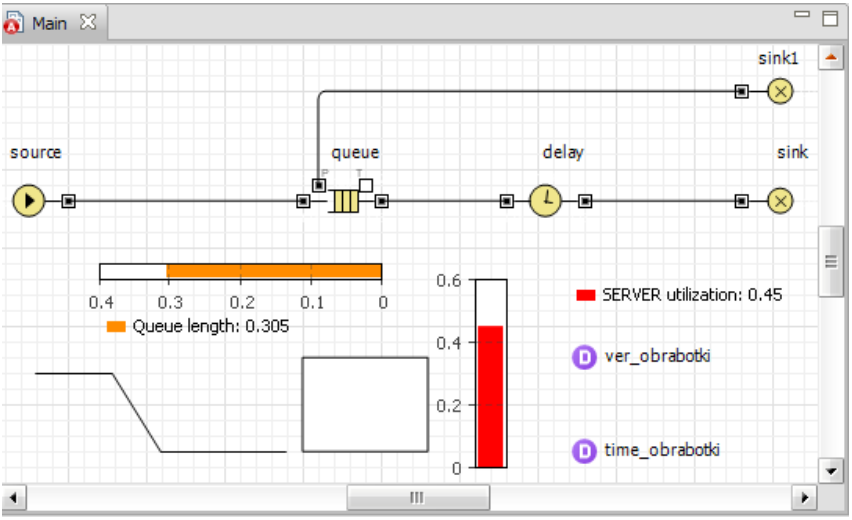


Рис. 3.37. Диаграмма после добавления элементов сбора статистики

Поэтому они не позволят явно обращаться к дополнительным полям класса `Inquiry`. Чтобы разрешить доступ к полям вашего нестандартного класса заявки в коде динамических параметров объектов потоковой диаграммы, вам нужно указать имя нестандартного класса заявки в качестве **Класса заявки** этого объекта. В нашей потоковой диаграмме с учетом блока **source** всего пять объектов. Измените их свойства.

1. Измените свойства объекта **source** (рис. 3.38):

введите `Inquiry` в поле **Класс заявки:**. Это позволит напрямую обращаться к полям класса заявки `Inquiry` в коде динамических параметров этого объекта;

введите `new Inquiry()` в поле **Новая заявка**. Теперь этот объект будет создавать заявки нашего типа `Inquiry`;

введите `entity.time_vxod=time();` в поле **Действие при выходе**. Код будет сохранять время создания заявки-запроса в переменной `time_vxod` нашего класса заявки `Inquiry`.

Функция `time()` возвращает текущее значение модельного времени.

2. Измените свойства объекта queue:

введите `Inquiry` в поле **Класс заявки:**.

3. Измените свойства объекта delay:

введите `Inquiry` в поле **Класс заявки:**.

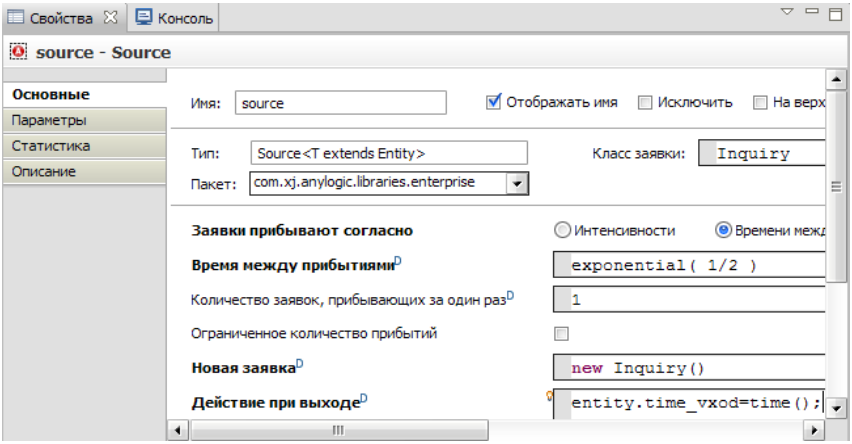


Рис. 3.38. Объект source с измененными свойствами

4. Измените свойства объекта sink1:
введите `Inquiry` в поле **Класс заявки:**.
5. Измените свойства объекта sink (рис. 3.39):
введите `Inquiry` в поле **Класс заявки:**;
введите в поле **Действие при входе**:

```
time_obrabotki.add(time()-entity.time_vxod);
```

Этот код добавляет время обработки одного запроса в объект сбора данных гистограммы `time_obrabotki`. Данное время определяется как разность между текущим модельным временем `time()` и временем входа запроса в модель. `add` — встроенная функция добавления элемента в массив.

```
entity.col_vixod=sink.count();
entity.col_vxod=source.count();
```

Эти коды заносят количество запросов, вошедших в блок sink и вышедших из блока source соответственно. `count()` — встроенная функция этих блоков, возвращает количество вошедших в блок sink и количество вышедших из блока source заявок.

```
ver_obrabotki.add(entity.col_vixod/entity.col_vxod);
```

Этот код добавляет относительную долю обработанных запросов в объект сбора данных гистограммы `ver_obrabotki` при поступлении каждого обработанного запроса в блок sink. На основе множества таких относительных долей определяется математическое ожидание вероятности обработки запросов сервером.

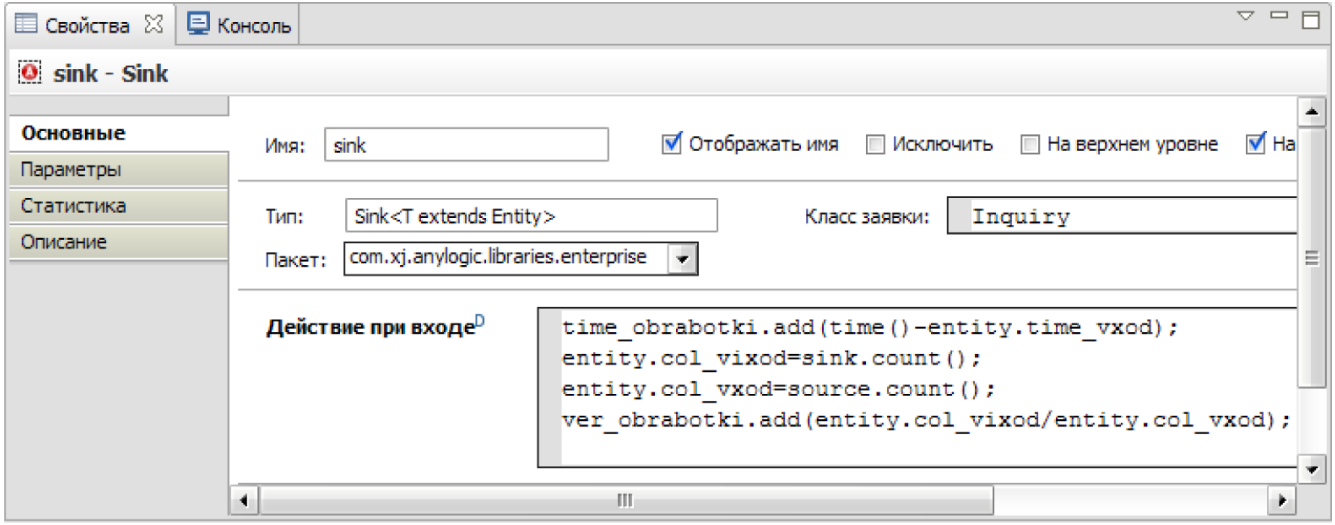


Рис. 3.39. Объект **sink** с измененными свойствами

3.1.6.4. Удаление и добавление новых полей класса заявок

Обратите внимание, что вам не пришлось использовать поле `time_vixod`, так как вместо него была использована функция `time()`, возвращающая, как вам уже известно, текущее значение модельного времени.

Удалите поле `time_vixod`.

1. В поле **Проект** дважды щелкните кнопку `Inquiry`. Откроется редактор кода (см. рис. 3.34).
2. Удалите обычным образом все, что касается `time_vixod`. Получите код, представленный на рис. 3.40.
3. Закройте редактор кода.

Main | Inquiry.java

```
/**
 * Inquiry
 */
public class Inquiry extends com.xj.anylogic.libraries.enterprise.Enti

    double time_vxod;

    float col_vixod;

    float col_vxod;


    /**
     * Конструктор по умолчанию
     */
    public Inquiry(){
    }

    /**
     * Конструктор, инициализирующий поля
     */
    public Inquiry(double time_vxod, float col_vixod, float col_vxod){
        this.time_vxod = time_vxod;
        this.col_vixod = col_vixod;
        this.col_vxod = col_vxod;
    }

    @Override
    public String toString() {
        return
            "time_vxod = " + time_vxod +" " +
            "col_vixod = " + col_vixod +" " +
            "col_vxod = " + col_vxod +" ";
    }
```

Рис. 3.40. Редактор кода после удаления поля `time_vixod`

Из процедуры удаления поля следует, что так же можно вводить новые дополнительные поля нестандартного класса заявок.

Итак, все условия постановки задачи выполнены. Запустите модель. Выберите виртуальное время. Наблюдайте за работой модели. Используйте при этом и окна инспекта (рис. 3.41).

3.1.7. Добавление параметров и элементов управления

Активный объект может иметь параметры. Параметры обычно используются для задания статических характеристик объекта. Но значения параметров при необходимости можно изменять во время работы модели.

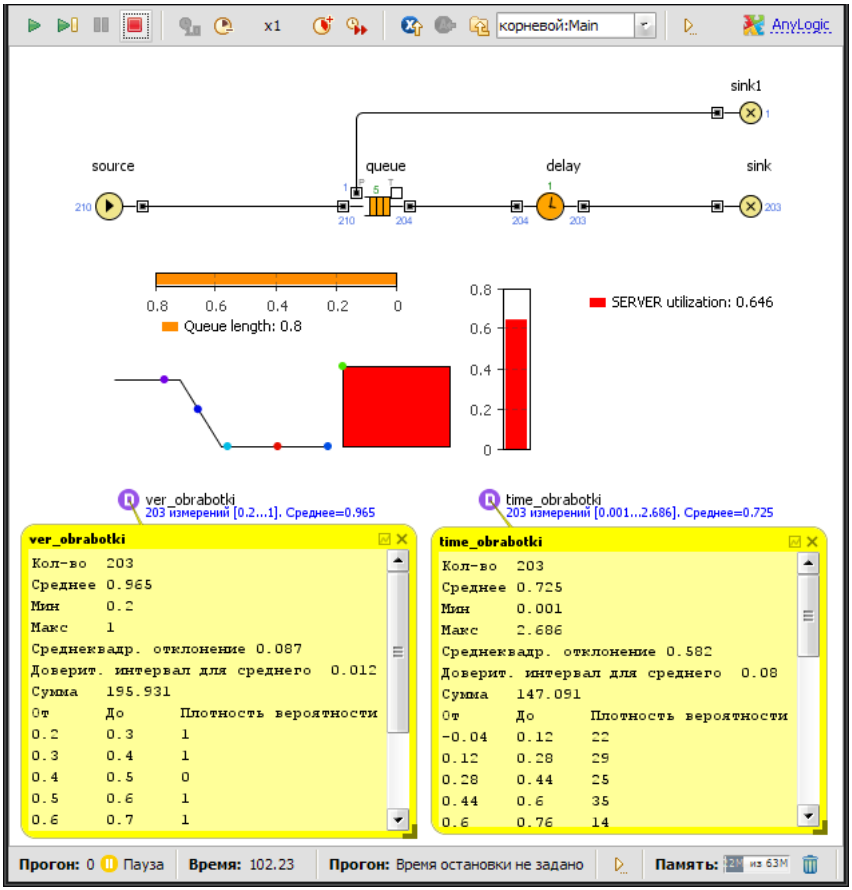


Рис. 3.41. Фрагмент работы модели

Для этого нужно написать код обработчика события, то есть действий, которые должны выполняться при изменении значения параметра.

Создайте параметр `time_mean` объекта `delay`.

1. В **Палитре** выделите **Основная**.
2. Перетащите элемент **Параметр** на диаграмму класса `Main` и разместите сверху объекта `delay`, чтобы было видно, к какому объекту относится параметр.
3. Перейдите на страницу **Основные** панели **Свойства** (рис. 3.42).
4. В поле **Имя** введите имя параметра `time_mean` (среднее время). По этому имени параметр будет доступен из кода.
5. Задайте тип параметра `double`.
6. В поле **Значение по умолчанию** установите 3. Если значение не будет задано явно, то по правилам Java оно будет равно нулю.
7. Выделите объект `delay`.
8. На странице **Основные** панели **Свойства** в поле **Время задержки** вместо выражения `exponential(1/3)` введите `exponential(1/time_mean)`.

Пусть вы хотите изменять среднее время обработки запросов `time_mean` в ходе моделирования. Используйте для этого элемент управления — бегунок.

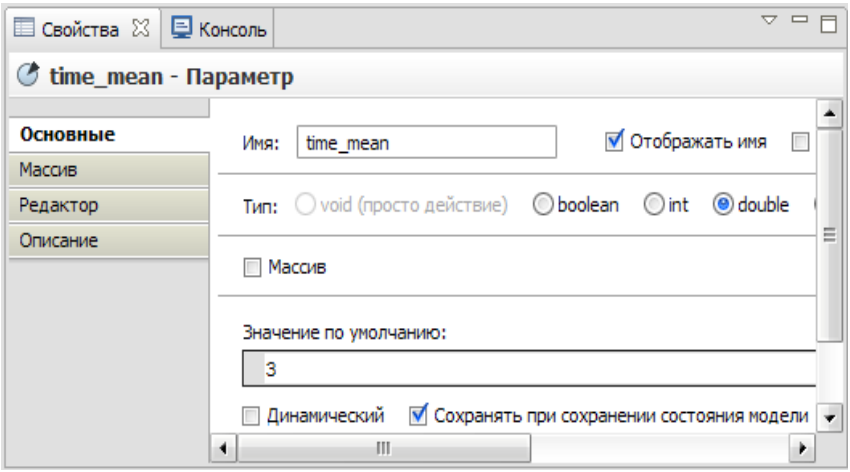


Рис. 3.42. Окно установки свойств элемента **Параметр**

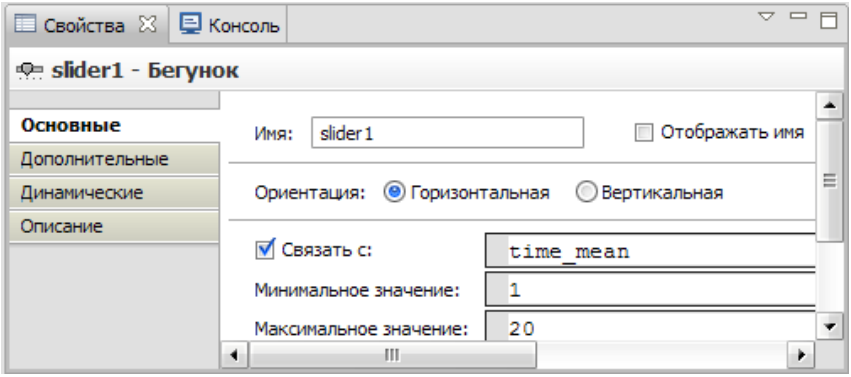


Рис. 3.43. Окно установки свойств элемента управления **Бегунок**

1. Откройте палитру **Элементы управления** и перетащите элемент **Бегунок** из палитры на диаграмму класса `Main`.
2. Поместите бегунок под параметром `time_mean`, чтобы было понятно, что с помощью этого бегунка будет меняться среднее время обработки запросов объектом delay.
3. Пусть вы хотите варьировать среднее время от 1 до 20. Поэтому введите 1 в поле **Минимальное значение**, а 20 — в поле **Максимальное значение** (рис. 3.43).
4. Установите флажок **Связать с:** и в активизированное поле введите `time_mean`.

Пусть теперь вы хотите также изменять емкость буфера в ходе моделирования. Используйте для этого также бегунок.

1. Откройте палитру **Элементы управления** и перетащите элемент **Бегунок** из палитры на диаграмму класса `Main`.
2. Поместите бегунок под объектом queue, чтобы было понятно, что с помощью этого бегунка будет меняться вместимость данного объекта, имитирующего входной буфер.
3. Пусть вы хотите варьировать емкость буфера от `0` до `15` запросов. Поэтому введите `15` в поле **Максимальное значение**.
4. Установите флажок **Связать с:** и в активизированное поле введите `queue.capacity` (рис. 3.44).
5. Запустите модель. Теперь вы можете изменять в процессе моделирования емкость входного буфера и среднее время обработки запросов с помощью бегунков. Можете также командой **Приостановить** приостановить работу модели, изменить значения параметров, а затем продолжить моделирование.

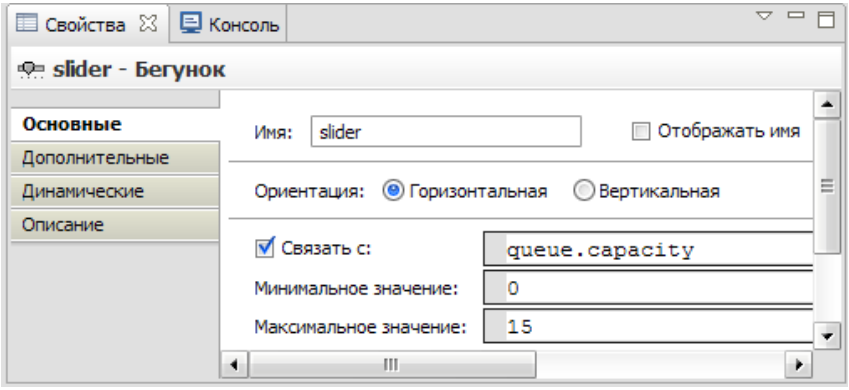


Рис. 3.44. Окно установки свойств элемента управления **Бегунок**

3.1.8. Добавление гистограмм

Теперь добавим на диаграмму нашего потока гистограмму, которая будет отображать собранную временную статистику.

1. Перетащите элемент **Гистограмма** из палитры **Статистика** в то место графического редактора, куда хотите ее поместить.
2. Укажите, какой элемент сбора данных хранит данные, которые вы хотите отображать на гистограмме: щелкните мышью кнопку **Добавить данные** и введите в поле **Данные** имя соответствующего элемента: `time_obrabotki` (рис. 3.45).

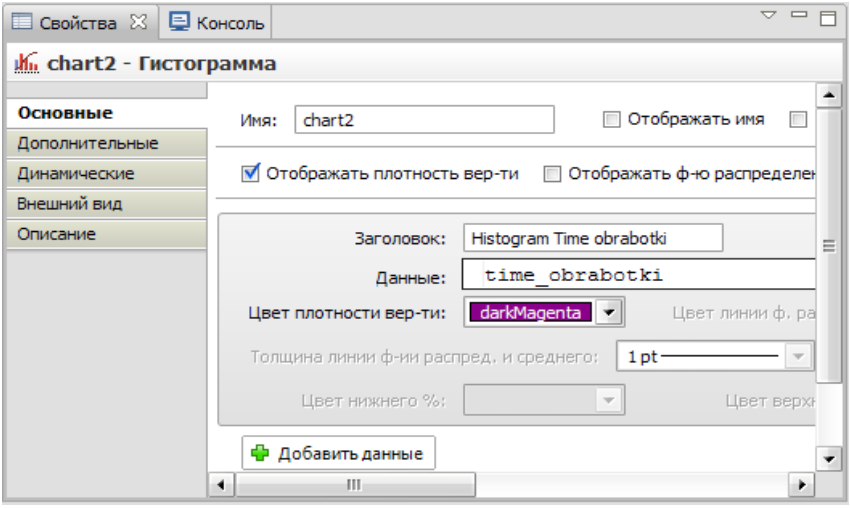


Рис. 3.45. Окно установки свойств элемента **Гистограмма**

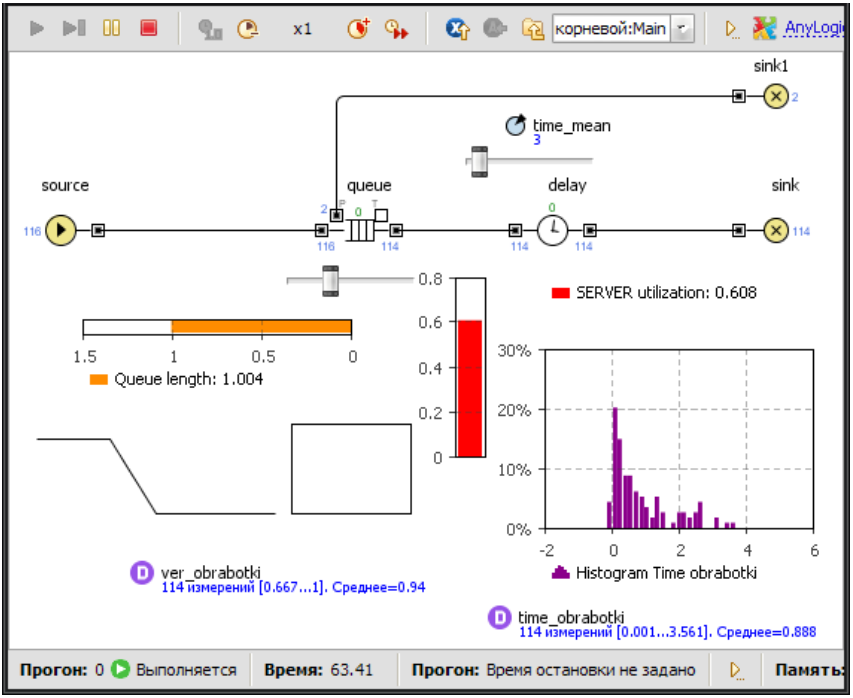


Рис. 3.46. Фрагмент работы модели с элементами управления и гистограммой

3. В поле **Заголовок:** введите `Histogram Time obrabotki`.
4. Запустите модель. Фрагмент работы показан на рис. 3.46.

3.2. Модель функционирования маршрутизатора сообщений

3.2.1. Постановка задачи

На маршрутизатор поступают сообщения через случайные промежутки времени от n_1 абонентов со средними интервалами времени $T_i = T / n_1$. Сообщения могут быть трех категорий n_2 с вероятностями появления $p_{к1}, p_{к2}, p_{к3}$ $(p_{к1} + p_{к2} + p_{к3} = 1)$ и вычислительными сложностями их обработки S_1, S_2, S_3 операций соответственно. Передаются сообщения по k направлениям с вероятностями $p_{н1}, p_{н2}, p_{н3}$ $(p_{н1} + p_{н2} + p_{н3} = 1)$.

Маршрутизатор имеет входной буфер 1 емкостью L_1 байт для хранения сообщений, ожидающих обработки. Сообщения 1-й категории обладают относительным приоритетом по отношению

к сообщениям остальных категорий при обработке. В накопителе сообщения размещаются согласно приоритетам. Сообщения обрабатываются с производительностью Q операций/с. При заполнении буфера 1, поступающие сообщения теряются.

После обработки сообщения в зависимости от направления передачи поступают в соответствующие буферы, стоящие на входах каждого i-го направления связи, $i=\overline{1,k}$. Каждый буфер имеет емкость L_{2i} байт, $i=\overline{1,k}$. В случае заполнения буфера направления сообщение теряется. Из буферов сообщения передаются по своим направлениям. Каждое направление имеет n_{3i} каналов связи. Скорость передачи сообщений по каналам связи каждого из направлений $V_{\text{пі}}$ бит/с, $i=\overline{1,k}$.

3.2.2. Исходные данные

$\text{Exponential}(T_1)=\text{Exponential}(T_2)\ldots=\text{Exponential}(T_{10})=$

$=\text{Exponential}(18/n_1);\ n_1=10;\ n_2=3;$

$k=3;\ p_{\text{н1}}=0,4;\ p_{\text{н2}}=0,35;\ p_{\text{н3}}=0,25;$

$n_2=3;\ p_{\text{к1}}=0,3;\ p_{\text{к2}}=0,2;\ p_{\text{к3}}=0,5;\ Q=3600$ оп/с; $L_1=5000000;$

$\text{Normal}(S_1, S_{\text{o1}})=\text{Normal}(530000, 6100);$

$\text{Normal}(S_2, S_{\text{o2}})==\text{Normal}(860000, 5000);$

$\text{Normal}(S_3, S_{\text{o3}})=\text{Normal}(660000, 7000);$

$V_{\text{пі}}=5000$ бит/с, $n_{3i}=1,\ L_{2i}=200000,\ i=\overline{1,3}.$

Интервалы поступления сообщений распределяются по экспоненциальному закону (Exponential), а вычислительные сложности сообщений — по нормальному закону (Normal).

3.2.3. Задание на исследование

Разработать имитационную модель функционирования маршрутизатора с целью исследования влияния емкости входного буфера 1, интервала времени поступления сообщений, их вычислительных сложностей и других параметров на среднее время передачи сообщений, коэффициент пропускной способности по категориям сообщений, направлениям и в целом сообщений всех категорий через маршрутизатор, а также оценки необходимости применения мер по повышению эффективности его использования.

3.2.4. Формализованное описание модели

Представим маршрутизатор как систему массового обслуживания (рис. 3.47).

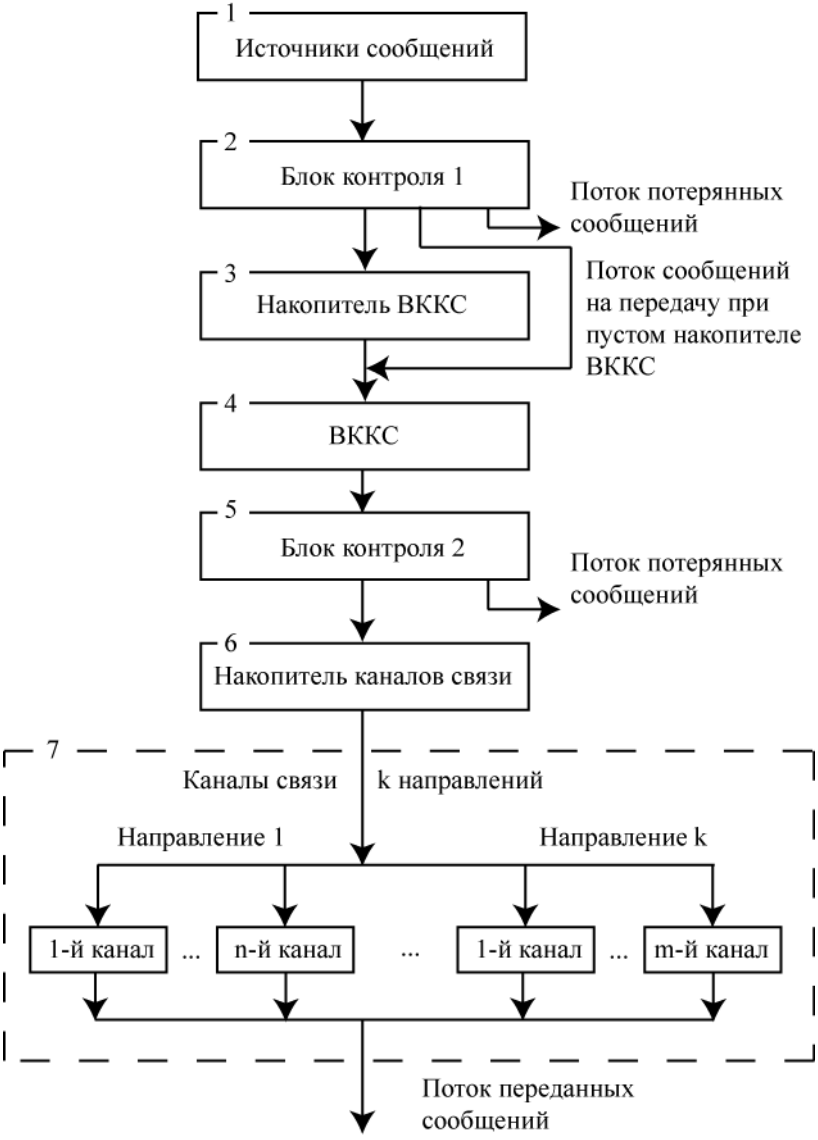


Рис 3.47. Маршрутизатор как система массового обслуживания

Видно, что маршрутизатор представляет собой многофазную многоканальную систему массового обслуживания разомкнутого типа с ограниченными емкостями буферов (накопителей).

Назначение и функции выделенных в структуре маршрутизатора блоков будет рассмотрено позже в ходе изложения технологии построения модели в AnyLogic.

Следует иметь в виду, что возможны и другие варианты декомпозиции маршрутизатора как системы. Тем более, что нами взята с учебной точки зрения одна из простых его схем.

Сообщения, поступающие на маршрутизатор, должны иметь следующие параметры:

`NumIst` — номер источника сообщений;

`NumKat` — номер категории сообщения;

`NumNapr` — номер направления передачи сообщения;

`time_post` — время поступления сообщения;

`Dlina` — длина сообщения, байт;

`TimeObrabotki` — время обработки сообщения;

`TimePered` — время передачи сообщения.

Эти параметры получаются путем розыгрыша по следующим исходным данным:

`KolIst` — количество источников сообщений;

`TimeMean` — среднее время поступления сообщений от одного источника;

`Kat={kat1, kat2, kat3}` — вероятности распределения видов категорий сообщений, `kat1+kat2+kat3=1`;

`Napr={napr1, napr2, napr3}` — вероятности распределения направлений передачи сообщений, `napr1+napr2+napr3=1`;

`DlKat={dlKat1, dlKat2, dlKat3}` — средние длины сообщений, байт, первой, второй и третьей категорий соответственно;

`DlKat0={dlKat01, dlKat02, dlKat03}` — стандартные отклонения длин сообщений, байт, первой, второй и третьей категорий соответственно;

`Proizvod` — производительность маршрутизатора при обработке сообщения, оп/с;

`Skor_pered={skor_pered1, skor_pered2, skor_pered3}` — среднее время передач сообщений по первому, второму и третьему направлениям соответственно;

Kol_kan_Napr={kol_kan_Napr1, kol_kan_Napr2, kol_kan_Napr3} — количество каналов связи первого, второго и третьего направлений соответственно.

В ходе моделирования собирается следующая статистика:

Kol_post_kat={kol_post_kat1, kol_post_kat2, kol_post_kat3, kol_post} — количество поступивших сообщений первой, второй, третьей и всех категорий соответственно;

Kol_pered_kat={kol_pered_kat1, kol_pered_kat2, kol_pered_kat3, kol_pered} — количество переданных сообщений первой, второй, третьей и всех категорий соответственно;

Kol_post_Napr={kol_post_Napr1, kol_post_Napr2, kol_post_Napr3} — количество поступивших сообщений для передачи по первому, второму и третьему направлениям соответственно;

Kol_pered_Napr={kol_ pered _ Napr 1, kol_ pered _ Napr 2, kol_ pered _ Napr 3} — количество переданных сообщений по первому, второму и третьему направлениям соответственно.

По этим статистическим данным рассчитываются:

Koef_prop_kat={koef_prop_kat1, koef_prop_kat2, koef_prop_kat3} — коэффициенты пропускной способности по категориям сообщений;

Koef_prop_Napr={koef_prop_Napr1, koef_prop_Napr2, koef_prop_Napr3} — коэффициенты пропускной способности по направлениям;

koef_prop_sposob — коэффициент пропускной способности маршрутизатора.

В модели также для выполнения условий ограничения емкостей буферов необходимо использовать:

emkost_bufera_1 — емкость входного буфера, байт;

tek_emkost_bufera1 — текущая емкость входного буфера, байт;

Emkost_bufera_Napr={emkost_bufera_Napr1, emkost_bufera_Napr2, emkost_bufera_Napr3} — емкости на входах каналов связи первого, второго и третьего направлений соответственно, байт;

Tek_emkost_bufera_Napr={tek_emkost_bufera_Napr1, tek_emkost_bufera_Napr2, tek_emkost_bufera_Napr3} — текущие емкости на входах каналов связи первого, второго и третьего направлений соответственно, байт.

3.2.5. Ввод исходных данных

Для ввода исходных данных используем элемент **Параметр** и элемент управления **Бегунок** (см. п. 3.1.7). Элемент **Параметр** может размещаться на диаграмме класса активного объекта или эксперимента. Используем первое. В этом случае в ходе моделирования будут видны значения параметров, которые с помощью бегунков можно будет изменять.

Исходные данные разобьем на две группы. В первую группу, назовем ее Initial_data_1, включим характеристики сообщений, а во вторую группу с именем Initial_data_2 — характеристики непосредственно маршрутизатора.

1. Выполните команду **Файл/Создать/Модель** на панели инструментов. Появится диалоговое окно **Новая модель** (см. рис. 3.1).
2. Задайте имя новой модели. В поле **Имя модели** введите `Router`. Выберите каталог, в котором будут сохранены файлы модели.
3. Щелкните кнопку **Далее**. Откроется вторая страница **Мастера создания модели** (см. рис. 3.2). Выберите **Начать создание модели «с нуля».** Щелкните кнопку **Далее.**
4. В **Палитре** выделите **Презентация**. Перетащите элемент **Скругленный прямоугольник** в нужное место.
5. Перетащите элемент **text** и на странице **Основные** панели **Свойства** в поле **Текст:** введите `Initial_data_1`.
6. В **Палитре** выделите **Основная**. Перетащите элементы **Параметр** на элемент с именем `Initial_data_1` и разместите их так, как показано на рис. 3.48. Значения свойств установите согласно табл. 3.1.
7. Для изменения размера прямоугольника выделите его. Вы увидите несколько маленьких голубых прямоугольников по краям фигуры. Они будут обозначать те точки, которые вы сможете перетаскивать мышью для изменения размеров фигуры.
8. Перетащите мышью (с нажатой левой кнопкой) прямоугольник, лежащий на той границе фигуры, которую вы хотите передвинуть (либо, если же вы хотите передвинуть сразу две границы — то прямоугольник, находящийся в углу фигуры, образованном этими двумя границами) в требуемое местоположение.
9. В **Палитре** выделите **Элементы управления**. Перетащите элементы **Бегунок** на элемент с именем `Initial_data_1` и разместите их так, как показано на рис. 3.48. Значения свойств установите также согласно табл. 3.1.

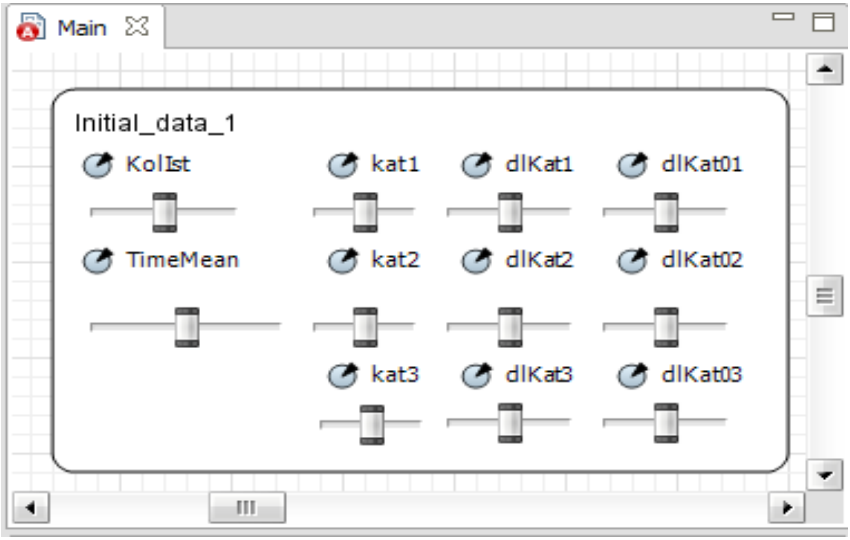


Рис. 3.48. Размещение элементов **Параметр** и **Бегунок** на Initial_data_1

Таблица 3.1

Элементы и их свойства					
Параметр			Бегунок		
Имя	Тип	Значение по умолча-нию	Связать с	Мини-мальное значение	Макси-мальное значение
Kol_Ist	int	10	Kol_Ist	2	100
Time_mean	double	18	Time_mean	1	50
kat1	double	0.5	kat1	0	1
kat2	double	0.3	kat2	0	1
kat3	double	0.2	kat3	0	1
dlKat1	double	53000	dlKat1	10000	100000
dlKat2	double	86000	dlKat2	10000	100000
dlKat3	double	66000	dlKat3	10000	100000
dlKat01	double	6100	dlKat01	2000	10000
dlKat02	double	5000	dlKat02	2000	10000
dlKat03	double	7000	dlKat03	2000	10000

1. В **Палитре** выделите **Презентация**. Перетащите элемент **Скругленный прямоугольник** в нужное место.

2. Перетащите элемент **text** и на странице **Основные** панели **Свойства** в поле **Текст:** введите `Initial_data_2`. Вводить текст можно и на русском языке, что мы и будем делать далее.

3. В **Палитре** выделите **Основная**. Перетащите элементы **Параметр** и **Простая переменная** на элемент с именем `Initial_data_2`. Разместите их так, как показано на рис. 3.49. Значения свойств установите согласно табл. 3.2.

4. В **Палитре** выделите **Элементы управления**. Перетащите нужное число элементов **Бегунок** на элемент с именем `Initial_data_2` и разместите их так, как показано на рис. 3.49. Значения свойств установите также согласно табл. 3.2.

Таблица 3.2

Элементы и их свойства					
Параметр			Бегунок		
Имя	Тип	Значение по умолчанию	Связать с	Минимальное значение	Максимальное значение
emkost_bufera_1	int	5000000	emkost_bufera_1	1000000	5000000
tek_emkost_bufera_1	int	0			
Proizvod	double	36000	Proizvod	10000	100000
napr1	double	0.4	napr1	0	1
napr2	double	0.35	napr2	0	1
napr3	double	0.25	napr3	0	1
kol_kan_Napr1	int	1	kol_kan_Napr1	10000	100000
kol_kan_Napr2	int	1	kol_kan_Napr2	10000	100000
kol_kan_Napr3	int	1	kol_kan_Napr3	10000	100000
emkost_bufera_Napr1	int	200000	emkost_bufera_Napr1	0	500000
emkost_bufera_Napr2	int	200000	emkost_bufera_Napr2	0	500000
emkost_bufera_Napr3	int	200000	emkost_bufera_Napr3	0	500000
tek_emkost_Napr1	int	0			
tek_emkost_Napr2	int	0			
tek_emkost_Napr3	int	0			
skor_pered_Napr1	double	50000	skor_pered_Napr1	10000	100000
skor_pered_Napr2	double	50000	skor_pered_Napr2	10000	100000
skor_pered_Napr3	double	50000	skor_pered_Napr3	10000	100000

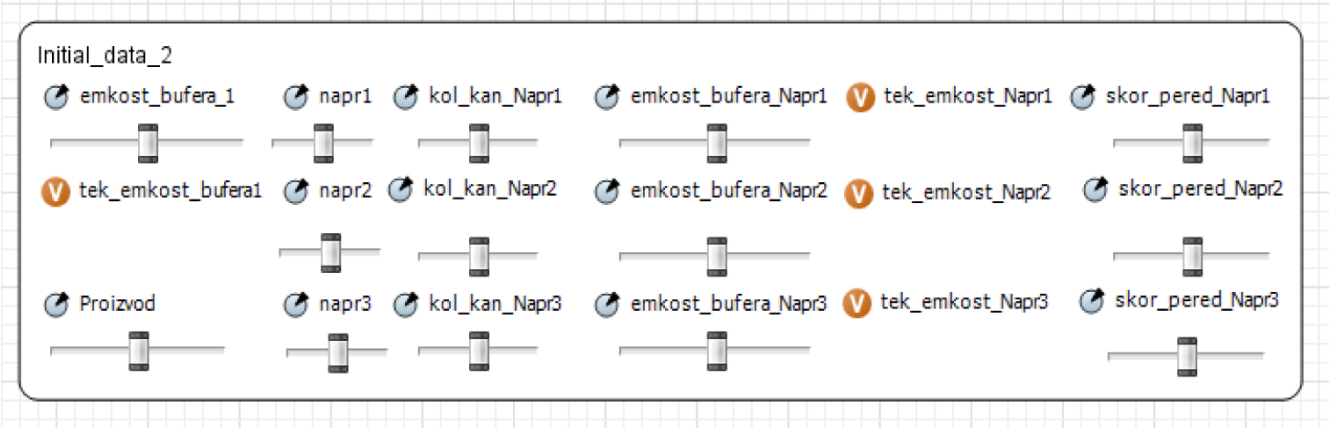


Рис. 3.49. Размещение элементов **Параметр**, **Простая переменная** и **Бегунок** на Initial_data_2

3.2.6. Организация вывода результатов моделирования

Для вывода результатов моделирования используем элемент **Простая переменная**. Для вывода среднего времени передачи сообщений — элемент **Данные гистограммы** (рис. 3.50).

1. В **Палитре** выделите **Презентация**. Перетащите элемент **Скругленный прямоугольник** в нужное место.
2. Перетащите элемент **text** и на странице **Основные** панели **Свойства** в поле **Текст:** введите `Results_of_modeling`.
3. В **Палитре** выделите **Основная**. Перетащите элементы **Простая переменная** на элемент с именем `Results_of_ modeling`, разместите их и дайте имена так, как показано на рис. 3.50. Тип всех переменных double. **Значение по умолчанию** равно 0.
4. Элемент `time_pered` установим позже.

3.2.7. Построение событийной части модели

Построение событийной части модели в AnyLogic будем осуществлять последовательной реализацией блоков, приведенных на рис. 3.47 (не путайте блоки маршрутизатора с блоками AnyLogic).

3.2.7.1. Блок Источники сообщений

Данный блок предназначен для имитации источников сообщений, поступления их через случайные интервалы времени, розыгрыша параметров сообщений и счета количества поступивших на маршрутизатор сообщений по категориям, направлениям передачи и всего, запоминания времени поступления каждого сообщения, используемого в последующем для расчета времени передачи.

Алгоритм блока **Источники сообщений** приведен на рис. 3.51.

В AnyLogic он реализуется путем программного описания основных параметров блока **source** — реплицированного экземпляра стандартного объекта (см. п. 2.4.1) библиотеки Enterprise Library.

1. Разместите элементы для ввода исходных данных и вывода результатов моделирования так, чтобы будущие объекты модели вы могли размещать выше их. Для перемещения, например группы `Initial_data_1`, щелкните мышью левее и выше левого верхнего угла прямоугольника группы и, не отпуская кнопку и двигаясь вправо, выделите все элементы группы. Теперь отпустите кнопку, наведите курсор на выделенную группу элементов и снова щелкните мышью и, не отпуская ее, перетащите группу элементов в нужное место.

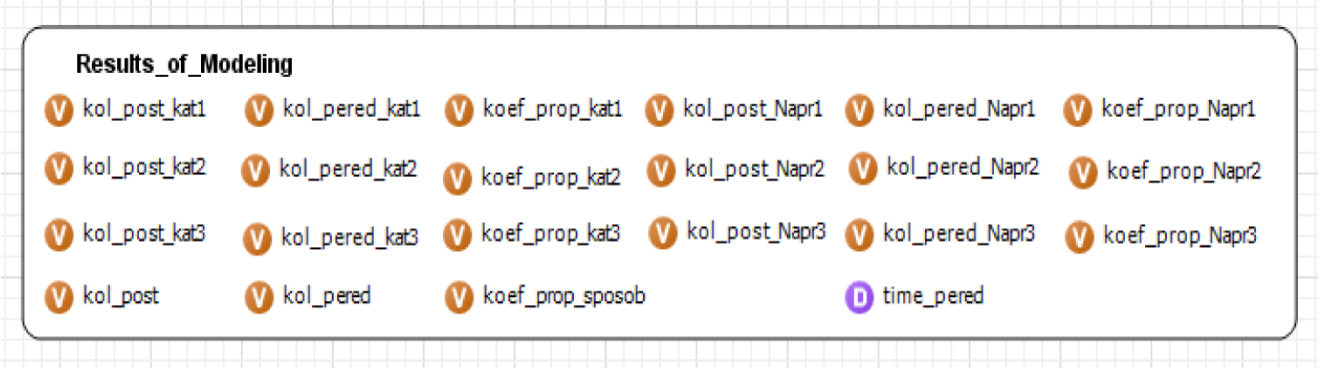


Рис. 3.50. Размещение элементов **Простая переменная** на Results_of_modeling

Рис 3.51. Алгоритм блока **Источники сообщений**

2. В **Палитре** выделите Enterprise Library.
3. Перетащите объект **source** на диаграмму.
4. Для ввода в сообщения дополнительных полей для записи и хранения параметров необходимо создать нестандартный класс заявки. Создайте класс заявки `Message`.
5. В панели **Проект** щелкните правой кнопкой мыши элемент модели верхнего уровня дерева и выберите **Создать/Java класс**.
6. Появится диалоговое окно **Новый Java класс** (см. рис. 3.32). В поле **Имя:** введите имя нового класса: `Message`.
7. В поле **Базовый класс:** выберите из выпадающего списка `com.xj.anylogic.libraries.enterprise.Entity` в качестве базового класса. Щелкните кнопку **Далее**.

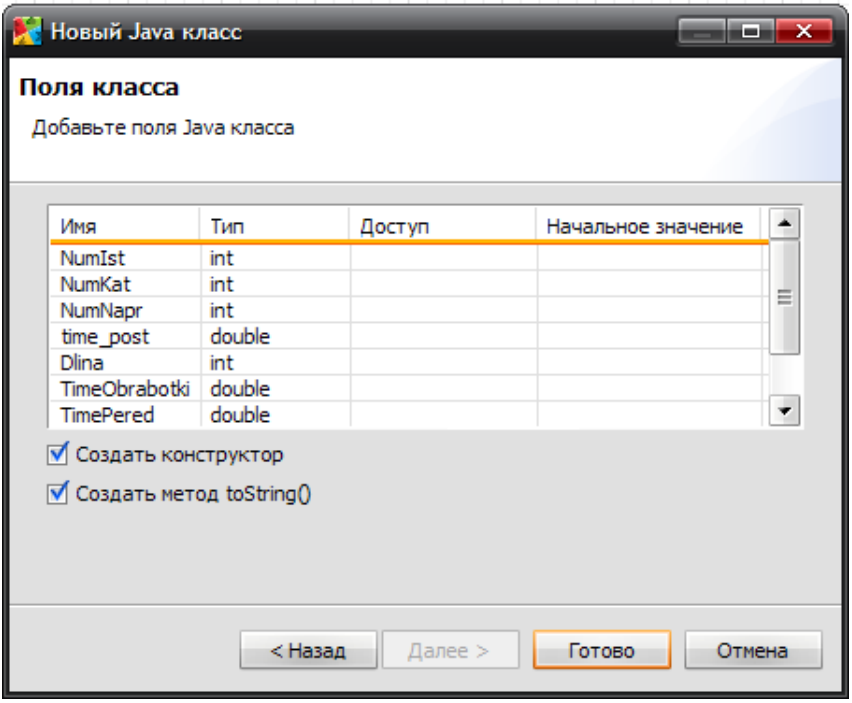


Рис. 3.52. Дополнительные поля класса заявок `Message`

8. Появится вторая страница **Мастера создания Java класса**. Добавьте поля Java класса, показанные на рис. 3.52.

9. Оставьте выбранными флажки **Создать конструктор** и **Создать метод toString ()**.

10. Щелкните кнопку **Готово**. Вы увидите редактор кода, в котором будет показан автоматически созданный код Вашего Java класса.

11. Дополните его строками Java кода `Vixod`, выделенными в приведенном ниже листинге жирным шрифтом. Из комментария следует, что этим кодом разыгрываются номера источников сообщений, их категорий и направлений передачи.

```
/**
 * Message
 */
public class Message extends
com.xj.anylogic.libraries.enterprise.Entity imple-
ments java.io.Serializable {
```

```
   int NumIst;
   int NumNapr;
   int NumKat;
   double time_post;
   int Priority;
   int Dlina;
   double TimeObrabotki;
   double TimePered;
     /**
      * Конструктор по умолчанию
      */
     public Message(){
     }
void Vihod(double kat1, double kat2, double kat3,
double napr1, double napr2, double napr3, double
KolIst){

          double a=0;
          double b=KolIst;

   /**
   * Розыгрыш номера источника сообщения
   */
          a=uniform(1);
          for (int i=1; i<(KolIst+1); i++)
          {if (a<=((1/KolIst)*b))
             NumIst=i;
             b=b-1;}

    /**
    * Розыгрыш номера направления передачи сообщения

    */
          a=uniform(1);
          if (a<=(napr1+napr2+napr3)) NumNapr=3;
       if (a<=(napr1+napr2)) NumNapr=2;
       if (a<=napr1) NumNapr=1;

     /**
     * Розыгрыш номера категории и приоритета сооб-
щения
     */
          a=uniform(1);
          if (a<=(kat1+kat2+kat3)) NumKat=3;
```

```
        if (a<=(kat1+kat2)) NumKat=2;
          if (a<=kat1)  NumKat=1;

     /**
     * Назначение приоритета сообщению первой категории
     */
        if (NumKat==1) Priority=1;
       }
    /**
     * Конструктор, инициализирующий поля
     */
    public Message(int NumIst, int NumNapr, int NumKat, double time_post, int Priority, int Dlina, double TimeObrabotki, double TimePered){
     this.NumIst = NumIst;
     this.NumNapr = NumNapr;
     this.NumKat = NumKat;
     this.time_post = time_post;
     this.Priority = Priority;
     this.Dlina = Dlina;
     this.TimeObrabotki = TimeObrabotki;
     this.TimePered = TimePered;
    }
  @Override
  public String toString() {
     return
     "NumIst = " + NumIst +" " +
     "NumNapr = " + NumNapr +" " +
     "NumKat = " + NumKat +" " +
     "time_post = " + time_post +" " +
     "Priority = " + Priority +" " +       "Dlina = " + Dlina +" " +
     "TimeObrabotki = " + TimeObrabotki +" " +
     "TimePered = " + TimePered +" ";
  }
  /**
   * Это число используется при сохранении состояния модели<br>
   * Его рекомендуется изменить в случае изменения класса
   */
private static final long serialVersionUID = 1L;
}
```

12. Закройте редактор кода, щелкнув крестик в закладке рядом с его названием.

13. Перетащите элемент **text** и на странице **Основные** панели **Свойства** в поле **Текст:** введите `Источник сообщений`.

14. Выделите блок **source**. На странице **Основные** панели **Свойства** (рис. 3.53) уберите флажок **Отображать имя**. В полях **Класс заявки:** и **Новая заявка** `Entity` замените `Message`. Установите, что **Заявки прибывают согласно Времени между прибытиями**. Введите `exponential(1/(Time_mean/KolIst))` в поле **Время между прибытиями**.

15. В поле **Действие при выходе** введите приведенный ниже Java код. Этим кодом разыгрываются длины сообщений, осуществляется подсчет поступивших сообщений, запоминается время поступления, определяется время обработки и передачи сообщения.

```
int a = 0;
double b = 0;
entity.Vihod(kat1, kat2, kat3, napr1, napr2, napr3,
KolIst);
if (entity.NumKat == 1)
   {b = normal(dlKat01, dlKat1);
    kol_post_kat1++;}
        kol_post++;
if (entity.NumKat == 2)
     {b=normal(dlKat02, dlKat2);
        kol_post_kat2++;}
if (entity.NumKat == 3)
     {b = normal(dlKat03, dlKat3);
        kol_post_kat3++;}
        a = (int)b;
        b = (double)a;
        entity.TimeObrabotki = b/Proizvod;
        entity.time_post = time();
if (entity.NumNapr == 1)
   {entity.TimePered =(b/skor_pered_Napr1)*8;
    kol_post_Napr1++;}
if (entity.NumNapr == 2)
     {entity.TimePered =(b/skor_pered_Napr2)*8;
        kol_post_Napr2++;}
if (entity.NumNapr == 3)
     {entity.TimePered =(b/skor_pered_Napr3)*8;
        kol_post_Napr3++;}
          entity.Dlina = a;
```

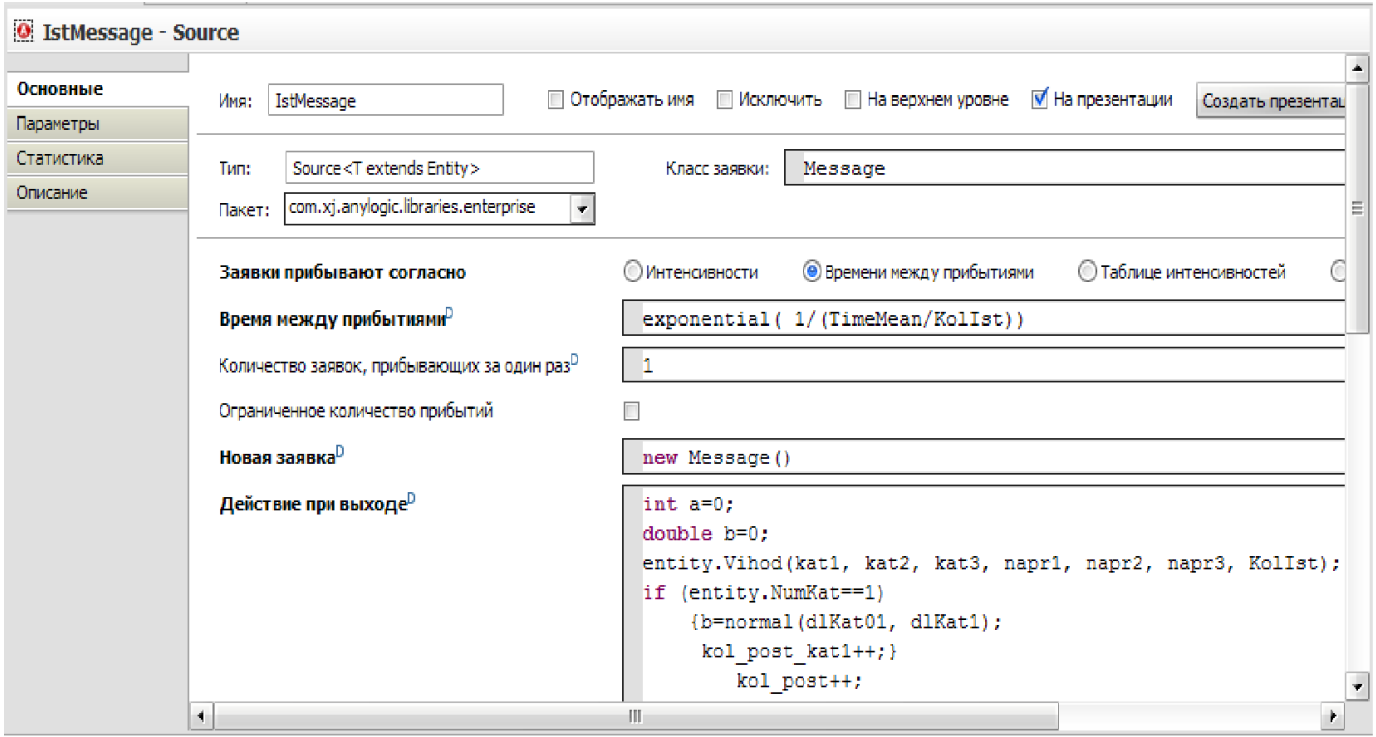


Рис. 3.53. Блок sourse с установленными параметрами

Для вывода результатов моделирования нами использован элемент **Простая переменная**, но можно было бы взять элемент **Параметр**, который обычно используется для задания статических характеристик объекта. Значение параметра в ходе моделирования можно изменять, если сброшен флажок **Динамический**. Тогда значения параметра будут изменяться при вызове функции `set_<имя параметра>()` (при простом присваивании, например, как `parameter=5`, код вызываться не будет). Значение параметра будет изменяться, например, при вызове функции `set_kol_post(kol_post+1);` и других функций, имя которых начинается с `set_`.

3.2.7.2. Блок контроля 1

Блок предназначен для контроля текущей емкости буфера 1 маршрутизатора. Он анализирует наличие в буфере 1 свободной памяти, достаточной для хранения поступившего сообщения, и в зависимости от результата анализа сообщение либо помещается в буфер 1, либо уничтожается.

Алгоритм работы **Блока контроля 1** представлен на рис. 3.54.

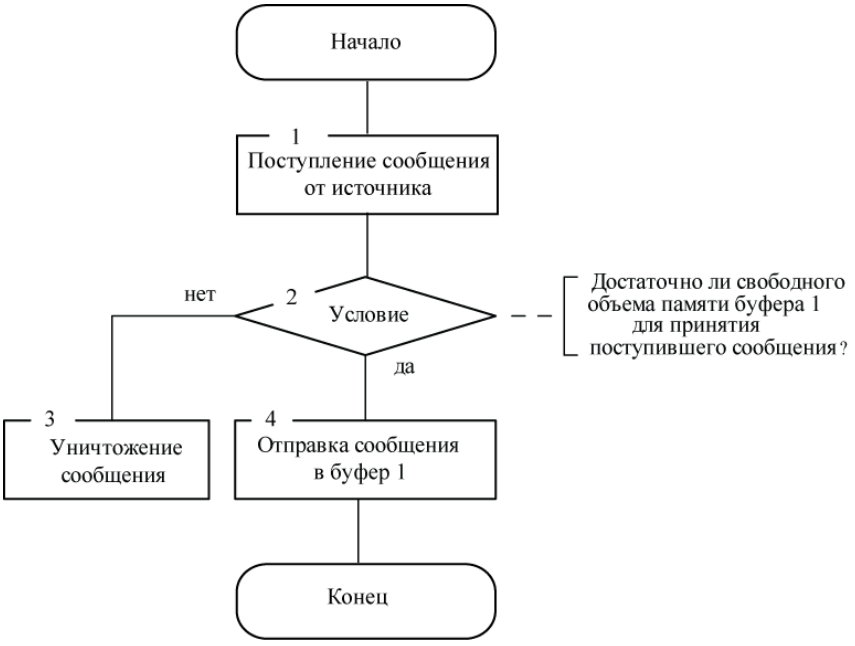


Рис. 3.54. Алгоритм работы **Блока контроля 1**

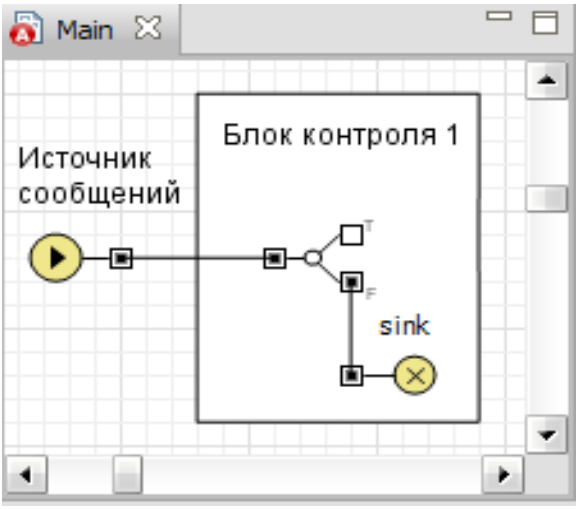


Рис. 3.55. Добавлены блоки **selectOutput** и **sink**

В AnyLogic этот алгоритм реализуется блоками **selectOutput** и **sink** — экземплярами объектов библиотеки Enterprise Library.

1. Перетащите блоки **selectOutput** и **sink** на диаграмму класса Main, разместите и соедините их так, как показано на рис. 3.55.
2. В **Палитре** выделите **Презентация**. Перетащите элемент **Прямоугольник** на блоки **selectOutput** и **sink** и разместите так, как показано на рис. 3.55.
3. Перетащите элемент **text** и на странице **Основные** панели **Свойства** в поле **Текст:** введите `Блок контроля 1`.
4. Выделите объект **selectOutput**. Объект **selectOutput** направляет входящие заявки в один из двух выходных портов `T` и `F` в зависимости от выполнения заданного условия. Условие может зависеть как от заявки, так и от каких-то внешних факторов. Поступившая заявка покидает объект в тот же момент времени. Может использоваться для сортировки заявок согласно заданному критерию, для случайного разделения потока заявок (заданием вероятностей) на части и т. д.
5. В поле **Имя:** вместо **selectOutput** введите `BlokKontrol_1`.
6. В поле **Класс заявки:** `Entity` замените `Message`.
7. Установите **Выход true выбирается При выполнении условия**.
8. В поле **Условие** введите условие:

```
emkost_bufera_1-tek_emkost_bufera1>=entity.Dlina
```

При выполнении условия заявка направляется на выход `T` (выходной порт для заявок, для которых выбирается выход true) и на выход `F` (выходной порт для заявок, для которых выбирается выход `false`), если условие не выполняется, соответственно.

Объекты класса **Sink** уничтожают входящие заявки. Для успешного уничтожения заявки нужно выполнение трех условий:

1. Если заявка находится в сети, то она должна быть удалена из этой сети с помощью объекта `NetworkExit`.
2. Заявка не должна обладать ресурсами.
3. Если заявка содержит другие заявки, то они тоже должны удовлетворять вышеуказанным условиям.

Если какое-то условие не выполняется, **sink** выдает ошибку.

3.2.7.3. Блок Буфер 1

Алгоритм работы блока **Буфер 1** приведен на рис. 3.56.

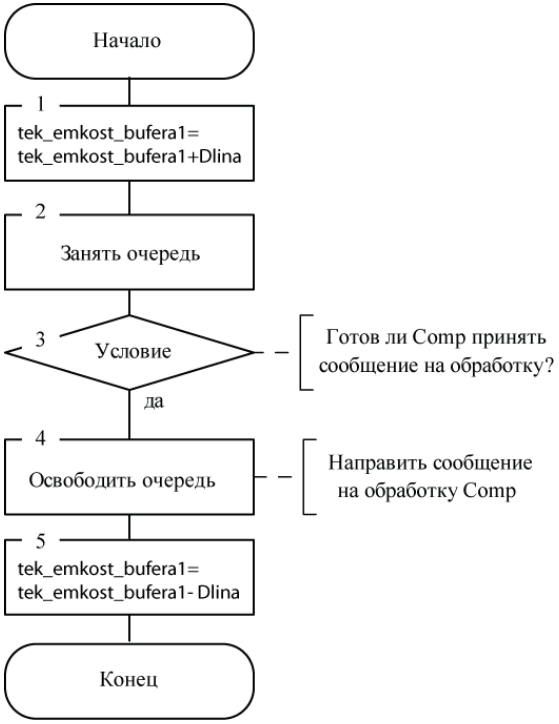


Рис. 3.56. Алгоритм работы блока **Буфер 1**

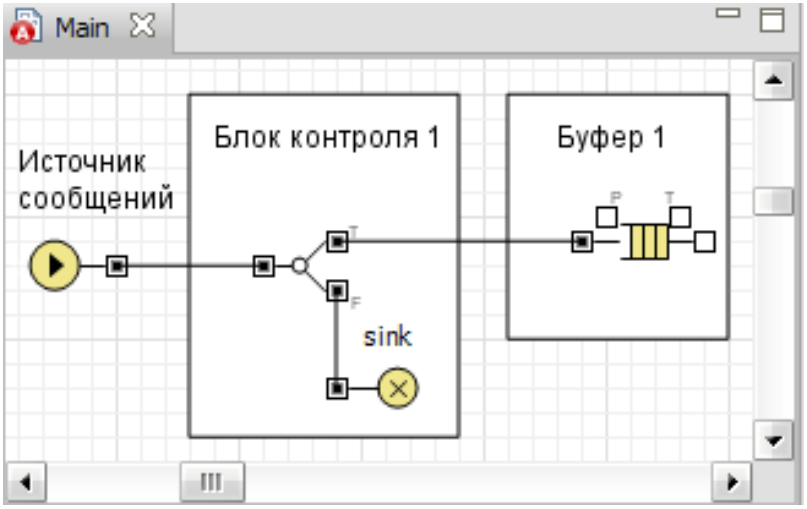


Рис. 3.57. Добавлен блок **queue**

Блок **Буфер 1** предназначен для приема, размещения и хранения согласно приоритету поступающих на обработку сообщений.

В AnyLogic алгоритм блока **Буфер 1** реализуется блоком **queue**, который выполняет функции очереди (FIFO).

1. Перетащите блок **queue** из библиотеки Enterprise Library, разместите и соедините с **Блок контроля 1** так, как на рис. 3.57.
2. В **Палитре** выделите **Презентация**. Перетащите элемент **Прямоугольник** на блок **queue** и разместите так, как на рис. 3.57.
3. На странице **Основные** панели **Свойства** в поле **Имя:** оставьте `Rectangle`. Не устанавливайте флажок **Отображать имя**.
4. Выделите объект **queue**.
5. В поле **Имя:** вместо **queue** введите `bufer_1`.
6. В поле **Класс заявки:** `Entity` замените `Message`.
7. Установите флажок **Максимальная вместимость**.
8. При помещении сообщения в буфер его текущая емкость увеличивается, поэтому в поле **Действие при входе** введите:

```
tek_emkost_bufera1 += entity.Dlina;
```

9. При выходе сообщения из буфера его текущая емкость уменьшается, поэтому в поле **Действие при выходе** введите:

```
tek_emkost_bufera1 -= entity.Dlina;
```

10. Поставьте флажок **Включить сбор статистики**.

Рис. 3.58. Алгоритм работы **Блока обработки сообщений**

3.2.7.4. Блок обработки сообщений

Блок предназначен для имитации обработки сообщения.

Алгоритм работы блока приведен на рис. 3.58.

Для реализации алгоритма **Блока обработки сообщений** в AnyLogic используется блок **delay** — экземпляр класса Delay.

1. Перетащите блок **delay** из библиотеки Enterprise Library, разместите и соедините с `bufer_1` так, как на рис. 3.59.
2. В **Палитре** выделите **Презентация**. Перетащите элемент **Прямоугольник** на блок **delay** и разместите так, как на рис. 3.59. На странице **Основные** панели **Свойства** в поле **Имя:** оставьте `Rectangle`. Не устанавливайте флажок **Отображать имя**.
3. Выделите объект **delay**.
4. На странице **Основные** панели **Свойства** в поле **Имя:** вместо **delay** введите `Computer`.
5. В поле **Класс заявки:** `Entity` замените `Message`.
6. **Задержка задается** установите **Явно**.

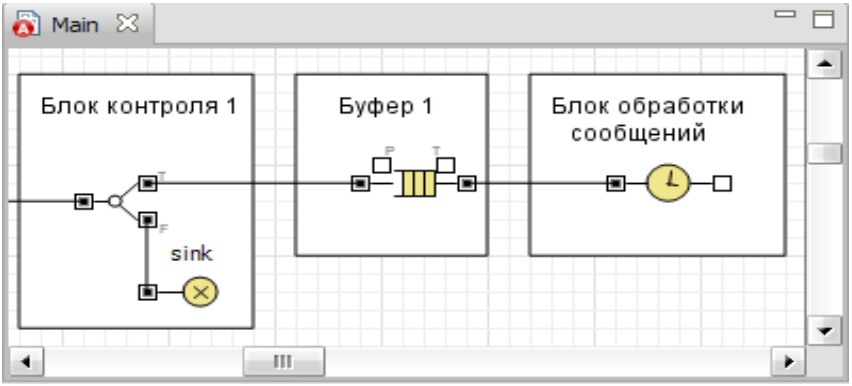


Рис. 3.59. Добавлен блок **delay**

7. В поле **Время задержки** введите: `entity.TimeObrabotki`
8. Оставьте **Вместимость** `1`.
9. Установите флажок **Включить сбор статистики**.

3.2.7.5. Блок контроля 2

Блок контроля 2 предназначен для распределения сообщений по направлениям и контроля текущих емкостей буферов (накопителей) направлений передачи сообщений.

Алгоритм работы **Блока контроля 2** приведен на рис. 3.60.

Рис. 3.60. Алгоритм блока **Блок контроля 2**

Вначале определяется номер направления, по которому должно быть передано поступившее сообщение. Затем определяется наличие достаточной свободной памяти в буфере этого направления. При отсутствии нужного объема памяти сообщение теряется.

В AnyLogic **Блок контроля 2** реализуется четырьмя объектами **selectOutput** и блоком **sink**.

1. Перетащите четыре блока **selectOutput** и один блок **sink** из библиотеки Enterprise Library на диаграмму класса Main, разместите и соедините их так, как на рис. 3.61.
2. Выделите блок **selectOutput**, вход которого соединен с выходом `Computer`.
3. На странице **Основные** панели **Свойства** в поле **Имя:** введите `kontr_Napr1`.
4. В поле **Класс заявки:** `Entity` замените `Message`.
5. **Выход true выбирается** установите **При выполнении условия**.
6. В поле **Условие** введите `entity.NumNapr==1`.

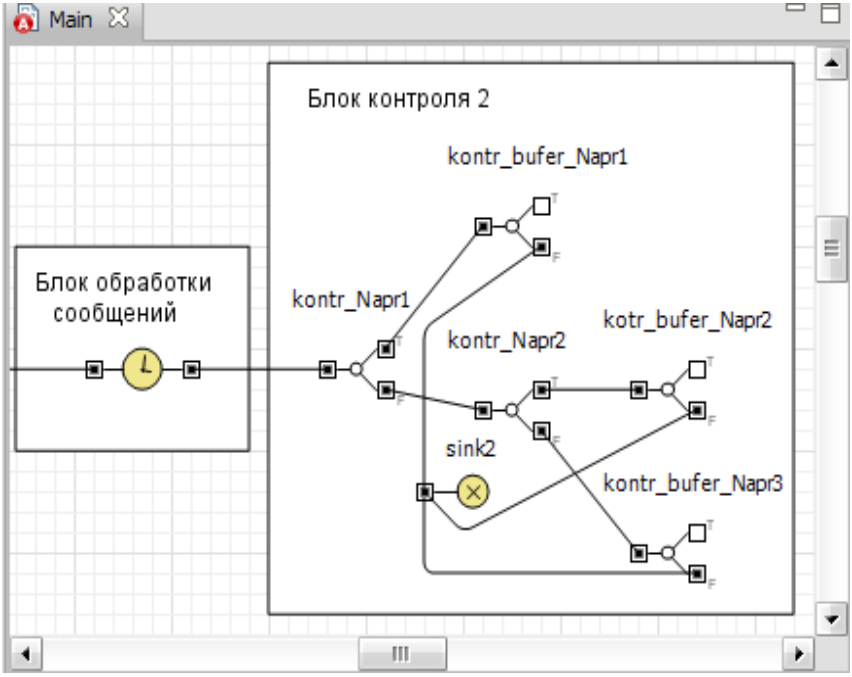


Рис. 3.61. Добавлены блоки **selectOutput** и **sink**

7. Выделите блок **selectOutput**, вход которого соединен с выходом **true** блока `kontr_Napr1`.

8. На странице **Основные** панели **Свойства** в поле **Имя:** введите `kontr_bufer_Napr1`.

9. В поле **Класс заявки:** `Entity` замените `Message`.

10. **Выход true выбирается** установите **При выполнении условия**. В поле **Условие** введите:

```
emkost_bufera_Napr1-tek_emkost_Napr1>=entity.Dlina
```

11. Выделите блок **selectOutput**, вход которого соединен с выходом **false** блока `kontr_Napr1`.

12. На странице **Основные** панели **Свойства** в поле **Имя:** введите `kontr_Napr2`.

13. В поле **Класс заявки:** `Entity` замените `Message`.

14. **Выход true выбирается** установите **При выполнении условия**. В поле **Условие** введите: `entity.NumNapr==2`.

15. Выделите блок **selectOutput**, вход которого соединен с выходом **true** блока `kontr_Napr2`.

16. На странице **Основные** панели **Свойства** в поле **Имя:** введите `kontr_bufer_Napr2`.

17. В поле **Класс заявки:** `Entity` замените `Message`.

18. **Выход true выбирается** установите **При выполнении условия**. В поле **Условие** введите:

```
emkost_bufera_Napr2-tek_emkost_Napr2>=entity.Dlina
```

19. Выделите блок **selectOutput**, вход которого соединен с выходом **false** блока `kontr_Napr2`.

20. На странице **Основные** панели **Свойства** в поле **Имя:** введите `kontr_bufer_Napr3`.

21. В поле **Класс заявки:** `Entity` замените `Message`.

22. **Выход true выбирается** установите **При выполнении условия**. В поле **Условие** введите:

```
emkost_bufera_Napr3-tek_emkost_Napr3>=entity.Dlina
```

23. В **Палитре** выделите **Презентация**. Перетащите элемент **Прямоугольник** на добавленные блоки и разместите так, как на рис. 3.61.

24. Перетащите элемент **text** и на странице **Основные** панели **Свойства** в поле **Текст:** введите `Блок контроля 2`.

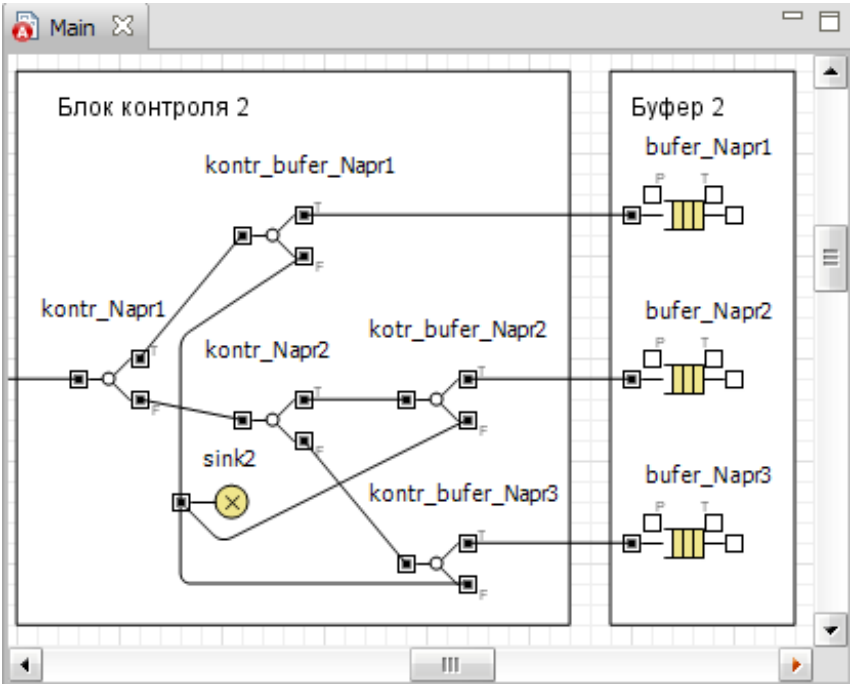


Рис. 3.62. Добавлены блоки **queue**

3.2.7.6. Блок Буфер 2

Блок **Буфер 2** предназначен для приема и хранения сообщений, передаваемых по каналам направлений. Он состоит из трех буферов — для каждого направления свой буфер.

Алгоритм работы буфера каждого из направлений такой же, как и алгоритм работы буфера 1 (см. рис. 3.56).

Реализуется каждый из буферов также блоком **queue.**

1. Перетащите три блока **queue** из библиотеки Enterprise Library на диаграмму класса Main, разместите и соедините их так, как на рис. 3.62.
2. Выделите блок **queue**, вход которого соединен с выходом `kontr_bufer_Napr1`.
3. На странице **Основные** панели **Свойства** в поле **Имя:** введите `bufer_Napr1`.
4. В поле **Класс заявки:** `Entity` замените `Message`.
5. Установите флажок **Максимальная вместимость**.

6. В поле **Действие при входе** введите:

```
tek_emkost_Napr1 += entity.Dlina;
```

7. В поле **Действие при выходе** введите:

```
tek_emkost_Napr1 -= entity.Dlina;
```

8. Установите флажок **Включить сбор статистики**.
9. Выделите блок **queue**, вход которого соединен с выходом `kontr_bufer_Napr2`.
10. На странице **Основные** панели **Свойства** в поле **Имя:** введите `bufer_Napr2`.
11. В поле **Класс заявки:** `Entity` замените `Message`.
12. Установите флажок **Максимальная вместимость**.
13. В поле **Действие при входе** введите:

```
tek_emkost_Napr2 += entity.Dlina;
```

14. В поле **Действие при выходе** введите:

```
tek_emkost_Napr2 -= entity.Dlina;
```

15. Установите флажок **Включить сбор статистики**.
16. Выделите блок **queue**, вход которого соединен с выходом `kontr_bufer_Napr3`.
17. На странице **Основные** панели **Свойства** в поле **Имя:** введите `bufer_Napr3`.
18. В поле **Класс заявки:** `Entity` замените `Message`.
19. Установите флажок **Максимальная вместимость**.
20. В поле **Действие при входе** введите:

```
tek_emkost_Napr3 += entity.Dlina;
```

21. В поле **Действие при выходе** введите:

```
tek_emkost_Napr3 -= entity.Dlina;
```

22. Установите флажок **Включить сбор статистики**.
23. В **Палитре** выделите **Презентация**. Перетащите элемент **Прямоугольник** на добавленные блоки и разместите так, как на рис. 3.62.
24. Перетащите элемент **text** и на странице **Основные** панели **Свойства** в поле **Текст:** введите `Буфер 2`.
25. Не устанавливайте флажок **Отображать имя**.

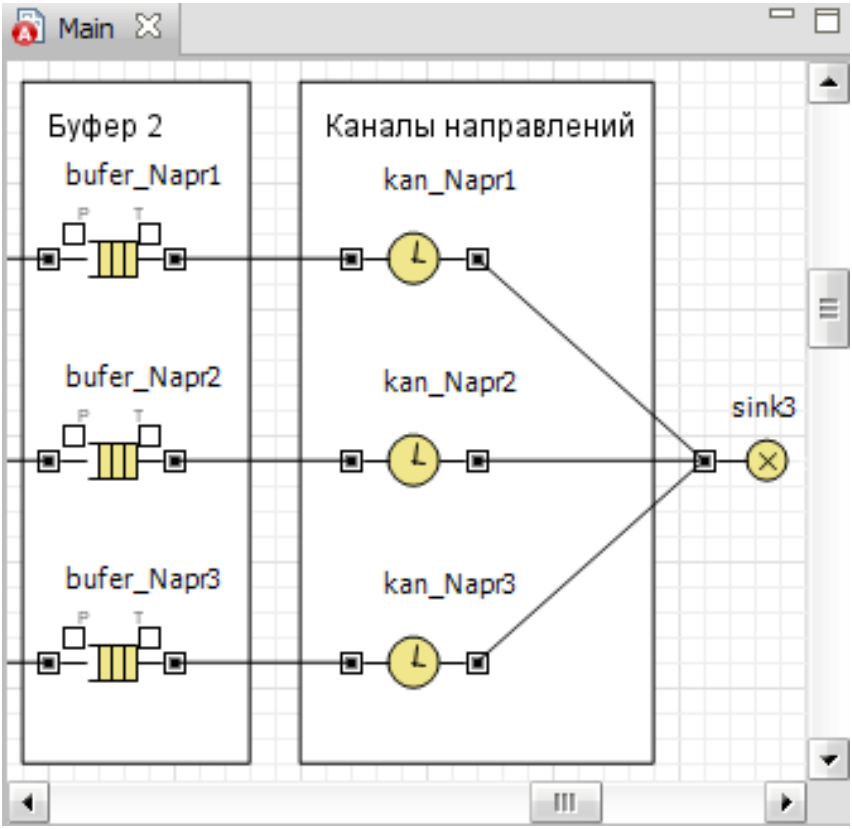


Рис. 3.63. Добавлены блоки **delay** и **sink**

3.2.7.7. Блок Каналы направлений

Данный блок предназначен для имитации передачи сообщений.

Для его реализации в AnyLogic используются три блока **delay**, каждый из которых и имитирует передачу сообщения по каналу связи с помощью встроенного метода delay(), выполняющего функцию задержки времени.

1. Перетащите три блока **delay** и один блок **sink** из библиотеки Enterprise Library на диаграмму класса Main, разместите и соедините их так, как на рис. 3.63. Блок **sink** необходим для завершения построения модели.

2. Выделите блок **delay**, вход которого соединен с выходом `bufer_Napr1`.

3. На странице **Основные** панели **Свойства** в поле **Имя:** введите `kan_Napr1`.

4. Установите флажок **Отображать имя**.

5. В поле **Класс заявки:** `Entity` замените `Message`.

6. **Задержка задается** установите **Явно**.

7. В поле **Время задержки** введите `entity.TimePered`.

8. В поле **Вместимость** введите `kol_kan_Napr1`.

9. Установите флажок **Включить сбор статистики**.

10. Выделите блок **delay**, вход которого соединен с выходом `bufer_Napr2`.

11. На странице **Основные** панели **Свойства** в поле **Имя:** введите `kan_Napr2`.

12. Установите флажок **Отображать имя**.

13. В поле **Класс заявки:** `Entity` замените `Message`.

14. **Задержка задается** установите **Явно**.

15. В поле **Время задержки** введите `entity.TimePered`.

16. В поле **Вместимость** введите `kol_kan_Napr2`.

17. Установите флажок **Включить сбор статистики**.

18. Выделите блок **delay**, вход которого соединен с выходом `bufer_Napr3`.

19. На странице **Основные** панели **Свойства** в поле **Имя:** введите `kan_Napr3`.

20. В поле **Класс заявки:** `Entity` замените `Message`.

21. **Задержка задается** установите **Явно**.

22. В поле **Время задержки** введите `entity.TimePered`.

23. В поле **Вместимость** введите `kol_kan_Napr3`.

24. Установите флажок **Включить сбор статистики**.

26. В **Палитре** выделите **Презентация**. Перетащите элемент **Прямоугольник** на добавленные блоки и разместите так, как показано на рис. 3.63.

27. Перетащите элемент **text** и на странице **Основные** панели **Свойства** в поле **Текст:** введите `Initial_data_1`.

28. На странице **Основные** панели **Свойства** в поле **Имя:** введите `Kan_Napr`.

25. Установите флажок **Отображать имя**.

26. Выделите блок **sink**.

27. На странице **Основные** панели **Свойства** в поле **Класс заявки:** введите `Message` (рис. 3.64).

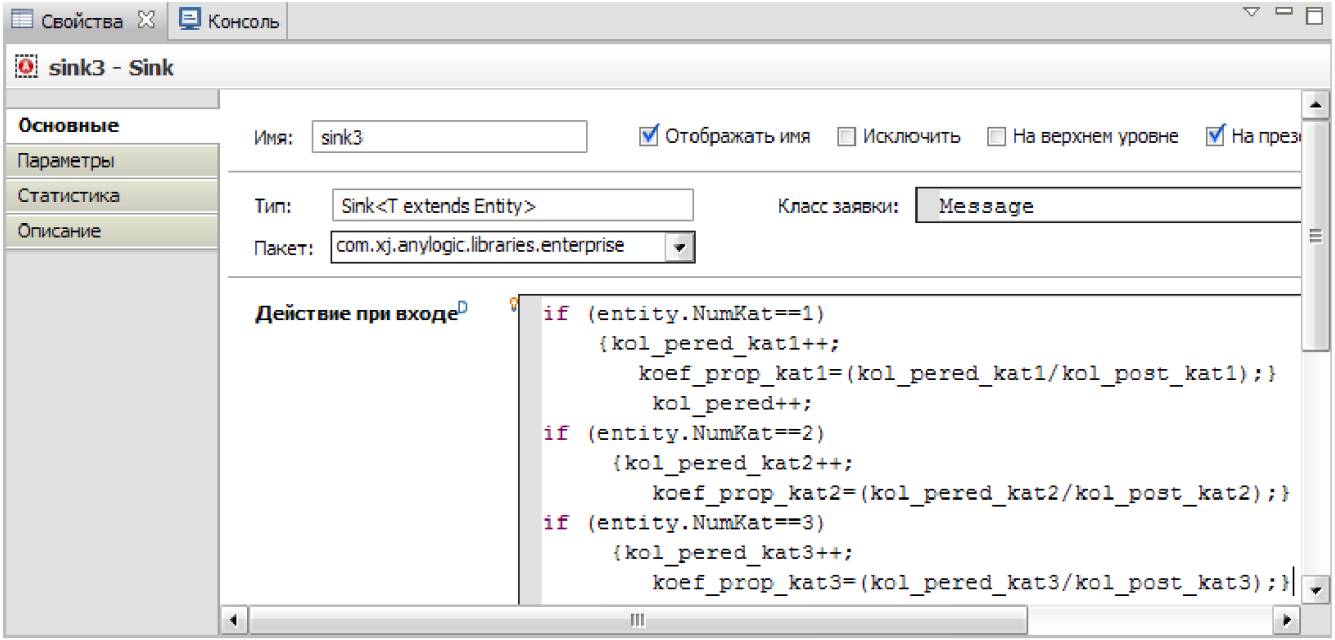


Рис. 3.64. Блок sink3 с установленными свойствами

28. В поле **Действие при входе** введите приведенный ниже Java код:

```
if (entity.NumKat==1)
   {kol_pered_kat1++;
  koef_prop_kat1=kol_pered_kat1/kol_post_kat1;}
      kol_pered++;
       koef_prop_sposob=kol_pered/kol_post;
if (entity.NumKat==2)
     {kol_pered_kat2++;
  koef_prop_kat2=kol_pered_kat2/kol_post_kat2;}
if (entity.NumKat==3)
     {kol_pered_kat3++;
koef_prop_kat3=kol_pered_kat3/kol_post_kat3;}
if (entity.NumNapr==1)
   {kol_pered_Napr1++;
koef_prop_Napr1=kol_pered_Napr1/kol_post_Napr1;}
if (entity.NumNapr==2)
     {kol_pered_Napr2++;
koef_prop_Napr2=kol_pered_Napr2/kol_post_Napr2;}
if (entity.NumNapr==3)
     {kol_pered_Napr3++;
koef_prop_Napr3=kol_pered_Napr3/kol_post_Napr3;}
   time_pered.add(time()-entity.time_post);
```

Построение варианта модели функционирования маршрутизатора завершено.

Запустите модель и проведите исследование. Фрагмент работы модели приведен на рис. 3.65, а фрагмент данных о направлениях связи `Initial_data_2` и результатах моделирования `Results_of_modelinq` — на рис. 3.66.

Дальнейшее совершенствование модели вы можете продолжить самостоятельно. Например, дополнить модель объектами и кодом, позволяющими делить сообщения на пакеты, учитывать приоритеты сообщений и т. д.

Кроме того, подумать об универсальности модели, то есть, чтобы она не требовала бы существенной модификации при увеличении количества направлений связи.

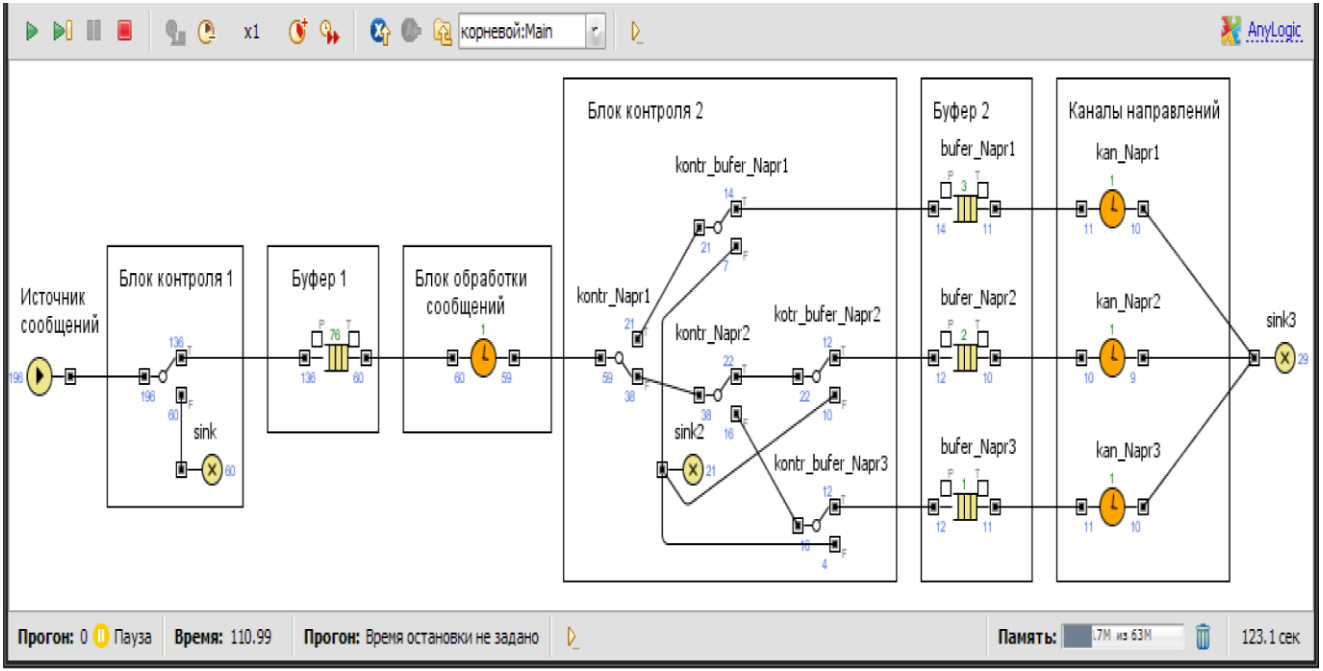


Рис. 3.65. Фрагмент работы модели Router

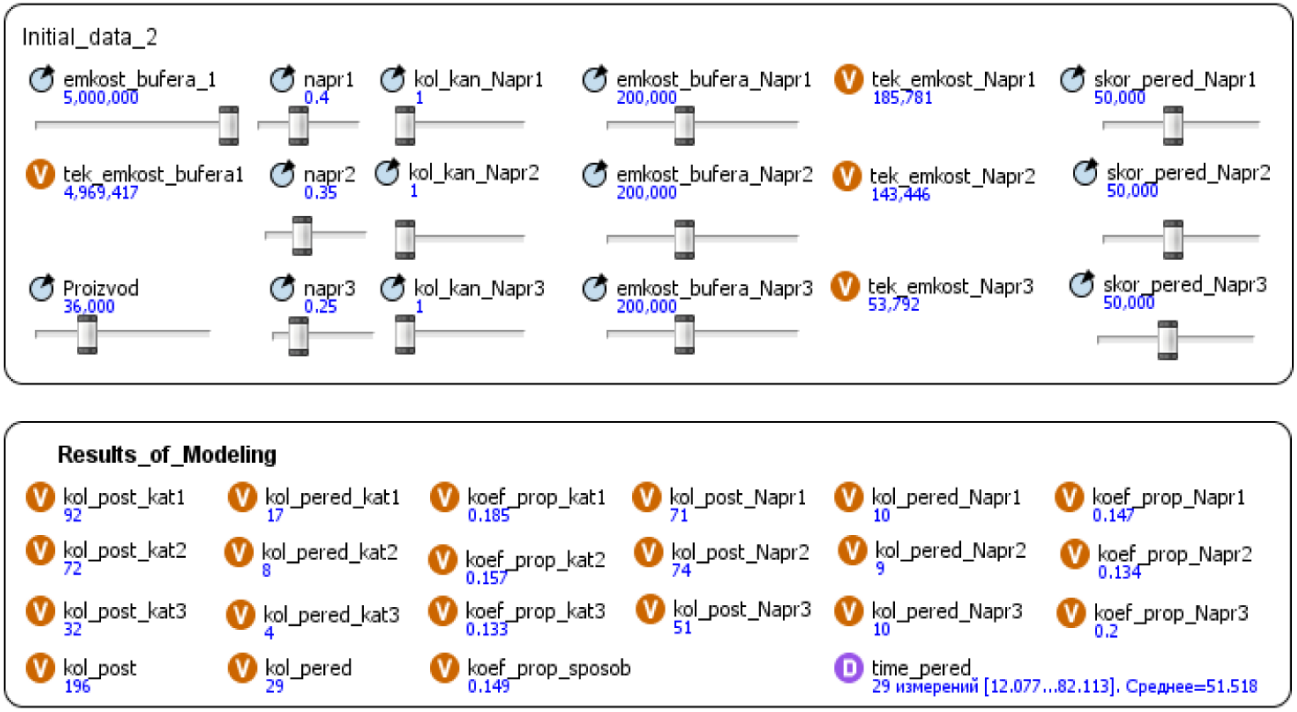


Рис. 3.66. Фрагмент данных Initial_data_2 и Results_of_Modeling

3.3. Модель функционирования системы связи

3.3.1. Постановка задачи

На дежурстве находятся n1 средств связи (СС) n2 типов (n21 + n22 + … + n2n2 = n2) в течение n3 часов.

Каждое СС может в любой момент времени выйти из строя. В этом случае его заменяют резервным, причем либо сразу, либо по мере его появления. Тем временем, вышедшее из строя СС ремонтируют, после чего содержат в качестве резервного. Всего количество резервных СС — n4.

Ремонт неисправных СС производят n5 мастеров. Время T1, T2, …, Tn2 ремонта случайное и зависит от типа СС, но не зависит от того, какой мастер это СС ремонтирует. Интервалы времени T21, T22, …, T2n2 между отказами СС, находящихся на дежурстве, случайные.

Прибыль от СС, находящихся на дежурстве, составляет S1 денежных единиц в час. Почасовой убыток при отсутствии на дежурстве одного СС — S2 денежных единиц. Оплата мастера за ремонт неисправного СС — S31, S32, …, S3n2 денежных единиц в час. Затраты на содержание одного резервного СС составляют S4 денежных единиц в час.

3.3.2. Задание на исследование

Разработать имитационную модель бизнес-процесса предоставления услуг по средствам связи в течение 25 суток (1000 часов).

Исследовать через промежутки времени ΔT = 5*40=200 (пять недель) влияние на ожидаемую прибыль различного количества резервных СС и мастеров.

Определить абсолютные величины и относительные коэффициенты ожидаемой прибыли для каждого промежутка времени.

Сделать выводы об использовании СС, мастеров по промежуткам времени ΔT и необходимых мерах по совершенствованию системы связи.

3.3.3. Формализованное описание модели

Уясним задачу на разработку имитационной модели, предварительно представив структуру системы предоставления услуг связи (рис. 3.67).

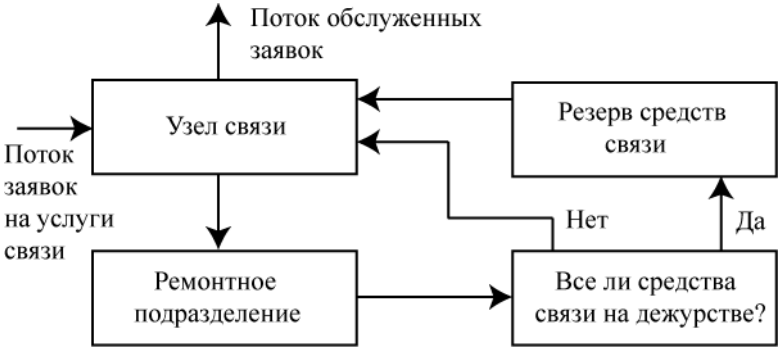


Рис. 3.67. Система связи как СМО

Система связи представляет собой многофазную многоканальную систему массового обслуживания замкнутого типа.

Таким образом, модель системы связи должна состоять из следующих сегментов (рис. 3.68):

имитации постановки на дежурство СС;

имитации дежурства СС;

имитации функционирования ремонтного подразделения;

вывода результатов моделирования.

Рис. 3.68. Концептуальная схема модели системы связи

Заявки как средства связи, поступившие на дежурство, должны иметь следующие параметры (поля):

tipCC – код типа СС;

timeMeanOtkaz – среднее время между отказами СС;

timeMeanRem – среднее время ремонта одного СС;

nach – время начала ремонта в ремонтном подразделении;

konec – время окончания ремонта в ремонтном подразделении.

Код типа СС в виде чисел 1, 2, 3, 4, 5 определяется в самом начале моделирования и остается неизменным. Для его определения используются следующие исходные данные:

KCC1 … KCC5 — количество СС первого … пятого типов соответственно;

KCCP1 … KCCP5 – количество резервных СС первого … пятого типов соответственно.

По этим же данным определяются количества всех СС по типам KolCC1 … KolCC5, а также общее количество СС всех типов KolCC.

В параметр timeMeanOtkaz заносится интенсивность выхода из строя соответствующего типа СС. Интенсивность рассчитывается по средним значениям интервалов выхода из строя СС первого … пятого типов timeOtkaz1 … timeOtkaz5.

В параметр timeMeanRem заносится интенсивность ремонта соответствующего типа СС. Интенсивность рассчитывается по средним значениям времени ремонта СС соответственно первого … пятого типов timeRem1 … timeRem5.

Рассчитанные интенсивности, например, timeMeanOtkaz = 1/timeOtkaz1 используются для обращения к генератору exponential(timeMeanOtkaz).

Параметры nach и konec изменяются при каждом поступлении СС в ремонтное подразделение. Они используются при расчете затрат на ремонт неисправного СС. В них заносится время начала и окончания ремонта соответственно.

Кроме рассмотренных, СС имеют еще следующие параметры (не заносимые в дополнительные поля заявок, имитирующих СС):

doxDegCC1 … doxDegCC5 — доход от дежурства одного СС первого … пятого типов соответственно;

zatrResCC1 … zatrResCC5 — затраты на содержание резерва одного СС первого … пятого типов соответственно;

stoimRem1 … stoimRem5 — стоимость ремонта одного СС первого … пятого типов соответственно.

В ходе моделирования через каждую единицу времени расчитываются:

PribCC1 … PribCC5, SumPribil — абсолютные величины ожидаемой прибыли по каждому типу СС и в целом;

KoefPribCC1 … KoefPribCC5, KoefPribil — относительные коэффициенты ожидаемой прибыли по каждому типу СС и в целом.

Рассмотрим вычисление этих показателей на примере PribCC1 и KoefPribCC1.

Предполагается, что максимальный доход DoxMaxCC1 от дежурства будет в случае, когда все СС первого типа будут постоянно находиться на дежурстве, то есть:

```
DoxMaxCC1=DoxMaxCC1+KCC1*doxDegCC1.
```

Фактический доход DoxDegCC1 от дежурства СС первого типа составит:

```
DoxDegCC1=DoxDegCC1+degCC1*doxDegCC1,
```

где degCC1 — количество СС первого типа, находившихся на дежурстве в данный момент времени.

При отсутствии на дежурстве СС первого типа убыток составит:

```
UbitokCC1=UbitokCC1+(KCC1-degCC1)*ubitCC1,
```

где ubitCC1 — убыток при отсутствии СС первого типа на дежурстве.

За единицу времени затраты на ремонт неисправных СС и на содержание резервных СС составят:

```
ZatrRemCC1=ZatrRemCC1+(konec-nach)*stoiRemCC1;
    ZatrRemCC1=ZatrResCC1+KCC1*zatrResCC1.
```

Абсолютная величина ожидаемой прибыли составит:

```
PribCC1=DoxDegCC1-(ZatrRemCC1+ZatrRemCC1+UbitokCC1).
```

Относительный коэффициент прибыли равен:

```
KoefPribCC1=PribCC1/DoxMaxCC1.
```

Показатели в целом за систему связи:

```
SumPribil=SumDoxDeg-
(SumZatrRes+SumZatrRem+SumUbitok),
KoefPrib=SumPrib/SumDoxMax,
```

где `SumDoxMax`, `SumDoxDeg`, `SumZatrRes`, `SumZatrRem`, `SumUbitok` — соответствующие доходы и затраты за систему.

3.3.4. Сегмент Постановка на дежурство

Сегмент предназначен для имитации поступления основных и резервных СС всех типов и постановки их на дежурство.

1. Выполните команду **Файл/Создать/Модель** на панели инструментов. Появится диалоговое окно **Новая модель** (см. рис. 3.1).
2. Задайте имя новой модели. В поле **Имя модели** введите `ComSystem`. Выберите каталог для сохранения файлов модели.
3. Щелкните кнопку **Далее**. Откроется вторая страница **Мастера создания модели** (см. рис. 3.2). Выберите **Начать создание модели «с нуля»**. Щелкните кнопку **Далее**.

3.3.4.1. Область просмотра

В случае сложных моделей, активные объекты которых содержат большое количество элементов, может возникнуть неудобство: все элементы активного объекта могут просто физически не поместиться в ту область диаграммы, которая будет отображена в окне презентации во время выполнения модели.

Версия 6 AnyLogic предоставляет в распоряжение пользователей специальный элемент для решения этой проблемы — *область просмотра*. С помощью этого элемента вы можете выделить на диаграмме активного объекта некоторые области, содержащие логически обособленные группы элементов или участки диаграммы. Задав такие области, вы сможете легко переключаться между ними во время выполнения модели с помощью специальных средств навигации, что позволит быстро переходить к тому или иному участку диаграммы активного объекта. При этом в окне презентации запущенной модели будут отображаться те элементы активного объекта, которые попали в заданную Вами ранее и сделанную в текущий момент активной область просмотра.

Используем три области просмотра. В первой области просмотра разместим элементы сегмента **Постановка на дежурство**, во второй — сегмента **Имитация дежурства** СС и сегмента имитации функционирования ремонтного подразделения, в третьей — сегмента **Статистика**.

Первую и третью области просмотра поместим на диаграмму класса Main, а для второй области просмотра создадим новый класс активного объекта **Degurstvo**.

Создайте область просмотра на диаграмме класса Main для размещения элементов сегмента **Постановка на дежурство**.

1. В **Палитре** выделите **Презентация**. Перетащите элемент **Область просмотра** в нужное место.
2. Вы увидите на диаграмме значок якоря этой области просмотра  . Чтобы в дальнейшем изменить свойства этой области, Вам нужно будет выделить этот значок мышью.
3. Перейдите на страницу **Основные** панели **Свойства**.
4. В поле **Имя:** введите `Postanovka`.
5. Задайте, как будет располагаться область просмотра относительно ее якоря, с помощью элемента управления **Выравнивать по:** `Верхн. левому углу`.
6. Выберите режим масштабирования из выпадающего списка **Масштабирование:** `Подогнать под окно`.

3.3.4.2. Ввод исходных данных

Исходные данные, другие необходимые параметры разделим и разместим в той области или в том сегменте модели, где они, по нашему мнению, нужны и их удобно использовать.

Организуйте ввод исходных данных для сегмента **Постановка на дежурство**.

1. Перетащите элемент **Прямоугольник** на элемент **Область просмотра**, если вы хотите видеть данные в ходе моделирования. Если нет, поместите этот элемент вне области просмотра.
2. Перетащите элемент **text** и на странице **Основные** панели **Свойства** в поле **Текст:** введите `Initial_data_PD` (здесь `PD` — постановка на дежурство).
3. В **Палитре** выделите **Основная**. Перетащите элементы **Параметр** и **Бегунок** на элемент с именем `Initial_data_PD` и разместите их так, как показано на рис. 3.69.
4. Значения свойств установите согласно табл. 3.3.

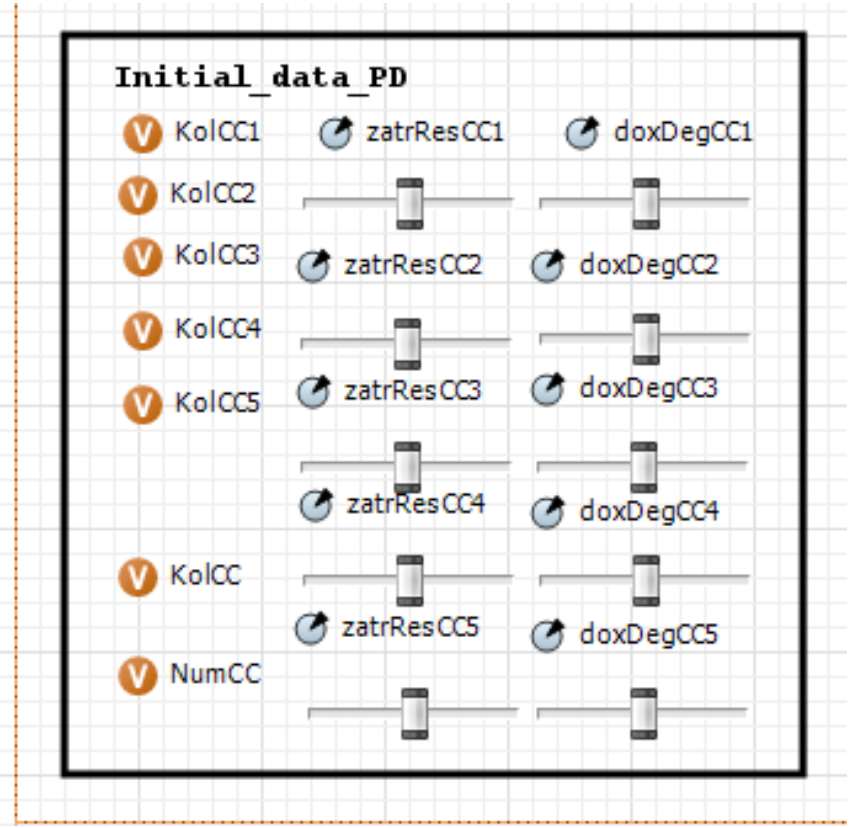


Рис. 3.69. Размещение элементов **Параметр** и **Бегунок**

Простые переменные с именами KolCC1…KolCC5 — количество СС по типам, а KolCC — количество СС всех типов. Эти переменные мы будем вычислять по исходным значениям КСС1 … КСС5 и КССР1 … КССР5 и использовать при генерации равного значению KolCC заявок, имитирующих СС. Количество СС изменять можно только перед началом моделирования, так как заявки, имитирующие СС, генерируются только один раз.

Простая переменная NumCC предназначена для порядковой нумерации всех СС.

Изменение затрат на содержание резервных СС zatrResCC1… zatrResCC5, а также дохода doxdegCC1… doxdegCC5 от дежурства одного СС можно производить в ходе моделирования. Для этого использованы элементы управления **Бегунок.**

Таблица 3.3

Свойства элементов на `Initial_data_PD`

Имя	Тип	Значение по умолчанию	Отображать имя
KolCC1	int	0	Установить флажок во всех элементах
KolCC2	int	0	
KolCC3	int	0	
KolCC4	int	0	
KolCC5	int	0	
KolCC	int	0	
NumCC	int	0	
doxdegCC1	double	20	
doxdegCC2	double	24.2	
doxdegCC3	double	32.8	
doxdegCC4	double	23	
doxdegCC5	double	25.5	
zatrResCC1	double	21	
zatrResCC2	double	24.2	
zatrResCC3	double	28	
zatrResCC4	double	26	
zatrResCC5	double	25.5	

3.3.4.3. Вывод результатов моделирования в сегменте Postanovka

Для вывода результатов моделирования используйте также элемент **Простая переменная**. Хотя можно было бы использовать и другие средства, например, функции, массивы. Для ввода исходных данных также можно было использовать массивы, что способствовало бы некоторой универсальности модели. Освоив предлагаемые в данном пособии приемы построения имитационных моделей в AnyLogic, вам нетрудно будет разобраться и применять функции, массивы и др.

1. Перетащите элементы **Простая переменная** и разместите так, как на рис. 3.70.
2. Дайте имена согласно рис. 3.70. Установите во всех элементах флажок **Отображать имя** и тип **double**.
3. Для расчета и вывода результатов моделирования воспользуемся способом *Событие.* Как это делается, рассмотрим в п. 3.3.7, когда будут построены все сегменты модели и станут известными нужные для расчета результатов моделирования идентификаторы с учетом требований языка Java и AnyLogic.

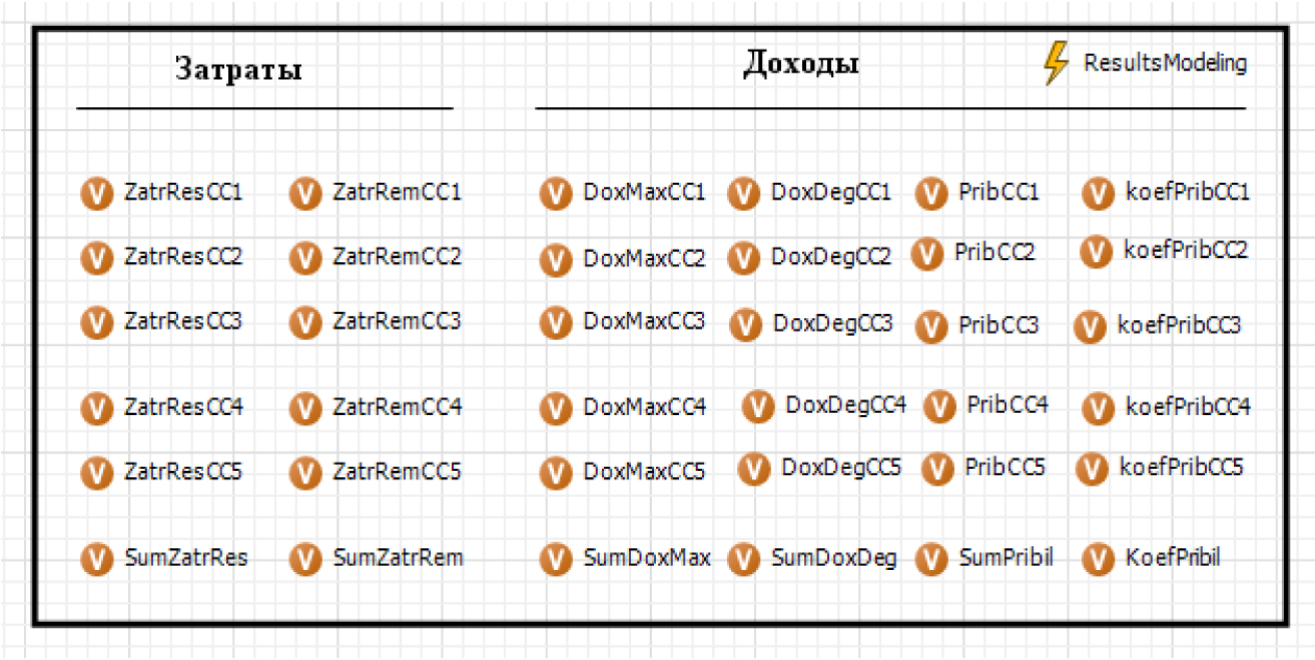


Рис. 3.70. Результаты моделирования в сегменте **Postanovka**

3.3.4.4. Имитация поступления средств связи

1. Перетащите элемент **Скругленный прямоугольник** на элемент **Область просмотра**. На нем мы будем размещать элементы сегмента **Постановка на дежурство**.

2. Перетащите элемент **text** и на странице **Основные** панели **Свойства** в поле **Текст:** введите `Постановка на дежурство`. Поместите этот текст посредине в верхней части элемента **Скругленный прямоугольник**.

3. Перетащите элемент **source** из библиотеки Enterprise Library на диаграмму класса Main.

4. Для записи и хранения параметров СС в дополнительные поля заявок необходимо создать нестандартный класс заявки. Создайте класс заявки `ComFacility`.

5. В панели **Проект** щелкните правой кнопкой мыши элемент модели верхнего уровня дерева и выберите **Создать/Java класс**.

6. Появится диалоговое окно **Новый Java класс** (см. рис. 3.32). В поле **Имя:** введите имя нового класса `ComFacility`.

7. В поле **Базовый класс:** выберите из выпадающего списка `com.xj.anylogic.libraries.enterprise.Entity` в качестве базового класса. Щелкните кнопку **Далее**.

8. Появится вторая страница **Мастера создания Java класса**. Добавьте следующие поля Java класса:

```
int tipCC;
double timeMeanRem;
double nach;
double konec;
double timeMeanOtkaz;
```

9. Оставьте выбранными флажки **Создать конструктор** и **Создать метод toString ()**.

10. Щелкните кнопку **Готово**. Вы увидите редактор кода и в автоматически созданный код вашего Java класса. Закройте код.

11. Перетащите элемент **text** и на странице **Основные** панели **Свойства** в поле **Текст:** введите `Имитация поступления СС`.

12. Выделите блок **source**. На странице **Основные** панели **Свойства** уберите флажок **Отображать имя**. В полях **Класс заявки:** и **Новая заявка** `Entity` замените `Message`. Установите: **Заявки прибывают согласно Интенсивности**. Остальные свойства — как на рис. 3.71. **Source** сгенерирует только одну заявку.

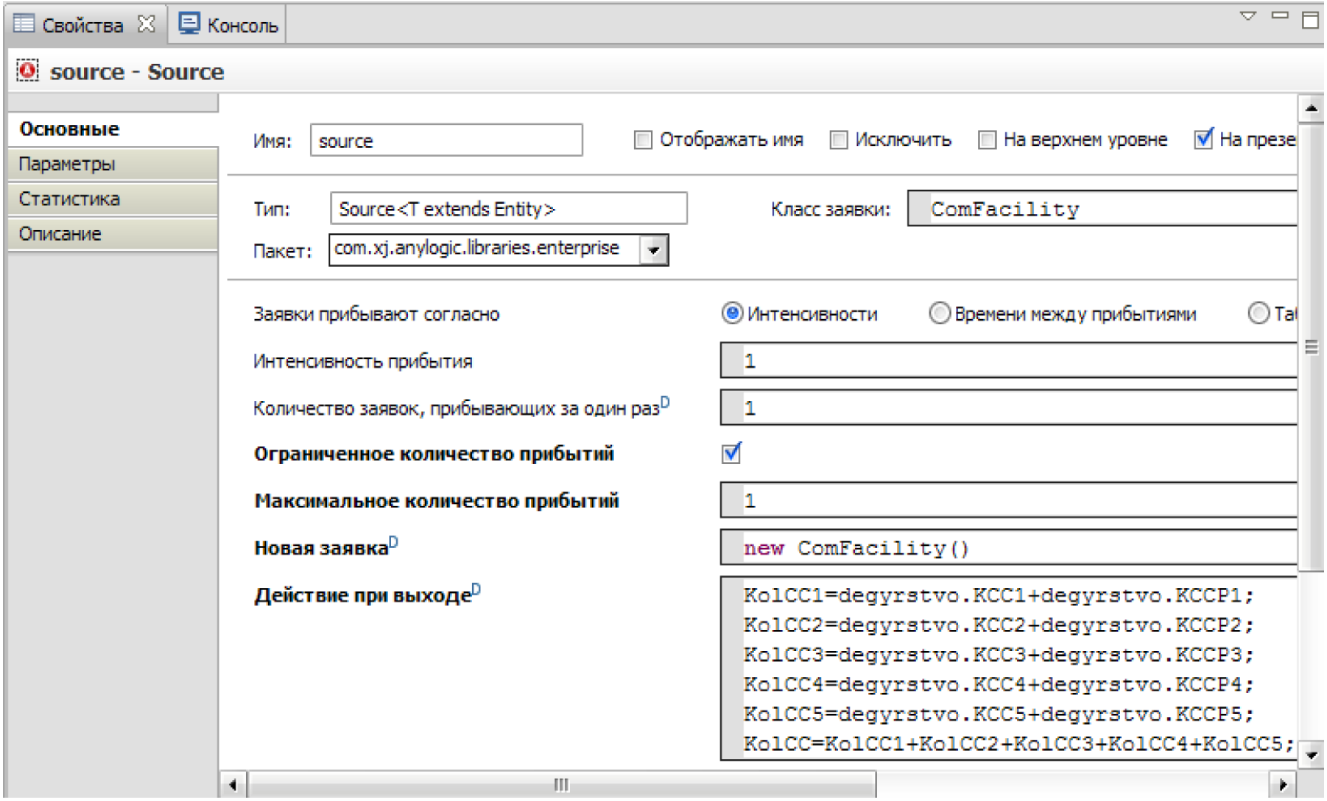


Рис. 3.71. Блок **source** с установленными свойствами

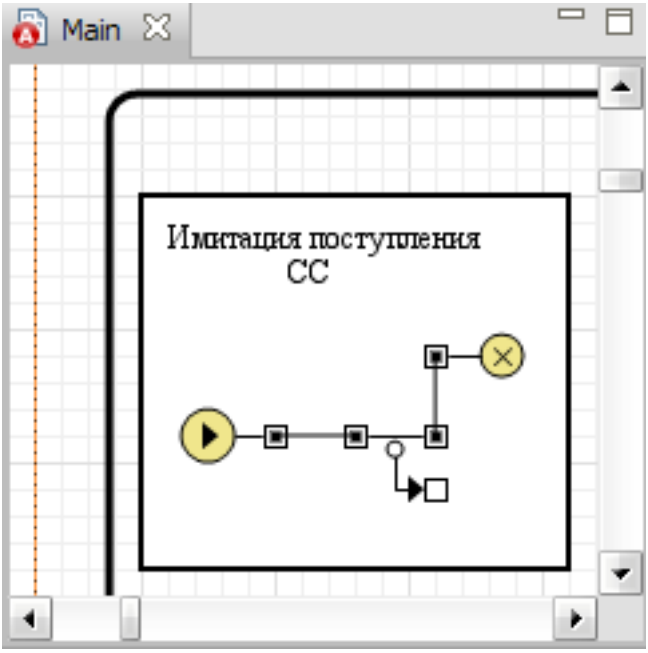


Рис. 3.72. Добавлены блоки **split** и **sink**

13. В поле **Действие при выходе** введите Java код, которым определяется количество СС всех типов, включая и резервные средства связи. Эти данные необходимы в блоке **split**. Количества СС как исходные данные будут размещены на активном классе Degyrstvo, который мы создадим позже, поэтому в коде используется доступ к ним в виде, например, degyrstvo.KCC1.

```
KolCC1=degyrstvo.KCC1+degyrstvo.KCCP1;
KolCC2=degyrstvo.KCC2+degyrstvo.KCCP2;
KolCC3=degyrstvo.KCC3+degyrstvo.KCCP3;
KolCC4=degyrstvo.KCC4+degyrstvo.KCCP4;
KolCC5=degyrstvo.KCC5+degyrstvo.KCCP5;
KolCC=KolCC1+KolCC2+KolCC3+KolCC4+KolCC5;
```

14. Выделите блок **split**. Установите свойства как на рис. 3.73. В поле **Действие при выходе копии** введите Java код:

```
NumCC = NumCC + 1;
if (NumCC <= KolCC)
  entity.tipCC = 5;
if (NumCC <= (KolCC1+KolCC2+KolCC3+KolCC4))
```

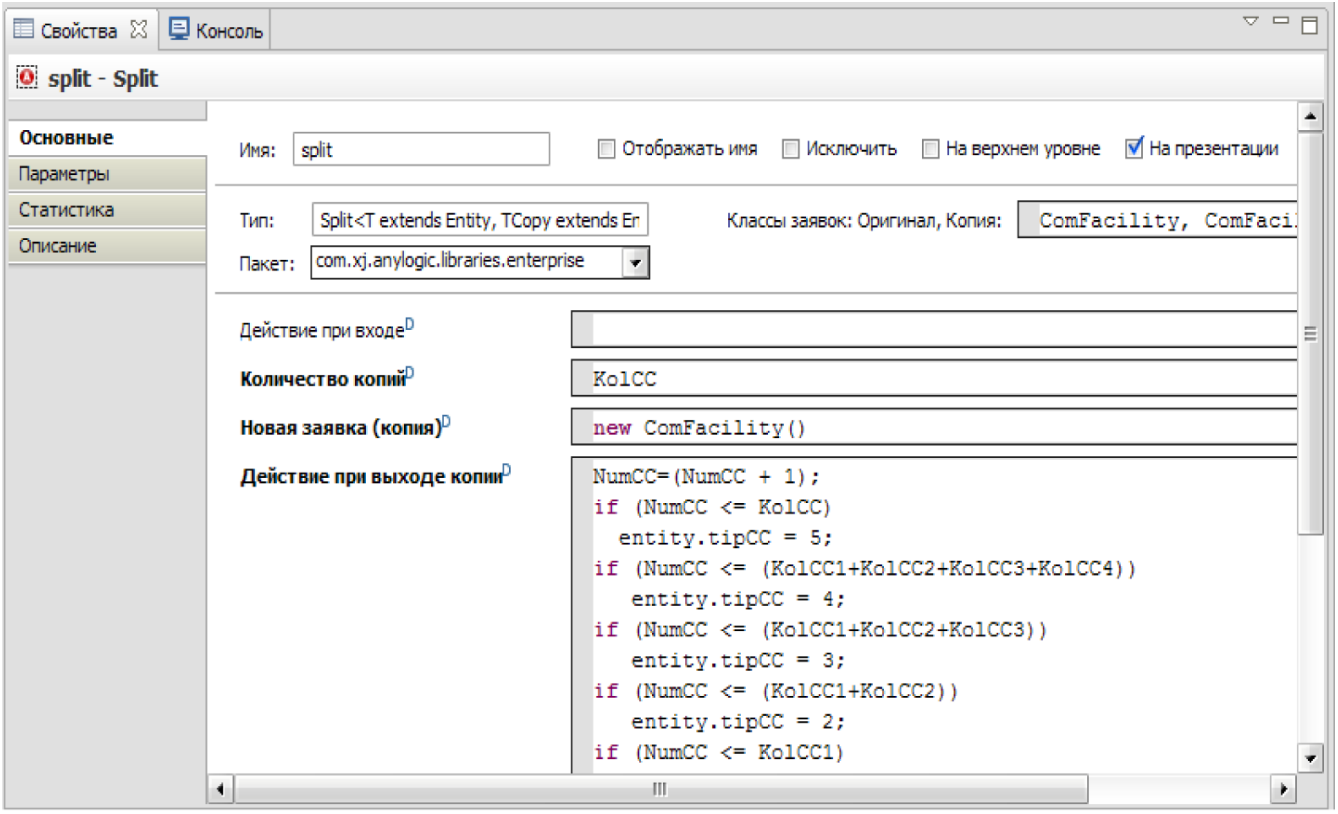


Рис. 3.73. Блок **split** с установленными свойствами

```
    entity.tipCC = 4;
if (NumCC <= (KolCC1+KolCC2+KolCC3))
    entity.tipCC = 3;
if (NumCC <= (KolCC1+KolCC2))
    entity.tipCC = 2;
if (NumCC <= KolCC1)
    entity.tipCC = 1;
```

В поле `entity.tipCC` запоминается соответствующий код типа СС.

Блок **sink** уничтожает заявку-оригинал.

3.3.4.5. Распределитель средств связи

Блок **Распределитель средств связи** предназначен для распределения СС согласно их типам, т. е. общее количество поступивших СС он должен разделить по типам.

Данный блок реализуется четырьмя блоками **SelectOutput** и одним блоком **queue** (рис. 3.74).

Блок **queue** предназначен для приема, хранения и отправки на дежурство поступающих из ремонта СС.

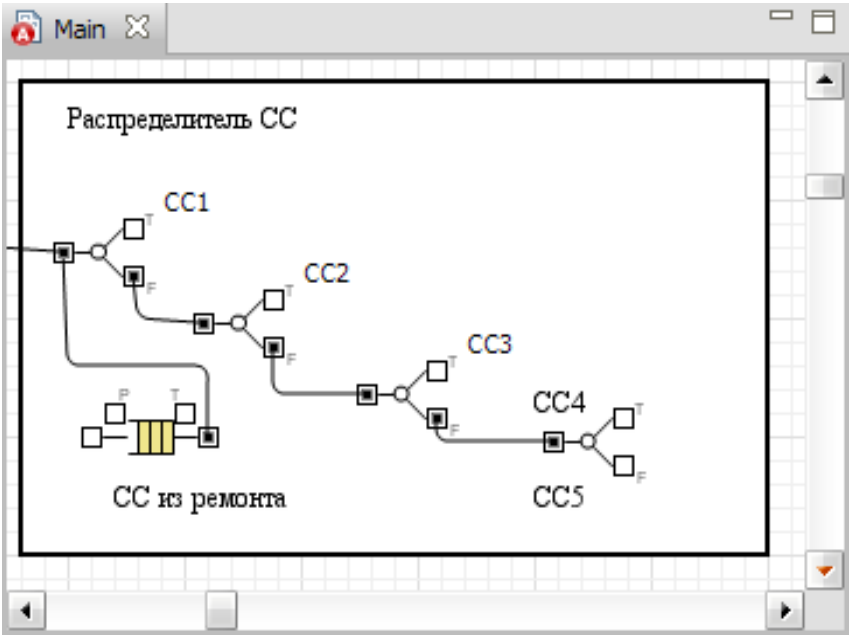


Рис. 3.74. Добавлены блоки **SelectOutput** и **queue**

1. Перетащите четыре блока **SelectOutput** и один блок **queue** из библиотеки Enterprise Library на диаграмму класса Main. Соедините их так, как показано на рис. 3.74.

2. Установите на странице **Основные** панели **Свойства** свойства блоков **SelectOutput** согласно табл. 3.4.

Таблица 3.4

Имя	Отображать имя	Класс заявки:	Выход true выбирается	Условие
CC1	Установите флажки	ComFacility	При выполнении условия	entity.tipCC==1
CC2		ComFacility		entity.tipCC==2
CC3		ComFacility		entity.tipCC==3
CC4		ComFacility		entity.tipCC==4

3. CC пятого типа будут направлены на выход false блока CC4, поэтому пятый элемент **SelectOutput** не нужен.

4. Оставьте имя блока **queue** и не устанавливайте флажок **Отображать имя**.

5. Установите **Вместимость** `100` и флажок **Включить сбор статистики**.

6. Перетащите из палитры **Презентация** три элемента **text** и в соответствующих полях **Текст:** введите текст, как на рис. 3.74.

3.3.4.6. Создание нового класса активного объекта

На рис. 3.75 показан в окончательном виде сегмент **Постановка на дежурство** после добавления блока **На дежурство**. Для добавления этого блока, который должен выполнить «связь» между сегментом **Постановка на дежурство** и сегментом **Имитация дежурства**, создадим новый класс активного объекта.

1. На панели **Проект** щелкните правой кнопкой мыши Main, с которым вы работаете в данный момент, и выберите **Создать/Класс активного объекта** из контекстного меню.

2. Откроется окно **Новый класс активного объекта**.

3. Задайте в поле **Имя:** имя нового класса **Degurstvo**.

4. Если нужно, в поле **Описание**: введите описание сущности, моделируемой этим классом.

5. Щелкните кнопку **Готово**.

Создайте область просмотра на диаграмме класса **Degurstvo** для размещения элементов сегмента **Имитация дежурства**.

1. В **Палитре** выделите **Презентация**. Перетащите элемент **Область просмотра** в нужное место.

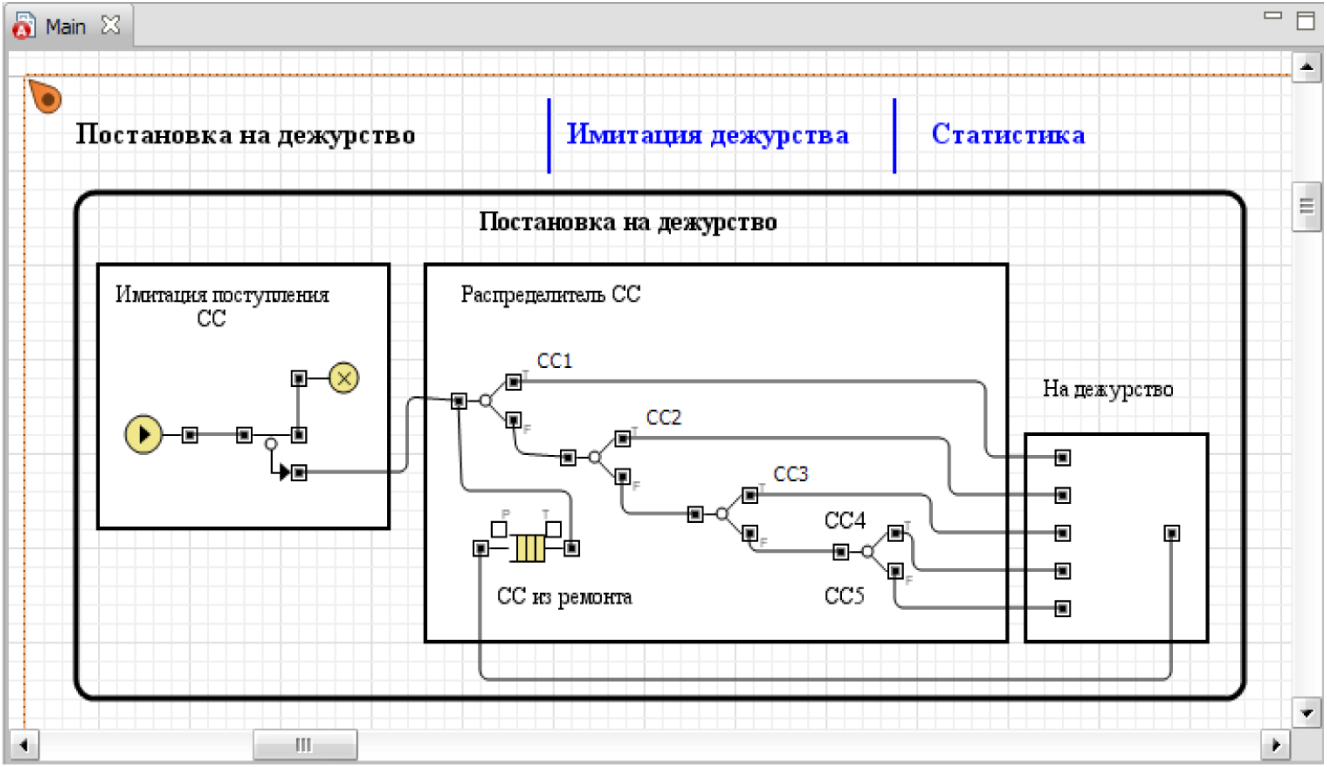


Рис. 3.75. Сегмент **Постановка на дежурство**

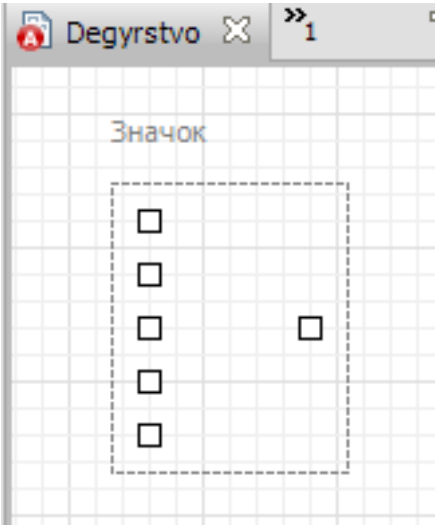


Рис. 3.76. Добавлены на **Degyrstvo** шесть портов

2. Перейдите на страницу **Основные** панели **Свойства**.
3. В поле **Имя:** введите `Degyr`.
4. Задайте, как будет располагаться область просмотра относительно ее якоря, с помощью элемента управления **Выравнивать по:** `Верхн. левому углу`.
5. Выберите режим масштабирования из выпадающего списка **Масштабирование:** `Подогнать под окно`.

3.3.4.7. Создание элемента нового класса активного объекта

Созданный новый класс активного объекта **Degyrstvo** является вложенным объектом. Как вы помните, нам нужно сделать так, чтобы СС передавались на дежурство в сегмент **Имитация дежурства** из сегмента **Постановка на дежурство**, а отремонтированные СС после ремонта из сегмента **Имитация дежурства** возвращались в сегмент **Постановка на дежурство**.

1. Перетащите шесть элементов **Порт** ▣ из палитры **Основная** и разместите так, как показано на рис. 3.76. Автоматически они будут объединены прямоугольником (с пунктирными линиями) и появится надпись **Значок**.
2. Возвратитесь на диаграмму класса Main.
3. На панели **Проект** выделите **Degyrstvo**, перетащите на диаграмму класса Main и соедините так, как на рис. 3.75.

3.3.4.8. Переключение между областями просмотра

Области просмотра используются как для навигации по графическому редактору во время создания модели, так и для навигации по окну презентации во время выполнения модели.

Чтобы перейти к другой области просмотра в режиме создания модели:

1. Щелкните мышью в графическом редакторе, чтобы сделать его активным.
2. Щелкните по кнопке панели инструментов **Области просмотра** и выберите из выпадающего списка, к какой именно области просмотра вы хотите перейти.

Чтобы перейти к другой области просмотра в режиме выполнения модели:

1. Щелкните правой кнопкой мыши в области обрисовки окна презентации, выберите пункт контекстного меню **Область** и затем выберите из списка, к какой именно области просмотра вы хотите перейти.
2. Или же щелкните кнопку панели инструментов **Показать область**... и выберите из выпадающего списка, к какой именно области просмотра вы хотите перейти (эта кнопка принадлежит секции панели инструментов **Вид**, и возможно, чтобы она стала видна, Вам нужно будет вначале показать эту секцию панели инструментов).

Вы можете также добавлять свои собственные элементы презентации, щелчком на которых будет производиться переход к той или иной области просмотра. Воспользуйтесь последним.

1. В **Палитре** выделите **Презентация**. Перетащите элемент **text**, разместите и введите в поле **Текст:** `Постановка на дежурство`, как на рис. 3.75.
2. На панели **Свойства** выделите **Динамические** и в поле **Действие по щелчку:** введите следующий Java код: `get_Main().Postanovka.navigateTo();`
3. Перетащите второй элемент **text**, разместите и введите в поле **Текст:** `Имитация дежурства`.
4. На панели **Свойства** выделите **Динамические** и в поле **Действие по щелчку:** введите следующий Java код: `get_Main().Degyr.navigateTo();`
5. Проделайте то же для **Статистика**. Введите Java код: `get_Main().statistika.navigateTo();`

3.3.5. Сегмент Имитация дежурства

3.3.5.1. Ввод исходных данных

Организуйте ввод исходных данных для сегмента **Имитация дежурства**. Для ввода исходных данных используйте также элемент **Параметр**, а для изменения их в ходе моделирования — элемент **Бегунок**.

1. Перетащите элемент **Прямоугольник** на элемент **Область просмотра**, если вы хотите видеть данные в ходе моделирования. Если нет, поместите этот элемент вне области просмотра.
2. Перетащите элемент **text** и на странице **Основные** панели **Свойства** в поле **Текст:** введите `Initial_data_D` (здесь `D` — дежурство).
3. В **Палитре** выделите **Основная**. Перетащите элементы **Параметр** и **Ползунок** на элемент с именем `Initial_data_D` и разместите их так, как показано на рис. 3.77.
4. На странице **Основные** панели **Свойства** каждого элемента **Параметр** установите свойства согласно табл. 3.5.
5. На странице **Основные** панели **Свойства** каждого элемента **Бегунок** `stoimRem1 … stoimRem5` и `timeRem1 … timeRem5` установите: **Минимальное значение:** `1`, **Максимальное значение:** `100`.
6. На странице **Основные** панели **Свойства** каждого элемента **Бегунок** `timeOtk1 … timeOtk5` установите: **Минимальное значение:** `1`, **Максимальное значение:** `1000`.
7. У всех элементов **Бегунок** установите флажок **Связать с:** и в поле введите имя соответствующего элемента **Параметр**.

3.3.5.2. Вывод результатов моделирования

Здесь выводятся только затраты на ремонт неисправных СС по их типам и в целом за все СС (см. рис. 3.77).

1. Перетащите элемент **Прямоугольник** и расположите его так, как на рис. 3.77.
2. Перетащите элемент **text** и в поле **Текст:** введите `Затраты на ремонт`.
3. Перетащите шесть элементов **Параметр** и разместите их в прямоугольнике согласно рис. 3.77.
4. Дайте имена элементам **Параметр** согласно рис. 3.77. Установите флажок **Отображать имя** и тип `double`.

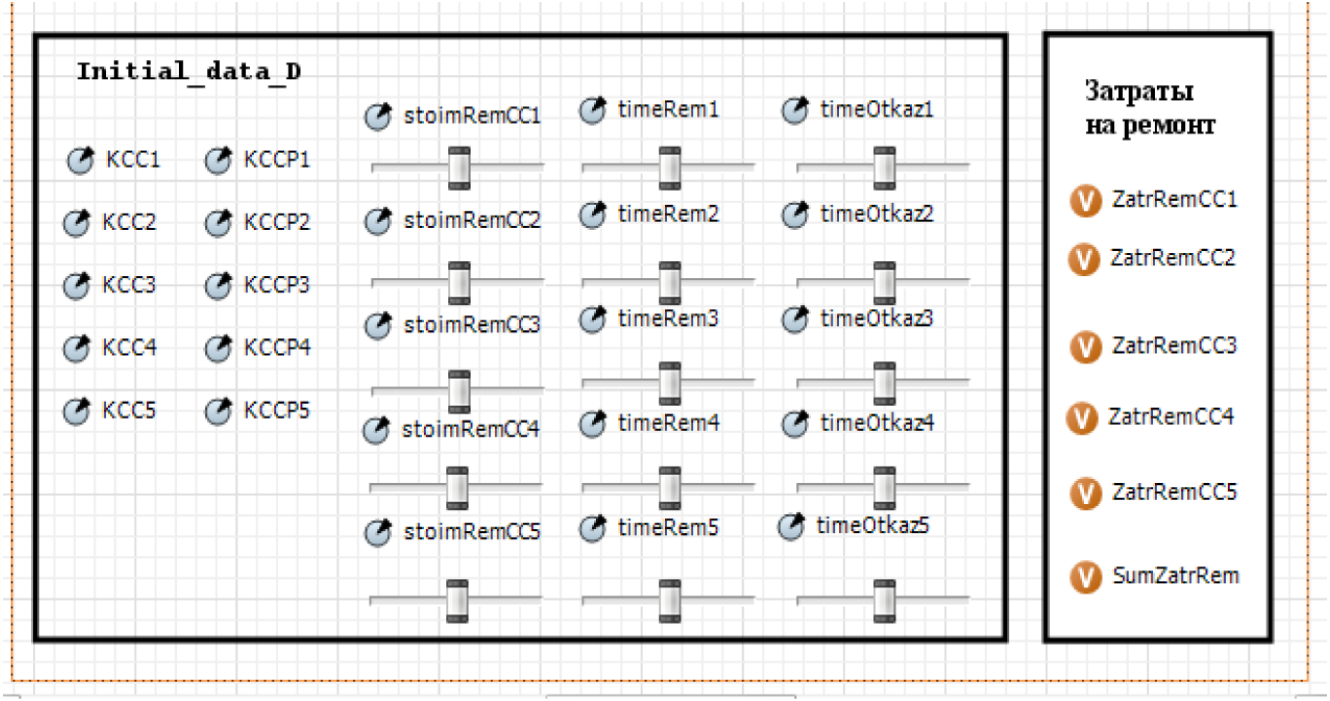


Рис. 3.77. Размещение элементов на Initial_data_D

Таблица 3.5

Свойства элементов Параметр на `Initial_data_D`

Имя	Тип	Значение по умолчанию	Отображать имя
KCC1	int	55	Установить флажок во всех элементах
KCC2	int	100	
KCC3	int	60	
KCC4	int	45	
KCC5	int	60	
KCCP1	int	2	
KCCP2	int	4	
KCCP3	int	4	
KCCP4	int	3	
KCCP5	int	4	
stoimRemCC1	double	17	
stoimRemCC2	double	18	
stoimRemCC3	double	16	
stoimRemCC4	double	20	
stoimRemCC5	double	21	
timeRem1	double	6.5	
timeRem2	double	4.2	
timeRem3	double	2.8	
timeRem4	double	3.0	
timeRem5	double	5.5	
timeOtkaz1	double	373	
timeOtkaz2	double	301	
timeOtkaz3	double	482	
timeOtkaz4	double	325	
timeOtkaz5	double	470	
kol_master	int	3	

3.3.5.3. Событийная часть сегмента Имитация дежурства

Реализация событийной части сегмента показана на рис. 3.78.

1. Перетащите на диаграмму класса **Degyrstvo** последовательно: пять блоков queue, пять блоков delay, пять блоков queue, один блок delay и соедините их так как показано на рис. 3.78.
2. На странице **Основные** панели **Свойства** каждого блока установите свойства согласно табл. 3.6.
3. Перетащите три элемента **Прямоугольник** и разместите их так, как на рис. 3.78.
4. Перетащите три элемента **text** и введите названия в соответствующие поля **Текст:** согласно рис. 3.78.

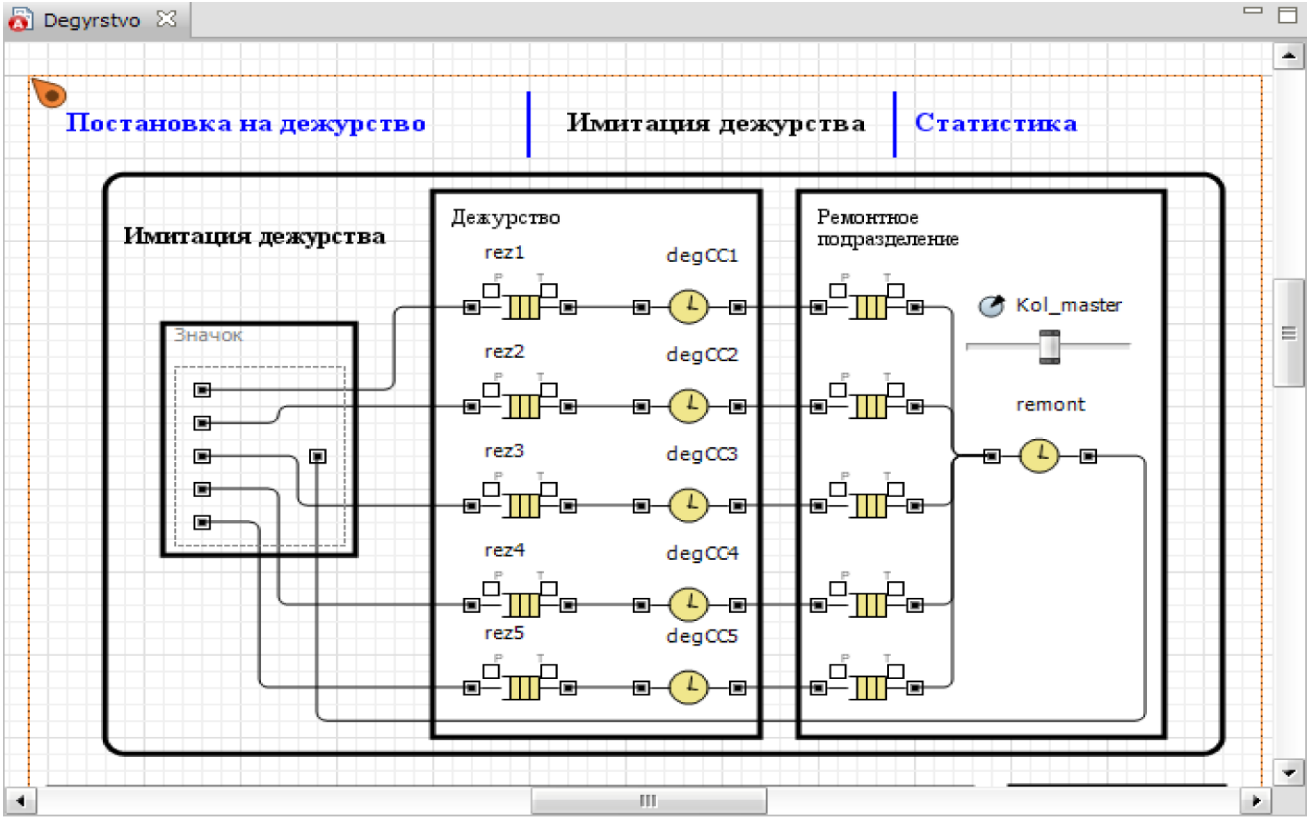


Рис. 3. 78. Сегмент **Имитация дежурства**

Таблица 3.6

Элементы сегмента Имитация дежурства и их свойства

queue		
Имя	Вместимость	Действие при выходе
rez1	KCCP1	`entity.timeOtkaz=1/timeOtkaz1`
rez2	KCCP2	`entity.timeOtkaz=1/timeOtkaz2`
rez3	KCCP3	`entity.timeOtkaz=1/timeOtkaz3`
rez4	KCCP4	`entity.timeOtkaz=1/timeOtkaz4`
rez5	KCCP5	`entity.timeOtkaz=1/timeOtkaz5`
neispr1	KCC1	`entity.timeMeanRem=1/timeRem1;`
neispr2	KCC2	`entity.timeMeanRem=1/timeRem2;`
neispr3	KCC3	`entity.timeMeanRem=1/timeRem3;`
neispr4	KCC4	`entity.timeMeanRem=1/timeRem4;`
neispr5	KCC5	`entity.timeMeanRem=1/timeRem5;`

delay		
Имя	Время задержки	Вместимость
degCC1	`exponential(entity.timeOtkaz)`	KCC1
degCC2	`exponential(entity.timeOtkaz)`	KCC2
degCC3	`exponential(entity.timeOtkaz)`	KCC3
degCC4	`exponential(entity.timeOtkaz)`	KCC4
degCC5	`exponential(entity.timeOtkaz)`	KCC5
remont	`exponential(entity.timeMeanRem)`	3

Примечание.

Для всех элементов поставить флажок **Включить сбор статистики**.

Для элемента **delay** с именем remont также ввести Java коды в следующие свойства:

Действие при входе `entity.nach=time();`

Действие при выходе

```
entity.konec=time();
if (entity.tipCC == 1)
   {ZatrRemCC1=ZatrRemCC1+((entity.konec-
entity.nach)*stoimRemCC1);
    SumZatrRem=SumZatrRem+((entity.konec-
entity.nach)*stoimRemCC1);}
if (entity.tipCC == 2)
   {ZatrRemCC2=ZatrRemCC2+((entity.konec-
entity.nach)*stoimRemCC2);
   SumZatrRem=SumZatrRem +((entity.konec-
entity.nach)*stoimRemCC2);}
if (entity.tipCC == 3)
```

```
   {ZatrRemCC3=ZatrRemCC3 +((entity.konec-
entity.nach)*stoimRemCC3);
   SumZatrRem=SumZatrRem +((entity.konec-
entity.nach)*stoimRemCC3);}
if (entity.tipCC == 4)
   {ZatrRemCC4=ZatrRemCC4+((entity.konec-
entity.nach)*stoimRemCC4);
   SumZatrRem=SumZatrRem +((entity.konec-
entity.nach)*stoimRemCC4);}
if (entity.tipCC == 5)
   {ZatrRemCC5=ZatrRemCC5+((entity.konec-
entity.nach)*stoimRemCC5);
   SumZatrRem=SumZatrRem +((entity.konec-
entity.nach)*stoimRemCC5);}
```

3.3.5.4. Переключение между областями просмотра

1. В **Палитре** выделите **Презентация**. Перетащите элемент **text**, разместите и введите в поле **Текст:** `Постановка на дежурство`, как на рис. 3.78.

2. На панели **Свойства** выделите **Динамические** и в поле **Действие по щелчку:** введите следующий Java код: `get_Main().Postanovka.navigateTo();`

3. Перетащите второй элемент **text**, разместите и введите в поле **Текст:** `Имитация дежурства`.

4. На панели **Свойства** выделите **Динамические** и в поле **Действие по щелчку:** введите следующий Java код: `Degyr.navigateTo();`

5. Проделайте то же для **Статистика**. Введите Java код: `get_Main().statistika.navigateTo();`

3.3.6. Сегмент Статистика

Результаты моделирования выводятся в сегментах **Постановка на дежурство** и **Имитация дежурства**. Тем не менее, организуем вывод результатов моделирования, можно сказать, в более презентабельном виде. Для этого создадим сегмент **Статистика** (рис. 3.79).

1. Вернитесь на диаграмму класса Main. Создайте область просмотра для размещения элементов сегмента **Статистика**.

2. В **Палитре** выделите **Презентация**. Перетащите элемент **Область просмотра** в нужное место.

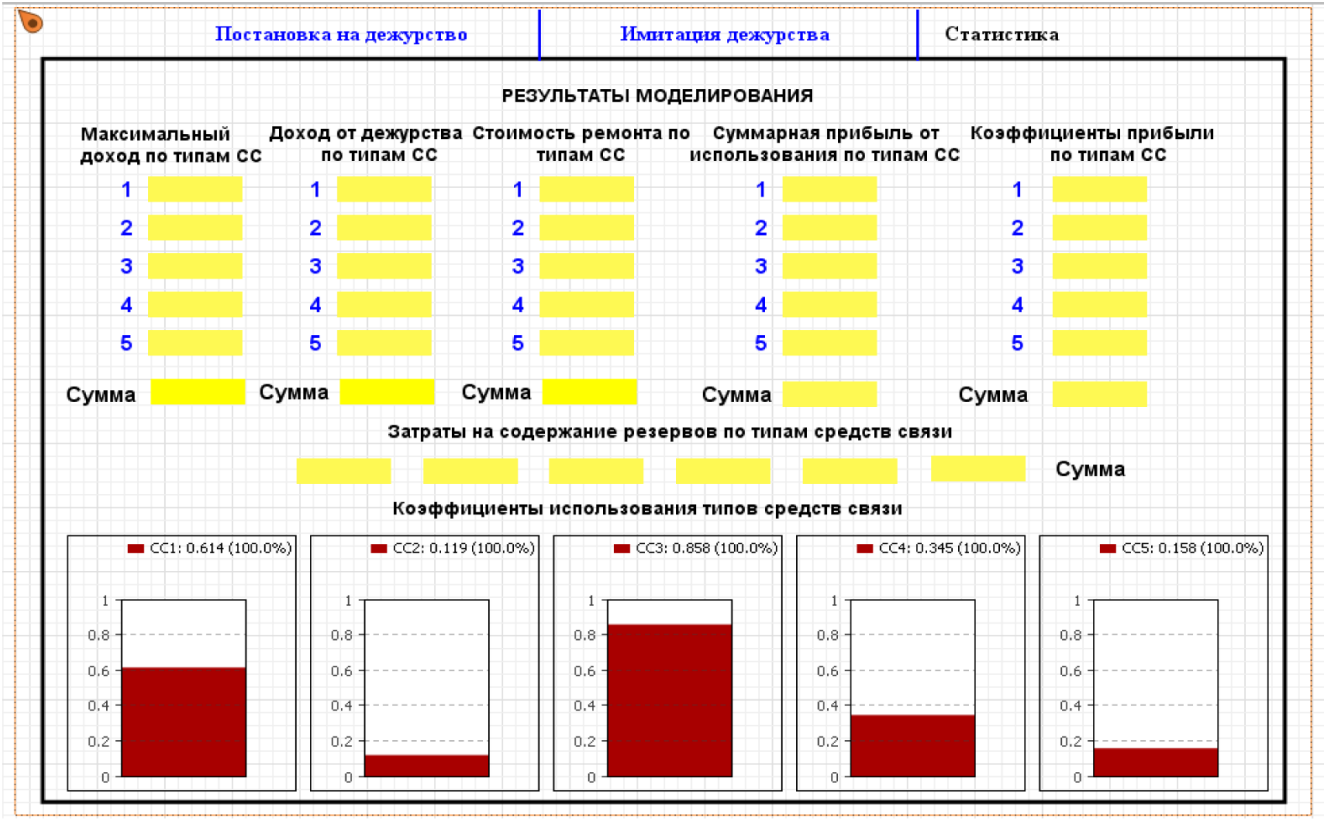


Рис. 3.79. Сегмент **Статистика**

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

4. В поле **Имя:** введите `Statistika`.

5. Задайте, как будет располагаться область просмотра относительно ее якоря, с помощью элемента управления **Выравнивать по:** `Верхн. левому углу`.

6. Выберите режим масштабирования из выпадающего списка **Масштабирование:** `Подогнать под окно`.

7. Перетащите элемент **Прямоугольник** и разместите как на рис. 3.79.

8. Перетащите элемент **text** и в поле **Текст:** `введите Результаты моделирования`.

9. Перетаскивая элемент **text**, разместите и введите в соответствующие поля **Текст:** надписи, как на рис. 3.79.

3.3.6.1. Использование элемента Текстовое поле

Текстовое поле является простейшим текстовым элементом управления, позволяющим пользователю вводить небольшие объемы текста. Вы можете также связать этот элемент управления с переменной или параметром типа String, double или int.

1. Перетащите элемент **Текстовое поле** из палитры **Элементы управления** и разместите согласно рис. 3.79.

2. Выделяя последовательно каждый элемент **Текстовое поле**, переходите на страницу **Основные** панели **Свойства** и в поле **Имя:** давайте имя элементу согласно табл. 3.7.

Таблица 3.7

Имена элементов Текстовое поле

1	2	3	4	5	6
Максимальный доход по типам СС					
editbox1	editbox2	editbox3	editbox4	editbox5	editbox
Доход от дежурства по типам СС и всего					
editbox11	editbox12	editbox13	editbox14	editbox15	editbox16
Стоимость ремонта по типам СС и всего					
editbox21	editbox22	editbox23	editbox24	editbox25	editbox26
Суммарная прибыль от использования СС и всего					
editbox31	editbox32	editbox33	editbox34	editbox35	editbox36
Коэффициенты прибыли по типам СС и всего					
editbox41	editbox42	editbox43	editbox44	editbox45	editbox46
Затраты на содержание резервов по типам СС и всего					
editbox6	editbox7	editbox8	editbox9	editbox10	editbox17

3.3.6.2. Использование элемента Диаграмма

С помощью диаграмм AnyLogic позволяет динамически визуализировать данные, собираемые в результате работы модели. Набор диаграмм схож с тем, что предлагается программой MS Excel. Библиотека обладает мощным и удобным интерфейсом, не требующим при создании диаграммы программирования.

Термин диаграмма используется для обозначения, как обычных диаграмм, так и гистограмм. Гистограммы отображают статистически обработанные данные в виде функции плотности вероятности (PDF) и интегральной функции распределения (CDF), учитывающие все когда-либо добавленные на гистограмму значения. Диаграммы отображают текущие значения элементов данных (а некоторые — также недавнюю историю изменения значений).

AnyLogic поддерживает несколько видов диаграмм.

Простые диаграммы:

столбиковая диаграмма;
диаграмма с накоплением;
круговая диаграмма.

Диаграммы с историей (временные диаграммы):

график;
временной график;
временная диаграмма с накоплением;
временная цветовая диаграмма.

Используйте диаграмму с накоплением.

Диаграмма с накоплением показывает вклад нескольких элементов данных в суммирующий результат в виде столбцов, расположенных друг над другом. Высота каждого столбца пропорциональна значению соответствующего элемента данных.

Самый нижний столбец соответствует элементу данных, добавленному Вами на диаграмму первым, и т. д. Не допускается присутствие отрицательных значений — в этом случае возникнет ошибка. Вы можете изменить направление роста столбца и его ширину с помощью свойств **Направление** и **Относительная ширина** (они находятся на странице свойств **Внешний вид**).

1. Перетащите элемент **Диаграмма с накоплением** из палитры **Статистика** и разместите согласно рис. 3.79.

2. Выделяя последовательно каждый элемент **Диаграмма с накоплением**, переходите на страницу **Внешний вид** панели **Свойства** и установите:

Смещение по оси X: 40
Смещение по оси Y: 20
Ширина: 92
Высота: 138
Относительная ширина: 52
Направление: вертикальное

Цвет текста, цвет фона и цвет границы установите по своему усмотрению.

3. После установки этих свойств выделите левый элемент.
4. Перейдите на страницу **Основные** панели **Свойства**.
5. Щелкните **Добавить элемент данных**.
6. В поле **Заголовок:** введите CC1.
7. В поле **Значение:** введите Java код

```
degyrstvo.degCC1.statsUtilization.mean()
```

8. Установите **Масштаб:** Фиксированный и **Обновлять автоматически**. Цвет столбика, который будет отображать коэффициент использования группы CC одного типа, установите по своему усмотрению

9. Проделайте пп. 3…7 для остальных 2…5 элементов **Диаграмма с накоплением**. При этом в поле **Заголовок:** вводите: CC2, CC3, CC4, CC5 соответственно.

10. В поле **Значение:** вводите Java коды также для элементов 2…5 соответственно:

```
degyrstvo.degCC2.statsUtilization.mean()
degyrstvo.degCC3.statsUtilization.mean()
degyrstvo.degCC4.statsUtilization.mean()
degyrstvo.degCC5.statsUtilization.mean()
```

На этом реализация сегмента **Статистика** завершена. Осталось только организовать переключение между областями просмотра.

3.3.6.3. Переключение между областями просмотра

1. В **Палитре** выделите **Презентация**. Перетащите элемент **text**, разместите и введите в поле **Текст:** Постановка на дежурство, как на рис. 3.79.

2. На панели **Свойства** выделите **Динамические** и в поле **Действие по щелчку:** введите следующий Java код: Postanovka.navigateTo();

3. Перетащите второй элемент **text**, разместите и введите в поле **Текст:** `Имитация дежурства`.

4. На панели **Свойства** выделите **Динамические** и в поле **Действие по щелчку:** введите следующий Java код: `degyrstvo.degyr.navigateTo();`

5. Проделайте то же для **Статистика**. Введите Java код: `statistika.navigateTo();`

Все три сегмента модели построены. Теперь вернемся к выводу результатов моделирования с использованием способа событие.

3.3.7. Использование способа Событие

Событие является самым простым способом планирования действий в модели. События часто используются для моделирования задержек и таймаутов. Вы можете сделать это и с помощью срабатывающих по таймауту переходов диаграмм состояний, но использование событий является более рациональным. В некоторых случаях поведение может быть смоделировано только с помощью таймеров.

Есть три типа событий.

Событие, происходящее по истечении таймаута. Оно используется тогда, когда Вам нужно запланировать выполнение какого-то действия на определенный момент времени (отстоящий на заданное количество времени (таймаут) от текущего момента). Событие, происходящее по истечению таймаута, предоставляет дополнительные возможности: вы можете сделать событие циклическим, либо же вообще управлять этим событием «вручную».

Событие, происходящее при выполнении заданного условия. Оно используется тогда, когда Вам нужно отслеживать выполнение определенного условия и производить какое-то действие при его происхождении.

Событие, происходящее с заданной интенсивностью. Оно используется для моделирования потока независимых событий (пуассоновский поток). Это часто требуется при моделировании поступления, например, заявок в системах массового обслуживания.

AnyLogic поддерживает еще один тип события, задаваемый уже другим модельным элементом — *динамическое событие*. Динамические события используются для планирования сразу нескольких одновременных и независимых событий. Например, канал связи, параллельно передающий произвольное количество сообщений,

может быть смоделирован с помощью динамических таймеров, создаваемых для каждого сообщения.

Для вывода результатов моделирования воспользуйтесь событием, происходящим по истечении таймаута.

1. Перетащите элемент **Событие**  из палитры **Модель** на диаграмму класса активного объекта. Измените его имя на `ResulsModeling`. Нажмите Enter.
2. Установите флажок **Отображать имя**.
3. С помощью выпадающего списка **Тип события:** выберите **По таймауту**.
4. Установите **Режим: Циклический**.
5. **Время первого срабатывания (абсолютное)** установите равным 0.
6. Установите **Период:** 1.
7. В поле **Действие** введите Java код, который будет выполняться при появлении этого события.

```
//Расчет результатов по CC1
DoxMaxCC1=round((DoxMaxCC1+degyrstvo.KCC1*doxDegCC1)*100);
DoxMaxCC1=DoxMaxCC1/100;
DoxDegCC1=round((DoxDegCC1+degyrstvo.degCC1.size()*
doxDegCC1)*100);
DoxDegCC1=DoxDegCC1/100;
ZatrResCC1=round((ZatrResCC1+degyrstvo.KCCP1*
zatrResCC1)*100);
ZatrResCC1=ZatrResCC1/100;
ZatrRemCC1=round((degyrstvo.ZatrRemCC1)*100);
ZatrRemCC1=ZatrRemCC1/100;
PribCC1=DoxDegCC1-(ZatrResCC1+ZatrRemCC1);
koefPribCC1=round((PribCC1/DoxMaxCC1)*1000);
koefPribCC1=koefPribCC1/1000;

//Расчет результатов по CC2
DoxMaxCC2=round((DoxMaxCC2+degyrstvo.KCC2*doxDegCC2)*100);
DoxMaxCC2=DoxMaxCC2/100;
DoxDegCC2=round((DoxDegCC2+degyrstvo.degCC2.size()*
doxDegCC2)*100);
DoxDegCC2=DoxDegCC2/100;
ZatrResCC2=round((ZatrResCC2+degyrstvo.KCCP2*
zatrResCC2)*100);
ZatrResCC2=ZatrResCC2/100;
ZatrRemCC2=round((degyrstvo.ZatrRemCC2)*100);
ZatrRemCC2=ZatrRemCC2/100;
PribCC2=DoxDegCC2-(ZatrResCC2+ZatrRemCC2);
```

```
koefPribCC2=round((PribCC2/DoxMaxCC2)*1000);
koefPribCC2=koefPribCC2/1000;
```

//Расчет результатов по CC3

```
DoxMaxCC3=round((DoxMaxCC3+degyrstvo.KCC3*doxDegCC3)*100);
DoxMaxCC3=DoxMaxCC3/100;
DoxDegCC3=round((DoxDegCC3+degyrstvo.degCC3.size()*
doxDegCC3)*100);
DoxDegCC3=DoxDegCC3/100;
ZatrResCC3=round((ZatrResCC3+degyrstvo.KCCP3*
zatrResCC3)*100);
ZatrResCC3=ZatrResCC3/100;
ZatrRemCC3=round((degyrstvo.ZatrRemCC3)*100);
ZatrRemCC3=ZatrRemCC3/100;
PribCC3=DoxDegCC3-(ZatrResCC3+ZatrRemCC3);
koefPribCC3=round((PribCC3/DoxMaxCC3)*1000);
koefPribCC3=koefPribCC3/1000;
```

//Расчет результатов по CC4

```
DoxMaxCC4=round((DoxMaxCC4+degyrstvo.KCC4*doxDegCC4)*100);
DoxMaxCC4=DoxMaxCC4/100;
DoxDegCC4=round((DoxDegCC4+degyrstvo.degCC4.size()*
doxDegCC4)*100);
DoxDegCC4=DoxDegCC4/100;
ZatrResCC4=round((ZatrResCC4+degyrstvo.KCCP4*
zatrResCC4)*100);
ZatrResCC4=ZatrResCC4/100;
ZatrRemCC4=round((degyrstvo.ZatrRemCC4)*100);
ZatrRemCC4=ZatrRemCC4/100;
PribCC4=DoxDegCC4-(ZatrResCC4+ZatrRemCC4);
koefPribCC4=round((PribCC4/DoxMaxCC4)*1000);
koefPribCC4=koefPribCC4/1000;
```

//Расчет результатов по CC5

```
DoxMaxCC5=round((DoxMaxCC5+degyrstvo.KCC5*doxDegCC5)*100);
DoxMaxCC5=DoxMaxCC5/100;
ZatrResCC5=round((ZatrResCC5+degyrstvo.KCCP5*
zatrResCC5)*100);
ZatrResCC5=ZatrResCC5/100;
DoxDegCC5=round((DoxDegCC5+degyrstvo.degCC5.size()*
doxDegCC5)*100);
DoxDegCC5=DoxDegCC5/100;
ZatrRemCC5=round((degyrstvo.ZatrRemCC5)*100);
ZatrRemCC5=ZatrRemCC5/100;
PribCC5=DoxDegCC5-(ZatrResCC5+ZatrRemCC5);
```

```
koefPribCC5=round((PribCC5/DoxMaxCC5)*1000);
koefPribCC5=koefPribCC5/1000;
```

//Расчет суммарных результатов

```
SumZatrRes=ZatrResCC1+ZatrResCC2+ZatrResCC3+
ZatrResCC4+ZatrResCC5;
SumDoxMax=DoxMaxCC1+DoxMaxCC2+DoxMaxCC3+DoxMaxCC4+
DoxMaxCC5;
SumDoxDeg=DoxDegCC1+DoxDegCC2+DoxDegCC3+DoxDegCC4+
DoxDegCC5;
SumZatrRem=round((degyrstvo.SumZatrRem)*100);
SumZatrRem=SumZatrRem/100;
SumPribil=round((SumDoxDeg-
(SumZatrRes+SumZatrRem))*100);
SumPribil=SumPribil/100;
koefPribil=round((SumPribil/SumDoxMax)*1000);
koefPribil=koefPribil/1000;
```

//вывод максимального дохода, дохода от дежурства

```
editbox1.setText(DoxMaxCC1);
editbox11.setText(DoxDegCC1);
editbox2.setText(DoxMaxCC2);
editbox12.setText(DoxDegCC2);
editbox3.setText(DoxMaxCC3);
editbox13.setText(DoxDegCC3);
editbox4.setText(DoxMaxCC4);
editbox14.setText(DoxDegCC4);
editbox5.setText(DoxMaxCC5);
editbox15.setText(DoxDegCC5);
editbox.setText(SumDoxMax);
editbox16.setText(SumDoxDeg);
```

//вывод стоимости ремонта по типам СС и всего

```
editbox21.setText(ZatrRemCC1);
editbox22.setText(ZatrRemCC2);
editbox23.setText(ZatrRemCC3);
editbox24.setText(ZatrRemCC4);
editbox25.setText(ZatrRemCC5);
editbox26.setText(SumZatrRem);
```

//вывод чистой прибыли от использования СС

```
editbox31.setText(PribCC1);
editbox32.setText(PribCC2);
editbox33.setText(PribCC3);
editbox34.setText(PribCC4);
editbox35.setText(PribCC5);
```

```
editbox36.setText(SumPribil);

//вывод коэффициентов прибыли по типам СС и всего
editbox41.setText(koefPribCC1);
editbox42.setText(koefPribCC2);
editbox43.setText(koefPribCC3);
editbox44.setText(koefPribCC4);
editbox45.setText(koefPribCC5);
editbox46.setText(koefPribil);

//вывод затрат на содержание резервов СС
editbox6.setText(ZatrResCC1);
editbox7.setText(ZatrResCC2);
editbox8.setText(ZatrResCC3);
editbox9.setText(ZatrResCC4);
editbox10.setText(ZatrResCC5);
editbox17.setText(SumZatrRes);
```

Из кода и комментария к нему следует, что циклически будут рассчитываться и выводится результаты моделирования.

Для округления результатов моделирования (коэффициентов прибыли до трёх знаков после запятой, а абсолютных величин прибыли и затрат — до двух знаков) использован метод round(). Предварительно результаты умножались на 1000 и 100 соответственно, а потом делились на эти же величины.

Для вывода результатов моделирования в текстовые поля `editbox` используется функция `setText()`, в качестве аргумента которой указывается имя соответствующего элемента **Параметр**.

Запустите модель. При наличии ошибок, устраните их. Система AnyLogic имеет достаточно эффективный отладчик. При двойном щелчке ошибки в окне **Ошибки** открывается код с выделенной ошибкой либо сообщается, например, о том, что параметр не инициализирован.

На рис. 3.80 и рис. 3.81 показаны фрагменты работы модели.

Для нахождения оптимального состава резервных средств связи и мастеров-ремонтников с целью получения максимальной прибыли перейдём к рассмотрению возможных экспериментов в AnyLogic и методике проведения интересующего нас оптимизационного эксперимента.

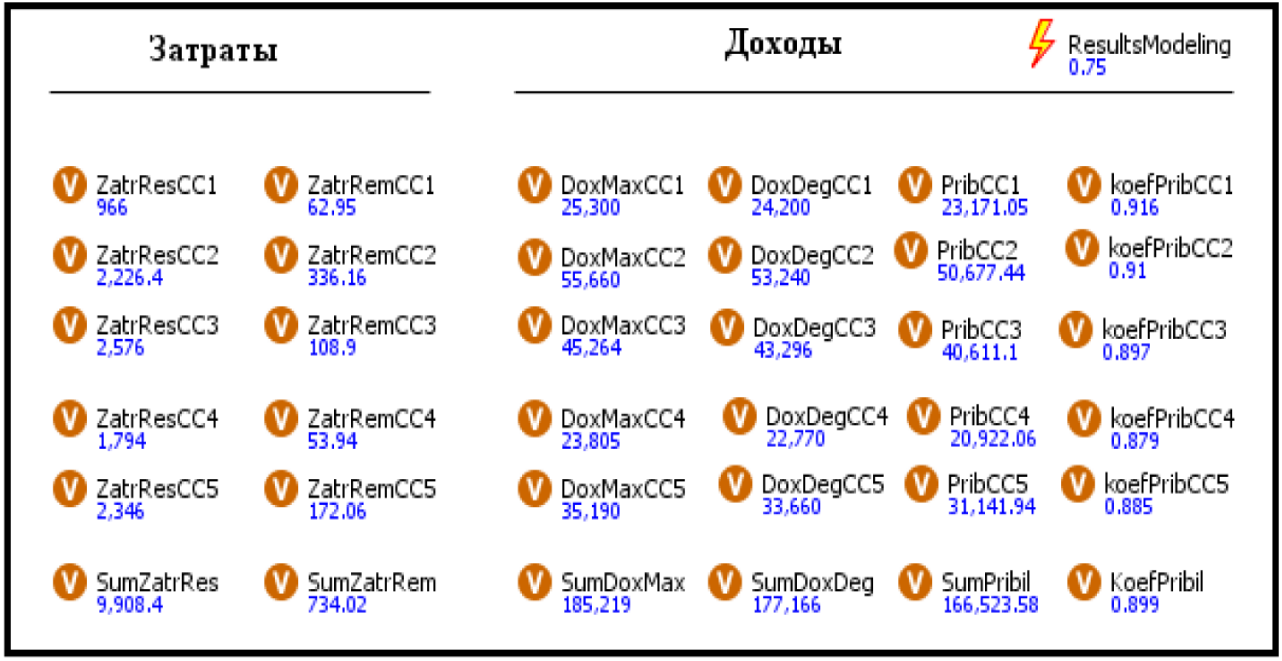


Рис. 3.80. Результаты моделирования

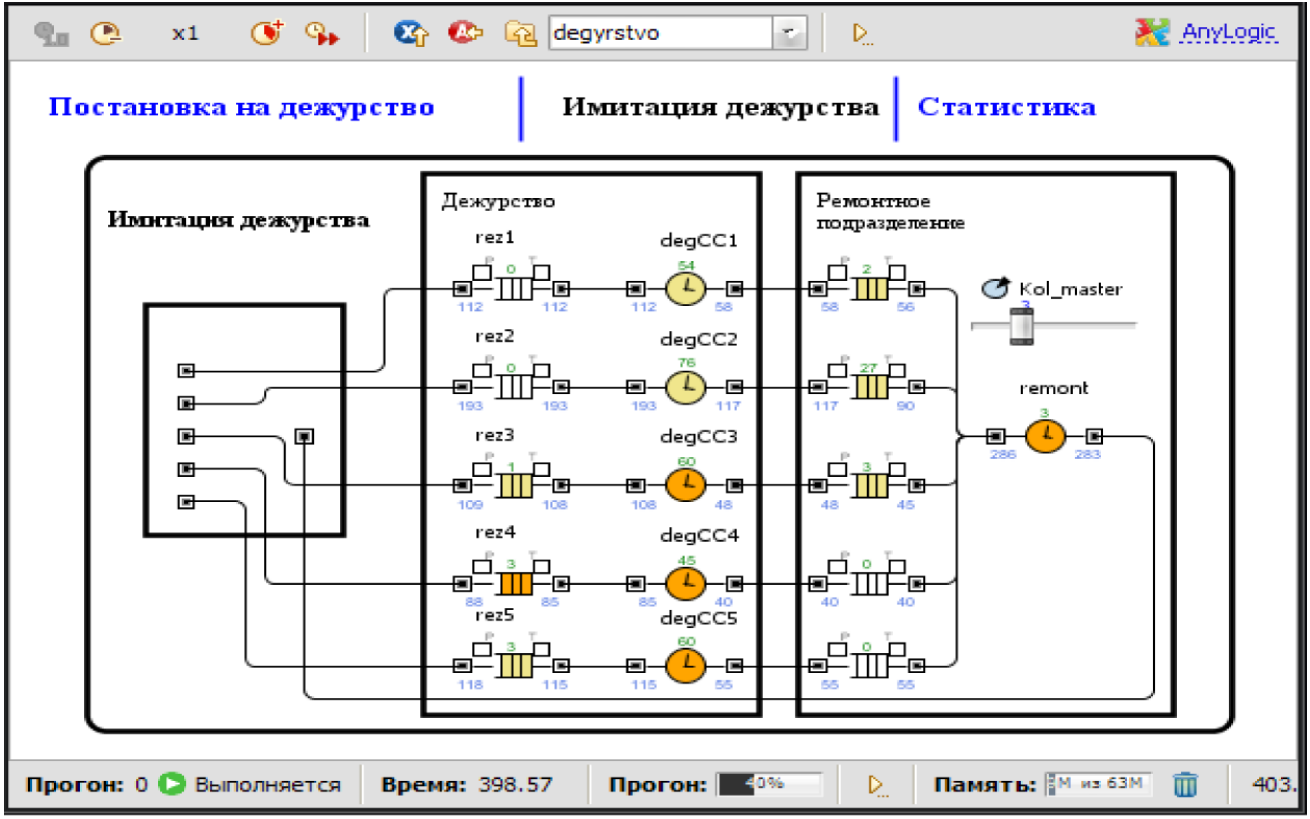


Рис. 3.81. Фрагмент работы модели. Сегмент **Имитация дежурства**

3.3.8. Проведение экспериментов

AnyLogic предоставляет пользователю возможность провести следующие эксперименты:

простой эксперимент;
оптимизация;
варьирование переменных;
сравнение прогонов;
Монте-Карло;
анализ чувствительности;
калибровка;
нестандартный.

Последние пять экспериментов доступны только в AnyLogic Professional.

3.3.8.1. Простой эксперимент

Простой эксперимент запускает модель с заданными значениями параметров, поддерживает режимы виртуального и реального времени, анимацию, отладку модели.

При создании модели автоматически создается один простой эксперимент, названный Simulation. Именно такой эксперимент мы с вами и рассматривали до сих пор.

Эксперимент этого типа используется в большинстве случаев. Другие эксперименты нужны тогда, когда важную роль играют значения параметров модели. То есть вам нужно проанализировать, как они влияют на поведение или эффективность моделируемой системы или если вам нужно найти оптимальные параметры вашей модели.

Далее в рамках данного пособия мы остановимся на экспериментах оптимизации и варьирования переменных, овладев методиками проведения которых, вы самостоятельно сможете выполнять в AnyLogic и другие эксперименты.

3.3.8.2. Связывание параметров

Начиная создавать модель в AnyLogic, мы ничего не говорили об экспериментах и особенностях их проведения. Поэтому все исходные данные разместили на Initial_data_PD (рис. 3.69) и Initial_data_D (рис. 3.77) так, как нам представлялось удобным для построения модели и управления ею в ходе проведения простого эксперимента.

Однако при проведении экспериментов и наличии в модели, как в нашем случае, вложенных объектов, необходимо связывать параметры, размещенные на корневом объекте и вложенных объектах. Связывание необходимо потому, что изменять в ходе эксперимента можно только параметры корневого объекта.

В результате связывания значение параметра любого уровня вложенного объекта будет равно значению параметра объекта самого верхнего уровня.

Но следует иметь в виду, что *связываются только параметры одного типа* и что *передача значения параметра производится лишь параметру объекта, находящегося ниже уровнем в иерархическом дереве модели*. То есть связываются параметры последовательно от одного уровня к другому, а не через уровень или уровни.

Разместите элементы на Initial_data_D и Initial_data_ PD так, как показано на рис. 3.82 и 3.83. Внесите соответствующие изменения в модель. Обратите внимание на различие имен связываемых параметров, например, КСС_1 и КСС1.

Свяжите параметры корневого объекта Main с параметрами вложенного объекта класса Degyrstvo.

1. Откройте диаграмму класса активного объекта Main.
2. Выберите на диаграмме вложенный объект degyrstvo.
3. Перейдите на страницу **Параметры** панели **Свойства**.
4. В таблице **Параметры** в поле **Значение** введите имя параметра класса объекта-владельца Main, значение которого нужно передавать этому параметру вложенного объекта. В результате у вас должно быть так, как на рис. 3.84.

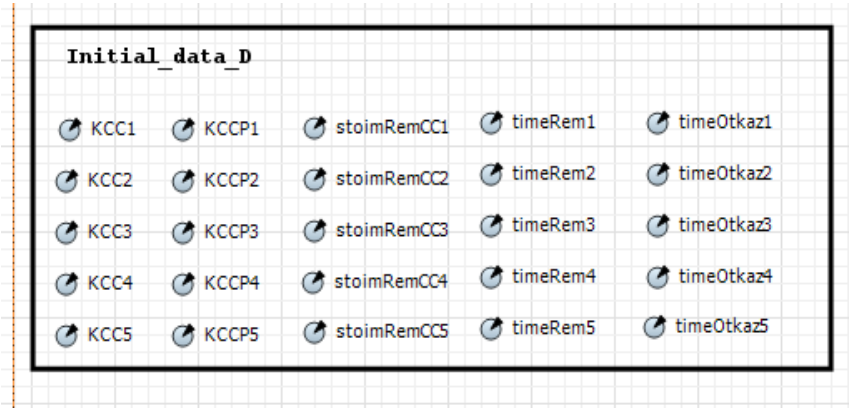


Рис. 3.82. Размещение элементов на Initial_data_D для экспериментов

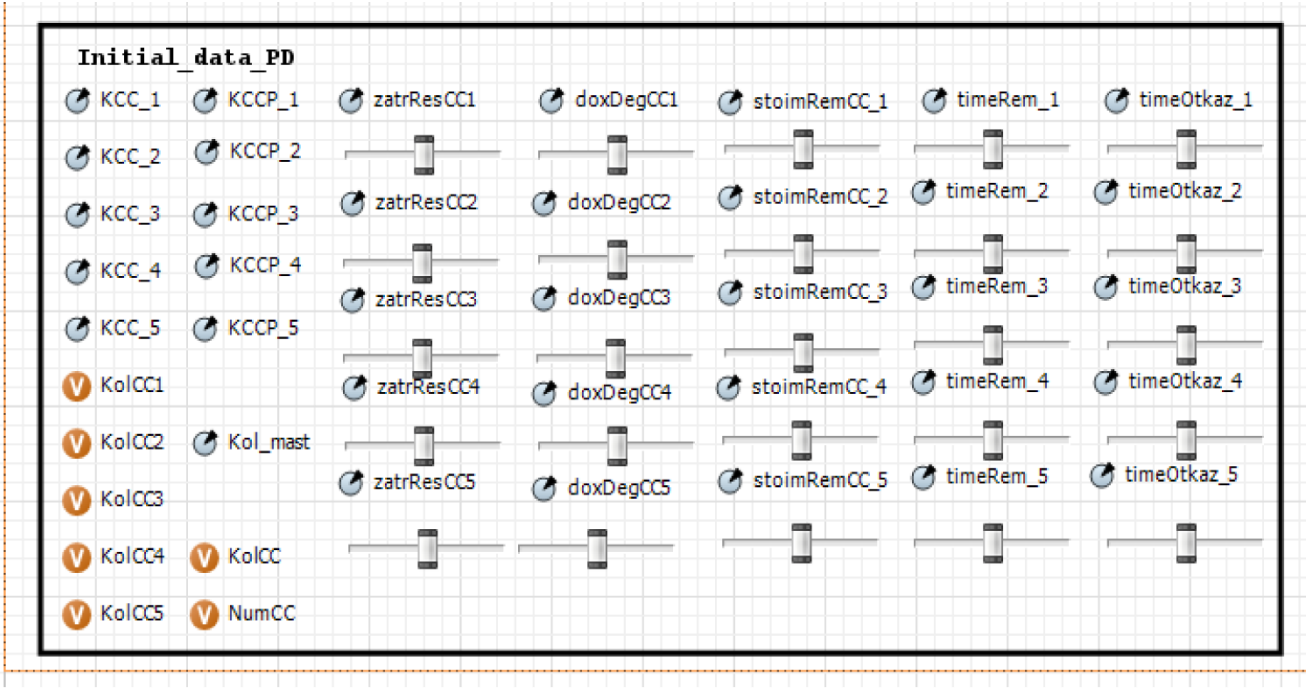


Рис. 3.83. Размещение элементов на Initial_data_PD для проведения экспериментов

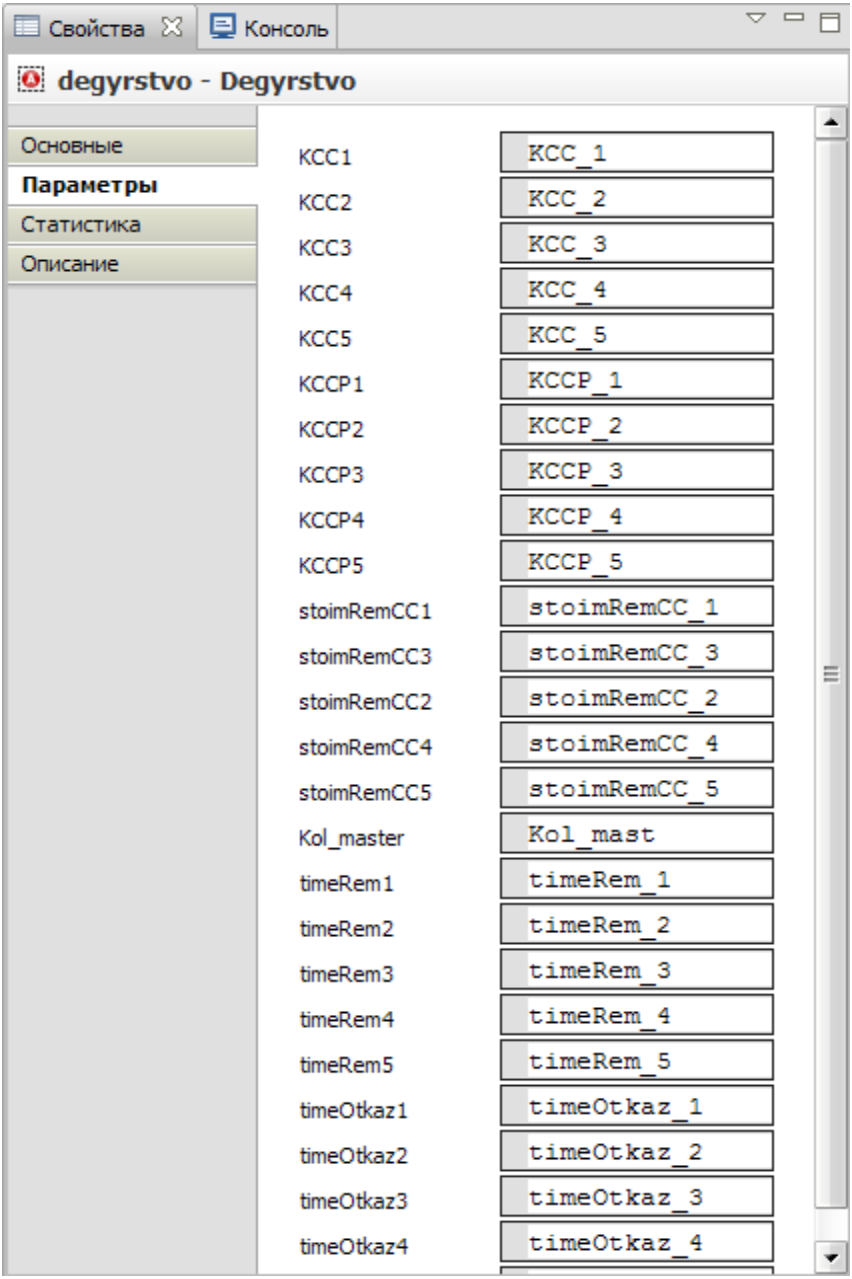


Рис. 3.84. Фрагмент страницы **Параметры** после связывания параметров

3.3.8.3. Эксперимент Оптимизация стохастических моделей

Эксперимент **Оптимизация** может проводиться в AnyLogic оптимизатором OptQuest для детерминированных и стохастических моделей.

Наша модель ComSystem стохастическая. Создайте оптимизационный эксперимент стохастической модели с целью определения максимального коэффициента прибыли в зависимости от количества резервных СС и мастеров-ремонтников.

1. В панели **Проект** щелкните правой кнопкой мыши элемент модели ComSystem и из контекстного меню выберите **Создать/ Эксперимент**.
2. В появившемся диалоговом окне из списка **Тип эксперимента:** выберите **Оптимизация** (рис. 3.85).
3. В поле **Имя** введите имя эксперимента, например, `Opt-ComSystem`. Имя эксперимента должно начинаться с заглавной буквы — таково правило названия классов в Java.
4. В поле **Корневой класс модели:** выберите Main. Этим действием вы задали корневой (главный) класс эксперимента. Объект этого класса будет играть роль корня иерархического дерева объектов модели, запускаемой оптимизационным экспериментом.
5. Если вы хотите применить к создаваемому эксперименту временные установки другого эксперимента, оставьте установленным флажок **Копировать установки модельного времени из:** и выберите эксперимент из расположенного справа выпадающего списка.
6. Оставьте выбранной опцию **Случайное начальное число (уникальные прогоны)**. В этом случае при каждом запуске модели генератор случайных чисел будет инициализироваться другим числом. Поэтому прогоны будут уникальными, невоспроизводимыми при очередном запуске модели.
7. Щелкните кнопку **Готово**. Появится страница **Основные** панели **Свойства** (рис. 3.86).
8. Установите опцию **максимизировать**.
9. Оставьте установленным флажок **Количество итераций:**. Под итерацией понимается один опыт (одно наблюдение). Количество итераций — это цель стратегического планирования эксперимента — определение количества наблюдений и уровней факторов в них для получения полной и достоверной информации о модели.

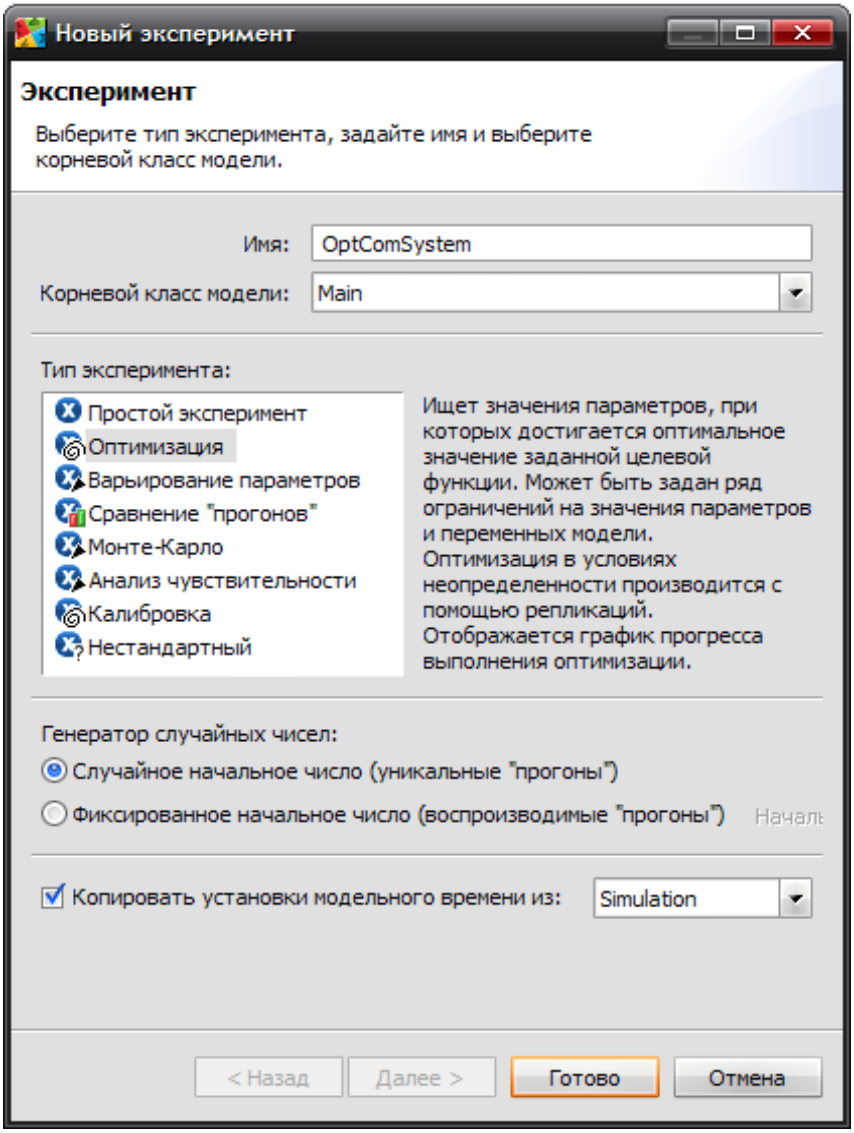


Рис. 3.85. Диалоговое окно **Новый эксперимент**

10. В нашей модели нужно менять количество резервных средств связи КССР_1…КССР_5 и количество мастеров Kol_mast, то есть всего m=6 факторов. Примем, что каждый фактор имеет три уровня k=3, тогда число итераций $I = k^m = 3^6 = 729$ [1].

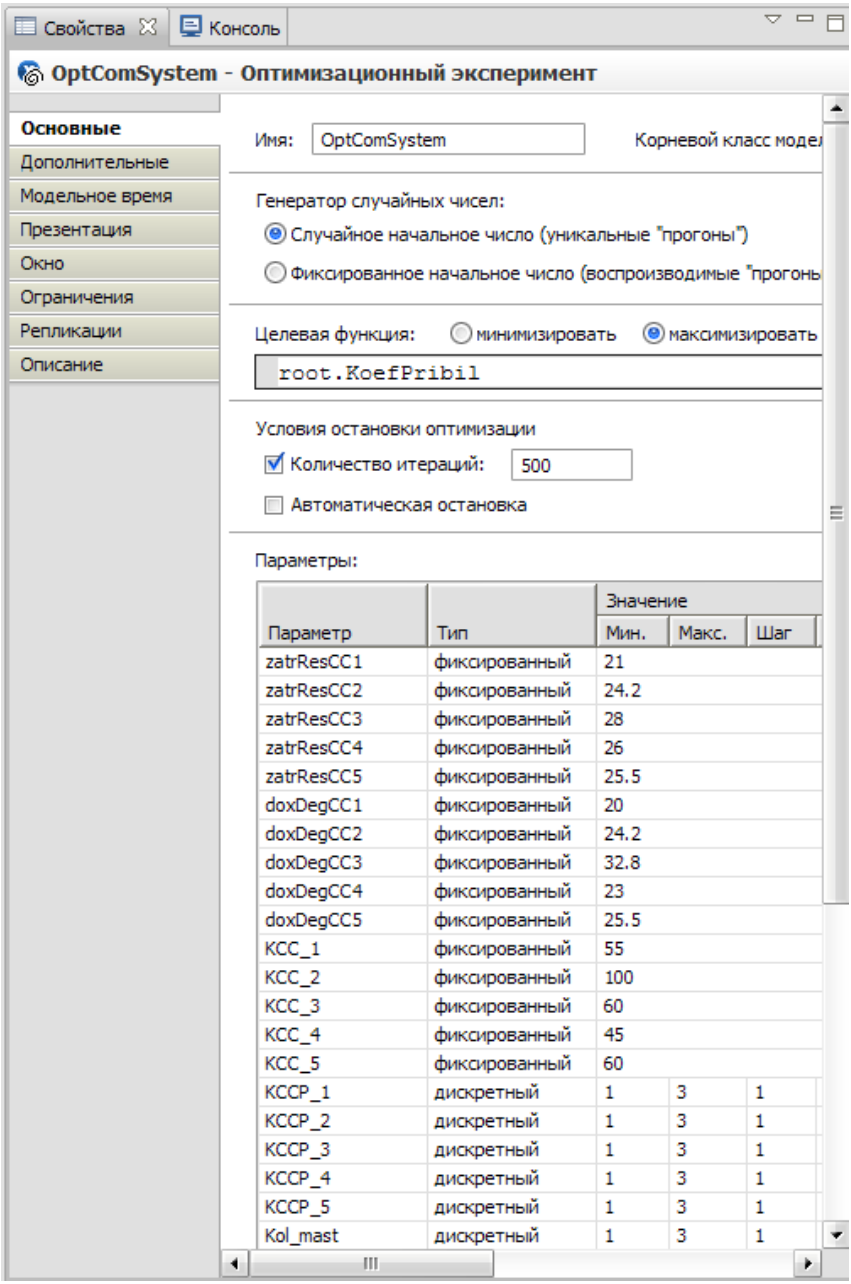


Рис. 3.86. Вкладка **Основные** оптимизационного эксперимента

11. Оставьте 500 в поле **Количество итераций:**, так как данная версия AnyLogic ограничена этим количеством итераций.

12. Задайте параметры, значения которых будут меняться. В таблице на рис. 3.86 перечислены все параметры корневого объекта Main.

13. Чтобы разрешить варьирование параметров оптимизатором, перейдите на строку с параметром KCCP_1. Щелкните мышью в ячейке **Тип**. Выберите тип параметра, отличный от значения **фиксированный**. Так как параметр KCCP_1 целочисленный типа int, выберите **дискретный.**

14. Задайте диапазон допустимых значений параметра. Для чего введите в ячейку **Мин** минимальное значение 1, в ячейку **Макс** максимальное значение 3. Так как параметр дискретный, в ячейке **Шаг** укажите величину шага 1.

15. Задайте так же остальные параметры, как на рис. 3.86.

16. Перейдите на страницу **Репликации** панели **Свойства** (рис. 3.87).

17. Установите флажок **Использовать репликации**.

18. Число репликаций (прогонов) в одной итерации (наблюдении) может быть фиксированным или переменным. *Фиксированное число репликаций*, например, при доверительной вероятности $\alpha = 0{,}95$, точности $\varepsilon = 0{,}1$ и стандартном отклонении $\sigma = 0{,}1$ может быть определено по формуле [1]:

$$N = t_\alpha^2 \frac{\sigma^2}{\varepsilon^2} = 1{,}96^2 \frac{0{,}1^2}{0{,}1^2} = 3{,}8416 \approx 4,$$

где $t_\alpha = 1{,}96$ — табулированный аргумент функции Лапласа.

Возможность *переменного количества репликаций* позволяет оптимизатору OptQust проверять на статистическую значимость разницу между средним значением целевой функции в текущей итерации (текущее среднее значение) и лучшим значением, найденным за предыдущие итерации (лучшее значение). Целью такой проверки является удаление худших решений без потери времени на их получение. Таким образом, процесс может быть значительно ускорен за счет прекращения поиска неподходящих решений вместо выполнения заданного максимального количества реализаций.

19. Выберите опцию **Фиксированное количество репликаций** и в соответствующем поле установите 4.

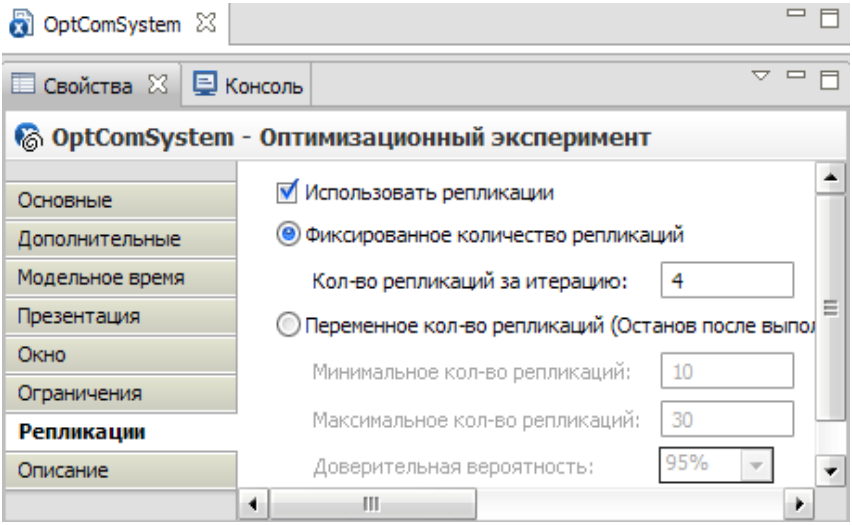


Рис. 3.87. Страница **Репликации** оптимизационного эксперимента

20. Вернитесь на страницу **Основные** и щелкните кнопку **Создать интерфейс**. Кнопка находится в правом верхнем углу страницы **Основные**. После щелчка удаляется содержимое презентации эксперимента и создается интерфейс эксперимента заново (рис. 3.88) согласно его текущим установкам (набору оптимизационных параметров и их свойствам и т. д.). Поэтому *создавать интерфейс нужно только после окончания задания параметров эксперимента*. На интерфейсе видны знаки вопросов напротив оптимизационных параметров.

21. В меню запуск выполните **ComSystem2/OptComSystem.**

22. Щелкните **Запустить оптимизацию**. Начнет выполняться эксперимент. Во время эксперимента можно видеть на графике изменение значения целевой функции. После 500*4=2 000 прогонов (см. рис. 3.89) эксперимент остановится.

23. Результаты оптимизационного эксперимента приведены на рис. 3.89. Наилучшее значение целевой функции — коэффициент прибыли — равно 0,95. Получено оно на 263 итерации при следующих оптимальных значениях параметров: КССР_1= КССР_2=2, КССР_3= КССР_4=КССР_5=1, Kol_mast=3.

Замечание. Возможно, что результат оптимизационного эксперимента будет отличаться по коэффициенту прибыли в большую сторону при снятии ограничения в 500 итераций.

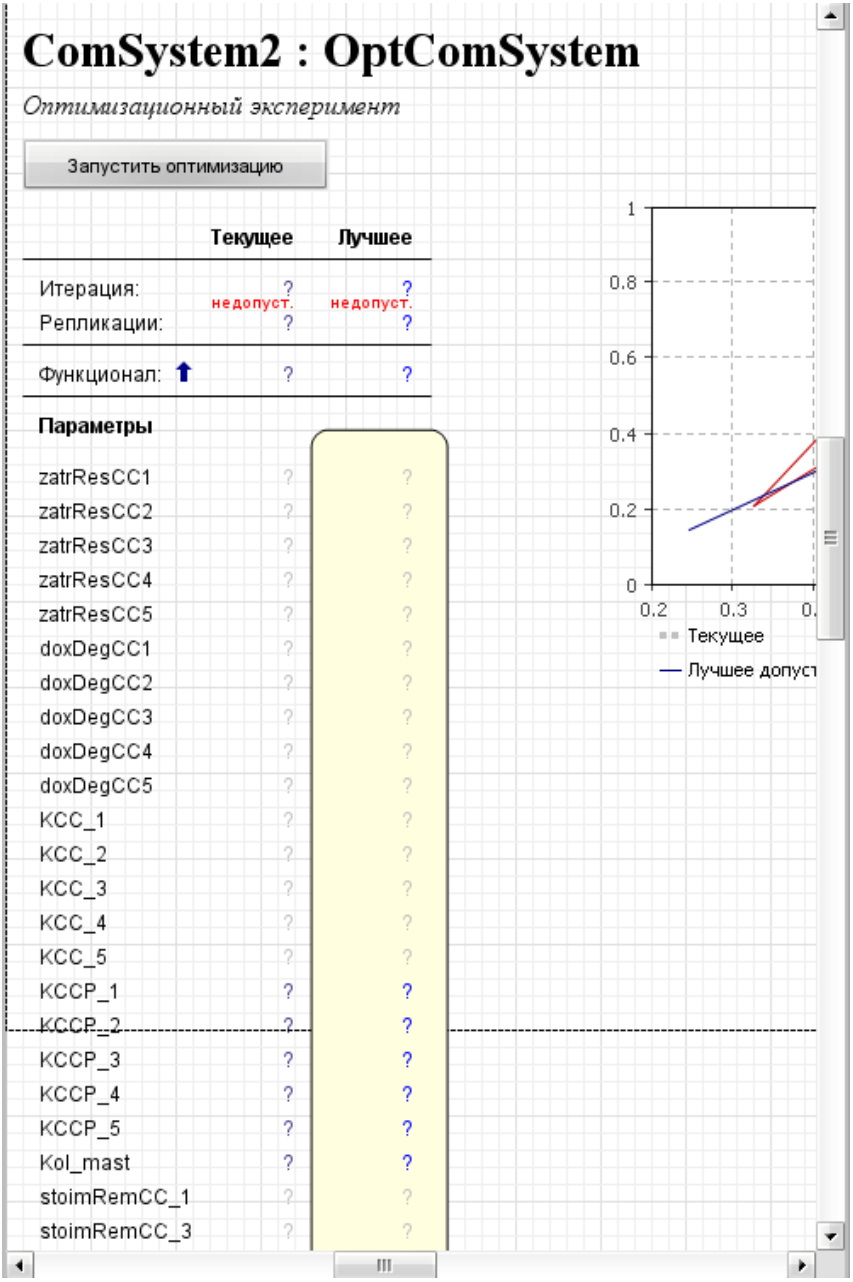


Рис. 3.88. Интерфейс оптимизационного эксперимента

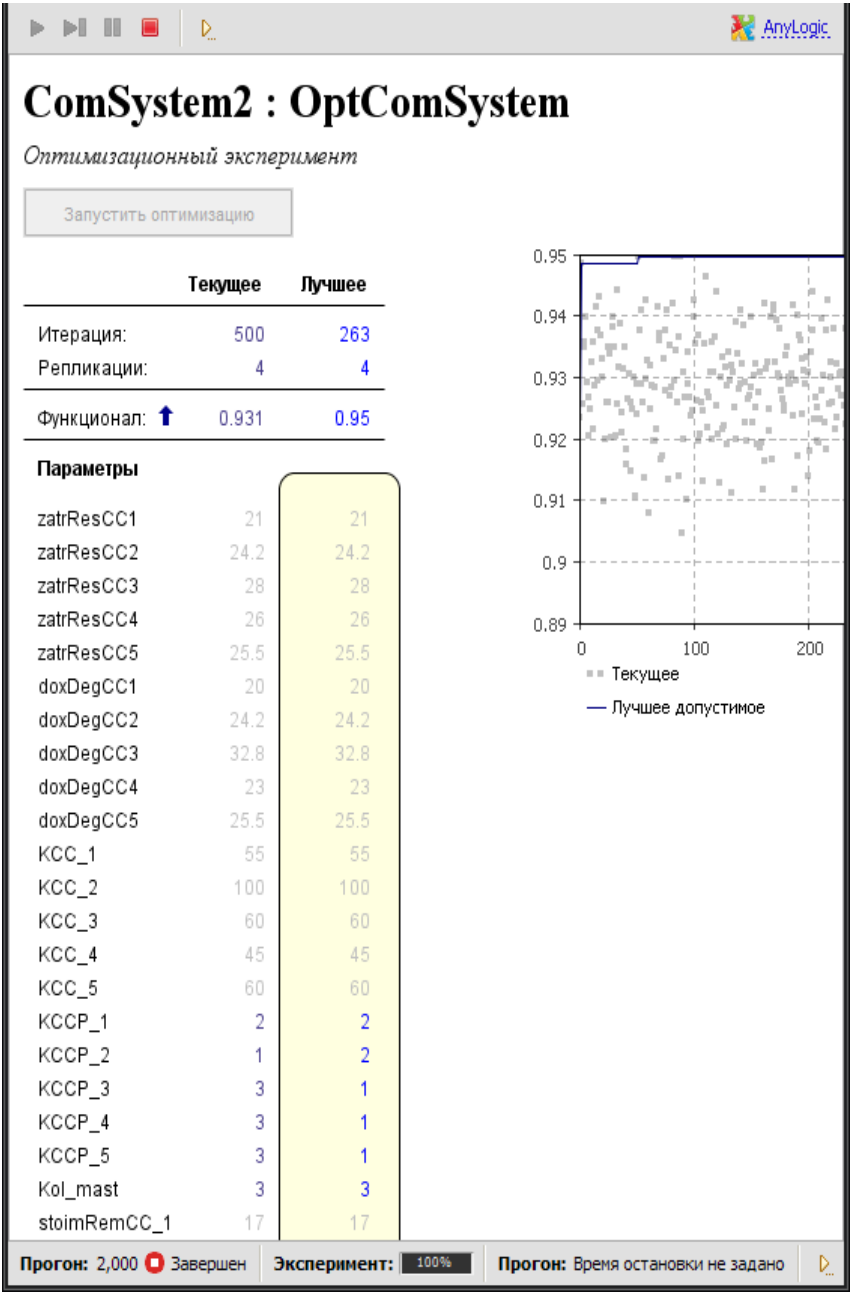


Рис. 3.89. Результаты оптимизационного эксперимента

3.3.8.4. Эксперимент Варьирование параметров

Эксперимент **Варьирование параметров** также может проводиться для детерминированных и стохастических моделей.

Создайте эксперимент **Варьирование параметров** для стохастической модели ComSystem с целью наблюдения за изменением коэффициента прибыли в зависимости от дохода за дежурство средств связи.

1. В панели **Проект** щелкните правой кнопкой мыши элемент модели ComSystem2 и из конткстного меню выберите **Создать эксперимент**.
2. В появившемся диалоговом окне из списка **Тип эксперимента:** выберите **Варьирование параметров** (рис. 3.90).

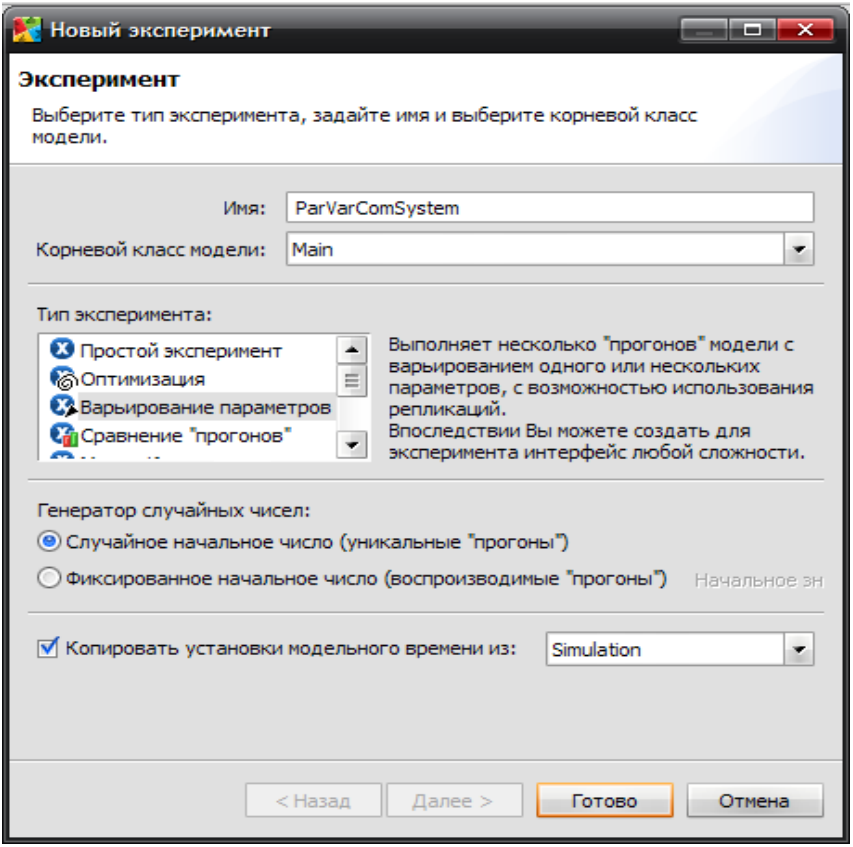


Рис. 3.90. Окно **Новый эксперимент** с выбранным экспериментом

3. В поле **Имя** введите имя эксперимента, например, `VarParComSystem`.

4. Остальные установки оставьте такими, как на рис. 3.90. Назначение их тоже, что и в оптимизационном эксперименте (см. с. 229).

5. Щелкните кнопку **Готово**. Появится страница **Основные** панели **Свойства** (рис. 3.91).

6. Обратите внимание, что на странице **Основные** по сравнению с оптимизационным экспериментом отсутствуют опции **минимизировать**, **максимизировать**, **Количество итераций**. Последнее определяется AnyLogic в зависимости от диапазонов и шагов изменения параметров.

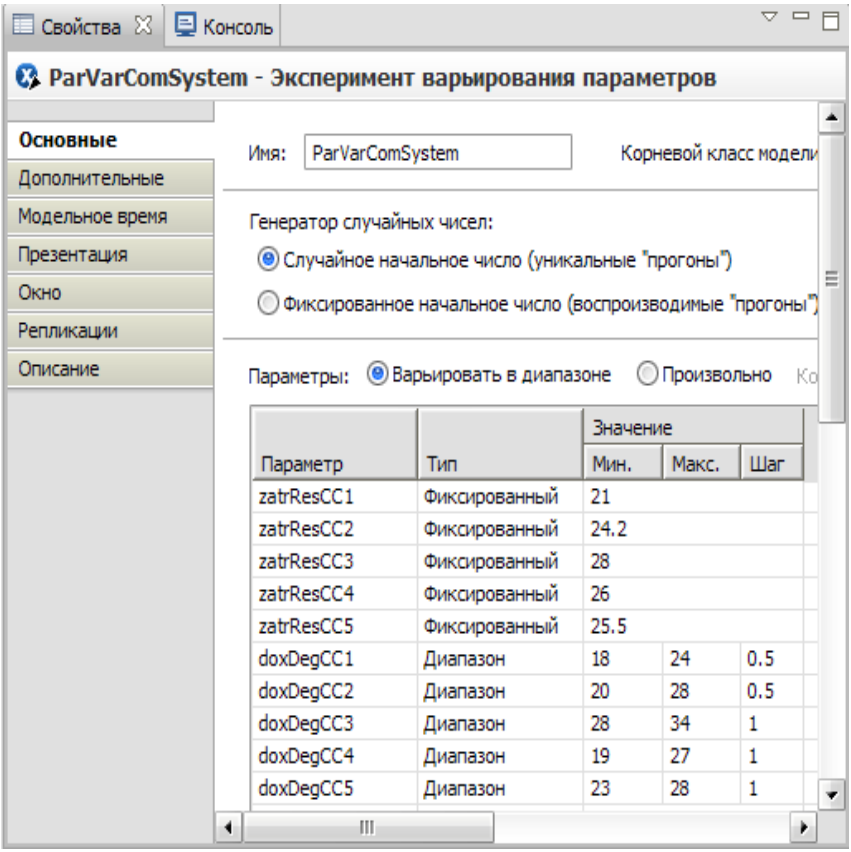


Рис. 3.91. Страница **Основные** эксперимента варьирования параметров

7. Задайте диапазон допустимых значений параметра doxDegCC1. Перейдите в таблице на рис. 3.91 на строку с этим параметром. Щелкните мышью в ячейке **Тип**. Выберите тип параметра, отличный от значения **фиксированный**. Параметр doxDegCC1 типа double, поэтому он может изменяться в диапазоне. Выберите **Диапазон**. В ячейку **Мин** введите минимальное значение `18`, в ячейку **Макс** — максимальное значение `24`, в ячейке **Шаг** укажите величину шага `0.5`.

8. Задайте также остальные параметры, как на рис. 3.91.

9. Перейдите на страницу **Репликации** панели **Свойства** и установите флажок **Использовать репликации**. В поле **Кол-во репликаций за итерацию**, установите, как и в предыдущем эксперименте, `4`.

10. Вернитесь на страницу **Основные** и щелкните кнопку **Создать интерфейс**.

11. В эксперименте **Варьирование параметров** в отличие от эксперимента **Оптимизация** интерфейс создает пользователь. Связано это с тем, что выходными результатами данного эксперимента могут быть любые показатели моделируемой системы.

12. Создайте интерфейс, показанный на рис. 3.92. Здесь вы видите график, который будет отображать значение коэффициента прибыли для каждой итерации.

13. Перетащите элемент **График** из палитры **Статистика** на диаграмму активного класса.

14. Щелкните **Добавить набор данных**.

15. Установите опцию **Набор данных**. **Заголовок:** `KoefPribil`. **Набор данных:** `dataset`. Установите **Не обновлять автоматически**.

16. Перейдите на страницу **Дополнительные** и установите: **X:** `260`, **Y:** `100`, **Ширина:** `510`, **Высота:** `400`, **Цвет фона:** `Нет заливки`, **Цвет границы:** `Нет линии`.

17. Перейдите на страницу **Внешний вид**. Установите: **Смещение по X:** `40`, **Смещение по Y:** `20`, **Ширина:** `450`, **Высота:** `330`.

18. Также из палитры **Статистика** перетащите элемент **Набор данных**. Установите опцию **Не обновлять автоматически**.

19. Из палитры **Основная** перетащите элемент **Простая переменная**. На странице **Основные** панели **Свойства** в поле **Имя:**

введите `valueOfKoefPribil`. Установите **Уровень доступа:** `public`. **Тип:** `double`.

20. Щелкните диаграмму класса. Перейдите на страницу **Дополнительные** панели **Свойства** и введите коды:

в поле **Действие после прогона модели:**

```
valueOfKoefPribil = root.KoefPribil;
```

в поле **Действие после итерации**

```
dataset.add(getCurrentIteration(), valueOfKoefPribil);
```

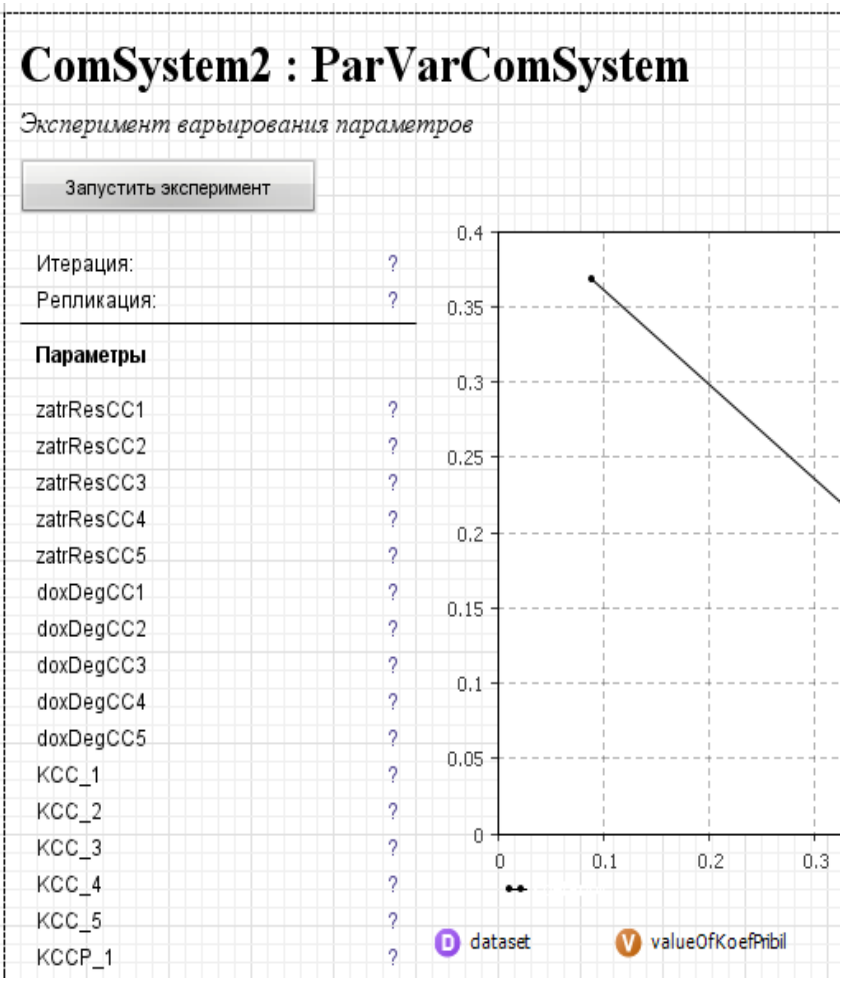


Рис. 3.92. Интерфейс эксперимента варьирования параметров

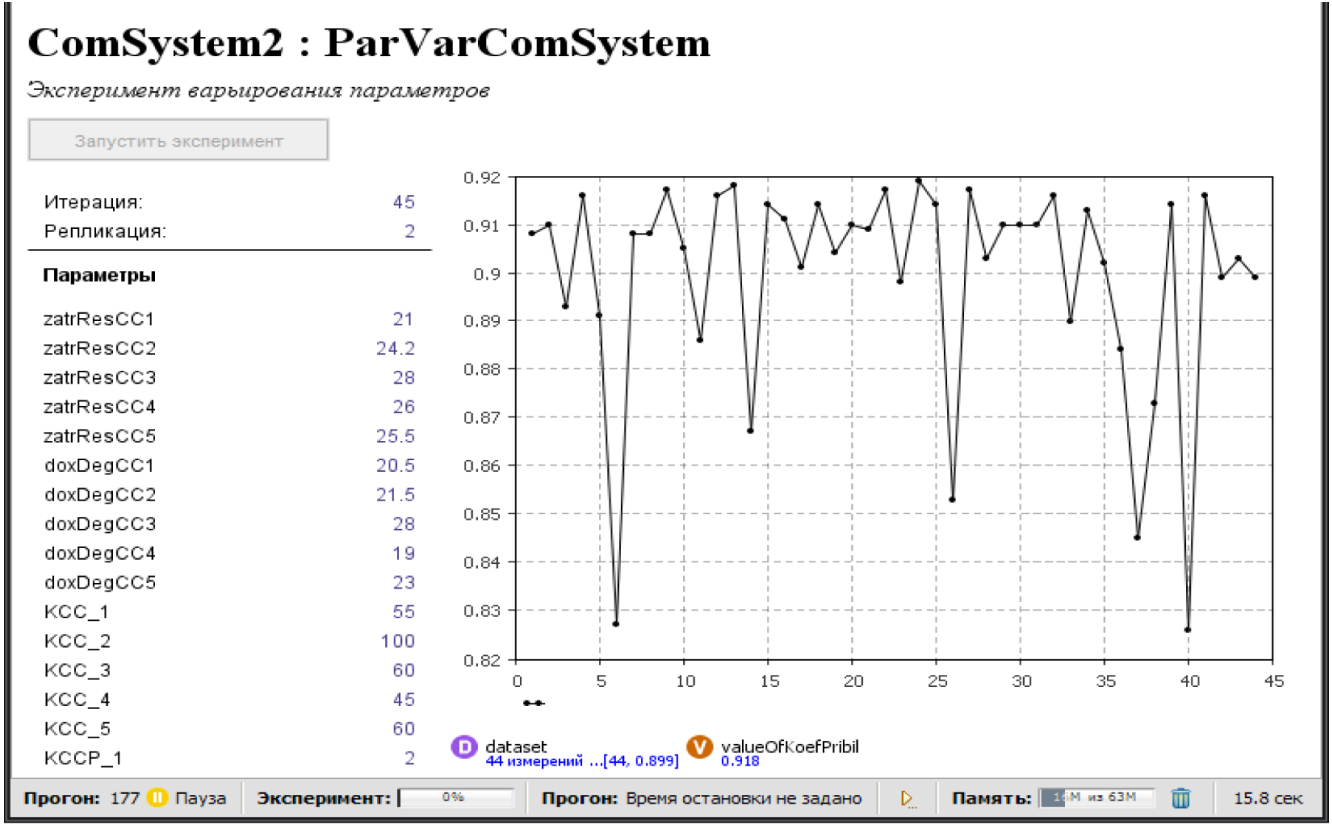


Рис. 3.93. Фрагмент результатов эксперимента **Варьирование параметров**

21. В меню запуск выполните **ComSystem2/ VarParComSystem.**

22. Щелкните **Запустить эксперимент**. Начнет выполняться эксперимент. Во время эксперимента можно видеть на графике изменение значения коэффициента прибыли. Фрагмент результатов выполнения эксперимента **Варьирование параметров** приведен на рис. 3.93.

23. Эксперимент был приостановлен после 177 прогона, то есть на 177:4 → 45-й итерации. Коэффициент прибыли составляет 0,918 при doxDegCC1 = 20,5, doxDegCC2 = 21,5, doxDegCC3 = 28, doxDegCC4 = 19, doxDegCC5 = 23.

В обоих экспериментах мы использовали опцию **Случайное начальное число (уникальные прогоны).** Поэтому при каждом новом запуске модели генератор случайных чисел инициализируется другим числом и результаты оптимизации могут отличаться.

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

Создайте оптимизационный эксперимент OptComSystem1, который будет отличаться от OptComSystem лишь выбранной опцией **Фиксированное начальное число (воспроизводимые прогоны)**. В поле **Начальное число** укажите `1133`.

Запустите эксперимент. Вы получите следующие результаты. Наилучшее значение целевой функции — коэффициент прибыли — равно 0,949. Получено оно на 120 итерации при следующих оптимальных значениях параметров: КССР_1=2, КССР_2=КССР_3= КССР_4=КССР_5=1, Kol_mast=3.

3.3.9. Экспорт модели как Java апплета

Модели AnyLogic являются приложением Java, поэтому их можно запускать на большинстве современнных платформ, а также помещать на веб-сайты в виде апплетов.

Наличие такой возможности позволяет удалённым пользователям запускать интерактивные модели в веб-браузере при отсутствии системы моделирования AnyLogic или какого-либо другого программного обеспечения. В этом случае на клиентской машине будут запускаться скопированные из сети файлы модели с такой

же поддержкой интерактивной работы, что и при запуске AnyLogic.

Экспортируйте модель ComSystem2 в виде Java апплета.

1. Щелкните в панели **Проекты** модель ComSystem2 и выберите **Экспорт…** из контекстного меню.
2. Откроется диалоговое окно **Экспорт модели**. Щелчком мыши раскройте список **Экспортировать эксперимент** и выберите в нем **Simulation**. Настройки этого эксперимента будут применены к экспортируемой модели.
3. В поле **Каталог для создаваемых файлов** укажите каталог, в который вы хотите поместить файлы экспортируемой модели. Можно также выбрать каталог с помощью диалогового окна, которое становится доступным при нажатии кнопки **Выбрать**.
4. Из группы кнопок **Экспортировать как** выберите опцию **Java апплет (запускается в веб-браузере)**.
5. Оставьте установленным флажок **Открыть апплет модели в веб-браузере**.
6. Щелкните кнопку **Готово**. Откроется диалоговое окно, в котором будет сообщение об успешном завершении экспортирования модели ComSystem2.

Модель, экспортированная как Java апплет, представляет собой набор следующих файлов (рис. 3.94):

файл .html, используемый для запуска Java апплета;
файл (com.xj.anylogic.engine.jar) исполняющего модуля AnyLogic;
скомпилированный .jar файл модели (model.jar);
.jar файлы и классы, необходимые для построения модели.

При публикации апплета модели в сети Интернет нужно предоставить доступ ко всем этим файлам из кода апплета. Это значит, что если вы добавляете ссылку на .html файл модели на веб страницу, то необходимо разместить все эти файлы в той же директории, где и этот .html файл. Для показа апплета на своей веб странице следует скопировать код апплета из .html файла модели в код своей страницы, и добавить все файлы, сгенерированные при экспорте модели, в тот же каталог, в котором находится ваша страница.

Запустите апплет модели, дважды щелкнув ComSystem2.html. Фрагмент работы апплета показан на рис. 3.95.

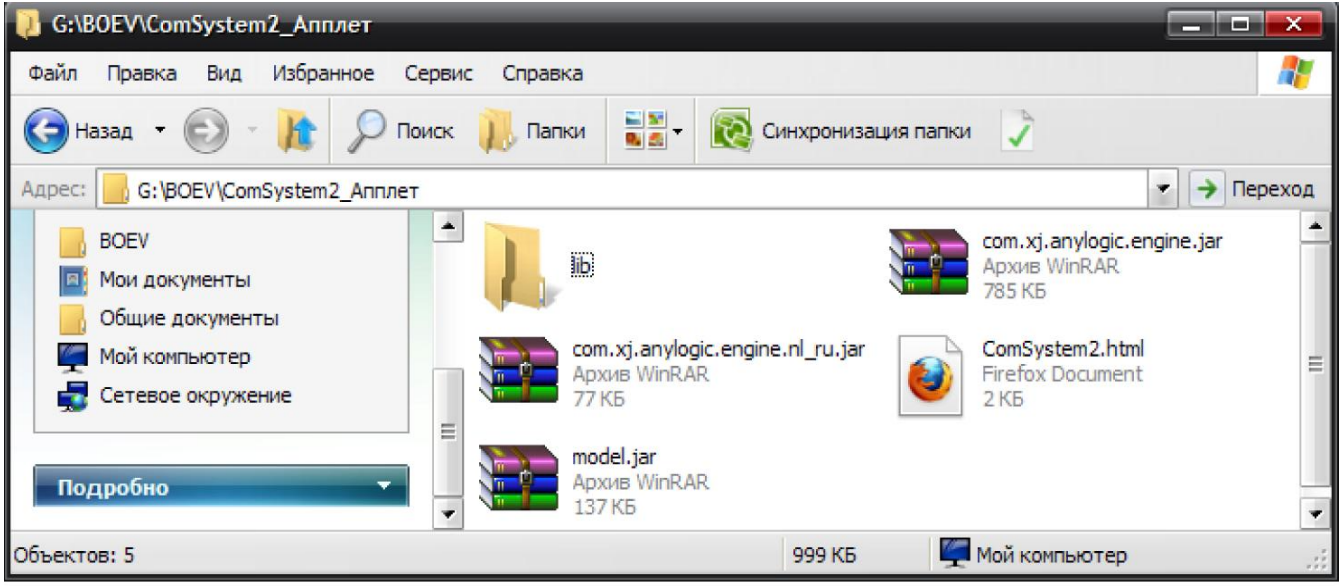


Рис. 3.94. Файлы, сгенерированные при экспорте модели

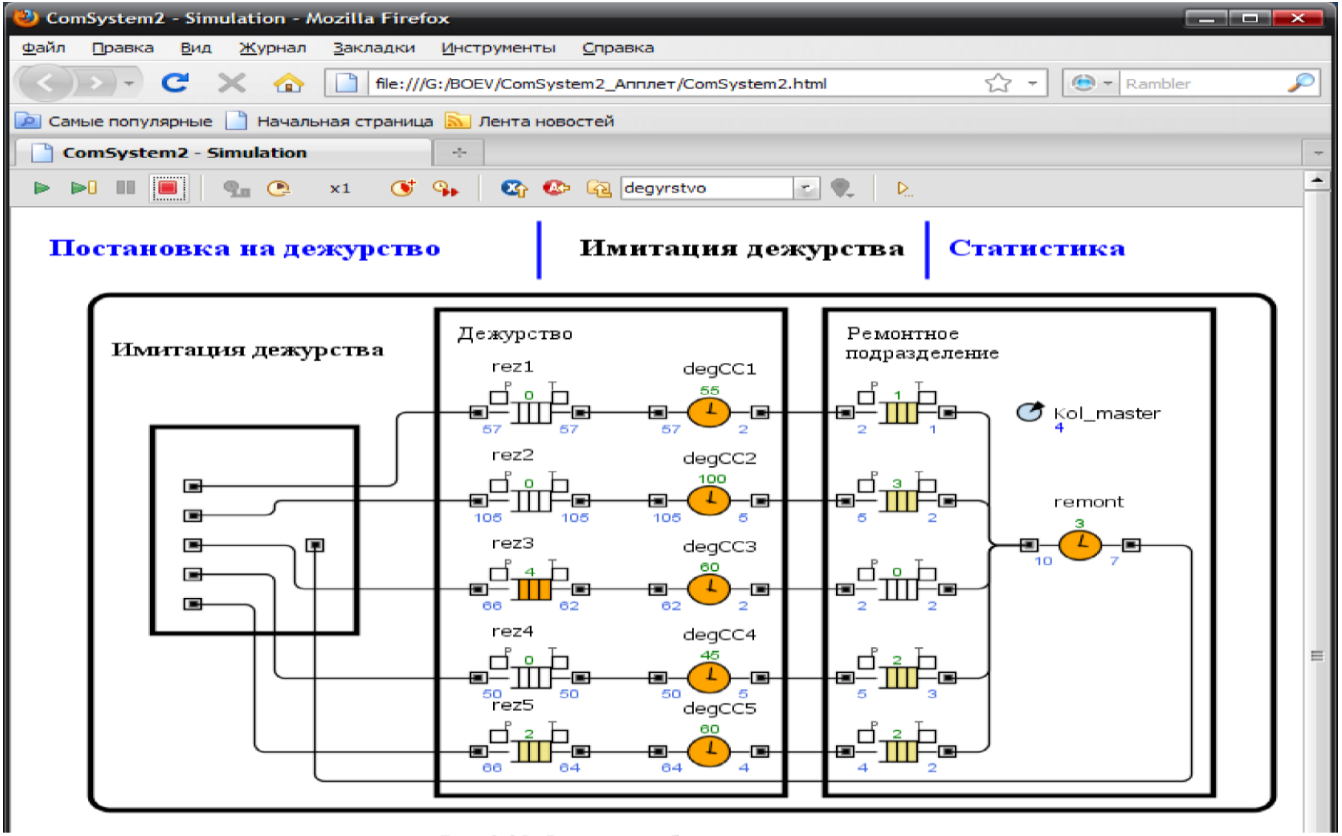


Рис. 3.95. Фрагмент работы апплета модели

3.4. Модель функционирования предприятия

3.4.1. Постановка задачи

Предприятие имеет n_1 цехов, производящих n_1 типов блоков, т. е. каждый цех производит блоки одного типа. Себестоимости комплектующих блоков $C_{к1}, C_{к2}, ..., C_{кn1}$. Стоимости изготовления блоков $C_{изг1}, C_{изг2}, ..., C_{изгn1}$. Интервалы выпуска блоков $T_1, T_2, ..., T_{n_1}$ — случайные. Из n_1 типов блоков собирается одно изделие.

Перед сборкой каждый тип блоков проверяется на $n_{11}, n_{12}, ..., n_{1n}$ соответствующих постах контроля. Длительности контроля одного блока $T_{11}, T_{12}, ..., T_{1n}$ случайные. Стоимости проверки блоков $C_{пр1}, C_{пр2}, ..., C_{прn1}$. На каждом посту бракуется q_{11}, $q_{12}, ..., q_{1n}$ % блоков соответственно. Забракованные блоки в дальнейшем процессе сборки не участвуют и удаляются с постов контроля в брак.

Прошедшие контроль, т. е. не забракованные блоки поступают на один из n_2 пунктов сборки. На пункте сборки одновременно собирается только одно изделие. Сборка начинается только тогда, когда имеются все необходимые n_1 блоков различных типов. Время сборки $T_с$ случайное. Стоимость сборки одного изделия $C_{сб}$.

После сборки изделие поступает на один из n_3 стендов выходного контроля. На одном стенде одновременно проверяется только одно изделие. Время проверки $T_п$ случайное. Стоимость проверки одного изделия $C_к$. По результатам проверки бракуется q_2 % изделий. В таком изделии с вероятностями $P_{к1} ... P_{кm}$ могут быть забракованы *m* блоков с порядковыми номерами из $1 ... n_1$.

Забракованное изделие направляется в цех сборки, где неработоспособные блоки заменяются новыми. Время замены $T_{замi}$ случайное. Стоимость замены *i*-го блока $C_{замi}$. После замены блоков изделие вновь поступает на один из стендов выходного контроля.

Прошедшее стенд выходного контроля изделие поступают в отдел приемки. Время приемки $T_{пр}$ одного изделия случайное. Стоимость приемки одного изделия $C_п$. По результатам приемки бракуется q_3 % изделий, которые направляются вновь на стенд вы-

ходного контроля. Принятые приемкой изделия направляются на склад предприятия.

3.4.2. Исходные данные

$n_1 = 4; T_1 = 19; T_2 = 11; T_3 = 15; T_4 = 18;$

$C_{к1} = 35; C_{к2} = 32; C_{к3} = 43; C_{к4} = 48;$

$C_{изг1} = 35; C_{изг2} = 27; C_{изг3} = 36; C_{изг4} = 37;$

$n_{11} = 2; n_{12} = 2; n_{13} = 2; n_{14} = 2; T_{11} = 12; T_{12} = 16; T_{13} = 21; T_{14} = 17;$

$C_{пр1} = 12; C_{пр2} = 23; C_{пр3} = 32; C_{пр4} = 28;$

$q_{11} = 0,02; q_{12} = 0,03; q_{13} = 0,04; q_{14} = 0,06;$

$n_2 = 2; T_с = 22; C_{сб} = 67; n_3 = 2; T_п = 26; C_к = 74; q_2 = 0,05;$

$m = 1; P_{к1} = 1,0; P_{бл1} = 0,25; P_{бл2} = 0,25; P_{бл3} = 0,25; P_{бл4} = 0,25;$

$T_{зам1} = 12; T_{зам2} = 15; T_{зам3} = 12; T_{зам4} = 21;$

$C_{зам1} = 34; C_{зам2} = 46; C_{зам3} = 38; C_{зам4} = 54;$

$n_4 = 2; T_{пр} = 18; C_п = 53; \ q_4 = 0,15.$

Интервалы времени между выпусками блоков, время контроля блоков и изделий, сборки и приема изделий подчинены экспоненциальному закону.

3.4.3. Задание на исследование

Разработать имитационную модель функционирования предприятия при изготовлении изделий из блоков.

Исследовать влияние качества изготовления блоков и других параметров (интервалов выпуска блоков из цехов, себестоимости комплектующих, стоимости изготовления блоков, проверки, сборки и др.) на себестоимость изделий.

Сделать выводы о загруженности подразделений предприятия и необходимых мерах по снижению себестоимости продукции.

3.4.4. Формализованное описание модели

Представим структуру предприятия по изготовлению изделий в виде схемы, показанной на рис. 3.96.

Видно, что предприятие представляет собой многоканальную многофазную систему массового обслуживания разомкнутого типа, хотя на отдельных ее фазах имеются обратные связи.

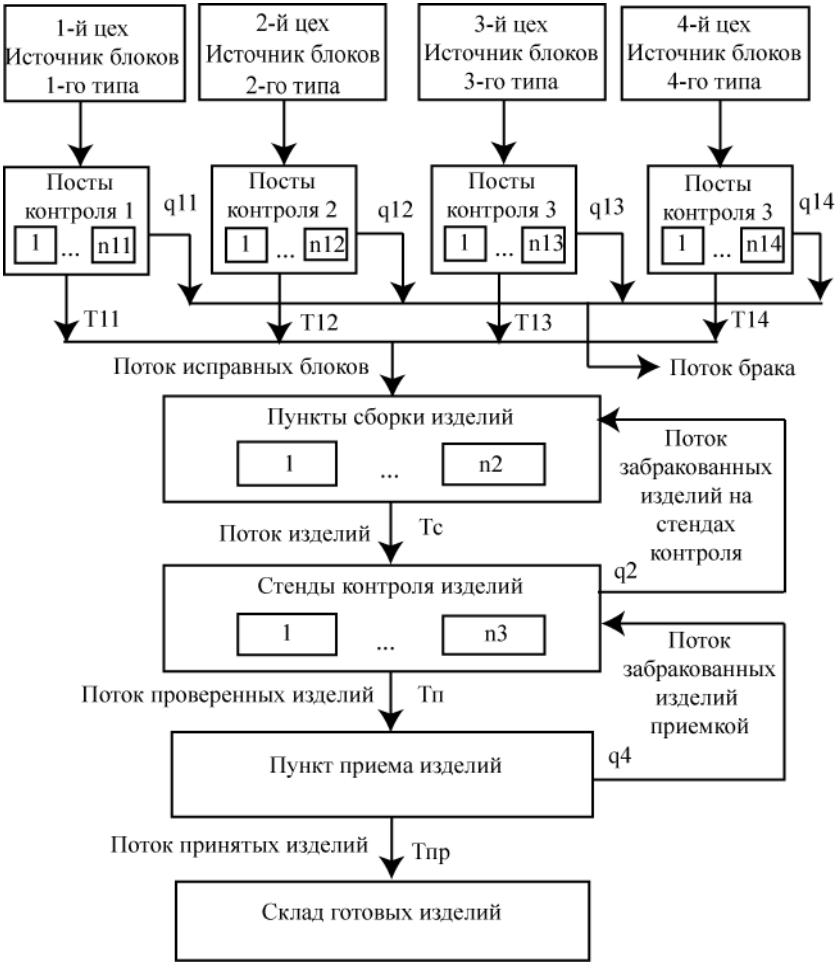


Рис. 3.96. Предприятие как система массового обслуживания

Модель, исходя из структуры предприятия, должна состоять из следующих сегментов:

ввода исходных данных;

имитации работы цехов по изготовлению блоков;

имитации работы постов контроля блоков;

имитации работы пунктов сборки изделий;

имитации работы стендов контроля собранных изделий;

имитации работы пунктов приема изделий;

имитации склада готовых изделий;

имитации склада бракованных блоков;

вывода результатов моделирования.

Блоки и изделия в модели следует имитировать заявками со следующими полями (параметрами):

`block` — номер цеха (коды 1…4 соответственно), выпустившего блок;

`sign1` — признак брака на постах контроля блоков и стендах контроля изделий;

`sign2` — признак брака на пунктах приема изделий;

`numBlBrak1` … `numBlBrak4` — признаки забракованных блоков 1…4 соответственно;

`timeSbor` — время сборки изделия (замены блоков).

Указанные поля предназначены для отслеживания технического состояния блоков и изделий и в зависимости от этого прохождения их по фазам изготовления изделия.

Поле `timSbor` введено для удобства построения модели. В это поле заносится либо время сборки изделия из четырех блоков, либо время замены каких-либо забракованных блоков в уже собранном изделии.

Для обеспечения работы модели вводятся следующие параметры процесса изготовления изделий из блоков:

`aveTimeShop1`…`aveTimeShop4` — средние интевалы времени выпуска блоков 1…4 цехами 1…4;

`stKomplBlock1`…`stKomplBlock4` — стоимости комплектующих блоков 1…4;

`stIzgBlock1`…`stIzgBlock4` — стоимость изготовления блоков 1…4;

`postKontr1`…`postKontr4` — количество постов контроля блоков 1…4;

`procBrakBlock1`…`procBrakBlock4` — проценты брака блоков 1…4 на постах контроля блоков;

`stTestBlock1`…`stTestBlock4` — стоимости тестирования (одного блока) блоков 1…4;

`kolPunSborki` — количество пунктов сборки изделий;

`timeSborki` — среднее время сборки одного изделия;

`stSborki` — стоимость сборки одного изделия;

`kolStendKontr` — количество стендов контроля собранных изделий;

timeKontrIzd — среднее время контроля на стенде одного собранного изделия;

procBrakIzd — процент брака собранных изделий на стендах контроля;

stKontrIzd — стоимость контроля на стенде одного собранного изделия;

verBlock1 — вероятность того, что забракованным в уже собранном изделии окажется один блок;

verBlock2 — вероятность того, что забракованными в уже собранном изделии окажутся два блока;

verBlNum1…verBlNum4 — вероятности брака на стенде контроля блоков 1…4;

timeZamBlock1… timeZamBlock4 — среднее время замены (одного блока) блоков 1…4;

stZamBlock1…stZamBlock4 — стоимость замены (одного блока) блоков 1…4;

kolPunPriem — количество пунктов приема изделий, прошедших пункты контроля;

timePriemIzd — среднее время приема одного изделия;

procBrakPriem — процент брака изделий на пунктах приема;

stPriemIzd — стоимость приема одного изделия.

В ходе моделирования на основе приведенных исходных данных формируется и выводится на текущее модельное время следующая информация:

kolIzBlock1…kolIzBlock4 — количество изготовленных блоков 1…4;

kolTestBlock1…kolTestBlock4 — количество протестированных блоков 1…4 (как исправных, так и забракованных);

brakBlock1…brakBlock4 — количество забракованных на постах контроля блоков 1…4;

gotBlock1…gotBlock4 — количество изготовленных всего готовых блоков 1…4;

ostGotBlock1…ostGotBlock4 — количество оставшихся готовых блоков 1…4;

kolSobrIzd, testSobrIzd, brakSobrIzd — количество собранных на пунктах сборки изделий, проверенных и забракованных на стендах контроля;

`kolPriemIzd, brakPriemIzd` — количество принятых и забракованных приемкой изделий соответственно;

`zamBlock1…zamBlock4` — количество замененных блоков 1…4, забракованных на стендах контроля и пунктах приема изделий;

`allBrakBlock1…allBrakBlock4` — всего забракованных блоков 1…4;

`kolGotIzd` — количество готовых изделий, отправленных на склад.

На текущее модельное время собирается статистика по следующим стоимостным показателям:

`costTestBlock1…costTestBlock4, costTestBlock` — стоимости тестирования блоков 1…4 на постах контроля и суммарная стоимость тестирования всех блоков;

`costIzgBlock1…costIzgBlock4, costIzgBlock` — стоимости изготовления блоков 1…4 и суммарная стоимость изготовления всех блоков (без учета стоимости тестирования блоков);

`sumCostBlock1…sumCostBlock4, sumCostBlock` — суммарные стоимости изготовления блоков 1…4 с учетом тестирования и суммарная стоимость изготовления всех блоков;

`costSborIzd, costTestIzd, costPriemIzd` — стоимости сборки, проверки и приемки изделий;

`costBrakBlock` — суммарные затраты на все забракованные блоки;

`costGotIzd` — затраты на выпуск готовых изделий;

`costIzd` — стоимость одного изделия;

`koefIncrCostIzd` — коэффициент увеличения себестоимости изделия.

Минимальная себестоимость изделий будет тогда, когда не будет бракованных блоков и изделий. В терминах постановки задач это условие имеет вид:

$$C_{\min} = C_{\text{изд}} \cdot N_{\text{изд}},$$

где $N_{\text{изд}}$ — количество готовых изделий (поступивших на склад);

$C_{\text{изд}}$ — себестоимость производства одного изделия, вычисяется по формуле:

$$C_{\text{изд}} = \sum_{i=1}^{n1} \left(C_{\text{к}i} + C_{\text{изг}i} + C_{\text{пр}i} \right) + C_{\text{сб}} + C_{\text{к}} + C_{\text{п}}.$$

В случае наличия брака себестоимость производимой продукции увеличится и составит $C_{\max}$. То есть коэффициент увеличения себестоимости будет равен

$$K_c = C_{\max} / C_{\min}.$$

Запишем в идентификаторах исходных данных и результатов моделирования то же самое:

$$C_{\min} = [(\text{stKomplBlock1}+\text{stKomplBlock2}+\text{stKomplBlock3}+$$
$$+\text{stKomplBlock}) + \text{stSborki} + \text{stKontrIzd} + \text{stPriemIzd}]\cdot \text{kolGotIzd};$$

$$C_{\max} = \text{costKomplBlock} + \text{costIzgBlock} + \text{costTestBlock} +$$
$$+ \text{costSborki} + \text{costTestIzd} + \text{costPriemkiIzd} + \text{costBrakBlock};$$

$$K_c = \text{koefIncrCostIzd} = \frac{C_{\max}}{C_{\min}}.$$

Замечание. В приведеных для расчета результатов моделирования выражениях мы не учитывали те блоки, которые были произведены, но не были использованы для сборки изделий.

3.4.5. Ввод исходных данных

Для ввода исходных данных используем элементы **Параметр** и **Бегунок** (см. п. 3.1.7).

1. Выполните команду **Файл/Создать/Модель** на панели инструментов.
2. В поле **Имя модели** диалогового окна **Новая модель** (см. рис. 3.1) введите `Enterprise`. Выберите каталог, в котором будут сохранены файлы модели. Щелкните кнопку **Далее**.
3. На открывшейся второй странице **Мастера создания модели** (см. рис. 3.2) выберите **Начать создание модели «с нуля»**. Щелкните кнопку **Далее**.
4. В **Палитре** выделите **Презентация**. Создайте область просмотра `Data` для размещения элементов исходных данных.
5. Перетащите элемент **Прямоугольник** в нужное место.
6. Перетащите элемент **text** и на странице **Основные** панели **Свойства** в поле **Текст:** введите `Исходные данные`.
7. В **Палитре** выделите **Основная**. Перетащите элементы **Параметр** на элемент с именем `Исходные данные`. Разместите их так, как показано на рис. 3.97. Значения свойств установите согласно табл. 3.8.

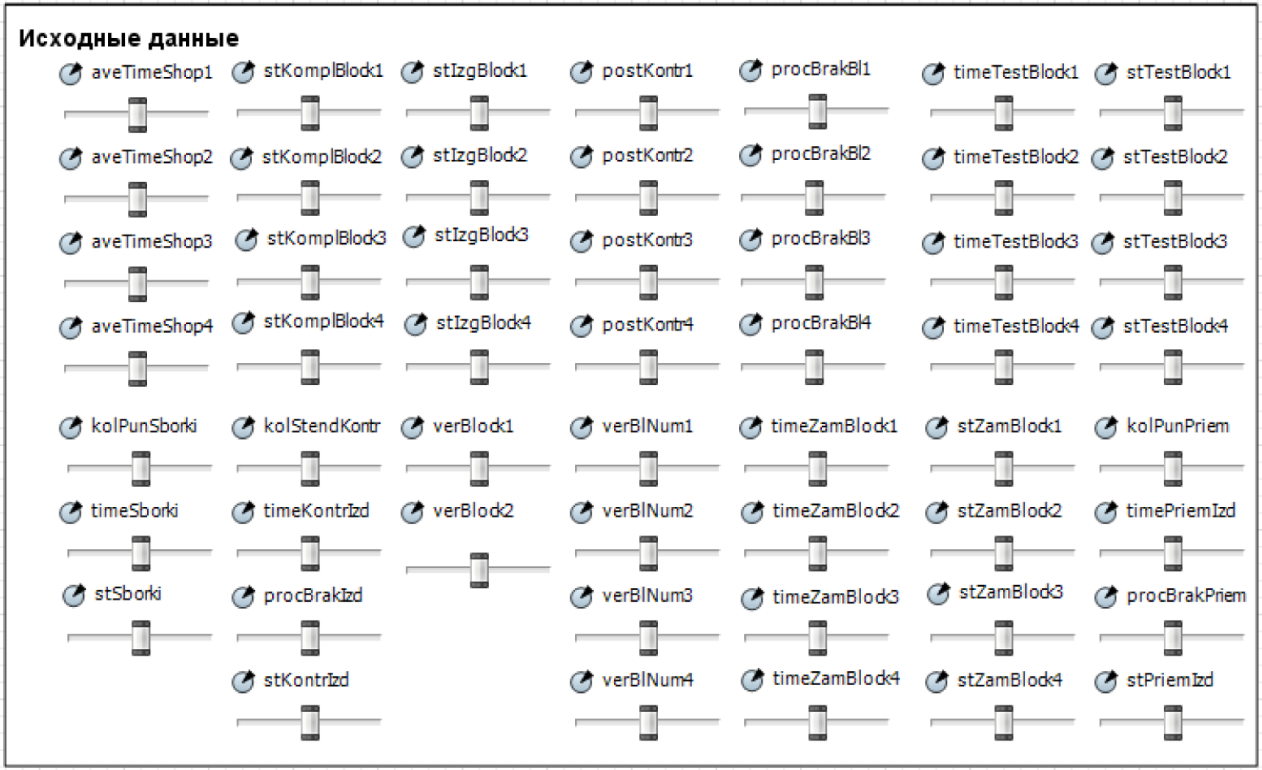


Рис. 3.97. Размещение элементов **Параметр** и **Бегунок** для ввода исходных данных

Таблица 3.8

Элементы и их свойства					
Параметр			Бегунок		
Имя	Тип	Значение по умолча-нию	Связать с	Минимальное значение	Максимальное значение
aveTimeShop1	double	19	aveTimeShop1	10	100
aveTimeShop2	double	11	aveTimeShop2	10	100
aveTimeShop3	double	15	aveTimeShop3	10	100
aveTimeShop4	double	18	aveTimeShop4	10	100
stKomplBlock1	double	35	stKomplBlock1	20	100
stKomplBlock2	double	32	stKomplBlock2	20	100
stKomplBlock3	double	43	stKomplBlock3	20	100
stKomplBlock4	double	48	stKomplBlock4	20	100
stIzgBlock1	double	35	stIzgBlock1	20	100
stIzgBlock2	double	27	stIzgBlock2	20	100
stIzgBlock3	double	36	stIzgBlock3	20	100
stIzgBlock4	double	37	stIzgBlock4	20	100
postKontr1	int	2	postKontr1	1	20
postKontr2	int	2	postKontr2	1	20
postKontr3	int	2	postKontr3	1	20
postKontr4	int	2	postKontr4	1	20
procBrakBlock1	double	0.02	procBrakBlock1	0	0.35
procBrakBlock2	double	0.03	procBrakBlock2	0	0.35
procBrakBlock3	double	0.04	procBrakBlock3	0	0.35
procBrakBlock4	double	0.06	procBrakBlock4	0	o.35
timeTestBlock1	double	12	timeTestBlock1	10	100
timeTestBlock2	double	16	timeTestBlock2	10	100
timeTestBlock3	double	21	timeTestBlock3	10	100
timeTestBlock4	double	17	timeTestBlock4	10	100
stTestBlock1	double	12	stTestBlock1	10	100
stTestBlock2	double	23	stTestBlock2	10	100
stTestBlock3	double	32	stTestBlock3	10	100
stTestBlock4	double	28	stTestBlock4	10	100
kolPunSborki	int	2	kolPunSborki	1	20
timeSborki	double	22	timeSborki	10	100
stSborki	double	67	stSborki	30	100
kolStendKontrIzd	Int	2	kolStendKontrIzd	1	20
timeKontrIzd	double	26	timeKontrIzd	10	100
procBrakIzd	double	0.05	procBrakIzd	0	0.35
stKontrIzd	double	74	stKontrIzd	40	120
verBlock1	double	0.99	verBlock1	0.8	1
verBlock2	double	0.9999	verBlock2	0.9	1

Окончание табл. 3.8

Элементы и их свойства					
Параметр			Бегунок		
Имя	Тип	Значение по умолчанию	Связать с	Минимальное значение	Максимальное значение
verBlockNum1	double	0.25	verBlockNum1	0	0.35
verBlockNum2	double	0.25	verBlockNum2	0	0.35
verBlockNum3	double	0.25	verBlockNum3	0	0.35
verBlockNum4	double	0.25	verBlockNum4	0	0.35
timeZamBlock1	double	12	timeZamBlock1	10	60
timeZamBlock2	double	15	timeZamBlock2	10	60
timeZamBlock3	double	12	timeZamBlock3	10	60
timeZamBlock4	double	21	timeZamBlock4	10	60
stZamBlock1	double	34	stZamBlock1	20	100
stZamBlock2	double	46	stZamBlock2	20	100
stZamBlock3	double	38	stZamBlock3	20	100
stZamBlock4	double	54	stZamBlock4	20	100
kolPunPriem	int	2	kolPunPriem	1	20
timePriemIzd	double	18	timePriemIzd	15	100
procBrakPriem	double	0.15	procBrakPriem	0	0.35
stPriemIzd	double	53	stPriemIzd	40	100

8. В **Палитре** выделите **Элементы управления**. Перетащите элементы **Бегунок** на элемент с именем `Исходные данные` и разместите их так, как показано на рис. 3.97. Значения свойств установите также согласно табл. 3.8.

3.4.6. Вывод результатов моделирования

Для вывода результатов моделирования используем элемент **Простая переменная**. Для удобства обозрения и анализа результатов моделирования разделим их на две группы. В первую группу включим данные о количестве подготовленных и забракованных блоков и изделий, во вторую группу — стоимостные показатели функционирования предприятия.

1. В **Палитре** выделите **Презентация**. Создайте область просмотра `Result` для размещения элементов **Простая переменная**.

2. Перетащите элемент **text** и на странице **Основные** панели **Свойства** в поле **Текст:** введите `Данные о количестве подготовленных и забракованных блоков и изделий`.

3. Перетащите второй элемент **text** и на странице **Основные** панели **Свойства** в поле **Текст:** введите `Стоимостные показатели функционирования предприятия`.

4. В **Палитре** выделите **Основная**. Перетащите элементы **Простая переменная**. Разместите их и дайте им имена так, как показано на рис. 3.98. Тип всех переменных, используемых для вывода количества подготовленных и забракованных блоков и изделий на текущее модельное время — `int`. Тип переменных для вывода стоимостных показателей работы предприятия — `double`. **Значение по умолчанию** равно 0.

3.4.7. Построение событийной части модели

В событийную часть модели включим указанные ранее сегменты согласно представлению предприятия как системы массового обслуживания (см. п. 3.44).

1. В **Палитре** выделите **Презентация**. Перетащите элементы **Сгруленный прямоугольник** и разместите так, как на рис. 3.99.

2. Перетащите элементы **text** и на странице **Основные** панели **Свойства** в поле **Текст:** каждого элемента введите названия элементов, показанные на рис. 3.99.

Модель будет содержать три активных объекта, элементы которых не могут физически вместиться в область диаграммы в окне презентации при выполнении модели, поэтому используем для каждого акивного объекта специальный элемент — *область просмотра*. Впоследствии организуем переключение между областями просмотра.

Создайте область просмотра для размещения элементов на диаграмме класса Main.

1. В **Палитре** выделите **Презентация**. Перетащите элемент **Область просмотра** в левый верхний угол (см. рис. 3.99). На диаграмме появится значок якоря этой области просмотра.

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

3. В поле **Имя:** введите `Mainview`.

4. Задайте, как будет располагаться область просмотра относительно ее якоря, с помощью элемента управления **Выравнивать по:** `Верхн. левому углу`.

5. Выберите режим масштабирования из выпадающего списка **Масштабирование:** `Подогнать под окно`.

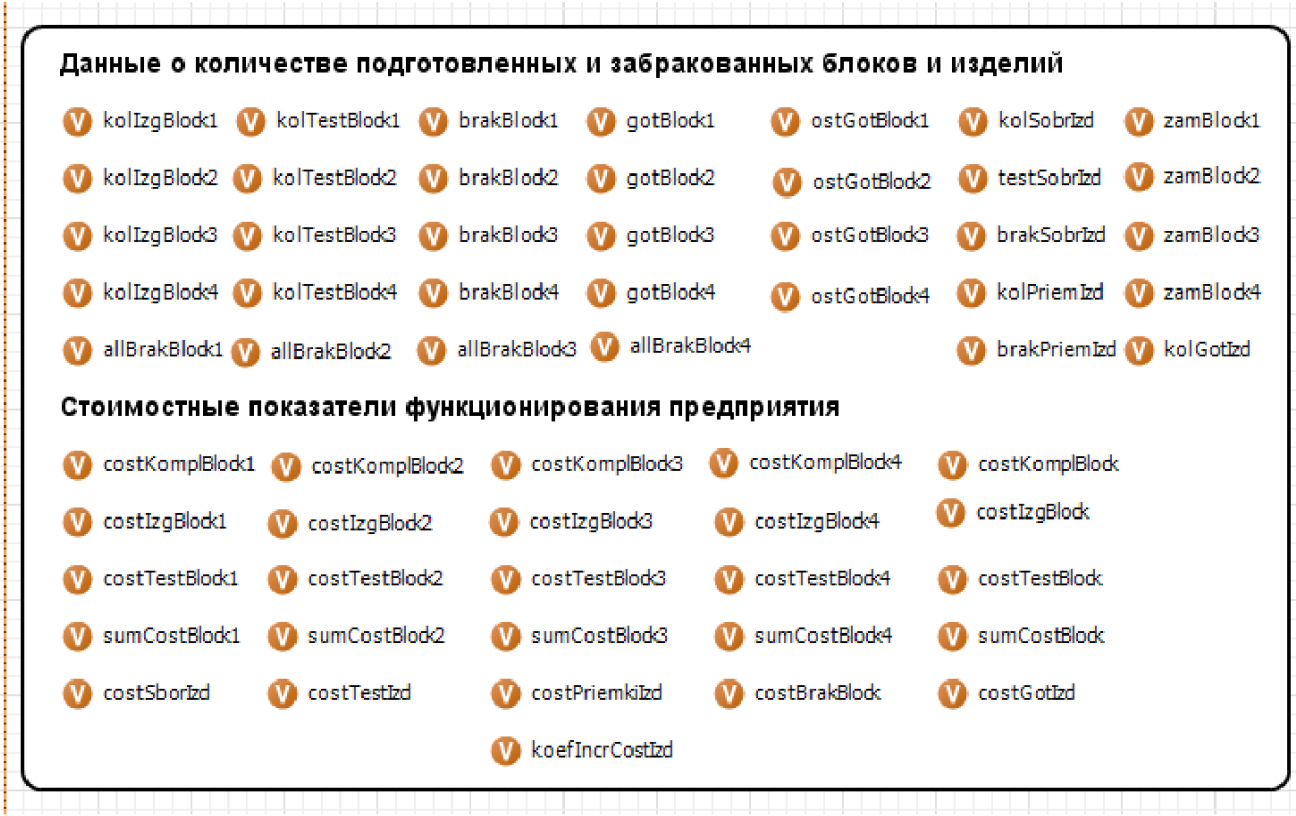


Рис. 3.98. Размещение элементов Простая переменная для вывода результатов моделирования

Рис. 3.99. Сегменты событийной части модели

3.4.7.1. Имитация работы цехов предприятия

Данный сегмент предназначен для имитации работы цехов, то есть изготовления и выпуска блоков через случайные интервалы времени, счета количества изготовленных блоков, стоимости комплектующих изготовленных блоков по типам и за предприятие, стоимости изготовления блоков по типам и суммарной стоимости за предприятие.

1. В **Палитре** выделите Enterprise Library.
2. Перетащите элемент **sourse** на диаграмму класса Main и разместите в скругленном прямоугольнике с именем `Цеха`.
3. Для записи и хранения параметров блоков и изделий в дополнительные поля заявок нужно создать нестандартный класс заявки. Создайте класс заявки `Product`.
4. В панели **Проект** щелкните правой кнопкой мыши элемент модели верхнего уровня дерева и выберите **Создать Java класс**.
5. Появится диалоговое окно **Новый Java класс** (см. рис. 3.32). В поле **Имя:** введите имя нового класса `Product`.
6. В поле **Базовый класс:** выберите из выпадающего списка `com.xj.anylogic.libraries.enterprise.Entity` в качестве базового класса. Щелкните кнопку **Далее**.
7. Появится вторая страница **Мастера создания Java класса**. Добавьте следующие поля Java класса, которые потребуются в дальнейшем при разработке модели:

```
int numBlock;
int sign1;
int numBlBrak1;
int numBlBrak2;
int numBlBrak3;
int numBlBrak4;
double timeSbor;
```

8. Оставьте выбранными флажки **Создать конструктор** и **Создать метод toString()**.
9. Щелкните кнопку **Готово**. Появится редактор кода и автоматически созданный код вашего Java класса. Закройте код.
10. Выделите элемент **sourse**. На странице **Основные** панели **Свойства** установите свойства согласно рис. 3.100.
11. Щелкните выделенный **sourse** правой кнопкой мыши и в контекстном меню выберите **Копировать**.

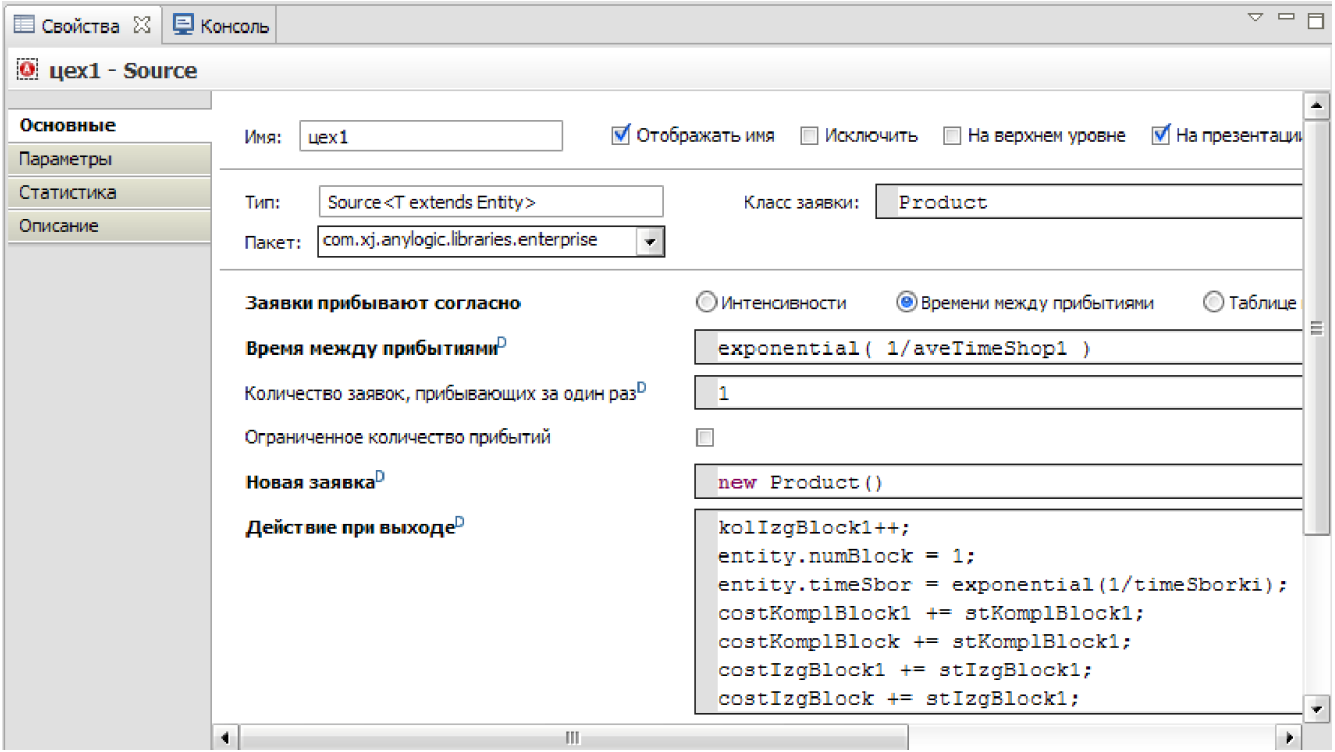


Рис. 3.100. Блок **sourse** с установленными свойствами

12. Вставьте в скругленный прямоугольник с именем **Цеха** еще три элемента **sourse**. Разместите их вертикально один под другим. Во время вставки имена элементов будут изменяться: `цех2, цех3, цех4`.

13. Однако остальные свойства останутся такими же, как и у элемента `цех1`. Поэтому, последовательно выделяя второй, третий и четвертый элементы **sourse**, скорректируйте их свойства согласно табл. 3.9.

3.4.7.2. Имитация работы постов контроля блоков

Каждый цех имеет посты контроля блоков одного типа. Посты контроля предназначены для приема блоков из цеха, тестирования их, отправки исправных блоков на пункты сборки изделий, а брака — на склад забракованных блоков.

Для размещения элементов, имитирующих работу постов контроля блоков, создайте новый класс активного объекта `Test`.

1. На панели **Проект** щелкните Main, с которым вы сейчас работаете, правой кнопкой мыши и выберите из контекстного меню **Создать/Класс активного объекта**.

Таблица 3.9

Свойства элементов sourse

Имя	Свойства	Значения
`цех2`	Отображать имя	Установите флажок
	Класс заявки	`Product`
	Заявки прибывают согласно	Времени между прибытиями
	Время между прибытиями	`exponential(1/aveTimeShop2)`
	Количество заявок, прибывающих за один раз	1
	Новая заявка	`New Product()`

	Действие при выходе	`kolIzgBlock2++;`
		`entity.numBlock = 2;`
		`costKomplBlock2 += stKomplBlock2;`
		`costKomplBlock += stKomplBlock2;`
		`costIzgBlock2 += stIzgBlock2;`
		`costIzgBlock += stIzgBlock2;`

Окончание табл. 3.9

Имя	Свойства	Значения

цех3	Отображать имя
	Класс заявки
	Заявки прибывают согласно
	Время между прибытиями
	Количество заявок, прибывающих за один раз
	Новая заявка
	Действие при выходе
		Установите флажок
		Product
		
		Времени между прибытиями
		
		Exponential(1/aveTimeShop3)
		
		1
		New Product()
		kolIzgBlock3++;
		entity.numBlock = 3;
		costKomplBlock3 +=
		stKomplBlock3;
		costKomplBlock +=
		stKomplBlock3;
		costIzgBlock3 += stIzgBlock3;
		costIzgBlock += stIzgBlock3;

цех4	Отображать имя
	Класс заявки
	Заявки прибывают согласно
	Время между прибытиями
	Количество заявок, прибывающих за один раз
	Новая заявка
	Действие при выходе
		Установите флажок
		Product
		
		Времени между прибытиями
		
		Exponential(1/aveTimeShop4)
		
		1
		New Product()
		kolIzgBlock4++;
		entity.numBlock = 4;
		costKomplBlock4 +=
		stKomplBlock4;
		costKomplBlock +=
		stKomplBlock4;
		costIzgBlock4 += stIzgBlock4;
		costIzgBlock += stIzgBlock4;

2. Откроется окно **Новый класс активного объекта**.
3. В поле **Имя:** задайте имя нового класса `Test`.
4. Если нужно, в поле **Описание:** введите описание сущности, моделируемой этим классом.

5. Щелкните кнопку **Готово**.

Создайте область просмотра на диаграмме класса Test для размещения элементов сегмента **Посты контроля блоков**.

1. В **Палитре** выделите **Презентация**. Перетащите элемент **Область просмотра** в нужное место.
2. Перейдите на страницу **Основные** панели **Свойства**.
3. В поле **Имя:** введите `Kontrol`.
4. Задайте, как будет располагаться область просмотра относительно ее якоря, с помощью элемента управления **Выравнивать по:** `Верхн. левому углу`.
5. Выберите режим масштабирования из выпадающего списка **Масштабирование:** `Подогнать под окно`.

Согласно указанному ранее назначению постов контроля блоков нужно сделать так, чтобы четыре типа блоков передавались из цехов на свои посты контроля, четыре типа тестированных и исправных блоков поступали на пункты сборки изделий, а четыре типа забракованных блоков — на склад забракованных блоков.

Создайте элемент нового класса активного объекта Test.

1. Из Палитры **Основная** перетащите элемент **Порт** и разместите сверху в крайнем левом ряду (рис. 3.101).

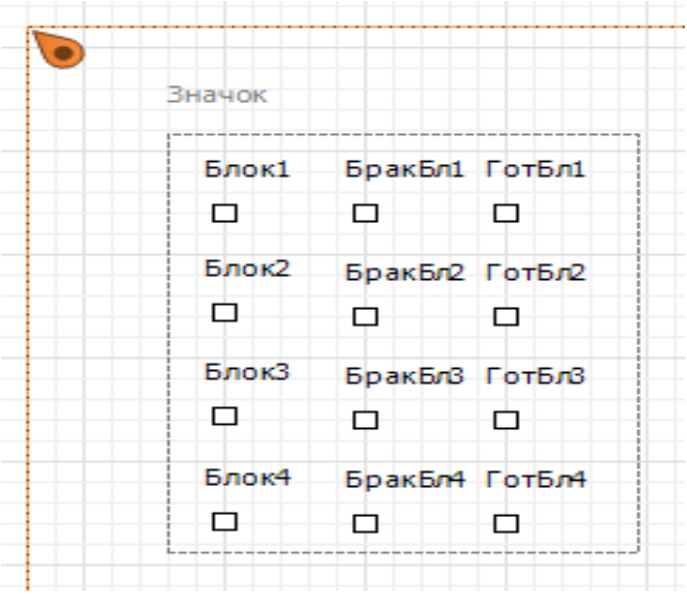


Рис. 3.101. Размещение элементов **Порт** на объекте Test

2. На странице **Основные** панели **Свойства** имя port замените именем `Блок1`.

3. Скопируйте элемент **Порт** с именем `Блок1`.

4. Вставьте три элемента **Порт** (см. рис. 3.101). При вставке последовательно будут изменяться их имена: `Блок2`, `Блок3`, `Блок4`.

5. Из Палитры **Основная** перетащите элемент **Порт** и разместите сверху в среднем ряду (см. рис. 3.101).

6. На странице **Основные** панели **Свойства** имя port замените именем `БракБл1`.

7. Скопируйте элемент **Порт** с именем `БракБл1`.

8. Вставьте три элемента **Порт** (см. рис. 3.101). При вставке последовательно будут изменяться их имена: `БракБл2`, `БракБл3`, `БракБл4`.

9. Из Палитры **Основная** перетащите элемент **Порт** и разместите сверху в крайнем правом ряду (см. рис. 3.101).

10. На странице **Основные** панели **Свойства** имя port замените именем `ГотБл1`.

11. Скопируйте элемент **Порт** с именем `ГотБл1`.

12. Вставьте три элемента **Порт** (см. рис. 3.101). При вставке последовательно будут изменяться их имена: `ГотБл2`, `ГотБл3`, `ГотБл4`.

13. По мере размещения элементов **Порт** они автоматически будут объединяться прямоугольником (с пунктирными линиями) и появится надпись `Значок`.

14. Возвратитесь на диаграмму класса Main.

15. На панели **Проект** выделите **Test**, перетащите на диаграмму класса Main элемент класса Test, разместите и соедините так, как на рис. 3.102. Порты БракБл1…БракБл4 соедините с элементом sink сегмента **Склад бракованных блоков**, предварительно разместив его там.

16. Элемент диаграммы активного класса Test создан. Возвратитесь на диаграмму класса Test.

Для имитации работы постов контроля блоков одного типа (одного цеха) нам потребуются элементы:

queue — имитация склада изготовленных цехом блоков;

delay — имитация времени тестирования блока;

selectOutPut — имитация процесса браковки блоков.

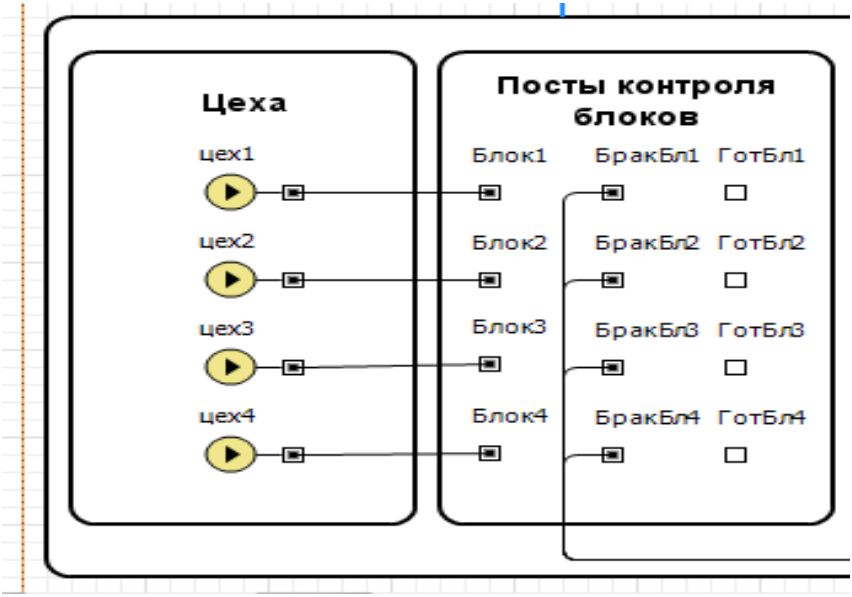


Рис. 3.102. Добавлен элемент диаграммы класса Test

Добавьте на диаграмму класса Test элементы класса Queue.

1. Из библиотеки Enterprise Library перетащите элемент queue и разместите слева сверху, как на рис. 3.103.
2. На странице **Основные** панели **Свойства** замените имя queue именем `склИзгБл1` (склад изготовленных блоков цеха 1).
3. В поле **Класс заявки:** `Entity` замените `Product`.
4. Установите **Вместимость:** `Максимальная`.
5. Скопируйте элемент с именем `склИзгБл1`.
6. Вставьте и разместите три элемента класса Queue (см. рис. 3.103).

Добавьте на диаграмму класса Test элементы класса Delay.

1. Из библиотеки Enterprise Library перетащите элемент delay и разместите справа рядом с элементом `склИзгБл1`, как на рис. 3.103.
2. На странице **Основные** панели **Свойства** замените имя delay именем `постКонтрБлок1` (посты контроля блоков цеха 1).
3. В поле **Класс заявки:** `Entity` замените `Product`.
4. Введите в поле **Время задержки:** `exponential (1/get_Main().timeTestBlock1)`.

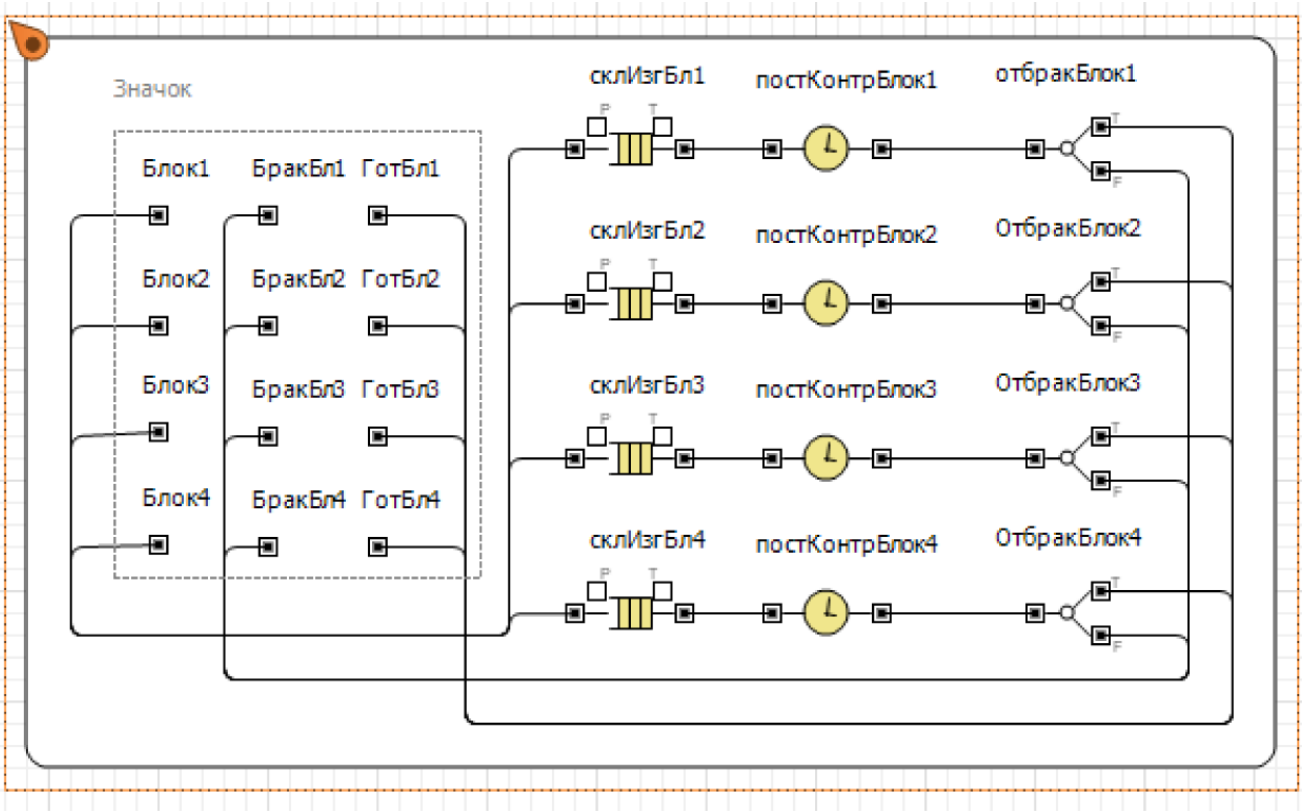


Рис. 3.103. Размещение элементов на диаграмме активного класса Test

5. Введите `get_Main().postKontr1` в поле **Вместимость:**.

6. **Действие при выходе**

```
get_Main().kolTestBlock1++;
get_Main().sumCostBlock1=get_Main().costKomplBlock1+
get_Main().costIzgBlock1+get_Main().costTestBlock1;
get_Main().sumCostBlock=get_Main().sumCostBlock1+
get_Main().sumCostBlock2+get_Main().sumCostBlock3+
get_Main().sumCostBlock4;
```

7. Скопируйте элемент с именем `постКонтрБлок1`.

8. Вставьте и разместите три элемента delay (см. рис. 3.103).

9. Последовательно выделите и внесите правки в их свойства (табл. 3.10). Внесите правки и в свойство **Действие при выходе**.

Таблица 3.10

Имя	Время задержки	Вместимость
постКонтрБлок2	`exponential (1/get_Main().timeTestBlock2)`	`get_Main(). postKontr2`
постКонтрБлок3	`exponential (1/get_Main().timeTestBlock3)`	`get_Main(). postKontr3`
постКонтрБлок4	`exponential (1/get_Main().timeTestBlock4)`	`get_Main(). postKontr4`

Добавьте на диаграмму Test элементы класса SelectOutPut.

1. Перетащите элемент selectOutPut и разместите справа рядом с элементом `постКонтрБлок1`, как на рис. 3.103.

2. На странице **Основные** панели **Свойства** замените имя selectOutPut именем `ОтбракБлок1` (отбраковка блоков цеха 1).

3. Установите свойства элемента согласно рис. 3.104.

4. Скопируйте элемент с именем `ОтбракБлок1`.

5. Вставьте три элемента класса SelectOutPut (рис. 3.103).

6. Последовательно выделите вставленные элементы и скорректируйте значения их свойств согласно табл. 3.11.

7. Соедините входы и выходы элементов диаграммы класса Test согласно рис. 3.103.

Код в свойство **Действие при выходе (true)** введен для учета, например, для цеха 1:

gotBlock1 — количества готовых блоков цеха 1;

costTestBlock1 — стоимости контроля блоков цеха 1;

costTestBlock — стоимости контроля блоков всех цехов.

Код в свойстве **Действие при выходе (false)** обеспечивает счет количества brakBlock1 забракованных блоков цеха 1.

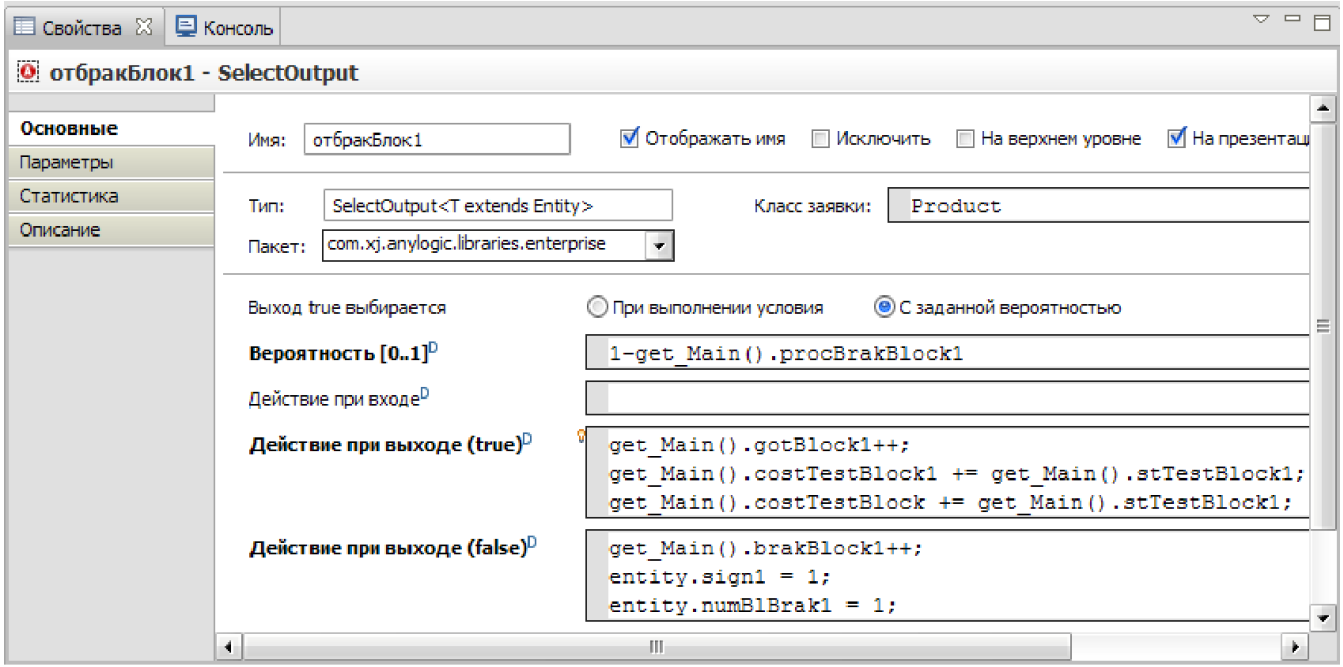


Рис. 3.104. Свойства элемента **selectOutPut** с именем отбракБлок1

Таблица 3.11

Свойства	Значение

Имя
Выход true выбирается
Вероятность [0…1]
Действие при выходе (true)
 	
Действие при выходе (false)
	ОтбракБлок2
	С заданной вероятностью
	1-get_Main().procBrakBlock2
	get_Main().gotBlock2++;
	get_Main().costTestBlock2 +=
	get_Main().stTestBlock2;
	get_Main().costTestBlock +=
	get_Main().stTestBlock2;
	get_Main().brakBlock2++;
	entity.sign1 = 1;
	entity.numBlBrak2 = 1;

Имя
Выход true выбирается
Вероятность [0…1]
Действие при выходе (true)
 	
Действие при выходе (false)
	ОтбракБлок3
	С заданной вероятностью
	1-get_Main().procBrakBlock3
	get_Main().gotBlock3++;
	get_Main().costTestBlock3 +=
	get_Main().stTestBlock3;
	get_Main().costTestBlock +=
	get_Main().stTestBlock3;
	get_Main().brakBlock3++;
	entity.sign1 = 1;
	entity.numBlBrak3 = 1;

Имя
Выход true выбирается
Вероятность [0…1]
Действие при выходе (true)
 	
Действие при выходе (false)
	ОтбракБлок4
	С заданной вероятностью
	1-get_Main().procBrakBlock4
	get_Main().gotBlock4++;
	get_Main().costTestBlock4 +=
	get_Main().stTestBlock4;
	get_Main().costTestBlock +=
	get_Main().stTestBlock4;
	get_Main().brakBlock4++;
	entity.sign1 = 1;
	entity.numBlBrak4 = 1;

В поле entity.sign1 записывается 1 — признак брака на постах контроля, а также единица записывается в поле entity.numBlock1. Это нужно для раздельного счета забракованных блоков.

Перейдите на диаграмму класса Main.

3.4.7.3. Имитация работы пунктов сборки изделий

Пункты сборки изделий предназначены для приема прошедших тестирование блоков с постов контроля, сборки изделий из блоков, замены забракованных блоков и отправки их на склад забракованных блоков.

Для размещения элементов имитации работы пунктов сборки изделий создайте новый класс активного объекта Sborka.

1. На панели **Проект** щелкните Main правой кнопкой мыши и выберите из контекстного меню **Создать/Класс активного объекта**.
2. Откроется окно **Новый класс активного объекта**.
3. В поле **Имя:** задайте имя нового класса `Sborka`.
4. Если нужно, в поле **Описание:** введите описание сущности, моделируемой этим классом.
5. Щелкните кнопку **Готово**.

Создайте область просмотра на диаграмме класса Sborka для размещения элементов сегмента **Пункты сборки изделий**.

1. В **Палитре** выделите **Презентация**. Перетащите элемент **Область просмотра** в нужное место.
2. Перейдите на страницу **Основные** панели **Свойства**.
3. В поле **Имя:** введите `Sbor`.
4. Задайте, как будет располагаться область просмотра относительно ее якоря, с помощью элемента управления **Выравнивать по:** `Верхн. левому углу`.
5. Выберите режим масштабирования из выпадающего списка **Масштабирование:** `Подогнать под окно`.

Согласно назначению пунктов сборки изделий для связи их с другими сегментами модели необходимо иметь: четыре порта для приема готовых блоков с постов контроля и один порт — для отправки собранных изделий на стенды контроля. Также нужен один порт — для приема бракованных изделий с целью определения и замены в них забракованных блоков и один порт — для отправки забракованных блоков после их замены на склад забракованных блоков.

1. Из Палитры **Основная** перетащите элемент **Порт** и разместите сверху в крайнем левом ряду (рис. 3.106).
2. На странице **Основные** панели **Свойства** имя port замените именем `ГотБлок1`.
3. Скопируйте элемент **Порт** с именем `ГотБлок1`.

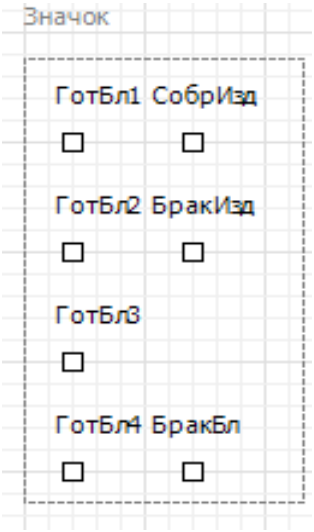


Рис. 3.106. Размещение элементов **Порт** на объекте Sborka

4. Вставьте три элемента **Порт** (см. рис. 3.106). При вставке последовательно будут изменяться их имена: `ГотБлок2`, `ГотБлок3`, `ГотБлок4`.

5. Из Палитры **Основная** перетащите еще три элемента **Порт**, разместите их и дайте имена как на рис. 3.106.

6. Возвратитесь на диаграмму класса Main.

7. На панели Проект выделите Sborka, перетащите на диаграмму класса Main элемент sborka, разместите и соедините так, как на рис. 3.107. Порт БракБл соедините с элементом sink сегмента **Склад бракованных блоков**.

8. Элемент диаграммы активного класса Sborka создан. Возвратитесь на диаграмму класса Sborka.

Для реализации сегмента имитации работы пунктов сборки изделий нам потребуются элементы, но прежде мы рассмотрим вариант построения этого сегмента.

Сборка изделия начинается тогда, когда будут готовы четыре блока (по одному каждого из четырех типов). Полагаем, что время готовности блоков различное. Значит, нужно воспользоваться такими элементами AnyLogic, которые предназначены для синхронизации движения заявок (в данном случае блоков). Элементы класса Match предназначены именно для синхронизации движения двух заявок.

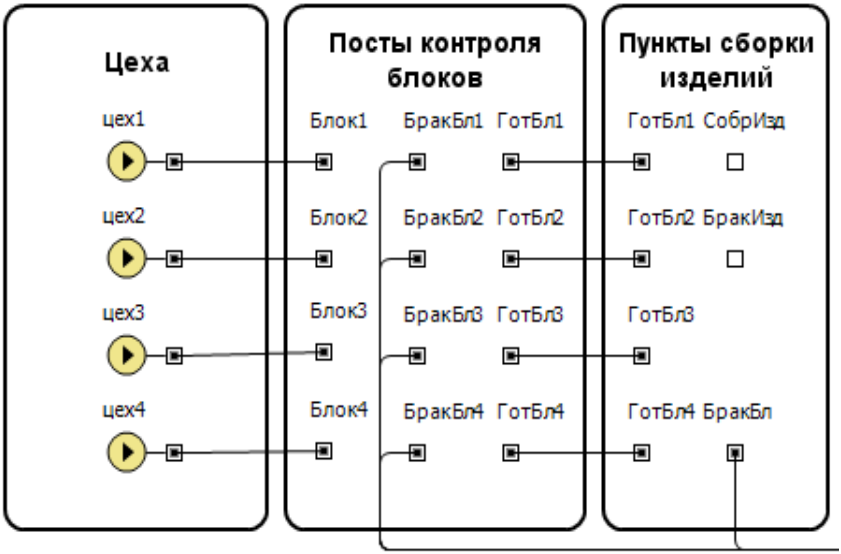


Рис. 3.107. Добавлен элемент диаграммы класса Sborka

После готовности четырех блоков для дальнейшей имитации процесса сборки нужно эти четыре заявки объединить в одну и интерпретировать ее как изделие. В AnyLogic имеются элементы некоторых классов, которые позволяют осуществить нужное нам объединение. Применим элементы класса Combine, позволяющие из двух заявок получить одну.

Процесс объединения будет происходить по мере готовности блоков, значит, на пункте сборки будет создаваться очередь, для имитации которой нужно использовать элемент queue.

Для имитации непосредственно процесса сборки следует взять элемент delay, а для разделения потока изделий на собранные первично и на изделия с заменными блоками после браковки — элемент selectOutPut.

Реализуйте, согласно изложенному, часть сегмента имитации работы пунктов сборки.

1. Из библиотеки Enterprise Library перетащите четыре элемента match и три элемента combine. Расположите и соедините их так, как на рис. 3.108.

2. Перетащите элементы queue, delay, selectOutPut, разместите, соедините и дайте им имена согласно рис. 3.108.

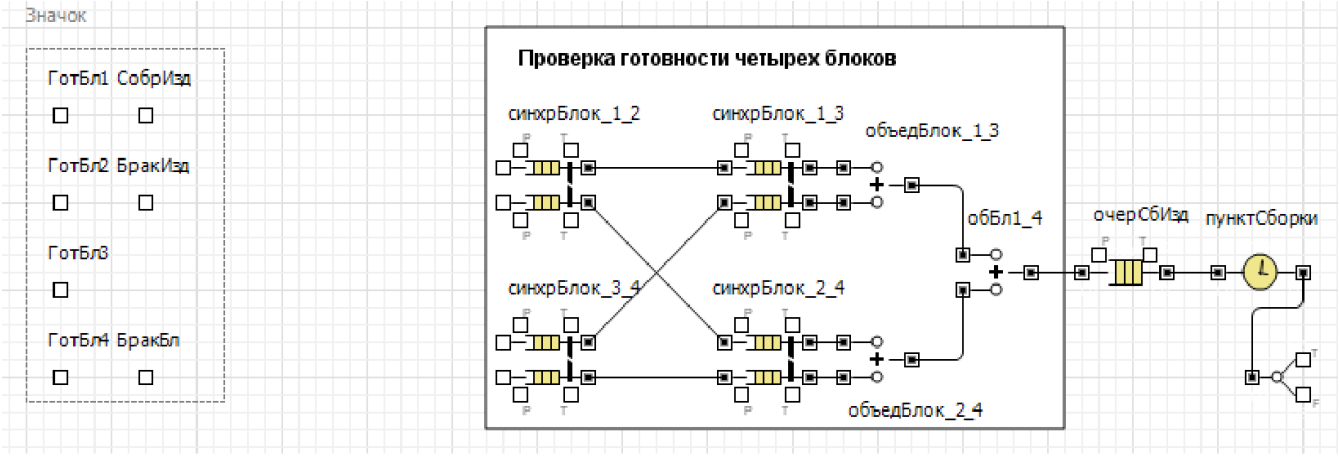


Рис. 3.108. Фрагмент сегмента имитации работы пунктов сборки

Поочередно выделите каждый из этих элементов и на страницах **Основные** панели **Свойства** установите для:

match…match3:

Классы заявок: Вход1, Вход2: `Product, Product`

Условия соответствия `true`

Максимальная вместимость 1

Максимальная вместимость 2

combine…combine2:

Классы заявок: Вход1, Вход2, Выход: `Product, Product, Product`

Объединенная заявка entity1

queue:

Класс заявки: `Product`

Максимальная вместимость

selectOutPut:

Класс заявки: `Product`

Выход true выбирается При выполнении условия

Условие `entity.sign1 == 0`

Элементы match и match1 обеспечивают синхронизацию движения блоков 1 и 2, 3 и 4 соответственно. А элементы match2 и match3 — блоков 1 и 3, 2 и 4 соответственно. Таким образом, обеспечивается синхронизация движения четырех блоков.

Указание свойства entity1 в элементах combine…combine2 позволяет получить на выходе сначала combine, а потом combine2 заявку, имитирующую изначально блок 1. Но теперь эта заявка будет имитировать изделие. Поэтому только для нее ранее было определено значение поля `entity.timeSbor` (см. рис. 3.100).

Изделия после элемента `пунктСборки` разделяются на два потока: собранные изделия первично (`entity.sign1=0`) и изделия с замененными блоками (`entity.sign1=2`). Разделение необходимо для учета количества изделий с замененными блоками и количества замененных блоков различных типов, затрат на первичную сборку изделий и на замену бракованных блоков (рис. 3.109).

Разделение потока изделий осуществляется элементом selectOutPut по условию `entity.sign1 == 0`, так как в случае, если изделие было забраковано на стендах выходного контроля или на пунктах приема изделий, признак `entity.sign1` будет равен 2.

Свойства элемента delay установите согласно рис. 3.109.

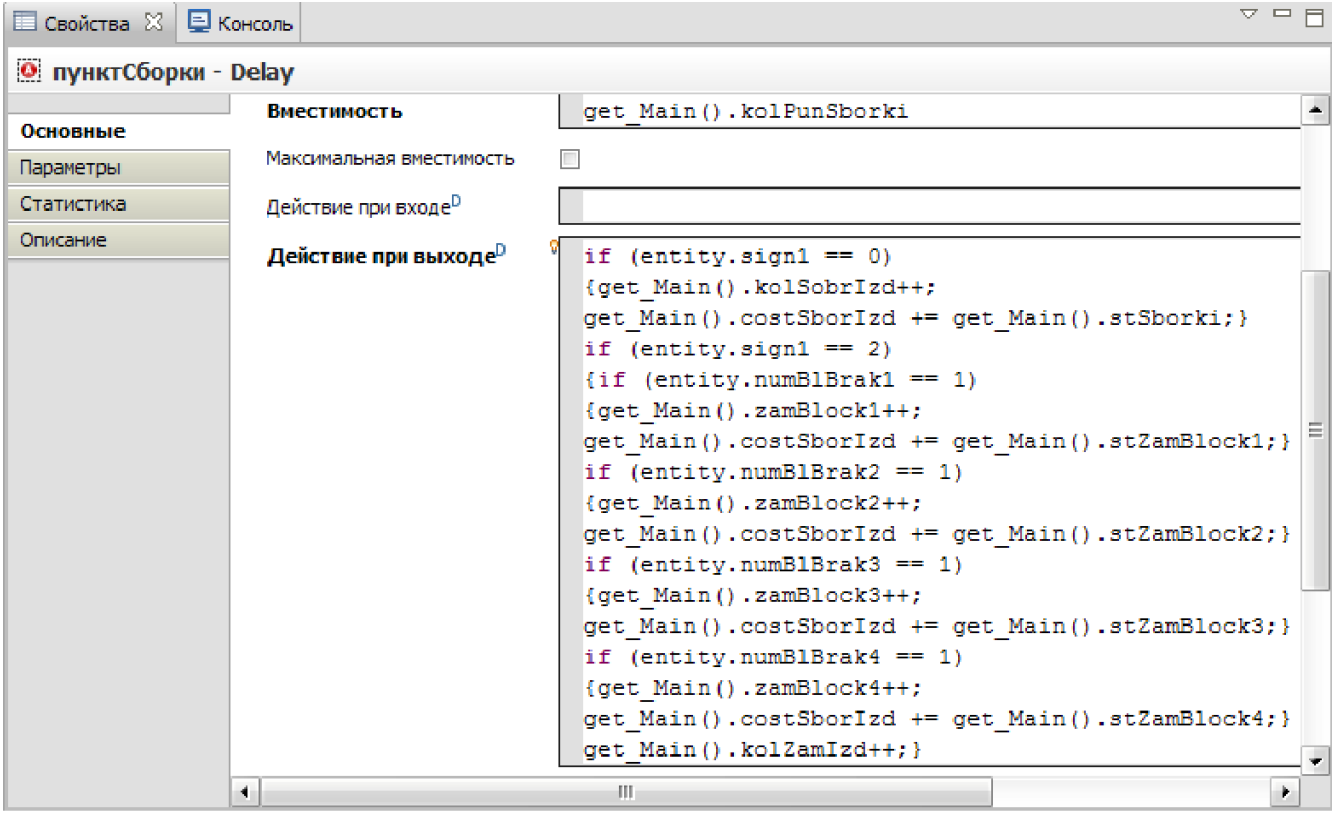


Рис. 3.109. Свойства элемента delay с именем пунктСборки

Продолжим построение сегмента имитации работы пунктов сборки изделий (рис. 3.110).

Бракованные изделия поступают через порт `БракИзд`. В заявке, имитирующей такое изделие, поле `etity.sign1 = 2`. В одном из полей `etity.numBlBrak1 … etity.numBlBrak4` этой же заявки также записана единица. Запись произведена при отбраковке на стендах контроля или пунктах приема изделий. Это мы сделаем (запишем) позже при построении этих сегментов. Для определения, какой из блоков 1 … 4 забракован в изделии, нужно использовать элемент selectOutput5. Этот элемент в отличие от элемента selectOutPut позволяет проверять пять условий и в зависимости от результата проверки направляет заявку на один из пяти выходов (обратите внимание на имена элементов: selectOut**Put** и selectOut**put**5).

Готовые блоки поступают через порты `ГотБл1 … ГотБл4`. Поэтому их надо было бы соединить с соответствующими входами элементов match … match1. Но в забракованном изделии нужно заменить какой-то из блоков. Значит, в случае наличия забракованного изделия надо направить из поступающих блоков соответствующий блок на замену. Сделать это можно с использованием элементов selectOutPut. Каждый порт `ГотБл1 … ГотБл4` соединить с входом соответствующего элемента selectOutPut. Выходы true этих элементов соединить с соответсвующими входами элементов match … match1. Таким образом, с выходов true готовые блоки будут направляться для первичной сборки изделий, а с выходов false — для замены бракованных блоков. В качестве условия разделения потоков можно использовать простые переменные `БрИздБл1 … БрИздБл4`. Например, если `БрИздБл2 ≠ 0`, то имеется забракованное изделие с блоком 2.

Для дальнейшей реализации процесса замены блока нужно объединить две заявки — имитирующую забракованное изделие и имитирующую блок для замены — в одну заявку. Для объединения двух заявок в одну воспользуйтесь уже известными вам элементами combine.

С выходов элементов combine заявки, имитирующие изделия для замены блоков, направляются в очередь `очерСбИзд` на входе непосредственно пунктов сборки изделий `пунктСборки`.

Как уже отмечалось ранее, изделия после пунктов сборки разделяются на два потока.

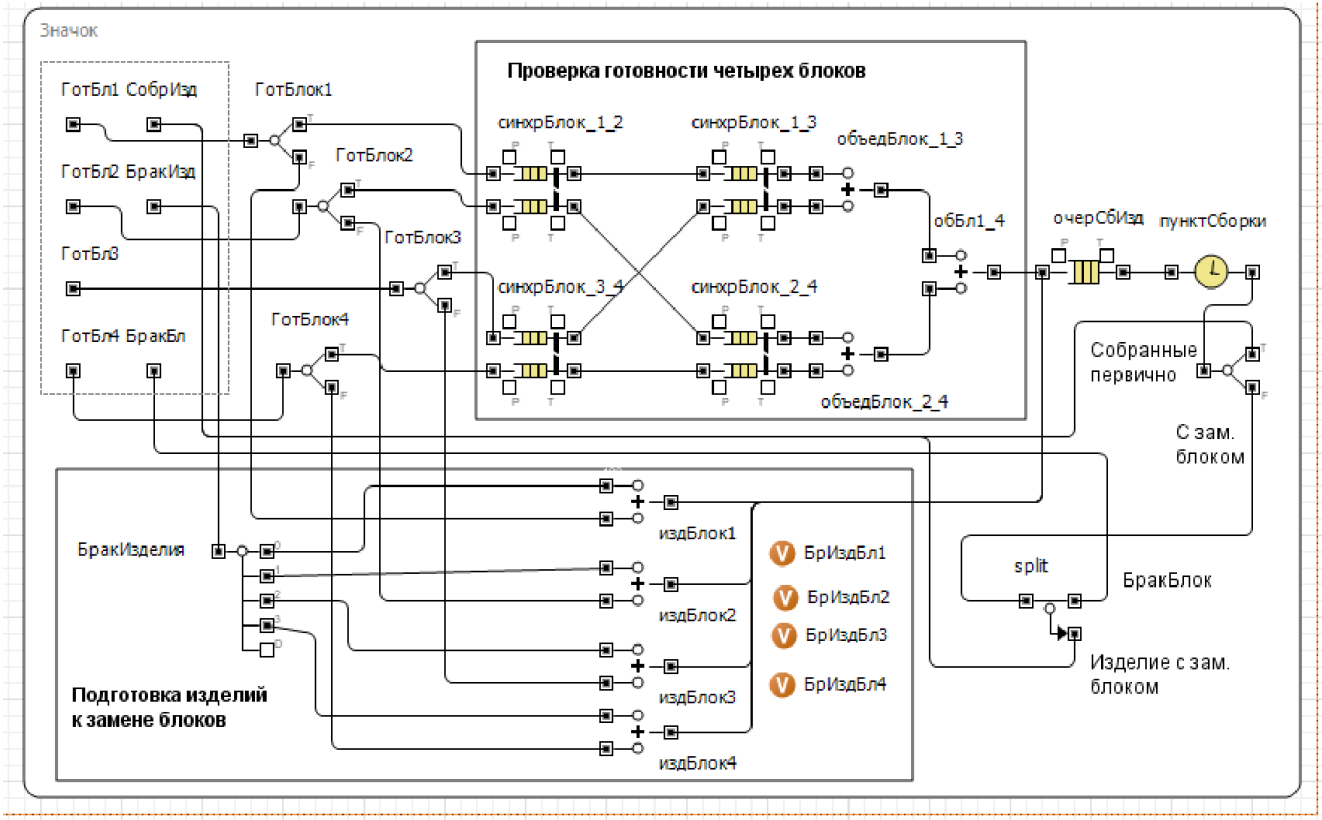


Рис. 3.110. Сегмент имитации пунктов сборки изделий

Собранные первично изделия с выхода true элемента selectOutPut направляются на стенды контроля, поэтому этот выход нужно соединить с портом `СобрИзд`.

Изделие с замененным блоком нужно вновь направить на стенды контроля. В тоже время его нужно учесть как забракованный блок и отправить на склад забракованных блоков. Значит, необходимо из одной заявки сделать две. Для этого используйте элемент split. Один выход этого элемента соедините с портом `БракБл`, а другой — с портом `СобрИзд`.

Выполните изложенные рекомендации практически. Перетащите четыре элемента класса SelectOutPut (или перетащите один элемент, а остальные три скопируйте), один элемент класса SelectOutPut5, четыре элемента класса Combine, один элемент класса Split и четыре простых переменных. Разместите, дайте им имена и соедините так, как на рис. 3.110.

Поочередно выделите новые элементы и установите их свойства согласно табл. 3.12.

Таблица 3.12

Свойства	Значение
selectOutPut … selectOutPut3	

Имя
Выход true выбирается
Условие
	`ГотБлок1`
	При выполнении условия
	`БракИздБл1 == 0`

Имя
Выход true выбирается
Условие
	`ГотБлок2`
	При выполнении условия
	`БракИздБл2 == 0`

Имя
Выход true выбирается
Условие
	`ГотБлок3`
	При выполнении условия
	`БракИздБл3 == 0`

Имя
Выход true выбирается
Условие
	`ГотБлок4`
	При выполнении условия
	`БракИздБл4 == 0`

selectOutput5	

Имя
Использовать
Условие 0
Условие 1
Условие 2
Условие 3
	`БракИзделия`
	Условия
	`entity.numBlBrak1 == 1`
	`entity.numBlBrak1 == 1`
	`entity.numBlBrak1 == 1`
	`entity.numBlBrak1 == 1`

Окончание табл. 3.12

Свойства	Значение
combine	

Имя
Классы заявок: Вход1, Вход2, Выход
Действие при входе 1
 	
Объединенная заявка
Действие при выходе
	`издБлок1`
	
	`Product, Product, Product`
	`if (entity.numBlBrak1 == 1) БрИздБл1 ++;`
	`entity1`
	`if (entity.numBlBrak1 == 1) БрИздБл1 --;`

Имя
Классы заявок: Вход1, Вход2, Выход
Действие при входе 1
 	
Объединенная заявка
Действие при выходе
	`издБлок2`
	
	`Product, Product, Product`
	`if (entity.numBlBrak2 == 1) БрИздБл2 ++;`
	`entity1`
	`if (entity.numBlBrak2 == 1) БрИздБл2 --;`

Имя
Классы заявок: Вход1, Вход2, Выход
Действие при входе 1
 	
Объединенная заявка
Действие при выходе
	`издБлок3`
	
	`Product, Product, Product`
	`if (entity.numBlBrak3 == 1) БрИздБл3 ++;`
	`entity1`
	`if (entity.numBlBrak3 == 1) БрИздБл3 --;`

Имя
Классы заявок: Вход1, Вход2, Выход
Действие при входе 1
 	
Объединенная заявка
Действие при выходе
	`издБлок4`
	
	`Product, Product, Product`
	`if (entity.numBlBrak4 == 1) БрИздБл4 ++;`
	`entity1`
	`if (entity.numBlBrak4 == 1) БрИздБл4 --;`

split	

Классы заявок: Оригинал, Копия:
Количество копий
Новая заявка (копия)
Действие при выходе копии
	`Product, Product`
	`1`
	`new Product()`
	`entity.sign1 = 0;`

3.4.7.4. Имитация работы стендов контроля изделий

Стенды контроля изделий предназначены для приема первично собранных изделий и изделий после замены забракованных блоков, непосредственно процесса контроля изделий, отправки прошедших контроль изделий на пункты приема, забракованных изделий — на пункты сборки, а также для приема забракованных изделий с пунктов приема изделий.

На стендах будет создаваться очередь, для имитации которой используйте элемент queue.

Для имитации непосредственно процесса контроля изделий используйте элемент delay.

По результатам контроля некоторые изделия будут признаны браком. Для отбраковки примените элементы selectOutPut.

1. Перетащите элементы queue, delay и два элемента selectOutPut на прямоугольник с именем **Стенды контроля изделий** диаграммы класса Main.
2. Разместите и соедините их согласно рис. 3.111.
3. Выделите элемент queue и установите на странице **Основные** панели **Свойства**:

Имя: `очерСтенКонтр`

Класс заявки: `Product`

Максимальная вместимость

4. Выделите элемент delay и установите его свойства:

Класс заявки: `Product`

Задержка задается `Явно`

Время задержки `exponential( 1/timeKontrIzd )`

Вместимость `kolStendKontrIzd`

Действие при выходе `testSobrIzd++;`

5. Выделите левый элемент selectOutPut и установите его свойства:

Имя: `БрПунКон`

Класс заявки: `Product`

Выход true выбирается `При выполнении условия`

Условие `entity.sign1 == 0`

Действие при выходе (false)

```
double a=0;
int numBlock1 = 0;
a = random();
if (a < 1) numBlock1 = 4;
```

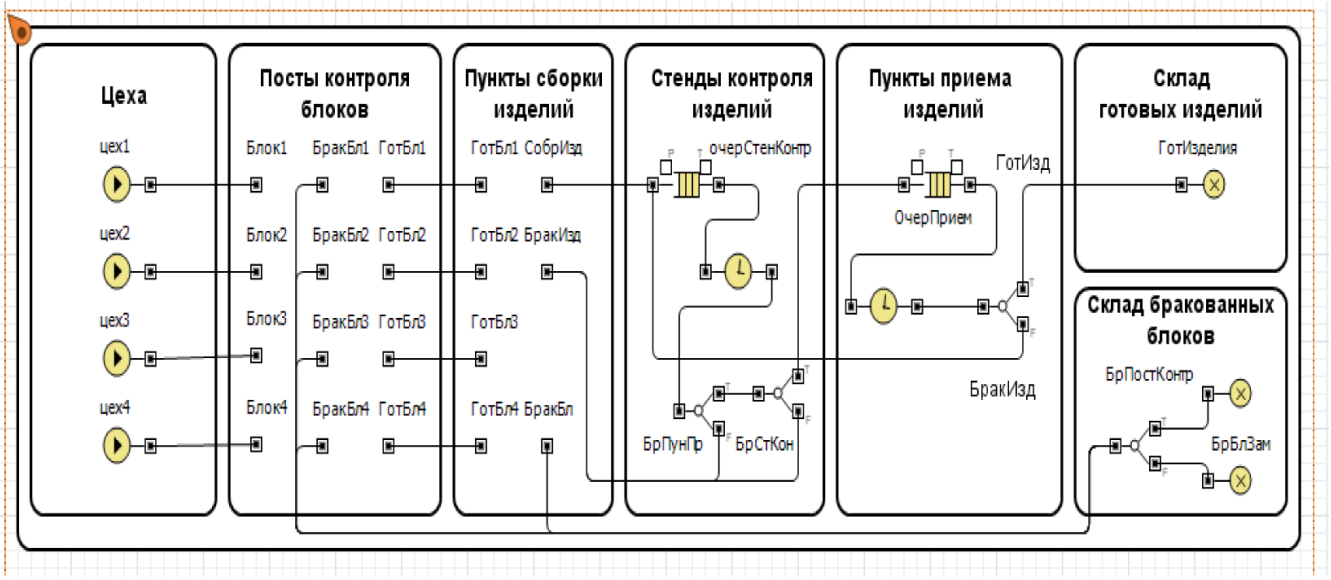


Рис. 3.111. Диаграмма класса Main с элементами всех сегментов

```
if (a <= (verBlNum1+verBlNum2 + verBlNum3)) numBlock=3;
if (a <= (verBlNum1 + verBlNum2)) numBlock1=2;
if (a <= verBlNum1) numBlock1=1;
if (numBlock1 == 1) {entity.numBlBrak1 = 1;
entity.timeSbor = exponential(1/timeZamBlock1);}
if (numBlock1 == 2) {entity.numBlBrak2 = 1;
entity.timeSbor = exponential(1/timeZamBlock2);}
if (numBlock1 == 3) {entity.numBlBrak3 = 1;
entity.timeSbor = exponential(1/timeZamBlock3);}
if (numBlock1 == 4) {entity.numBlBrak4 = 1;
entity.timeSbor = exponential(1/timeZamBlock4);}
entity.sign1 = 2;
```

6. Выделите правый элемент selectOutPut и установите его свойства:

Имя: `БрСтКон`

Класс заявки: `Product`

Выход true выбирается `С заданной вероятностью`

Вероятность[0..1] `1-procBrakIzd`

Действие при выходе (true) `costTestIzd += stKontrIzd;`

Действие при выходе (false)

```
double a = 0;
int numBlock = 0;
entity.sign1 = 2;
a = random();
if (a < 1) numBlock = 4;
if (a <= (verBlNum1+ verBlNum2 + verBlNum3)) numBlock=3;
if (a <= (verBlNum1 + verBlNum2)) numBlock = 2;
if (a <= verBlNum1) numBlock = 1;
if (numBlock == 1) {entity.numBlBrak1 = 1;
entity.timeSbor = exponential(1/timeZamBlock1);}
if (numBlock == 2) {entity.numBlBrak2 = 1;
entity.timeSbor = exponential(1/timeZamBlock2);}
if (numBlock == 3) {entity.numBlBrak3 = 1;
entity.timeSbor = exponential(1/timeZamBlock3);}
if (numBlock == 4) {entity.numBlBrak4 = 1;
entity.timeSbor = exponential(1/timeZamBlock4);}
brakSobrIzd ++;
```

Код в свойство **Действие при выходе (false)** обоих элементов selectOutPut введен для розыгрыша номера забракованного в изделии блока. В результате розыгрыша в одно из полей `entity.numBlBrak1… entity.numBlBrak4` заносится 1 —

признак брака. В поле `entity.timeSbor` — время замены соответствующего блока на пункте сборки. Полю `entity.sign1` присваивается значение 2 — признак брака в изделии.

3.4.7.5. Имитация работы пунктов приема изделий

Пункты приема изделий предназначены для приема прошедших стенды контроля изделий, непосредственно приема изделий, отправки прошедших прием изделий на склад готовых изделий, а забракованных изделий — на стенды контроля.

На пунктах приема будет создаваться очередь, для имитации которой используйте элемент queue.

Для имитации непосредственно процесса приема изделий используйте элемент delay.

По результатам контроля некоторые изделия будут признаны браком. Для отбраковки воспользуйтесь элементом selectOutPut.

1. Перетащите элементы queue, delay и selectOutPut на прямоугольник с **Пункты приема изделий** диаграммы класса Main.
2. Разместите и соедините их согласно рис. 3.111.
3. Выделите элемент queue и установите на странице **Основные** панели **Свойства**:

Имя: `очерПрием`

Класс заявки: `Product`

Максимальная вместимость

4. Выделите элемент delay и установите его свойства:

Класс заявки: `Product`

Задержка задается `Явно`

Время задержки `exponential( 1/timePriemIzd )`

Вместимость `kolPunPriem`

Действие при выходе `kolPriemIzd++;`

5. Выделите элемент selectOutPut и установите его свойства:

Класс заявки: `Product`

Выход true выбирается `С заданной вероятностью`

Вероятность[0..1] `1-procBrakPriem`

Действие при выходе (true) `costPriemkiIzd+=stPriemIzd;`

Действие при выходе (false)

```
entity.sign1 = 2;
brakPriemIzd++;
```

Код свойства **Действие при выходе** элемента delay введен для счета количества `kolPriemIzd` принятых всего изделий.

Код свойства **Действие при выходе (false)** элемента selectOutPut считает количество `brakPriemIzd` забракованных изделий и полю `entity.sign1` присваивает 2 — признак брака в изделии.

3.4.7.6. Имитация склада готовых изделий

Для имитации склада готовых изделий используйте элемент sink со следующими свойствами:

Имя: `ГотИзделия`

Класс заявки: `Product`

Действие при выходе

```
kolGotIzd++;
costGotIzd+=(stKomplBlock1+stKomplBlock2+
stKomplBlock3+stKomplBlock4)+
(stIzgBlock1+stIzgBlock2+stIzgBlock3+stIzgBlock4)+
(stTestBlock1+stTestBlock2+stTestBlock3+
stTestBlock4)+stSborki+stKontrIzd+stPriemIzd;
koefIncrCostIzd = (costGotIzd+costBrakBlock)/costGotIzd;
ostGotBlock1 = gotBlock1 - kolGotIzd - zamBlock1;
ostGotBlock2 = gotBlock2 - kolGotIzd - zamBlock2;
ostGotBlock3 = gotBlock3 - kolGotIzd - zamBlock3;
ostGotBlock4 = gotBlock4 - kolGotIzd - zamBlock4;
```

Код свойства **Действие при выходе** введен для счета количества `kolGotIzd` готовых изделий, их стоимости `costGotIzd` и коэффициента `koefIncrCostIzd` увеличения себестоимости изделия вследствие наличия брака.

Кроме того, ведется счет готовых для сборки блоков, то есть изготовленных цехами и проверенных на постах контроля, но не использованных для сборки изделий блоков по типам `ostGotBlock1`… `ostGotBlock4` на текущее модельное время.

3.4.7.7. Имитация склада бракованных блоков

Ранее (см. п. 3.4.6.2) для имитации склада бракованных блоков было рекомендовано использовать один блок sink.

Предположим, что требуется вести раздельный учет забракованных блоков на постах контроля цехов и забракованных на стендах контроля и пунктах приема изделий. Для разделения потока брака полю `entity.sign1` присваивались признаки 1 и 2.

1. Перетащите элементы selectOutPut и sink на прямоугольник с именем **Склад бракованных блоков** диаграммы класса Main.

2. Разместите и соедините их согласно рис. 3.111.

3. Поочередно выделите элементы имитации склада бракованных блоков, начиная с элемента selectOutPut, и установите свойства согласно табл. 3.13.

Коды в свойстве **Действие при выходе** обоих элементов sink одинаковые. Они предназначены для счета:

стоимости `costzbrakBlock` забракованных блоков не по типам, а в целом всех блоков;

количества забракованных блоков по типам `allBrakBlock1`… `allBrakBlock4`.

Таблица 3.13

Свойства	Значение
selectOutPut	

Класс заявки:
Выход true выбирается
Условие
	`Product`
	При выполнении условия
	`entity.sign1 == 1`

sink	

Имя
Класс заявки:
Действие при выходе
	`БрБлЗам`
	`Product`
	`if (entity.numBlBrak1 == 1)`
	`{costBrakBlock += (stKomplBlock1 + stIzgBlock1 + stTestBlock1 + stZamBlock1);`
	`allBrakBlock1++;}`
	`if (entity.numBlBrak2 == 1)`
	`{costBrakBlock += (stKomplBlock2 + stIzgBlock2 + stTestBlock2 + stZamBlock2);`
	`allBrakBlock2++;}`
	`if (entity.numBlBrak3 == 1)`
	`{costBrakBlock += (stKomplBlock3 + stIzgBlock3 + stTestBlock3 + stZamBlock3);`
	`allBrakBlock3++;}`
	`if (entity.numBlBrak4 == 1)`
	`{costBrakBlock += (stKomplBlock4 + stIzgBlock4 + stTestBlock4 + stZamBlock4);`
	`allBrakBlock4++;}`

Окончание табл. 3.13

Свойства	Значение

Имя
Класс заявки:
Действие при выходе
	БрПостКонтр
	Product
	`if (entity.numBlBrak1 == 1) {costBrakBlock += (stKomplBlock1 + stIzgBlock1 + stTestBlock1); allBrakBlock1++;}`
	`if (entity.numBlBrak2 == 1) {costBrakBlock += (stKomplBlock2 + stIzgBlock2 + stTestBlock2); allBrakBlock2++;}`
	`if (entity.numBlBrak3 == 1) {costBrakBlock += (stKomplBlock3 + stIzgBlock3 + stTestBlock3); allBrakBlock3++;}`
	`if (entity.numBlBrak4 == 1) {costBrakBlock += (stKomplBlock4 + stIzgBlock4 + stTestBlock4); allBrakBlock4++;}`

Замечание. Можно обойтись и одним элементом sink для имитации склада бракованных блоков. Для этого нужно оба кода (см. табл. 3.13) занести в его свойство **Действие при выходе.**

В начало первого кода добавить:

```
if (entity.sign1 == 1)
{  код для БрПостКонтр  }
```

а во второго

```
if (entity.sign1 == 2)
{  код для БрБлЗам      }
```

Количественный результат будет аналогичным, только во время моделирования не сможете визуально наблюдать раздельные потоки забракованных блоков.

3.4.7.8. Организация переключения между областями просмотра

Добавьте свои собственные элементы презентации на ранее созданные области просмотра Mainview, Data, Result, Kontrol и Sbor, щелчком на которых будет производиться переход к той или иной области просмотра. Сделайте так, чтобы из любой области можно было переходить в любую другую область.

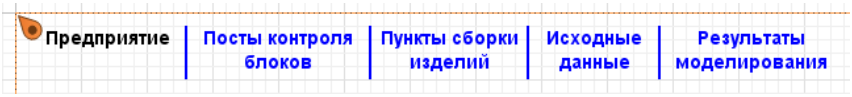


Рис. 3.112. Элементы презентации для переключения

Для этого разместите на каждой области просмотра элементы презентации, показанные на рис. 3.112.

Начните с области просмотра Mainview.

1. В **Палитре** выделите **Презентация**. Перетащите элемент **text** на область просмотра Mainview, разместите и введите в поле **Текст:** `Предприятие` (рис. 3.112). На странице **Основные** панели **Свойства** установите в поле **Цвет:** black, а также выберите выравнивание текста и шрифт.

2. Перетащите второй элемент **text**, разместите и введите в поле **Текст:** `Посты контроля блоков`. На странице **Основные** панели **Свойства** установите в поле **Цвет:** blue, а также выберите выравнивание текста и шрифт. На панели **Свойства** выделите **Динамические** и в поле **Действие по щелчку:** введите код: `test.Kontrol.navigateTo();`

Замечание. Можете выбрать и другие цвета элементов презентации. При этом нужно исходить из того, чтобы различались по цвету текст открытой области просмотра и тексты закрытых областей просмотра.

3. Перетащите третий элемент **text**, разместите и введите в поле **Текст:** `Пункты сборки изделий`. На панели **Свойства** выделите **Динамические** и в поле **Действие по щелчку:** введите: `sborka.Sbor.navigateTo();`

4. Перетащите четвертый элемент **text**, разместите и введите в поле **Текст:** `Исходные данные`. На панели **Свойства** выделите **Динамические** и в поле **Действие по щелчку:** введите: `Data.navigateTo();`

5. Перетащите пятый элемент **text**, разместите и введите в поле **Текст:** `Результаты моделирования`. На панели **Свойства** выделите **Динамические** и в поле **Действие по щелчку:** введите: `Result.navigateTo();`

Таким образом, с области просмотра Mainview можно будет переходить на любую из четырех областей просмотра. При этом элемент презентации открытой области высвечивается черным цветом, а остальные элементы презентации — голубым цветом.

1. Скопируйте элементы презентации так, как они показаны на рис. 3.112.

2. Последовательно переходите на остальные области просмотра, вставьте скопированные элементы презентации и внесите правки в их свойства согласно табл. 3.14.

Таблица 3.14

Элемент презентации	Действие по щелчку
Data	

Предприятие
Посты контроля блоков
Пункты сборки изделий
Исходные данные
Результаты моделирования
	`Mainview.navigateTo();`
	`test.Kontrol.navigateTo();`
	`sborka.Sbor.navigateTo();`
	
	`Result.navigateTo();`

Result	

Предприятие
Посты контроля блоков
Пункты сборки изделий
Исходные данные
Результаты моделирования
	`Mainview.navigateTo();`
	`test.Kontrol.navigateTo();`
	`sborka.Sbor.navigateTo();`
	`Data.navigateTo();`

Kontrol	

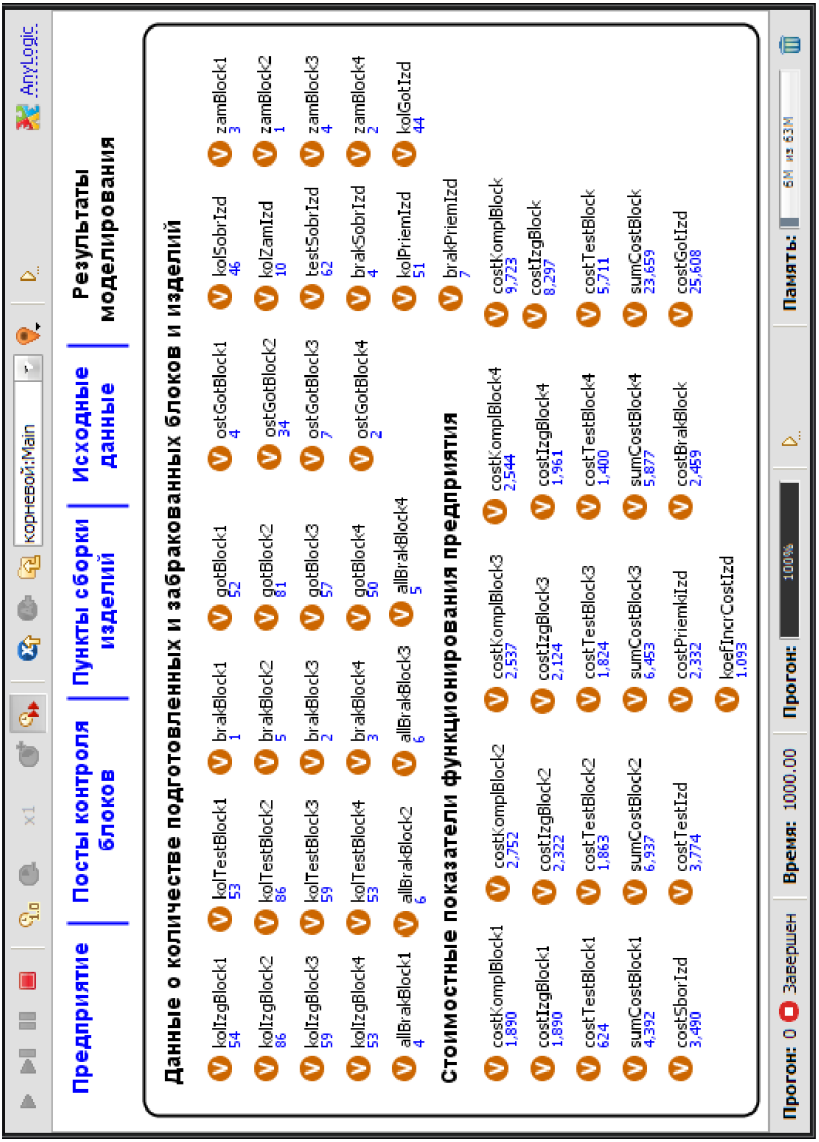
Предприятие
Посты контроля блоков
Пункты сборки изделий
Исходные данные
Результаты моделирования
	`get_Main().Mainview.navigateTo();`
	
	`get_Main().sborka.Sbor.navigateTo();`
	`get_Main().Data.navigateTo();`
	
	`get_Main().Result.navigateTo();`

Sbor	

Предприятие
Посты контроля блоков
Пункты сборки изделий
Исходные данные
Результаты моделирования
	`get_Main().Mainview.navigateTo();`
	`get_Main().test.Kontrol.navigateTo();`
	
	`get_Main().Data.navigateTo();`
	
	`get_Main().Result.navigateTo();`

Построение модели для проведения простого эксперимента завершено. Теперь отладьте модель и проведите эксперимент. Результаты моделирования приведены на рис. 3.113 и рис. 3.114. Изготовлено 44 изделия (kolGotIzd). При этом коэффициент увеличения себестоимости изделий составил 1,093 (koefIncrCostIzd).

Замечание. При каждом запуске модели результаты будут повторяться, если в модели используются только распределения и случайные функции из AnyLogic. Например, если вместо uniform(1) использовать random(), которая является функцией Java, то результаты моделирования воспроизводиться не будут.



3.113. Результаты моделирования. Область просмотра Result

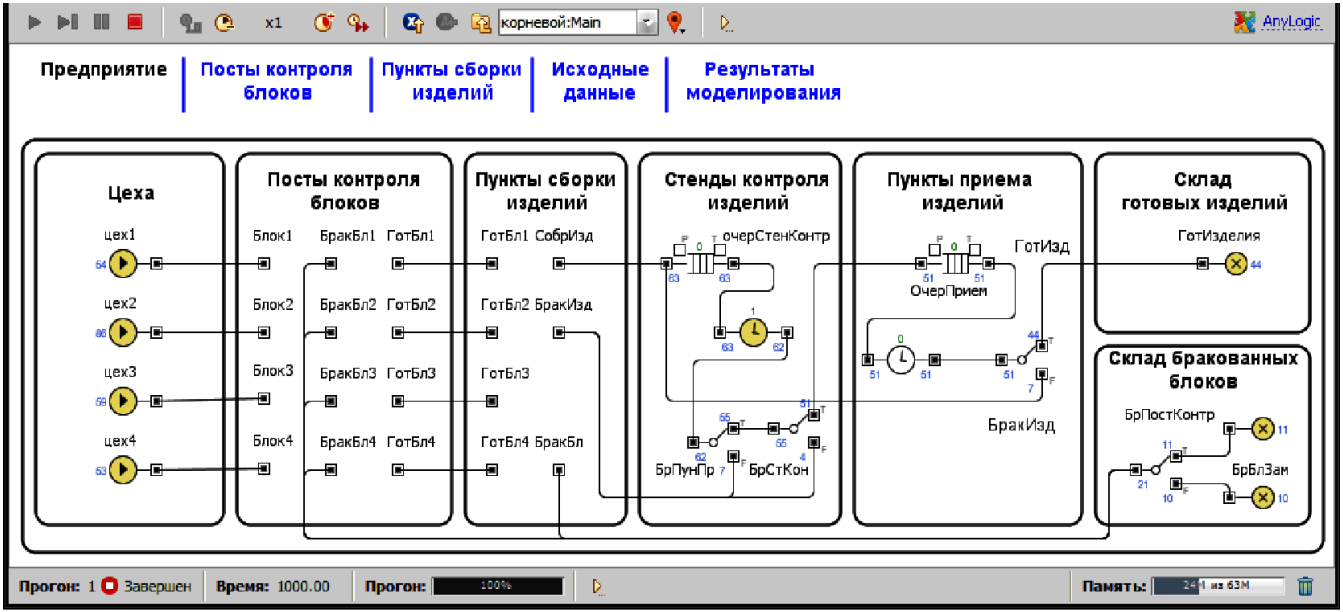


Рис. 3.114. Результаты моделирования. Диаграмма класса Main

Глава 4. Задания на проектирование. Основные направления

Вариант 1

Постановка задачи

Направление связи состоит из n1 основных, n2 резервных каналов связи, общего накопителя емкостью на L сообщений, n источников. Интервалы T1, T2, …, Tn поступления сообщений случайные. При нормальной работе сообщения передаются по основным каналам. Время Тп1, Тп2, …, Тпn передачи случайное.

Основные каналы подвержены отказам. Интервалы времени Тот1, Тот2, …, Тотn1 между отказами случайные. Если отказ происходит во время передачи, то отыскивается исправный и свободный основной канал. Если такого нет, включается один из резервных каналов, если он исправен и свободен. Время Твк1, Твк2, …, Твкn2 включения постоянное для соответствующего канала. Сообщение, передача которого была прервана, передается по включенному резервному каналу. Время Тпр1, Тпр2, …, Тпрn2 передачи случайное.

Отказавший основной канал восстанавливается. Время Тв1, Тв2, …, Твn1 восстановления случайное. После восстановления резервный канал выключается, и восстановленный канал продолжает работу с передачи очередного сообщения.

Резервные каналы также подвержены отказам. Интервалы времени Тотр1, Тотр2, …, Тотрn2 между отказами случайные. Отказавший резервный канал восстанавливается. Время Твр1, Твр2, …, Тврn2 восстановления случайное. Для прерванного сообщения отыскивается возможность передачи по любому исправному и свободному каналу.

В случае полного заполнения накопителя, поступающие сообщения теряются.

Задание на исследование

Разработать имитационную модель функционирования направления связи. Исследовать влияние емкости L накопителя, интервалов времени поступления сообщений на время передачи направлением связи N сообщений. Провести дисперсионный анализ. Факторы и их уровни выбрать самостоятельно. Результаты моделиро-

вания необходимо получить с точностью ε = 0,1 и доверительной вероятностью α = 0,95.

Сделать выводы о загруженности каналов связи и необходимых мерах по повышению эффективности функционирования направления связи.

Вариант 2

Постановка задачи

Направление связи состоит из n1 основных, n2 резервных каналов связи, общего накопителя емкостью на L сообщений, n источников. Интервалы T1, T2, …, Tn поступления сообщений случайные. При нормальной работе сообщения передаются по основным каналам. Время Тп1, Тп2, …, Тпn передачи случайные.

Основные каналы подвержены отказам. Интервалы времени Тот1, Тот2, …, Тотn1 между отказами случайные. Если отказ происходит во время передачи, то отыскивается исправный и свободный основной канал. Если такого нет, включается один из резервных каналов, если он исправен и свободен. Время Твк1, Твк2, …, Твкn2 включения постоянное для соответствующего канала. Сообщение, передача которого была прервана, передается по включенному резервному каналу. Время Тпр1, Тпр2, …, Тпрn2 передачи случайное.

Отказавший основной канал восстанавливается. Время Тв1, Тв2, …, Твn1 восстановления случайное. После восстановления резервный канал выключается, и восстановленный канал продолжает работу с передачи очередного сообщения.

Резервные каналы также подвержены отказам. Интервалы времени Тотр1, Тотр2, …, Тотрn2 между отказами случайные. Отказавший резервный канал восстанавливается. Время Твр1, Твр2, …, Тврn2 восстановления случайное. Для прерванного сообщения отыскивается возможность передачи по любому исправному и свободному каналу.

В случае полного заполнения накопителя, поступающие сообщения теряются.

Задание на исследование

Разработать имитационную модель функционирования направления связи. Исследовать влияние емкости накопителя, интервалов времени поступления сообщений и количества каналов на вероятность отказа в передаче сообщений от каждого источника и по на-

правлению связи в целом. Время моделирования — T часов. Провести дисперсионный анализ. Факторы и их уровни выбрать самостоятельно.

Сделать выводы о загруженности каналов связи и необходимых мерах по повышению эффективности функционирования направления связи.

Вариант 3

Постановка задачи

Направление связи состоит из n1 основных, n2 резервных каналов связи, общего накопителя емкостью на L сообщений, n источников. Интервалы T1, T2, …, Tn поступления сообщений случайные. При нормальной работе сообщения передаются по основным каналам. Время Тп1, Тп2, …, Тпn передачи случайные.

Основные каналы подвержены отказам. Интервалы времени Тот1, Тот2, …, Тотn1 между отказами случайные. Если отказ происходит во время передачи, отыскивается исправный и свободный основной канал. Если такого нет, включается один из резервных каналов, если он исправен и свободен. Время Твк1, Твк2, …, Твкn2 включения постоянное для соответствующего канала. Сообщение, передача которого была прервана, передается по включенному резервному каналу. Время Тпр1, Тпр2, …, Тпрn2 передачи случайное. Отказавший основной канал восстанавливается. Время Тв1, Тв2, …, Твn1 восстановления случайное. После восстановления резервный канал выключается, и восстановленный канал продолжает работу с передачи очередного сообщения.

Резервные каналы также подвержены отказам. Интервалы времени Тотр1, Тотр2, …, Тотрn2 между отказами случайные. Отказавший резервный канал восстанавливается. Время Твр1, Твр2, …, Тврn2 восстановления случайное. Для прерванного сообщения отыскивается возможность передачи по любому исправному и свободному каналу.

Сообщения источника 1 обладают абсолютным приоритетом по отношению к сообщениям других источников. Вследствие этого, если при поступлении сообщения от источника 1 все каналы заняты также передачей сообщений от источника 1, то прерывания не происходит и заявка считается потерянной. Если же есть передача сообщений от других источников, то передача любого из них прерывается и начинается передача сообщения от источника 1. Сооб-

щения более низких категорий теряются. В случае полного заполнения накопителя, поступающие сообщения теряются.

Задание на исследование

Разработать имитационную модель функционирования направления связи в течение T часов. Исследовать влияние емкости накопителя, интервалов времени поступления сообщений и количества каналов на вероятность отказа в передаче сообщений от каждого источника и по направлению связи в целом.

Провести дисперсионный анализ. Факторы и их уровни выбрать самостоятельно. Результаты моделирования необходимо получить с точностью $\varepsilon = 0{,}01$ и доверительной вероятностью $\alpha = 0{,}95$.

Сделать выводы о загруженности каналов связи и необходимых мерах по повышению эффективности функционирования направления связи.

Вариант 4

Постановка задачи

Предприятие имеет n1 цехов, производящих n1 типов блоков, т. е. каждый цех производит блоки одного типа. Интервалы выпуска блоков T1, T2, …, Tn1 — случайные. Из n1 блоков собирается одно изделие.

Перед сборкой каждый тип блоков проверяется на n11, n12, …, n1n соответствующих постах. Длительности контроля одного соответствующего блока T11, T12, …, T1n — случайные. На каждом посту бракуется q11, q12, …, q1n % блоков соответственно. Эти блоки в дальнейшем процессе сборки не участвуют и удаляются с постов контроля.

Прошедшие контроль, т. е. не забракованные блоки поступают на один из n2 пунктов сборки. На каждом пункте сборки одновременно собирается только одно изделие. Сборка начинается только тогда, когда имеются все необходимые n1 блоков различных типов. Время сборки Tc случайное.

После сборки изделие поступает на один из n3 стендов выходного контроля. На одном стенде одновременно проверяется только одно изделие. Время проверки Tп случайное. По результатам проверки бракуется q2 % изделий.

Забракованное изделие направляется в цех сборки, где неработоспособные блоки заменяются новыми. Время замены Tз случай-

ное. После замены блоков изделие вновь поступает на один из стендов выходного контроля.

Прошедшие стенд выходного контроля изделия поступают в отдел военной приемки. Время приемки Тпр одного изделия случайное. По результатам приемки бракуется q4 % изделий, которые направляются вновь на стенд выходного контроля.

Принятые военной приемкой изделия направляются на склад.

Задание на исследование

Разработать имитационную модель функционирования предприятия. Исследовать влияние интервалов выпуска блоков из цехов на количество и среднее время подготовки изделий, принятых военной приемкой в течение недели.

Провести дисперсионный анализ. Факторы и их уровни выбрать самостоятельно. Результаты моделирования необходимо получить с точностью $\varepsilon = 1$ и доверительной вероятностью $\alpha = 0{,}95$.

Сделать выводы о загруженности подразделений предприятия и необходимых мерах по повышению эффективности их функционирования.

Вариант 5

Постановка задачи

Предприятие имеет n1 цехов, производящих n1 типов блоков, т. е. каждый цех производит блоки одного типа. Интервалы выпуска блоков T1, T2, …, Tn1 — случайные. Из n1 блоков собирается одно изделие.

Перед сборкой каждый тип блоков проверяется на n11, n12, …, n1n соответствующих постах. Длительности контроля одного соответствующего блока T11, T12, …, T1n — случайные. На каждом посту бракуется q11, q12, …, q1n % блоков соответственно. Эти блоки в дальнейшем процессе сборки не участвуют и удаляются с постов контроля.

Прошедшие контроль, т. е. не забракованные блоки поступают на один из n2 пунктов сборки. На каждом пункте сборки одновременно собирается только одно изделие. Сборка начинается только тогда, когда имеются все необходимые n1 блоков различных типов. Время сборки Тс случайное.

После сборки изделие поступает на один из n3 стендов выходного контроля. На одном стенде выходного контроля одновременно может проверяться только одно собранное изделие. Время про-

верки Тп случайное. По результатам проверки бракуется q2 % изделий.

Забракованное изделие направляется в цех сборки, где неработоспособные блоки заменяются новыми. Время замены Тз случайное. После замены блоков изделие вновь поступает на один из стендов выходного контроля.

Прошедшие стенд выходного контроля изделия поступают в отдел военной приемки. Время приемки Тпр одного изделия случайное. По результатам приемки бракуется q4 % изделий, которые направляются вновь на стенд выходного контроля.

Принятые военной приемкой изделия направляются на склад.

Задание на исследование

Разработать имитационную модель функционирования предприятия. Исследовать влияние интервалов выпуска блоков из цехов и их качества на время выпуска принятых военной приемкой N изделий.

Провести дисперсионный анализ. Факторы и их уровни выбрать самостоятельно. Результаты моделирования необходимо получить с точностью $\varepsilon = 0{,}1$ и доверительной вероятностью $\alpha = 0{,}95$.

Сделать выводы о загруженности подразделений предприятия и необходимых мерах по повышению эффективности его функционирования.

Вариант 6

Постановка задачи

Предприятие имеет n1 цехов, производящих n1 типов блоков, т. е. каждый цех производит блоки одного типа. Интервалы выпуска блоков T1, T2, …, Tn1 — случайные. Из n1 блоков собирается одно изделие.

Перед сборкой каждый тип блоков проверяется на n11, n12, …, n1n соответствующих постах. Длительности контроля одного соответствующего блока T11, T12, …, T1n — случайные. На каждом посту бракуется q11, q12, …, q1n % блоков соответственно. Эти блоки в дальнейшем процессе сборки не участвуют и удаляются с постов контроля.

Прошедшие контроль, т. е. не забракованные блоки поступают на один из n2 пунктов сборки. На каждом пункте сборки одновременно собирается только одно изделие. Сборка начинается только

тогда, когда имеются все необходимые n1 блоков различных типов. Время сборки Тс случайное.

После сборки изделие поступает на один из n3 стендов выходного контроля. На одном стенде одновременно проверяется одно изделие. Время проверки Тп случайное. По результатам проверки бракуется q2 % изделий. Причиной брака может быть от одного до q3 блоков.

Забракованное изделие направляется в цех сборки, где неработоспособные блоки заменяются новыми. Время замены Тз одного блока случайное. После замены блоков изделие вновь поступает на один из стендов выходного контроля. Блоки, которые были заменены только один раз, вновь направляются на соответствующие посты входного контроля. Блоки, замененные более одного раза, в дальнейшем в процессе сборки изделия не участвуют и удаляются.

Прошедшие стенд выходного контроля изделия поступают в отдел военной приемки. Время приемки Тпр одного изделия случайное. По результатам приемки бракуется q4 % изделий, которые направляются вновь на стенд выходного контроля и снова проверяются.

Принятые военной приемкой изделия направляются на склад.

Задание на исследование

Разработать имитационную модель функционирования предприятия. Исследовать влияние качества изготовления блоков на количество принятых военной приемкой изделий в течение недели (42 часов).

Провести дисперсионный анализ. Факторы и их уровни выбрать самостоятельно. Результаты моделирования необходимо получить с точностью $\varepsilon = 1$ и доверительной вероятностью $\alpha = 0{,}99$.

Сделать выводы о загруженности подразделений предприятия и необходимых мерах по повышению эффективности его функционирования.

Вариант 7

Постановка задачи

На вычислительный комплекс коммутации сообщений (ВККС) поступают сообщения от n1 абонентов с интервалами времени T1, T2, …, Tn1. Сообщения могут быть n2 категорий с вероятностями p1, p2, …, pn2 (p1+p2+...+pn2 = 1) и вычислительными сложностя-

ми S1, S2,... , Sn2 операций (оп) соответственно. Вычислительные сложности случайные. Сообщения 1-й категории обладают относительным приоритетом по отношению к сообщениям остальных категорий. ВККС имеет входной накопитель емкостью L1 байт для хранения сообщений, ожидающих передачи. В буфере сообщения размещаются в соответствии с приоритетом.

ВККС обрабатывает сообщения с производительностью Q оп/с. После обработки сообщения передаются по n3 направлениям, каждое из которых имеет k1, k2, …, kn3 каналов связи. Вероятности передачи сообщений по направлениям от любого источника p1, p2, …, pn3 (p1+p2+…+pn3 = 1). Скорость передачи Vп бит/с.

Если после обработки сообщения все каналы связи направления заняты, то обработанное сообщение помещается в накопитель каналов связи, если в нем есть место. При отсутствии места в накопителе каналов связи сообщение теряется. Емкость накопителя каналов связи ограничена L2 сообщениями.

ВККС и каналы связи имеют конечную надежность. Интервалы времени Тот1 и Тот2 между отказами ВККС и каналов связи случайные. Длительности восстановления Тв1 и Тв2 ВККС и каналов связи случайные.

При отказе канала связи передаваемые сообщения 1-й категории сохраняются в накопителе каналов, если в нем есть место. При выходе из строя ВККС с вероятностью Рс все сообщения в накопителе ВККС и накопителе каналов связи сохраняются, обрабатываемое сообщение теряется, а прием ВККС и передача сообщений по каналам связи прекращается. Поступающие в это время сообщения теряются.

Задание на исследование

Разработать имитационную модель функционирования ВККС. Исследовать влияние емкостей входных накопителей, интервалов времени и вероятностей на время передачи N сообщений и среднее время обработки одного сообщения.

Провести дисперсионный анализ. Факторы и их уровни выбрать самостоятельно. Результаты моделирования необходимо получить с точностью $\varepsilon = 0{,}1$ и доверительной вероятностью $\alpha = 0{,}99$.

Сделать выводы о загруженности элементов ВККС и необходимых мерах по повышению эффективности его функционирования.

Вариант 8

Постановка задачи

На вычислительный комплекс коммутации сообщений (ВККС) поступают сообщения от n1 абонентов с интервалами времени T1, T2, …, Tn1. Сообщения могут быть n2 категорий с вероятностями p1, p2, …, pn2 (p1+p2+...+pn2 = 1) и вычислительными сложностями S1, S2,... , Sn2 операций (оп) соответственно. Вычислительные сложности случайные. ВККС имеет входной накопитель емкостью L1 байт для хранения сообщений, ожидающих передачи. Сообщения 1-й категории обладают относительным приоритетом по отношению к сообщениям остальных категорий при обработке на ВККС. В буфере сообщения размещаются в соответствии с приоритетом.

ВККС обрабатывает сообщения с производительностью Q оп/с. После обработки сообщения передаются по n3 направлениям, каждое из которых имеет k1, k2, …, kn3 каналов связи. Вероятности передачи сообщений по направлениям от любого источника p1, p2, …, pn3 (p1+p2+…+pn3 = 1). Скорость передачи Vп бит/с.

Если после обработки сообщения все каналы связи направления заняты, то обработанное сообщение помещается в накопитель каналов связи, если в нем есть место. При отсутствии места в накопителе каналов связи сообщение теряется. Емкость накопителя каналов связи ограничена L2 сообщениями.

ВККС и каналы связи имеют конечную надежность. Интервалы времени Тот1 и Тот2 между отказами ВККС и каналов связи случайные. Длительности восстановления Тв1 и Тв2 ВККС и каналов связи случайные.

При отказе канала связи передаваемые сообщения 1-й категории сохраняются в накопителе каналов, если в нем есть место. При выходе из строя ВККС с вероятностью Рс все сообщения в накопителе ВККС и накопителе каналов связи сохраняются, обрабатываемое сообщение теряется, а прием ВККС и передача сообщений по каналам связи прекращается. Все поступающие в это время сообщения теряются.

Задание на исследование

Разработать имитационную модель функционирования ВККС. Исследовать влияние емкостей входных накопителей, интервалов времени, вероятностей, Q и Vп на вероятности передачи сообще-

ний по категориям и в целом через ВККС в течение Т часов. Провести дисперсионный анализ. Факторы и их уровни выбрать самостоятельно. Результаты моделирования необходимо получить с точностью ε = 0,01 и доверительной вероятностью α = 0,99.

Сделать выводы о загруженности элементов ВККС и необходимых мерах по повышению эффективности его функционирования.

Вариант 9

Постановка задачи

На вычислительный комплекс коммутации сообщений (ВККС) поступают сообщения от n1 абонентов с интервалами времени T1, T2, …, Tn1. Сообщения могут быть n2 категорий с вероятностями p1, p2, …, pn2 (p1+p2+...+pn2 = 1) и вычислительными сложностями S1, S2,... , Sn2 операций (оп) соответственно. Вычислительные сложности случайные. ВККС имеет входной накопитель емкостью L1 байт для хранения сообщений, ожидающих передачи. Сообщения 1-й категории обладают абсолютным приоритетом по отношению к сообщениям остальных категорий при обработке на ВККС. В буфере сообщения размещаются в соответствии с приоритетом.

ВККС обрабатывает сообщения с производительностью Q оп/с. После обработки сообщения передаются по n3 направлениям, каждое из которых имеет k1, k2, …, kn3 каналов связи. Вероятности передачи сообщений по направлениям от любого источника p1, p2, …, pn3 (p1+p2+…+pn3 = 1). Скорости передачи по любому каналу направлений Vп1, Vп2, …, Vпn3 бит/с соответственно.

При передаче сообщения 1-й категории обладают абсолютным приоритетом по отношению к сообщениям других категорий. Поэтому если после обработки сообщения все каналы связи направления заняты, обработанное сообщение помещается в накопитель каналов связи, если в нем есть место, иначе — теряется. Емкость накопителя каналов связи ограничена L2 сообщениями.

ВККС и каналы связи имеют конечную надежность. Интервалы времени Тот1 и Тот2 между отказами ВККС и каналов связи случайные. Длительности восстановления Тв1 и Тв2 ВККС и каналов связи случайные. При отказе канала связи передаваемые сообщения 1-й категории сохраняются в накопителе каналов, если в нем есть место. При выходе из строя ВККС с вероятностью Рс все сообщения в накопителе ВККС и накопителе каналов связи сохра-

няются, обрабатываемое сообщение теряется,а прием ВККС и передача сообщений по каналам связи прекращается. Все поступающие в это время сообщения теряются.

Задание на исследование

Разработать имитационную модель функционирования ВККС. Исследовать влияние емкостей входных накопителей, интервалов времени, вероятностей, Q и Vп на вероятность передачи и среднее время обработки одного сообщения.

Провести дисперсионный анализ. Факторы и их уровни выбрать самостоятельно. Результаты моделирования необходимо получить с точностью $\varepsilon = 0{,}01$ и доверительной вероятностью $\alpha = 0{,}95$. Время моделирования — T часов.

Сделать выводы о загруженности элементов ВККС и необходимых мерах по повышению эффективности его функционирования.

Вариант 10

Постановка задачи

На вычислительный комплекс коммутации сообщений (ВККС) поступают сообщения от n1 абонентов с интервалами времени T1, T2, …, Tn1. Сообщения могут быть n2 категорий с вероятностями p1, p2, …, pn2 (p1+p2+...+pn2 = 1) и вычислительными сложностями S1, S2,... , Sn2 операций (оп) соответственно. Вычислительные сложности случайные. ВККС имеет входной накопитель емкостью L1 байт для хранения сообщений, ожидающих передачи. Сообщения 1-й категории обладают абсолютным приоритетом по отношению к сообщениям остальных категорий при обработке на ВККС. В буфере сообщения размещаются в соответствии с приоритетом.

ВККС обрабатывает сообщения с производительностью Q оп/с. После обработки сообщения передаются по n3 направлениям, каждое из которых имеет k1, k2, …, kn3 каналов связи. Вероятности передачи сообщений по направлениям от любого источника p1, p2, …, pn3 (p1+p2+…+pn3 = 1). Скорости передачи по любому каналу направлений Vп1, Vп2, …, Vпn3 бит/с соответственно.

При передаче сообщения 1-й категории обладают абсолютным приоритетом по отношению к сообщениям других категорий. Поэтому если после обработки сообщения все каналы связи направления заняты, обработанное сообщение помещается в накопитель

направления, если в нем есть место, иначе — теряется. Емкости накопителей направлений связи ограничены L21, L22, …, L2n3 сообщениями соответственно.

ВККС и каналы связи имеют конечную надежность. Интервалы времени Тот1 и Тот2 между отказами ВККС и каналов связи случайные. Длительности восстановления Тв1 и Тв2 ВККС и каналов связи случайные.

При отказе канала связи передаваемые сообщения 1-й категории сохраняются в накопителе каналов, если в нем есть место. При выходе из строя ВККС с вероятностью Рс все сообщения в накопителе ВККС и накопителе каналов связи сохраняются, обрабатываемое сообщение теряется,а прием ВККС и передача сообщений по каналам связи прекращается. Все поступающие в это время сообщения теряются.

Задание на исследование

Разработать имитационную модель функционирования ВККС. Исследовать влияние емкостей входных накопителей, интервалов времени, вероятностей, Q и Vп на вероятность передачи и среднее время обработки одного сообщения. Провести дисперсионный анализ. Факторы и их уровни выбрать самостоятельно. Результаты моделирования необходимо получить с точностью $\varepsilon = 0{,}01$ и доверительной вероятностью $\alpha = 0{,}95$. Время моделирования — Т часов.

Сделать выводы о загруженности элементов ВККС и необходимых мерах по повышению эффективности его функционирования.

Вариант 11

Постановка задачи

На дежурстве находятся n1 средств связи (СС) n2 типов (n21 + n22 + … + n2n2 = n2) в течение n3 часов.

Каждое СС может в любой момент времени выйти из строя. В этом случае его заменяют резервным, причем либо сразу, либо по мере его появления. Тем временем, вышедшее из строя СС ремонтируют, после чего содержат в качестве резервного. Всего количество резервных СС n4.

Ремонт неисправных СС производят n5 мастеров. Время Т1, Т2, …, Тn2 ремонта случайное и зависит от типа СС, но не зависит от

того, какой мастер это СС ремонтирует. Интервалы времени T21, T22, …, T2n2 между отказами находящихся на дежурстве СС случайные.

Прибыль от СС, находящихся на дежурстве, составляет S1 денежных единиц в час. Почасовой убыток при отсутствии на дежурстве одного СС — S2 денежных единиц. Оплата мастера за ремонт неисправного СС S31, S32, …, S3n2 денежных единиц в час. Затраты на содержание одного резервного СС составляют S4 денежных единиц в час.

Задание на исследование

Разработать имитационную модель функционирования системы ремонта СС. Исследовать влияние на ожидаемую прибыль различного количества резервных СС и мастеров. Определить абсолютные величины и относительные коэффициенты ожидаемой прибыли по каждому типу СС и в целом за систему. Результаты моделирования необходимо получить с точностью $\varepsilon = 0{,}01$ и доверительной вероятностью $\alpha = 0{,}99$.

Сделать выводы о загруженности средств связи, мастеров и необходимых мерах по совершенствованию системы ремонта.

Вариант 12

Постановка задачи

На дежурстве находятся n1 средств связи (СС) n2 типов (n21 + n22 + … + n2n2 = n2) в течение n3 часов.

Каждое СС может в любой момент времени выйти из строя. В этом случае его заменяют резервным, причем либо сразу, либо по мере его появления. Тем временем, вышедшее из строя СС ремонтируют, после чего содержат в качестве резервного. Всего количество резервных СС n4.

Ремонт неисправных СС производят n5 мастеров. Время T1, T2, …, Tn2 ремонта случайное и зависит от типа СС, но не зависит от того, какой мастер это СС ремонтирует. Интервалы времени T21, T22, …, T2n2 между отказами находящихся на дежурстве СС случайные.

Прибыль от СС, находящихся на дежурстве, составляет S1 денежных единиц в час. Почасовой убыток при отсутствии на дежурстве одного СС — S2 денежных единиц. Оплата мастера за ремонт неисправного СС S31, S32, …, S3n2 денежных единиц

в час. Затраты на содержание одного резервного СС составляют S4 денежных единиц в час.

Задание на исследование

Разработать имитационную модель функционирования системы ремонта средств связи. Исследовать через промежутки времени ΔT влияние на ожидаемую прибыль различного количества резервных средств связи и мастеров. Определить абсолютные величины и относительные коэффициенты ожидаемой прибыли для каждого промежутка ΔT по каждому типу средств связи и в целом. Результаты моделирования необходимо получить с точностью ε = 0,01 и доверительной вероятностью α = 0,99.

Сделать выводы о загруженности средств связи, мастеров по промежуткам ΔT и необходимых мерах по совершенствованию системы ремонта.

Вариант 13

Постановка задачи

На дежурстве находятся n1 средств связи (СС) n2 типов (n21 + n22 + … + n2n2 = n2) в течение n3 часов.

Каждое СС может в любой момент времени выйти из строя. В этом случае его заменяют резервным, причем либо сразу, либо по мере его появления. Тем временем, вышедшее из строя СС ремонтируют, после чего содержат в качестве резервного. Всего количество резервных СС n4.

Ремонт неисправных СС производят n5 мастеров. Время T1, T2, …, Tn2 ремонта случайное и зависит от типа СС, но не зависит от того, какой мастер это СС ремонтирует. Интервалы времени T21, T22, …, T2n2 между отказами находящихся на дежурстве СС случайные.

С вероятностями p1, p2, …, pn2 неисправные СС n2 типов соответственно не подлежат ремонту. Считается, что уменьшилось число резервных СС, если они были. Через T часов не подлежащие ремонту СС заменяется новыми, стоимость каждого из которых S51, S52, …, S5n2 денежных единиц соответственно.

Прибыль от СС, находящихся на дежурстве, составляет S1 денежных единиц в час. Почасовой убыток при отсутствии на дежурстве одного СС — S2 денежных единиц. Оплата мастера за ремонт неисправного СС S31, S32, …, S3n2 денежных единиц

в час. Затраты на содержание одного резервного СС составляют S4 денежных единиц в час.

Задание на исследование

Разработать имитационную модель функционирования системы ремонта средств связи. Исследовать через промежутки времени ΔТ влияние на ожидаемую прибыль различного количества резервных средств связи и мастеров. Определить абсолютные величины и относительные коэффициенты ожидаемой прибыли для каждого промежутка ΔТ по каждому типу средств связи и в целом. Результаты моделирования необходимо получить с точностью ε = 0,01 и доверительной вероятностью α = 0,99.

Сделать выводы о загруженности средств связи, мастеров по промежуткам ΔТ и необходимых мерах по совершенствованию системы ремонта.

Вариант 14

Постановка задачи

Автоматическая телефонная станция (АТС) обслуживает n1 телефонных аппаратов (ТА) первой категории (ТА1), n2 ТА второй категории (ТА2) и имеет n3 выходов в сеть связи. Интервал времени T1/n1 между звонками с ТА первой категории случайный. Вероятность звонка с i-го ТА первой категории p1i = 1/n1. Вероятность того, что при этом для разговора потребуется внешняя линия связи p2= n3/(n2+n3), соединение с ТА второй категории p3 = n2/(n2+n3). При этом может быть занята любая свободная линия связи, а вероятность звонка на j-й ТА второй категории p4j = 1/n2. Длительность t1 разговора с ТА первой категории случайная. Время тож1 ожидания при занятости ТА или внешних линий связи случайное. Вероятность того, что ТА второй категории не ответит, p5. При этом время тож2 также случайное.

Интервал времени T2/n2 между звонками с ТА второй категории случайный. Вероятность звонка с k-го ТА второй категории p6k = 1/n2. Вероятности того, что при этом для разговора потребуются внешняя линия связи p7 = n3/(n1+n3), соединение с ТА первой категории p8 = n1/(n1+n3). Для разговора может быть занята любая свободная внешняя линия связи, а вероятность звонка на l-й ТА первой категории p9l = 1/n1. Длительность t2 разговора с ТА второй категории случайная. Время тож3 при занятости ТА

или внешних линий связи случайное. Вероятность того, что ТА первой категории не ответит, p10. При этом время tож4 также случайное.

Звонки с ТА первой категории обладают абсолютным приоритетом по отношению к звонкам с ТА второй категории при занятости внешнего выхода в сеть связи. Вследствие этого, если при поступлении заявки на разговор по внешнему выходу с ТА первой категории все внешние выходы будут заняты разговорами также с ТА первой категории, то прерывания не происходит и заявка считается потерянной. Если же некоторые внешние выходы будут заняты разговорами с ТА второй категории, то после tож1 один из этих разговоров прерывается (теряется) и начинается разговор по этому выходу с ТА первой категории.

Задание на исследование

Разработать имитационную модель функционирования АТС. Исследовать зависимость вероятности разговоров с ТА первой и второй категории от интервалов времени, времени разговоров и вероятностей.

Провести дисперсионный анализ, факторы и значения их уровней выбрать самостоятельно, включить в число факторов эксперимента и количество телефонных аппаратов. Результаты моделирования необходимо получить с точностью $\varepsilon = 0{,}01$ и доверительной вероятностью $\alpha = 0{,}99$.

Вариант 15

Постановка задачи

Автоматическая телефонная станция (АТС) обслуживает n1 телефонных аппаратов (ТА) первой категории (ТА1), n2 ТА второй категории (ТА2), n4 ТА третьей категории и имеет n3 выходов в сеть связи.

Интервал времени T1/n1 между звонками с ТА первой категории случайный. Вероятность звонка с i-го ТА первой категории p1i = 1/n1. Вероятность того, что при этом для разговора потребуется внешняя линия связи p2= n3/(n2+n3), соединение с ТА второй категории p3 = n2/(n2+n3). При этом может быть занята любая свободная линия связи, а вероятность звонка на j-й ТА второй категории p4j = 1/n2. Длительность t1 разговора с ТА первой категории случайная. Время tож1 ожидания при занятости ТА или внешних

линий связи случайное. Вероятность того, что ТА второй категории не ответит, p5. При этом время tож2 также случайное.

Интервал времени T2/n2 между звонками с ТА второй категории случайный. Вероятность звонка с k-го ТА второй категории p6k = 1/n2. Вероятности того, что при этом для разговора потребуются внешняя линия связи p7 = n3/(n1+n3), соединение с ТА первой категории p8 = n1/(n1+n3). Для разговора может быть занята любая свободная внешняя линия связи, а вероятность звонка на l-й ТА первой категории p9l = 1/n1. Длительность t2 разговора с ТА второй категории случайная. Время tож3 при занятости ТА или внешних линий связи случайное. Вероятность того, что ТА первой категории не ответит, p10. При этом время tож4 также случайное.

Интервал времени T3/n4 между звонками с ТА третьей категории случайный. Вероятность звонка с k-го ТА третьей категории p11k = 1/n4. Для разговора всегда требуется внешняя линия, так как звонок на ТА либо первой, либо второй категорий. Звонок на любой из этих ТА равновероятен p12k=1/(n1+n2). Длительность t4 разговора с ТА третьей категории случайная. Время tож5 при занятости ТА или внешних линий связи случайное. Вероятность того, что ТА первой или второй категории не ответит, p13. При этом время tож5 также случайное.

Звонки с ТА первой категории обладают абсолютным приоритетом по отношению к звонкам с ТА второй категории при занятости внешнего выхода. Вследствие этого, если при поступлении заявки на разговор по внешнему выходу с ТА первой категории все внешние выходы будут заняты разговорами также с ТА первой категории, то прерывания не происходит и заявка считается потерянной. Если же некоторые внешние выходы будут заняты разговорами с ТА второй или третьей категории, то после tож1 раньше начавшийся из этих разговоров прерывается (теряется) и начинается разговор по этому выходу с ТА первой категории.

Задание на исследование

Разработать имитационную модель функционирования АТС. Исследовать зависимость вероятности разговоров с ТА первой и второй категории от интервалов времени, времени разговоров и вероятностей.

Провести дисперсионный анализ, факторы и значения их уровней выбрать самостоятельно, включить в число факторов экспери-

мента и количество телефонных аппаратов. Результаты моделирования необходимо получить с точностью ε = 0,01 и доверительной вероятностью α = 0,99.

Вариант 16

Постановка задачи

В организации локальная вычислительная сеть (ЛВС) имеет n автоматизированных рабочих места (АРМ) и сервер, соединенных общей шиной. В ЛВС организована распределенная обработка данных. Запросы поступают от АРМ, которые имеют свои базы данных (БД). Поступающие запросы первично обрабатываются на АРМ. С вероятностью P1, P2, …, Pn требующаяся информация обнаруживается в БД АРМ, после чего продолжается дальнейшая обработка запроса. В противном случае необходима посылка запроса на сервер. После посылки запроса на сервер АРМ обрабатывает другие поступающие на него запросы. Получив ответ с сервера, АРМ завершает обработку запроса.

Интервалы времени поступления запросов на АРМ распределены по экспоненциальному закону со средним значением T11, T12, …, T1n.

Канал передачи данных, соединяющий АРМ и сервер, не имеет накопителей и передача по нему возможна только тогда, когда он свободен. Когда он занят, АРМ находится в режиме ожидания и только после освобождения канала передает запрос. При передаче данных с сервера по запросу АРМ этим данным присваивается более высокий приоритет по сравнению с поступающими на АРМ запросами. Этим обеспечивается дисциплина обслуживания «раньше пришел — раньше обслужен» при завершении обработки запроса на АРМ после получения ответа с сервера.

На сервере имеются два накопителя. Накопитель 1 предназначен для поступающих запросов, а второй — для передаваемых ответов на запросы. Емкость накопителя 1 ограничена на Emk1 запросов, то есть поступающие с АРМ запросы могут теряться в случае полного заполнения накопителя 1. Емкость накопителя 2 практически неограниченна, то есть ответы на запросы С АРМ не теряются.

Время первичной обработки запроса, его передачи при надобности на сервер, обработки на сервере и обратной передачи на нужное АРМ подчинены экспоненциальному закону.

Задание на исследование

Разработать имитационную модель. Промоделировать функционирование ЛВС в течение Т часов.

Определить:

рациональную емкость накопителя 1 на сервере, при которой не происходит потерь запросов с АРМ;

вероятность отказа в ответе на запрос с АРМ вследствие полного заполнения накопителя 1 на сервере;

время реакции ЛВС на запрос пользователя;

вероятности обработки запросов, поступающих на АРМ;

загрузку АРМ, канала передачи данных и сервера.

Провести дисперсионный анализ. Факторы и значения уровней факторов выбрать самостоятельно.

Результаты моделирования необходимо получить с точностью $\varepsilon = 0{,}01$ и доверительной вероятностью $\alpha = 0{,}99$.

По полученным результатам моделирования сделать выводы о совершенствовании информационных потоков в организации.

Вариант 17

Постановка задачи

В организации локальная вычислительная сеть (ЛВС) имеет n автоматизированных рабочих места (АРМ) и сервер, соединенных общей шиной. В ЛВС организована распределенная обработка данных. Запросы поступают от АРМ, которые имеют свои базы данных (БД). Поступающие запросы первично обрабатываются на АРМ. С вероятностью P1, P2, …, Pn требующаяся информация обнаруживается в БД АРМ, после чего продолжается дальнейшая обработка запроса. В противном случае необходима посылка запроса на сервер. После посылки запроса на сервер АРМ обрабатывает другие поступающие на него запросы. Получив ответ с сервера, АРМ завершает обработку запроса.

Интервалы времени поступления запросов на АРМ распределены по экспоненциальному закону со средним значением T11, T12, …, T1n.

Канал передачи данных, соединяющий АРМ и сервер, не имеет накопителей и передача по нему возможна только тогда, когда он свободен. Когда он занят, АРМ находится в режиме ожидания и только после освобождения канала передает запрос. При передаче данных с сервера по запросу АРМ этим данным присваивается

более высокий приоритет по сравнению с поступающими на АРМ запросами. Этим обеспечивается дисциплина обслуживания «раньше пришел — раньше обслужен» при завершении обработки запроса на АРМ после получения ответа с сервера.

На сервере имеются два накопителя. Накопитель 1 предназначен для поступающих запросов, а второй – для передаваемых ответов на запросы. Емкость накопителя 1 ограничена на Emk1 запросов, то есть поступающие с АРМ запросы могут теряться в случае полного заполнения накопителя 1. Емкость накопителя 2 практически бесконечна, то есть ответы на запросы не теряются.

Время первичной обработки запроса, его передачи при надобности на сервер, обработки на сервере и обратной передачи на нужное АРМ подчинены экспоненциальному закону.

Задание на исследование

Разработать имитационную модель.

Определить:

среднее время обработки N запросов;

рациональную емкость накопителя 1 на сервере, при которой не происходит потерь запросов с АРМ;

вероятность отказа в ответе на запрос с АРМ вследствие полного заполнения накопителя 1 на сервере;

время реакции ЛВС на запрос пользователя;

вероятности обработки запросов, поступающих на АРМ;

загрузку АРМ, канала передачи данных и сервера.

Провести дисперсионный анализ. Факторы и значения уровней факторов выбрать самостоятельно.

Результаты моделирования необходимо получить с точностью $\varepsilon = 0{,}01$ и доверительной вероятностью $\alpha = 0{,}99$.

По полученным результатам моделирования сделать выводы о совершенствовании информационных потоков в организации.

Вариант 18

Постановка задачи

В организации локальная вычислительная сеть (ЛВС) имеет n автоматизированных рабочих места (АРМ) и сервер, соединенных общей шиной. В ЛВС организована распределенная обработка данных. Запросы поступают от АРМ, которые имеют свои базы данных (БД). Поступающие запросы первично обрабатываются на

АРМ. С вероятностью P1, P2, …, Pn требующаяся информация обнаруживается в БД АРМ, после чего продолжается дальнейшая обработка запроса. В противном случае необходима посылка запроса на сервер. После посылки запроса на сервер АРМ обрабатывает другие поступающие на него запросы. Получив ответ с сервера, АРМ завершает обработку запроса.

Интервалы времени поступления запросов на АРМ распределены по экспоненциальному закону со средним значением T11, T12, …, T1n.

Запросы с АРМ1 имеют абсолютный приоритет по отношению к запросам с остальных АРМ. Поэтому если по каналу передается запрос с других АРМ, то передача прерывается и запрос теряется. Также прерывается обработка запроса с других АРМ на сервере.

Канал передачи данных, соединяющий АРМ и сервер, не имеет накопителей и передача по нему возможна только тогда, когда он свободен. Когда он занят, АРМ находится в режиме ожидания и только после освобождения канала передает запрос. При передаче данных с сервера по запросу АРМ этим данным присваивается более высокий приоритет по сравнению с поступающими на АРМ запросами. Этим обеспечивается дисциплина обслуживания «раньше пришел — раньше обслужен» при завершении обработки запроса на АРМ после получения ответа с сервера.

На сервере имеются два накопителя. Накопитель 1 предназначен для поступающих запросов, а второй — для передаваемых ответов на запросы. Емкость накопителя 1 ограничена на Emk1 запросов, то есть поступающие с АРМ запросы могут теряться в случае полного заполнения накопителя 1. Емкость накопителя 2 практически неограниченна, то есть ответы на запросы С АРМ не теряются.

Время первичной обработки запроса, его передачи при надобности на сервер, обработки на сервере и обратной передачи на нужное АРМ подчинены экспоненциальному закону.

Задание на исследование

Разработать имитационную модель. Промоделировать функционирование ЛВС в течение T часов.

Определить:

рациональную емкость накопителя 1 на сервере, при которой не происходит потерь запросов с АРМ;

вероятность отказа в ответе на запрос с АРМ вследствие полного заполнения накопителя 1 на сервере;

время реакции ЛВС на запрос пользователя;

вероятности обработки запросов, поступающих на АРМ;

загрузку АРМ, канала передачи данных и сервера.

Провести дисперсионный анализ. Факторы и значения уровней факторов выбрать самостоятельно.

Результаты моделирования необходимо получить с точностью $\varepsilon = 0{,}01$ и доверительной вероятностью $\alpha = 0{,}99$.

По полученным результатам моделирования сделать выводы о совершенствовании информационных потоков в организации.

Вариант 19

Постановка задачи

Изготовление в цехе детали начинается через случайное время Тн. Выполнению операций предшествует подготовка. Длительность подготовки зависит от качества заготовки, из которой будет сделана деталь. Всего различных видов заготовок n1.

Время подготовки подчинено экспоненциальному закону. Частота появления различных видов заготовок и средние значения времени их подготовки заданы следующей таблицей дискретного распределения:

Частота	0,05	0,13	0,16	0,22	0,29	0,15
Среднее время	10	14	21	22	28	25

Для изготовления детали последовательно выполняются n операций, продолжительностями T1, T2, …, Tn соответственно. После каждой операции в течение времени Тк1, Тк2, …, Ткn следует контроль. Время контроля — случайное. Контроль не проходят q1, q2, …, qn % деталей соответственно.

Забракованные детали поступают в окончательный блок контроля и проходят в нем проверку в течение случайного времени Ток1, Ток2, …, Токn. В результате из общего количества не прошедших контроль деталей q(n+1) % деталей идут в брак, а оставшиеся 1-q(n+1) % деталей подлежат повторному выполнению тех операций, после которых они не прошли контроль. Если деталь повторно не проходит контроль после повторного выполнения операции, она бракуется.

Задание на исследование

Разработать имитационную модель процесса изготовления деталей. Модель должна позволять определять абсолютное и относительное количество готовых и забракованных деталей, среднее время изготовления одной детали. Исследовать зависимость количества изготовленных деталей от качества выполнения операций.

Провести дисперсионный анализ. Факторы и значения уровней факторов выбрать самостоятельно.

Результаты моделирования необходимо получить с точностью $\varepsilon = 0{,}01$ и доверительной вероятностью $\alpha = 0{,}95$. Время моделирования — T часов.

Сделать выводы о загруженности пунктов выполнения операций и необходимых мерах по повышению количества изготовления деталей.

Вариант 20

Постановка задачи

Изготовление в цехе детали начинается через случайное время Тн. Выполнению операций предшествует подготовка. Длительность подготовки зависит от качества заготовки, из которой будет сделана деталь. Всего различных видов заготовок n1.

Время подготовки подчинено экспоненциальному закону. Частота появления различных видов заготовок и средние значения времени их подготовки заданы следующей таблицей дискретного распределения:

Частота	0,05	0,13	0,16	0,22	0,29	0,15
Среднее время	10	14	21	22	28	25

Для изготовления детали последовательно выполняются n операций, продолжительностями T1, T2, …, Tn соответственно. После каждой операции в течение времени Тк1, Тк2, …, Ткn следует контроль. Время контроля — случайное. Контроль не проходят q1, q2, …, qn % деталей соответственно.

Забракованные детали поступают в окончательный блок контроля и проходят в нем проверку в течение случайного времени Ток1, Ток2, …, Токn. В результате из общего количества не прошедших контроль деталей q(n+1) % деталей идут в брак, а оставшиеся 1-q(n+1) % деталей подлежат повторному выполнению тех операций, после которых они не прошли контроль. Если деталь

повторно не проходит контроль после повторного выполнения операции, она бракуется.

Задание на исследование

Разработать имитационную модель процесса изготовления деталей. Модель должна позволять определять абсолютное и относительное количество готовых и забракованных деталей, среднее время изготовления одной детали. Исследовать зависимость времени изготовления N деталей от качества выполнения операций.

Провести дисперсионный анализ. Факторы и значения уровней факторов выбрать самостоятельно.

Результаты моделирования необходимо получить с точностью $\varepsilon = 0{,}1$ и доверительной вероятностью $\alpha = 0,99$.

Сделать выводы о загруженности пунктов выполнения операций и необходимых мерах по сокращению времени изготовления деталей.

Вариант 21

Постановка задачи

В ремонтное подразделение средств связи (СС) поступают неисправные СС n типов с вероятностями p1, p2, …, pn соответственно. Интервалы времени Тп между двумя очередными поступлениями случайные. Каждое СС любого типа может требовать одного из трех видов ремонта с вероятностями p11, p21 или p31 соответственно.

В ремонтном подразделении имеются n1, n2, …, nn групп мастеров для ремонта СС каждого типа соответственно.

Мастера группы n1 ремонтируют СС первого типа. Если их нет и мастера n2, …, nn групп заняты, они ремонтируют СС этих типов. При этом поступающие СС первого типа ожидают их освобождения.

Мастера группы n2 ремонтируют СС второго типа. Если их нет и мастера n3, n4, …, nn групп заняты, они ремонтируют СС этих типов. При этом поступающие СС второго типа ожидают их освобождения. Аналогичные обязанности и у мастеров остальных групп. Лишь мастера последней группы nn ремонтируют СС только одного n-го типа.

Время ремонта n-го типа СС случайное, не зависит от мастера, а зависит только от вида ремонта: T11, T12, T13 – для СС первого

типа, T21, T22, T23 – для СС второго типа, …, Tn1, Tn2, …, Tnn – для СС n-го типа.

Прием и распределение неисправных СС между мастерами осуществляется диспетчером. Время, затрачиваемое диспетчером на одно СС, случайное. Диспетчером не допускается к ремонту q % СС всех типов.

Задание на исследование

Разработать имитационную модель функционирования ремонтного подразделения. Исследовать зависимость времени и вероятностей выполнения ремонта СС первого и второго типов от интервала и вероятностей поступления их в ремонт.

Провести дисперсионный анализ. Факторы и значения уровней факторов выбрать самостоятельно.

Результаты моделирования необходимо получить с точностью $\varepsilon = 0{,}01$ и доверительной вероятностью $\alpha = 0, 95$.

Сделать выводы о загруженности каждой группы мастеров и необходимых мерах по повышению эффективности работы ремонтного подразделения.

Вариант 22

Постановка задачи

В ремонтное подразделение средств связи (СС) поступают неисправные СС n типов с вероятностями p1, p2, …, pn соответственно. Интервалы времени Tп между двумя очередными поступлениями случайные. Каждое СС любого типа может требовать одного из трех видов ремонта с вероятностями p11, p21 или p31 соответственно.

В ремонтном подразделении имеются n1, n2, …, nn групп мастеров для ремонта СС каждого типа соответственно. Мастера группы n1 ремонтируют СС первого типа. Если их нет и мастера n2, …, nn групп заняты, они ремонтируют СС этих типов. При этом поступающие СС первого типа ожидают их освобождения. Мастера группы n2 ремонтируют СС второго типа. Если их нет и мастера n3, n4, …, nn групп заняты, они ремонтируют СС этих типов. При этом поступающие СС второго типа ожидают их освобождения. Аналогичные обязанности и у мастеров остальных групп. Лишь мастера последней группы nn ремонтируют СС только одного n-го типа.

Время ремонта n-го типа СС случайное, не зависит от мастера, а зависит только от вида ремонта: Т11, Т12, Т13 – для СС первого типа, Т21, Т22, Т23 – для СС второго типа, …, Tn1, Tn2, …, Tnn – для СС n-го типа.

Прием и распределение неисправных СС между мастерами осуществляется диспетчером. Время, затрачиваемое диспетчером на одно СС, случайное. Диспетчером не допускается к ремонту q % СС всех типов.

Задание на исследование

Разработать имитационную модель функционирования ремонтного подразделения. Исследовать зависимость времени и вероятностей выполнения ремонта N СС всех типов от интервала и вероятностей поступления их в ремонт.

Провести дисперсионный анализ. Факторы и значения уровней факторов выбрать самостоятельно.

Результаты моделирования необходимо получить с точностью $\varepsilon = 0{,}01$ и доверительной вероятностью $\alpha = 0,95$.

Сделать выводы о загруженности каждой группы мастеров и необходимых мерах по повышению эффективности работы ремонтного подразделения.

Вариант 23

Постановка задачи

Сеть передачи данных (вариант на рис. 4.1) представляет собой СМО вида: многофазная, многоканальная, с несколькими неоднородными потоками заявок на обслуживание, разомкнутая, конечной надежности, с очередями ограниченной емкости на отдельных фазах обслуживания. Внешняя часть сети (ВС) организуется узлами типа К2 и К3. Каждый узел К2 вместе с узлами типа К1 образуют ЛВС. Источники информации находятся на узлах К1.

Сеть имеет следующие характеристики:

количество элементов сети различных типов;

количество источников и видов сообщений;

средние интервалы времени поступления запросов на передачу сообщений всех видов от всех источников;

средние интервалы времени между отказами элементов сети;

среднее время восстановления элементов сети;

емкости накопителей (буферов) элементов сети;

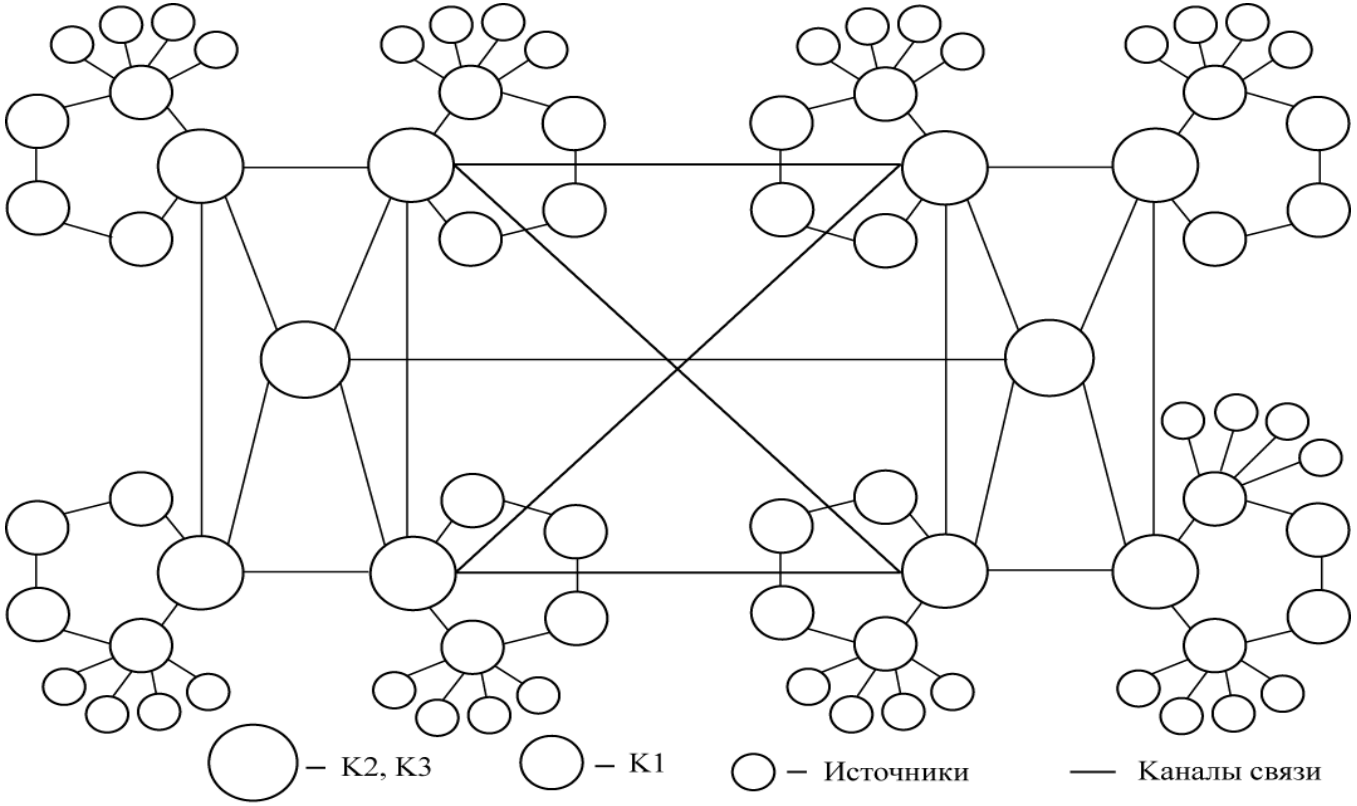


Рис. 4.1. Вариант структуры сети передачи данных

средние длины сообщений всех видов от всех источников;

длины линий связи между узлами сети;

скорости обработки сообщений элементами сети;

маршруты передачи сообщений.

Необходимо разработать методику оценки качества обслуживания сети по следующим показателям:

$P_{\text{пр}} = \frac{S_{\text{пол}}}{S_{\text{отп}}}$ — вероятность пропускной способности сети, $S_{\text{пол}}$ — количество полученных адресатами сообщений, $S_{\text{отп}}$ — количество отправленных сообщений (поступивших на входы сети);

$t_{\text{ср}} = \frac{T_{\text{пер}}}{S_{\text{пол}}}$ — среднее время передачи сообщения, $T_{\text{пер}}$ — суммарное время передачи $S_{\text{пол}}$ сообщений;

$P_{\text{пот}} = 1 - P_{\text{пр}}$ — вероятность потерь сообщений.

Методика оценки должна состоять из двух частей. Первая часть — интерфейс ко второй части — имитационной модели в среде GPSS World.

Задание 1 на исследование

Разработать интерфейс с использованием системы визуального программирования Delphi или Си++. Он предназначен для:

построения сети в виде графа;

ввода характеристик сети;

нахождения кратчайших путей между узлами сети по методу Дийкстры или по методу Флойда;

разбиения сети на элементы, необходимые для имитационного моделирования;

порядковой нумерации элементов сети;

составления матриц кратчайших маршрутов передачи сообщений в номерах элементов сети;

преобразования исходных данных в форму, «понятную» GPSS World;

запуска имитационной модели и вывода результатов моделирования;

сравнительной оценки средних времен передачи сообщений, полученных по методу Дийкстры или Флойда, со средними временами, полученными по результатам моделирования.

Задание 2 на исследование

Разработать в среде GPSS World вторую часть методики — имитационную модель функционирования сети передачи данных, алгоритм которой приведен на рис. 4.2.

В блоке 1 вводятся исходные данные. Блок 2 предназначен для имитации источников сообщений. Принято, что среднее время интервалов поступления сообщений для всех источников одинаковое. Интервалы же поступления сообщений от одного источника распределены по экспоненциальному закону.

В блоке 3 по заданному количеству узлов, источников информации и каналов связи, согласно которому проведена поэлементная декомпозиция сети и каждому элементу присвоен порядковый номер, разыгрывается адрес отправителя. Здесь же разыгрывается и источник (вид информации). Данные адреса отправителя заносятся в соответствующие параметры транзакта-сообщения.

В блоке 4 по экспоненциальному закону разыгрывается длина сообщения и также заносится в параметр транзакта-сообщения.

Адресные данные получателя разыгрываются в блоке 5 и запоминаются в параметре транзакта-сообщения так же, как и адресные данные отправителя в блоке 3.

В блоке 6 сравниваются данные адресов отправителя и получателя. По результату сравнения определяется нахождение получателя. Если получатель находится на К1, входящим в ЛВС отправителя, то в параметр транзакта-сообщения заносится признак передачи по ЛВС (блок 7). В противном случае — в блоке 17 признак передачи по внешней сети.

После этого согласно занесенному признаку в блоке 8 или блоке 18 сообщение делится на пакеты, то есть либо из блока 8, либо из блока 18 выходят транзакты, количество которых равно количеству пакетов в транзакте-сообщении. Каждый транзакт-пакет имеет те же значения параметров, что и породившее их транзакт-сообщение.

Транзакты-пакеты из блока 8 поступают на блок 9, а из блока 18 — на блок 19.

Предположим, что сообщение должно быть передано по ЛВС. Тогда транзакт-сообщение разделено на транзакты-пакеты в блоке 8 и они поступают на блок 9. На этот же блок 9 поступают транзакты-пакеты из ВС, которые также должны быть переданы по данной ЛВС К1-получателю.

Рис. 4.2. Алгоритм имитационной модели (начало, лист 1)

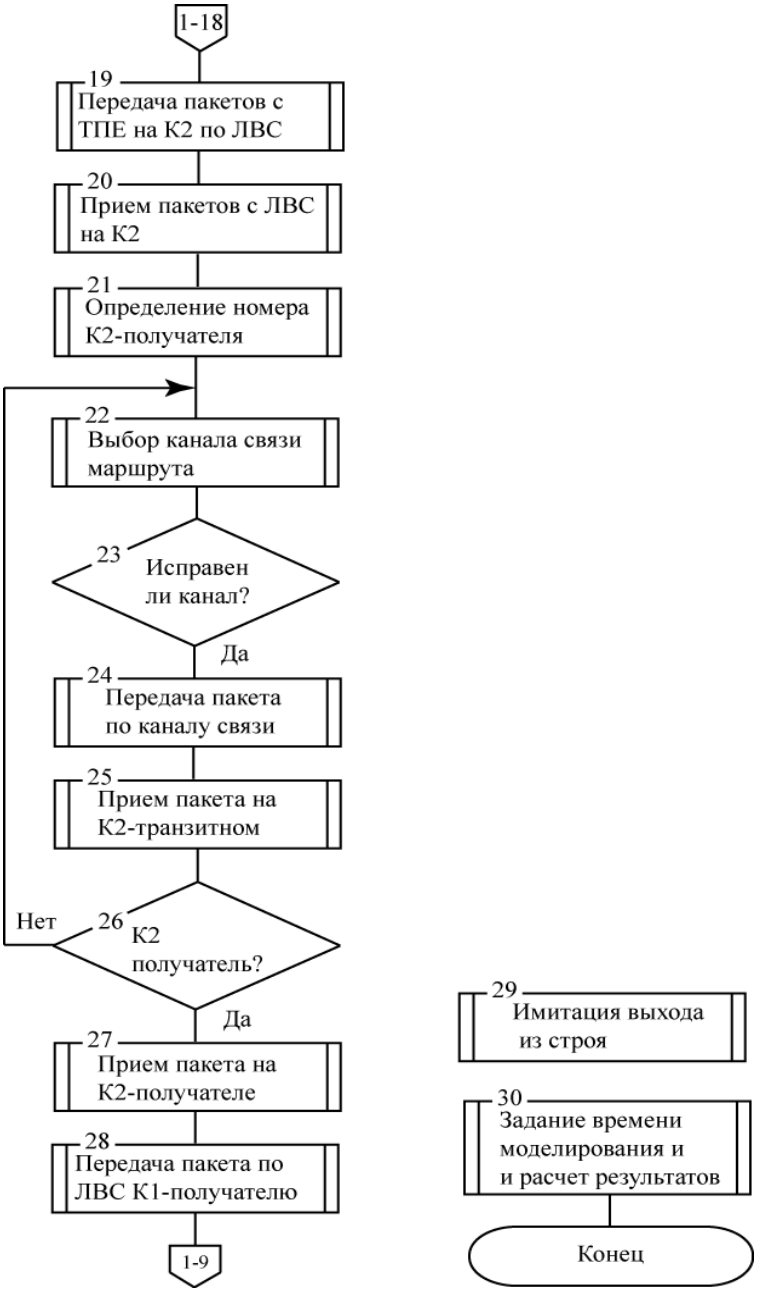


Рис. 4.2. Алгоритм имитационной модели (окончание, лист 2)

В блоке 9 происходит разделение входного потока транзактов-пакетов на два потока: поток транзактов-пакетов из ВС и поток транзактов-пакетов от К1-отправителей данной ЛВС.

Пусть транзакта-пакет поступил из ВС (на К2 данной ЛВС). Блоком 9 он передается в блок 10, в котором ставится в очередь на передачу, так как общая шина может быть занята. Как только общая шина становится свободной, транзакта-пакет поступает в блок 11, имитирующий передачу транзакта-пакета из ВС на К1-получатель.

В блоке 12 имитируется прием транзактов-пакетов К1-получателем, а в блоке 13 — сбор («сшивание») транзактов-пакетов одного сообщения.

После «сшивания» уже транзакт-сообщение поступает в блок 14, который осуществляет сбор статистики о переданных, то есть дошедших до получателей сообщениях, и вывод транзактов-сообщений из модели.

Вернемся к блоку 9. Теперь предположим, что транзакт-пакет, поступил не из ВС, а должен быть передан с какого-либо К1 данной ЛВС также на К1 этой же ЛВС. Тогда он передается в блок 15, который осуществляет постановку в очередь на К1-отправителе. Затем, когда наступает очередь этого транзакта-пакета, он переходит в блок 16, имитирующий передачу по ЛВС. Далее транзакт-пакет с К1 обрабатывается блоками 12…14 аналогично транзакту-пакету из ВС.

Возвратимся к блоку 18, когда сообщение с К1-отправителя данной ЛВС должно быть передано по ВС К1-получателю другой ЛВС.

Транзакт-пакет из блока 18 поступает в блок 19, который имитирует передачу с К1-отправителя на К2 данной ЛВС.

В блоке 20 имитируется прием транзакта-пакета, а в блоке 21 — определяется номер К2-получателя ВС.

Блоки 22…26 имитируют передачу транзакта-пакета по ВС.

Сначала в блоке 22 согласно таблице маршрутизации выбирается канал связи для передачи транзакта-пакета следующему К2 маршрута.

Затем в блоке 23 проверяется исправность этого канала, то есть возможность передачи по нему. При отсутствии такой возможности передача не производится до тех пор, пока работоспособность канала не будет восстановлена.

Если канал связи работоспособен, транзакт-пакет последовательно проходит блоки 24 и 25. Первый из них имитирует передачу по каналу связи, а второй — прием на К2-транзитном. Здесь же анализируется емкость входного буфера К2-транзитного. Если буфер заполнен, то транзакт-пакет теряется.

Однако К2-транзитный может оказаться К2-получателем. Проверка этого производится в блоке 26. Если К2-транзитный не К2-получатель, то транзакт-пакет передается в блок 22 и процесс имитации передачи к следующему К2 маршрута повторяется.

Если же К2-транзитный окажется К2-получателем, то транзакт-пакет передается сначала в блок 27, имитирующий его прием на К2-получателе, а затем в блок 28 для передачи по ЛВС К1-получателю, то есть в блок 9.

Блок 29 имитирует выход из строя каналов связи. Принято, что интервалы времени между отказами и время восстановления канала связи распределены по экспоненциальному закону. При функционировании абонентской сети в дуплексном режиме возможна имитация одновременного выхода из строя и восстановления прямого и обратного каналов, а также неодновременный отказ и восстановление работоспособности прямого и обратного каналов. Транзакт-пакет, находящийся в канале в момент выхода его из строя, теряется.

Блок 30 задает условия окончания моделирования, обрабатывает накопленные статистические данные для получения результатов моделирования.

Замечание. Задания 1 и 2 выполняются в комплексе двумя исполнителями. Исполнитель первого задания в пояснительной записке представляет свои разработки, а второй исполнитель, используя их, представляет окончательные результаты методики — показатели оценки качества обслуживания сети передачи данных.

Задания 3 и 4 на исследование

На платформе AnyLogic разработать типовые объекты моделирования процессов сети передачи данных (рис. 4.1) согласно алгоритму, представленному на рис. 4.2. В качестве объектов могут быть ЛВС и маршрутизатор, из которых и может создаваться имитационная модель функционирования сети.

Замечание. Задания 3 и 4 выполняются в комплексе двумя исполнителями. Каждый исполнитель представляет свои разработки.

Заключение

При переходе от традиционных телекоммуникационных сетей к мультисервисным сетям связи следующего поколения (NGN — Next Generation NetWorks), их проектировании сталкиваются с рядом проблем: определение типов и параметров сетевого оборудования, выбор архитектуры сети. Среди множества технологических, методологических и других проблем центральное место занимает проблема обеспечения требуемого качества обслуживания (QoS — Quality of Service) для различных видов трафика, определения загруженности каналов связи сети при заданных её параметрах, а также исследование поведения трафика разных классов.

Качество обслуживания характеризуют следующие показатели:

вероятность (коэффициент) пропускной способности сети;

среднее время передачи (задержки) сообщений (пакетов);

вариация задержки;

вероятность потери сообщений (пакетов).

В современной сети одновременно передается информация разных видов (видео- и аудиоинформация, сжатая видео- и аудиоинформация, а также данные, менее чувствительные к задержкам) с разными показателями качества обслуживания, на ее работу существенно влияют методы управления трафиком. Для создания модели сети (в том числе задания её топологии, характеристик элементов сети), а также динамического моделирования её работы, анализа и оптимизации её характеристик, управления трафиком, несомненно, нужно использовать один из мощнейших инструментов исследования сложных систем — имитационное моделирование.

В настоящее время известен ряд специализированных программных продуктов (COMNET III, BONeS DESIGNER, OPNET Modeler, ns2 и др.), предназначенных непосредственно для моделирования сетей передачи данных. Однако по разным причинам они не нашли достаточно широкого распространения в нашей стране. В том числе и из-за высокой стоимости, не оправдывающей экономически решение задачи оценки качества обслуживания той или иной сети.

В упомянутых программных продуктах сеть передачи данных рассматривается как сеть массового обслуживания, а для моделирования используется дискретно-событийный метод. Этот же ме-

тод реализован и в известных системах моделирования общецелевого назначения Anylogic и GPSS World.

В настоящем пособии, в том числе и по этим причинам, основное внимание и было уделено методикам разработки имитационных моделей и проведения исследований с применением систем моделирования GPSS World и AnyLogic. Методики разработки имитационных моделей в AnyLogic применительно к данной дисциплине изложены впервые.

Рекомендованы основные направления возможных тем курсовых и дипломных проектов. В связи с методологической и практической необходимостью одним из актуальных направлений представляется создание программного комплекса, предназначенного для имитации работы и проектирования современных мультисервисных сетей связи военного назначения. Комплекс должен иметь базу данных сетевых устройств, позволяющую строить сеть нужной топологии путем выбора из них необходимых моделей, с нужными характеристиками. После этого комплекс должен предоставлять возможность имитации работы сети, построенной из реальных элементов. Основой для создания имитационной составляющей может послужить вариант 23 задания на проектирование.

Комплекс может быть создан на базе GPSS World с примененим визуальных объектно-ориентированных систем программирования, на платформе AnyLogic или других систем моделирования и программирования.

В рамках разработки программного комплекса проектирования современных сетей связи обучаемым рекомендуется с помощью преподавателей и самостоятельно ставить и разрабатывать курсовые и дипломные проекты, давать оценки результатов имитационного моделирования, полученных в разных системах моделирования.

Список литературы

1. Боев В. Д., Сыпченко Р. П. Компьютерное моделирование. Элементы теории и практики: Учеб. пособие. — СПб.: ВАС, 2009.
2. Боев В. Д. Моделирование систем. Инструментальные средства GPSS World: Учеб. пособие. — СПб.: БХВ-Петербург, 2004.
3. Боев В. Д., Кирик Д. И., Ушкань А. О. Методика поддержки руководства курсовым проектированием по дисциплине «Моделирование»: Статья — В сб. докладов Третьей Всероссийской конференции «Имитационное моделирование. Теория и практика» ИММОД-2007 — СПб.: ФГУП ЦНИИТС, 2007.
4. Боев В. Д., Ушкань А. О. Методика оценки качества обслуживания сети передачи данных: Статья — В сб. докладов Четвертой Всероссийской конференции «Имитационное моделирование. Теория и практика» ИММОД-2009 — СПб.: ЦТСиР, 2009.
5. Боев В. Д., Ушкань А. О. Вторичные модели оценки качества обслуживания сети передачи данных: Статья — В сб. докладов Четвертой Всероссийской конференции «Имитационное моделирование. Теория и практика» ИММОД-2009 — СПб.: ЦТСиР, 2009.
6. Боев В. Д., Сыпченко Р. П. Компьютерное моделирование: Руководство по курсовому проектированию: Учеб пособие — СПб.: ВУС, 2002.
7. Боев В. Д., Сыпченко Р. П. Компьютерное моделирование: Учебный курс. — ИНТУИТ.РУ, 2010.
8. Советов Б. Я., Яковлев С. А. Моделирование систем: Курсовое проектирование. — М.: Высшая школа, 1988.
9. «Экс Джей Текнолоджис» www.xjtek.ru.