621.4
Б759

В. Д. Боев

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ

Пособие для практических занятий, курсового и дипломного проектирования в AnyLogic 7

Санкт-Петербург
2014

УДК 681.142.1.001.57
681.142.33

Боев В. Д.

Компьютерное моделирование: Пособие для практических занятий, курсового и дипломного проектирования в AnyLogic7:. — СПб.: ВАС, 2014. — 432 с.

Пособие предназначено для проведения практических занятий, курсового и дипломного проектирования по дисциплинам «Проектирование и моделирование систем», «Моделирование», «Компьютерное моделирование» с использованием систем имитационного моделирования. Предлагаются методики разработки имитационных моделей проектируемых систем с применением инструментальных средств AnyLogic7. Приводятся сравнительные оценки результатов моделирования разнородных дискретных процессов, полученных на моделях одной и той же проектируемой системы в GPSS World и AnyLogic7 и 6. Интерпретируются результаты моделирования. Даются темы и основные направления курсового и дипломного проектирования.

Для студентов, аспирантов, преподавателей и научных работников.

© Боев В. Д., 2014

ОГЛАВЛЕНИЕ

Введение **8**
Глава 1. Модель обработки запросов сервером **10**
1.1. Постановка задачи 10
1.2. Создание диаграммы процесса 10
1.3. Изменение свойств блоков модели, её настройка и запуск 14
1.3.1. Изменение свойств блоков диаграммы процесса 14
1.3.2. Настройка запуска модели 18
1.3.3. Запуск модели 19
1.4. Создание анимации модели 24
1.5. Сбор статистики использования ресурсов 29
1.6. Уточнение модели согласно ёмкости входного буфера 34
1.7. Сбор статистики по показателям обработки запросов 37
1.7.1. Создание нестандартного Java класса 39
1.7.2. Добавление элементов статистики 45
1.7.3. Изменение свойств объектов диаграммы 46
1.7.4. Удаление и добавление новых полей типа заявок 50
1.8. Добавление параметров и элементов управления 51
1.9. Добавление гистограмм 56
1.10. Изменение времени обработки запросов сервером 57
1.11. Интерпретация результатов моделирования 60
Глава 2. Модель процесса изготовления в цехе деталей **65**
2.1. Постановка задачи 65
2.1.1. Исходные данные 65
2.1.2. Задание на исследование 66
2.1.3. Уяснение задачи на исследование 66
2.2. Модель в AnyLogic 67
2.2.1. Исходные данные. Использование массивов 67
2.2.2. Построение событийной части модели 71
2.2.2.1. Подготовка заготовки 71
2.2.2.2. Сегменты Операция 1, Операция 2, Операция 3 75
2.2.2.3. Создание нового активного объекта 78
2.2.2.4. Создание экземпляра нового типа агента 79
2.2.2.5. Создание области просмотра 82
2.2.2.6. Переключение между областями просмотра 83
2.2.2.7. Пункт окончательного контроля 85
2.2.2.8. Склад готовых деталей. Вывод результатов моделирования 86
2.2.2.9. Склад бракованных деталей.
Вывод результатов моделирования 87
2.2.2.10. Создание областей просмотра и переключение между ними 87
2.2.3. Добавление элементов для проведения исследований 88
2.3. Интерпретация результатов моделирования 91
Глава 3. Модель функционирования направления связи **96**
3.1. Постановка задачи 96
3.2. Уяснение задачи на разработку модели 96
3.3. Модель направления связи в AnyLogic 97
3.3.1. Исходные данные 97
3.3.2. Вывод результатов моделирования 98

3.3.3. Построение событийной части модели ... 99
3.3.3.1. Источники сообщений ... 100
3.3.3.2. Буфер, основной и резервный каналы ... 101
3.3.3.3. Имитатор отказов основного канала связи ... 105
3.4. Отладка модели ... 107
3.5. Интерпретация результатов моделирования ... 112
Глава 4. Модель функционирования сети связи ... 115
4.1. Модель в AnyLogic ... 115
4.1.1. Постановка задачи ... 115
4.1.2. Исходные данные ... 116
4.1.3. Задание на исследование ... 116
4.1.4. Формализованное описание модели ... 116
4.1.5. Создание новых типов агентов ... 121
4.1.6. Создание областей просмотра ... 122
4.1.7. Сегмент Абонент ... 122
4.1.7.1. Исходные данные ... 122
4.1.7.2. Результаты моделирования по каждому абоненту ... 125
4.1.7.3. Показатели качества обслуживания сети связи ... 128
4.1.7.4. Построение событийной части сегмента ... 131
4.1.8. Сегмент Маршрутизатор ... 143
4.1.8.1. Исходные данные ... 143
4.1.8.2. Событийная часть сегмента Маршрутизатор ... 144
4.1.8.2.1. Блок контроля 1 ... 144
4.1.8.2.2. Блок Буфер 1 ... 147
4.1.8.2.3. Блок обработки сообщений ... 148
4.1.8.2.4. Блок контроля 2 ... 149
4.1.8.2.5. Блок Буфер 2 ... 152
4.1.8.2.6. Организация входных и выходных портов ... 153
4.1.8.2.7. Имитатор отказов вычислительного комплекса ... 154
4.1.9. Сегмент Канал ... 155
4.1.9.1. Исходные данные ... 155
4.1.9.2. Событийная часть сегмента Каналы ... 155
4.1.9.3. Организация входного и выходного портов ... 157
4.1.9.4. Имитатор отказов каналов связи ... 158
4.1.10. Построение модели сети связи ... 159
4.1.11. Переключение между областями просмотра ... 171
4.1.12. Запуск и отладка модели ... 174
4.2. Интерпретация результатов моделирования ... 177
ГЛАВА 5. Модель функционирования системы связи ... 180
5.1. Модель в AnyLogic ... 180
5.1.1. Постановка задачи ... 180
5.1.2. Задание на исследование ... 180
5.1.3. Формализованное описание модели ... 180
5.1.4. Сегмент Постановка на дежурство ... 184
5.1.4.1. Ввод исходных данных ... 184
5.1.4.2. Имитация поступления средств связи ... 186
5.1.4.3. Распределитель средств связи ... 191
5.1.4.4. Создание нового активного объекта ... 192

5.1.4.5. Создание экземпляра нового типа агента 194
5.1.5. Сегмент Имитация дежурства 195
5.1.5.1. Ввод исходных данных 195
5.1.5.2. Вывод результатов моделирования 197
5.1.5.3. Событийная часть сегмента Имитация дежурства 199
5.1.6. Сегмент Статистика 202
5.1.6.1. Использование элемента Текстовое поле 204
5.1.6.2. Использование элемента Диаграмма 205
5.1.7. Использование способа Событие 207
5.1.8. Переключение между областями просмотра 211
5.1.9. Отладка модели 212
5.1.10. Проведение экспериментов 217
5.1.10.1. Простой эксперимент 217
5.1.10.2. Связывание параметров 217
5.1.10.3. Первый эксперимент Оптимизация стохастических моделей 220
5.1.10.4. Изменение порядка отображения параметров на странице свойств своего объекта 226
5.1.10.5. Второй эксперимент Оптимизация стохастических моделей 228
5.1.10.6. Эксперимент Варьирование параметров 230
5.2. Интерпретация результатов моделирования 235
Глава 6. Модель функционирования Предприятия 239
6.1. Постановка задачи 239
6.1.1. Исходные данные 240
6.1.2. Задание на исследование 240
6.1.3. Уяснение задачи на исследование 240
6.2. Модель в AnyLogic 242
6.2.1. Формализованное описание 242
6.2.2. Ввод исходных данных 246
6.2.3. Вывод результатов моделирования 249
6.2.4. Построение событийной части модели 251
6.2.4.1. Имитация работы цехов предприятия 253
6.2.4.2. Имитация работы постов контроля блоков 256
6.2.4.3. Имитация работы пунктов сборки изделий 263
6.2.4.4. Имитация работы стендов контроля изделий 274
6.2.4.5. Имитация работы пунктов приёма изделий 276
6.2.4.6. Имитация склада готовых изделий 277
6.2.4.7. Имитация склада бракованных блоков 278
6.2.4.8. Организация перек между областями просмотра 279
6.3. Интерпретация результатов моделирования 283
Глава 7. Модель функционирования терминала 286
7.1. Постановка задачи 286
7.2. Модель в AnyLogic 288
7.2.1. Исходные данные и результаты моделирования 288
7.2.2. Событийная часть модели 291
7.2.3. Результаты моделирования 298
7.3. Эксперименты 298

7.3.1. Первый оптимизационный эксперимент в AnyLogic 299
7.3.2. Второй оптимизационный эксперимент в AnyLogic 302
7.4. Интерпретация результатов экспериментов 303
ГЛАВА 8. Модель предоставления ремонтных услуг 306
8.1. Постановка задачи 306
8.1.1. Исходные данные 306
8.1.2. Задание на исследование 307
8.1.3. Формализованное описание модели 307
8.2. Модель в AnyLogic 309
8.2.1. Ввод исходных данных 309
8.2.2. Вывод результатов моделирования 311
8.2.3. Построение событийной части модели 311
8.2.3.1. Сегмент Источники заявок 312
8.2.3.2. Сегмент Диспетчеры 314
8.2.3.3. Сегмент Мастера 318
8.2.3.4. Сегмент Учёт выполненных заявок 321
8.2.3.5. Отладка модели 323
8.3. Интерпретация результатов моделирования 325
Глава 9. Модель функционирования системы воздушных перевозок 329
9.1. Модель в AnyLogic 329
9.1.1. Постановка задачи 329
9.1.2. Исходные данные 329
9.1.3. Задание на исследование 330
9.1.4. Формализованное описание модели 331
9.1.5. Создание областей просмотра 334
9.1.6. Ввод исходных данных 334
9.1.7. Вывод результатов моделирования 336
9.1.8. Имитация функционирования аэропорта 1 340
9.1.8.1. Прибытие самолётов в аэропорт 1. Ожидание погрузки 340
9.1.8.2. Поступление и учёт контейнеров в аэропорту 1 344
9.1.8.3. Погрузка контейнеров в аэропорту 1 346
9.1.8.4. Полёт из аэропорта 1 в аэропорт 2 350
9.1.8.5. Ожидание разгрузки в аэропорту 1 352
9.1.8.6. Разгрузка самолётов в аэропорту 1 354
9.1.9. Имитация функционирования аэропорта 2 358
9.1.9.1. Поступление и учёт контейнеров в аэропорту 2 358
9.1.9.2. Ожидание разгрузки в аэропорту 2 360
9.1.9.3. Разгрузка самолётов в аэропорту 2 361
9.1.9.4. Ожидание погрузки в аэропорту 2 365
9.1.9.5. Погрузка контейнеров в аэропорту 2 367
9.1.9.6. Полёт из аэропорта 2 в аэропорт 1 370
9.1.9.7. Вывод результатов моделирования с использованием способа Событие 371
9.1.10. Запуск и отладка модели 372
Глава 10. Модель обработки документов 374
в организации 374
10.1. Постановка задачи 374
10.2. Аналитическое решение задачи 374

10.3. Решение задачи в AnyLogic 375
10.4. Решение задачи в GPSS World..... 378
Глава 11. Решение обратных задач в AnyLogic 380
11.1. Определение среднего времени обработки
группы запросов сервером 380
11.2. Определение среднего времени изготовления деталей 385
Глава 12. Задания на проектирование 389
Заключение 422
Список литературы 423
Приложение 1..... 425
Приложение 2..... 429
Приложение 3..... 432

ВВЕДЕНИЕ

Пособие предназначено для проведения практических занятий, курсового и дипломного проектирования по дисциплинам «Проектирование и моделирование систем», «Моделирование», «Компьютерное моделирование», «Моделирование сложных систем и процессов» с использованием систем имитационного моделирования (ИМ).

При ИМ дискретных процессов в современной практике в качестве инструментального средства получила широкое распространение система общецелевого назначения GPSS World, являющаяся последним современным представителем семейства языков моделирования GPSS.

В последние годы наряду с ней применяется система моделирования AnyLogic (новая версия 7). AnyLogic разработана компанией «The AnyLogic Company» на основе современных концепций в области информационных технологий и результатов исследований в теории гибридных систем и объектно-ориентированного моделирования. Это комплексный инструмент, охватывающий в одной модели основные в настоящее время направления моделирования: дискретно-событийное, системной динамики, агентное. Многоподходность не характерна для существующих систем моделирования. Агентные модели не позволяет создавать ни одна из известных систем моделирования, в том числе и GPSS World.

Изложение материала ведётся согласно подходу, который можно назвать «обучение на примерах» или «делай как мы».

Последнее обстоятельство привело автора к необходимости детально изложить методики построения моделей, проведения экспериментов, интерпретации полученных результатов, что, несомненно, будет способствовать качественному проведению практических занятий.

В работе с целью получения обучаемыми всесторонних знаний рассматривается моделирование процессов в разнородных системах, но с использованием дискретно-событийного подхода. Модели демонстрируют достижение и учебной цели, и цели исследования. Возможна и практическая их пригодность. Для проверки качества приобретённых знаний приводятся темы проектирования.

Новая версия AnyLogic 7 позволяет использовать модели, разработанные в среде AnyLogic 6, но модернизация их в какой-то части невозможна. Это, а также освоение новых информационных

технологий и побудило автора к полной переработке пособий [8, 13]. Результаты моделирования в AnyLogic 6 и GPSS World оставлены для сравнительной оценки с результатами в AnyLogic 7.

Первая глава на примере имитационной модели обработки запросов сервером посвящена первичному знакомству с AnyLogic 7 и приёмам работы с ней. Уделено также внимание приёмам разработки презентаций.

Во второй — девятой главах излагаются методики разработки ИМ из различных предметных областей. В конце каждой главы интерпретируются результаты экспериментов с моделями, сравниваются результаты моделирования в AnyLogic 7, 6 и GPSS World. Детальные методики разработки моделей в GPSS World не приводятся. При необходимости их можно найти в [13, 20]. При изложении этих глав автор посчитал более важным сосредоточить внимание именно на разработках ИМ без презентаций, полагая, что, ознакомившись в первой главе с некоторыми из приёмов построения презентаций, пользователи смогут сделать их самостоятельно.

В десятой главе на примере одной из задач теории массового обслуживания сравниваются результаты аналитического решения и решений в системах имитационного моделирования.

В одиннадцатой главе даются методики решения обратных задач в среде AnyLogic 7.

Двенадцатая глава содержит возможные темы и направления курсового и дипломного проектирования.

В приложении даны условные обозначения объектов **Библиотеки моделирования процессов** и элементов палитр AnyLogic7.

Большое спасибо сотрудникам фирмы «The AnyLogic Company» П. А. Лебедеву, С. А. Суслову за плодотворное сотрудничество и рекомендации по построению моделей, а также руководству фирмы за предоставленную версию AnyLogic 7.

За допечатную подготовку автор благодарен Д. В. Боеву.

Автор

ГЛАВА 1. МОДЕЛЬ ОБРАБОТКИ ЗАПРОСОВ СЕРВЕРОМ

1.1. 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 ёмкостью 5 запросов.

Построить имитационную модель для определения математического ожидания времени и вероятности обработки запросов.

Сервер представляет собой однофазную систему массового обслуживания разомкнутого типа с ограниченной входной емкостью, то есть с отказами, и абсолютной надёжностью (рис. 1.1).

На рис 1.1 приведены также объекты AnyLogic, которые будут использованы для создания диаграммы процесса. На них мы остановимся позже. Приступим к созданию диаграммы процесса.

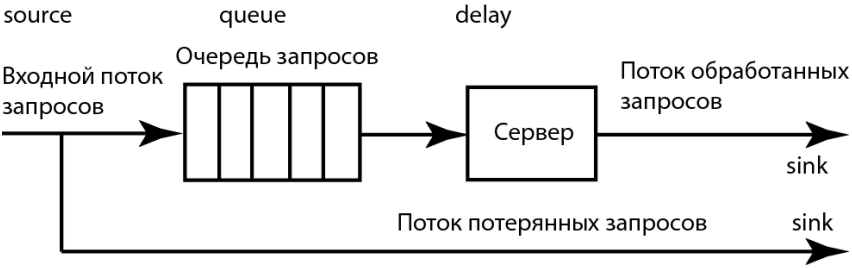


Рис. 1.1. Сервер как система массового обслуживания

1.2. Создание диаграммы процесса

1. Выполните **Файл/Создать/Модель** на панели инструментов. Появится диалоговое окно **Новая модель** (рис. 1.2).

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**.

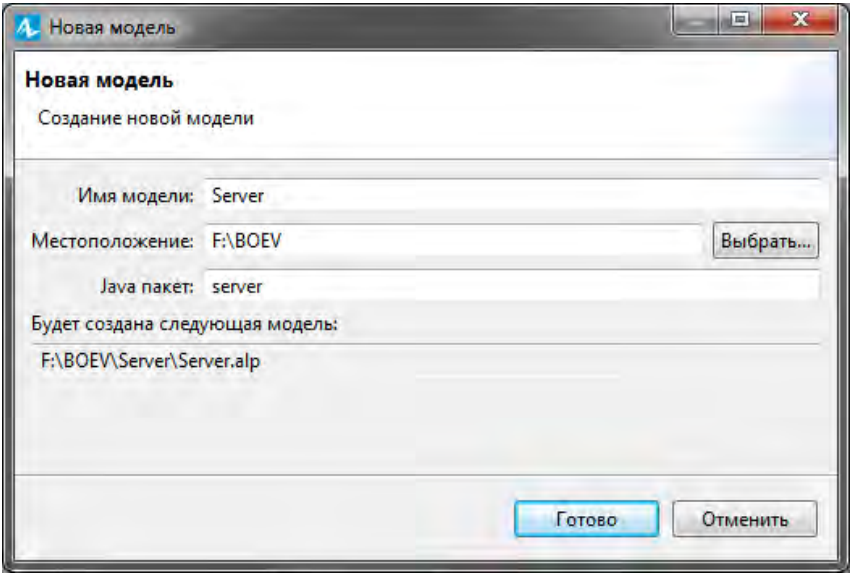


Рис. 1.2. Диалоговое окно **Новая модель**

4. Щёлкните кнопку **Готово**. Откроется пользовательский интерфейс (рис. 1.3). Остановимся на нём.

5. В левой части рабочей области находятся панель **Проекты** и панель **Палитра**. Панель **Проект** обеспечивает навигацию по элементам моделей, открытых в текущий момент времени. Модель организована иерархически. 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.

6. Панель **Палитра** (левый вертикальный столбец) содержит разделённые по категориям элементы, которые могут быть добавлены на графическую диаграмму типа агентов или эксперимента. На рис. 1.3 раскрыта палитра **Презентация**. Для того чтобы открыть нужную палитру, следует подвести курсор к иконке и щелкнуть мышью. Иконка становится светлой.

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

8. В центре рабочей области AnyLogic размещён **графический редактор** диаграммы агента Main.

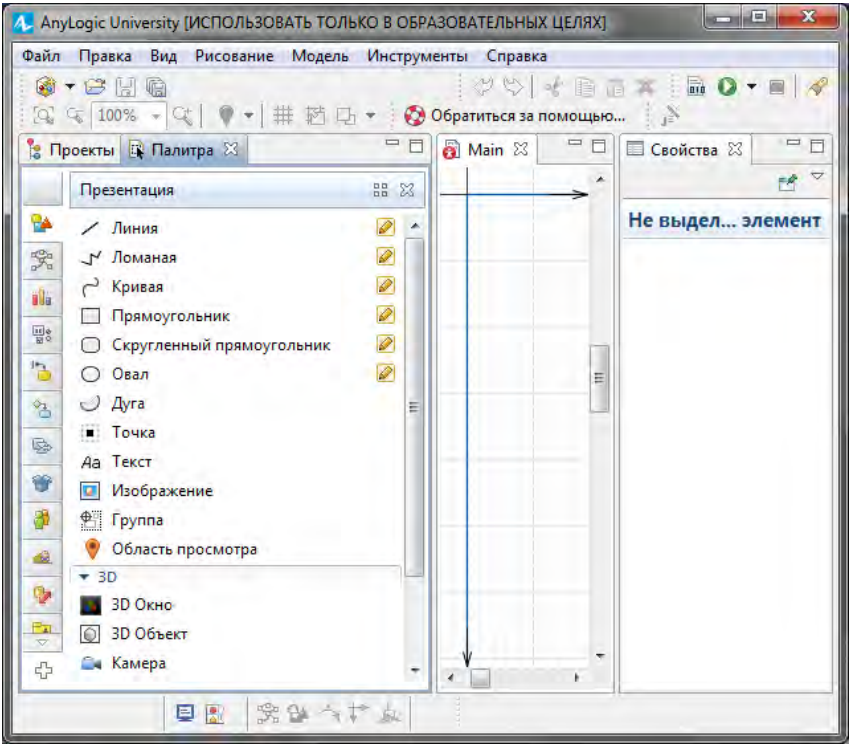


Рис. 1.3. Пользовательский интерфейс

9. При работе не забывайте сохранять производимые изменения с помощью кнопки панели инструментов Сохранить .

10. Создайте диаграмму процесса. Для этого в Палитре выделите **Библиотеку моделирования процессов** (рис. 1.4). Из неё перетащите объекты на диаграмму и соедините, как показано на рис. 1.5. Для добавления объекта на диаграмму, надо щёлкнуть его мыщью и, не отпуская её, перетащить в графический редактор.

11. При перетаскивании объектов вы можете видеть, как появляются соединительные линии между объектами. Обращайте внимание на то, чтобы эти линии, если необходимо, соединяли только порты, находящиеся с правой или левой стороны иконок. Если у вас не получится автоматическое соединение нужных объектов, дважды щёлкните начальный порт. Он станет зелёным. Протащите курсор к конечному порту. Он также станет зелёным. Для завершения процесса соединения дважды щёлкните конечный порт.

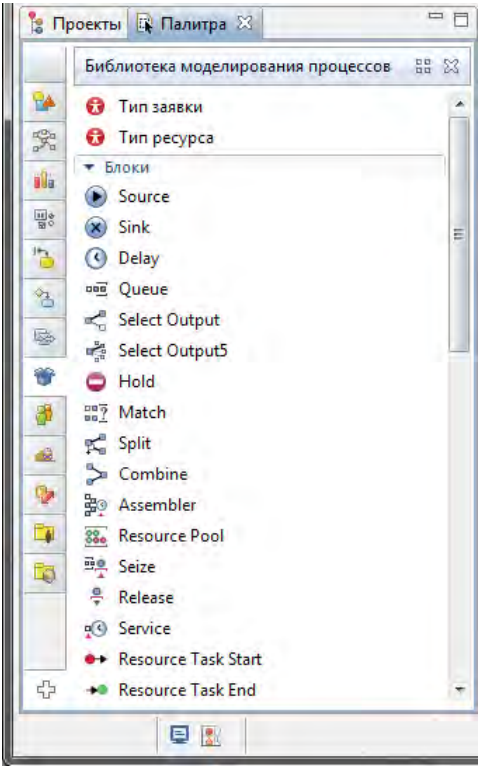


Рис. 1.4. Библиотека моделирования процессов

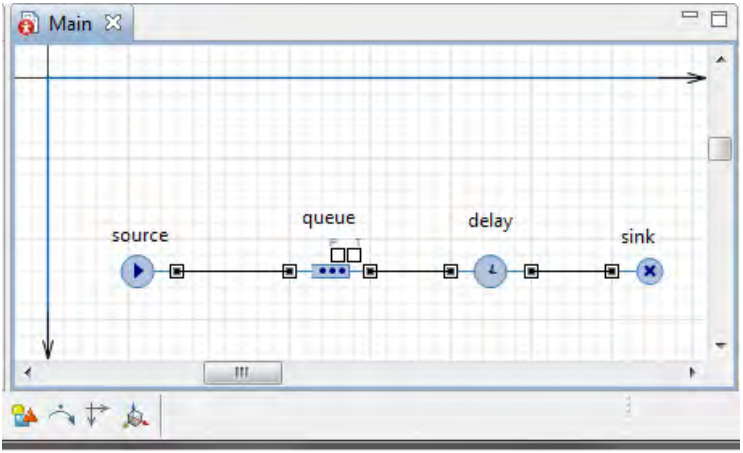


Рис. 1.5. Диаграмма системы массового обслуживания

Дадим краткую характеристику объектов диаграммы.

12. Объект **Source** генерирует заявки определенного типа. Обычно он используется в качестве начальной точки диаграммы процесса, формализующей поток заявок. В нашем примере заявками будут запросы на обработку сервером, а объект Source будет моделировать их поступление.

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

14. Объект **Delay** задерживает заявки на заданный период времени. Он представляет в нашей модели сервер, обрабатывающий запросы.

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

1.3. Изменение свойств блоков модели, её 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 объектов (**Библиотеки моделирование процессов**), задающих последовательность операций, которые будут производиться над проходящими по диаграмме процесса заявками.

Как вы уже знаете диаграмма процесса в AnyLogic создаётся путем добавления объектов библиотеки из палитры на диаграмму класса активного объекта, соединения их портов и изменения значений свойств блоков в соответствии с требованиями модели.

Всё, что нам нужно, чтобы сделать созданную диаграмму модели (см. рис. 1.5) адекватной постановке задачи — это изменить некоторые свойства объектов.

1.3.1. 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 **Проекты**.

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, объекты 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 класса **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, сообщения в моделях систем связи и т.д. В нашем примере заявками будут запросы на обработку данных, а объект source будет моделировать поступление запросов на сервер.

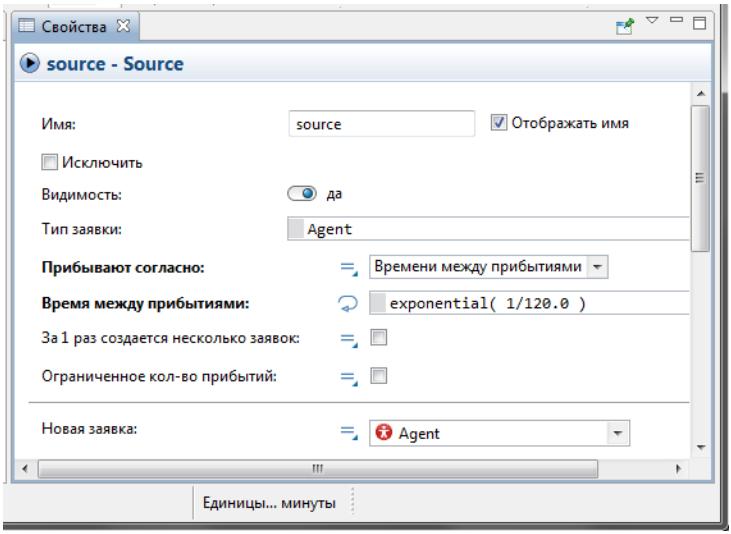


Рис. 1.6. Свойства объекта **source**

В нашем случае объект создает заявки через временной интервал, распределенный по показательному (экспоненциальному) закону со средним значением 2 мин.

Установим среднее время поступления запросов и среднее время их обработки в секундах. Однако имеется возможность установить время в минутах, часах, днях, в чем вы убедитесь несколько позднее, когда будете устанавливать модельное время.

1. Выделите объект source. В выпадающем списке **Прибывают согласно:** укажите, что запросы поступают согласно **Времени между прибытиями:** (рис. 1.6).

2. В поле **Время между прибытиями** появится запись exponential(1). Установите согласно постановке задачи среднее значение интервалов времени поступления запросов на сервер, изменив свойства объекта **source**. Для этого вместо характеристики распределения 1 введите `1/120.0`.

В языке программирования Java символ / означает целочисленное деление, т.е. если оба числа целые, то и результат будет целым. В нашем случае отношение `1/120` было бы равно нулю. Для получения вещественного результата, необходимо, чтобы хотя бы одно из чисел было вещественным (double). Поэтому в качестве характеристики экспоненциального распределения (интенсивности поступления запросов) необходимо указать `1/120.0` или `1.0/120`.

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

Измените свойства объекта **queue** (рис. 1.7).

1. Задайте 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. Для вывода, например, средней длины очереди, нужно в модели предусмотреть Java код.

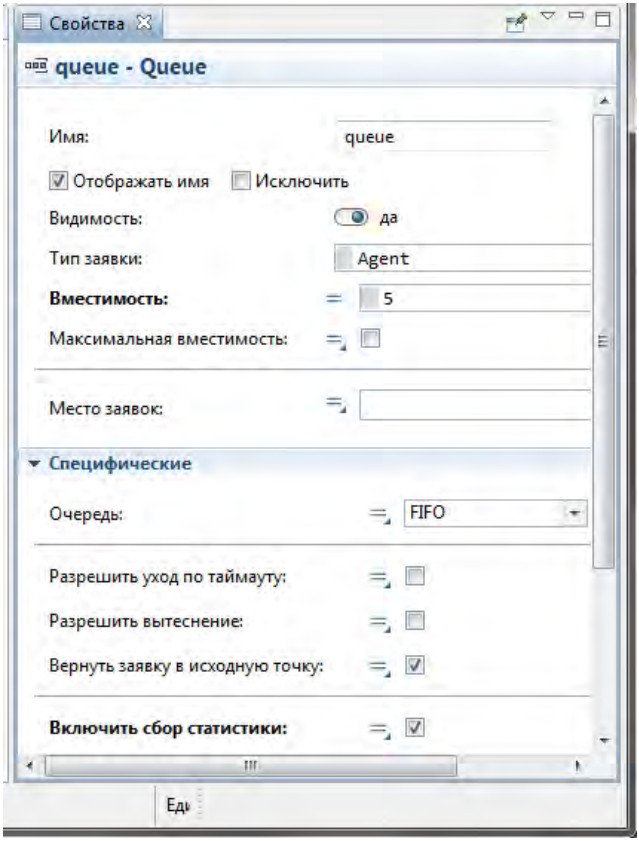


Рис. 1.7. Свойства объекта **queue**

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.

Измените свойства объекта **delay** (рис. 1.8).

1. Обработка одного запроса занимает примерно 3 мин. Задайте время обслуживания, распределенное по экспоненциальному закону со средним значением 3 мин. Для этого введите в поле **Время задержки:** `exponential(1/180.0)`. Функция exponential() является стандартной функцией генератора случайных чисел AnyLogic. AnyLogic предоставляет функции и других случайных распределений, таких как нормальное, треугольное, и т. д.
2. Установите флажок **Включить сбор статистики**.

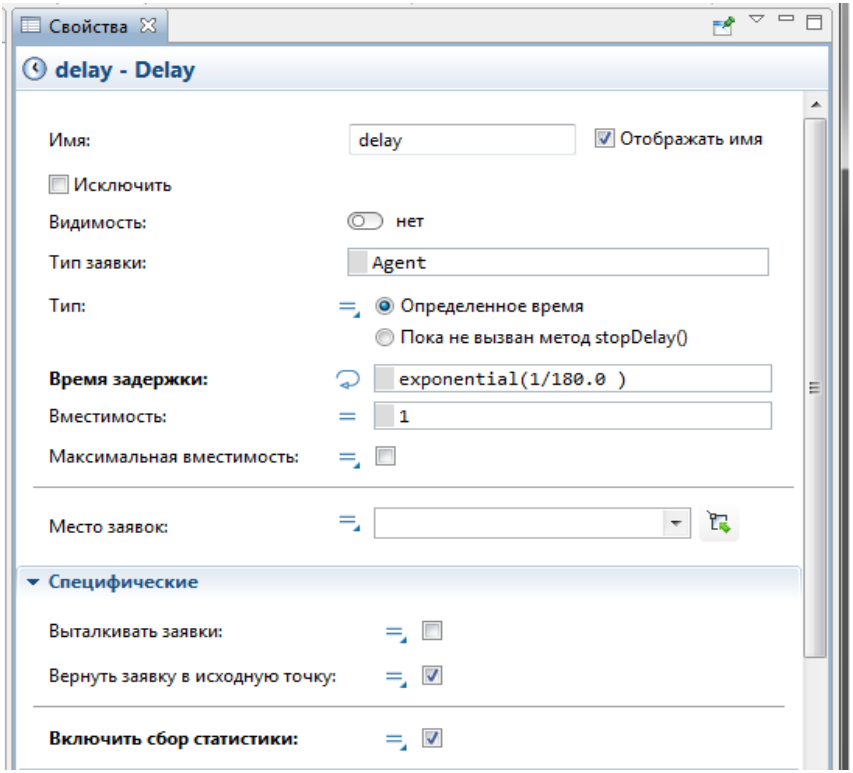


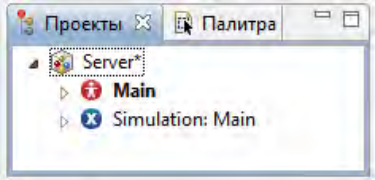
Рис. 1.8. Свойства объекта **delay**

Для вывода коэффициента использования объекта **delay** в модели также следует предусмотреть соответствующий Java код.

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.

1.3.2. 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 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 вы хотите наблюдать поведение модели в течение длительного периода (до того момента, пока вы сами не остановите выполнение модели), то по умолчанию времени остановки нет. Обработку запросов сервером мы планируем исследовать в течение одного часа, т.е. 3600 с.

1. В панели **Проект** выделите эксперимент **Simulation:Main**.
2. Щелчком раскройте вкладку **Модельное время**.
3. Установите **Виртуальное время (максимальная скорость)** (рис. 1.9).
4. В поле **Остановить:** выберите из списка `В заданное время`.
5. В поле **Конечное время:** установите `3600`.
6. Раскройте вкладку **Случайность**.
7. Выберите опцию **Фиксированное начальное число (воспроизводимые прогоны)**.
8. В поле **Начальное число:** установите `9`.
9. В панели **Проект**, выделите **Server** (рис. 1.10).
10. Из выпадающего списка **Единицы модельного времени:** выберите `секунды`.

1.3.3. 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 её. При этом откроется соответствующее место ошибки.

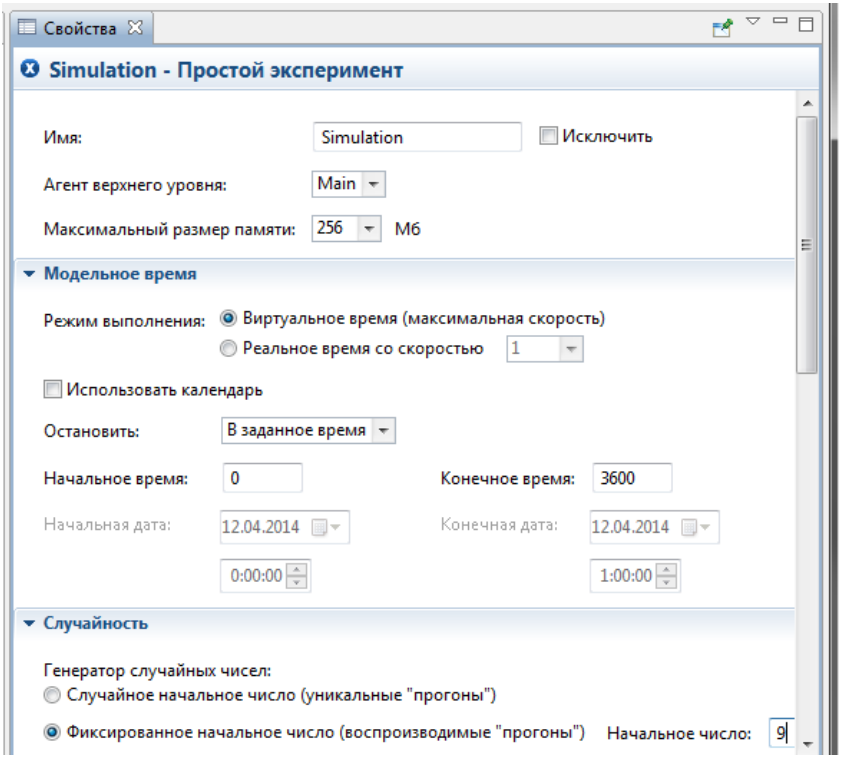


Рис. 1.9. Установка свойств эксперимента

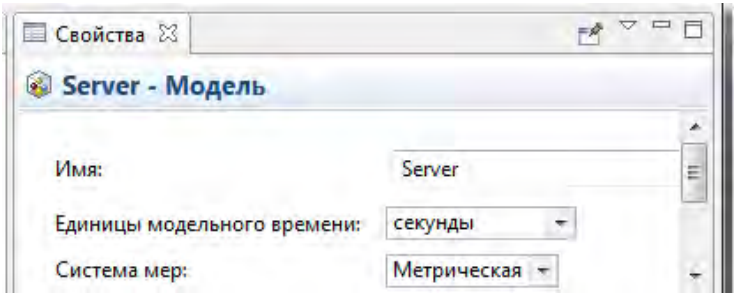


Рис. 1.10. Установка модельного времени

После исправления ошибок и построения модели, запустите её:

1. Щёлкните 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**.

2. В дальнейшем нажатием кнопки **Запустить** (или кнопки F5) будет запускаться тот эксперимент, который запускался вами в последний раз. Чтобы выбрать 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. 1. 11). 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).

4. Щёлкните данную кнопку. Этим щелчком вы запустите модель и перейдете к презентации корневого класса активного объекта запущенного эксперимента. Для каждой модели, созданной в **Библиотеке моделирования процессов**, 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. 1.12).

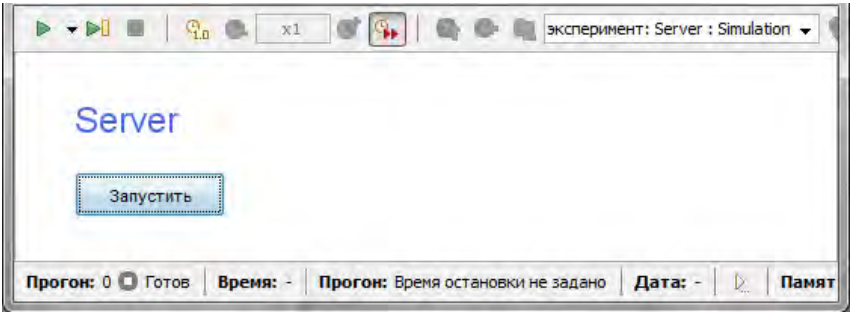


Рис. 1.11. Окно презентации модели

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 её принять. Если

заявка должна покинуть объект, а следующий объект не готов её принять, и заявка не может ждать, то модель останавливается с ошибкой (рис. 1.12). Ошибка означает, что запрос не может покинуть объект source и войти в блок queue, так как его ёмкость, равная 5, заполнена. Также выдаётся сообщение о логической ошибке в модели (рис. 1.13)

Рис. 1.12. Модель остановилась с ошибкой

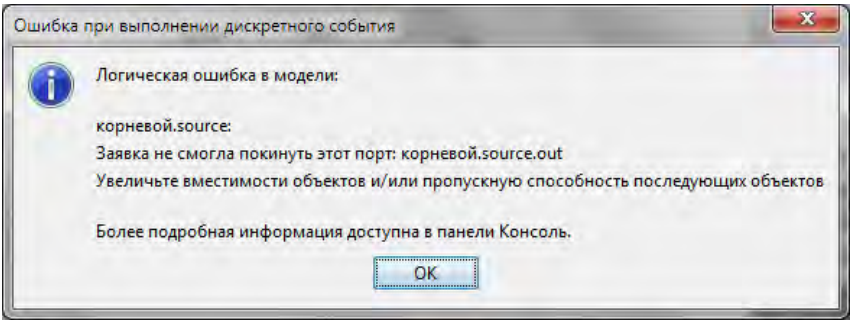


Рис. 1.13. Сообщение о логической ошибке в модели

6. Нажмите OK. Далее измените свойства объекта **queue**, т. е. увеличьте длину очереди (см. рис. 1.7). Для этого введите в поле **Вместимость** `15`. Можете убедиться, что при увеличении ёмкости в пределах 6 … 14 модель по-прежнему останавливается с этой же ошибкой. Момент появления ошибки зависит от длительности времени моделирования.

7. Снова запустите модель.

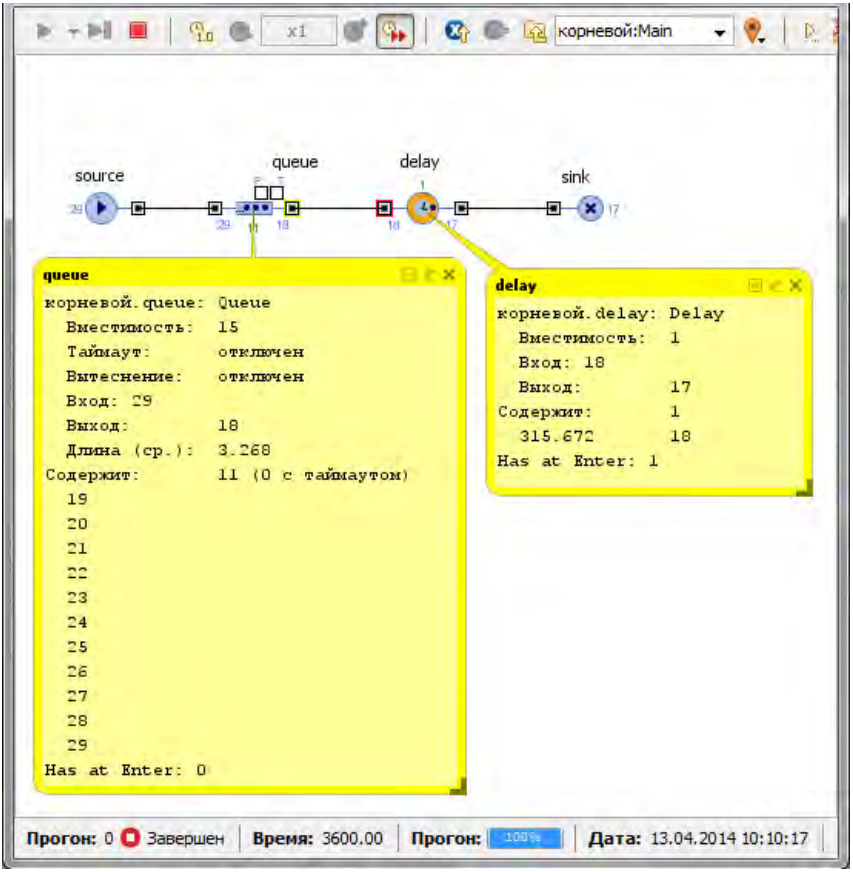


Рис. 1.14. Окно инспекта

8. Вы можете следить за состоянием любого объекта диаграммы процесса во время выполнения модели с помощью окна инспекта этого объекта. Чтобы открыть окно инспекта, щёлкните мышью по значку нужного блока. Окно инспекта, подведя курсор, можно перемещать в нужное вам место. Также, подведя курсор к правому нижнему углу окна инспекта, можно при необходимости изменять его размеры.

9. В окне инспекта будет отображена базовая информация по выделенному объекту: например, для объекта queue будут отображены вместимость очереди, количество заявок, прошедшее через каждый порт объекта и т. д. Такая же информация содержится в инспекте и для объекта delay (рис. 1.14).

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

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

1.4. Создание анимации модели

Можно было бы наблюдать, анализировать и интерпретировать работу запущенной модели с помощью визуализированной диаграммы процесса (см. рис. 1.12, 1.14).

Однако удобнее в ряде случаев иметь более наглядную визуализацию с помощью анимации. В этой задаче 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 прямоугольный узел, который будет обозначать на анимации сервер.

1. Откройте палитру **Разметка пространства** (рис. 1.15). Чтобы открыть какую-либо палитру, нужно щелчком из **Проекты** перейти в **Палитра** и щёлкнуть по иконке этой палитры.

2. Палитра **Разметка пространства** (рис. 1.15) содержит в качестве элементов различные примитивные фигуры, используемые для рисования презентаций моделей. Это путь, прямоугольный узел, многоугольный узел, точечный узел, аттрактор, стеллаж, масштаб.

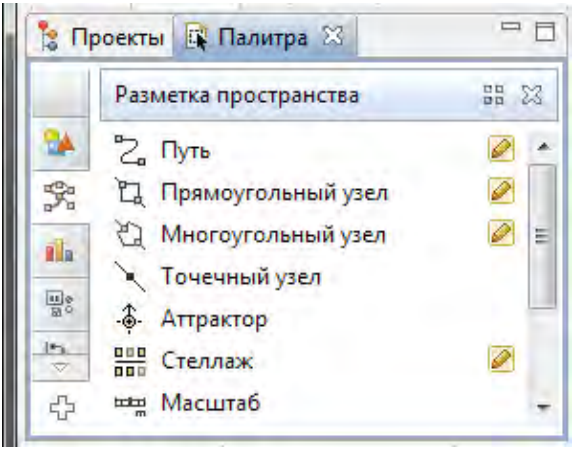


Рис. 1.15. Палитра **Разметка пространства**

3. Выделите элемент **Прямоугольный узел** и перетащите его на диаграмму класса активного объекта. Поместите элемент **Прямоугольный узел** так, как показано на рис. 1.16.

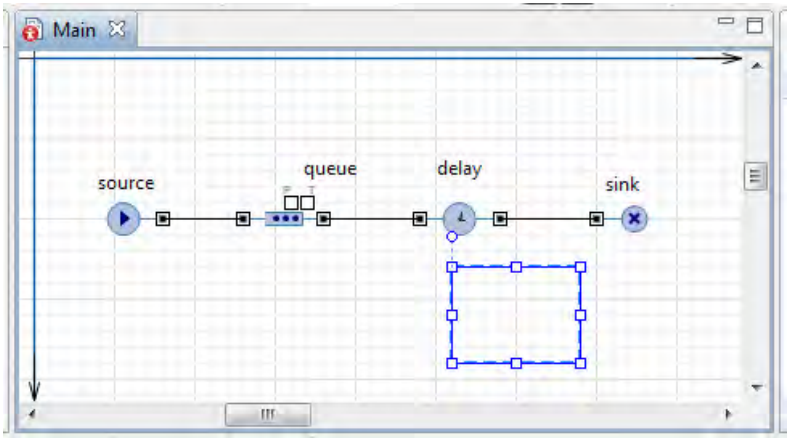


Рис. 1.16. Элемент **Прямоугольный узел** на диаграмме

4. Давайте сделаем так, чтобы цвет этого прямоугольного узла будет меняться в зависимости от того, обрабатывет ли сервер в данный момент времени запрос или нет.

5. Для этого выделите нарисованную нами фигуру на диаграмме. Перейдите на страницу **Внешний вид** панели свойств (рис. 1.17).

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

Возвращаемый результат вычисления будет присваиваться текущему значению этого свойства. Мы хотим, чтобы во время моделирования менялся цвет нашей фигуры, поэтому щёлкните в поле **Цвет заливки:** по стрелке, выберите **Динамическое значение** и введите там следующую строку:

```
delay.size()>0?red:green
```

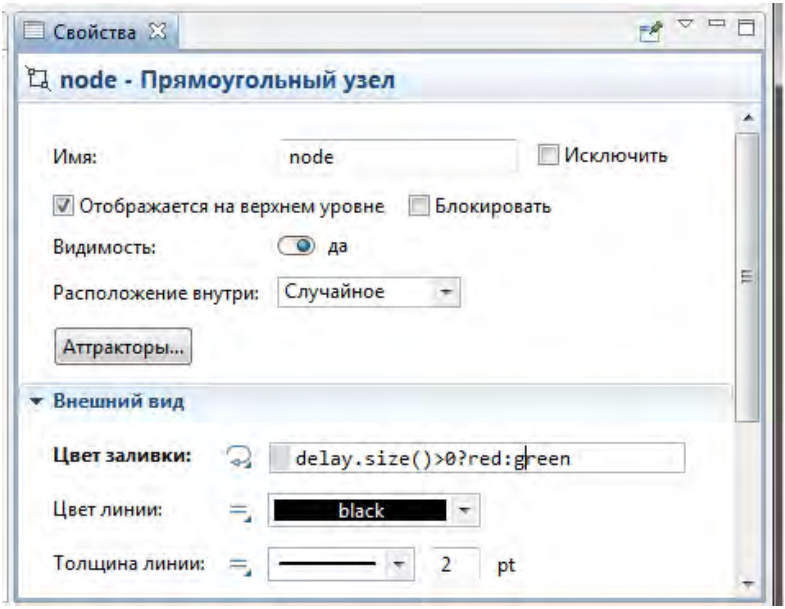


Рис. 1.17. Установлено динамическое значение цвета заливки

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 — зелёным.

6. Нарисуйте путь, который будет обозначать на анимации очередь к серверу (рис. 1.18). Чтобы нарисовать путь, сделайте двойной щелчок мышью по элементу **Путь** палитры **Разметка пространства**, чтобы перейти в *режим рисования*. Теперь вы мо-

жете рисовать путь точка за точкой, последовательно щелкая мышью в тех точках диаграммы, куда вы хотите поместить вершины пути. Чтобы завершить рисование, добавьте последнюю точку пути двойным щелчком мыши.

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

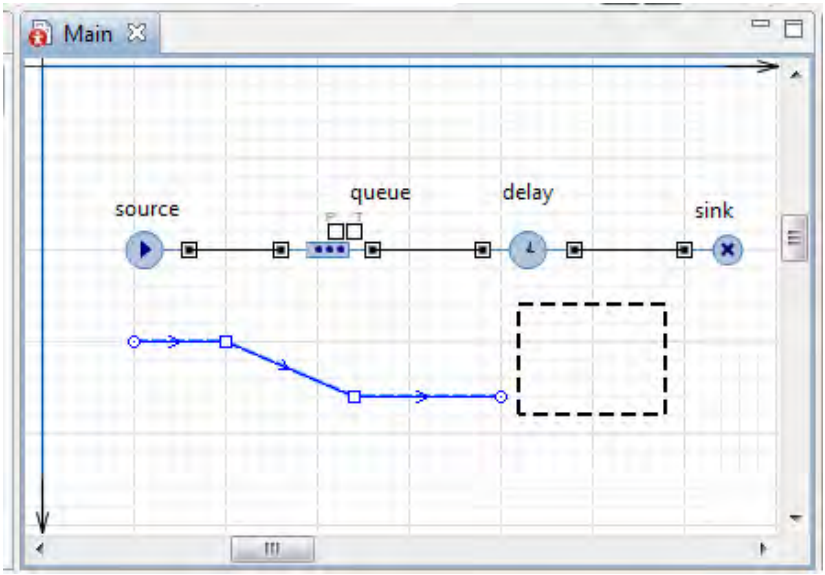


Рис. 1.18. Путь на диаграмме процесса

7. Теперь мы должны задать созданные анимационные объекты в качестве анимационных фигур объектов диаграммы нашего процесса. Задайте путь в качестве фигуры анимации очереди. Выделите объект queue. На странице свойств объекта queue в поле **Место заявок:** выберите из выпадающего списка `path` (рис. 1.19).

8. Задайте прямоугольный узел в качестве фигуры анимации сервера. Выделите объект delay. Введите в поле **Место заявок:** из выпадающего списка имя нашего прямоугольного узла: `node` (рис. 1.20).

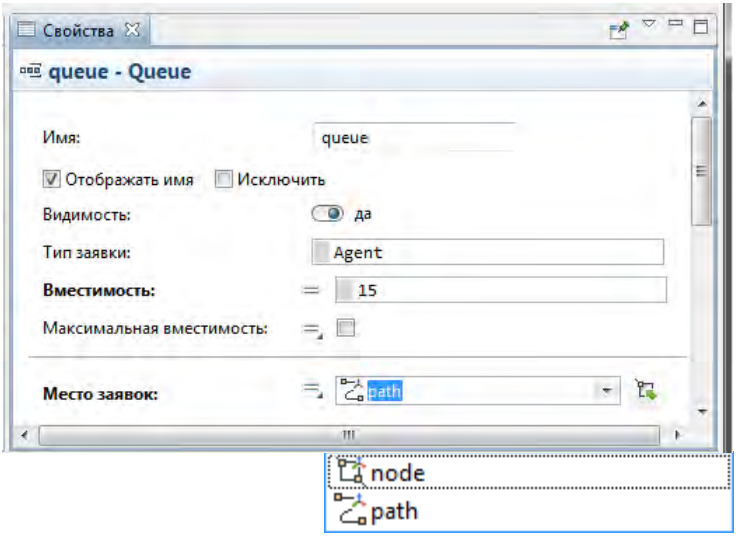


Рис. 1.19. Задание пути в качестве фигуры анимации очереди

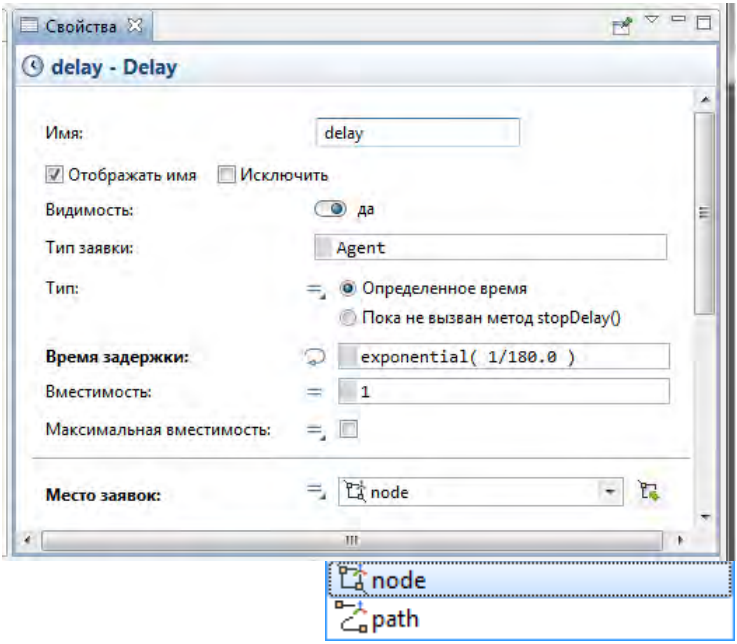


Рис. 1.20. Задание прямоугольного узла
в качестве фигуры анимации сервера

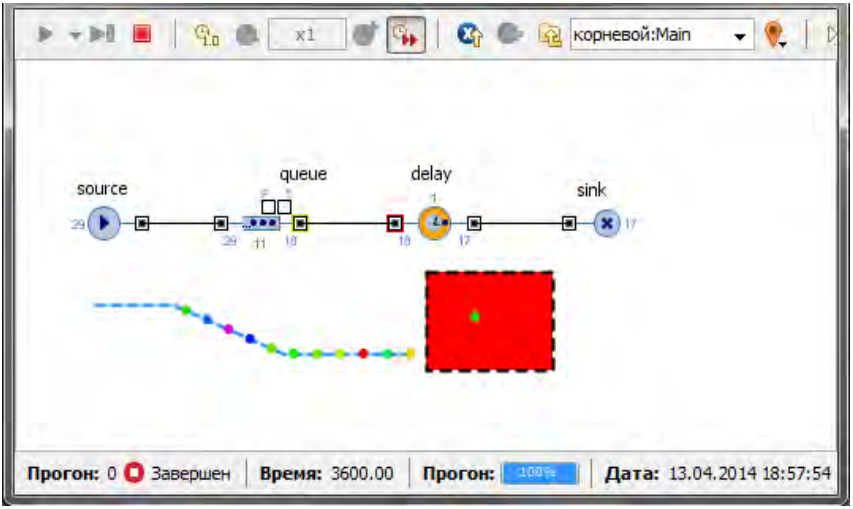


Рис. 1.21. Анимация модели

9. Запустите модель. Вы увидите, что у модели теперь есть простейшая анимация — сервер и очередь запросов к нему (рис. 1.21). Цвет фигуры сервера будет меняться в зависимости от того, обрабатывается ли запрос в данный момент времени или нет.

1.5. 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 среднего коэффициента использования 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. 1.32.

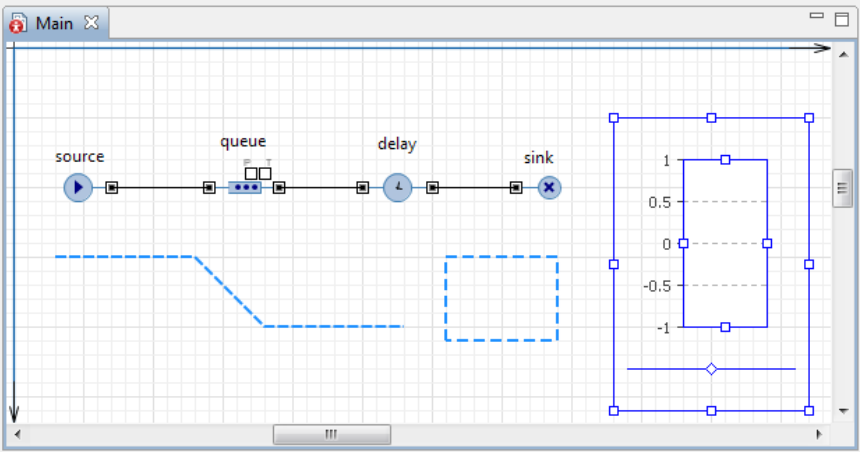


Рис. 1.22. Элемент **Столбиковая диаграмма** на диаграмме класса

3. Перейдите на панель **Свойства**. Щёлкните кнопку **Добавить элемент данных**. После щелчка появится секция свойств того элемента данных (**chart – Столбиковая диаграмма**), который будет отображаться на этой диаграмме (рис. 1.23).

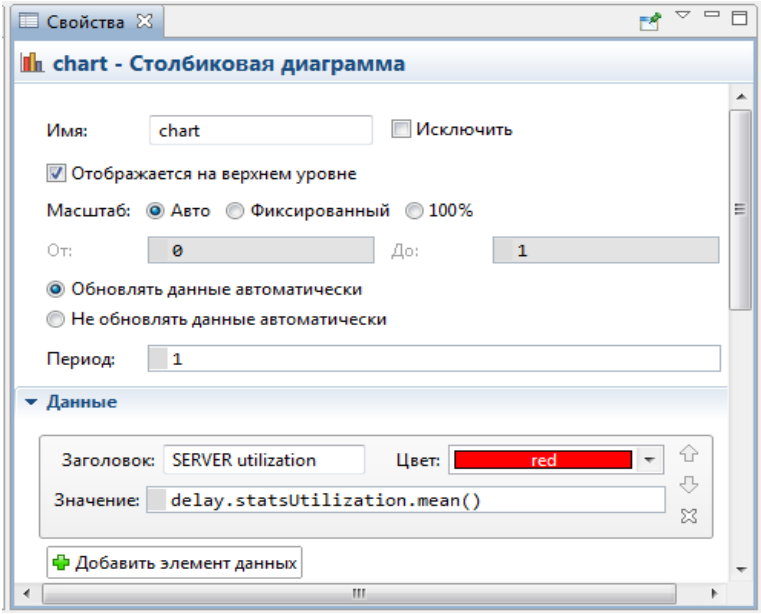


Рис. 1.23. Страница **Свойства**

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. Щёлкните **Внешний вид** (рис. 1.24). Установите свойства: направление столбцов, цвета фона, границ, меток, сетки, положение подписей у столбцов.

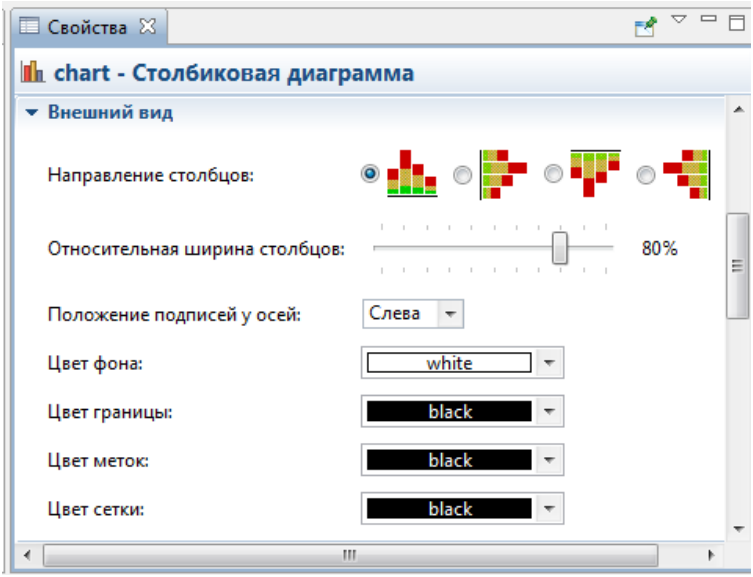


Рис. 1.24. Вкладка **Внешний вид**

7. Раскройте щелчками страницы (вкладки) **Местоположение и размер**, **Легенда**, **Область диаграммы** (рис. 1.25). Установите свойства, чтобы изменить расположение легенды относительно диаграммы (мы хотим, чтобы она отображалась внизу), размер диаграммы, высоту, ширину, координаты размещения на диаграмме, цвета текста, границы.

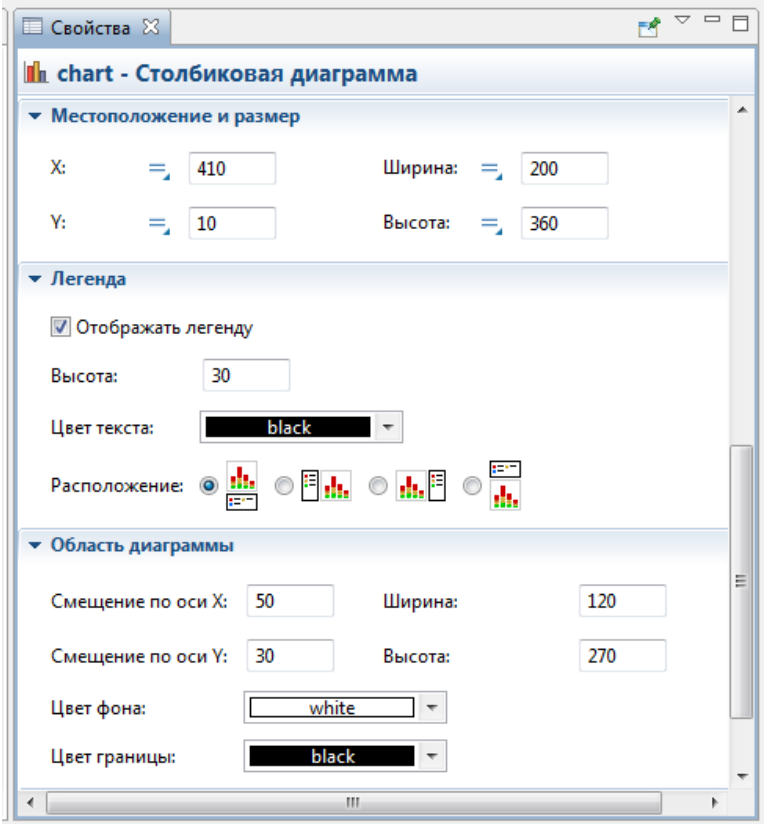


Рис. 1.25. Вкладки **Местоположение и размер**, **Легенда**, **Область диаграммы**

8. Аналогичным образом добавьте еще одну столбиковую диаграмму для отображения средней длины очереди.

9. На панели **Свойства** щёлкните **Добавить элемент данных**. После щелчка появится страница **Данные** свойств элемента данных (**chart1 – Столбиковая диаграмма**), который также будет отображаться на этой диаграмме (рис. 1.26).

10. **Заголовок:** и **Значение:** измените так, как показано на рис. 1.26. В поле **Заголовок:** введите `Queue lenqth`, а в поле **Значение:** введите `queue.statsSize.mean()`.

11. На страницах **Внешний вид**, **Местоположение и размер**, **Легенда**, **Область диаграммы** установите свойства самостоятельно. Столбцы диаграммы должны размещаться горизонтально.

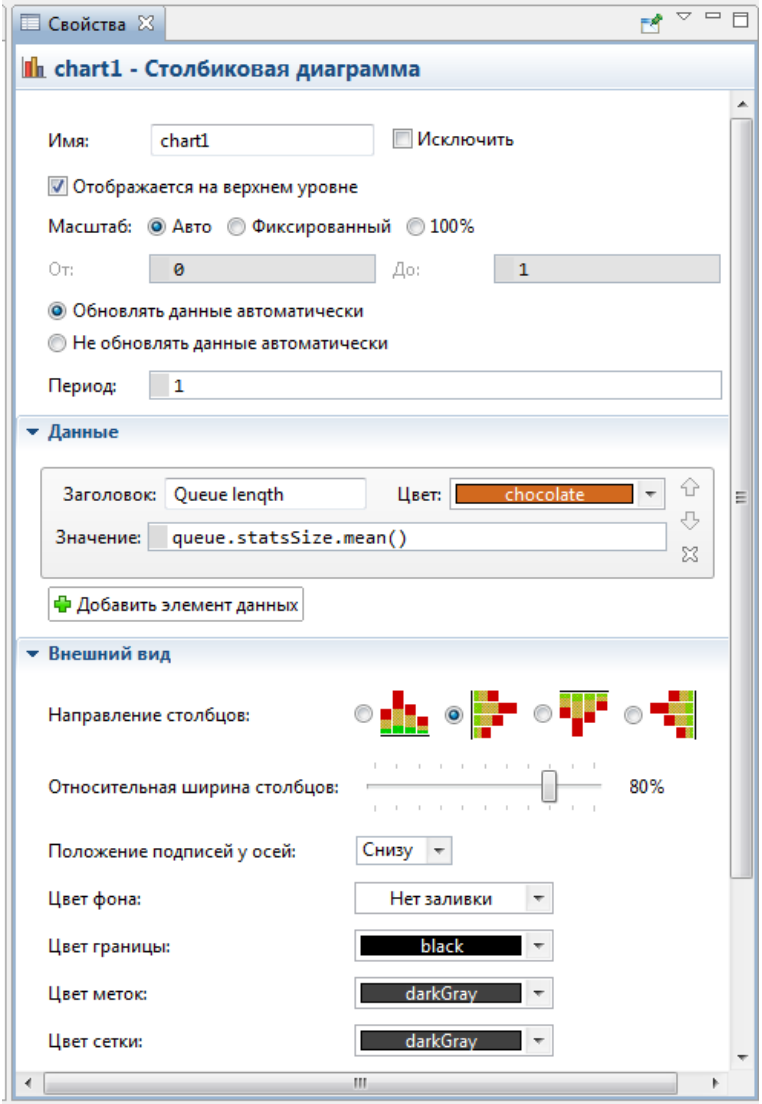


Рис. 1.26. Страницы **Данные**, **Внешний вид** панели **Свойства**

12. В поле **Значение:** 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.

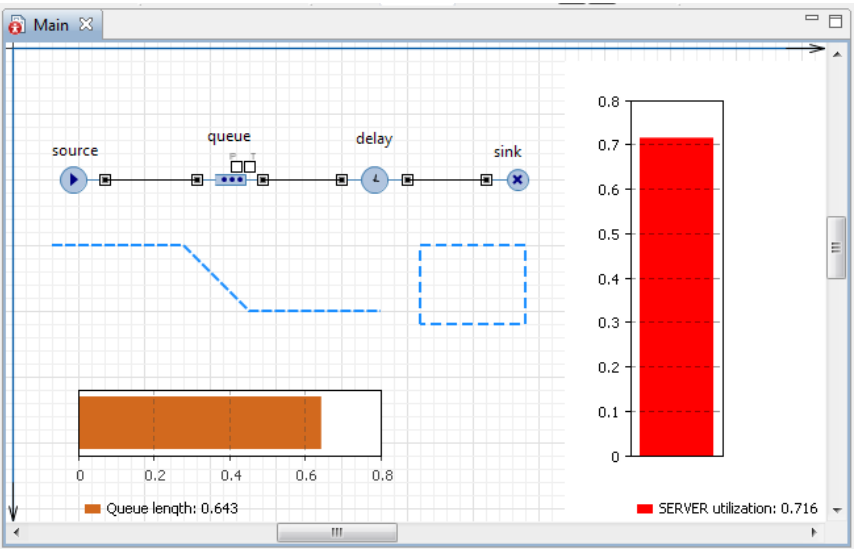


Рис. 1.27. Добавлены две столбиковые диаграммы

13. На странице **Внешний вид** панели **Свойства** выберите в секции свойств **Направление столбцов** вторую опцию (рис. 1.26), чтобы столбцы во второй столбиковой диаграмме, расположенной горизонтально, росли вправо (рис. 1.27).

14. Запустите модель с двумя столбиковыми диаграммами, установив модельное время 3600 единиц, и понаблюдайте за её работой (рис. 1.28).

1.6. Уточнение модели согласно ёмкости входного буфера

На рис. 1.28 (снимок сделан по окончании времени моделирования) видно, что длина очереди равна 14 запросам при установленной максимальной длине 15. Но ведь в постановке задачи ёмкость буфера была определена в 5 запросов. Нам не удалось до этого построить модель с такой ёмкостью из-за ошибки (см. рис. 1.22) — невозможности очередного запроса покинуть блок source, так как длина очереди уже была равна 5 запросам. Нам пришлось во избежание этой ошибки увеличить ёмкость буфера до 15 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.

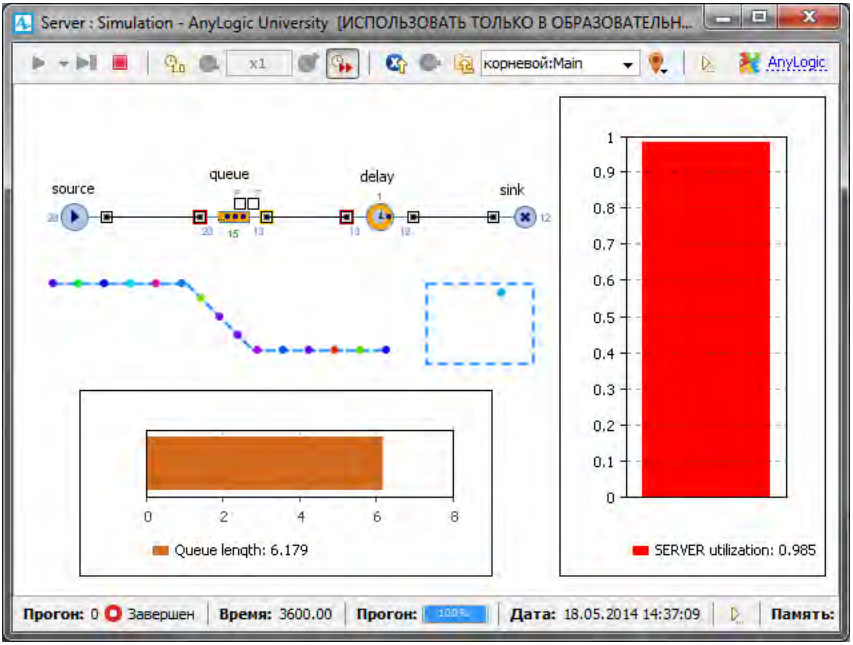


Рис. 1.28. Наблюдение за моделью с двумя столбиковыми диаграммами

Объект queue моделирует очередь заявок, ожидающих приёма 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 её приоритет, и поместит в очередь в позицию, соответствующую её 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 неё будет вытеснена именно эта новая заявка*.

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

1. Выделите объект queue. На панели **Свойства** измените **Вместимость** с 15 на 5 запросов.
2. Здесь же установите **Разрешить вытеснение**.
3. Для уничтожения потерянных запросов вследствие полного заполнения накопителя нужно добавить второй объект sink. Откройте в **Палитре Библиотеку моделирования процессов** и перетащите блок sink на диаграмму (рис. 1.29). При перетаскивании объект пытается автоматически соединиться с входами имеющимися на диаграмме объектами. Но это может вас не устраивать.
4. Тогда соедините порт outPreempted объекта queue с входным портом InPort блока sink1. Чтобы соединить порты, сделайте двойной щелчок мышью по одному порту, например, outPreempted, затем последовательно Щёлкните в тех местах диаграммы, где вы хотите поместить точки изгиба соединителя.
5. После двойного щелчка по второму порту вы увидите, что появится соединитель. Если выделить его мышью, то при правильном соединении портов конечные точки соединителя должны подсветиться зелеными точками. Если нет, то точки не были помещены точно внутрь портов, и их нужно будет туда передвинуть.

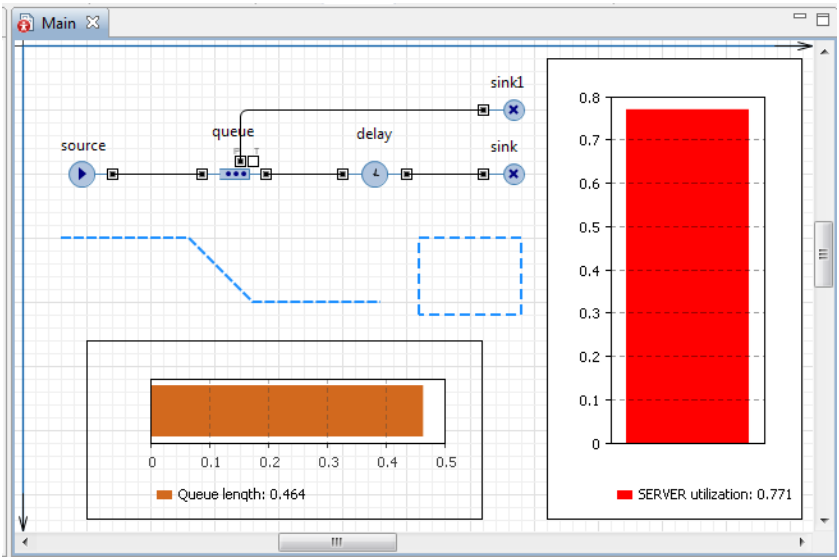


Рис. 1.29. Уточненная модель

6. Запустите уточненную модель и понаблюдайте за ее работой. Сравните рис. 1.30 с рис. 1.22. На рис. 1.30 видно, что запросы при длине очереди в 5 запросов теряются, и ошибки при этом не возникает. Модель по ограничению ёмкости 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.

1.7. 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 моделирования процессов**. Entity по существу является обычным Java классом с теми функциональными возможностями, которые необходимы и достаточны для обработки и отображения анимации заявки объектами **Библиотеки моделирования процессов**. Эти функциональные возможности можно расширить добавлением дополнительных полей и методов и работой с ними из объектов диаграммы, описывающей моделируемый процесс.

Согласно постановке задачи нужно определять математическое ожидание времени и вероятности обработки запросов сервером.

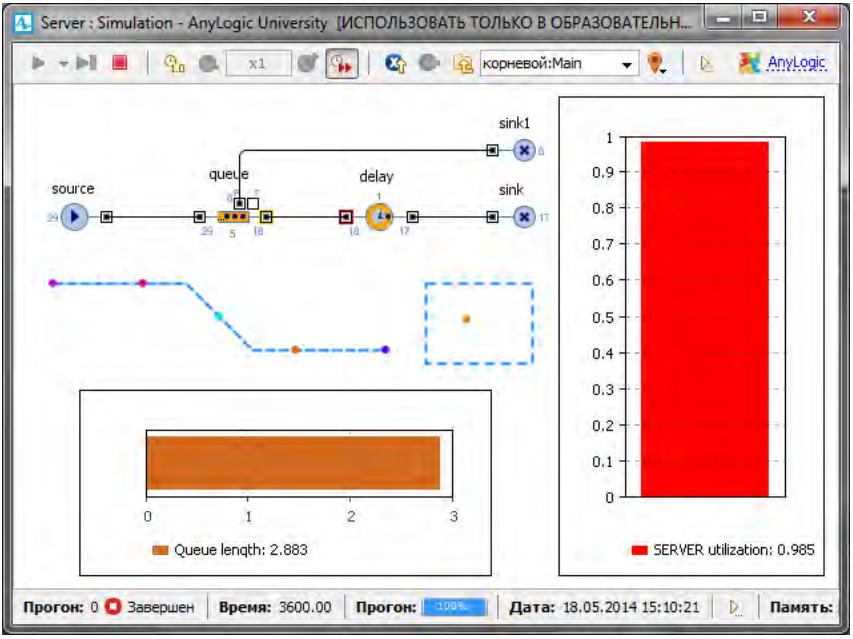


Рис. 1.30. Работа модели согласно ёмкости входного буфера

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

1.7.1. Создание нестандартного Java класса

Для включения в запросы дополнительных полей необходимо создать нестандартный тип заявки. Это возможно двумя способами. Создадим первым способом тип заявок `Inquiry`.

1. В панели **Проект** щёлкните правой кнопкой мыши элемент модели верхнего уровня дерева и выберите из контекстного меню **Создать/Java класс**.
2. Появится диалоговое окно **Новый Java класс** (рис. 1.31). В поле **Имя:** введите имя нового класса `Inquiry`.
3. В поле **Базовый класс:** выберите из выпадающего списка `Entity` в качестве базового класса. Щёлкните кнопку **Далее**.

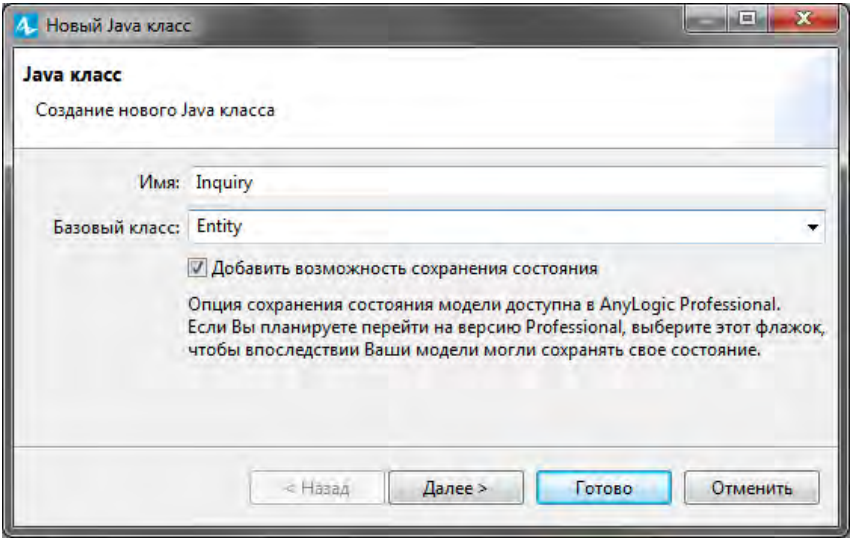


Рис. 1.31. Диалоговое окно создания нового Java класса

4. Появится вторая страница **Мастера создания Java класса**. (рис. 1.32).

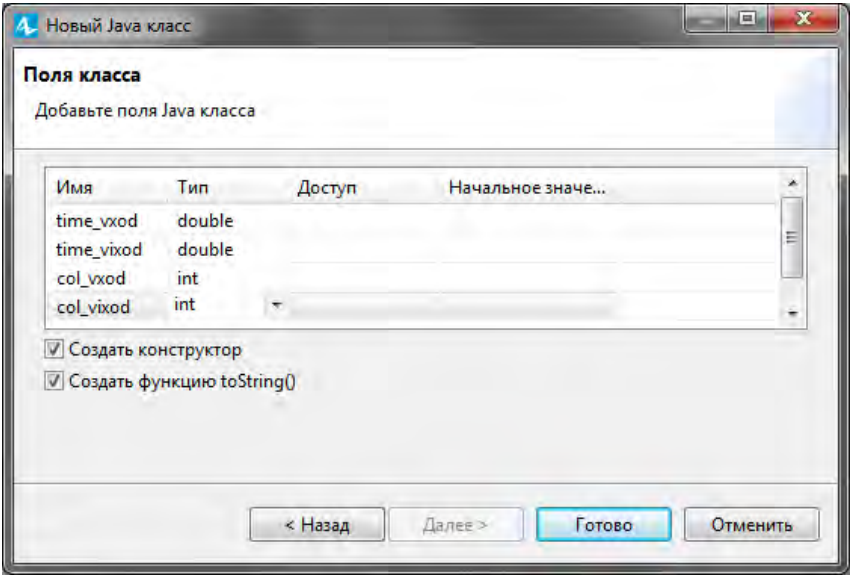


Рис. 1.32. Вторая страница **Мастера создания Новый Java класс**

5. Добавьте поля Java класса: `time_vxod` типа `double`, `time_vixod` типа `double`, `col_vxod` типа `int`, `col_vixod` типа `int`. Типы полей выбираются из выпадающего списка. Начальные значения всех параметров, поскольку не указаны, по умолчанию будут установлены равными нулю.

6. 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.

7. Щёлкните кнопку **Готово**. Вы увидите редактор кода, в котором будет показан автоматически созданный код вашего Java класса (рис. 1.33). Закройте редактор, щелкнув крестик в закладке рядом с его названием.

8. Теперь нужно преобразовать Java класс в тип агента. Для этого щёлкните правой кнопкой мыши в панели **Проект** только что созданный Java класс и в контекстном меню выберите **Преобразовать Java класс в тип агента**.

9. Появится окно с автоматически созданными параметрами нестандартного типа заявок Inquiry (см. рис. 1. 38).

Main | Inquiry.java

```
1  /**
2   * Inquiry
3   */
4  public class Inquiry extends com.xj.anylogic.libraries.enterprise.
5
6      double time_vxod;
7
8      double time_vixod;
9
10     int col_vxod;
11
12     int col_vixod;
13
14    /**
15     * Конструктор по умолчанию
16     */
17    public Inquiry() {
18    }
19
20    /**
21     * Конструктор, инициализирующий поля
22     */
23    public Inquiry(double time_vxod, double time_vixod, int col_vx
24        this.time_vxod = time_vxod;
25        this.time_vixod = time_vixod;
26        this.col_vxod = col_vxod;
27        this.col_vixod = col_vixod;
28    }
29
30    @Override
31    public String toString() {
32        return
33            "time_vxod = " + time_vxod +" " +
34            "time_vixod = " + time_vixod +" " +
35            "col_vxod = " + col_vxod +" " +
36            "col_vixod = " + col_vixod +" ";
37    }
38
39    /**
40     * Это число используется при сохранении состояния модели<br>
41     * Его рекомендуется изменить в случае изменения класса
42     */
43    private static final long serialVersionUID = 1L;
44
45 }
```

Рис. 1.33. Окно редактора кода созданного нестандартного класса

10. Закройте оно, щелкнув крестик в закладке рядом с его названием. Удалите всё, что касается только что созданного Java класса и заявки. Перейдём к созданию непосредственно нестандартного типа заявки вторым способом.

Создайте тип заявок `Inquiry`.

1. Откройте палитру **Библиотека моделирования процесов**.
2. Перетащите элемент **Тип заявки** в графический редактор. Появится тип заявки `Entity` (рис. 1.34).
3. Появится диалоговое окно **Создание агента** (рис. 1.35).

Шаг 1. Анимация агента.

В поле **Имя нового агента:** введите `Inquiry`.

Выберите анимацию агента: установите 2D и выберите из выпадающего списка, например, **Сообщение**.

Щёлкните **Далее**.

4. Появится диалоговое окно **Создание агента. Шаг 2. Параметры агента** (рис. 1.36).
5. Щёлкните **<добавить…>**. В поле **Параметр:** введите `time_vxod` (рис. 1.37).
6. Из выпадающего списка **Тип:** выберите `double`.
7. Щёлкните второй раз **<добавить…>**. В поле **Параметр:** введите `time_vixod`.
8. Из выпадающего списка **Тип:** выберите `double`.
9. Щёлкните третий раз **<добавить…>**. В поле **Параметр:** введите `col_vxod`.
10. Из выпадающего списка **Тип:** оставьте `int`.

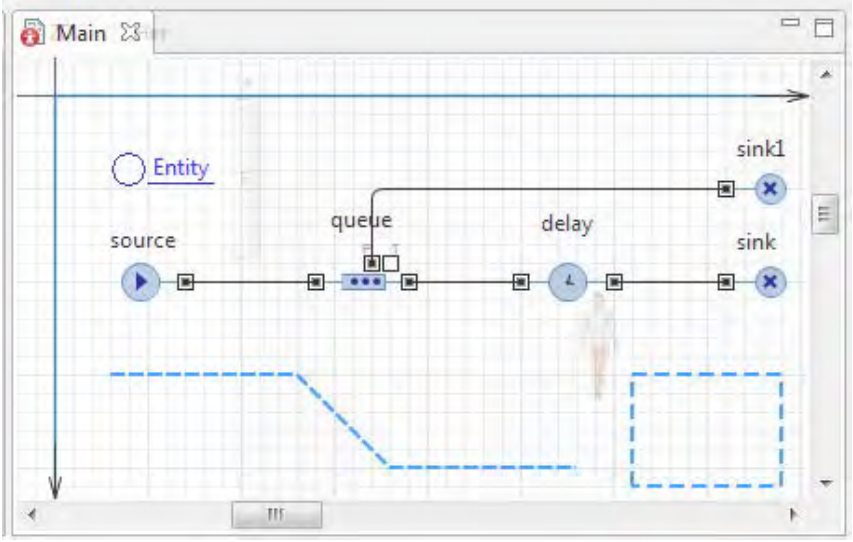


Рис. 1.34. Появился тип заявки `Entity`

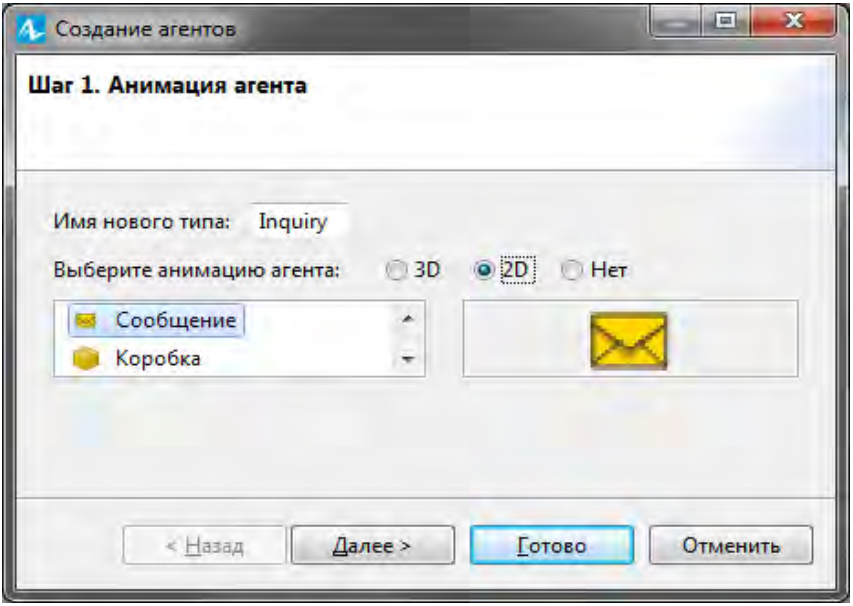


Рис. 1.35. Диалоговое окно **Создание агента. Шаг 1. Анимация агента**

1. Щёлкните третий раз **<добавить…>**. В поле **Параметр:** введите `col_vixod`.
2. Из выпадающего списка **Тип:** оставьте `int`.

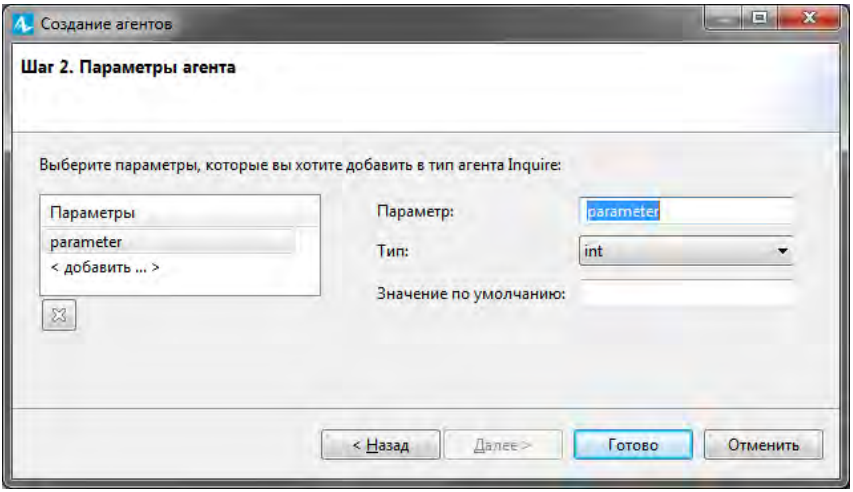


Рис. 1.36. Диалоговое окно **Создание агента. Шаг 2. Параметры агента**

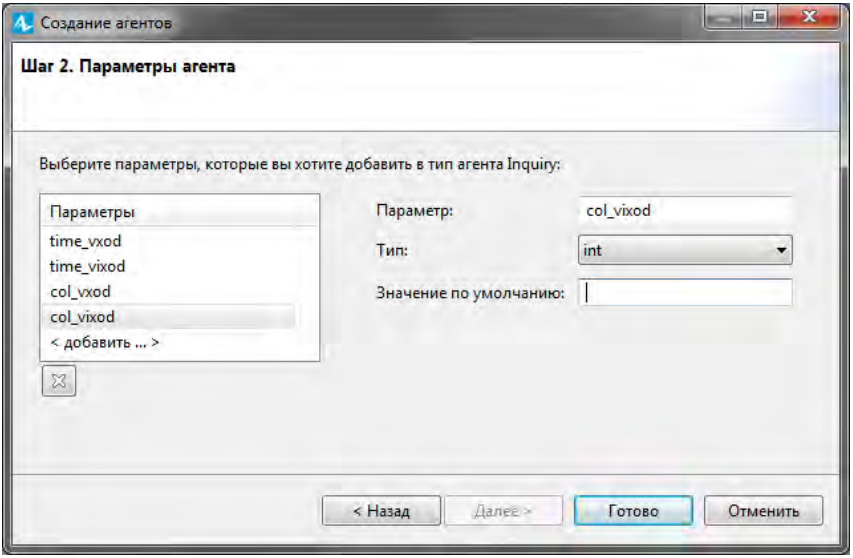


Рис. 1.37. Диалоговое окно **Создание агента. Шаг 2. Параметры агента** с установленными параметрами нестандартного типа заявок Inquiry

3. Так как в поле **Значение по умолчанию** мы не устанавливали никаких значений, то всем параметрам будет установлен 0.

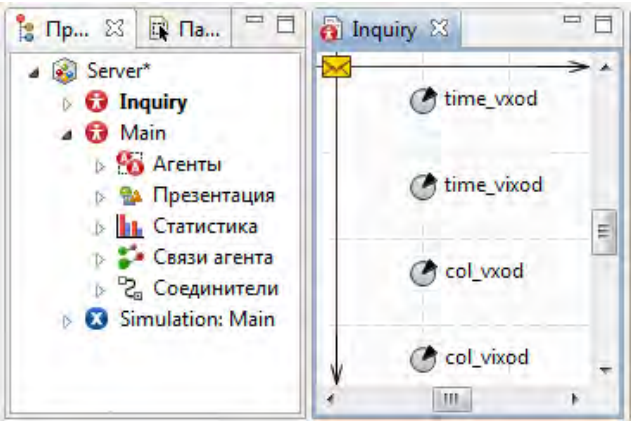


Рис. 1.38. Окно с параметрами нестандартного типа заявки Inquiry

4. Щёлкните кнопку **Готово**. Вы увидите окно, в котором будут показаны автоматически созданные параметры нестандартного типа заявок Inquiry (рис. 1.38). Закройте оно, щелкнув крестик в закладке рядом с его названием.

1.7.2. Добавление элементов статистики

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

1. Чтобы добавить элемент сбора данных гистограммы на диаграмму, перетащите элемент **Данные гистограммы** с палитры **Статистика** на диаграмму активного класса.

2. Задайте свойства элемента (рис. 1.39):

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

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

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

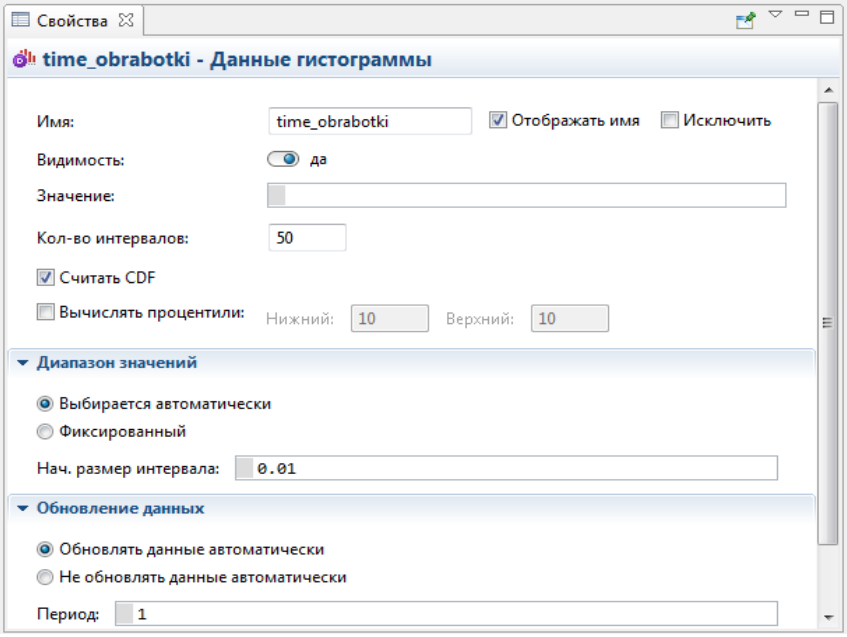


Рис. 1.39. Элемент сбора статистики о времени обработки запросов

Добавьте еще элемент сбора статистики для определения вероятности обработки запросов.

1. Перетащите элемент **Данные гистограммы** с палитры **Статистика** на диаграмму активного класса.

2. Задайте свойства элемента (рис. 1.40):

измените **Имя:** на `ver_obrabotki`;

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

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

Диаграмма после добавления элементов сбора статистики представлена на рис. 1.41.

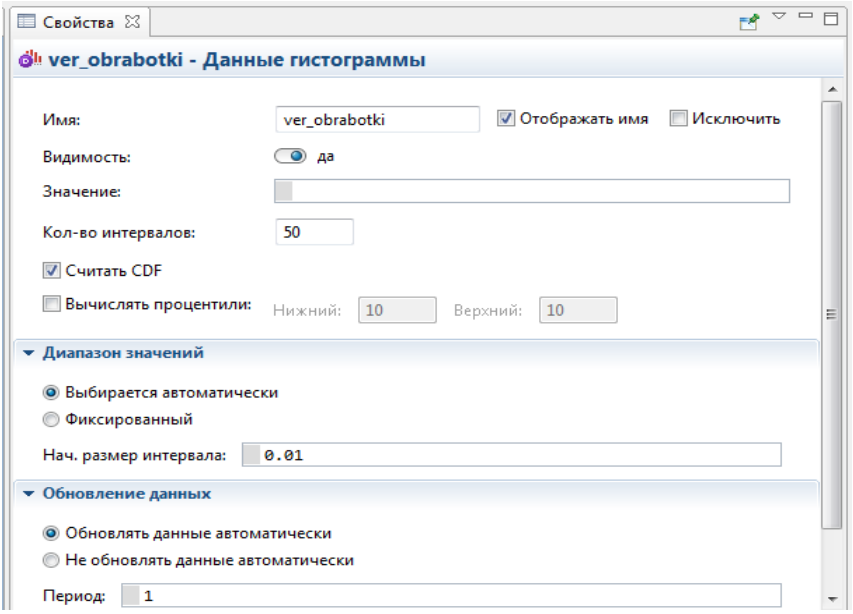


Рис. 1.40. Элемент сбора статистики о вероятности обработки запросов

1.7.3. 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`.

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

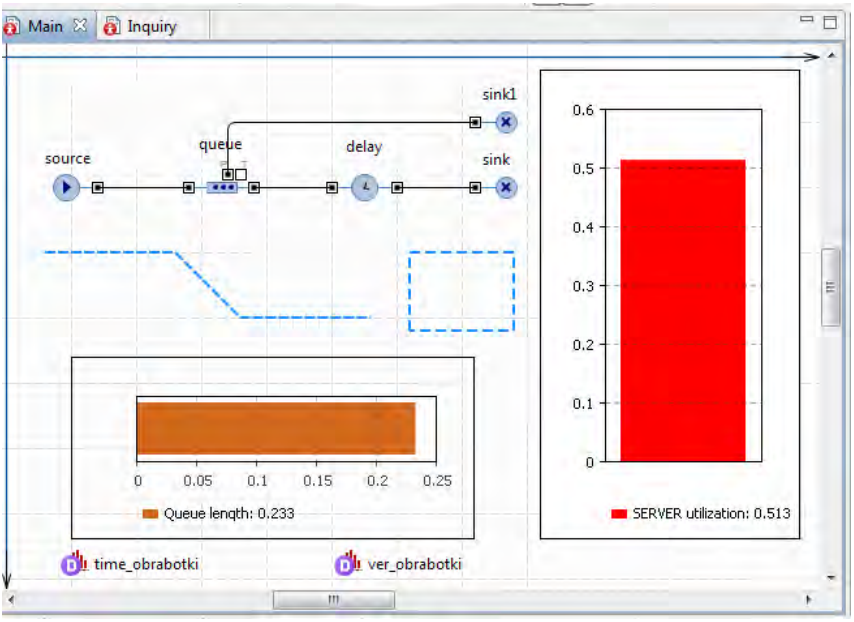


Рис. 1.41. Диаграмма после добавления элементов сбора статистики

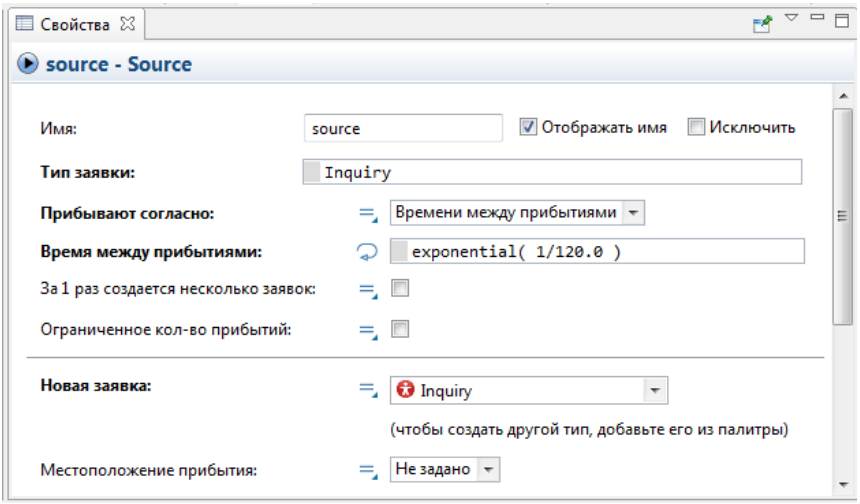


Рис. 1.42. Объект source с изменёнными свойствами

Измените их свойства.

1. Измените свойства объекта **source** (рис. 1.42):

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

выберите из выпадающего списка `Inquiry()` в поле **Новая заявка:**. Теперь этот объект будет создавать заявки нашего типа `Inquiry`;

введите `entity.time_vxod=time();` в поле **Действия При выходе:**. Код будет сохранять время создания заявки-запроса в параметре `time_vxod` нашего типа заявки `Inquiry`.

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

2. Измените свойства объекта queue:

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

3. Измените свойства объекта delay:

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

4. Измените свойства объекта sink1:

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

5. Измените свойства объекта sink:

введите `Inquiry` в поле **Тип заявки:**;

введите в поле **Действие при входе** следующие коды:

```
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.

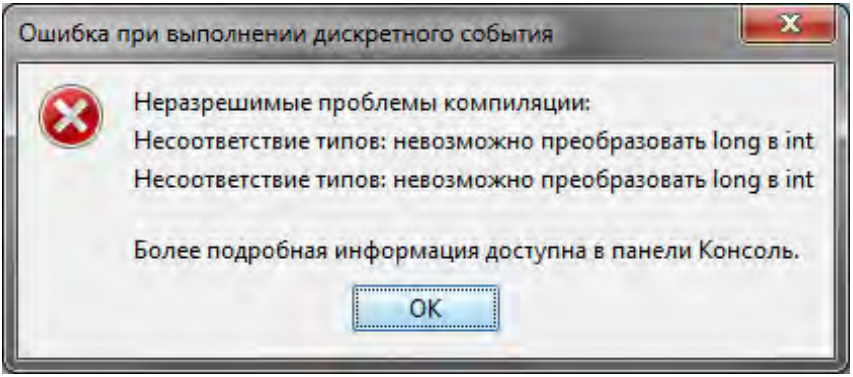


Рис. 1.43. Второе сообщение об ошибке

6. Запустите модель. Появится сообщение об ошибке. Щёлкните **Продолжить**. Появится второе сообщение (рис. 1.43).

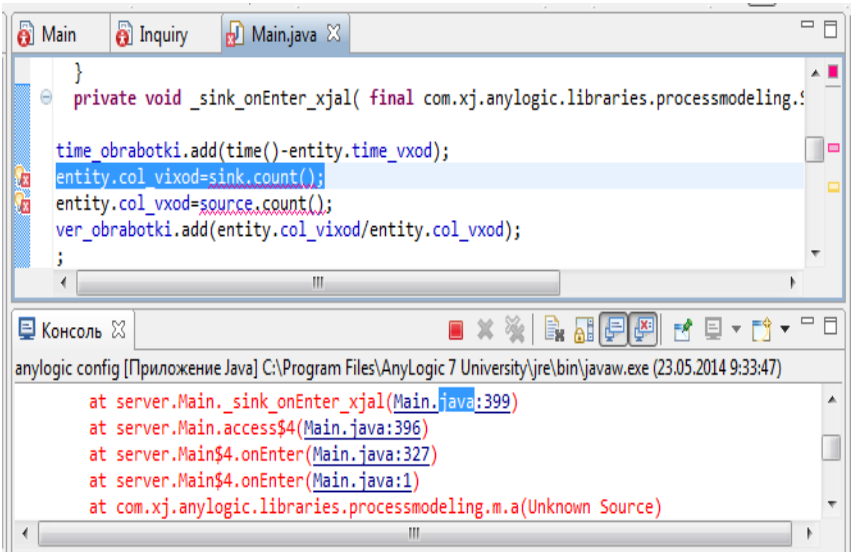


Рис. 1.44. Информация об ошибке в панели **Консоль**

7. Щёлкните выделенный Java код в панели **Консоль** `Main.java.399` (рис. 1.44). Появится код с выделенными ошибками.

Мы установили тип `int` для `col_vxod` и `col_vixod` (см. рис. 1.32). Изменим этот тип на `double`.

1.7.4. Удаление и добавление новых полей типа 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 Щёлкните кнопку `Inquiry`. Откроется окно Inquiry (см. рис. 1.38).
2. Удалите `time_vixod`, выделив его и нажав Delete.
3. Выделите `col_vxod`. Из выпадающего списка **Тип:** вместо `int` выберите `double` (рис. 1.45).
4. Выделите `col_vixod`. Из выпадающего списка **Тип:** вместо `int` выберите `double`.

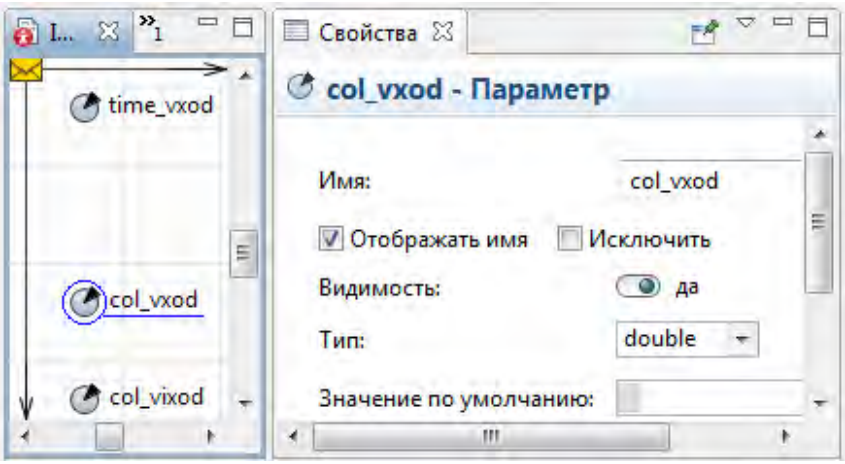


Рис. 1.45. Окно после удаления поля `time_vixod`

5. Из процедуры удаления поля следует, что так же можно вводить новые дополнительные поля нестандартного типа заявок. Например, перетащите из библиотеки **Основная** элемент **Параметр**. Дайте ему любое имя и установите **Тип:** из выпадающего списка. В дальнейшем это поле нестандартного типа заявки вы можете использовать в кодах модели.

Итак, все условия постановки задачи выполнены. Чтобы наблюдать за работой модели, установите, что время остановки модели не задано. Запустите модель. (рис. 1.46).

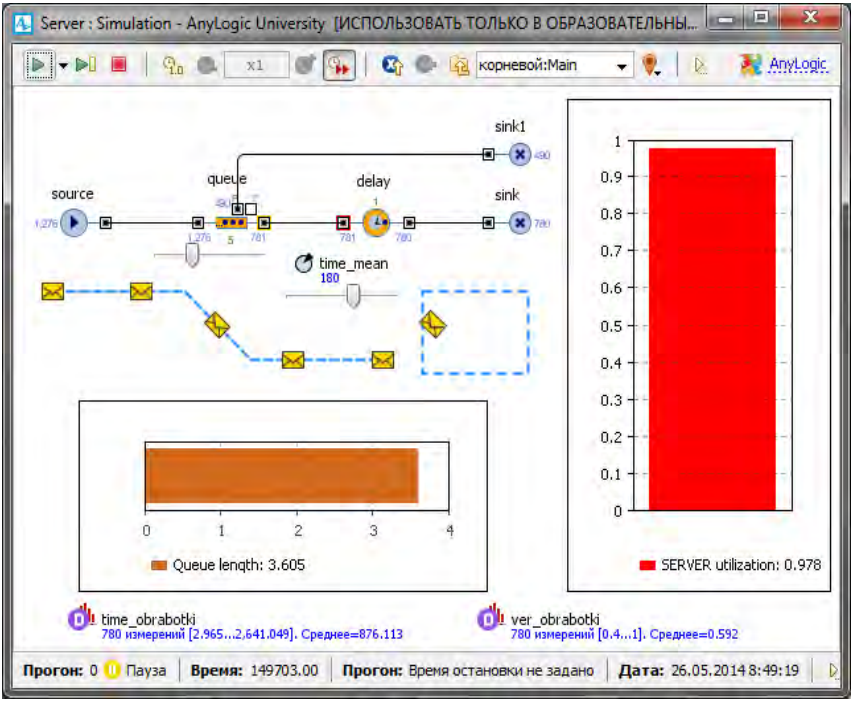


Рис. 1.46.Фрагмент работы модели

1.8. 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. 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 ниже объекта `delay`, чтобы было видно, к какому объекту относится параметр.
3. Перейдите на панель **Свойства** (рис. 1.47).
4. В поле **Имя** введите имя параметра `time_mean` (среднее время). По этому имени параметр будет доступен из кода.
5. Задайте тип параметра `double`.

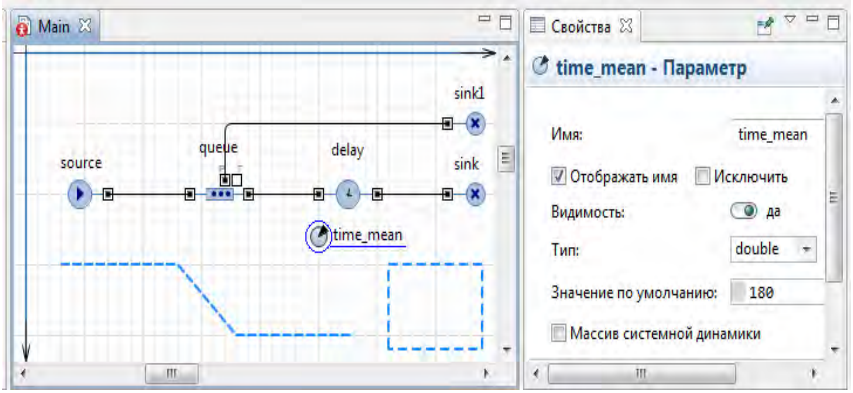


Рис. 1.47. Окно установки свойств элемента **Параметр**

6. В поле **Значение по умолчанию** установите 180. Если значение не задано явно, по правилам Java оно будет равно нулю.

7. Выделите объект `delay`.

8. На панели **Свойства** в поле **Время задержки** вместо выражения `exponential(1/180.0)` введите выражение `exponential(1/time_mean)`.

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

1. Откройте палитру **Элементы управления** и перетащите элемент **Бегунок** из палитры на диаграмму класса `Main` (рис. 1.48).

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 300. Поэтому введите 1 в поле **Минимальное значение:**, а 300 — в поле **Максимальное значение:** (рис. 1.49).

4. Установите флажок **Связать с:** и в активизированное поле введите `time_mean`.

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

1. Откройте палитру **Элементы управления** и перетащите элемент **Бегунок** из палитры на диаграмму класса `Main` (рис. 1.49).

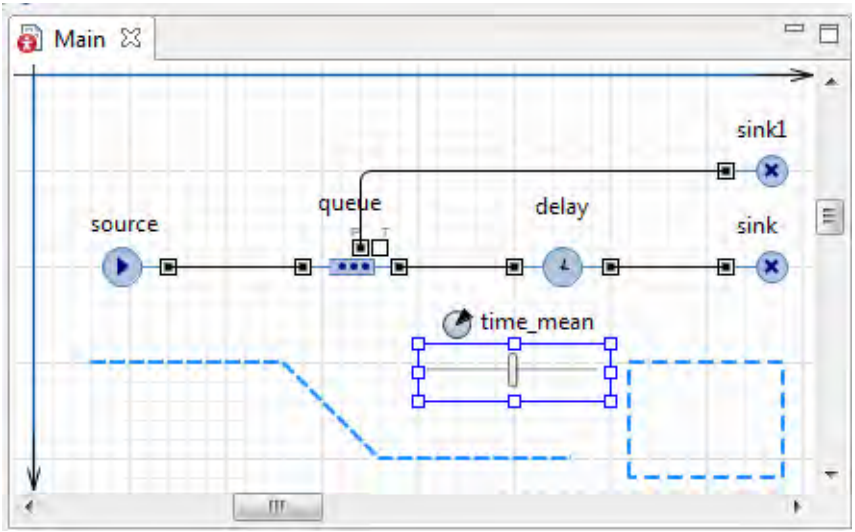


Рис. 1.48. Установка элемента управления **Бегунок**

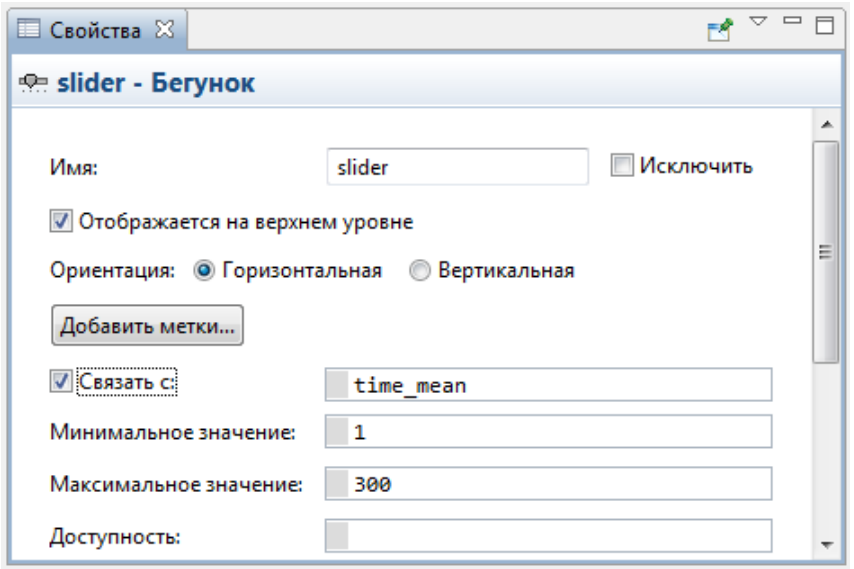


Рис. 1.49. Окно установки свойств элемента управления **Бегунок**

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 ёмкость 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`.

5. Запустите модель. Теперь вы можете изменять в процессе моделирования ёмкость 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. 1.50).

6. Остановите модель и перейдите на диаграмму класса Main.

Мы научились добавлять элементы **Параметр** и **Бегунок**. Но согласитесь, что какие параметры модели нужно будет менять, и в каких интервалах, заранее определить затруднительно. Также при использовании элемента **Бегунок** существуют трудности точного установления значения характеристики, так как невозможно предусмотреть нужный масштаб или цену деления **Бегунка**.

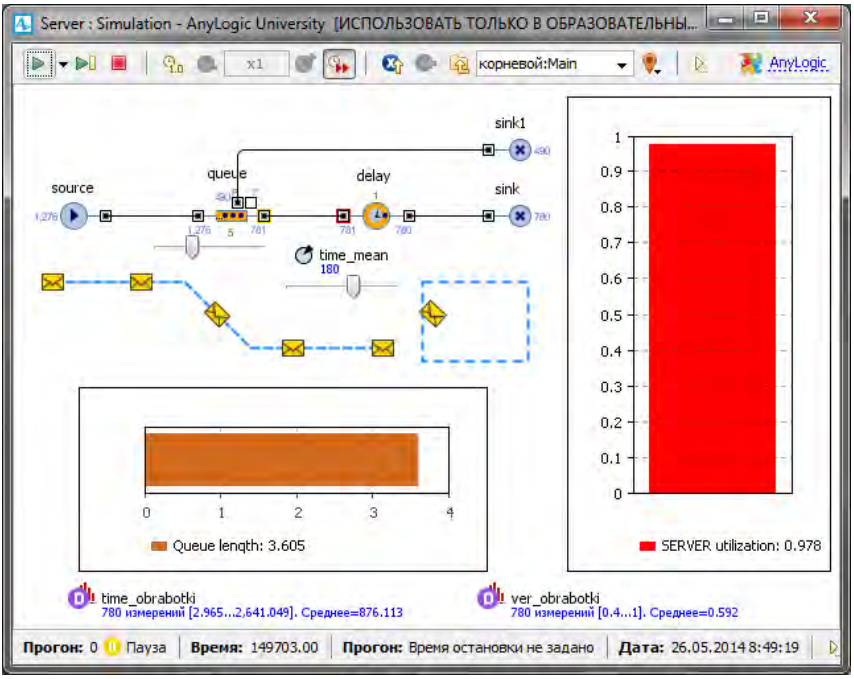


Рис. 1.50. Фрагмент работы модели с добавленным элементом **Параметр** и элементами управления

Существует и другой способ изменения свойств объектов во время выполнения модели: нужно щёлкнуть по элементу, войти в режим редактирования и ввести новое значение в одной из закладок всплывающего окна инспекта. Поэтому заранее не нужно продумывать, значения каких параметров планируется изменять, и не добавлять специальные элементы управления (например, бегунки).

Изменение значения в окне инспекта поддерживается для следующих элементов: простая переменная; параметр; накопитель.

И для следующих типов: численные; логический (boolean); текстовый (String).

1. Удалите элемент **Бегунок** для **Параметра** time_mean.
2. Запустите модель и приостановите её.
3. Щёлкните по значку **Параметра** time_mean.
4. Перейдите в режим редактирования (рис. 1.51).
5. Введите новое значение: `240.0`.
6. Закройте инспект. Рядом с элементом **Параметр** вы увидите введённое вами значение 240.0.
7. Запустите модель с новым свойством объекта delay.

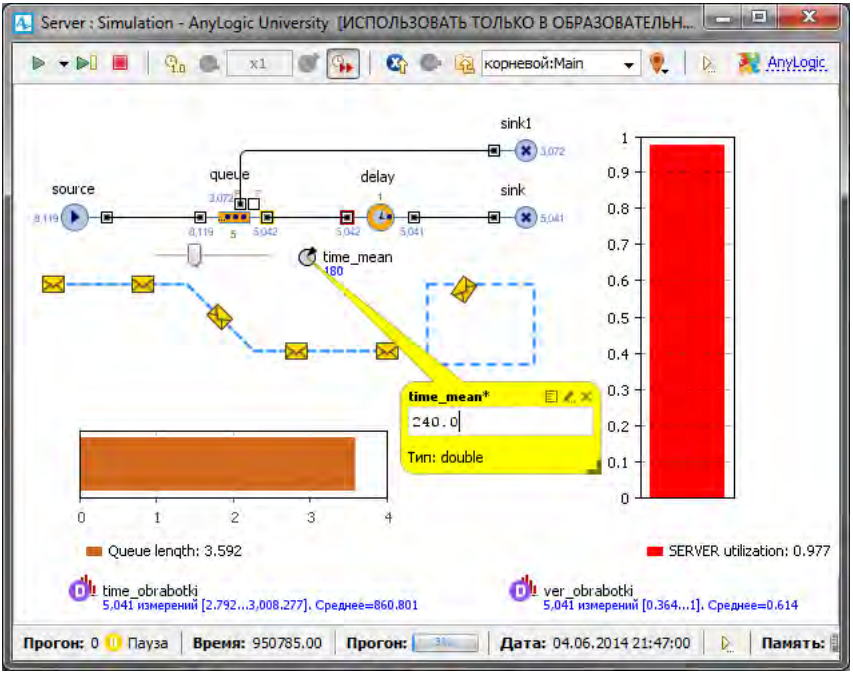


Рис. 1.51. Ввод нового значения параметра в окно инспекта

1.9. 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: щёлкните кнопку **Добавить данные** и введите в поле **Данные** имя соответствующего элемента: `time_obrabotki` (рис. 1.52). Установите **Отображать среднее**.
3. В поле **Заголовок:** введите `Histogram Time obrabotki`.

Добавим на диаграмму нашего потока гистограмму, которая будет отображать собранную вероятностную статистику.

1. Перетащите элемент **Гистограмма** из палитры **Статистика**.
2. Щёлкните кнопку **Добавить данные** и введите в поле **Данные** имя элемента: `ver_obrabotki`. Установите **Отображать среднее**.
3. В поле **Заголовок:** введите `Histogram Ver obrabotki`.
4. Запустите модель. Фрагмент работы показан на рис. 1.53.

Замечание. Обратите внимание, что после нового запуска модели time_mean=180, хотя ранее мы изменили его значение на 240.

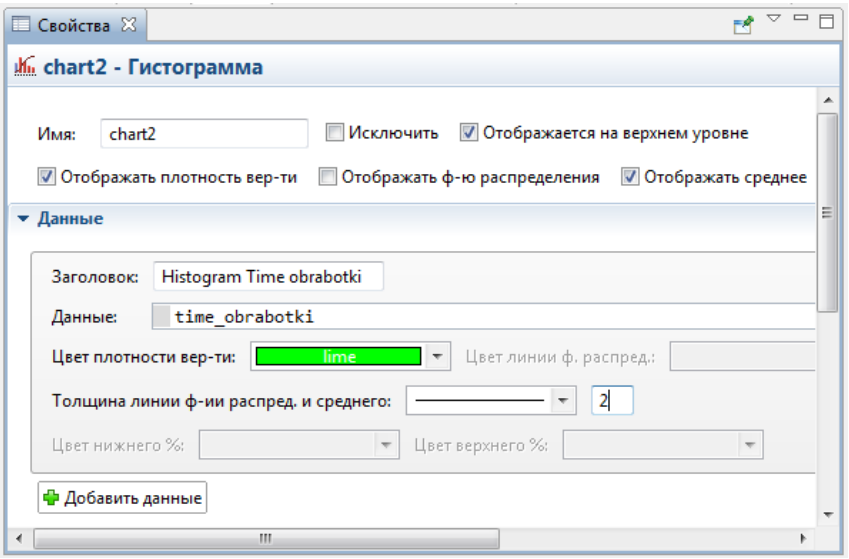


Рис. 1.52. Окно установки свойств элемента **Гистограмма**

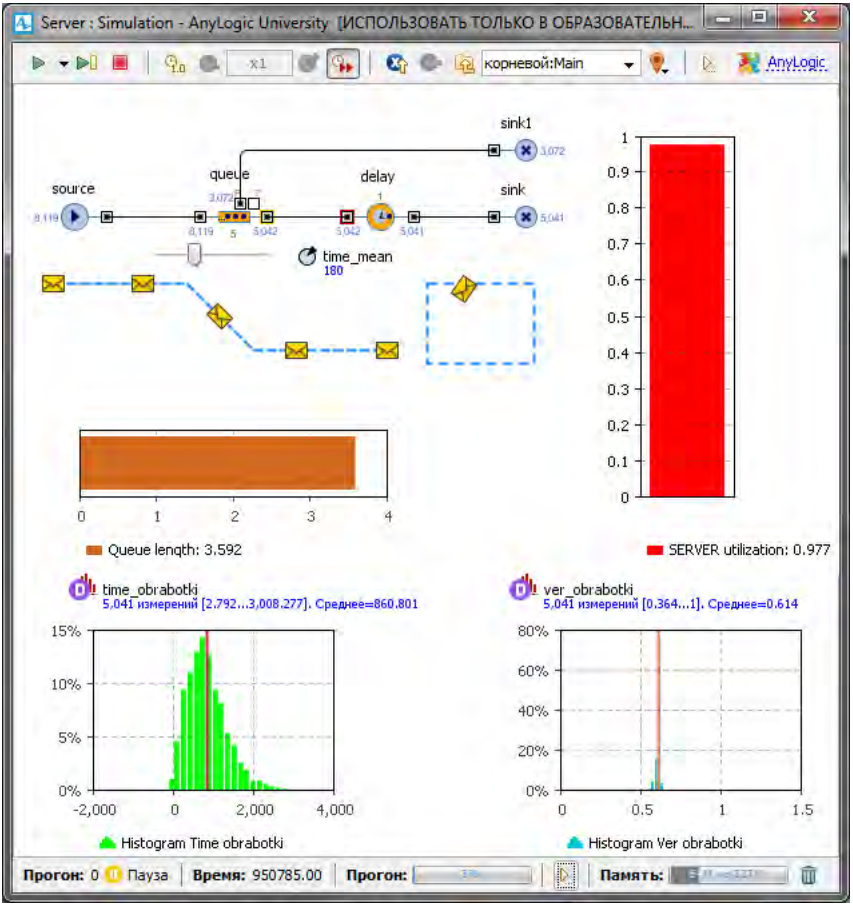


Рис. 1.53. Фрагмент работы модели с элементом управления и гистограммами

1.10. Изменение времени обработки запросов сервером

Построенная модель соответствует постановке задачи (п. 1.2.1). В ней, с целью упрощения процесса построения первой модели, время обработки запросов сервером было принято распределённым по показательному (экспоненциальному) закону со средним значением $T2 = 3$ мин.

Однако в модели время обработки поступающих запросов зависит от производительности сервера $Q = 6 \cdot 10^5$ оп/с и вычислительной сложности запросов, распределенной по нормальному закону

с математическим ожиданием $S1 = 6 \cdot 10^7$ оп и среднеквадратическим отклонением $S2 = 2 \cdot 10^5$ оп:

Кроме того, в модели определяется среднее количество запросов, обработанных за время моделирования 3600 с.

Внесите в модель изменения для аналогичного расчёта времени обработки запросов.

1. Удалите элемент **Параметр** с именем time_mean элемент **Бегунок** для элемента queue.
2. Из палитры **Основная** перетащите три элемента **Параметр** на диаграмму класса Main (рис. 1.54).
3. В поле **Имя** каждого из элементов введите `S1_`, `S2_` и `Q_` соответственно. Выберите **Тип** double.
4. В поле **Значение по умолчанию** каждого из элементов введите `60000000`, `200000` и `600000` соответственно.
5. Перетащите элемент **Переменная**.
6. В поле **Имя** укажите `KolZap`.
7. Выделите объект delay.
8. В поле **Время задержки** вместо exponential(1/time_mean) введите: `(normal(S2_,S1_))/Q_`
9. Выделите объект sink. В поле **Действие при входе** к имеющемуся там коду добавьте код:

```
KolZap=sink.in.count()/9604.0;
```

Для получения результатов моделирования с доверительной вероятностью $\alpha = 0,95$ и точностью $\varepsilon = 0,01$ нужно выполнить 9604 прогонов модели:

$$N = t_\alpha^2 \frac{p(1-p)}{\varepsilon^2} \approx 1,96^2 \frac{0,5^2}{0,01^2} \approx 9604,$$

где $t_\alpha = 1,96$ — табулированный аргумент функции Лапласа, p – ожидаемая вероятность исхода события, в данном случае вероятность обаботки запросов сервером.

Расчёт проведен для так называемого «худшего» случая, то есть в предположении, что ожидаемая вероятность обработки запросов $p = 0,5$.

Увеличим время моделирования в AnyLogic-модели в 9604 раз. А так как статистические данные о количестве обработанных запросов собираются за всё время моделирования, увеличенное в

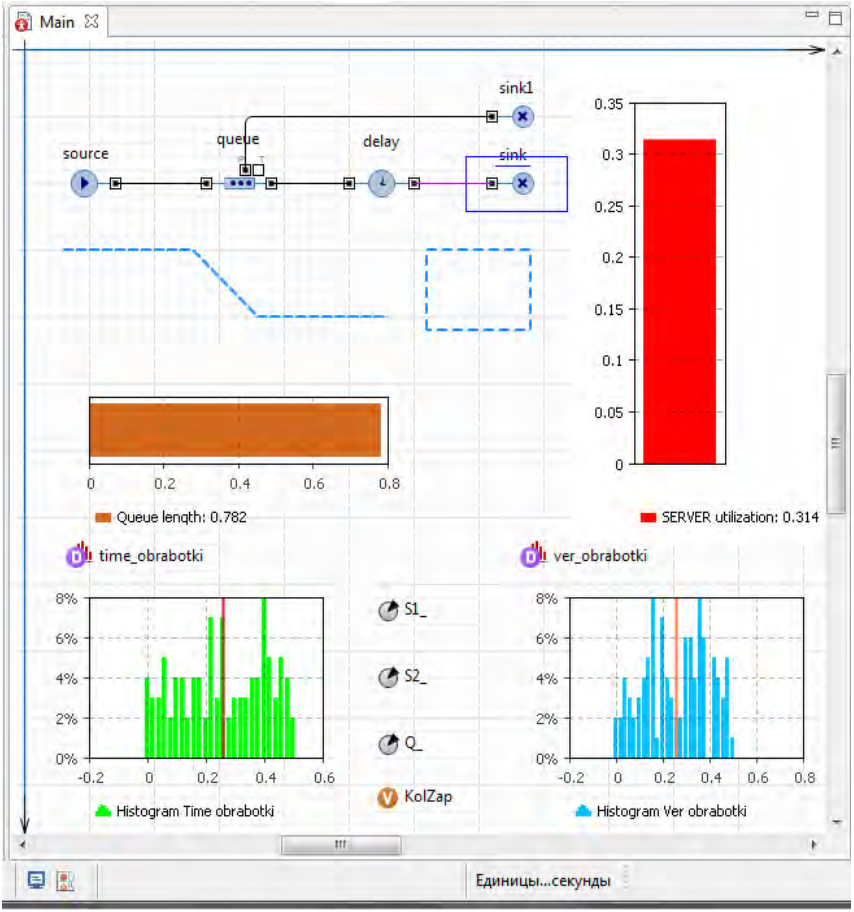


Рис. 1.54. Элементы AnyLogic-модели, соответствующие постановке

9604 раз, то для получения среднего значения это количество нужно разделить на 9604, что и предусмотрено в коде.

10. Показатели моделируемой системы нужно определить в течение 3600 с, поэтому время моделирования в AnyLogic составит 3600*9604 = 34574400 единиц модельного времени.

11. В панели **Проект** выделите Simulation. На странице **Модельное время** в поле **Установить** выберите **В заданное время**.

12. В поле **Конечное время** установите `34574400`.

13. Запустите модель и дождитесь окончания моделирования.

Результаты моделирования приведены на рис. 1.55.

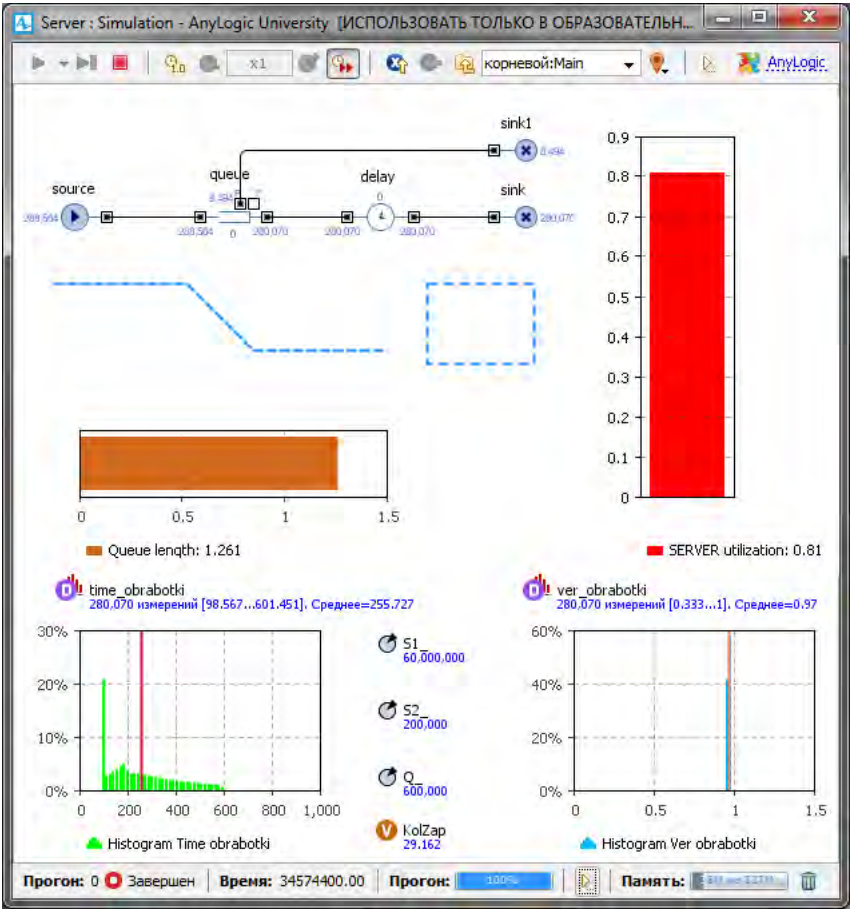


Рис. 1.55. Результаты моделирования обработки запросов сервером

1.11. Интерпретация результатов моделирования

Для проведения исследований на модели сделайте ещё несколько дополнений и изменений. Можно было бы обойтись и без них, но они улучшат эксплуатацию модели.

1. Из палитры **Основная** перетащите три элемента **Параметр** на диаграмму агента Main. Разместите их выше параметров S1_, S2_, Q_. Можно перетащить один параметр, скопировать его и вставить остальные два. Если установить свойства первого элемента **Параметр**, то они будут и в скопированных элементах. Не будем их устанавливать. Установим после копирования и вставки.

2. Выделите первый элемент **Параметр**. В поле **Имя:** первого параметра введите `timeMean` — среднее время поступления запросов для обработки на сервере. Оставьте тип double.

3. В поле **Значение по умолчанию** введите `120`.

4. Выделите объект source. В поле **Время между прибытиями** вместо 120.0 введите `timeMean`. Теперь вам при изменении среднего времени поступления запросов не придётся искать нужный код в свойствах объекта модели.

5. Выделите второй элемент **Параметр**. В поле **Имя:** введите `kolProg` — количество прогонов модели. Как и в предыдущем случае, при корректировке числа прогонов код в свойствах искать будет не нужно. Оставьте тип double.

6. В поле **Значение по умолчанию** введите `9604`.

7. Выделите объект sink.

8. В поле **Действие при входе** в имеющемся там коде замените последнюю строку следующей:

```
KolZap=round(sink.in.count()/kolProg);
```

На рис. 1. количество обработанных запросов KolZap сервером выдаётся с дробной частью. Теперь, вследствие применения процедуры round, количество запросов будет целым.

Как уже отмечалось, при изменении количества прогонов модели не надо будет искать код для требуемой корректировки. Однако всё-таки потребуется изменить модельное время. Например, если нужно выполнять с моделью 1000 прогонов, то модельное время следует установить равным 3600*1000=3600000.

9. Выделите третий элемент **Параметр**. В поле **Имя:** введите `emkBuf` — ёмкость в сообщениях входного буфера сервера. Установите тип `int`.

10. В поле **Значение по умолчанию** введите `5`.

11. Выделите объект queue. В поле **Вместимость** введите `emkBuf`.

12. Запустите модель. Результаты моделирования на рис. 1.56.

Проведём несколько экспериментов. Будем изменять среднее время поступления запросов timeMean в предположении, что с течением времени количество источников запросов будет расти, то есть timeMean будет уменьшаться. В сторону увеличения будем изменять ёмкость входного буфера emkBuf и производительность Q_ сервера.

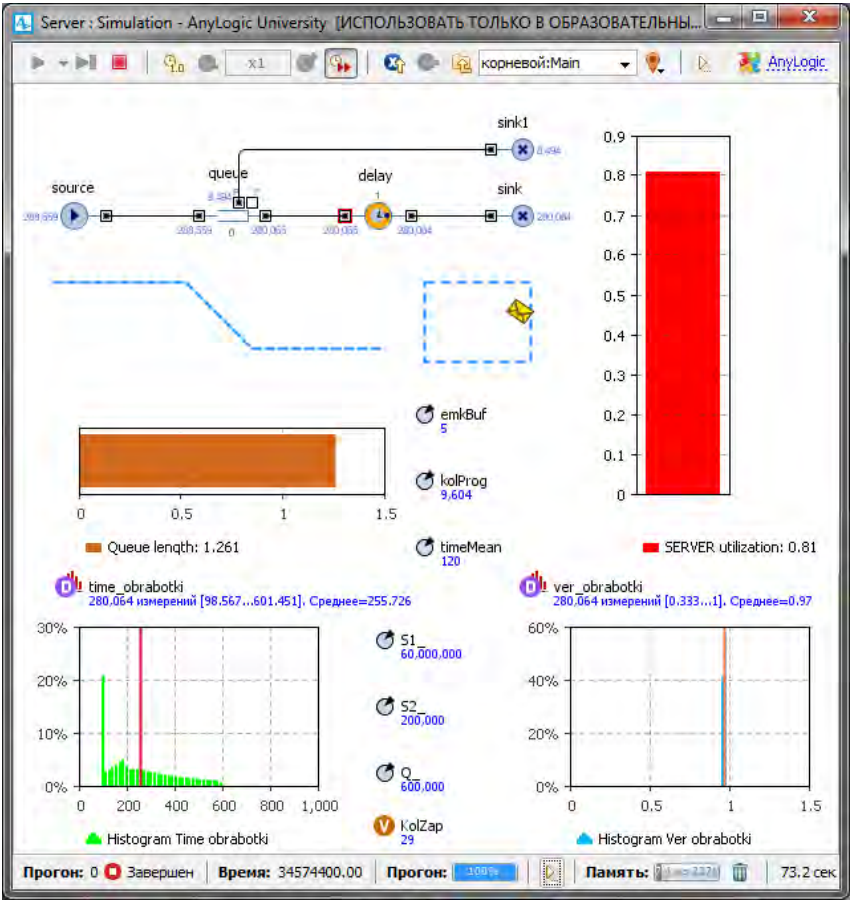


Рис. 1.56. Результаты моделирования при условиях постановки задачи

Результаты экспериментов приведены в табл. 1.1.

Из экспериментов 1 и 2 следует, что при увеличении ёмкости входного буфера в два раза вероятность обработки запросов увеличивается незначительно на 0,025, то есть количество обработанных запросов практически одно и тоже. Среднее время обработки одного запроса возрастает в 1,27 раза вследствие увеличения длины очереди в 1,48 раза.

Увеличение интенсивности поступления запросов в три раза при увеличении ёмкости входного буфера в два раза (эксперименты 3 и 4) также не даёт существенного увеличения количества обработанных запросов: 36 вместо 30.

Таблица 1.1

Показатели обработки запросов сервером

Показатели	GPSS World	AnyLogic6	AnyLogic7
1) timeMean = 120, emkBuf = 5			
Количество обработанных запросов	29	29	29
Вероятность обработки запросов	0,970	0,971	0,970
Среднее время обработки одного запроса	255,262	254,942	255,727
Средняя длина очереди запросов к серверу		1,254	1,261
Коэффициент использования сервера	0,810	0,810	0,810
2) timeMean = 120, emkBuf = 10			
Количество обработанных запросов	29	30	30
Вероятность обработки запросов	0,995	0,995	0,995
Среднее время обработки одного запроса	328,328	321,540	325,017
Средняя длина очереди запросов к серверу		1,841	1,870
Коэффициент использования сервера	0,833	0,831	0,831
3) timeMean = 40, emkBuf = 5			
Количество обработанных запросов	36	36	36
Вероятность обработки запросов	0,400	0,399	0,400
Среднее время обработки одного запроса	1055,335	1055,108	554,983
Средняя длина очереди запросов к серверу		4,552	4,550
Коэффициент использования сервера	1,00	1,00	1,00
4) timeMean = 40, emkBuf = 10			
Количество обработанных запросов	36	36	36
Вероятность обработки запросов	0,399	0,399	0,400
Среднее время обработки одного запроса	591,860	591,304	1054,907
Средняя длина очереди запросов к серверу		9,551	9,549
Коэффициент использования сервера	1,00	1,00	1,00
5) timeMean = 40, emkBuf = 10, Q_ = 1000000			
Количество обработанных запросов	59	60	60
Вероятность обработки запросов	0,666	0,665	0,667
Среднее время обработки одного запроса	591,860	591,304	591,129

Окончание табл. 1.1

Показатели	GPSS World	AnyLogic6	AnyLogic7
Средняя длина очереди запросов к серверу		8,855	8,852
Коэффициент использования сервера	1,00	1,00	1,00
6) timeMean = 40, emkBuf = 15, Q_ = 2000000			
Количество обработанных запросов	90	90	90
Вероятность обработки запросов	1,00	1,00	1,00
Среднее время обработки одного запроса	74,859	75,049	75,096
Средняя длина очереди запросов к серверу		1,096	1,128
Коэффициент использования сервера	0,751	0,750	0,751

При этом уменьшается вероятность обработки запросов в 2,5 раза, а время обработки одного запроса возрастает в 3,25 раза. Возросла и средняя длина очереди запросов к серверу

Коэффициент использования сервера равен 1. Из этого следует, что добиться увеличения вероятности и количества обработанных запросов можно только увеличением производительности сервера.

В экспериментах 5 и 6 увеличена производительность сервера до 1000000 и 2000000 оп/с соответственно.

По сравнению с экспериментом 1 количество обработанных запросов в эксперименте 5 увеличилось в 2 раза, а в эксперименте 6 — в 3 раза (вероятность обработки запросов равна 1). Среднее время обработки одного запроса всё равно примерно в 2 раза больше в эксперименте 5, а в эксперименте 6 — в 3,4 раза меньше. Можно полагать, что такая разница в среднем времени обработки одного запроса вызвана тем, что в эксперименте 5 средняя длина очереди запросов к серверу 8,852, а в эксперименте 6 — 1,128, то есть в 7,85 раза меньше.

Коэффициент использования сервера в эксперименте 6 равен 0,751. Из этого следует, что дальнейщее увеличение производительности сервера приведёт к уменьшению длины очереди и среднего времени обработки одного запроса.

Машинное время выполнения модели в AnyLogic примерно 70…90 с.

ГЛАВА 2. МОДЕЛЬ ПРОЦЕССА ИЗГОТОВЛЕНИЯ В ЦЕХЕ ДЕТАЛЕЙ

2.1. Постановка задачи

Изготовление в цехе детали начинается через случайное время $T_{\text{п}}$. Выполнению операций предшествует подготовка. Длительность подготовки зависит от качества заготовки, из которой будет сделана деталь. Всего различных видов заготовок n_1. Время подготовки подчинено экспоненциальному закону. Частота появления различных заготовок и средние значения времени их подготовки заданы табл. 2.1 дискретного распределения:

Таблица 2.1

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 со средними временами $T_1, T_2, ..., T_n$ соответственно. После каждой операции в течение времени $T_{\text{к}1}, T_{\text{к}2}, ..., T_{\text{к}n}$ следует контроль. Время выполнения операций и контроля — случайное. Контроль не проходят $q_1, q_2, ..., q_n$ % деталей соответственно.

Забракованные детали поступают на пункт окончательного контроля и проходят на нем проверку в течение времени, распределённого по экспоненциальному закону со средним значением $T_{\text{к}}$. В результате из общего количества не прошедших контроль деталей q_{n+1} % идут в брак, а оставшиеся $(1-q_{n+1})\%$ деталей подлежат повторному выполнению операций, после которых они не прошли контроль. Если деталь во второй раз не проходит контроль, она окончательно бракуется.

2.1.1. Исходные данные

$n_1 = 6$; $\text{Exponential}(T_{\text{п}}) = \text{Exponential}(30)$; $q_1 = 12\ \%, q_2 = 15\ \%$;

$n = 3$; $\text{Exponential}(T_1) = \text{Exponential}(30)$; $q_3 = 10\ \%, q_4 = 80\ \%$;

$\text{Exponential}(T_2) = \text{Exponential}(25)$; $\text{Exponential}(T_3) = \text{Exponential}(35)$;

$\text{Exponential}(T_{\text{к}1}) = \text{Exponential}(4)$; $\text{Exponential}(T_{\text{к}2}) = \text{Exponential}(5)$;

$\text{Exponential}(T_{\text{к}3}) = \text{Exponential}(15)$; $\text{Exponential}(T_{\text{к}}) = \text{Exponential}(8)$.

2.1.2. Задание на исследование

Разработать имитационную модель для определения оценки математического ожидания количества деталей, изготовленных цехом в течение 8 часов.

Модель должна также позволять определять относительное количество готовых и забракованных деталей, среднее время изготовления одной детали.

Результаты моделирования необходимо получить с точностью $\varepsilon = 0{,}01$ и доверительной вероятностью $\alpha = 0{,}99$.

2.1.3. Уяснение задачи на исследование

Процесс изготовления в цехе деталей представляет собой процесс, протекающий в многофазной разомкнутой системе массового обслуживания с ожиданием (рис. 2.1). Есть также признаки замкнутой системы — потоки брака для повторной обработки.

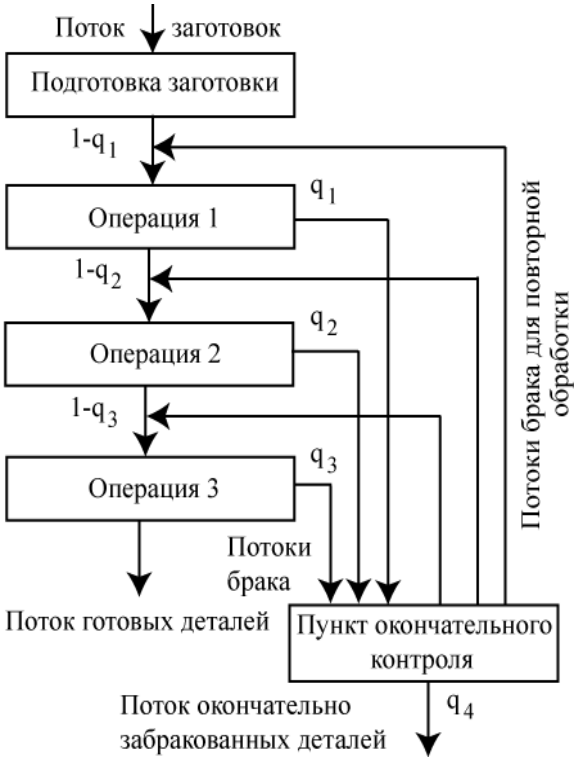


Рис. 2.1. Цех как система массового обслуживания

Представим, что подготовка заготовки и операции 1, 2 и 3 производятся на станках — одноканальных устройствах (ОКУ) 1, 2, 3 и 4 соответственно. Пункт окончательного контроля можно также представить ОКУ. Необходимые для их имитации средства GPSS приведены на рис. 2.1.

Время подготовки заготовки и время выполнения операций даны в мин. Возьмём 1 ед. мод. вр. = 1мин.

Рассчитаем количество прогонов, которые нужно выполнить в каждом наблюдении. При этом примем относительное количество готовых деталей как ожидаемую вероятность. Поскольку она заранее не известна, то расчёт проведём для худшего случая:

$$N = t_{\alpha}^{2} \cdot \frac{\sigma^{2}}{\varepsilon^{2}} = 2,58^{2} \cdot \frac{0,5(1-0,5)}{0,01^{2}} = 6,656\frac{0,25}{0,0001} \approx 16641.$$

2.2. Модель в AnyLogic

AnyLogic-модель процесса изготовления в цехе деталей будет включать согласно представлению как система массового обслуживания (рис. 2.1) следующие сегменты:

исходные данные;
подготовка заготовки;
операция 1;
операция 2;
операция 3;
пункт окончательного контроля;
склад готовых деталей;
склад бракованных деталей.
результаты моделирования.

2.2.1. Исходные данные. Использование массивов

Для ввода исходных данных используем элементы **Параметр**.

1. Выполните команду **Файл/Создать/Модель** на панели инструментов.
2. В поле **Имя модели** диалогового окна **Новая модель** введите `Изготовление_в_цехе_деталей`. Выберите каталог, в котором будут сохранены файлы вашей модели.
3. Щёлкните кнопку **Готово**. Создайте область просмотра для размещения элементов сегмента **Цех**.
4. Из палитры **Презентация** перетащите элемент **Область просмотра**.

5. На панели **Свойства** в поле **Имя:** введите `цех`.

6. Задайте **Выравнивать по:** `Верхн. левому углу`.

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

8. На странице **Местоположение и размер** введите в поля:

X: `0`, **Y:** 0, **Ширина:** `780`, **Высота:** `530`.

9. Перетащите элемент **Скруглённый прямоугольник**. На странице **Местоположение и размер** установите:

X: `280`, **Y:** `300`, **Ширина:** `250`, в поле **Высота:** `220`.

10. Перетащите элемент **text** и на странице **Основные** панели **Свойства** в поле **Текст:** введите `Исходные данные`.

11. В **Палитре** выделите **Основная**. Перетащите элементы **Параметр** на элемент с именем `Исходные данные`. Разместите их так, как показано на рис. 2.4. Значения свойств установите согласно табл. 2.3.

12. На рис. 2.4, как вы, наверное, уже заметили, два элемента **Параметр** отличаются от остальных. Они используются для ввода данных табл. 2.1 как одномерных массивов с именами верВарЗаг (вероятности вариантов заготовок) и срВрПодгЗаг (среднее время подготовки варианта заготовки).

Рис. 2.4. Размещение элементов **Параметр** для ввода исходных данных

Таблица 2.3

Элементы и их свойства					
Параметр			Параметр		
Имя	Тип	Значение по умолчанию	Имя	Тип	Значение по умолчанию
Tn	double	35	Tk1	double	4
T1	double	30	Tk2	double	5
T2	double	25	Tk3	double	15
T3	double	35	Tk	double	8
q1	double	0.12	врМод	double	480
q2	double	0.15	колПрог	double	16641
q3	double	0.10			
q4	double	0.80			

Создайте размерности массивов. В данном случае они одинаковые. Элементов в одной строке табл. 2.1 шесть. Предположим, что число видов заготовок может увеличиться до 10. Значит размерность одного массива 10 элементов.

1. Щёлкните правой кнопкой мышки в панели **Проекты** и в контекстном меню выберите **Создать/Размерность**.
2. В открывшемся окне **Размерность** в поле **Имя** введите `КолВарЗаг`.
3. Установите **Тип размерности: Диапазон**.
4. В открывшееся поле **Диапазон:** введите `1-10`.
5. Щёлкните **Готово**.

Теперь создайте непосредственно массивы. Начните с массива `верВарЗаг` для вероятностей появления видов заготовок.

1. Из Палитры **Основные** перетащите элемент **Параметр**.
2. На странице **Основные** панели **Свойства** в поле **Имя:** введите `верВарЗаг`. **Тип:** `double`.
3. Установите флажок **Массив**. Откроется окно **Размерности** (рис. 2.5). Щёлкните по расположенной справа от окна и подсвеченной зелёным кнопке.
4. Откроется окно **Edit dimensions** (рис. 2.6). В окошке **Возможные размерности:** выделите `КолВарЗаг`.
5. Щёлкните по кнопке . Размерность `КолВарЗаг` появится в окошке **Выбранные размерности**.
6. Закройте окно. Вы вернётесь на панель **Свойства**. В окошке **Размерности** вы увидите размерность `КолВарЗаг`.

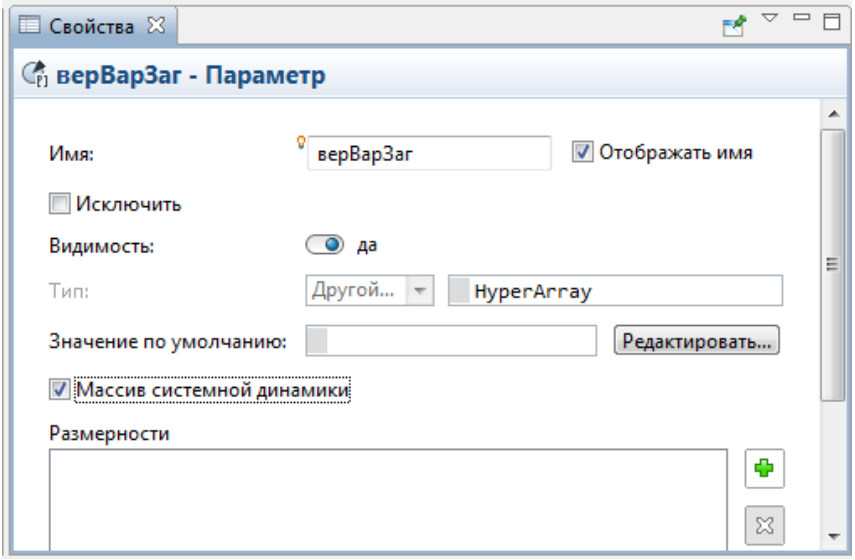


Рис. 2.5. Окно **Параметр**-массив

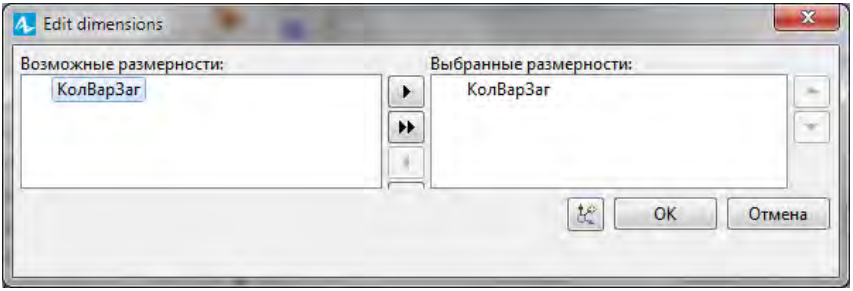


Рис. 2.6. Окно **Выбор размерности**

7. Щёлкните **Редактировать значения массива**. Откроется одноимённое диалоговое окно (2.7).

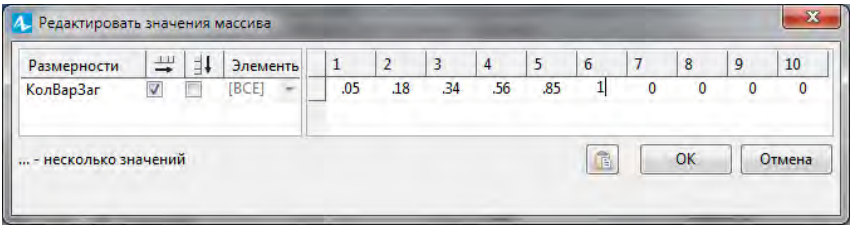


Рис. 2.7. Окно **Редактирование значений массива**

8. В левой части окна стрелками показано размещение элементов массива. Оставим горизонтальное. Элементы массива имеют разные значения. Поэтому не используем **[ВСЕ]**.

9. В правой части окна введите значения элементов массива:

```
0.05,0.18,0.34,0.56,0.85,1,0,0,0,0
```

Обратите внимание, что данные из первой строки табл. 2.1 введены в порядке возрастания, причём, второй элемент = первый элемент табл. 2.1 + второй табл. 2.1, третий = второй + третий табл. 2.1 и т.д. Эта особенность будет учтена в последующем программном коде. Хотя можно было бы ввести и так, как в табл. 2.1.

10. Щёлкните ОК. Вы вернётесь на панель **Свойства**. В поле **Значение по умолчанию:** появятся введённые вами значения шести элементов массива. Остальные четыре элемента равны нулю. Обратите также внимание на то, что элементы массива заключены в фигурные скобки {…}.

11. Аналогичным образом создайте второй массив с именем `срВрПодгЗаг` для среднего времени подготовки заготовки.

12. В поле **Значение по умолчанию:** должно быть:

```
{10,14,21,22,28,25,0,0,0,0}
```

2.2.2. Построение событийной части модели

В событийную (функциональную) часть модели включим указанные ранее сегменты. Поскольку построение модели это итерационный процесс, то размещение сегментов и объектов AnyLogic будем корректировать до тех пор, пока не посчитаем достаточным их взаимное расположение для корректной с нашей точки зрения работы модели и её презентации.

Начнём с сегмента имитации процесса подготовки заготовки.

2.2.2.1. Подготовка заготовки

Данный сегмент предназначен для имитации поступления заготовки, ожидания в очереди, имитации непосредственно подготовки заготовки и отправки на выполнение первой операции.

1. В **Палитре** выделите **Презентация**. Перетащите элемент **Прямоугольник**. На странице **Местоположение и размер** установите: **X:** `20`, **Y:** `300`, **Ширина:** `240`, в поле **Высота:** `150`.

2. Перетащите элемент **text** и на странице **Текст** панели **Свойства** введите: `Подготовка заготовки` (рис. 2.8).

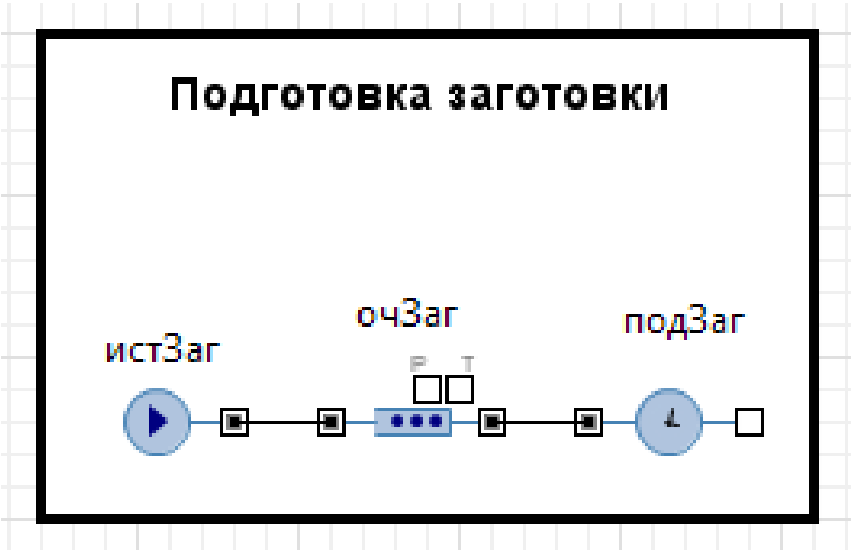


Рис. 2.8. Сегмент **Подготовка заготовки**

3. В **Палитре** выделите **Библиотека моделирования процессов**. Перетащите объект **source** на агента Main и разместите в прямоугольнике с именем `Подготовка заготовки`.

4. Для записи и хранения параметров детали в дополнительные поля заявок нужно создать нестандартный класс заявки. Создайте класс заявки `Detail`.

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

6. Появится диалоговое окно **Новый Java класс**. В поле **Имя:** введите имя нового класса `Detail`.

7. В поле **Базовый класс:** выберите из выпадающего списка `Entity` в качестве базового класса. Щёлкните кнопку **Далее**.

8. Появится вторая страница **Мастера создания Java класса**. Добавьте следующие поля Java класса, которые потребуются в дальнейшем при разработке модели:

```
double n;
double a;
double Tn1;
```

Поле `n` будет использоваться для занесения номера выполняемой с деталью операции. Эти номера необходимы для определения

операций, на которые нужно повторно отправить детали с пункта окончательного контроля. Поле `a` предназначено для занесения кода первого или повторного выполнения операций. Этот код используется для того, чтобы при повторном обнаружении брака не отправлять для выполнения соответствующей оерации в третий раз.

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

10. Щёлкните кнопку **Готово**. Появится редактор кода и автоматически созданный код вашего Java класса. Закройте код.

11. Теперь нужно преобразовать Java класс в тип агента. Для этого щёлкните правой кнопкой мыши в панели **Проект** только что созданный Java класс и в контекстном меню выберите **Преобразовать Java класс в тип агента**.

12. Появится окно с автоматически созданными параметрами нестандартного типа заявок `Detail`.

13. Из Палитры **Основная** перетащите на сегмент **Исходные данные** элемент **Переменная**. На странице **Основные** панели **Свойства** дайте **Имя:** `b`, установите **Тип:** `double`.

14. Выделите объект **source**. На страницах панели **Свойства** установите свойства согласно рис. 2.9.

В коде, приведенном ниже, который вы ввели в поле свойства **Действия При выходе:**, используются данные созданных ранее двух массивов.

```
entity.n=uniform_pos();
if (entity.n <= верВарЗаг.get(1))
entity.Tn1=срВрПодгЗаг.get(1);
else if (entity.n > верВарЗаг.get(1) && entity.n
<= верВарЗаг.get(2)) entity.Tn1=срВрПодгЗаг.get(2);
else if (entity.n > верВарЗаг.get(2) && entity.n
<= верВарЗаг.get(3)) entity.Tn1=срВрПодгЗаг.get(3);
else if (entity.n > верВарЗаг.get(3) && entity.n
<= верВарЗаг.get(4)) entity.Tn1=срВрПодгЗаг.get(4);
else if (entity.n > верВарЗаг.get(4) && entity.n
<= верВарЗаг.get(5)) entity.Tn1=срВрПодгЗаг.get(5);
else if (entity.n > верВарЗаг.get(5) && entity.n
<=верВарЗаг.get(6)) entity.Tn1=срВрПодгЗаг.get(6);
```

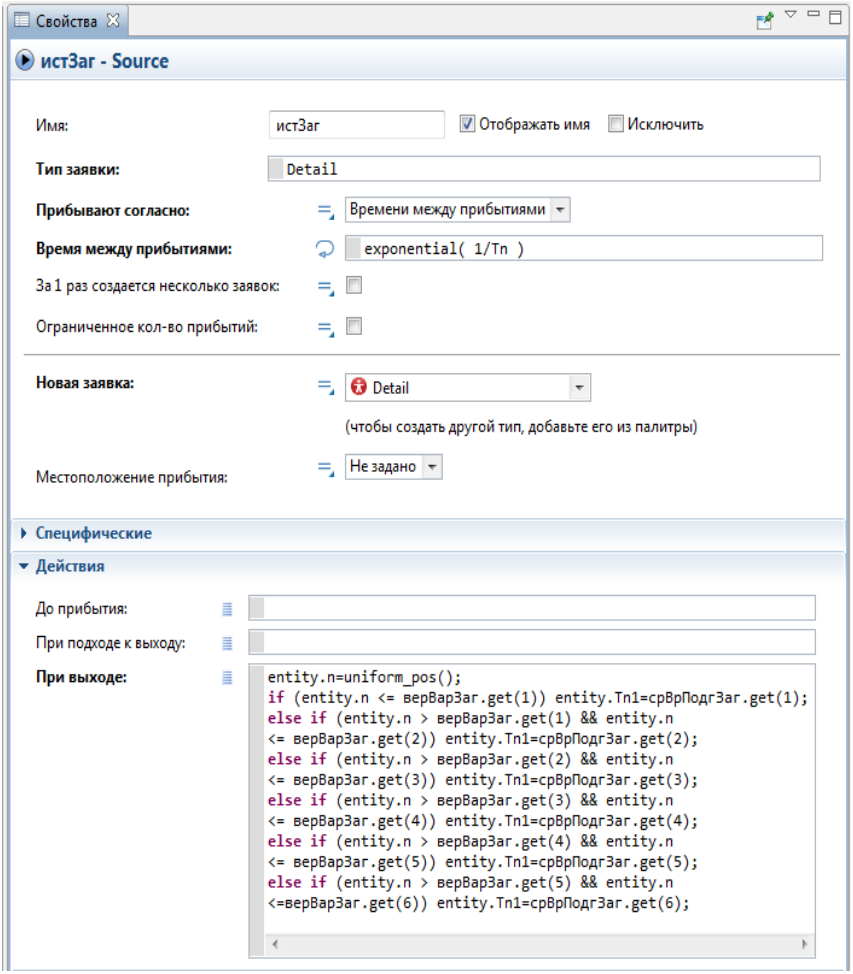


Рис. 2.9. Объект source с установленными свойствами

1. Выделите объект **queue** и на страницах панели **Свойства** установите свойства:

Имя: `очЗаг`

Тип заявки: `Detail`

2. **Максимальная вместимость:** установить флажок.

3. Выделите объект **delay** и на странице **Основные** панели **Свойства** установите свойства:

Имя: `подЗаг`

Тип заявки: `Detail`
Тип: Определённое время
Время задержки: `exponential (1/entity.Tn1)`
Вместимость: `1`
Действия При выходе: `entity.a = 1;`

2.2.2.2. Сегменты Операция 1, Операция 2, Операция 3

Каждый из сегментов операций 1, 2 и 3 предназначен для имитации выполнения соответствующей операции, включающей ожидание в очереди, непосредственно выполнение операции, контроль её качества, отправку на пункт окончательного контроля в случае брака, приём на повторное выполнение операции и контроль.

1. Из палитры **Презентация** Перетащите три элемента **Прямоугольник** так, как на рис. 2.10. На странице **Местоположение и размер** панели **Свойства** введите: для первого прямоугольника **X:** `20`, **Y:** `60;` для второго — **X:** `280`, **Y:** `60;` для третьго — **X:** `530`, **Y:** `60.` Для всех **Ширина:** `240`, **Высота:** `180`.
2. Перетащите три элемента **text** и на странице **Основные** панели **Свойства** в поле **Текст:** каждого из них введите `Операция 1`, `Операция 2`, `Операция 3` соответственно (рис. 2.10).
3. Из **Библиотеки моделирования процессов** перетащите для каждого сегмента два объекта **queue**, два объекта **delay** и один объект **selectOutput**, разместите и соедините как на рис. 2.7.
4. Выделите поочередно объекты, начиная с объекта **queue** сегмента **Операция 1**, и на странице **Основные** панели **Свойства** установите свойства согласно рис. 2.10 и табл. 2.4.

На сегменте **Операция 1** поясним принятые имена объектов: `очОп1` — очередь на операцию 1; `выпОп1` — имитация выполнения операции 1; `очКонОп1` — очередь на контроль после операции 1; `конОп11` — имитация контроля после операции 1; `конОп12` — розыгрыш результата контроля.

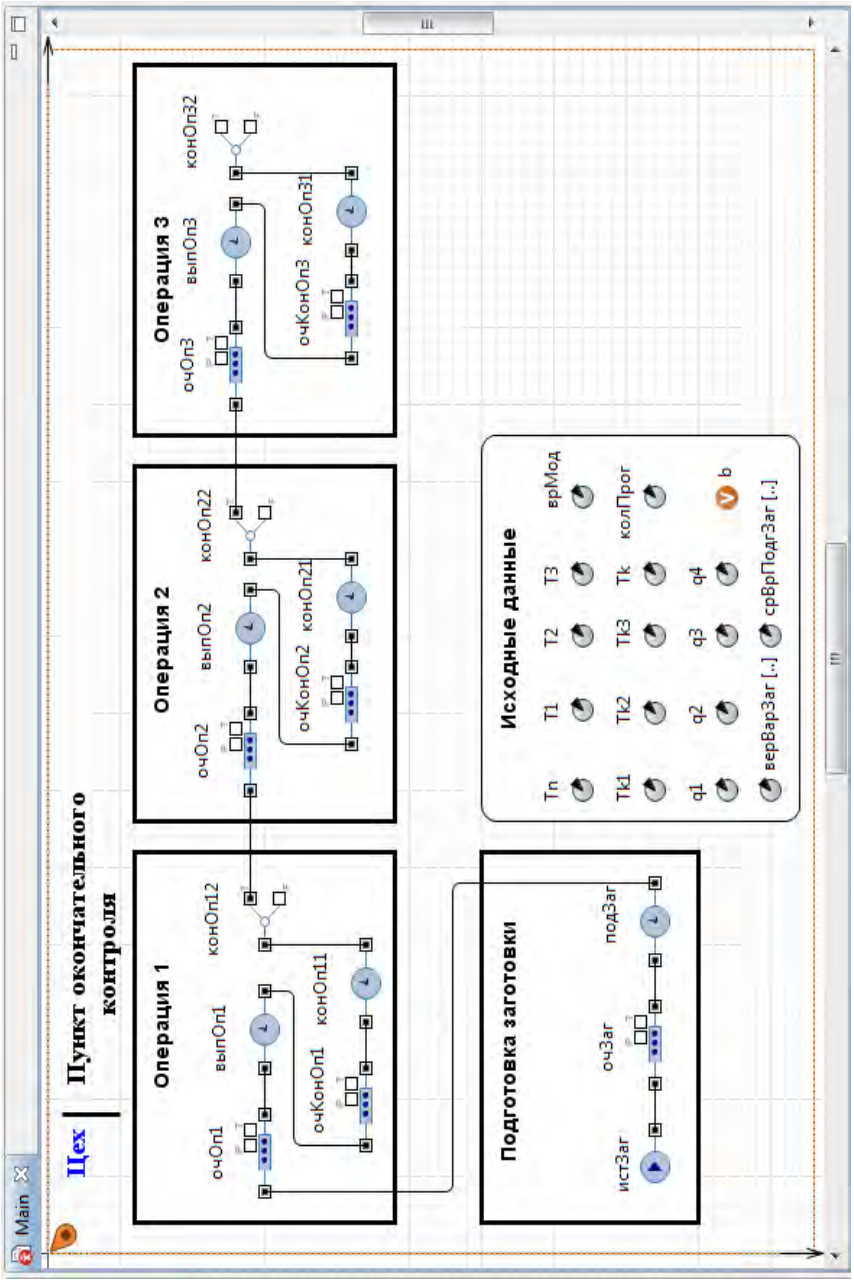


Рис. 2.10. Добавлены сегменты Операция 1, Операция 2, Операция 3

Таблица 2.4

Объект	Свойства	Значения
Сегмент **Операция 1**		
queue	Имя Тип заявки Максимальная вмести-мость	очОп1 Detail Установите флажок
delay	Имя Тип заявки Тип Время задержки Вместимость	выпОп1 Detail Определённое время exponential (1/T1) 1
queue1	Имя Тип заявки Максимальная вмести-мость	очКонОп1 Detail Установите флажок
delay1	Имя Тип заявки Тип Время задержки Вместимость **Действия При выходе**	конОп11 Detail Определённое время exponential (1/Tk1) 1 entity.n = 1;
selectOutput	Имя Тип заявки Выход true выбирается Вероятность	конОп12 Detail Заданной вероятностью 1-q1
Сегмент **Операция 2**		
queue	Имя Тип заявки Максимальная вмести-мость	очОп2 Detail Установите флажок
delay	Имя Тип заявки Тип Время задержки Вместимость	выпОп2 Detail Определённое время exponential (1/T2) 1
queue1	Имя Тип заявки Максимальная вмести-мость	очКонОп2 Detail Установите флажок

Окончание табл. 2.4

Объект	Свойства	Значения
delay1	Имя Тип заявки Тип Время задержки Вместимость **Действия При выходе**	конОп21 Detail Определённое время exponential (1/Tk2) 1 entity.n = 2
selectOutput	Имя Тип заявки Выход true выбирается Вероятность	конОп22 Detail Заданной вероятностью 1-q2
Сегмент **Операция 3**		
queue	Имя Тип заявки Максимальная вместимость	очОп3 Detail Установите флажок
delay	Имя Тип заявки Тип Время задержки Вместимость	выпОп3 Detail Определённое время exponential (1/T3) 1
queue1	Имя Тип заявки Максимальная вместимость	очКонОп3 Detail Установите флажок
delay1	Имя Тип заявки Тип Время задержки Вместимость **Действия При выходе**	конОп31 Detail Определённое время exponential (1/Tk3) 1 entity.n = 3
selectOutput	Имя Тип заявки Выход true выбирается Вероятность	конОп32 Detail Заданной вероятностью 1-q3

2.2.2.3. Создание нового активного объекта

На рис. 2.10 показаны четыре функциональных сегмента модели, которые построены с использованием 18 объектов AnyLogic. Если вы начнёте создавать очередной сегмент, то после перетаски-

вания второго объекта появится сообщение: *Ограничение ознакомительной версии: нельзя создавать более 20 вложенных объектов*. Поэтому остальные сегменты модели нам нужно разместить на втором агенте (активном объекте). Создайте этого агента.

1. На панели **Проекты** щёлкните правой кнопкой мыши Main, с которым вы работаете в данный момент, и выберите из контекстного меню **Создать/Тип агента**.
2. Откроется окно **Шаг 1. Создание нового типа агента**.
3. Оставьте **Не использовать шаблоны типов агентов**.
4. Задайте в поле **Имя нового агента:** `Kontrol`.
5. Если нужно, в поле **Описание**: введите описание сущности, моделируемой этим типом агента.
6. Щёлкните кнопку **Готово**.

2.2.2.4. Создание экземпляра нового типа агента

Согласно логике процесса изготовления деталей надо после выполнения каждой из трёх операций в случае брака отправить забракованные детали на пункт окончательного контроля. С последнего получить и отправить детали на повторное выполнение тех операций, после которых они были забракованы. Кроме того, готовые детали необходимо передать на склад готовых деталей. Таким образом, для связи с активным объектом Main потребуются семь портов (3+3+1).

Создайте экземпляр нового типа агента **Kontrol**.

1. Из Палитры **Основная** перетащите элемент **Порт** и разместите сверху в левом ряду (рис. 2.11).
2. На странице **Основные** панели **Свойства** имя port замените именем `наОп1`.
3. Скопируйте элемент **Порт** с именем `наОп1`.
4. Вставьте два элемента **Порт** (см. рис. 2.11). При вставке последовательно будут изменяться их имена: `наОп2`, `наОп3`.

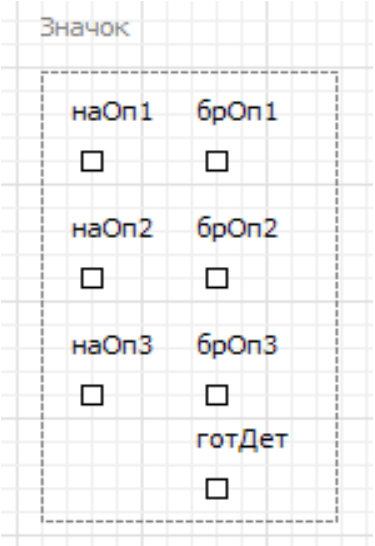


Рис. 2.11. Порты

5. Из Палитры **Основная** перетащите элемент **Порт** и разместите сверху в правом ряду (см. рис. 2.11).

6. На странице **Основные** панели **Свойства** имя port замените именем `брОп1` (идентификатор означает, что через этот порт отправляются бракованные детали после операции 1 на пункт окончательного контроля).

7. Скопируйте элемент **Порт** с именем `брОп1`.

8. Вставьте два элемента **Порт** (см. рис. 2.11).

9. Из Палитры **Основная** перетащите элемент **Порт** и разместите внизу в правом ряду (см. рис. 2.11).

10. На странице **Основные** панели **Свойства** имя port замените именем `готДет`.

11. По мере размещения элементов **Порт** они автоматически будут объединяться прямоугольником (с пунктирными линиями) и появится надпись `Значок`.

12. Возвратитесь на Main. На панели **Проект** выделите Kontrol, перетащите на тип агента Main экземпляр типа агента Kontrol, разместите как на рис. 2.12.

13. Экземпляр нового типа агента Kontrol создан. На странице **Основные** панели **Свойства** уберите флажок **Отображать имя** и в поле **Имя:** введите `на_контроль`.

14. В **Палитре** выделите **Презентация**. Перетащите элемент **Прямоугольник** и разместите как на рис. 2.12. На странице **Местоположение и размер** панели **Свойства** введите: **X:** `570`, **Y:** `270`, **Ширина:** `140`, **Высота:** `220`.

15. Перетащите элемент **text** и на странице **Основные** панели **Свойства** в поле **Текст** введите `На окончательный контроль`.

Теперь для движения заявок-деталей в модели необходимо надлежащим образом соединить входы и выходы объектов сегментов **Операция 1**, **Операция 2** и **Операция 3** с портами.

1. Соедините выходы F (false) объектов конОп12, конОп22, конОп32 с портами брОп1, брОп2, брОп3 соответственно.

2. Соедините порты наОп1, наОп2, наОп3 с входами объектов очОп1, очОп2, очОп3 соответственно.

3. Соедините выход T (true) объекта конОп32 сегмента **Операция 3** с портом готДет.

На рис. 2.12 вы видите ещё объекты. Перейдём к размещению их на агенте Main.

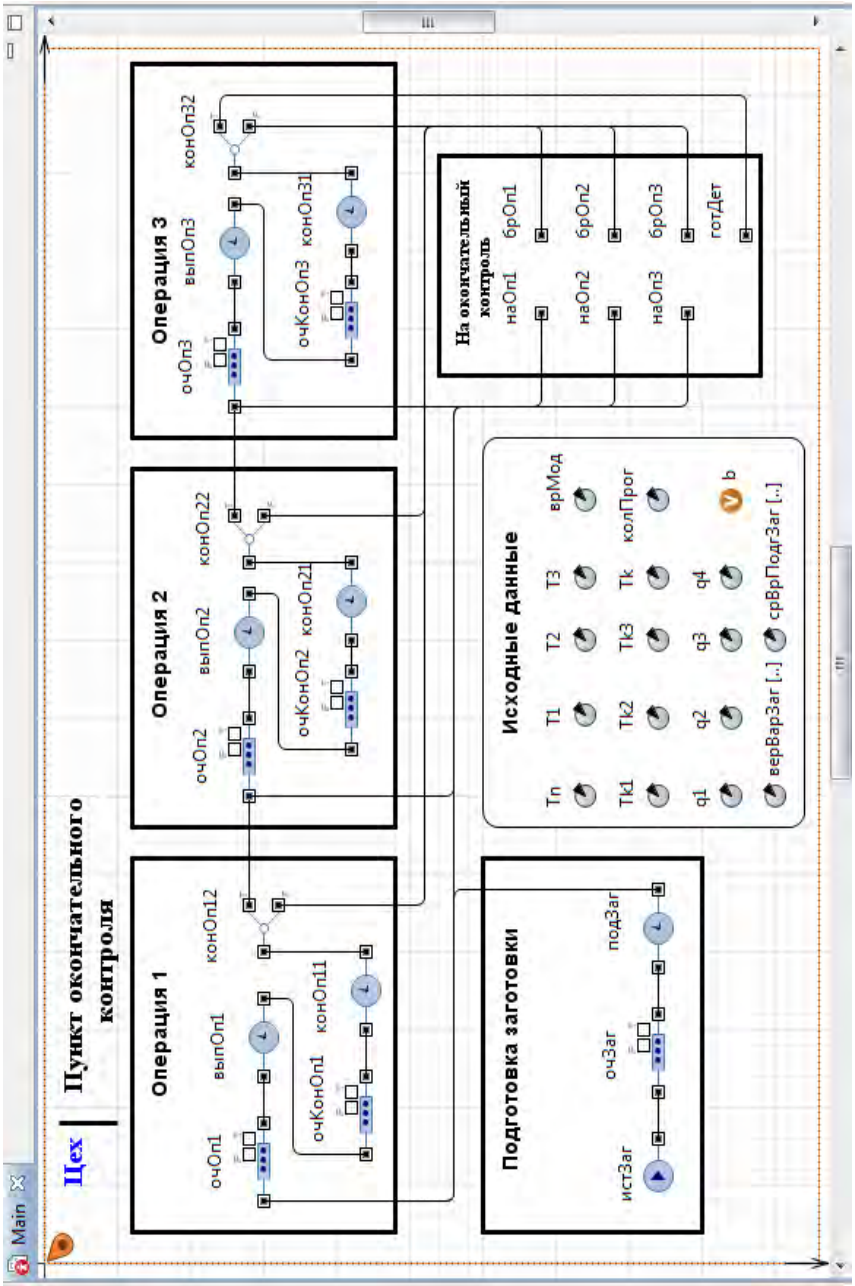


Рис. 2.12. Main с сегментами модели и экземпляром типа агента Kontrol

2.2.2.5. Создание области 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 7 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 две области просмотра. В первой области просмотра разместим объекты первых пяти сегментов (см. рис. 2.12), во второй — сегменты **Пункт окончательного контроля**, **Склад готовых деталей** и **Склад бракованных деталей**.

Создайте область просмотра на диаграмме типа агента Main для размещения объектов сегмента **Постановка на дежурство**.

1. В **Палитре** выделите **Презентация**. Перетащите элемент **Область просмотра**.
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 `zex`.
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. На странице **Местоположение и размер** установите: **X:** `0`, **Y:** `0`, **Ширина:** `780`, **Высота:** `530`.

2.2.2.6. 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. Щёлкните мышью в графическом редакторе, чтобы сделать его активным.
2. Щёлкните 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. Щёлкните 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 щёлкните 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:** `Цех`, как на рис. 2.12.
2. Перетащите второй элемент **text**, разместите и введите в поле **Текст:** `Пункт окончательного контроля`.
3. На панели **Свойства** раскройте **Специфические** и в поле **Действие по щелчку:** введите следующий Java код: `на_контроль.kontr.navigateTo();`

Во введённом коде `на_контроль` — имя экземпляра нового типа агента Kontrol, а `kontr` — имя области просмотра, которую мы создадим позднее на новом типе агента Kontrol.

На рис. 2.13 показан тип агента Kontrol с размещёнными на нём тремя сегментами модели. Создадим эти сегменты.

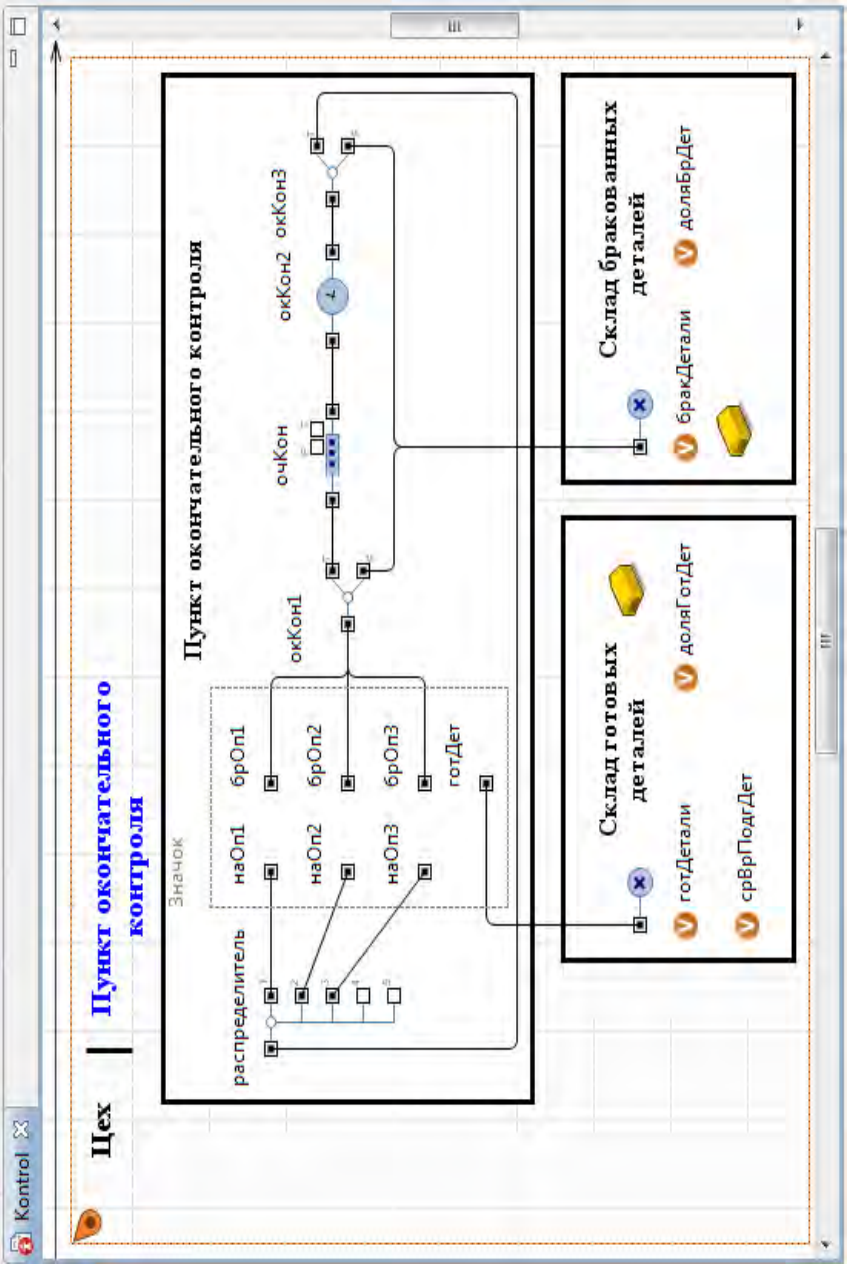


Рис. 2.13. Тип агента Kontrol с размещёнными на нём тремя сегментами

2.2.2.7. Пункт окончательного контроля

1. Из **Презентации** перетащите три элемента **Прямоугольник** и разместите так, как на рис. 2.10. На странице **Местоположение и размер** панели **Свойства** для верхнего прямоугольника введите: **X:** 110, **Y:** 70, **Ширина:** 580, **Высота:** 240. Для нижних прямоугольников: **X:** 190, **Y:** 330, **Ширина:** 250, **Высота:** 140; **X:** 460, **Y:** 330, **Ширина:** 250, **Высота:** 140.

2. Перетащите три элемента **text** и на странице **Основные** панели **Свойства** в поле **Текст:** каждого из них вместо имеющегося там слова text введите Пункт окончательного контроля, Склад готовых деталей, Склад бракованных деталей соответственно (рис. 2.13).

3. Из **Библиотеки моделирования процессов** перетащите два объекта **selectOutput**, объект **queue**, объект **delay** и один объект **selectOutput5**, разместите в верхнем прямоугольнике и соедините так, как показано на рис. 2.13. Порты брОп1, брОп2 и брОп3 соединяются с входом объекта selectOutput. Выход T (true) объекта окКон3 соединяется с входом объекта selectOutput.

4. Выделите поочередно объекты, начиная с левого объекта **selectOutput**, и на странице **Основные** панели **Свойства** установите свойства согласно рис. 2.13 и табл. 2.5. Во всех объектах должен быть установлен флажок **Отображать имя** и **На презентации**.

Таблица 2.5

Объект	Свойства	Значения
selectOutput	Имя Тип заявки Выход true выбирается Условие	окКон1 Detail При выполнении условия entity.a<2
queue	Имя Тип заявки Максимальная вместимость	очКон Detail Установите флажок
delay	Имя Тип заявки Тип Время задержки Вместимость	окКон2 Detail Определённое время exponential (1/main.Tk) 1

Окончание табл. 2.5

selectOutput	Имя Тип заявки Выход true выбирается Вероятность Действия При выходе(true)	окКон3 Detail Заданной вероятностью 1-main.q4 entity.a=2
selectOutput5	Имя Тип заявки Использовать: Условие 1 Условие 2 Условие 3	распределитель Detail Условия entity.n==1 entity.n==2 entity.n==3

2.2.2.8. Склад готовых деталей. Вывод результатов моделирования

1. Из библиотеки **Основная** перетащите на левый нижний прямоугольник три элемента **Переменная**. На странице **Основные** панели **Свойства** в поле **Имя:** каждого элемента введите соответствующие имена, показанные на рис. 2.13. Установите **Тип:** double.

2. Из **Библиотеки моделирования процессов** перетащите объект **sink**.

3. На странице **Основные** установите следующие свойства:

Имя: склГотДет

Отображать имя сбросьте флажок;

Тип заявки: Detail

Действия При входе:

```
готДетали = склГотДет.count()/main.колПрог;
доляГотДет = готДетали/(готДетали + бракДетали);
срВрПодгДет =
(main.врМод*main.колПрог)/склГотДет.count();
```

Код предназначен для расчёта результатов моделирования: абсолютного готДетали и относительного доляГотДет количества готовых деталей, среднего времени срВрПодгДет подготовки одной детали.

4. Из библиотеки **Картинки** перетащите картинку Склад и разместите как на рис. 2.13.

2.2.2.9. Склад бракованных деталей. Вывод результатов моделирования

1. Из библиотеки **Основная** перетащите на правый нижний прямоугольник два элемента **Переменная**. На странице **Основные** панели **Свойства** в поле **Имя:** каждого элемента введите соответствующие имена, показанные на рис. 2.13. **Тип:** `double`.
2. Из **Библиотеки моделирования процессов** перетащите объект **sink**.
3. На странице **Основные** панели **Свойства** установите следующие свойства:

Имя: `склБракДет`

Отображать имя сбросьте флажок;

Тип заявки: `Detail`

Действия При входе

```
бракДетали = склБракДет.count()/main.колПрог;
доляБрДет = бракДетали/(готДетали+бракДетали);
```

Код предназначен для расчёта результатов моделирования: абсолютного `бракДетали` и относительного `доляБрДет` количества бракованных деталей.

4. Из библиотеки **Картинки** перетащите картинку `Склад` и разместите как на рис. 2.13.

Так как все исходные данные размещены на диаграмме типа агента Main, то ссылка на них из диаграммы Kontrol, производится, например, так: `main.колПрог;`

2.2.2.10. Создание областей просмотра и переключение между ними

1. Из библиотеки **Презентация** перетащите элемент **Область просмотра**.
2. Перейдите на страницу **Основные** панели **Свойства**.
3. В поле **Имя:** введите `kontr`.
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`.
6. На странице **Местоположение и размер** панели **Свойства** введите: **X:** 30, **Y:** 10, **Ширина:** `670`, **Высота:** `480`.

7. Сбросьте флажок **Исключить**, если он был установлен.

8. Перетащите элемент **text**, разместите и введите в поле **Текст:** `Цех`, как на рис. 2.13.

9. На панели **Свойства** раскройте **Специфические** и в поле **Действие по щелчку:** введите следующий Java код: `main.zex.navigateTo();`

10. Перетащите второй элемент **text**, разместите и введите в поле **Текст:** `Пункт окончательного контроля`.

11. Теперь в ходе моделирования вы можете перемещаться между двумя областями просмотра. Чёрный цвет, например, `Цех`, означает, что открыта область просмотра, на которой размещен `Пункт окончательного контроля`. При этом цвет названия `Пункт окончательного контроля` другой.

2.2.3. Добавление элементов для проведения исследований

Для удобства считывания статистических данных о коэффициентах использования (загрузки) пункта подготовки заготовок и пунктов выполнения операций 1…3 дополним модель элементами **Переменная** и необходимыми java-кодами.

1. Из **Презентации** перетащите элемент **Скругленный прямоугольник** на диаграмму Kontrol и разместите как на рис. 2.14. Можно было бы всё это сделать на диаграмме Main, но тогда бы пришлось каждый раз после проведения эксперимента для считывания результатов переключаться между областями просмотра.

2. Перетащите элемент **text**, разместите и введите в поле **Текст:** `Коэффициенты использования`.

3. Из библиотеки **Основная** перетащите элемент **Переменная**. Разместите и дайте имя `коэфИспПодЗаг`, как на рис. 2.14. Оставьте тип double.

4. Перетащите ещё один элемент **Переменная** и дайте ему имя `коэфИспВыпОп1`. Оставьте тип double.

5. Скопируйте элемент с именем `коэфИспВыпОп1`. Вставьте ещё два таких элемента, которым системой будут присвоены имена `коэфИспВыпОп2` и `коэфИспВыпОп3` (см. рис. 2.14). Тип будет double.

6. Далее при вводе в свойства соответствующих объектов нужных кодов следует также на странице **Специфические** устанавливать флажки **Включить сбор статистики**.

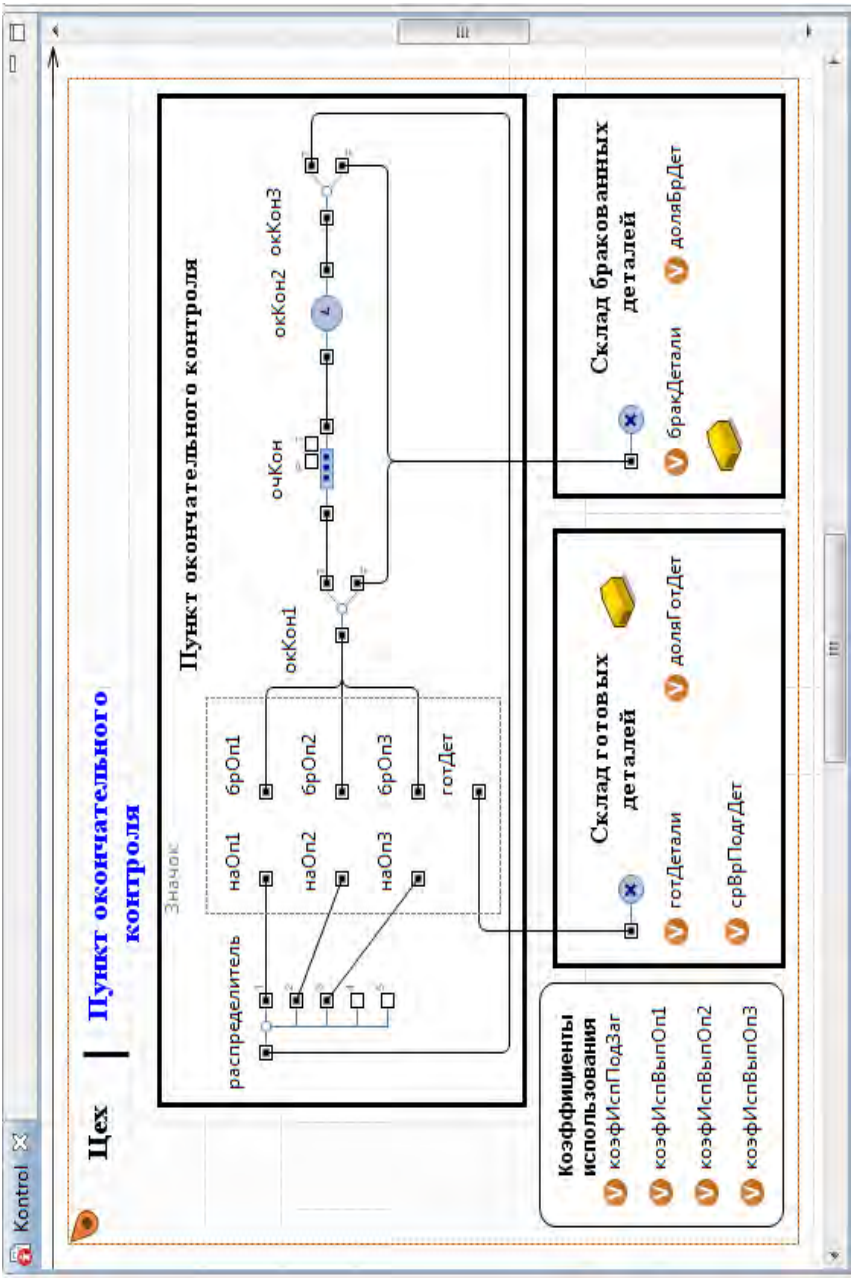


Рис. 2.14. Добавлены элементы для вывода коэффициентов использования

7. Выделите объект **подЗаг**. В поле **Действия При выходе** добавьте код:

```
на_контроль.коэфИспПодЗаг=
подЗаг.statsUtilization.mean();
```

8. Выделите объект **выпОп1**. В поле **Действия При выходе** введите код:

```
на_контроль.коэфИспВыпОп1=
выпОп1.statsUtilization.mean()
```

9. Выделите объект **выпОп2**. В поле **Действия При выходе** введите код:

```
на_контроль.коэфИспВыпОп2=
выпОп2.statsUtilization.mean()
```

10. Выделите объект **выпОп3**. В поле **Действия При выходе** введите код:

```
на_контроль.коэфИспВыпОп3=
выпОп3.statsUtilization.mean()
```

11. Время подготовки заготовки и время выполнения операций даны в мин. Возьмём 1 ед. мод. вр. = 1мин.

Рассчитаем количество прогонов, которые нужно выполнить в каждом наблюдении. При этом примем относительное количество готовых деталей как ожидаемую вероятность. Поскольку она заранее не известна, то расчёт проведём для худшего случая:

$$N = t_\alpha^2 \cdot \frac{\sigma^2}{\varepsilon^2} = 2,58^2 \cdot \frac{0,5(1-0,5)}{0,01^2} = 6,656\frac{0,25}{0,0001} \approx 16641.$$

12. Перейдите на агента верхнего уровня Main. В панели **Проекты** выделите Simulation:Main.

13. На странице **Модельное время** установите `Виртуальное время (максимальная скорость)`. В поле **Остановить:** из выпадающего списка выберите `В заданное время`.

14. В поле **Конечное время** введите 7987680 (480*16641 = 7987680).

15. В AnyLogic начальное число генератора случайных чисел устанавливается один раз перед запуском модели. На странице **Случайность** установите **Фиксированное начальное число (воспроизводимые прогоны)**. В поле **Начальное число** введите 23.

2.3. Интерпретация результатов моделирования

Мы провели эксперименты, увеличив в AnyLogic модельное время в 16641 раз (в GPSS World в каждом эксперименте выполнено 16641 прогонов).

Всего выполнено 8 экспериментов, результаты которых сведены в табл. 2.6. Первый эксперимент соответствует постановке задачи. Результаты первого эксперимента представлены на рис. 2.15 и в табл. 2.6.

В каждом следующем эксперименте параметры, установленные в предыдущем эксперименте, либо остаются неизменными, либо изменяются. Указываются только новые значения параметров в строке, предшествующей результатам следующего эксперимента. Например, во втором эксперименте уменьшено среднее время поступления заготовок с Tn = 35 до Tn = 30, а остальные параметры остались неизменными (табл. 2.6).

Всего изменялись значения шести параметров. Кроме того, в восьмом эксперименте были уменьшены средние времена подготовки вариантов заготовок (значения элементов одномерного массива).

Сравнительный анализ результатов экспериментов свидетельствует об адекватности GPSS World и AnyLogic7, так как их различия несущественны.

Например, относительная доля готовых изделий и относительная доля забракованных отличаются на 0 … 0,002, а среднее время изготовления одной детали — на 0,023 … 0,238.

Коэффиценты использования пункта подготовки заготовок и пунктов выполнения операций 1…3 в некоторых экспериментах или одинаковы, или различаются на 0,001 … 0,006.

По результатам экспериментов (табл. 2.6) можно сделать выводы об эффективности работы цеха по производству деталей и обнаружить «узкие» места.

Изменение параметров в каждом следующем эксперименте преследовало цель увеличения количества годных деталей и сокращения времени подготовки одной детали. Если в первом эксперименте готовых деталей было 9,916 (9,885), а среднее время изготовления одной детали 48,407 (48,559), то в последнем восьмом эксперименте цель была достигнута — 21,220 (21,161) и 22,620 (22,683) соответственно. При этом доля брака уменьшилась с 0,277 до 0,116.

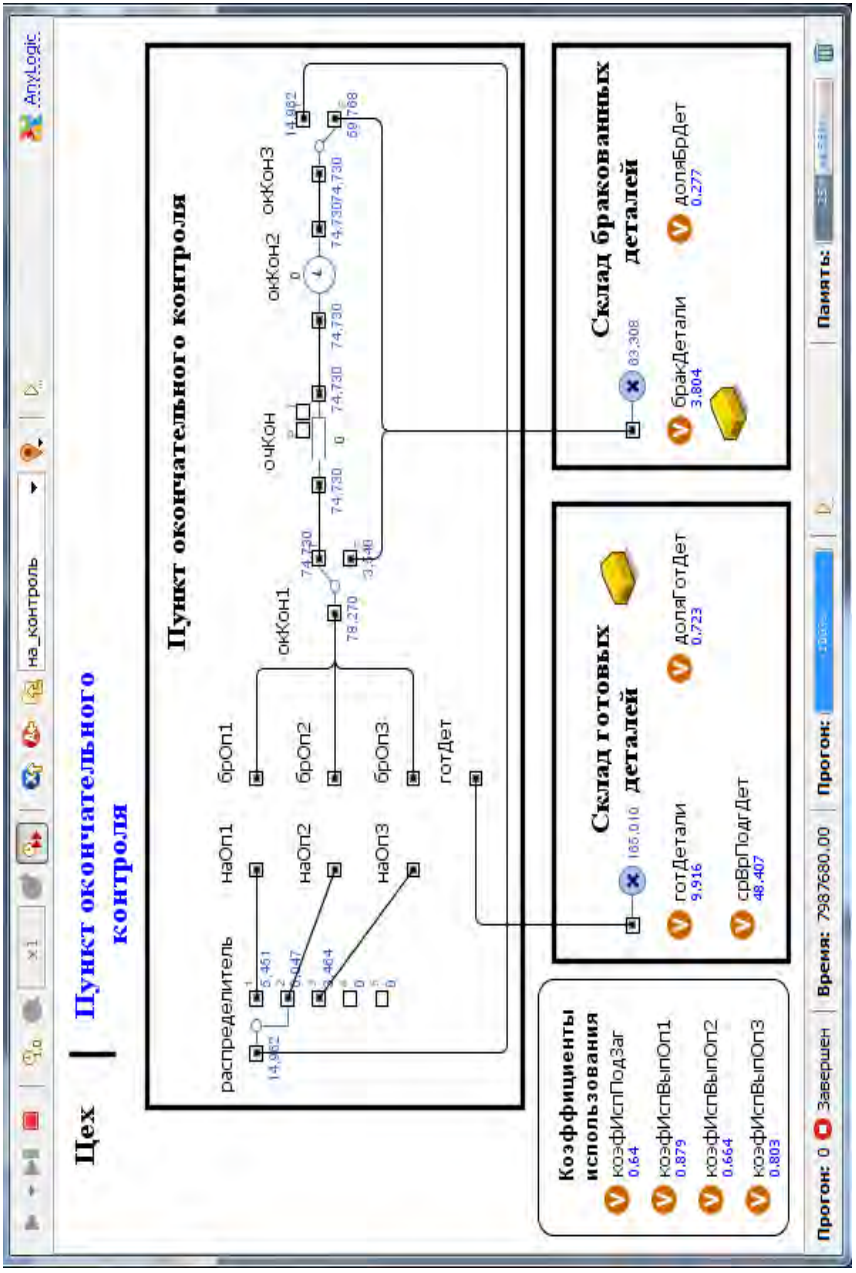


Рис. 2.15. Вариант результатов моделирования

Таблица 2.6

Показатели функционирования цеха

Показатели	GPSS World	AnyLogic6	AnyLogic7
1) согласно постановке задачи			
готДетали	9,885	9,882	9,916
доляГотДет	0,721	0,723	0,723
бракДетали	3,821	3,778	3,804
доляБрДет	0,279	0,277	0,277
срВрПодгДет	48,559	48,573	48,407
коэфИспПодЗаг	0,639	0,638	0,640
коэфИспВыпОп1	0,879	0,877	0,879
коэфИспВыпОп2	0,662	0,658	0,664
коэфИспВыпОп3	0,801	0,801	0,803
Δ_{11} доляГотДет	**0,002**		
Δ_{12} доляБрДет	**0,002**		
Δ_{13} срВрПодгДет	**0,152**		
2) Tn = 30			
готДетали	11,284	11,333	11,294
доляГотДет	0,722	0,723	0,724
бракДетали	4,338	4,343	4,312
доляБрДет	0,278	0,277	0,276
срВрПодгДет	42,538	42,353	42,500
коэфИспПодЗаг	0,749	0,746	0,746
коэфИспВыпОп1	1,000	1,000	1,000
коэфИспВыпОп2	0,752	0,762	0,755
коэфИспВыпОп3	0,915	0,916	0,916
Δ_{21} доляГотДет	**0,002**		
Δ_{22} доляБрДет	**0,002**		
Δ_{23} срВрПодгДет	**0,038**		
3) T1 = 25, T3 = 30			
готДетали	11,552	11,555	11,558
доляГотДет	0,722	0,724	0,723
бракДетали	4,459	4,414	4,437
доляБрДет	0,278	0,276	0,277
срВрПодгДет	41,553	41,542	41,530
коэфИспПодЗаг	0,748	0,746	0,748
коэфИспВыпОп1	0,852	0,852	0,852
коэфИспВыпОп2	0,774	0,770	0,770
коэфИспВыпОп3	0,802	0,801	0,802
Δ_{31} доляГотДет	**0,001**		
Δ_{32} доляБрДет	**0,001**		
Δ_{33} срВрПодгДет	**0,023**		

Продолжение табл. 2.6

Показатели функционирования цеха

Показатели	GPSS World	AnyLogic6	AnyLogic7
4) Tn = 25, T1 = 20, T2 = 15, T3 = 25			
готДетали	13,845	13,900	13,882
доляГотДет	0,723	0,724	0,724
бракДетали	5,314	5,305	5,303
доляБрДет	0,277	0,276	0,276
срВрПодгДет	34,669	34,532	34,577
коэфИспПодЗаг	0,895	0,899	0,897
коэфИспВыпОп1	0,817	0,822	0,818
коэфИспВыпОп2	0,555	0,558	0,557
коэфИспВыпОп3	0,801	0,803	0,801
Δ_{41} доляГотДет	**0,001**		
Δ_{42} доляБрДет	**0,001**		
Δ_{43} срВрПодгДет	**0,112**		
5) Tn = 20, T3 = 20			
готДетали	15,453	15,499	15,572
доляГотДет	0,723	0,724	0,724
бракДетали	5,932	5,915	5,925
доляБрДет	0,277	0,276	0,276
срВрПодгДет	31,063	30,969	30,825
коэфИспПодЗаг	1,000	1,000	1,000
коэфИспВыпОп1	0,911	0,913	0,918
коэфИспВыпОп2	0,619	0,621	0,623
коэфИспВыпОп3	0,716	0,718	0,720
Δ_{51} доляГотДет	**0,001**		
Δ_{52} доляБрДет	**0,001**		
Δ_{53} срВрПодгДет	**0,238**		
6) q3 = 0,05			
готДетали	16,200	16,206	16,219
доляГотДет	0,755	0,757	0,757
бракДетали	5,246	5,192	5,202
доляБрДет	0,245	0,243	0,243
срВрПодгДет	29,629	29,619	29,595
коэфИспПодЗаг	1,000	1,000	1,000
коэфИспВыпОп1	0,916	0,911	0,913
коэфИспВыпОп2	0,622	0,620	0,621
коэфИспВыпОп3	0,710	0,711	0,711
Δ_{61} доляБрДет	**0,002**		
Δ_{62} доляГотДет	**0,002**		
Δ_{63} срВрПодгДет	**0,034**		

Окончание табл. 2.6

Показатели функционирования цеха

7) q1= 0,05, q2 = 0,05			
Показатели	GPSS World	AnyLogic6	AnyLogic7
готДетали	18,903	18,856	18,991
доляГотДет	0,883	0,883	0,884
бракДетали	2,504	2,491	2,491
доляБрДет	0,117	0,117	0,116
срВрПодгДет	25,393	25,456	25,275
коэфИспПодЗаг	1,000	1,000	1,000
коэфИспВыпОп1	0,901	0,896	0,907
коэфИспВыпОп2	0,650	0,646	0,651
коэфИспВыпОп3	0,828	0,829	0,833
Δ_{71} доляГотДет	**0,001**		
Δ_{72} доляБрДет	**0,001**		
Δ_{73} срВрПодгДет	**0,018**		
8) T1=15, срВрПодгЗаг={7,11,18,19,23,22,0,0,0,0}			
готДетали	21,161	21,250	21,220
доляГотДет	0,883	0,884	0,884
бракДетали	2,801	2,785	2,795
доляБрДет	0,117	0,116	0,116
срВрПодгДет	22,683	22,588	22,620
коэфИспПодЗаг	0,939	0,940	0,938
коэфИспВыпОп1	0,756	0,760	0,759
коэфИспВыпОп2	0,725	0,729	0,728
коэфИспВыпОп3	0,929	0,932	0,931
Δ_{81} доляГотДет	**0,001**		
Δ_{82} доляБрДет	**0,001**		
Δ_{83} срВрПодгДет	**0,063**		

Но уже по второму эксперименту видно, что коэфИспВыпОп1=0,879, а коэфИспВыпОп3=0,803, то есть приближаются к 1. Поэтому изменение других параметров ничего не даст. Нужно уменьшить среднее время выполнения операций 1 и 3. Третий эксперимент это подтвердил.

В экспериментах 5…7 коэффициент загрузки пункта подготовки заготовок равен 1. Изменение доли брака лишь немного увеличил искомые показатели. Поэтому дальнейший рост годных деталей и среднего времени подготовки одной детали возможен был лишь при сокращении средних времён подготовки заготовок в зависимости от их типов.

ГЛАВА 3. МОДЕЛЬ ФУНКЦИОНИРОВАНИЯ НАПРАВЛЕНИЯ СВЯЗИ

3.1. Постановка задачи

Направление связи состоит из двух каналов (основного и резервного) и общего входного буфера емкостью на Emk сообщений.

На направление поступают два потока сообщений с экспоненциально распределенными интервалами времени, средние значения которых T1 = 3 мин и T2 = 4 мин. При нормальной работе сообщения передаются по основному каналу. Время передачи одного сообщения распределено по экспоненциальному закону со средним значением T3 = 2 мин.

В основном канале происходят сбои через интервалы времени, распределенные по экспоненциальному закону со средним значением T4 = 15 мин. Если сбой происходит во время передачи, то сообщение теряется. За время T5 = 5 с запускается резервный канал, который передает сообщения, начиная с очередного. Время передачи одного сообщения распределено по экспоненциальному закону со средним значением T6 = 3 мин.

Основной канал восстанавливается. Время восстановления канала подчинено экспоненциальному закону со средним значением T7 = 2 мин. После восстановления резервный канал выключается и основной канал продолжает работу с очередного сообщения.

Необходимо разработать имитационную модель и провести исследование функционирования направления связи в течение 2 ч.

Определить:

рациональную емкость накопителя;

загрузку основного и резервного каналов связи;

вероятности передачи сообщений потока 1 и потока 2;

вероятность передачи сообщений направлением связи в целом.

3.2. Уяснение задачи на разработку модели

Направление связи представляет собой систему массового обслуживания разомкнутого типа с ожиданием и с отказами из-за ограниченной ёмкости входного буфера. А также с выходами из строя (временного не функционирования) основного канала.

В модели сообщения следует представлять заявками, основной и резервный канал — одноканальными устройствами (ОКУ),

входной буфер (накопитель) — очередью. В очереди следует использовать дисциплину обслуживания FIFO.

Введём масштабирование: 1 единица модельного времени соответствует 1 с, то есть, например, время моделирования равно 2 часам, тогда 2*60*60 = 7200 единиц модельного времени. Аналогично T1 = 120, T2 =240 и т.д.

Декомпозиция системы и состав сегментов модели определяются разработчиком. Введём в модели функционирования направления связи следующие сегменты:

исходные данные;

источники сообщений;

буфер, основной и резервный каналы связи;

имитатор отказов основного канала;

результаты моделирования.

3.3. Модель направления связи в AnyLogic

3.3.1. Исходные данные

Для ввода исходных данных используем элементы **Параметр**.

1. Выполните команду **Файл/Создать/Модель** на панели инструментов.
2. В поле **Имя модели** диалогового окна **Новая модель** введите `Направление связи`. Выберите каталог, в котором будут сохранены файлы модели. Щёлкните кнопку **Готово**.
3. Полагаем вначале, что все сегменты модели мы сможем разместить так, что они будут видны в ходе работы модели. В **Палитре** выделите **Презентация**.
4. Перетащите элемент **Скругленный прямоугольник** для размещения элементов исходных данных.
5. На странице **Местоположение и размер** панели **Свойства**: введите: **X:** 630, **Y:** 20, **Ширина:** 320, **Высота:** 280.
6. Перетащите элемент **text** и на странице **Текст** панели **Свойства** вместо text введите `Исходные данные`.
7. В **Палитре** выделите **Основная**. Перетащите элементы **Параметр** на элемент с именем **Исходные данные**. Разместите их и дайте имена так, как показано на рис. 3.1. Значения свойств установите согласно табл. 3.1.

Замечание. В данной модели (а это возможно и в любых других моделях) все идентификаторы на русском языке.

Рис. 3.1. Размещение элементов для ввода исходных данных

Таблица 3.1

Параметр		
Имя	Тип	Значение по умолчанию
интер_сообщ_потока1 интер_сообщ_потока2	double double	180 240
ёмкость_буфера	int	5
время_передачи_осн_кан время_передачи_рез_кан	double double	120 180
время_вкл_рез_кан	double	10
время_нараб_отказ_осн_кан время_восстан_осн_кан	double double	900 120

3.3.2. Вывод результатов моделирования

Для вывода результатов моделирования используем элемент **Переменная**.

1. В **Палитре** выделите **Презентация**. Перетащите элемент **Скругленный прямоугольник** для размещения элементов **Переменная**.

2. На странице **Местоположение и размер** панели **Свойства**: введите: **X:** 470, **Y:** 330, **Ширина:** 490, **Высота:** 320.

3. Перетащите элемент **text** и на странице **Текст** панели **Свойства** вместо text введите Результаты моделирования.

4. В **Палитре** выделите **Основная**. Перетащите элементы **Переменная**. Разместите их и дайте им имена так, как показано на рис. 3.2. Тип всех переменных double, кроме переменной — текущая ёмкость буфера. Её тип — int.

3.3.3. Построение событийной части модели

В событийную часть модели, к построению которой мы приступаем, включим указанные ранее три сегмента (кроме исходных данных и результатов моделирования).

1. В **Палитре** выделите **Презентация**. Перетащите три элемента **Прямоугольник** и разместите так, как на рис. 3.3.

2. На странице **Местоположение и размер** панели **Свойства**: для размещения объектов имитации источников сообщений введите: **X:** 20, **Y:** 20, **Ширина:** 150, **Высота:** 190.

Рис. 3.2. Размещение элементов для вывода результатов моделирования

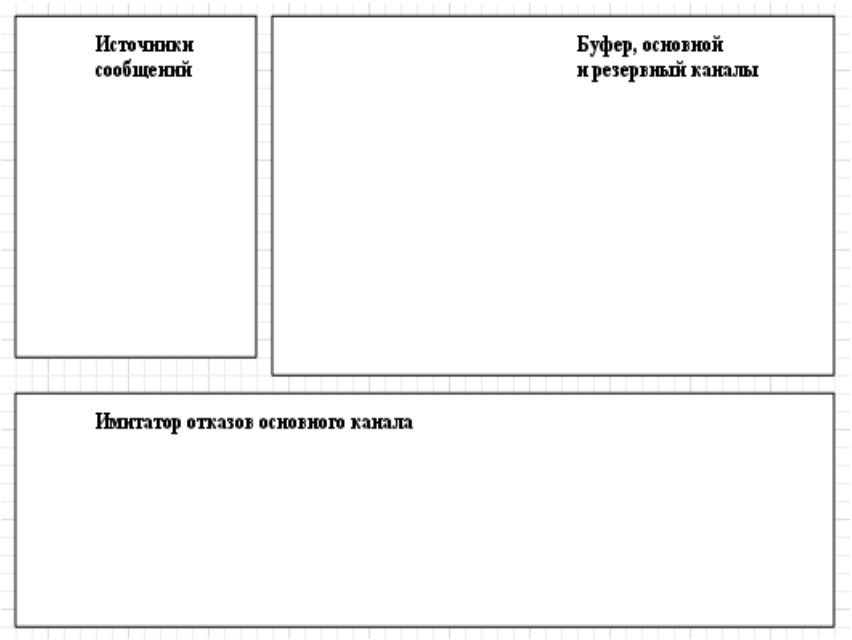


Рис. 3.3. Элементы для размещения сегментов событийной части

3. Для размещения объектов имитации буфера, основного и резервного каналов введите:

X: 190, **Y:** 20, **Ширина:** 370, **Высота:** 200.

4. Для размещения объектов имитации отказов основного канала введите:

X: 20, **Y:** 350, **Ширина:** 380, **Высота:** 130.

5. Перетащите также три элемента **text** и на странице **Текст** панели **Свойства** вместо text каждого элемента введите названия, показанные на рис. 3.3.

3.3.3.1. Источники сообщений

Данный сегмент предназначен для имитации поступления сообщений, счета суммарного количества поступающих сообщений на направление связи и по потокам 1 и 2.

1. В **Палитре** выделите **Библиотека моделирования процессов**.

2. Перетащите два объекта **source** на диаграмму типа агента Main и разместите в прямоугольнике с именем Источники сообщений.

3. Для записи и хранения параметров сообщений в дополнительные поля заявок нужно создать новый тип заявки. Создайте тип заявки `Message`.

4. В панели **Проект** щёлкните правой кнопкой мыши элемент модели верхнего уровня дерева и выберите **Создать Java класс**.

5. Появится диалоговое окно **Новый Java класс**. В поле **Имя:** введите имя нового класса `Message`.

6. В поле **Базовый класс:** выберите из выпадающего списка `Entity` в качестве базового класса. Щёлкните кнопку **Далее**.

7. Появится вторая страница **Мастера создания Java класса**. Добавьте следующее поле Java класса, которое потребуется в дальнейшем для разделения переданного направлением потока сообщений на поток 1 и поток 2:

```
int  numPotok
```

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

9. Щёлкните кнопку **Готово**. Появится редактор кода и автоматически созданный код вашего Java класса. Закройте код.

10. Теперь нужно преобразовать Java класс в тип агента. Для этого щёлкните правой кнопкой мыши в панели **Проект** только что созданный Java класс и в контекстном меню выберите **Преобразовать Java класс в тип агента**.

11. Появится окно с автоматически созданными параметрами нового типа заявок `Detail`.

12. Выделите последовательно первый и второй объекты **source**. На странице **Основные** панели **Свойства** установите их свойства согласно табл. 3.3.

3.3.3.2. Буфер, основной и резервный каналы

Сегмент предназначен для приёма поступающих сообщений, имитации передачи их, счета переданных и потерянных сообщений, расчета вероятности передачи сообщений.

1. В **Палитре** выделите **Библиотеку моделирования процессов** .

2. Перетащите на диаграмму Main и разместите в прямоугольнике с именем `Буфер, основной и резервный каналы` объекты, показанные на рис. 3.4. Соедините их между собой, а также с объектами сегмента **Источники сообщений**.

Таблица 3.3

Свойства объектов source

Имя	Свойства	Значения

`Поток_1`	Отображать имя
	Тип заявки
	Прибывают согласно
	Время между прибытиями
	Новая заявка
	Действия
	При выходе:
		Установите флажок
		`Message`
		
		Времени между прибытиями
		`exponential(1/интер_сообщ_потока1)`
		`Message`
		
		`entity.numPotok = 1;`
		`поступило_сообщ_потока1 ++;`
		`всего_сообщ_поступило ++;`

`Поток_2`	Отображать имя
	Тип заявки
	Прибывают согласно
	Время между прибытиями
	Новая заявка
	Действия
	При выходе:
		Установите флажок
		`Message`
		
		Времени между прибытиями
		`exponential(1/интер_сообщ_потока2)`
		`Message`
		
		`entity.numPotok = 2;`
		`поступило_сообщ_потока2 ++;`
		`всего_сообщ_поступило ++;`

3. Объект **source** вам известен. Объект `hold` из **Библиотеки моделирования процессов** нет. Он блокирует/разблокировывает поток заявок на определенном участке блок-схемы. Если объект находится в заблокированном состоянии, то заявки не будут поступать на его входной порт, и будут ждать, пока объект не будет разблокирован.

4. Состоянием объекта можно управлять программно с помощью метода `setBlocked()`. Метод блокирует входной порт, если в качестве значения аргумента передано `true`, и разблокировывает его при передаче аргумента `false`.

5. Метод `isBlocked()` возвращает `true`, если входной порт заблокирован. Если порт не заблокирован — возвращает `false`. Мы в дальнейшем воспользуемся этими методами.

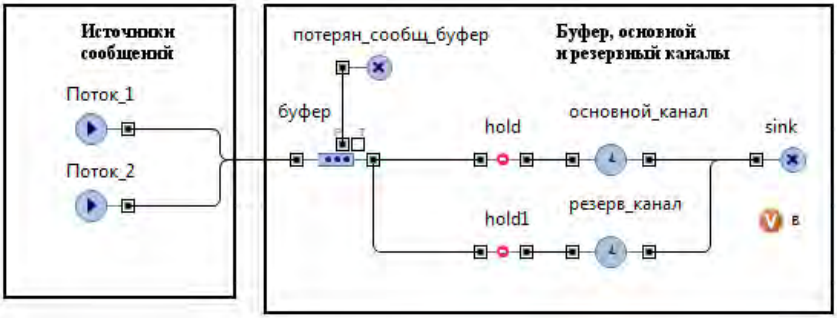


Рис. 3.4. Объекты двух сегментов модели

6. Последовательно выделите объекты и установите им свойства согласно табл. 3.4.

Таблица 3.4

Свойства	Значение
Имя Отображать имя Тип заявки Вместимость Действия При входе: Действия При выходе: Разрешить вытеснение	Буфер Установить флажок Message ёмкость_буфера текущая_ёмкость_буфера=буфер.size(); текущая_ёмкость_буфера=буфер.size(); Установить флажок
Имя Отображать имя Тип заявки Изначально заблокирован	hold1 Установить флажок Message Установить флажок
Имя Тип заявки	hold Message
Имя Отображать имя Тип заявки Действия При входе:	потер_сообщ_буфер Установить флажок Message **if** (entity.numPotok == 1) потеряно_сообщ_потока1 ++; **if** (entity.numPotok == 2) потеряно_сообщ_потока2 ++; всего_потеряно_сообщ ++;
Имя Отображать имя Тип заявки Тип	основной_канал Установить флажок Message Определённое время

Окончание табл. 3.4

Время задержки Вместимость Включить сбор статистики	exponential(1/время_передачи_осн_кан) 1 Установить флажок
Имя Отображать имя Тип заявки Вместимость Тип Время задержки Действия При входе: Действия При выходе: Включить сбор статистики	резерв_канал Установить флажок Message 1 Определённое время exponential(1/время_передачи_рез_кан) **if** (a==0) в = время_передачи_рез_кан; **if** (a==1) {в = время_передачи_рез_кан + время_вкл_рез_кан; a=0;} **if** (hold.isBlocked()== **true**) hold1.setBlocked(**false**); Установить флажок
Имя Отображать имя Тип заявки Действие при входе	sink Установить флажок Message **if** (entity.numPotok == 1) {передано_сообщ_потока1 ++; вероят- ность_передачи_сообщ_потока1 = передано_сообщ_потока1 /поступило_сообщ_потока1;} **if** (entity.numPotok == 2) {передано_сообщ_потока2 ++; вероят- ность_передачи_сообщ_потока2 = передано_сообщ_потока2 /поступило_сообщ_потока2;} всего_передано_сообщ ++; вероятность_передачи_сообщ = всего_передано_сообщ /всего_сообщ_поступило; вероятность_потери_сообщ = 1- вероятность_передачи_сообщ;

3.3.3.3. Имитатор отказов основного канала связи

Данный сегмент предназначен для розыгрыша интервала времени до очередного отказа, блокирования основного канала, разблокирования резервного канала, имитации восстановления основного канала, его разблокирования и блокирования резервного канала.

Сегмент построен из объектов и элементов, показанных на рис. 3.5. Идея его работы заключается в следующем. Генератор вырабатывает одну заявку, и становится неактивным. Заявка поступает на объект задержки, разыгрывающий время до очередного отказа. После этого заявка поступает на второй объект задержки, имитирующий время восстановления основного канала.

С выхода второго объекта задержки заявка поступает опять на вход первого объекта задержки. Процесс имитации отказов повторяется в цикле.

Аналогичным образом построен сегмент имитации отказов основного канала и в GPSS-модели (см. п. 3.2).

Если построить сегмент так, что время до очередного отказа будет разыгрывать генератор, то это не логично, так как при таком варианте отсчет времени до очередного отказа не будет начинаться от момента окончания восстановления канала. Возникнут ситуации, когда очередной отказ придется на время, когда идет процесс восстановления канала.

Постройте сегмент имитации отказов основного канала связи.

1. Перетащите из **Библиотеки моделирования процессов** source и два объекта delay, соедините их как на рис. 3.5.
2. Последовательно выделите и установите свойства объектов согласно табл. 3.5.

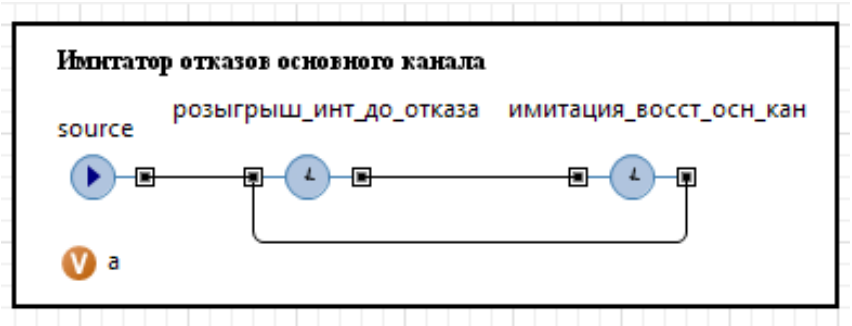


Рис. 3.5. Сегмент имитации отказов основного канала связи

Таблица 3.5

Свойства	Значение
Имя Отображать имя Тип заявки Прибывают согласно Интенсивность прибытия Ограниченное количество прибытий Количество заявок, прибывающих за один раз	source Установить флажок Agent Интенсивности 1 Установить флажок 1

Имя
Отображать имя
Тип заявки
Тип
Время задержки
 	
Вместимость
Действия
При выходе:
 	
Включить сбор статистики
	розыгрыш_инт_до_отказа
	Установить флажок
	Agent
	Определённое время
	exponential (1/время_нараб_отказ_осн_кан)
	1
	hold.setBlocked(**true**);
	if (основной_канал.size()!=0) {основ-
	ной_канал.remove((Message) основной_канал.get(0));
	всего_потеряно_сообщ ++;}
	hold1.setBlocked(**false**);
	a=1;
	Установить флажок

Имя
Тип заявки
Тип
Время задержки
 	
Вместимость
Действия При выходе:
 	
Включить сбор статистики
	имитация_восст_осн_кан
	Agent
	Определённое время
	exponential (1/время_восстан_осн_кан)
	1
	hold.setBlocked(**false**);
	hold1.setBlocked(**true**);
	Установить флажок

Обратим внимание на переменные a и в, предназначенные для организации включения резервного канала таким образом, чтобы время на включение учитывалось только при поступлении первого сообщения на резервный канал. При последующих поступлениях это время не учитывается. И это каждый раз повторяется при выходе из строя основного канала, так как после восстановления

основного канала резервный канал выключается. Резервный канал выключается, но передача сообщения по нему, если это было в момент включения в работу основного канала, продолжается. Таким образом, какое-то время каналы работают параллельно. Потерь сообщений при выключении резервного канала нет.

Также организовано и в GPSS-модели, но для этого использована сохраняемая ячейка Kont (см. п. 3.2).

В модели AnyLogic после занятия сообщением резервного канала элемент hold1 блокируется. Может быть так, что в процессе передачи сообщения резервным каналом возобновит работу основной канал, то есть элемент hold будет разблокирован, и сообщения пойдут на основной канал. Чтобы такая ситуация учитывалась и в GPSS-модели, в нее добавлена команда

```
UNLINK Nak,Prov3,1
```

выводящая из буфера очередное сообщение на основной канал, не дожидаясь окончания передачи сообщения резервным каналом.

3.4. Отладка модели

Запустите только что созданную модель **Направление_связи**.

После запуска, если вы все рекомендации выполнили корректно, сразу появится сообщение об ошибке (рис. 3.6). Согласно ему к выходному порту корневого объекта `буфер.out` подключены два входных порта корневых объектов `hold1.in` и `hold.in`.

Разрешается соединять несколько выходных портов с одним входным и наоборот. Что мы и сделали (см. рис. 3.4).

Если несколько выходных портов соединены с одним входным портом, и сразу несколько объектов хотят передать заявку, выбор будет произведен «справедливым» образом, согласно циклическому (round-robin) алгоритму, реализованному во входном порте.

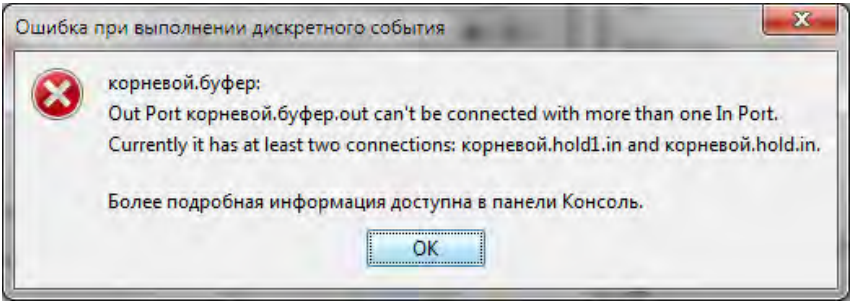


Рис. 3.6. Сообщение об ошибке подключения портов

Если один выходной порт соединен с несколькими входными портами, и сразу несколько портов готовы принять заявку, то тот порт, куда она будет переслана, будет выбираться согласно определенному порядку. Этот выбор будет зависеть от того, как исполняющий модуль AnyLogic обрабатывает одновременные события. Таким образом, полной гарантии того, что этот выбор будет справедливым, нет. Рекомендуется не использовать такие соединения, а лучше пользоваться объектом SelectOutput или механизмом сложной маршрутизации, чтобы иметь полный контроль над распределением заявок.

Механизмом сложной маршрутизации мы воспользуемся в модели функционирования сети связи, а сейчас применим объект SelectOutput.

1. Удалите соединения выходного порта объекта **буфер** с входными портами объектов hold1 и hold.
2. Перетащите объект selectOutput и соедините его входной порт с выходным портом объекта **буфер**, а выходные порты — с входными портами объектов hold1 и hold (рис. 3.7).
3. Полагаем, что выходной порт true объекта selectOutput выбирается по условию, когда объект hold не заблокирован, то есть hold.isBlocked().
4. Замените **Тип заявки:** Agent на тип заявки Message.
5. Установите **Выход true выбирается:** `При выполнении условия`.
6. В поле **Условие:** введите `hold.isBlocked()`.
7. Запустите модель.

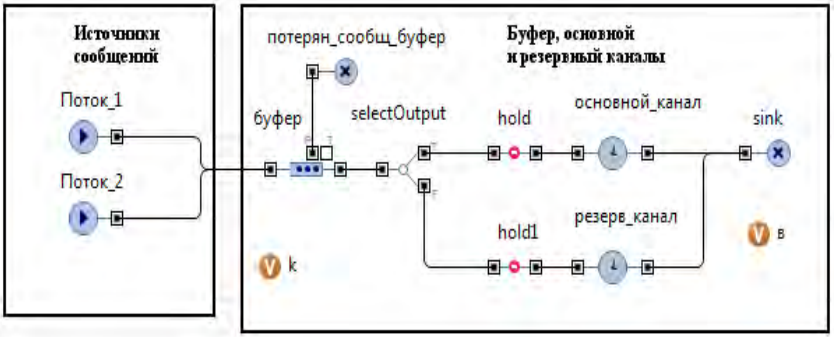


Рис. 3.7. Для соединения портов использован объект `selectOutput`

8. В результате моделирования получили вероятность_передачи_сообщ = 0,568, коэф_использ_кан = 0,690, в том числе коэф_использ_осн_кан = 0,614.

9. Корректен ли полученный результат? Нужно проверить. Как это сделать? Применим способ, суть которого заключается в следующем: два канала, один из которых резервный, должны иметь большую вероятность пропускной способности, чем один основной канал, также выходящий из строя, как и при резервировании. Если это не так, то модель построена некорректно.

10. Удалите объекты selectOutput, hold1, **резерв_канал** и соединения между ними.

11. Соедините выходной порт объекта **буфер** с входным портом объекта hold.

12. Запустите модель. Появится сообщение об ошибке. Щёлкните **Отменить**. Увидите две ошибки:

`Невоможно разрешить hold1`

`Невоможно разрешить hold1`

13. Дважды щёлкните по первой ошибке. В открывшемся окне удалите код с hold1.

14. Дважды щёлкните по второй ошибке. В открывшемся окне также удалите код с hold1.

15. Запустите модель. Получите результат моделирования: вероятность_передачи_сообщ = 0,691, то есть больше, чем с резервированием. При этом коэф_использ_осн_кан = 0,805, то есть также больше. Делаем вывод, что модель работает неверно.

16. Продолжим корректировку модели. Возвратимся к исходному варианту построения модели (см. рис. 3.7). Для этого используем команду **Правка**. Последовательно щёлкая мышью, возвращаемся к построению модели с резервным каналом.

17. Попробуем обойтись без объектов hold. Удалите объекты hold и hold1 и их соединения с выходами объекта selectOutput.

18. Соедините выходы объекта selectOutput со входами объектов **основ_канал** и **резерв_канал** (рис. 3.8).

19. Из палитры **Основная** перетащите элемент **Переменная**. Дайте имя `основной_канал_работает`. **Тип:** `boolean`. **Начальное значение:** `true`.

20. Выделите selectOutput. Замените **Тип заявки:** Agent на тип заявки Message. Установите **Выход true выбирается:** `При выполнении условия`.

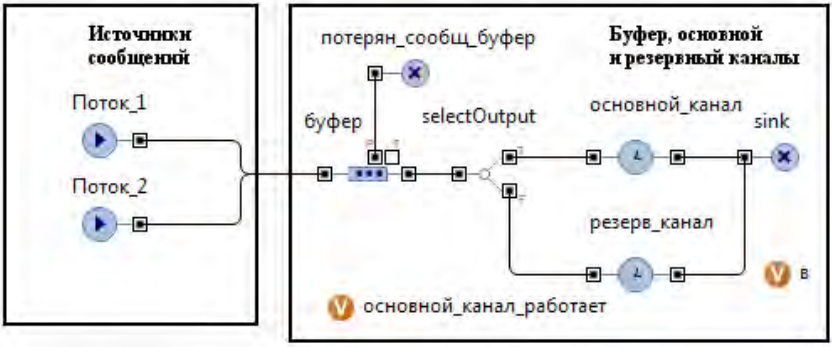


Рис. 3.8. Вариант модели без объектов hold

21. В поле **Условие:** введите `основной_канал_работает`.

22. Выделите объект **розыгрыш_инт_до_отказа**. В поле **Действия При выходе:** замените имеющийся там код следующим:

```
основной_канал_работает = false;
if (основной_канал.size()!=0){
  Message m = основной_канал.get(0);
  основной_канал.stopDelayForAll();
  всего_потеряно_сообщ ++;

  всего_передано_сообщ--;

  if (m.numPotok == 1){
     передано_сообщ_потока1--;
  }
  if (m.numPotok == 2){
     передано_сообщ_потока2--;
  }
}
a=1;
```

Код после первого `if` выполняется тогда, когда в объекте **основной_канал** находится сообщение. В коде функция `stopDelayForAll()` останавливает задержку для всех заявок, которые в этот момент задерживаются объектом **основной_канал**. В нашей модели это может быть только одно сообщение. Поэтому увеличивается количество потерянных сообщений на одно сообщение. Также уменьшается на одно количество переданных сообщений в целом направлением связи и по потокам.

23. Выделите объект **имитация_восст_осн_кан**. В поле **Действия При выходе:** замените код следующим кодом:

```
основной_канал_работает = true;
коэф_безотк_раб_осн_кан=
1-имитация_восст_осн_кан.statsUtilization.mean();
```

24. Выделите объект **резерв_канал**. В поле **Действия При выходе:** оставьте следующий код:

```
коэф_использ_резерв_кан=
резерв_канал.statsUtilization.mean()
```

25. Запустите модель. Результаты моделирования приведены на рис. 3.9. Видим, что вероятность передачи сообщений 0,773 > 0,568. При этом суммарный коэффициент использования обеих каналов равен 1,003, что свидетельствует о параллельной работе каналов в некоторые моменты времени. Забегая вперёд заметим, что близкие к этим результаты получены и в GPSS World. И точно такие же в AnyLogic 6 при построении модели с использованием объектов hold (см. рис. 3.4). Теперь можно перейти к проведению экспериментов и интерпретации полученных результатов.

Рис. 3.9. Результаты моделирования

3.5. Интерпретация результатов моделирования

Проведите моделирование и сравните полученные результаты. Результаты наших экспериментов приведены в табл. 3.6.

Всего выполнено 8 экспериментов. Здесь, напомним, как и в главе 2, первый эксперимент соответствует постановке задачи. В каждом следующем эксперименте параметры, установленные в предыдущем эксперименте, либо остаются неизменными, либо изменяются. Указываются только новые значения параметров в строке, предшествующей результатам следующего эксперимента. Например, во втором эксперименте увеличена ёмкость входного буфера с 5 до 10 сообщений, а остальные параметры остались неизменными (табл. 3.6).

Для получения результатов моделирования с точностью $\varepsilon = 0,01$ и доверительной вероятностью $\alpha = 0,95$ в GPSS World необходимо выполнить 9604 прогонов модели. В каждом эксперименте выполнялось 10000 прогонов.

Время моделирования в AnyLogic было увеличено в 10 000 раз и составляло 72 000 000 единиц модельного времени. Следует заметить, что если в GPSS World выполнить с этим же модельным временем один прогон, то результаты получаются такими же, что и при 10 000 прогонов модели.

Согласно данным табл. 3.6 во втором, третьем и шестом экспериментах экспериментах вероятность передачи сообщений отличается на 0,002 … 0,004. В остальных экспериментах вероятности передачи сообщений, полученные в GPSS World и AnyLogic7, отличаются на 0,017 … 0,029, то есть на порядок больше.

По результатам экспериментов можно сделать вывод о чувствительности модели к изменению параметров направления связи. Например, при увеличении ёмкости входного буфера с 5 до 10 сообщений вероятность передачи возрастает с 0,773 (0,752) до 0,831 (0,829).Уменьшение интервалов (увеличение интенсивности) поступления сообщений потоков 1 и 2 в два раза (90 и 120) снижает вероятности передачи сообщений с 0,831 (0,829) до 0,456 (0,438). В тоже время повышение скорости передачи основного канала в два раза (60) и увеличение не менее чем в 5 раз времени наработки на отказ основного канала приводит к возрастанию вероятностей передачи сообщений с 0,456 (0,438) до 0,815 (0,844).

Машинное время выполнения модели в обеих системах составляет 5…7 сек (в AnyLogic7 в виртуальном режиме).

Таблица 3.6

Показатели функционирования направления связи

Показатели	GPSS World	AnyLogic6	AnyLogic7
1) объем_буфера = 5			
вероятность_передачи_сообщ вероятность_передачи_сообщ_потока1 вероятность_передачи_сообщ_потока2	0,752 0,752 0,753	0,773 0,772 0,773	0,773 0,771 0,774
Δ_1 вероятности_передачи_сообщ	**Δ_1 = 0,021**		
вероятность_потери_сообщ	0,248	0,227	0,227
коэф_использ_осн_кан коэф_использ_рез_кан	0,777 0,152	0,757 0,222	0,718 0,286
сум_коэф_использ_кан	0,929	0,979	1,003
2) объем_буфера = 10			
вероятность_передачи_сообщ вероятность_передачи_сообщ_потока1 вероятность_передачи_сообщ_потока2	0,829 0,829 0,829	0,861 0,861 0,862	0,831 0,832 0,831
Δ_2 вероятности_передачи_сообщ	**Δ_2 =0,002**		
вероятность_потери_сообщ	0,171	0,139	0,169
коэф_использ_осн_кан коэф_использ_рез_кан	0,861 0,157	0,841 0,250	0,756 0,327
сум_коэф_использ_кан	1,018	1,091	1,084
3) интер_сообщ_потока1 = 90, интер_сообщ_потока2 = 120			
вероятность_передачи_сообщ вероятность_передачи_сообщ_потока1 вероятность_передачи_сообщ_потока2	0,438 0,438 0,438	0,454 0,454 0,453	0,456 0,456 0,456
Δ_3 вероятности_передачи_сообщ	**Δ_3 = 0,002**		
вероятность_потери_сообщ	0,562	0,546	0,544
коэф_использ_осн_кан коэф_использ_рез_кан	0,882 0,209	0,882 0,266	0,808 0,393
сум_коэф_использ_кан	1,091	1,148	1,201
4) время_передачи_осн_кан = 60, время_нараб_отказ_осн_кан = 5000			
вероятность_передачи_сообщ вероятность_передачи_сообщ_потока1 вероятность_передачи_сообщ_потока2	0,844 0,844 0,845	0,848 0,848 0,848	0,815 0,815 0,815
Δ_4 вероятности_передачи_сообщ	**Δ_4 = 0,029**		
вероятность_потери_сообщ	0,156	0,152	0,185
коэф_использ_осн_кан коэф_использ_рез_кан	0,971 0,043	0,970 0,058	0,922 0,088
сум_коэф_использ_кан	1,014	1,028	1,010

Продолжение табл. 3.6

Показатели функционирования направления связи

Показатели	GPSS World	AnyLogic6	AnyLogic7
5) время_передачи_рез_кан = 90, время_восстан_осн_кан = 60			
вероятность_передачи_сообщ вероятность_передачи_сообщ_потока1 вероятность_передачи_сообщ_потока2	0,851 0,851 0,851	0,857 0,857 0,857	0,834 0,833 0,835
Δ_5 вероятности_передачи_сообщ	**Δ_5 = 0,017**		
вероятность_потери_сообщ	0,149	0,143	0,166
коэф_использ_осн_кан коэф_использ_рез_кан	0,982 0,017	0,980 0,029	0,945 0,043
сум_коэф_использ_кан	0,999	1,009	0,988
6) интер_сообщ_потока1 = 45, интер_сообщ_потока2 = 60			
вероятность_передачи_сообщ вероятность_передачи_сообщ_потока1 вероятность_передачи_сообщ_потока2	0,430 0,431 0,429	0,432 0,432 0,431	0,434 0,434 0,434
Δ_6 вероятности_передачи_сообщ	**Δ_6 = 0,004**		
вероятность_потери_сообщ	0,570	0,568	0,566
коэф_использ_осн_кан коэф_использ_рез_кан	0,988 0,024	0,988 0,029	0,980 0,048
сум_коэф_использ_кан	1,012	1,017	1,028
7) время_передачи_осн_кан = 30, время_передачи_рез_кан = 45			
вероятность_передачи_сообщ вероятность_передачи_сообщ_потока1 вероятность_передачи_сообщ_потока2	0,850 0,851 0,850	0,853 0,853 0,853	0,832 0,831 0,832
Δ_7 вероятности_передачи_сообщ	**Δ_7 = 0,018**		
вероятность_потери_сообщ	0,150	0,147	0,168
коэф_использ_осн_кан коэф_использ_рез_кан	0,982 0,015	0,982 0,021	0,952 0,028
сум_коэф_использ_кан	0,997	1,003	0,980
8) время_вкл_рез_кан = 1, время_восстан_осн_кан = 30			
вероятность_передачи_сообщ вероятность_передачи_сообщ_потока1 вероятность_передачи_сообщ_потока2	0,851 0,851 0,851	0,855 0,855 0,855	0,833 0,833 0,832
Δ_8 вероятности_передачи_сообщ	**Δ_8 = 0,018**		
вероятность_потери_сообщ	0,149	0,145	0,167
коэф_использ_осн_кан коэф_использ_рез_кан	0,988 0,008	0,987 0,015	0,957 0,021
сум_коэф_использ_кан	0,996	1,002	0,978

ГЛАВА 4. МОДЕЛЬ ФУНКЦИОНИРОВАНИЯ СЕТИ СВЯЗИ

4.1. Модель в AnyLogic

4.1.1. Постановка задачи

В сети связи имеются n_1 абонентов, обменивающихся между собой сообщениями. Адресация сообщений организована посредством маршрутизаторов. На маршрутизатор поступают сообщения через случайные промежутки времени от n_1 абонентов со средними интервалами времени $T_1, T_2, ..., T_{n_1}$.

Сообщения могут быть n_2 категорий с вероятностями их появления $p_{к1}, p_{к2}, ..., p_{кn_2}$ $(p_{к1} + p_{к2} + ... + p_{кn_2} = 1)$ и вычислительными сложностями обработки $S_1, S_2, ..., S_{n_2}$ операций соответственно.

Маршрутизатор имеет k входов и k выходов, входной буфер 1 ёмкостью L_1 байт для хранения сообщений, ожидающих обработки. В маршрутизаторе сообщения обрабатываются вычислительным комплексом (ВК) с производительностью Q операций/с. В случае полного заполнения буфера 1 поступающие на маршрутизатор сообщения теряются. Принято допущение, что одна операция вычислительной сложности соответствует одному байту при размещении сообщения в буфере.

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 ёмкость L_{2i} байт, $i = \overline{1,k}$. В случае полного заполнения буфера направления поступающее сообщение теряется.

Из буферов сообщения передаются по своим направлениям. Каждое направление имеет основной и резервный каналы связи. Скорость передачи сообщений по основному и резервному каналам связи каждого из направлений $V_{пi}$ бит/с, $i = \overline{1,k}$.

ВК и основные каналы связи имеют конечную надёжность. Интервалы времени $T_{отВК}$ и $T_{отК_1}, T_{отК_2}, ..., T_{отК_n}$ между отказами ВК и каналов связи случайные. Длительности восстановления ВК и каналов связи $T_{вВК}$ и $T_{вК_1}, T_{вК_2}, ..., T_{вК_n}$ также случайные.

При отказе обрабатываемые ВК и передаваемые по каналам связи сообщения теряются.

4.1.2. Исходные данные

$n_1 = 6;\ k = 4;$

$\text{exponential}(T_1) = \text{exponential}(T_2) = ... = \text{exponential}(T_6) = \text{exponential}(30);$

$n_2 = 4;\ p_{к1} = 0,3;\ p_{к2} = 0,2;\ p_{к3} = 0,2;\ p_{к4} = 0,3;$

$Q = 40000$ оп/с; $L_1 = 5000000;$

$\text{normal}(S_1, S_{о1}) = \text{normal}(53000, 6100);$

$\text{normal}(S_3, S_{о3}) = \text{normal}(66000, 7000);$

$\text{normal}(S_4, S_{о4}) = \text{normal}(50000, 500);$

$\text{exponential}(T_{отК_1}) = ... = \text{exponential}(T_{отК_8}) = \text{exponential}(360);$

$\text{exponential}(T_{вК_1}) = ... = \text{exponential}(T_{вК_8}) = \text{exponential}(3.2);$

$\text{exponential}(T_{отВК}) = \text{exponential}(3600),\ \text{exponential}(T_{вВК}) = \text{exponential}(3.7);$

$V_{пi} = 5000$ бит/с, $n_{3i} = 1,\ L_{2i} = 250000,\ i = \overline{1,4}.$

4.1.3. Задание на исследование

Разработать имитационную модель функционирования сети связи. Исследовать влияние ёмкостей буферов, интервалов времени поступления сообщений, их вычислительных сложностей и других параметров на показатели функционирования сети с целью их оценки и принятия решений при необходимости по улучшению качества обслуживания сети.

4.1.4. Формализованное описание модели

Сообщения поступают от $n_1 = 6$ источников. Интервалы поступления сообщений, интервалы между отказами и время восстановления работоспособности распределяются по экспоненциальному закону (exponential), а вычислительные сложности сообщений в зависимости от категорий — по нормальному закону (normal). Для некоторых одинаковых параметров с целью упрощения принято, что они имеют равные значения, например, средние значения интервалов поступления сообщений. Модель же будет построена в некотором приближении универсальной так, что все эти значения могут быть любыми.

Вариант сети связи при принятых исходных данных (в том числе количестве входов и выходов маршрутизатора) может быть представлен в следующем виде (рис. 4.1).

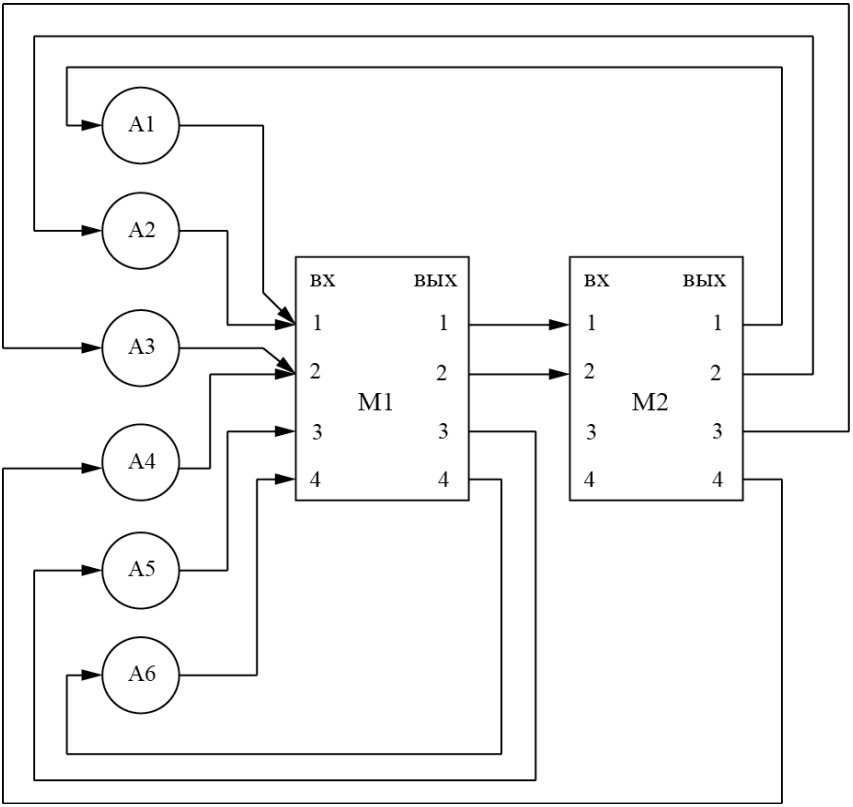


Рис. 4.1. Вариант схемы сети связи

Сообщения абонентов А1 и А2 поступают на вход 1 маршрутизатора М1, абонентов А3 и А4 — на вход 2 маршрутизатора М1, абонента А5 — на вход 3 маршрутизатора М1, абонента А6 — на вход 4 маршрутизатора М1.

Маршрутизатор М1 настроен таким образом, что сообщения, адресованные абонентам А1 и А2 поступают на вход 1 маршрутизатора М2, а абонентам А3 и А4 — на вход 2 маршрутизатора М2.

Маршрутизатор М2 настроен так, что его выходы 1…4 подключены к каналам связи, по которым передаются сообщения, адресованные абонентам А1…А4 соответственно.

Видно, что система связи представляет собой многофазную многоканальную систему массового обслуживания замкнутого типа с ограниченными ёмкостями буферов (накопителей), то есть с отказами.

Теперь представим маршрутизатор также как систему массового обслуживания и в общем виде (рис. 4.2).

Рис 4.2. Маршрутизатор как система массового обслуживания

В маршрутизаторе выделены блок контроля 1, накопитель (буфер) ВК, непосредственно ВК, блок контроля 2, накопители направлений связи.

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

Следует иметь в виду, что возможны и другие варианты декомпозиции сети связи как системы. Тем более что нами взята с учебной точки зрения одна из простых схем лишь для демонстрации цели работы.

Сообщения, поступающие на маршрутизатор, должны иметь следующие параметры:

`numAbOtpr` — номер абонента-отправителя сообщения;

`numAbPol` — номер абонента-получателя сообщения;

`numKat` — номер категории сообщения;

`timeOtpr` — время отправления сообщения абонентом;

`dlina` — длина сообщения, байт;

`timeObr` — время обработки сообщения ВК, с;

`timePered` — время передачи сообщения по каналу связи, с.

Эти параметры получаются путем розыгрыша по следующим исходным данным:

`kolAbonent` — количество абонентов;

`timeAbonent` — среднее время отправления сообщений абонентом, с;

`verKat={verKat1, verKat2, verKat3, verKat4}` — вероятности распределения видов сообщений первой, второй, третьей и четвёртой категорий соответственно,

`verKat1+verKat2+verKat3+verKat4=1`;

`dlKat={dlKat1, dlKat2, dlKat3, dlKat4}` — средние длины сообщений, байт, первой, второй, третьей и четвёртой категорий соответственно;

`dlKat0={dlKat01, dlKat02, dlKat03, dlKat04}` — стандартные отклонения длин сообщений, байт, первой, второй, третьей и четвёртой категорий соответственно;

`proizvod` — производительность ВК при обработке сообщения, оп/с;

`skorPeredKan` — среднее время передачи сообщений по каналу связи, бит/с;

skorPeredKanR — среднее время передачи сообщений по резервному каналу связи, бит/с;

timeBklKanR — время включения резервного канала связи.

В ходе моделирования собирается следующая статистика:

kolOtprKat1, kolOtprKat2, kolOtprKat3, kolOtprKat4, kolOtpr — количество отправленных абонентом сообщений первой, второй, третьей, четвёртой и всех категорий соответственно;

kolPolKat1, kolPolKat2,kolPolKat3,kolPolKat4, kolPol — количество полученных абонентом сообщений первой, второй, третьей, четвёртой и всех категорий соответственно;

всегоОтпрКат1, всегоОтпрКат2, всегоОтпрКат3, всегоОтпрКат4, всегоОтпр — количество отправленных в сети сообщений первой, второй, третьей, четвёртой и всех категорий соответственно;

всегоПолКат1, всегоПолКат2, всегоПолКат3, всегоПолКат4, всегоПол — количество полученных в сети сообщений первой, второй, третьей, четвёртой и всех категорий соответственно;

количество отправленных сообщений каждым абонентом каждому абоненту сети;

количество полученных сообщений каждым абонентом от каждого абонента сети.

По этим статистическим данным рассчитываются:

коэффициенты пропускной способности между всеми абонентами сети;

коэфПропСпособ — коэффициент пропускной способности сети связи;

врПередачи — среднее время передачи абоненту одного сообщения;

врПередСооб — среднее время передачи в сети связи одного сообщения.

Коэффициенты пропускной способности определяются как отношение количества полученных сообщений к отправленным сообщениям.

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

emkBuferVx — ёмкость входного буфера абонента, байт;

`tekEmkBuferVx` — текущую ёмкость входного буфера абонента, байт;

`emkBufer1` — ёмкость входного буфера маршрутизатора, байт;

`tekEmkBufer1` — текущая ёмкость входного буфера, байт;

`emkBuferNapr1`, `emkBuferNapr2`, `emkostBuferNapr3`, `emkBuferNapr4` — ёмкости на входах каналов связи первого, второго, третьего и четвёртого направлений соответственно, байт;

`tekEmkNapr1`, `tekEmkNapr2`, `tekEmkNapr3`, `tekEmkNapr4` — текущие ёмкости на входах каналов связи первого, второго, третьего и четвёртого направлений соответственно, байт.

Отказы и восстановление ВК и каналов связи разыгрываются по следующим данным:

`timeOtkBK` — среднее время между отказами ВК;

`timeVosstBK` — среднее время восстановления ВК;

`timeOtkKan` — среднее время между отказами канала связи;

`timeVosstKan` — среднее время восстановления канала связи.

4.1.5. Создание новых типов агентов

Построение модели начнём с создания активных объектов — экземпляров типов агентов. Напомним, что согласно рис. 4.1 в сети связи выделены следующие сегменты (компоненты):

абонент;

маршрутизатор;

канал.

Реализуем эти сегменты средствами AnyLogic, а потом из них, как из созданных нами объектов, будем строить сеть на корневом объекте Main.

Создайте три новых типа агентов: **Абонент**, **Канал**, **Маршрутизатор**.

1. Выполните команду **Файл/Создать/Модель** на панели инструментов. Появится диалоговое окно **Новая модель**.
2. Задайте имя новой модели. В поле **Имя модели** введите `Сеть_связи`. Выберите каталог, в котором будут сохранены файлы модели. Щёлкните **Готово**.
3. На панели **Проекты** щёлкните правой кнопкой Main и выберите из контекстного меню **Создать/Тип агента**.
4. Откроется диалоговое окно **Новый тип агента**.

5. Введите в поле **Имя:** имя нового типа агента `Абонент`.
6. Щёлкните **Готово**.
7. Также создайте типы агентов **Канал** и **Маршрутизатор**.

4.1.6. Создание областей просмотра

Последовательно переходите на типы агентов Main, **Абонент**, **Канал**, **Маршрутизатор** и создайте на каждом области просмотра. На них мы будем размещать элементы событийной части модели, а также исходные данные, переменные для промежуточных вычислений и результаты моделирования.

Значения свойств элементов **Область просмотра** установите согласно табл. 4.1.

Для всех областей просмотра оставьте **Выравнивать по:** `Верхнему левому углу`, установите из списка **Масштабирование:** `Подогнать под окно`.

Таблица 4.1

Свойства	Объекты			
	Main	Абонент	Маршрутизатор	Канал
Имя: X: Y: Ширина: Высота:	облСеть 0 0 770 470	облАбонент 0 0 810 360	облМарш1 0 0 740 470	облКанал 0 0 400 340
Имя: X: Y: Ширина: Высота:	viewData 870 600 470 620	viewData 0 700 450 420	viewData 0 550 330 250	viewData 0 700 400 180
Имя: X: Y: Ширина: Высота:	viewResult 0 600 442 820	viewResult 0 1200 470 432		

Приступим к созданию сегментов сети связи.

4.1.7. Сегмент Абонент

4.1.7.1. Исходные данные

1. Откройте тип агента **Абонент**.
2. Перейдите на область просмотра viewData.

3. Перетащите из палитры **Презентация** элемент **Скругленный прямоугольник**. На странице **Основные** оставьте **Имя:**, предложенное системой.

4. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `5`, **Y:** `745`, **Ширина:** `435`, **Высота:** `365`.

5. Перетащите два элемента **text** и на странице **Текст** панели **Свойства** вместо text введите `Исходные данные` и `Результаты` соответственно (рис. 4.3).

6. Переход между областями просмотра в ходе моделирования организуйте позднее. Поэтому на относящиеся к этому элементы **Сеть** и **Абонент1** пока не обращайте внимания.

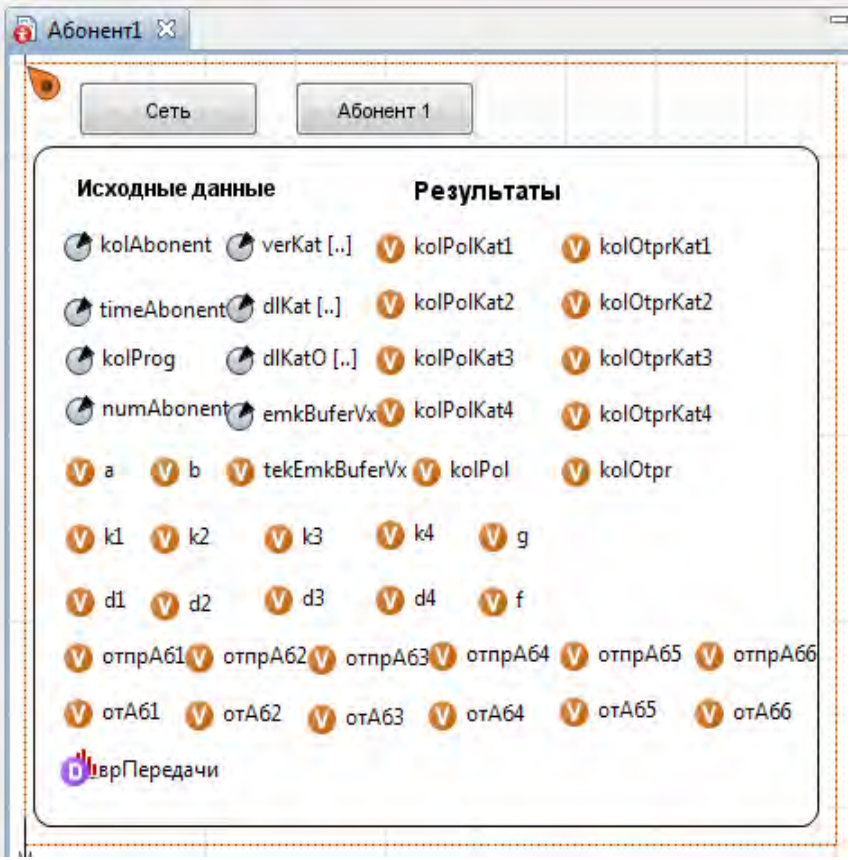


Рис. 4.3. Размещение элементов **Параметр** и **Переменная**

7. Перетащите из палитры **Основная** элементы **Переменная**, разместите и дайте им имена согласно рис. 4.3. Тип всех простых переменных, кроме tekEmkBuferVx, установите double. Тип tekEmkBuferVx — int. Переменные предназначены для промежуточных вычислений и сбора статистических данных за одного абонента сети.

8. Перетащите элементы **Параметр**, разместите и дайте имена как на рис. 4.3. Тип всех элементов **Параметр**, кроме emkBuferVx и numAbonent, double. Тип emkBuferVx и numAbonent — int. Значения свойств установите согласно табл. 4.2.

Таблица 4.2

Имя	Массив	Размерность	Значение по умолчанию
kolAbonent			6
timeAbonent			30
kolProg			100
numAbonent			1
emkBuferVx			80000
verKat	Установить	KolKat	{0.3, 0.5, 0.7, 1.0}
dlKat	флажки	KolKat	{53000,86000,66000,50000}
dlKatO		KolKat	{6100,5000,7000,500}

На рис. 4.2 видно, что элементы verKat, dlKat, dlKatO используются как массивы, размерность которых равна числу категорий сообщений. Создайте размерность KolKat.

1. Щёлкните правой кнопкой мышки в панели **Проекты** и в контекстном меню выберите **Размерность**.

2. В открывшемся окне **Размерность** в поле **Имя:** введите `KolKat`. Если введёте `kolKat`, в окне **Ошибки** появится сообщение: `По соглашению, имена типов Java обычно начинаются с заглавной буквы.`

3. Установите **Тип размерности: Диапазон**.

4. В открывшееся поле **Диапазон:** введите `1-4`.

5. Щёлкните **Готово**.

Теперь создайте массив verKat.

1. Выделите элемент **Параметр** (см. рис. 4.3).

2. На странице **Основные** панели **Свойства** установите флажок **Массив**. Откроется окно **Размерности**. Щёлкните по расположенной справа от окна и подсвеченной зелёным кнопке.

3. Откроется окно **Edit dimensions**. В окошке **Возможные размерности:** выделите `KolКат`.

4. Щёлкните по кнопке . Размерность `KolКат` появится в окошке **Выбранные размерности**.

5. Закройте окно. Вы вернётесь на панель **Свойства**. В окошке **Размерности** вы увидите размерность `KolКат`.

6. Щёлкните **Редактировать значения массива**. Откроется одноимённое диалоговое окно.

7. В левой части окна стрелками показано размещение элементов массива. Оставим горизонтальное. Элементы массива имеют разные значения. Поэтому не используем **[ВСЕ]**.

8. В правой части окна введите значения элементов массива:

```
0.3, 0.5, 0.7, 1.0
```

9. Создайте также массивы `dlKat`, `dlKatO` одинаковой размерности KolKat. Данные в поле **По умолчанию** введите из табл. 4.2.

Для получения стандартной статистической информации о времени передачи сообщений по каждому абоненту добавьте элемент сбора статистики.

1. Перетащите элемент **Данные гистограммы** с палитры **Статистика**.

2. Задайте свойства элемента:

измените **Имя:** на `врПередачи`;

сделайте **Количество интервалов:** равным `10`;

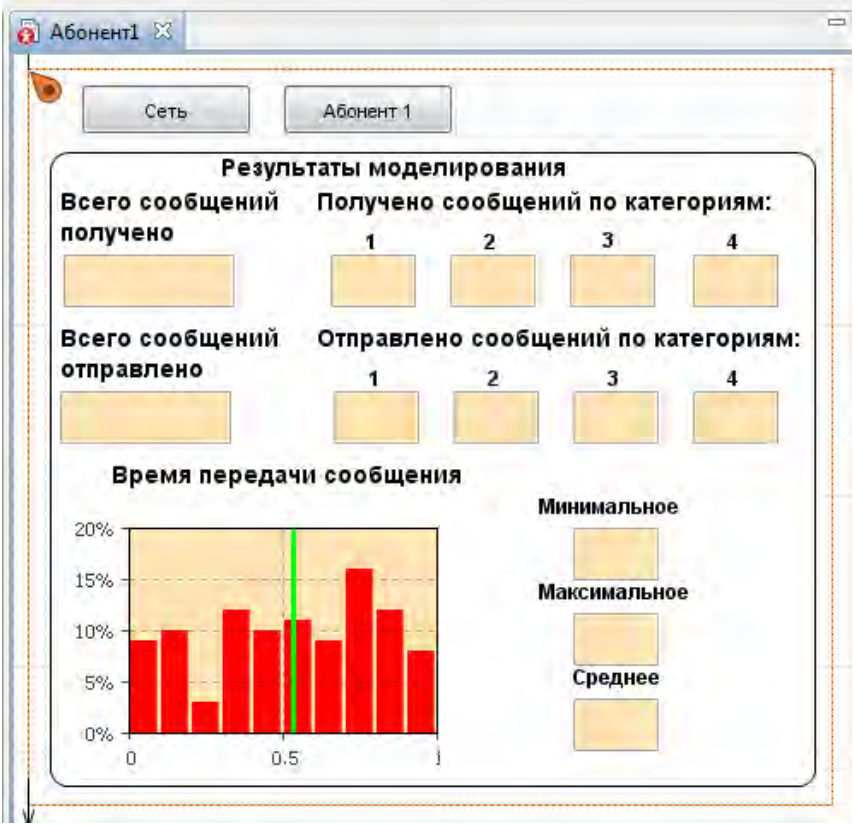
задайте **Нач. размер интервалов:** `0.01`.

Теперь можно было бы приступить к построению событийной части сегмента. Однако вы решили вывести результаты моделирования в более презентабельном виде. Поэтому целесообразнее организовать этот вывод, поскольку полученные при этом данные (имена, опции и др.) необходимы будут в дальнейшем при написании кода в событийной части сегмента.

4.1.7.2. Результаты моделирования по каждому абоненту

Вариант размещения элементов для вывода результатов моделирования показан на рис. 4.4.

Для вывода количественных показателей использован элемент **Текстовое поле** из палитры **Элементы управления** и элемент **Гистограмма** из палитры **Статистика**.



4.4. Размещение элементов для вывода результатов моделирования

1. В **Палитре** выделите **Презентация**. Перетащите элемент **Скругленный прямоугольник** в область просмотра **облАбонент**.

2. На странице **Основные** в поле **Имя:** оставьте имя, предложенное системой. Сбросьте флажок **Отображать имя**.

3. На странице **Местоположение и размер** панели **Свойства**. Введите в поля **X:** `13`, **Y:** `1250`, **Ширина:** `447`, **Высота:** `371`.

4. Перетащите элемент **text**. В поле **Текст:** введите `Результаты моделирования`.

5. В **Палитре** выделите **Элементы управления**. Перетащите элементы **Текстовое поле** на элемент `Результаты моделирования`. Разместите их так, как показано на рис. 4.4. Значения свойств установите согласно табл. 4.3. У элементов editbox и editbox5 **Ширина:** `100`, **Высота:** `31`. У остальных — **Ширина:** `50`.

Осталось поместить элемент для обработки и вывода статистической информации о времени передачи одного сообщения.

1. В **Палитре** выделите **Статистика**.

2. Перетащите элемент **Гистограмма**. Характеристики его размещения даны в табл. 4.3.

Таблица 4.3

<table>
<tr><td>Всего сообщений получено</td><td colspan="4">Получено сообщений по категориям</td></tr>
<tr><td>editbox
X: 21
Y: 1310</td><td>editbox1
X: 177
Y: 1310</td><td>editbox2
X: 247
Y: 1310</td><td>editbox3
X: 317
Y: 1310</td><td>editbox4
X: 389
Y: 1310</td></tr>
<tr><td>Всего сообщений отправлено</td><td colspan="4">Отправлено сообщений по категориям</td></tr>
<tr><td>editbox5
X: 19
Y: 1390</td><td>editbox6
X: 179
Y: 1390</td><td>editbox7
X: 249
Y: 1390</td><td>editbox8
X: 319
Y: 1390</td><td>editbox9
X: 389
Y: 1390</td></tr>
<tr><td colspan="3">Время передачи сообщения</td><td colspan="2"></td></tr>
<tr><td>Минимальное</td><td>Максимальное</td><td>Среднее</td><td colspan="2"></td></tr>
<tr><td>editbox13
X: 319
Y: 1470</td><td>editbox12
X: 319
Y: 1520</td><td>editbox15
X: 319
Y: 1570</td><td colspan="2"></td></tr>
<tr><td colspan="5">chart-Гистограмма</td></tr>
<tr><td>X: 19</td><td>Y: 1450</td><td>Ширина: 240</td><td colspan="2">Высота: 160</td></tr>
</table>

3. На странице **Основные** панели **Свойства** в поле **Имя:** оставьте имя, предложенное системой. Сбросьте флажок **Отображать имя**.

4. Щёлкните кнопку **Добавить данные** и введите в поле **Данные:** имя `врПередачи`.

5. В поле **Заголовок:** введите `Время передачи`.

6. Значения остальных свойств (цвет и др.) установите по собственному усмотрению.

7. Добавьте остальные элементы презентации (номера категорий сообщений, другую текстовую информацию, например, Всего сообщений получено и др.) согласно рис. 4.4.

И снова, прежде чем перейти к построению событийной части сегмента **Абонент**, разместим на корневом агенте Main необходимые элементы для сбора статистических данных, обработки, расчёта показателей качества обслуживания сети и их вывода. Это необходимо сделать по той причине, что также имена этих элементов нужны будут при написании кода в сегменте **Абонент**.

4.1.7.3. Показатели качества обслуживания сети связи

1. Откройте корневой объект Main.
2. Перейдите на область просмотра **ViewData**.

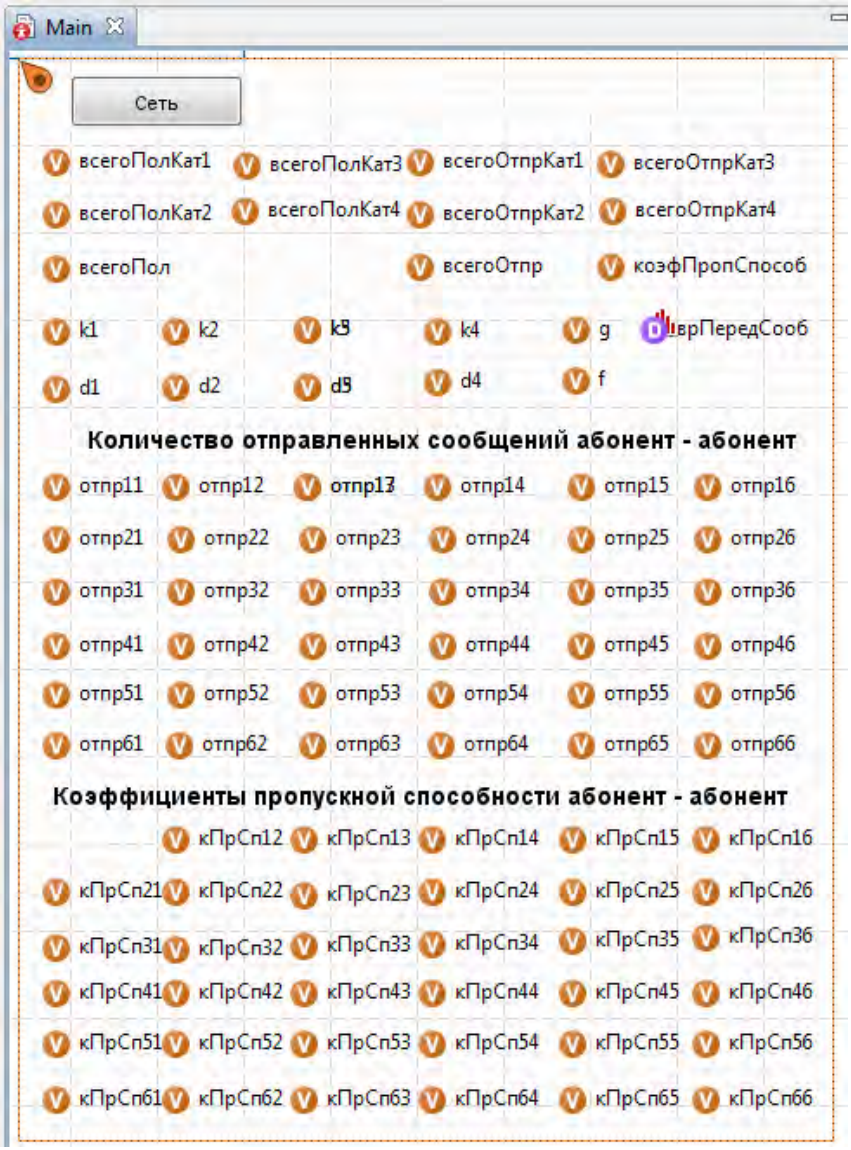


Рис. 4.5. Элементы для сбора и вывода данных за сеть в целом

3. Перетащите или скопируйте элементы **Переменная**, разместите и дайте имена согласно рис. 4.5.

4. Тип всех переменных double. Переменные отпр11…отпр66 предназначены для накопления статистических данных о количестве отправленных сообщений абонент — абонент. Например, отпр23 — количество сообщений, отправленных абонентом 2 абоненту 3, а отпр32 — количество сообщений, отправленных абонентом 3 абоненту 2. По этим данным рассчитываются коэффициенты пропускной способности кПрСп12…кПрСп66 абонент — абонент.

5. Выровняйте элементы. Выделите нужные элементы, Щёлкните правой кнопкой мышки, выберите **Выравнивание**.

6. Перетащите элемент **Данные гистограммы** с палитры **Статистика**.

7. Задайте свойства элемента:

измените **Имя:** на `врПередСооб`;

сделайте **Количество интервалов:** равным `10`;

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

Теперь организуем вывод в более презентабельном виде. На рис. 4.6 показаны элементы для вывода количественных показателей качества обслуживания сети связи. Часть результатов моделирования будем выводить в таком же виде и такие же, как и для каждого абонента.

1. Перейдите на область просмотра **ViewResult**.

2. Откройте объект **Абонент**.

3. Скопируйте элементы в скруглённом прямоугольнике с именем **Результаты моделирования**.

4. Перейдите на корневой объект Main и вставьте скопированные элементы в верхнюю часть области просмотра **ViewResult** (см. рис. 4.6).

5. Добавьте элемент **Текстовое поле** (editbox10) для вывода коэффициента пропускной способности сети.

6. Внесите правки в текстовые сообщения согласно рис. 4.6.

7. В **Палитре** выделите **Статистика**. Перетащите элемент **Гистограмма**.

8. На странице **Основные** панели **Свойства** в поле **Имя:** оставьте имя, предложенное системой. Сбросьте флажок **Отображать имя**.

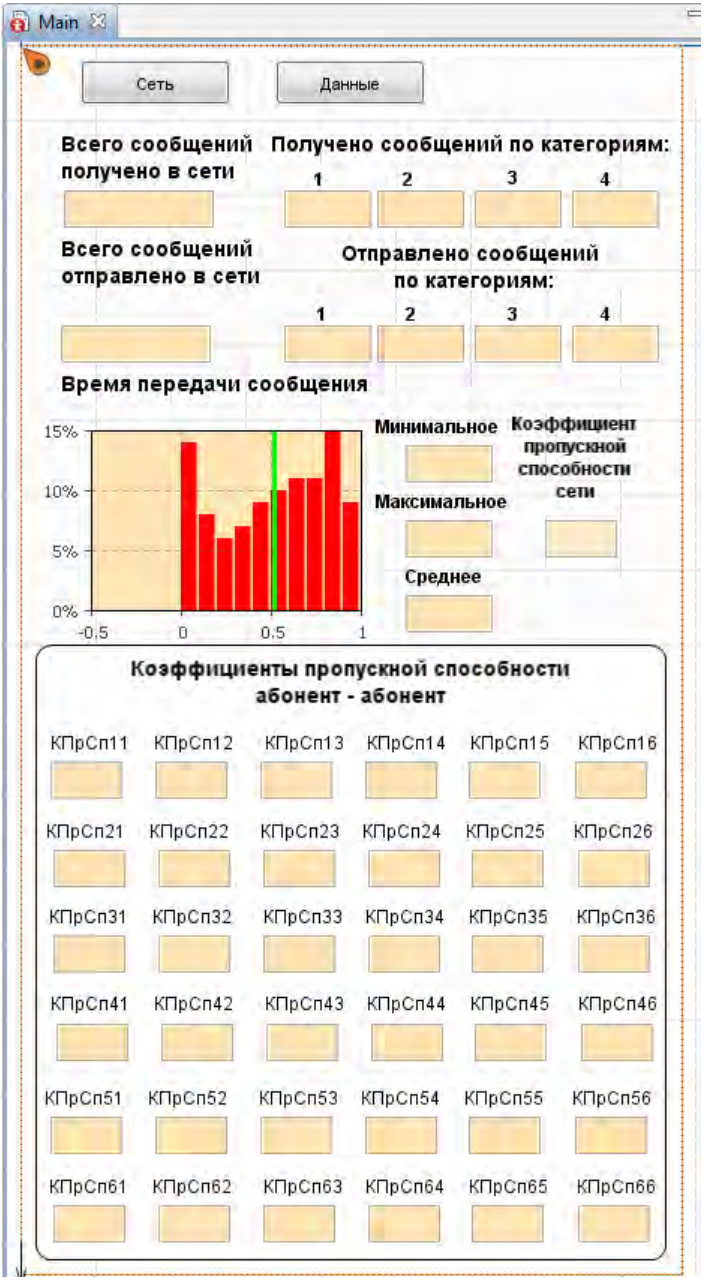


Рис. 4.6. Вывод показателей качества обслуживания сети

9. Щёлкните кнопку **Добавить данные** и введите в поле **Данные:** имя `врПередачи`.

10. В поле **Заголовок:** введите `Время передачи`.

11. Значения остальных свойств (цвет и др.) установите по собственному усмотрению.

12. В **Палитре** выделите **Презентация**. Перетащите элемент **Скругленный прямоугольник** в область просмотра результатов **ViewResult**.

13. На странице **Основные** в поле **Имя:** оставьте имя, предложенное системой. Сбросьте флажок **Отображать имя**.

14. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `10`, **Y:** `1000`, **Ширина:** `420`, **Высота:** `410`.

15. Перетащите элемент **text**. В поле **Текст:** введите `Коэффициенты пропускной способности абонент-абонент`.

16. В **Палитре** выделите **Элементы управления**. Перетащите элемент **Текстовое поле**. Разместите **X:** `20`, **Y:** `1078`, **Ширина:** `48`, **Высота:** `25`. В поле **Имя:** введите `КПрСп11`. Сбросьте флажок **Отображать имя**.

17. Перетащите элемент **text**. В поле **Текст:** введите `КПрСп11`.

18. Скопируйте элементы **text** и **Текстовое поле**. Вставьте их пять раз согласно рис. 4.6. При этом в полях **Имя:** последовательно меняются имена `КПрСп12`… `КПрСп16`. Но поле **Текст:** останется неизменным, т.е. `КПрСп11`. Внесите правки в соответствии с рис. 4.6. Элемент **text** добавлен потому, что в ходе моделирования имя элемента **Текстовое поле** не отображается.

19. Аналогичным образом добавьте остальные элементы **text** и **Текстовое поле** согласно рис. 4.6.

Теперь можно перейти к построению событийной части сегмента.

4.1.7.4. Построение событийной части сегмента

Событийная часть сегмента, назовём её блок **Абонент1**, предназначена для имитации отправителя-получателя сообщений, поступления сообщений через случайные интервалы времени, розыгрыша параметров сообщений и счёта количества всего отправленных-полученных сообщений по категориям и абонентам, запоминания времени поступления каждого сообщения, используемого в последующем для расчёта минимального, максимального и среднего времени передачи одного сообщения.

Алгоритм блока **Абонент1** приведен на рис. 4.7, а его реализация средствами AnyLogic — на рис. 4.8.

1. Из палитры **Презентация** перетащите элемент **Прямоугольник**.

2. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `10`, **Y:** `50`, **Ширина:** `450`, **Высота:** `380`.

3. Перетащите элемент **text** и в поле **Текст:** введите `Абонент1`.

Рис. 4.7. Алгоритм блока **Абонент1**

4. Перетащите из **Библиотеки моделирования процессов** объект source. Поместите его так же, как на рис. 4.8. Остальные объекты блока **Абонент1** вы будете перетаскивать и размещать по мере необходимости.

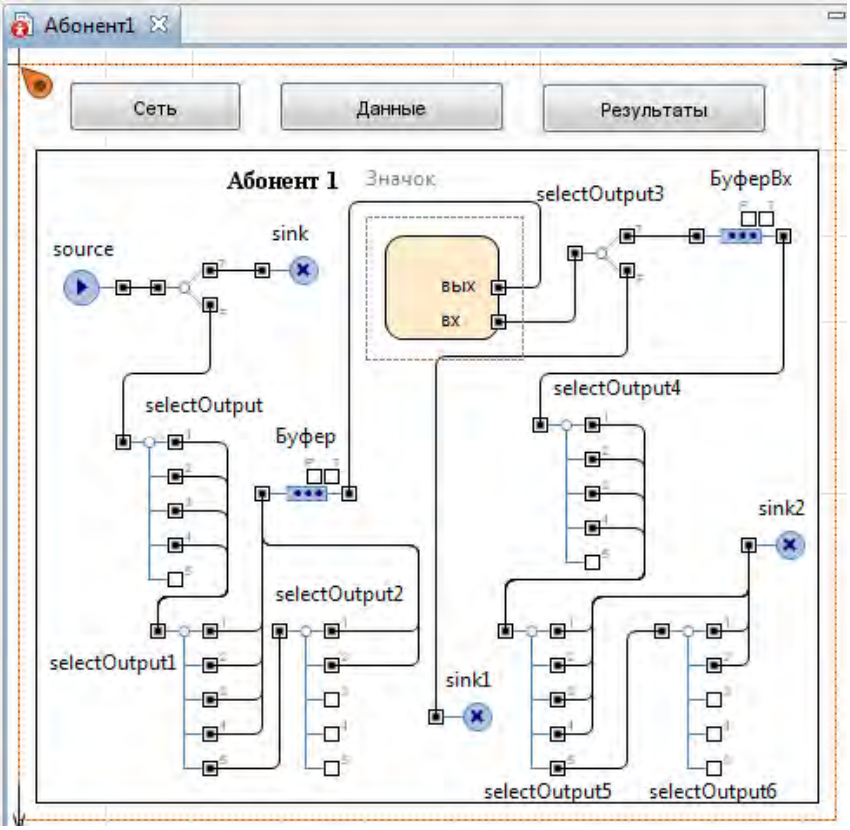


Рис. 4.8. Реализация блока **Абонент1** средствами AnyLogic

При построении модели нам придётся воспользоваться Java-кодом, в котором потребуются дополнительные поля сообщений. Для этого мы должны создать нестандартный тип заявки с дополнительными полями для записи и хранения параметров, о которых мы упоминали ранее (см. п. 4.1.4).

Создайте класс заявки `Message`.

1. Выделите объект **source**.
2. В панели **Проект** щёлкните правой кнопкой мыши элемент модели верхнего уровня дерева и выберите **Создать/Java класс**.
3. Появится диалоговое окно **Новый Java класс**. В поле **Имя:** введите имя нового класса: `Message`.
4. В поле **Базовый класс:** выберите из выпадающего списка `Entity` . Щёлкните кнопку **Далее**.

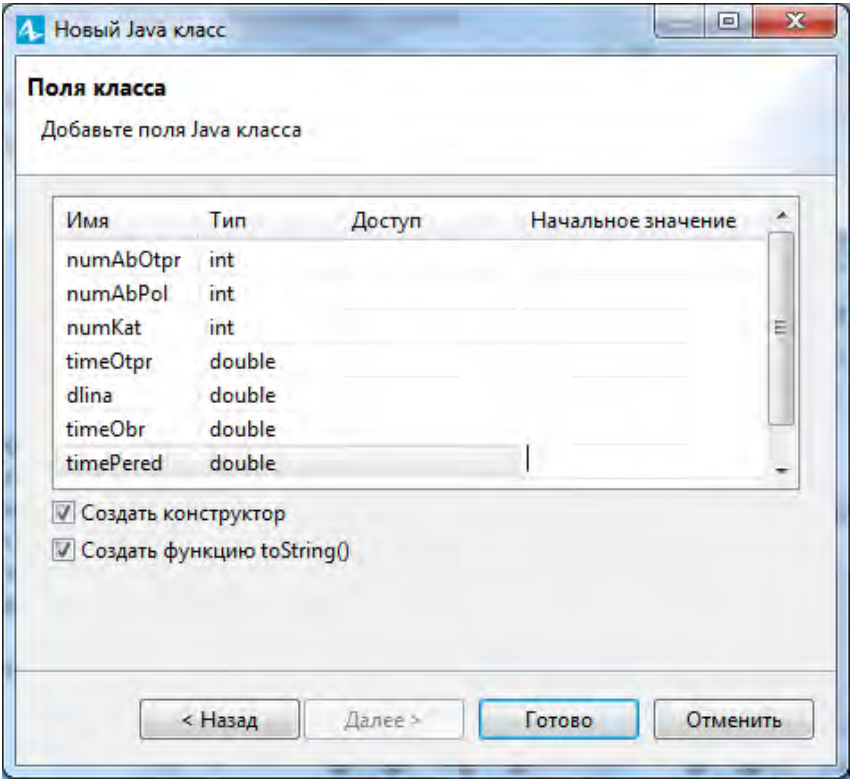


Рис. 4.9. Дополнительные поля класса заявок Message

5. Появится вторая страница **Мастера создания Java класса**. Добавьте поля Java класса, показанные на рис. 4.9.

6. Оставьте флажки **Создать конструктор** и **Создать метод toString ()**. Щёлкните **Готово**. Закройте редактор кода.

7. Щёлкните правой кнопкой мыши в панели **Проект** только что созданный Java класс и в контекстном меню выберите **Преобразовать Java класс в тип агента**. Закройте окно с параметрами.

Продолжим построение блока **Абонент1**.

1. Выделите объект source. На странице **Основные** панели **Свойства** установите следующие свойства:

Тип заявки: `Message`

Прибывают согласно `Времени между прибытиями`

Время между прибытиями `exponential(1/timeAbonent)`

Новая заявка `Message`

Действия При выходе:

```
double a=0;
double b=kolAbonent;
   // Розыгрыш номера получателя сообщения
    a=uniform();
 for (int i=1; i<(kolAbonent+1); i++)
     {if (a<=((1/kolAbonent)*b))
       entity.numAbPol=i;
         b=b-1;}
entity.timeOtpr=time();
entity.numAbOtpr=numAbonent;
```

Объект source будет генерировать сообщения с интервалами времени, распределёнными по экспоненциальному закону. Принято, что сообщения от данного абонента-отправителя с равной вероятностью могут быть отправлены любому абоненту сети. Поэтому разыгрывается номер абонента-получателя сообщения, и заносится в `numAbPol`. В поле `timeOtpr` заносится время отправления сообщения.

2. Перетащите объект **selectOutput**. Соедините его вход с выходом объекта source. Объект selectOutput выделяет и направляет объекту sink на уничтожение сообщения, адресованные абоненту-отправителю, то есть самому себе. Перетащите объект **sink**. Его вход соедините с выходом T объекта selectOutput.

3. Перетащите объект **selectOutput5**. Данный объект предназначен для розыгрыша категорий отправляемых сообщений. Его вход соедините с выходом F объекта **selectOutput**. На панели **Свойства** установите свойства согласно табл. 4.4. Эти свойства являются кодом. Код предназначен для счёта отправленных сообщений за сеть связи в целом и по категориям сообщений, определения количества отправленных сообщений за один прогон, розыгрыша длин сообщений по категориям, вывода расчётов по каждому абоненту, а также за сеть связи в целом.

4. Перетащите два объекта **selectOutput5** (имена selectOutput1 и selectOutput2). Объекты предназначены для разделения и счёта отправляемых сообщений по абонентам.

5. Перетащите объект **queue**.

6. Укажите его свойства: **Имя:** `Буфер`. **Тип заявки:** `Message`. **Вместимость** `100`. Соедините выходы 1…4 selectOutput1 с входом элемента **Буфер** (см. рис. 4.8).

7. Соедините выход по умолчанию selectOutput1 с входом selectOutput2. Через этот выход будут проходить на selectOutput2 сообщения, адресованные абонентам 5 и 6.

Таблица 4.4

Свойство	selectOutput
Тип заявки: **Выход true выбирается** **Условие**	Message При выполнении условия entity.numAbPol==numAbonent

Тип заявки:
Использовать:
Действия
При входе:
	Message
	Условия
	a=uniform();
	f++;
	kolOtpr=f/kolProg;
	editbox5.setText(kolOtpr, **true**);
	main.f++;
	main.всегоОтпр=main.f/kolProg;
	main.editbox5.setText(main.всегоОтпр, **true**);

Условие 1	a<=verKat.get(1)

Действия
При выходе 1
	entity.numKat=1;
	b=normal(dlKatO.get(entity.numKat), dlKat.get(entity.numKat));
	entity.dlina=(**int**)(b);
	d1++;
	kolOtprKat1=d1/kolProg;
	editbox6.setText(kolOtprKat1, **true**);
	main.d1++;
	main.всегоОтпрКат1=main.d1/kolProg;
	main.editbox6.setText(main.всегоОтпрКат1, **true**);

Условие 2	a<=ver verKat.get(2)

Действия
При выходе 2
	entity.numKat=2;
	b=normal(dlKatO.get(entity.numKat), dlKat.get(entity.numKat));
	entity.dlina = (**int**)(b);
	d2++;
	kolOtprKat2=d2/kolProg;
	editbox7.setText(kolOtprKat2, **true**);
	main.d2++;

Свойство	selectOutput
	main.всегоОтпрКат2=main.d2/kolProg; main.editbox7.setText(main. всегоОтпрКат2, true);
Условие 3	a<=verKat.get(3)

Действия При выходе 3	entity.numKat=3;
	b=normal(dlKatO.get(entity.numKat),
	dlKat.get(entity.numKat));
	entity.dlina=(int)(b);
	d3++;
	kolOtprKat3=d3/kolProg;
	editbox8.setText(kolOtprKat3, true);
	main.d3++;
	main.всегоОтпрКат3=main.d3/ kolProg;
	main.editbox8.setText(main.
	всегоОтпрКат3, true);

Условие 4	a<=verKat.get(4)

Действия При выходе 4	entity.numKat=4;
	b=normal(dlKatO.get(entity.numKat),
	dlKat.get(entity.numKat));
	entity.dlina=(int)(b);
	d4++;
	kolOtprKat4=d4/kolProg;
	editbox9.setText(kolOtprKat4, true);
	main.d4++;
	main.всегоОтпрКат4=main.d4/kolProg;
	main.editbox9.setText(main.
	всегоОтпрКат4, true);

Свойство	selectOutput1
Тип заявки: Использо-вать:	Message Условия
Условие 1	entity.numAbPol==1
Условие 2	entity.numAbPol==2
Действия При выходе 2	отпрАб2++; main.отпр12=отпрАб2;

Окончание табл. 4.4

Свойство	selectOutput1
Условие 3	`entity.numAbPol==3`
Действия При выходе 3	`отпрАб3++;` `main.отпр13=отпрАб3;`
Условие 4	`entity.numAbOtpr==4`
Действия При выходе 4	`отпрАб4++;` `main.отпр14=отпрАб4;`
Свойство	selectOutput2
Тип заявки: **Использовать:**	`Message` `Условия`
Условие 1	`entity.numAbPol==5`
Действия При выходе 1	`отпрАб5++;` `main.отпр15=отпрАб5;`
Условие 2	`entity.numAbPol==6`
Действия При выходе 2	`отпрАб6++;` `main.отпр16=отпрАб6;`

8. Выделите selectOutput2 и установите значения свойств, указанных также в табл. 4.4. Соедините все выходы selectOutput2 с входом элемента **Буфер**.

Отправляемое сообщение поступило в буфер. Дальше оно должно попасть в канал связи. Создайте порты для отправления и приёма сообщений.

1. Из палитры **Презентация** перетащите элемент **Скруглённый прямоугольник**.

2. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `210`, **Y:** 100, **Ширина:** `65`, **Высота:** `60`.

3. Из палитры **Основная** перетащите два элемента **порт**. Разместите их как на рис. 4.8. В поле **Имя:** замените предложенное системой на `вх` и `вых`.

4. *Обратите внимание на то, чтобы у элементов* **Скруглённый прямоугольник** *и* **порт** *был установлен флажок* **На верхнем уровне**. *У остальных элементов сегмента* **Абонент1** *этот флажок должен быть сброшенным.* В противном случае вы обнаружите их также на верхнем уровне — на агенте Main.

Итак, отправляемое сообщение поступило в канал связи, подключённый к порту `вых`.

Через порт `вх` сообщения поступают из канала связи. Продолжим построение блока **Абонент1**.

1. Перетащите объект selectOutput (имя selectOutput3). Он предназначен для контроля текущей ёмкости входного буфера. В случае заполнения буфера, сообщения теряются.
2. Соедините порт `вх` с входом объекта selectOutput3.
3. Перетащите объект sink (имя sink1). Его вход соедините с выходом F объекта selectOutput3.
4. Выделите объект selectOutput3 и установите его свойства:

Тип заявки: `Message`

Выход true выбирается `При выполнении условия`

Условие `emkBuferVx-tekEmkBuferVx>=entity.dlina`

5. Перетащите объект queue. Укажите его свойства:

Имя: `БуферВх`

Тип заявки: `Message`

Вместимость `emkBuferVx`

Действие при входе `tekEmkBuferVx+=entity.dlina;`

Действия При выходе `tekEmkBuferVx-=entity.dlina;`

Соедините его вход с выходом T объекта selectOutput3.

6. Перетащите selectOutput5 (имя selectOutput4). Он необходим для разделения потока полученных сообщений по категориям.
7. На страницах панели **Свойства** установите свойства согласно табл. 4.5. Эти свойства также являются кодом. Код предназначен для:

счёта полученных сообщений за сеть связи в целом и по категориям сообщений,

определения числа полученных сообщений за один прогон,

вывода расчётов за каждого абонента, а также за сеть связи в целом.

8. Соедините выход элемента `БуферВх` с выходом объекта selectOutput4.
9. Перетащите два объекта selectOutput5 (имена selectOutput5 и selectOutput6). Объекты предназначены для разделения и счёта получаемых сообщений по абонентам. Значения свойств установите также согласно табл. 4.5. Выход по умолчанию объекта selectOutput5 соедините с входом объекта selectOutput6.

Таблица 4.5

Свойство	selectOutput4

Тип заявки:
Использовать:
Действия
При входе:
	Message
	Условия
	g++;
	kolPol=g/kolProg;
	editbox.setText(kolPol, **true**);
	main.g++;
	main.всегоПол=main.g/
	kolProg;
	main.editbox.setText(main.
	всегоПол, **true**);

Условие 1	entity.numKat==1

Действия
При выходе 1
	k1++;
	kolPolKat1=k1/kolProg;
	editbox1.setText(kolPolKat1, **true**);
	main.k1++;
	main.всегоПолКат1=main.k1/kolProg;
	main.editbox1.setText(main.
	всегоПолКат1, **true**);

Условие 2	entity.numKat==2

Действия
При выходе 2
	k2++;
	kolPolKat2=k2/kolProg;
	editbox2.setText(kolPolKat2, **true**);
	main.k2++;
	main.всегоПолКат2=main.k2/kolProg;
	main.editbox2.setText(main.
	всегоПолКат2, **true**);

Условие 3	entity.numKat==3

Действия
При выходе 3
	k3++;
	kolPolKat3=k3/kolProg;
	editbox3.setText(kolPolKat3, **true**);
	main.k3++;
	main.всегоПолКат3=main.k3/kolProg;
	main.editbox3.setText(main.
	всегоПолКат3, **true**);

Условие 4	`entity.numKat==4`

Действия При выходе 4	`k4++;`
	`kolPolKat4=k4/kolProg;`
	`editbox4.setText(kolPolKat4, true);`
	`main.k4++;`
	`main.всегоПолКат4=main.k4/kolProg;`
	`main.editbox4.setText(main.всегоПолКат4, true);`

Свойство	selectOutput5
Тип заявки: **Использовать:**	`Message` `Условия`
Условие 1	`entity.numAbOtpr==1`
Условие 2	`entity.numAbOtpr==2`

Действия При выходе 2	`отАб2++;`
	`main.кПрСп21=отАб2/main.отпр21;`
	`main.КПрСп21.setText(main.кПрСп21, true);`

Условие 3	`entity.numAbOtpr==3`

Действия При выходе 3	`отАб3++;`
	`main.кПрСп31=отАб3/main.отпр31;`
	`main.КПрСп31.setText(main.кПрСп31, true);`

Условие 4	`entity.numAbOtpr==4`

Действия При выходе 4	`отАб4++;`
	`main.кПрСп41=отАб4/main.отпр41;`
	`main.КПрСп41.setText(main.кПрСп41, true);`

Свойство	selectOutput6
Тип заявки: **Использовать:**	`Message` `Условия`
Условие 1	`entity.numAbOtpr==5`

Действия При выходе 1	`отАб5++;`
	`main.кПрСп51=отАб5/main. отпр51;`
	`main.КПрСп51.setText(main.кПрСп51, true);`

Окончание табл. 4.5

Условие 2	`entity.numAbOtpr==6`

Действия При выходе 2	`отАб6++;`
	`main.кПрСп61=отАб6/main.отпр61;`
	`main.КПрСп61.setText(main.кПрСп61, true);`

10. Перетащите объект sink2. Его вход соедините с выходами 1…4 объекта selectOutput5 и выходами объекта selectOutput6.

11. Установите свойства объекта sink2 согласно табл. 4.6. Код предназначен для расчёта коэффициента пропускной способности сети связи и времени передачи одного сообщения.

Таблица 4.6

Свойство	sink2
Тип заявки:	`Message`

Действия При входе:	`врПередачи.add(time()-entity.timeOtpr);`
	`editbox11.setText(врПередачи.mean(), true);`
	`editbox12.setText(врПередачи.max(), true);`
	`editbox13.setText(врПередачи.min(), true);`
	`main.коэфПропСпособ=main.всегоПол/main.всегоОтпр;`
	`main.editbox10.setText(main.коэфПропСпособ, true);`
	`main.врПередСооб.add(time()-entity.timeOtpr);`
	`main.editbox11.setText(main.врПередСооб.mean(), true);`
	`main.editbox12.setText(main.врПередСооб.max(), true);`
	`main.editbox13.setText(main.врПередСооб.min(), true);`

Приступим к построению сегмента **Маршрутизатор**.

4.1.8. Сегмент Маршрутизатор

4.1.8.1. Исходные данные

1. Откройте тип агента **Маршрутизатор**.
2. Перейдите на область просмотра viewData.
3. Перетащите из палитры **Презентация** элемент **Скругленный прямоугольник**. На странице **Основные** в поле **Имя:** оставьте имя, предложенное системой.
4. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `10`, **Y:** `600`, **Ширина:** `314`, **Высота:** `190`.
5. Перетащите элемент **text** и в поле **Текст:** введите `Исходные данные` (рис. 4.9.).
6. Перетащите элементы **Параметр** и **Переменная**. Разместите их и дайте имена согласно рис. 4.10.
7. Значения по умолчанию элементов **Параметр** установите согласно табл. 4.7.
8. Тип timeOtkBK, timeVosstBK, proizvod установите double, остальных параметров и простых переменных — int.

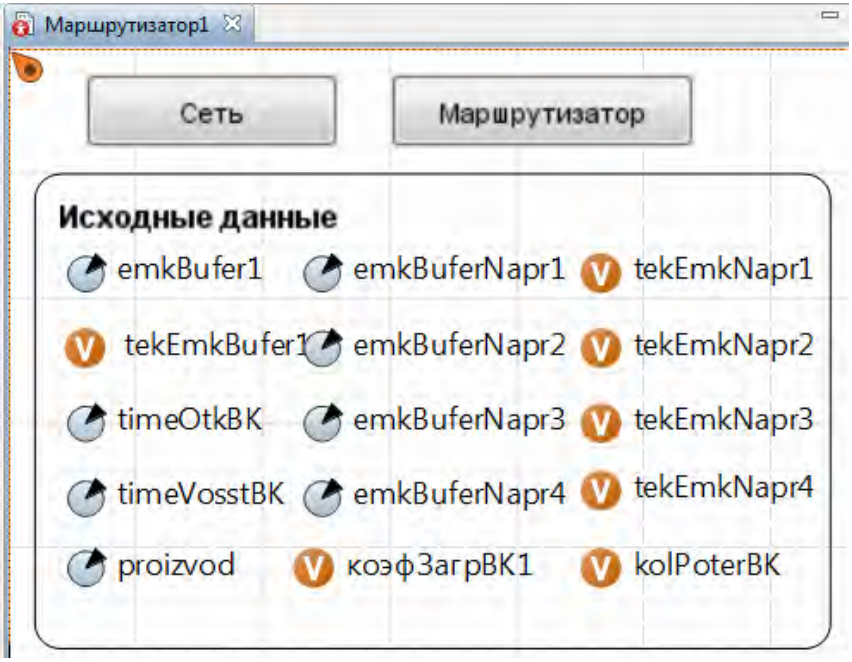


Рис. 4.10. Исходные данные сегмента Маршрутизатор

Таблица 4.7

Имя	Значение по умолчанию	Имя	Значение по умолчанию
emkBufer1	5000000	emkBuferNapr1	250000
timeOtkBK	3600	emkBuferNapr2	250000
timeVosstBK	3.7	emkBuferNapr3	250000
proizvod	40000	emkBuferNapr4	250000

4.1.8.2. Событийная часть сегмента Маршрутизатор

Элементы событийной части сегмента **Маршрутизатор** показаны на рис. 4.11. Сегмент включает **Вычислительный комплекс**, **Буфер 2**, **порты входа-выхода**, **Имитатор отказов вычислительного комплекса**.

В свою очередь вычислительный комплекс содержит **Блок контроля 1**, **Буфер 1**, **Блок обработки сообщений**.

Построим событийную часть сегмента **Маршрутизатор**.

4.1.8.2.1. 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. 4.12.

В AnyLogic этот алгоритм реализуется блоками **selectOutput**, **hold** и **sink**.

1. В **Палитре** выделите **Презентация**. Перетащите элемент **Прямоугольник**.
2. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `26`, **Y:** `50`, **Ширина:** 154, **Высота:** `150`.
3. Перетащите объекты **selectOutput**, **hold** и **sink** на диаграмму агента **Маршрутизатор**, разместите и соедините так, как показано на рис. 4.11.
4. Перетащите элемент **text** и на странице **Текст** панели **Свойства** вместо слова text введите `Блок контроля 1`.
5. Выделите объект **selectOutput**.
6. В поле **Имя:** вместо **selectOutput** введите `blokKontrol_1`.
7. В поле **Тип заявки:** `Agent` замените `Message`.

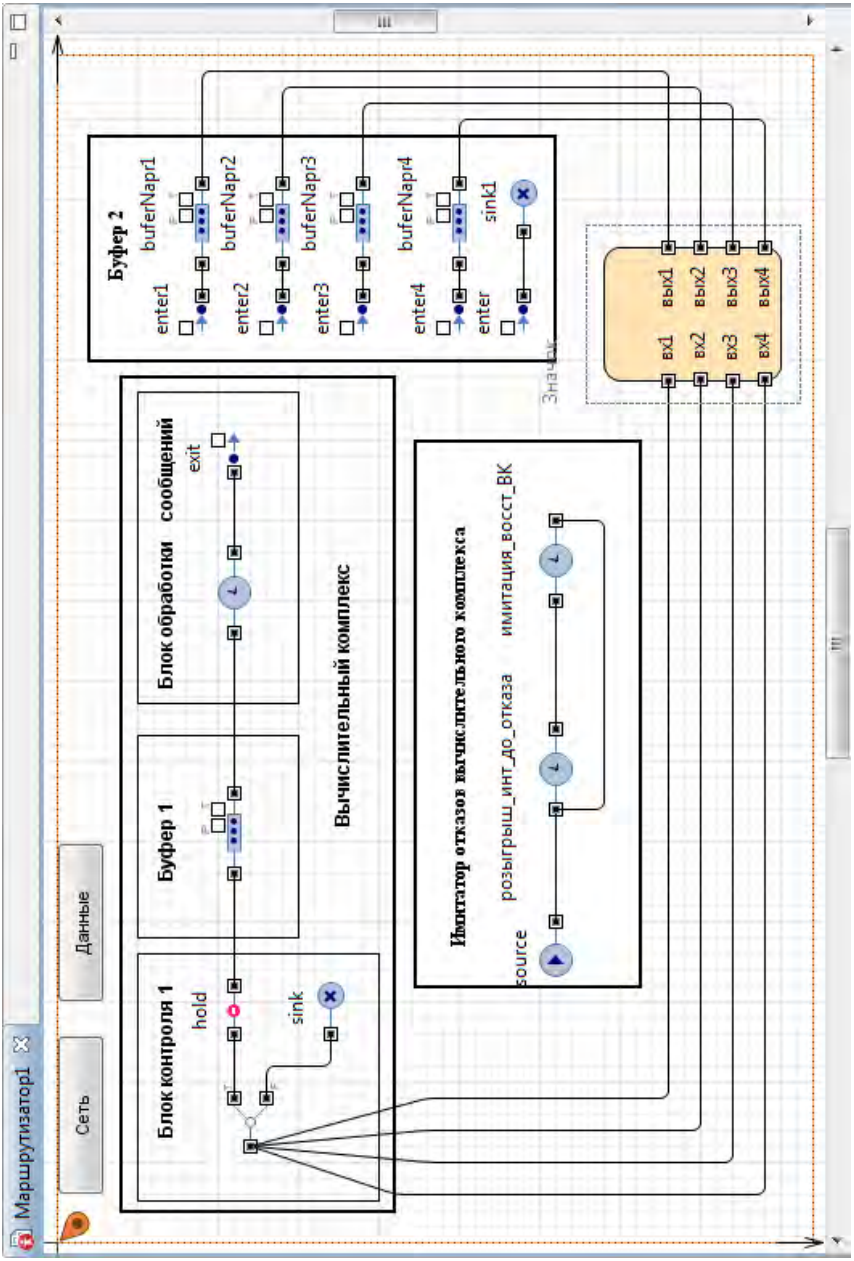


Рис. 4.11. Событийная часть сегмента **Маршрутизатор**

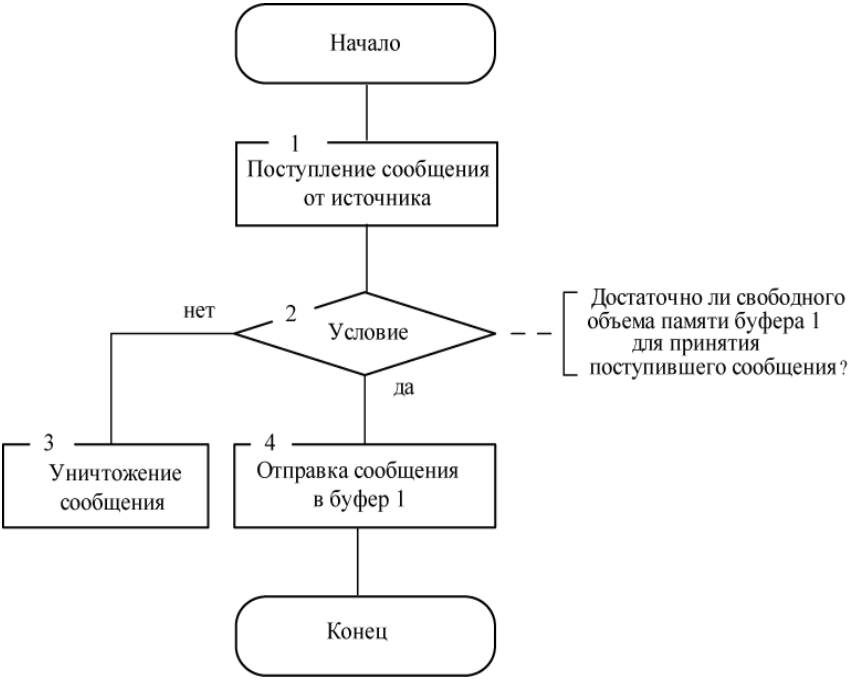


Рис. 4.12. Алгоритм работы **Блока контроля 1**

8. Установите **Выход true выбирается** `При выполнении условия`.

9. В поле **Условие** введите условие:

```
(emkostBufer1-tekEmkostBufer1>=entity.dlina)&&
(hold.isBlocked()==false)
```

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.

10. Выделите объект **sink**. В поле **Тип заявки:** `Agent` замените `Message`.

11. В поле **Действие при входе** введите `kolPotBK++`; для счёта количества сообщений, потерянных при отказе вычислительного комплекса.

12. Выделите элемент **hold**. В поле **Тип заявки:** `Agent` замените `Message`.

4.1.8.2.2. Блок Буфер 1

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

Алгоритм работы блока **Буфер 1** приведен на рис. 4.13.

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

1. В Палитре выделите **Презентация**. Перетащите элемент **Прямоугольник**.
2. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `190`, **Y:** `50`, **Ширина:** `126`, **Высота:** `100`.
3. На странице **Основные** панели **Свойства** в поле **Имя:** оставьте `Rectangle`. Не ставьте флажок **Отображать имя**.
4. Перетащите элемент **text** и на странице **Текст** панели **Свойства** вместо text введите `Буфер 1`.

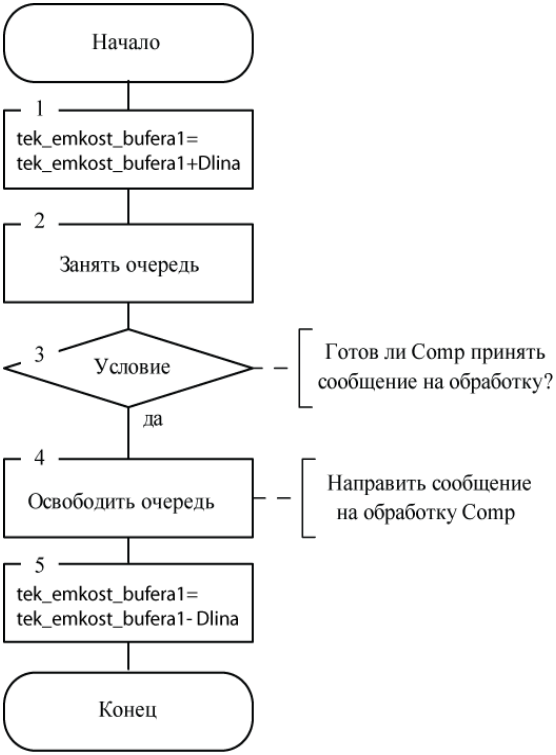


Рис. 4.13. Алгоритм работы блока **Буфер 1**

5. Выделите объект **queue**.
6. В поле **Имя:** вместо `queue` введите `bufer1`.
7. В поле **Тип заявки:** `Agent` замените `Message`.
8. В поле **Вместимость** введите `emkBufer1`.
9. При помещении сообщения в буфер его текущая емкость увеличивается, поэтому в поле **Действия При входе** введите:

```
tekEmkBufer1 += entity.dlina;
```

10. При выходе сообщения из буфера его текущая емкость уменьшается, поэтому в поле **Действия При выходе** введите:

```
tekEmkBufer1 -= entity.dlina;
entity.timeObr=entity.dlina/proizvod;
```

11. Поставьте флажок **Включить сбор статистики**.

4.1.8.2.3. Блок обработки сообщений

Блок предназначен для имитации обработки сообщений.

Алгоритм работы блока приведен на рис. 4.14.

Рис. 4.14. Алгоритм работы **Блока обработки сообщений**

Для реализации алгоритма **Блока обработки сообщений** в AnyLogic используется объект **delay**.

1. В **Палитре** выделите **Презентация**. Перетащите элемент **Прямоугольник**.
2. На странице **Основные** панели **Свойства** в поле **Имя:** оставьте `Rectangle`. Также не устанавливайте флажок **Отображать имя**.
3. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** 336, **Y:** `50`, **Ширина:** `194`, **Высота:** `100`.
4. Перетащите элемент **text** и на странице **Текст** панели **Свойства** вместо text введите название `Блок обработки сообщений`.
5. Перетащите объект **delay**, разместите и соедините с `bufer_1` так, как на рис. 4.11.
6. Выделите объект **delay**. На странице **Основные** панели **Свойства** в поле **Имя:** вместо **delay** введите `computer`.
7. В поле **Тип заявки:** `Agent` замените `Message`.
8. **Задержка задаётся** установите `Определённое время`.
9. В поле **Время задержки** введите:
   `exponential(1/entity.timeObr)`
10. Оставьте **Вместимость** `1`.
11. **Действие при выходе**
   `коэфЗагрВК1=computer.statsUtilization.mean();`
12. Установите флажок **Включить сбор статистики**.
13. В **Палитре** выделите **Презентация**. Перетащите элемент **Прямоугольник**.
14. На странице **Основные** панели **Свойства** в поле **Имя:** оставьте `Rectangle`. Не ставьте флажок **Отображать имя**.
15. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** 20, **Y:** 40, **Ширина:** `520`, **Высота:** `170`.
16. Перетащите элемент **text** и на странице **Текст** панели **Свойства** вместо слова text введите название `Вычислительный комплекс`.

4.1.8.2.4. Блок контроля 2

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

Алгоритм работы **Блока контроля 2** приведен на рис. 4.15.

Рис. 4.15. Алгоритм работы 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 объёма памяти сообщение теряется.

Для распределения сообщений по направлениям можно было бы использовать объекты **selectOutput5** и **sink**. Однако мы используем другие объекты AnyLogic: **exit** и **enter**. Они позволяют организовать сложную маршрутизацию, вследствие чего на рис.4.11 **Блок контроля 2** не выделен, хотя функционально он существует.

1. Перетащите объект **exit**, вход которого соедините с выходом `computer` (рис. 4.10).
2. В поле **Тип заявки:** `Agent` замените `Message`.
3. В поле **Действия При выходе** введите код:

```
int i;
  i=entity.numIstPol;
   {
       switch (i)
     {
          case 1:if (emkBuferNapr1-
tekEmkNapr1>=entity.dlina) {
                 enter1.take(entity);
```

```
                break;}
              else {enter.take(entity);
                break;}
          case 2:if (emkBuferNapr1-
tekEmkNapr1>=entity.dlina) {
                enter1.take(entity);
                break;}
              else {enter.take(entity);
                break;}
          case 3:if (emkBuferNapr2-
tekEmkNapr2>=entity.dlina) {
                enter2.take(entity);
                break;}
              else {enter.take(entity);
                  break;}
          case 4:if (emkBuferNapr2-
tekEmkNapr2>=entity.dlina) {
                enter2.take(entity);
                break;}
              else {enter.take(entity);
                break;}
          case 5:if (emkBuferNapr3-
tekEmkNapr3>=entity.dlina) {
                enter3.take(entity);
                break;}
              else {enter.take(entity);
                break;}
          case 6:if (emkBuferNapr4-
tekEmkNapr4>=entity.dlina) {
                enter4.take(entity);
                break;}
              else {enter.take(entity);
                break;}
        }
    }
```

Маршрутизатор настраивается определённым образом, например, таблицей маршрутизации. В данном случае он настраивается программным путём так, что сообщения первого и второго отправителей передаются по первому направлению, третьего и четвёртого отправителей — по второму направлению, пятого отправителя — по третьему и шестого отправителя — по четвёртому направлению. Такой вариант принят с учётом построения в дальнейшем сети связи (см. рис. 4.1).

4.1.8.2.5. Блок Буфер 2

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

Алгоритм работы буфера каждого из направлений такой же, как и алгоритм работы буфера 1 (см. рис. 4.13).

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

1. В **Палитре** выделите **Презентация**. Перетащите элемент **Прямоугольник** (см. рис. 4.11).
2. На странице **Основные** панели **Свойства** в поле **Имя:** оставьте предложенное системой `Rectangle`. Не устанавливайте флажок **Отображать имя**.
3. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** 550, **Y:** 20, **Ширина:** 140, **Высота:** `290`.
4. Перетащите элемент **text** и на странице **Основные** панели **Свойства** в поле **Текст:** введите `Буфер 2`.
5. Перетащите пять объектов **enter**, четыре объекта **queue** и один объект **sink**, разместите, дайте имена и соедините так, как на рис. 4.11.
6. Выделите элемент **buferNapr1**, вход которого соединён с выходом **enter1** и установите значения свойств.
7. В поле **Тип заявки:** `Agent` замените `Message`.
8. **Вместимость** `emkBuferNapr1`
9. В поле **Действия При входе** введите:

```
tekEmkNapr1 += entity.dlina;
```

10. В поле **Действия При выходе** введите:

```
tekEmkNapr1 -= entity.dlina;
```

11. Установите флажок **Включить сбор статистики**.
12. Выделите элемент **buferNapr2**, вход которого соединен с выходом **enter2** и установите значения свойств.
13. В поле **Тип заявки:** `Agent` замените `Message`.
14. **Вместимость** `emkBuferNapr2`
15. В поле **Действия При входе** введите:

```
tekEmkNapr2 += entity.dlina;
```

16. В поле **Действия При выходе** введите:

```
tekEmkNapr2 -= entity.dlina;
```

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

18. Выделите элемент **buferNapr3**, вход которого соединен с выходом **enter3** и установите значения свойств.

19. В поле **Тип заявки:** `Agent` замените `Message`.

20. **Вместимость** `emkBuferNapr3`

21. В поле **Действия При входе** введите:

```
tekEmkNapr3 += entity.dlina;
```

22. В поле **Действия При выходе** введите:

```
tekEmkNapr3 -= entity.dlina;
```

23. Установите флажок **Включить сбор статистики**.

24. Выделите элемент **buferNapr4**, вход которого соединен с выходом **enter4** и установите значения свойств.

25. В поле **Тип заявки:** `Agent` замените `Message`.

26. **Вместимость** `emkBuferNapr4`

27. В поле **Действия При входе** введите:

```
tekEmkNapr4 += entity.dlina;
```

28. В поле **Действия При выходе** введите:

```
tekEmkNapr4 -= entity.dlina;
```

29. Установите флажок **Включить сбор статистики**.

4.1.8.2.6. Организация входных и выходных портов

Также как и для источника сообщений, для маршрутизатора нужно создать выходы, через которые отправлять сообщения, и входы, по которым получать сообщения. Создайте эти входы и выходы.

1. Перетащите элемент **Скруглённый прямоугольник**.

2. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `536`, **Y:** 340, **Ширина:** 83, **Высота:** `110`.

3. Из палитры **Основная** перетащите восемь элементов **Порт**. Разместите их как на рис. 4.11. В полях **Имя:** предложенные системой имена замените согласно рис. 4.11. Установите флажки **Отображать имя**.

4. *Обратите также внимание на то, чтобы у элементов* **Скруглённый прямоугольник** *и* **Порт** *был установлен флажок* **На верхнем уровне**. *У остальных элементов сегмента* **Маршрутизатор** *этот флажок должен быть сброшенным.*

5. Соедините выходы элементов buferNapr1… buferNapr4 с соответствующими портами вых1… вых4, а порты вх1… вх4 — с входом элемента blokKontrol_1 (Блок контроля 1).

4.1.8.2.7. Имитатор отказов вычислительного комплекса

Принято, что вычислительный комплекс может выходить из строя и отказывать в обработке сообщений. Можно было бы построить имитатор отказов так, что генератор вырабатывает заявки-отказы в количестве, определяемом длительностью времени моделирования. Однако мы сделаем так, что генератор вырабатывает одну заявку, а затем этот процесс повторяется через интервалы времени, равные наработке до очередного отказа.

1. Перетащите элемент **Прямоугольник**.
2. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `160`, **Y:** 224, **Ширина:** 340, **Высота:** `140`.
3. Перетащите объект **source** и два объекта **delay**. Разместите и соедините их как на рис. 4.11.
4. Выделите **source** и установите значения его свойств:

Прибывают согласно `Интенсивности`

Интенсивность прибытия `1`

Количество заявок, прибывающих за один раз `1`

Ограниченное количество прибытий `1`

Максимальное количество прибытий `1`

5. Выделите первый объект **delay** и установите значения его свойств:

Имя: `розыгрыш_инт_до_отказа`

Задержка задаётся `Определённое время`

Время задержки `exponential(1/timeOtkBK)`

Вместимость `1`

Действия При выходе

```
hold.setBlocked(true);
   if (computer.size()!=0) {
      computer.remove((Message)computer.get(0));
         kolPoterBK ++;}
```

6. Выделите второй объект **delay** и установите значения его свойств:

Имя: `имитация_восст_ВК`

Задержка задаётся `Определённое время`

Время задержки `exponential(1/timeVosstBK)`

Вместимость `1`

Действия При выходе

```
hold.setBlocked(false)
```

4.1.9. Сегмент Канал

Данный сегмент предназначен для имитации передачи сообщений по каналам связи.

Для его реализации в AnyLogic используется имитационная модель направления связи (глава 2), которое состоит из основного и резервного каналов.

4.1.9.1. Исходные данные

1. Откройте объект **Канал**. Перейдите на область просмотра **viewData**.
2. Перетащите элемент **Скруглённый прямоугольник**.
3. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `10`, **Y:** 752, **Ширина:** 377, **Высота:** `118`.
4. Перетащите элементы **Параметр** и **Переменная**, разместите и дайте им имена согласно рис. 4.16.
5. Типы и значения по умолчанию установите согласно табл. 4 8.

Таблица 4.8

Имя	Тип	Значение по умолчанию
skorPeredKan	double	5000
skorPeredKanR	double	5000
timeOtkKan	double	360
timeVosstKan	double	3.2
timeBklResK	double	0.1
всего_потеряно_сообщ	int	0

4.1.9.2. Событийная часть сегмента Каналы

1. Перейдите на область просмотра **облКаналы**.
2. Перетащите элемент **Прямоугольник**.
3. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `186`, **Y:** 50, **Ширина:** 200, **Высота:** `135`.
4. Перетащите объект **selectOutput**, разместите как на рис. 4.17.
5. Оставьте предложенные системой имена и флажок **Отображать имя**.
6. В поле **Тип заявки:** `Agent` замените `Message`.

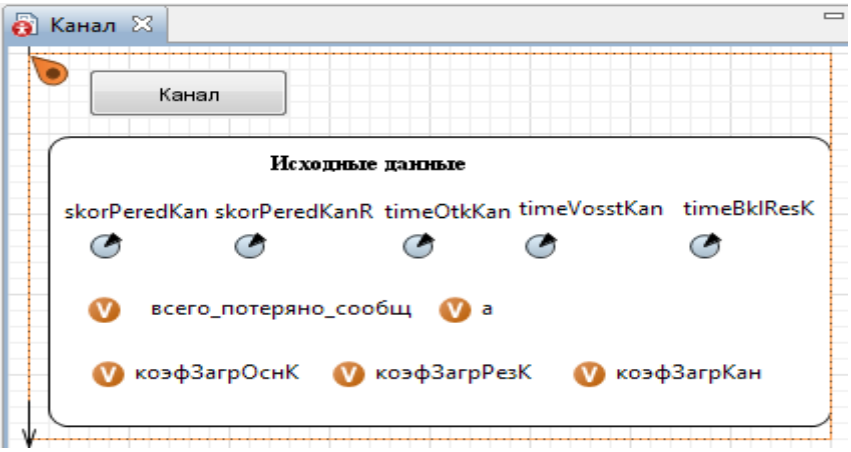


Рис. 4.16. Исходные данные сегмента **Канал**

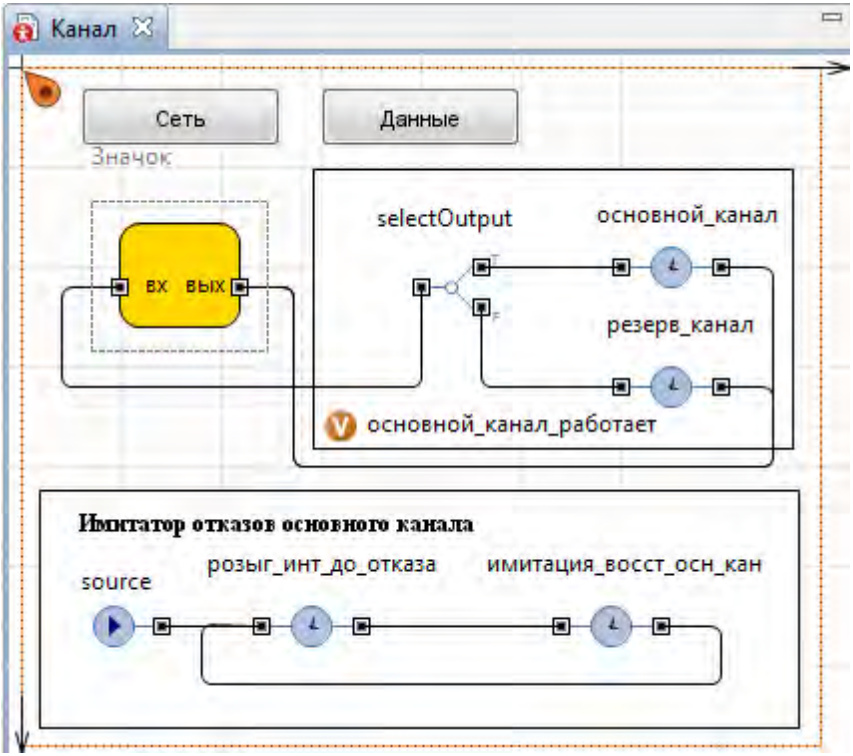


Рис. 4.17. Объекты событийной части сегмента **Канал**

7. **Выход true выбирается:** При выполнении условия.
8. **Условие:** основной_канал_работает.
9. **При выходе (true):**

```
entity.timePered=entity.dlina/skorPeredKan;
```

10. В поле **Вместимость** введите 1.
11. **При выходе (false):**

```
if (a==0)
entity.timePered=entity.dlina/skorPeredKanR;
if (a==1)
{entity.timePered=entity.dlina/skorPeredKanR +
timeBklResK;
a=0;}
a=1;
```

12. Установите флажок **Включить сбор статистики**.
13. Скопируйте объект **delay**. Вставьте его один раз. При этом изменится имя на основной_канал, а остальные свойства останутся неизменными.
14. Замените имя основной_канал на резерв_канал.
15. **Действия При выходе**

```
коэфЗагр-
РезК=резерв_канал.statsUtilization.mean();
коэфЗагрКан=коэфЗагрОснК+коэфЗагрРезК;
```

16. Соедините согласно рис. 4.17 объекты **selectOutput** и **delay**.

4.1.9.3. Организация входного и выходного портов

1. Перетащите элемент **Скруглённый прямоугольник**.
2. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** 50, **Y:** 80, **Ширина:** 60, **Высота:** 52.
3. Из палитры **Основная** перетащите два элемента **Порт**. Разместите их как на рис. 4.17. В полях **Имя:** предложенные системой имена замените согласно рис. 4.17. Установите флажки **Отображать имя**.
4. *Обратите также внимание на то, чтобы у элементов* **Скруглённый прямоугольник** *и* **Порт** *был установлен флажок* **На верхнем уровне**. *У остальных элементов сегмента* **Канал** *этот флажок должен быть сброшенным.*

5. Соедините выходы элементов **основной_канал** и **резерв_канал** с портом `вых`, а порт `вх` — с входом элемента selectOutput.

4.1.9.4. Имитатор отказов каналов связи

Имитатор отказов основного канала связи построен аналогично имитатору отказов вычислительного комплекса.

1. Перетащите элемент **Прямоугольник**.
2. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `10`, **Y:** 212, **Ширина:** 380, **Высота:** `120`.
3. Перетащите объект **source** и два объекта **delay**. Разместите и соедините их как на рис. 4.17.
4. Выделите **source** и установите значения его свойств:

Отображать имя установите флажок

Прибывают согласно `Интенсивности`

Интенсивность прибытия `1`

Количество заявок, прибывающих за один раз `1`

Ограниченное количество прибытий установите флажок

Максимальное количество прибытий `1`

Действия При выходе

5. Выделите первый объект **delay** и установите значения его свойств:

Имя: `розыг_инт_до_отказа`

Отображать имя установите флажок

Задержка задаётся `Определённое время`

Время задержки `exponential(1/timeOtkKan)`

Вместимость `1`

Действия При выходе:

```
основной_канал_работает = false;
if (основной_канал.size()!=0){
  Message m = основной_канал.get(0);
  основной_канал.stopDelayForAll();
  всего_потеряно_сообщ ++;}
```

6. Выделите второй объект **delay** и установите значения его свойств:

Имя `имитация_восст_осн_кан`

Отображать имя установите флажок

Задержка задаётся `Определённое время`

Время задержки `exponential(1/timeVosstkKan)`
Вместимость `1`
Действия При выходе:

```
основной_канал_работает = true;
```

4.1.10. Построение модели сети связи

Все необходимое для построения модели сети из активных объектов **Абонент**, **Канал** и **Маршрутизатор** нами создано. Приступим к построению модели сети.

1. Перейдите на Main к области просмотра **облСеть**.
2. Из окна **Проекты** перетащите объект **абонент1** и поместите как на рис. 4.18.
3. Объект **абонент1** имитирует абонента 1. В свойствах **абонент1** записан код для расчёта коэффициентов пропускной способности абонентов 2…6 с абонентом 1. Следовательно, **абонент2** должен иметь код для расчёта коэффициентов пропускной способности абонентов 1, 3…6 с абонентом 2 и т.д.
4. Создайте типы агентов **Абонент2**…**Абонент6**.
5. Откройте **Абонент1**. Скопируйте все объекты.
6. Вставьте скопированные объекты на объекты **Абонент2** … **Абонент6**.
7. Внесите правки в коды согласно табл. 4.9.
8. Из окна **Проекты** перетащите объекты **абонент2** … **абонент6**.
9. Из окна **Проекты** перетащите объект **канал** и поместите вверху (см. рис. 4.18). В поле **Имя:** добавьте к предложенному имени 1.
10. Скопируйте объект **канал1**. Вставьте пять объектов. Разместите их как на рис. 4.18. Нам потребуются ещё несколько каналов связи, но мы их образуем позже после создания второго маршрутизатора.
11. Выходы объектов **абонент1**…**абонент6** соедините с соответствующими входами объектов **канал1**…**канал6**.
12. Из окна **Проекты** перетащите объект **маршрутизатор**. В поле **Имя:** установите `маршрут1`.
13. Соедините выходы первого и второго абонентов с вх1, выходы третьего и четвёртого — с вх2, пятого — с вх3, шестого — с вх4 объекта **маршрут1**.

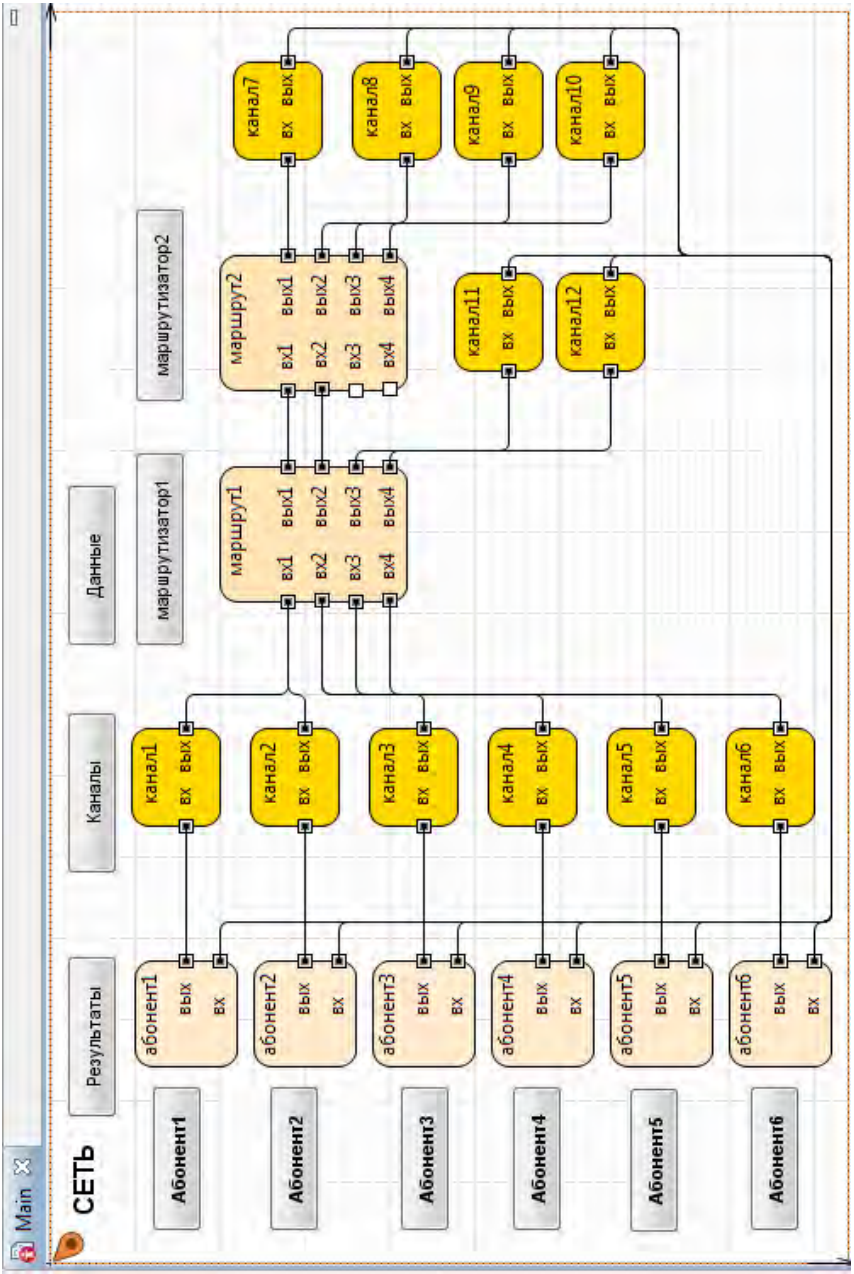


Рис. 4.18. Элементы модели функционирования сети связи

Таблица 4.9

Абонент2	
Свойство	selectOutput
Тип заявки: **Выход true выбирается** **Условие**	Message При выполнении условия entity.numAbPol == numAbonent
numAbonent	2
Свойство	selectOutput1
Тип заявки: **Использовать:**	Message Условия
Условие 1	entity.numAbPol==1
Действия **При выходе 1**	отпрАб1++; main.отпр21=отпрАб1;
Условие 2	entity.numAbPol==2
Условие 3	entity.numAbPol==3
Действия **При выходе 3**	отпрАб3++; main.отпр23=отпрАб3;
Условие 4	entity.numAbPol==4
Действия **При выходе 4**	отпрАб4++; main.отпр24=отпрАб4;
Свойство	selectOutput2
Тип заявки: **Использовать:**	Message Условия
Условие 1	entity.numAbPol==5
Действия **При выходе 1**	отпрАб5++; main.отпр25=отпрАб5;
Условие 2	entity.numAbPol==6
Действия **При выходе 2**	отпрАб6++; main.отпр26=отпрАб6;
Свойство	selectOutput5
Тип заявки: **Использовать:**	Message Условия
Условие 1	entity.numAbOtpr==1

Действия При выходе 1	`отАб1++;`
	`main.кПрСп12=отАб1/main.отпр12;`
	`main.КПрСп12.setText(main.кПрСп12,`
	`true);`

Условие 2	`entity.numAbOtpr==2`
Условие 3	`entity.numAbOtpr==3`

Действия При выходе 3	`отАб3++;`
	`main.кПрСп32=отАб3/main.отпр32;`
	`main.КПрСп32.setText(main.кПрСп32,`
	`true);`

Условие 4	`entity.numAbOtpr==4`

Действия При выходе 4	`отАб4++;`
	`main.кПрСп42=отАб4/main.отпр42;`
	`main.КПрСп42.setText(main.кПрСп42,`
	`true);`

Свойство	selectOutput6
Тип заявки: **Использовать:**	`Message` `Условия`
Условие 1	`entity.numAbOtpr==5`

Действия При выходе 1	`отАб5++;`
	`main.кПрСп52=отАб5/main.отпр52;`
	`main.КПрСп52.setText(main.кПрСп52,`
	`true);`

Условие 2	`entity.numAbOtpr==6`

Действия При выходе 2	`отАб6++;`
	`main.кПрСп62=отАб6/main.отпр62;`
	`main.КПрСп62.setText(main.кПрСп62,`
	`true);`

Абонент3	
Свойство	selectOutput

Тип заявки:
Выход true выбирается
Условие
	`Message`
	
	`При выполнении условия`
	`entity.numAbPol == numAbonent`

numAbonent	`3`

Продолжение табл. 4.9

Свойство	selectOutput1
Тип заявки: **Использовать:**	Message Условия
Условие 1	entity.numAbPol==1
Действия **При выходе 1**	отпрАб1++; main.отпр31=отпрАб1;
Условие 2	entity.numAbPol==2
Действия **При выходе 3**	отпрАб2++; main.отпр32=отпрАб2;
Условие 3	entity.numAbPol==3
Условие 4	entity.numAbOtpr==4
Действия **При выходе 4**	отпрАб4++; main.отпр34=отпрАб4;
Свойство	selectOutput2
Тип заявки: **Использовать:**	Message Условия
Условие 1	entity.numAbPol==5
Действия **При выходе 1**	отпрАб5++; main.отпр35=отпрАб5;
Условие 2	entity.numAbPol==6
Действия **При выходе 2**	отпрАб6++; main.отпр36=отпрАб6;
Свойство	selectOutput5
Тип заявки: **Использовать:**	Message Условия
Условие 1	entity.numAbOtpr==1
Свойство	selectOutput5
Действия **При выходе 1**	отАб1++; main.кПрСп13=отАб1/main.отпр13; main.КПрСп13.setText(main.кПрСп13, **true**);
Условие 2	entity.numAbOtpr==2

Действия При выходе 2	`отАб2++;`
	`main.кПрСп23=отАб2/main.отпр23;`
	`main.КПрСп23.setText(main.кПрСп23,`
	`true);`

Условие 3	`entity.numAbOtpr==3`
Условие 4	`entity.numAbOtpr==4`

Действия При выходе 4	`отАб4++;`
	`main.кПрСп43=отАб4/main.отпр43;`
	`main.КПрСп43.setText(main.кПрСп43,`
	`true);`

Свойство	selectOutput6
Тип заявки: **Использовать:**	`Message` `Условия`
Условие 1	`entity.numAbOtpr==5`

Действия При выходе 1	`отАб5++;`
	`main.кПрСп53=отАб5/main.отпр53;`
	`main.КПрСп53.setText(main.кПрСп53,`
	`true);`

Условие 2	`entity.numAbOtpr==6`

Действия При выходе 2	`отАб6++;`
	`main.кПрСп63=отАб6/main.отпр63;`
	`main.КПрСп63.setText(main.кПрСп63,`
	`true);`

Абонент4	
Свойство	selectOutput

Тип заявки:
Выход true выбирается
Условие
	`Message`
	
	`При выполнении условия`
	`entity.numAbPol == numAbonent`

numAbonent	`4`
Свойство	selectOutput1
Тип заявки: **Использовать:**	`Message` `Условия`
Условие 1	`entity.numAbPol==1`
Действия При выходе 1	`отпрАб1++;` `main.отпр41=отпрАб1;`

Условие 2	`entity.numAbPol==2`
Действия При выходе 3	`отпрАб2++;` `main.отпр42=отпрАб2;`
Условие 3	`entity.numAbPol==3`
Действия При выходе 3	`отпрАб3++;` `main.отпр43=отпрАб3;`
Условие 4	`entity.numAbOtpr==4`
Свойство	selectOutput2
Тип заявки: **Использовать:**	`Message` `Условия`
Условие 1	`entity.numAbPol==5`
Действия При выходе 1	`отпрАб5++;` `main.отпр45=отпрАб5;`
Условие 2	`entity.numAbPol==6`
Действия При выходе 2	`отпрАб6++;` `main.отпр46=отпрАб6;`
Свойство	selectOutput5
Тип заявки: **Использовать:**	`Message` `Условия`
Условие 1	`entity.numAbOtpr==1`

Действия При выходе 1	`отАб1++;`
	`main.кПрСп14=отАб1/main.`
	`отпр14;`
	`main.КПрСп14.setText(main.кПрСп14,`
	`true);`

Условие 2	`entity.numAbOtpr==2`

Действия При выходе 2	`отАб2++;`
	`main.кПрСп24=отАб2/main.отпр24;`
	`main.КПрСп24.setText(main.кПрСп24,`
	`true);`

Условие 3	`entity.numAbOtpr==3`

Действия При выходе 3	`отАб3++;`
	`main.кПрСп34=отАб3/main.отпр34;`
	`main.КПрСп34.setText(main.кПрСп34,`
	`true);`

Условие 4	`entity.numAbOtpr==4`
Свойство	selectOutput6
Тип заявки: **Использовать:**	`Message` `Условия`
Условие 1	`entity.numAbOtpr==5`

Действия
При выходе 1
	`отАб5++;`
	`main.кПрСп54=отАб5/main.отпр54;`
	`main.КПрСп54.setText(main.кПрСп54,`
	`true);`

Условие 2	`entity.numAbOtpr==6`

Действия
При выходе 2
	`отАб6++;`
	`main.кПрСп64=отАб6/main.отпр64;`
	`main.КПрСп64.setText(main.кПрСп64,`
	`true);`

Абонент5	
Свойство	selectOutput

Тип заявки:
Выход true выбирается
Условие
	`Message`
	
	`При выполнении условия`
	`entity.numAbPol == numAbonent`

numAbonent	5
Свойство	selectOutput1
Тип заявки: **Использовать:**	`Message` `Условия`
Условие 1	`entity.numAbPol==1`
Действия **При выходе 1**	`отпрАб1++;` `main.отпр51=отпрАб1;`
Условие 2	`entity.numAbPol==2`
Действия **При выходе2**	`отпрАб2++;` `main.отпр52=отпрАб2;`
Условие 3	`entity.numAbPol==3`
Действия **При выходе 3**	`отпрАб3++;` `main.отпр53=отпрАб3;`
Условие 4	`entity.numAbOtpr==4`

Действия При выходе 4	`отпрАб4++;` `main.отпр54=отпрАб4;`
Свойство	selectOutput2
Тип заявки: **Использовать:**	`Message` `Условия`
Условие 1	`entity.numAbPol==5`
Условие 2	`entity.numAbPol==6`
Действия При выходе 2	`отпрАб6++;` `main.отпр56=отпрАб6;`
Свойство	selectOutput5
Тип заявки: **Использовать:**	`Message` `Условия`
Свойство	selectOutput5
Условие 1	`entity.numAbOtpr==1`

Действия При выходе 1	`отАб1++;`
	`main.кПрСп15=отАб1/main.отпр15;`
	`main.КПрСп15.setText(main.кПрСп15,`
	`true);`

Условие 2	`entity.numAbOtpr==2`

Действия При выходе 2	`отАб2++;`
	`main.кПрСп25=отАб2/main.`
	`отпр25;`
	`main.КПрСп25.setText(main.кПрСп25,`
	`true);`

Условие 3	`entity.numAbOtpr==3`

Действия При выходе 3	`отАб3++;`
	`main.кПрСп35=отАб3/main.отпр35;`
	`main.КПрСп35.setText(main.кПрСп35,`
	`true);`

Условие 4	`entity.numAbOtpr==4`

Действия При выходе 4	`отАб4++;`
	`main.кПрСп45=отАб4/main.отпр45;`
	`main.КПрСп45.setText(main.кПрСп45,`
	`true);`

Продолжение табл. 4.9

Свойство	selectOutput6
Тип заявки: **Использовать:**	Message Условия
Условие 1	entity.numAbOtpr==5
Условие 2	entity.numAbOtpr==6
Действия **При выходе 2**	отАб6++; main.кПрСп65=отАб6/main.отпр65; main.КПрСп65.setText(main.кПрСп65, true);
Абонент6	
Свойство	selectOutput
Тип заявки: **Выход true выбирается** **Условие**	Message При выполнении условия entity.numAbPol == numAbonent
numAbonent	6
Свойство	selectOutput1
Тип заявки: **Использовать:**	Message Условия
Условие 1	entity.numAbPol==1
Действия **При выходе 1**	отпрАб1++; main.отпр61=отпрАб1;
Условие 2	entity.numAbPol==2
Действия **При выходе 2**	отпрАб2++; main.отпр62=отпрАб2;
Условие 3	entity.numAbPol==3
Действия **При выходе 3**	отпрАб3++; main.отпр63=отпрАб3;
Условие 4	entity.numAbPol==4
Действия **При выходе 4**	отпрАб4++; main.отпр64=отпрАб4;
Свойство	selectOutput2
Условие 1	entity.numAbPol==5

Окончание табл. 4.9

Действия При выходе 1	`отпрАб5++;` `main.отпр65=отпрАб5;`
Условие 1	`entity.numAbPol==6`
Свойство	selectOutput5
Тип заявки: **Использовать:**	`Message` `Условия`
Условие 1	`entity.numAbOtpr==1`

Действия При выходе 1	`отАб1++;`
	`main.кПрСп16=отАб1/main.отпр16;`
	`main.КПрСп16.setText(main.кПрСп16,`
	`true);`

Условие 2	`entity.numAbOtpr==2`

Действия При выходе 2	`отАб2++;`
	`main.кПрСп26=отАб2/main.отпр26;`
	`main.КПрСп26.setText(main.кПрСп26,`
	`true);`

Условие 3	`entity.numAbOtpr==3`

Действия При выходе 3	`отАб3++;`
	`main.кПрСп35=отАб3/main.отпр35;`
	`main.КПрСп35.setText(main.кПрСп35,`
	`true);`

Условие 4	`entity.numAbOtpr==4`

Действия При выходе 4	`отАб3++;`
	`main.кПрСп36=отАб3/main.отпр36;`
	`main.КПрСп36.setText(main.кПрСп36,`
	`true);`

Свойство	selectOutput6
Тип заявки: **Использовать:**	`Message` `Условия`
Условие 1	`entity.numAbOtpr==5`

Действия При выходе 1	`отАб5++;`
	`main.кПрСп56=отАб5/main.отпр56;`
	`main.КПрСп56.setText(main.кПрСп56,`
	`true);`

Условие 2	`entity.numAbOtpr==6`

Для того чтобы связь была между всеми абонентами и они могли бы обмениваться сообщениями, нам потребуется ещё один маршрутизатор. Но мы не можем использовать второй экземпляр этого же типа абонента, так как программно он настроен именно на наш вариант организации связи.

1. Создайте ещё тип агента **Маршрутизатор1**.
2. Откройте объект **Маршрутизатор**.
3. Выделите все объекты и скопируйте их.
4. Вставьте на **Маршрутизатор1** скопированные объекты.
5. Выделите элемент exit и в поле **Действия При выходе** замените имеющийся там код следующим кодом:

```
int i;
  i=entity.numAbPol;
   {
       switch (i) {
         case 1:if (emkBuferNapr1-
tekEmkNapr1>=entity.dlina)
{enter1.take(entity);
                break;}
              else {enter.take(entity);
                break;}
         case 2:if (emkBuferNapr2-
tekEmkNapr2>=entity.dlina) {
                enter2.take(entity);
                break;}
              else {enter.take(entity);
                break;}
         case 3:if (emkBuferNapr3-
tekEmkNapr3>=entity.dlina) {
                enter3.take(entity);
                break;}
              else {enter.take(entity);
                  break;}
         case 4:if (emkBuferNapr4-
tekEmkNapr4>=entity.dlina) {
                enter4.take(entity);
                break;}
              else {enter.take(entity);
                break;}
       }
     }
```

6. Теперь **Маршрутизатор1** настроен так, что сообщения от абонентов 1…4 будут направляться на его выходы 1…4.

7. Из окна **Проекты** перетащите элемент **маршрутизатор1**. В поле **Имя:** установите `маршрут2`.

8. Соедините вых1 **маршрут1** с вх1 **маршрут2**, а вых2 с вх2.

9. Скопируйте элемент **канал1**. Вставьте шесть элементов. Разместите их как на рис. 4.18.

10. Соедините вых3 **маршрут1** с вх **канал11**, вых4 — с вх **канал12.**

11. Соедините вых1…вых4 **маршрут2** с вх **канал7**…**канал10** соответственно.

12. Соедините вых **канал7**…**канал12** с входами абонент1…абонент6 соответственно.

Построение модели сети связи завершено. Но надо ещё организовать переключение между областями просмотра.

4.1.11. Переключение между областями просмотра

Переключение между областями просмотра организуем так, чтобы можно было из сети переходить к любому абоненту, каналу, маршрутизатору и обратно. В каждом активном объекте — к данным и обратно к событийной части объекта или в сеть.

Для переключения используем элемент **button** из палитры **Элементы управления**.

1. Перетащите элемент **button** (см. рис. 4.18).
2. На странице **Основные** панели **Свойства** укажите:

Метка: `Абонент1`

Действие: `абонент1.облАбонент1.navigateTo()`

3. Скопируйте кнопку **Абонент1**. Вставьте пять раз. Последовательно откройте и внесите соответствующие правки в полях **Метка:** и **Действие:**, например:

Метка: `Абонент2`

Действие: `абонент2.облАбонент2.navigateTo()`

4. Перетащите элемент **button**. Укажите свойства:

Метка: `Результаты`

Действие: `viewData.navigateTo()`

5. Перетащите элемент **button**. Укажите свойства:

Метка: `маршрутизатор1`

Действие: `маршрут1.облМарш1.navigateTo()`

6. Скопируйте кнопку **маршрутизатор1**. Её свойства:

Метка: `маршрутизатор2`

Действие: `маршрут2.облМарш2.navigateTo()`

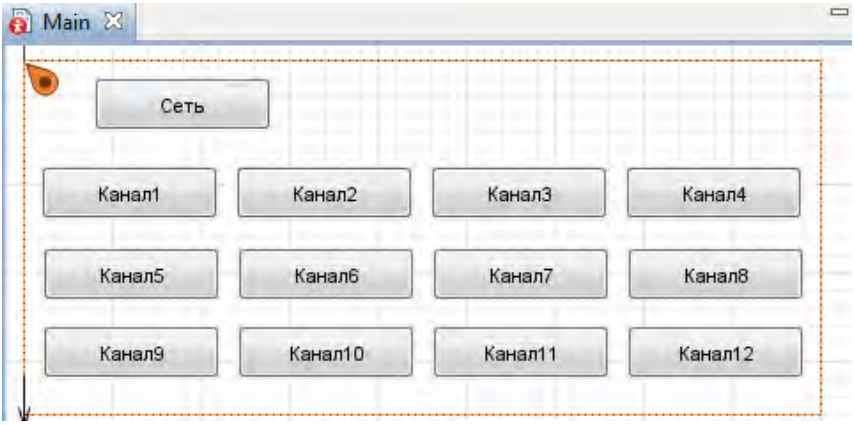


Рис. 4.19. Область просмотра **облКан**

Так как каналов 12, то давайте создадим ещё одну область просмотра **облКан**, и на ней разместим 12 кнопок (рис. 4.19).

1. Перетащите из палитры **Презентация** элемент **Область просмотра**. На странице **Основные** в поле **Имя:** введите `облКан`.
2. Перейдите на страницу **Местоположение и размер** панели **Свойства**. Введите в поля **X:** `0`, **Y:** `1780`, **Ширина:** `450`, **Высота:** `200`.
3. Перетащите элемент **button**.
4. На странице **Основные** панели **Свойства** укажите:

Метка: `Канал1`

Действие: `канал1.облКан.navigateTo()`

5. Скопируйте кнопку **Канал1**. Вставьте одиннадцать раз. Последовательно откройте и внесите соответствующие правки в полях **Метка:** и **Действие:**. Например:

Метка: `Канал2`

Действие: `канал2.облКан.navigateTo()`

6. Перетащите элемент **button**.
7. На странице **Основные** панели **Свойства** укажите:

Метка: `Сеть`

Действие: `облСеть.navigateTo()`

8. Перейдите на область просмотра **облСеть**. Перетащите элемент **button**.
9. На странице **Основные** панели **Свойства** укажите:

Метка: `Каналы`

Действие: `облКан.navigateTo()`

Нам осталось добавить элементы для переключения между областями просмотра на активных объектах и возвращения на корневой объект Main на область просмотра **облСеть**.

Последовательно переходите от объекта к объекту, добавляя на них нужное число элементов **button** и устанавливая значения свойств согласно табл. 4.10. Размещение этих элементов было уже показано на рис. 4.3… 4.6, 4.8, 4.10, 4.11, 4.16… 4.18.

Таблица 4.10

Объект	Область просмотра	Свойства	Значение
Абонент	облАбонент	**Метка:**	Сеть
		Действие:	main.облСеть.navigateTo()
	облАбонент	**Метка:**	Исходные данные
		Действие:	viewData.navigateTo()
	viewData	**Метка:**	Сеть
		Действие:	main.облСеть.navigateTo()
	viewData	**Метка:**	Абонент
		Действие:	облАбонент.navigateTo()
Канал	облКан	**Метка:**	Сеть
		Действие:	main.облСеть.navigateTo()
	облКан	**Метка:**	Данные
		Действие:	viewData.navigateTo()
	viewData	**Метка:**	Канал
		Действие:	облКан.navigateTo()
Маршрутизатор	облМарш	**Метка:**	Сеть
		Действие:	main.облСеть.navigateTo()
	облМарш	**Метка:**	Данные
		Действие:	viewData.navigateTo()
	viewData	**Метка:**	Сеть
		Действие:	main.облСеть.navigateTo()
	viewData	**Метка:**	Маршрутизатор
		Действие:	облМарш.navigateTo()
Маршрутизатор1	облМарш	**Метка:**	Сеть
		Действие:	main.облСеть.navigateTo()
	облМарш	**Метка:**	Данные
		Действие:	viewData.navigateTo()
	viewData	**Метка:**	Сеть
		Действие:	main.облСеть.navigateTo()
	viewData	**Метка:**	Маршрутизатор
		Действие:	облМарш.navigateTo()

4.1.12. Запуск и отладка модели

Прежде чем запустить модель:

1. В окне **Проекты** выделите **Сеть_связи**.
2. На странице **Основные** в поле **Единицы модельного времени:** установите `секунды`.
3. В окне **Проекты** выделите **Simulation: Main**.
4. На странице **Модельное время** установите:

Режим выполнения: `Виртуальное время (максимальная скорость)`

Остановить: `В заданное время`.

5. В поле **Конечное время:** введите `3600000`. Время моделирования увеличено в 1000 раз по числу прогонов модели.
6. На странице **Случайность** установите **Фиксированное начальное число (воспроизводимые «прогоны»)**.
7. В поле **Начальное число:** введите `897`.
8. Запустите модель. Если появятся ошибки, исправьте их.

При правильном построении модели вы получите результаты, показанные на рис. 4.20.

Среди них показатели качества обслуживания сети связи: коэффициент пропускной способности 0,815 и среднее время передачи одного сообщения 6,050. Коэффициент пропускной способности, например, абонент 2 — абонент 3 равен 0,816. Обратите внимание на существенную разницу между минимальным и максимальным временами передачи сообщения. Она объясняется принятым экспоненциальным законом распределения времени передачи сообщений: максимальное значение может отличаться от среднего значения в восемь раз.

Количество отправленных и полученных сообщений всего и по категориям рассчитано за один прогон модели, то есть за 3600 сек. Для расчёта, как вы помните, был введён параметр kolProg=1000 и в 1000 раз было увеличено модельное время. Всего отправлено сообщений 600,559, получено всего абонентами — 490,035. Если разделить число полученных сообщений на число отправленных, то и будет получен коэффициент пропускной способности сети.

Точно такие же результаты моделирования, коэффициенты пропускной способности, выводятся и по каждому абоненту сети.

Перейдите в область просмотра **Сеть**, щёлкнув кнопку **Сеть**. Затем, щёлкните, например, кнопку **Абонент2**. Вы увидите результаты моделирования, показанные на рис. 4.21.

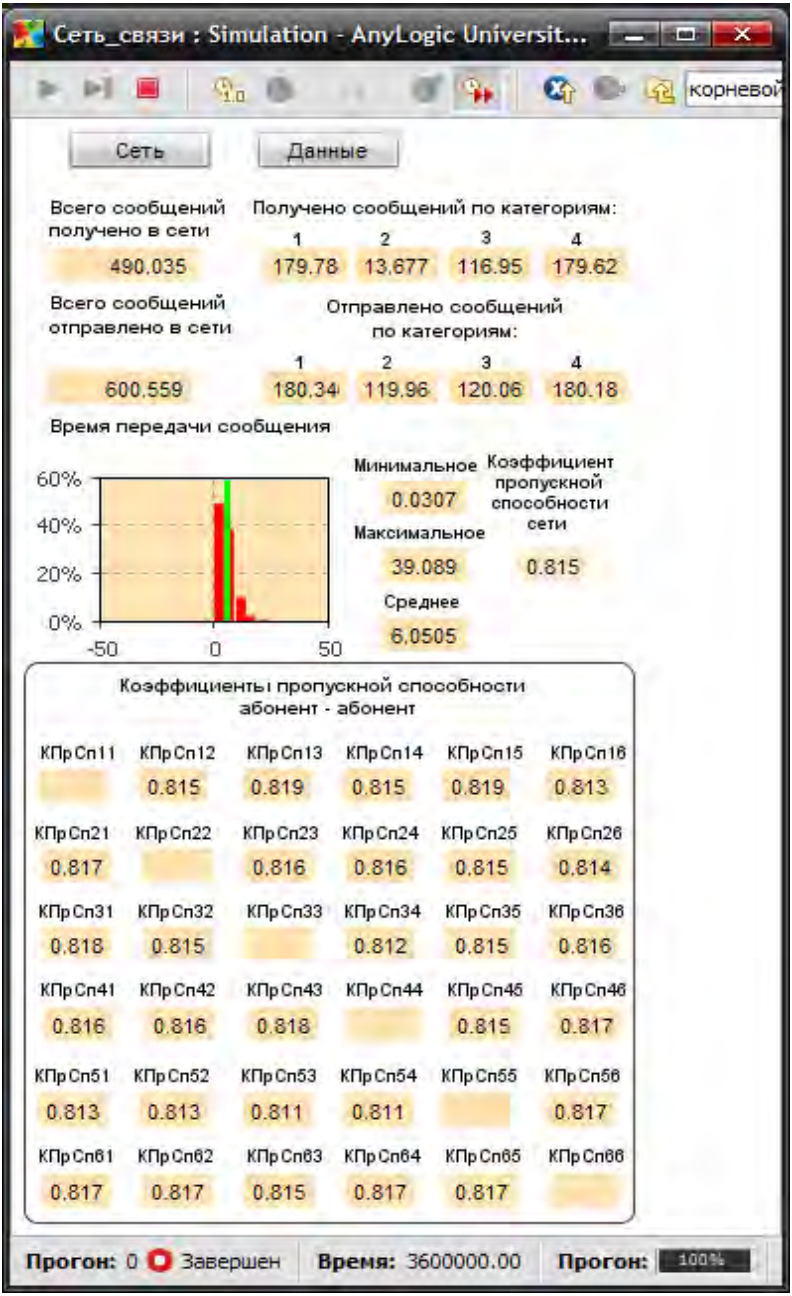


Рис. 4.20. Результаты моделирования сети связи

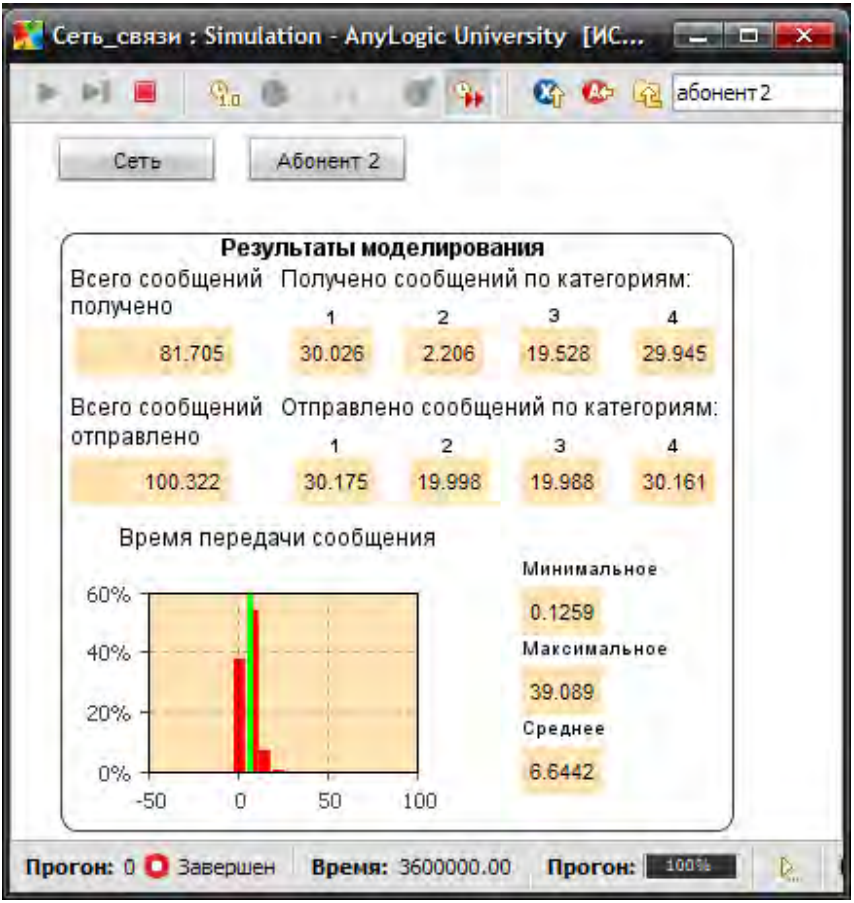


Рис. 4.21. Результаты моделирования по абоненту 2 сети связи

4.2. Интерпретация результатов моделирования

Всего выполнено 6 экспериментов. Изменение параметров сети связи проводилось также, как и в предыдущих моделях. То есть в каждом следующем эксперименте параметры, изменённые в предыдущем эксперименте, остаются такими же, если не указано их изменение. В шестом эксперименте кроме указанных параметров были также изменены средние интервалы поступления сообщений от всех абонентов (timeAbonent) с 30 до 20.

В Anylogic в поле **Конечное время:** было введено 3600000.0. Время моделирования увеличено в 1000 раз по числу прогонов модели. В GPSS указывается модельное время 3600 и 1000 прогонов модели.

Результаты экспериментов сведены в табл. 4.11. Видно, что результаты моделирования отличаются незначительно. Коэффициенты пропускной способности (Δ_{11} ... (Δ_{61}) отличаются на 0…0,001, а среднее время передачи одного сообщения (Δ_{12} … Δ_{62}) — на 0,071…1,064 с. Коэффициенты загрузки вычислительных комплексов также отличаются на 0…0,022, а коэффициенты загрузки основных каналов — на 0,000…0,001.

В четвёртом и пятом экспериментах в связи с уменьшением ёмкостей входных буферов ВК1 и ВК2 сообщения второй категории не передаются, так как их средняя длина из всех категорий максимальная.

Сравнительная оценка позволяет сделать вывод об адекватности результатов моделирования, полученных при использовании для построения моделей одной и той же системы инструментальных средств GPSS World и AnyLogic.

Машинное время выполнения модели в GPSS World составляет 22 … 24 с, а в AnyLogic примерно 15 мин.

Таблица 4.11

Показатели функционирования сети связи

Показатели	GPSS World	AnyLogic6	AnyLogic7
1) Согласно постановке задачи			
Коэффициент пропускной способности сети связи	0,816	0,815	0,817
	$\Delta_{11}=\lvert 0{,}816-0{,}816 \rvert = 0{,}001$		
Время передачи одного сообщения	6,127	6,052	6,056
	$\Delta_{12}=\lvert 6{,}127-6{,}056 \rvert =0{,}071$		
Коэффициент загрузки ВК1	0,255	0,255	0,255
Коэффициент загрузки ВК2	0,170	0,170	0,170
Коэффициент загрузки основных каналов	0,042	0,042	0,042
Количество полученных / отправленных сообщений	489,930/ 600,263	490,195/ 601,092	489,917/ 599,907
2) emkBuferVx =…= emkBuferVx = 70000			
Коэффициент пропускной способности сети связи	0,741	0,741	0,741
	$\Delta_{21}=\lvert 0{,}741-0{,}741 \rvert =0{,}000$		
Время передачи одного сообщения	6,127	5,867	5,880
	$\Delta_{12}=\lvert 6{,}127-5{,}880 \rvert =0{,}247$		
Коэффициент загрузки ВК1	0,255	0,255	0,255
Коэффициент загрузки ВК2	0,170	0,170	0,170
Коэффициент загрузки основных каналов	0,043	0,043	0,042
Количество полученных / отправленных сообщений	444,728/ 600,263	444,863/ 600,347	444,999/ 599,929
3) emkBuferVx =…= emkBuferVx = 60000			
Коэффициент пропускной способности сети связи	0,599	0,599	0,601
	$\Delta_{31}=\lvert 0{,}599-0{,}601 \rvert =0{,}002$		
Время передачи одного сообщения	6,132	5,660	5,658
	$\Delta_{32}=\lvert 6{,}132-5{,}658 \rvert =0{,}474$		
Коэффициент загрузки ВК1	0,255	0,256	0,277
Коэффициент загрузки ВК2	0,170	0,170	0,170
Коэффициент загрузки основных каналов	0,043	0,043	0,042
Количество полученных / отправленных сообщений	359,640/ 600,584	360,251/ 601,104	360,799/ 599,829

Окончание табл. 4.11

Показатели функционирования сети связи

Показатели	GPSS World	AnyLogic6	AnyLogic7
4) emkBufer1 = 4000000 (маршрутизаторов)			
Коэффициент пропускной способности сети связи	0,599	0,600	0,601
	Δ_{41}=	0,599-0,601	=0,002
Время передачи одного сообщения	6,132	5,650	5,658
	Δ_{42}=	6,132-5,658	=0,478
Коэффициент загрузки ВК1	0,255	0,251	0,255
Коэффициент загрузки ВК2	0,170	0,169	0,170
Коэффициент загрузки основных каналов	0,043	0,042	0,043
Количество полученных / отправленных сообщений	359,640/ 600,584	359,651/ 599,302	360,799/ 599,829
5) emkBufer1 = 3000000 (маршрутизаторов)			
Коэффициент пропускной способности сети связи	0,599	0,598	0,602
	Δ_{51}=	0,599-0,602	=0,003
Время передачи одного сообщения	6,132	5,648	5,658
	Δ_{52}=	6,132-5,658	=0,474
Коэффициент загрузки ВК1	0,255	0,256	0,255
Коэффициент загрузки ВК2	0,170	0,17	0,170
Коэффициент загрузки основных каналов	0,043	0,044	0,043
Количество полученных / отправленных сообщений	359,640/ 600,584	360,114/ 601,495	360,799/ 599,829
6) emkBuferVx =…= emkBuferVx = 100000, emkBufer1 = 6000000			
Коэффициент пропускной способности сети связи	0,996	0,996	0,997
	Δ_{61}=	0,996-0,997	=0,001
Время передачи одного сообщения	6,155	7,237	7,219
	Δ_{62}=	6,155-7,219	=1,064
Коэффициент загрузки ВК1	0,362	0,382	0,381
Коэффициент загрузки ВК2	0,238	0,254	0,254
Коэффициент загрузки основных каналов	0,064	0,064	0,064
Количество полученных / отправленных сообщений	848,296/ 851,388	896,072/ 899,333	896,792/ 898,748

ГЛАВА 5. МОДЕЛЬ ФУНКЦИОНИРОВАНИЯ СИСТЕМЫ СВЯЗИ

5.1. Модель в AnyLogic

5.1.1. 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. Интервалы времени T21, T22, …, T2n2 между отказами СС, находящимися на дежурстве, случайные. В случае выхода из строя СС заменяют резервным, причем либо сразу, либо по мере появления исправного СС. Тем временем, вышедшее из строя СС ремонтируют, после чего содержат в качестве резервного или направляют его на дежурство. 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.

Прибыль от СС, находящихся на дежурстве, составляет S1 денежных единиц в час. Почасовой убыток при отсутствии на дежурстве одного СС — S2, …, S2n2 денежных единиц в час. Оплата мастера за ремонт неисправного СС — S31, S32, …, S3n2 денежных единиц в час соответственно.

Затраты на содержание одного резервного СС составляют S4 денежных единиц в час.

5.1.2. Задание на исследование

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

Исследовать влияние на ожидаемую прибыль различного количества резервных СС и мастеров.

Определить абсолютные величины и относительные коэффициенты ожидаемой прибыли.

Сделать выводы об использовании СС, мастеров и необходимых мерах по совершенствованию системы предоставления услуг связи.

5.1.3. Формализованное описание модели

Уясним задачу на разработку модели, предварительно представив структуру системы предоставления услуг связи (рис. 5.1) как систему СМО.

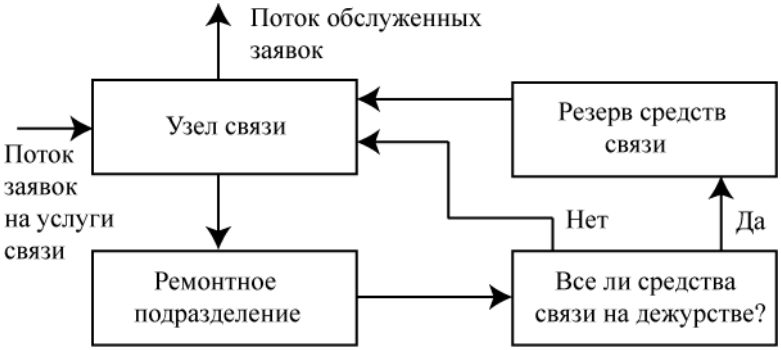


Рис. 5.1. Система предоставления услуг связи как СМО

Система предоставления услуг связи (далее система связи) представляет собой многофазную многоканальную систему массового обслуживания замкнутого типа с отказами.

Таким образом, модель системы связи должна состоять из следующих сегментов (рис. 5.2):

имитации постановки на дежурство СС;

имитации дежурства СС;

имитации функционирования ремонтного подразделения;

вывода результатов моделирования.

Рис. 5.2. Концептуальная схема модели системы связи

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

`tipCC` — код типа СС;

`timeOtkaz` — среднее время между отказами СС;

`timeMeanRem` — среднее время ремонта одного СС;

`nach` — время начала ремонта в ремонтном подразделении;

`nach1` — время начала дежурства.

Возьмём, например, $n_2 = 5$. Код типа СС в виде чисел 1, 2, 3, 4, 5 определяется в самом начале моделирования и остаётся 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 `timeOtkaz` 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, `timeOtkaz = 1/timeOtkaz1` используются для обращения к генератору `exponential(timeOtkaz)`.

Параметры `nach1` и `nach` изменяются при каждом поступлении СС на дежурство и в ремонтное подразделение соответственно. Они используются при расчётах дохода от дежурства и затрат на ремонт неисправного СС. В них заносится время начала дежурства и начала 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, а также по завершении моделирования рассчитываются:

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= KCC1*doxDegCC1*ВремяРабСист
```

где ВремяРабСист — время работы моделируемой системы.

Фактический доход DoxDegCC1 от дежурства СС первого типа составит:

```
DoxDegCC1+=(time()-entity.nach1)*
main.doxDegCC1;
```

где (time()-entity.nach1) — время нахождения СС первого типа на дежурстве.

При отсутствии на дежурстве СС первого типа убыток составит:

```
UbitokCC1=(1-degCC1.statsUtilization.mean())*
main.ubitokCC1*ВремяРабСист*KCC1;
```

где (1-degCC1.statsUtilization.mean()) — средний коэффициент отсутствия СС первого типа на дежурстве за всё время моделирования.

Затраты на ремонт неисправных СС и содержание резервных СС первого типа составят соответственно:

```
ZatrRemCC1+=(time()-entity.nach)*stoimRemCC1;
ZatrResCC1= KCCP1*zatrResCC1* ВремяРабСист
```

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.

5.1.4. 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**.
2. Дайте имя новой модели. В поле **Имя модели** введите `Система_связи`. Выберите каталог для сохранения файлов модели.
3. Щёлкните кнопку **Готово**.

5.1.4.1. Ввод исходных данных

Исходные данные разделим и разместим там, где они, по нашему мнению, нужны и их удобно использовать.

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

1. В **Палитре** выделите **Презентация**. Перетащите элемент **Область просмотра**. Перейдите на панель **Свойства**.
2. В поле **Имя:** введите `Исходные_данные_ПД`.
3. Задайте расположение области просмотра относительно ее якоря **Выравнивать по:** `Верхн. левому углу`.
4. Выберите режим масштабирования из выпадающего списка **Масштабирование:** `Подогнать под окно`.
5. Перейдите на страницу **Местоположение и размер**. Введите в поля **X:** `0`, **Y:** `500`, **Ширина:** `450`, **Высота:** `340`.
6. Перейдите на страницу **Местоположение и размер**. Введите в поля **X:** `0`, **Y:** `500`, **Ширина:** `450`, **Высота:** `340`.

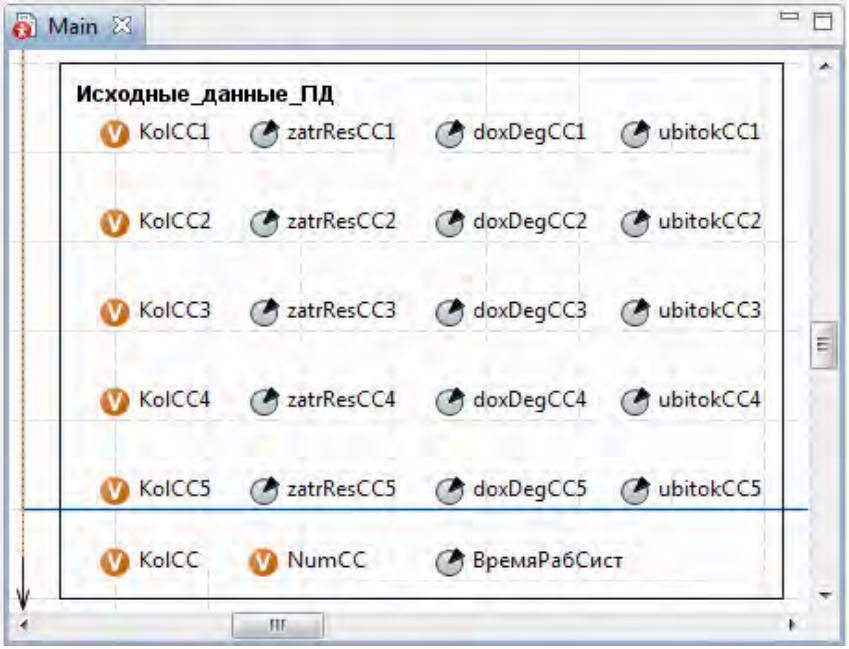


Рис. 5.3. Размещение элементов **Параметр** и **Переменная**

7. Перетащите элемент **Прямоугольник** на элемент **Область просмотра**.

8. Перейдите на страницу **Местоположение и размер**. Введите в поля **X:** 20, **Y:** 520, **Ширина:** 390, **Высота:** 300.

9. Перетащите элемент **text** и на странице **Текст** в поле **text** введите Исходные_данные_ПД (здесь ПД — постановка на дежурство).

10. В **Палитре** выделите **Основная**. Перетащите элементы **Параметр** и **Переменная** на элемент с именем Исходные_данные_ПД и разместите их так, как показано на рис. 5.3.

11. Значения свойств установите согласно табл. 5.1.

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.

Таблица 5.1

Свойства элементов на `Исходные_данные_ПД`

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	
ubitokCC1	double	32	
ubitokCC2	double	34.2	
ubitokCC3	double	37	
ubitokCC4	double	31	
ubitokCC5	double	32.5	

5.1.4.2. Имитация поступления средств связи

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

1. Из палитры **Презентация** перетащите элемент **Область просмотра**.
2. На панели **Свойства** в поле **Имя:** введите `постановка`.
3. Задайте **Выравнивать по:** `Верхн. левому углу`.
4. Выберите режим масштабирования из выпадающего списка **Масштабирование:** `Подогнать под окно`.
5. Перейдите на страницу **Местоположение и размер**. Введите в поля **X:** `0`, **Y:** `0`, **Ширина:** `700`, **Высота:** `350`.
6. Перетащите элемент **Скруглённый прямоугольник**. Оставьте имя, предложенное системой. В нём мы разместим объекты сегмента **Постановка на дежурство**.

7. Перейдите на страницу **Местоположение и размер**. Введите в поля **X:** `38`, **Y:** `62`, **Ширина:** 642, **Высота:** `268`.

8. На **Область просмотра** мы уже перетащили **Скругленный прямоугольник**. На нём мы будем размещать, как отмечалось ранее, объекты сегмента **Постановка на дежурство**.

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

10. Перетащите элемент **Прямоугольник** на **Область просмотра**. На странице **Местоположение и размер** введите в поля **X:** `50`, **Y:** 100, **Ширина:** `170`, **Высота:** `140`.

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

12. Перетащите объект **source** на **Прямоугольник**. Для записи и хранения параметров СС в поля заявок необходимо создать нестандартный тип агента. Создайте тип агента `ComFacility`.

13. В панели **Проекты** Щёлкните правой кнопкой мыши элемент модели верхнего уровня дерева и выберите **Создать/Java класс**.

14. Появится диалоговое окно **Новый Java класс**. В поле **Имя:** введите имя нового класса `ComFacility`.

15. В поле **Базовый класс:** выберите из выпадающего списка `Entity` в качестве базового класса. Щёлкните **Далее**.

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

```
tipCC          int;
timeMeanRem    double;
nach           double;
nach1          double;
timeOtkaz      double;
```

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

18. Щёлкните кнопку **Готово**. Закройте редактор кода.

19. Щёлкните правой кнопкой мыши в панели **Проект** только что созданный Java класс и в контекстном меню выберите **Преобразовать Java класс в тип агента**.

20. Появится окно с автоматически созданными параметрами нестандартного типа заявок `ComFacility`. Закройте его.

21. Выделите объект **source**.

22. На панели **Свойства** уберите флажок **Отображать имя**.

23. В полях **Тип заявки:** и **Новая заявка:** `Agent` замените `ComFacility`. Установите:

Прибывают согласно `Интенсивности`.

Интенсивность прибытия `1`

Ограниченное количество прибытий установите флажок

Максимальное количество прибытий `1`

На странице **Действия** в поле **При выходе:** введите Java код:

```
KolCC1=degurstvo.KCC1+degurstvo.KCCP1;
degurstvo.DoxMaxCC1=
round((degurstvo.KCC1*doxDegCC1)*ВремяРабСист*100);
degurstvo.DoxMaxCC1=degurstvo.DoxMaxCC1/100;
degurstvo.ZatrResCC1=
round((degurstvo.KCCP1*zatrResCC1)*ВремяРабСист*100);
degurstvo.ZatrResCC1=degurstvo.ZatrResCC1/100;
KolCC2=degurstvo.KCC2+degurstvo.KCCP2;
degurstvo.DoxMaxCC2=
round((degurstvo.KCC2*doxDegCC2)*ВремяРабСист*100);
degurstvo.DoxMaxCC2=degurstvo.DoxMaxCC2/100;
degurstvo.ZatrResCC2=
round((degurstvo.KCCP2*zatrResCC2)*ВремяРабСист*100);
degurstvo.ZatrResCC2=degurstvo.ZatrResCC2/100;
KolCC3=degurstvo.KCC3+degurstvo.KCCP3;
degurstvo.DoxMaxCC3=
round((degurstvo.KCC3*doxDegCC3)*ВремяРабСист*100);
degurstvo.DoxMaxCC3=degurstvo.DoxMaxCC3/100;
degurstvo.ZatrResCC3=
round((degurstvo.KCCP3*zatrResCC3)*ВремяРабСист*100);
degurstvo.ZatrResCC3=degurstvo.ZatrResCC3/100;
KolCC4=degurstvo.KCC4+degurstvo.KCCP4;
degurstvo.DoxMaxCC4=
round((degurstvo.KCC4*doxDegCC4)*ВремяРабСист*100);
degurstvo.DoxMaxCC4=degurstvo.DoxMaxCC4/100;
degurstvo.ZatrResCC4=
round((degurstvo.KCCP4*zatrResCC4)*ВремяРабСист*100);
degurstvo.ZatrResCC4=degurstvo.ZatrResCC4/100;
KolCC5=degurstvo.KCC5+degurstvo.KCCP5;
degurstvo.DoxMaxCC5=
round((degurstvo.KCC5*doxDegCC5)*ВремяРабСист*100);
degurstvo.DoxMaxCC5=degurstvo.DoxMaxCC5/100;
KolCC=KolCC1+KolCC2+KolCC3+KolCC4+KolCC5;
```

```
degurstvo.ZatrResCC5=
round((degurstvo.KCCP5*zatrResCC5)*ВремяРабСист*100);
degurstvo.ZatrResCC5=degurstvo.ZatrResCC5/100;
degurstvo.SumDoxMax=degurstvo.DoxMaxCC1+
degurstvo.DoxMaxCC2+degurstvo.DoxMaxCC3+
degurstvo.DoxMaxCC4+degurstvo.DoxMaxCC5;
degurstvo.SumZatrRes=degurstvo.ZatrResCC1+
degurstvo.ZatrResCC2+degurstvo.ZatrResCC3+
degurstvo.ZatrResCC4+degurstvo.ZatrResCC5;
```

Введённым кодом определяется количество СС всех типов, включая и резервные средства связи. Эти данные необходимы в последующем в объекте **split**. Количества СС как исходные данные будут размещены на агенте **Degyrstvo**, который мы создадим позже, поэтому в коде используется доступ к ним в виде, например, degyrstvo.KCC1.

Кодом рассчитываются также максимальный доход от каждого типа СС и суммарный максимальный доход, а также затраты на содержание резервных СС по типам и суммарные затраты на содержание резервных СС всех типов.

Такой подход к расчётам максимального дохода и затрат на содержание резервных СС принят для экономии машинного времени, поскольку эти показатели зависят только от продолжительности времени моделирования (`ВремяРабСист`), дохода от дежурства одного СС и затрат на содержание одного резервного СС. То есть данные для их расчёта не носят стохастический характер. Поэтому указанные показатели целесообразно рассчитать здесь, а затем использовать в конце моделирования при обработке накопленных статистических данных.

Для вывода результатов моделирования с двумя знаками после запятой использовался метод round(). Предварительно результат умножался на 100, а потом делился на эту же величину. Например:

```
degyrstvo.DoxMaxCC1=
round((degyrstvo.KCC1*doxDegCC1)*ВремяРабСист*100);
degyrstvo.DoxMaxCC1=degyrstvo.DoxMaxCC1/100;
```

24. Добавьте объекты **split** и **sink** (рис. 5.4).

25. Выделите объект **split**. Установите свойства как на рис. 5.5. Так как копии не унаследуют свойств от оригинала, поэтому на странице **Действия** в поле **При выходе копии:** введите Java код:

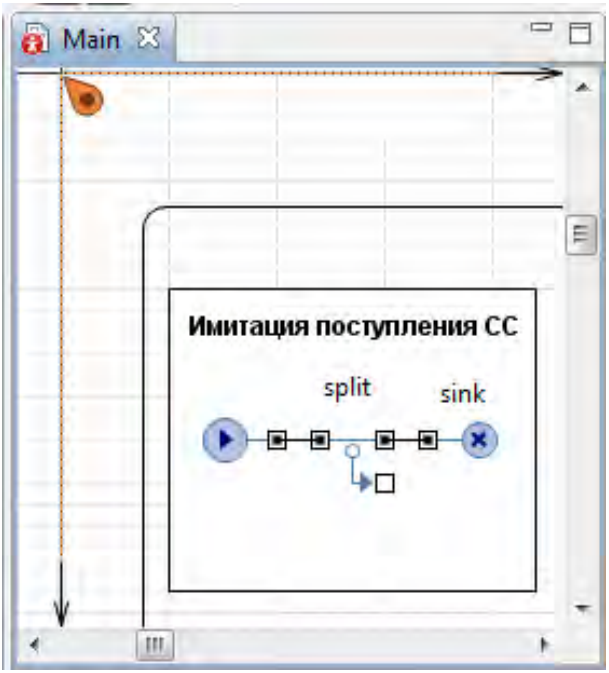


Рис. 5.4. Добавлены объекты **split** и **sink**

```
NumCC++;
if (NumCC <= KolCC)
  entity.tipCC = 5;
if (NumCC <= (KolCC1+KolCC2+KolCC3+KolCC4))
   entity.tipCC = 4;
if (NumCC <= (KolCC1+KolCC2+KolCC3))
   entity.tipCC = 3;
if (NumCC <= (KolCC1+KolCC2))
   entity.tipCC = 2;
if (NumCC <= KolCC1)
    entity.tipCC = 1;
```

Переменная `NumCC` предназначена для последовательного счёта СС всех типов. Она используется для определения принадлежности СС к соответствующему типу и занесения кода этого типа СС в поле `entity.tipCC`, например, `entity.tipCC = 5`, который необходим для нормального функционирования модели, то есть отличия СС по типам.

1. Объект **sink** уничтожает заявку-оригинал.

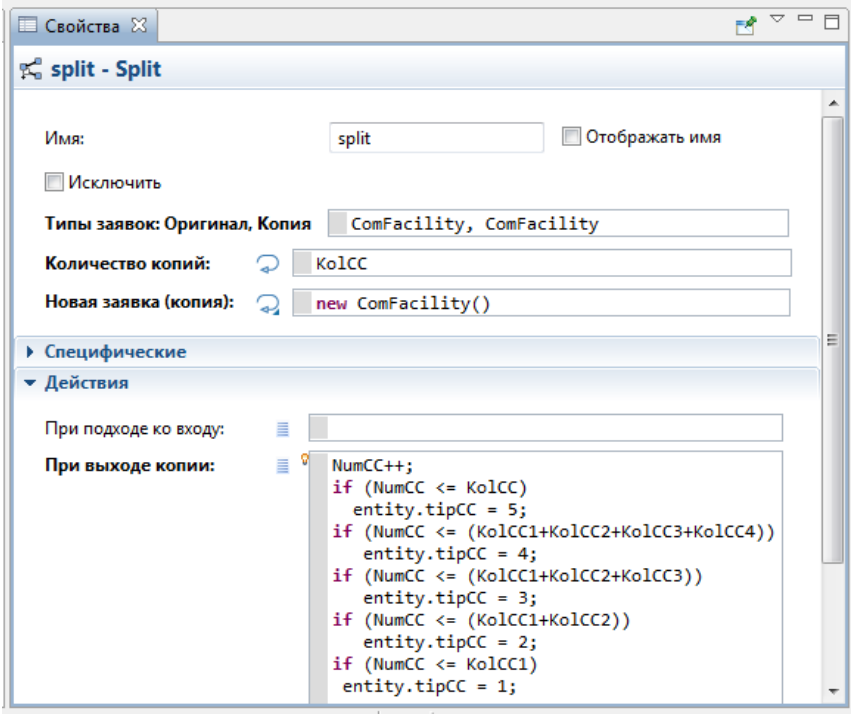


Рис. 5.5. Объект split с установленными свойствами

5.1.4.3. 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 объектами **selectOutput** и одним объектом **queue** (рис. 5.6). Возможна реализация этого блока и одним объектом **selectOutput5** совместно также с объектом **queue**.

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

1. Перетащите четыре объекта **selectOutput** и один объект **queue** из **Библиотеки моделирования процессов** на диаграмму агента Main. Соедините их так, как показано на рис. 5.6.

2. Установите на панели **Свойства** свойства объектов **selectOutput** согласно табл. 5.2 (при использовании объекта **selectOutput5** условия разделения СС по типам останутся такими же).

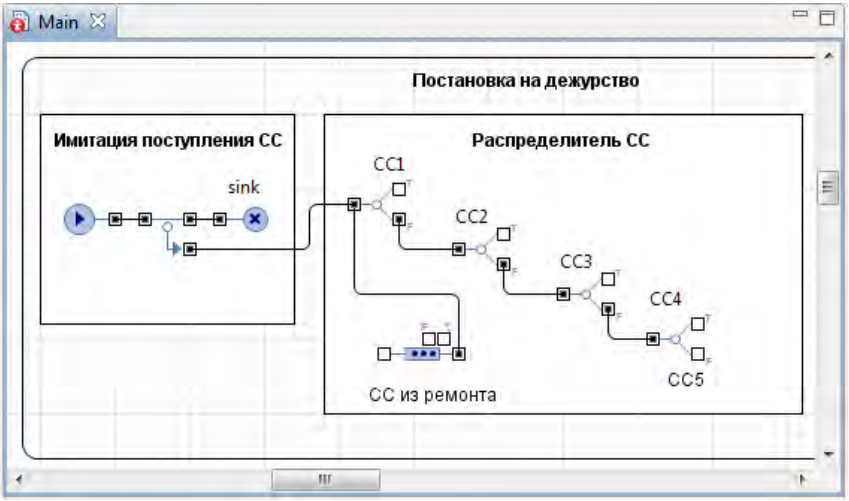


Рис. 5.6. Добавлены объекты **SelectOutput** и **queue**

Таблица 5.2

Имя	Отобра-жать имя	Тип заявки:	Выход true выбирается	Условие

CC1
CC2
CC3
CC4
	Установите флажки	`ComFacility`
		`ComFacility`
		`ComFacility`
		`ComFacility`
			При выпол-нении условия	`entity.tipCC==1`
				`entity.tipCC==2`
				`entity.tipCC==3`
				`entity.tipCC==4`

3. СС пятого типа будут направлены на выход false элемента CC4, поэтому пятый объект **selectOutput** не нужен.

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

5. Оставьте **Вместимость** `100` и на странице **Специфические** установите флажок **Включить сбор статистики**.

6. Перетащите из палитры **Презентация** три элемента **text** и в соответствующих полях **Текст:** введите текст, как на рис. 5.6.

5.1.4.4. Создание нового активного объекта

На рис. 5.7 показан в окончательном виде сегмент **Постановка на дежурство** после добавления блока **На дежурство**. Для добавления этого блока, который должен выполнять «связь» между сегментом **Постановка на дежурство** и сегментом **Имитация дежурства**, создадим новый тип агента. Экземпляром этого типа агента и будет нужный нам объект — блок **На дежурство**.

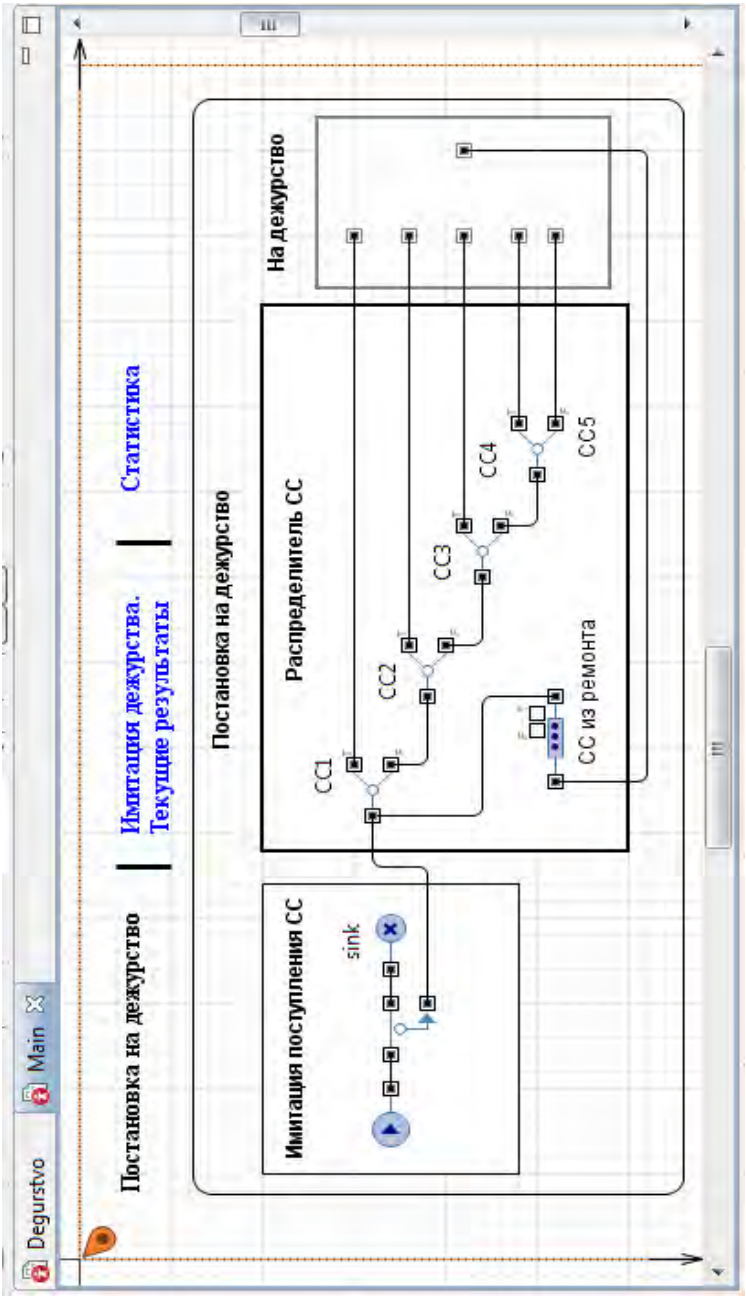


Рис. 5.7. Сегмент **Постановка на дежурство**

7. На панели **Проекты** Щёлкните правой кнопкой мыши Main, с которым вы работаете в данный момент, и выберите из контекстного меню **Создать/Тип агента**.

8. Откроется окно **Шаг 1. Создание нового типа агента**.

9. Оставьте **Не использовать шаблоны типов агентов**.

10. Задайте в поле **Имя нового агента:** `Degurstvo`.

11. Если нужно, в поле **Описание**: введите описание сущности, моделируемой этим типом агента.

12. Щёлкните кнопку **Готово**.

Создайте область просмотра на типе агента **Degurstvo** для размещения элементов сегмента **Имитация дежурства**.

13. В **Палитре** выделите **Презентация**. Перетащите элемент **Область просмотра**.

14. Перейдите на панель **Свойства**.

15. В поле **Имя:** введите `дежурство`.

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

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

1. Перейдите на страницу **Местоположение и размер**. Введите в поля **X:** 40, **Y:** 0, **Ширина:** 730, **Высота:** 420.

5.1.4.5. Создание экземпляра нового типа агента

Созданный новый тип агента **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 элемент **Прямоугольник**. На панели **Свойства** оставьте флажки **Исключить** и **Отображается на верхнем уровне**.

2. Перейдите на страницу **Местоположение и размер**. Введите в поля **X:** 110, **Y:** 130, **Ширина:** 100, **Высота:** 180.

3. Перетащите шесть элементов **Порт** ▣ из палитры **Основная** и разместите так, как показано на рис. 5.8. Для всех элементов **Порт** оставьте только флажок **Отображается на верхнем уровне**.

4. Возвратитесь на диаграмму класса Main.

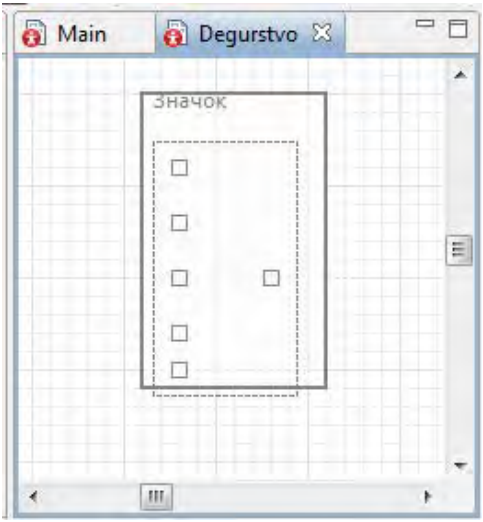


Рис. 5.8. Добавлены на **Degyrstvo** шесть портов

5. Из панели **Проекты** перетащите экземпляр типа агента **Degyrstvo**, и соедините так, как на рис. 5.7. При этом следует иметь в виду, что положение портов на экземпляре типа агента **Degyrstvo** изменить нельзя. Это можно сделать лишь на самом типе агента.

5.1.5. Сегмент Имитация дежурства

5.1.5.1. Ввод исходных данных

Организуйте ввод исходных данных для сегмента **Имитация дежурства**.

Замечание. В размещаемых далее элементах и объектах сегмента сбрасывайте флажок **Отображается на верхнем уровне**.

1. Перетащите элемент **Область просмотра**. Перейдите на панель **Свойства**. В поле **Имя:** введите `исходные_данные_Д`.
2. Задайте **Выравнивать по:** `Верхн. левому углу`.
3. Выберите масштабирование `Подогнать под окно`.
4. На странице **Местоположение и размер** введите в поля **X:** `40`, **Y:** `800`, **Ширина:** `650`, **Высота:** `240`.
5. Перетащите элемент **Прямоугольник**. На нём мы разместим элементы для ввода исходных данных.
6. Оставьте имя, предложенное системой.

7. Перейдите на страницу **Местоположение и размер**. Введите в поля **X:** 50, **Y:** 840, **Ширина:** 620, **Высота:** 190.

8. Перетащите элемент **text** и на панели **Свойства** в поле **Текст:** введите Исходные_данные_Д (здесь Д — дежурство).

9. Перетащите, используя копирование, элементы **Параметр** на элемент с именем Исходные_данные_Д и разместите их так, как показано на рис. 5.9. При копировании целесообразно указать имя первого элемента и его тип, например, KCC1, int. Значения по умолчанию нужно установить потом для каждого элемента своё.

10. На панели **Свойства** каждого элемента **Параметр** установите свойства согласно табл. 5.3.

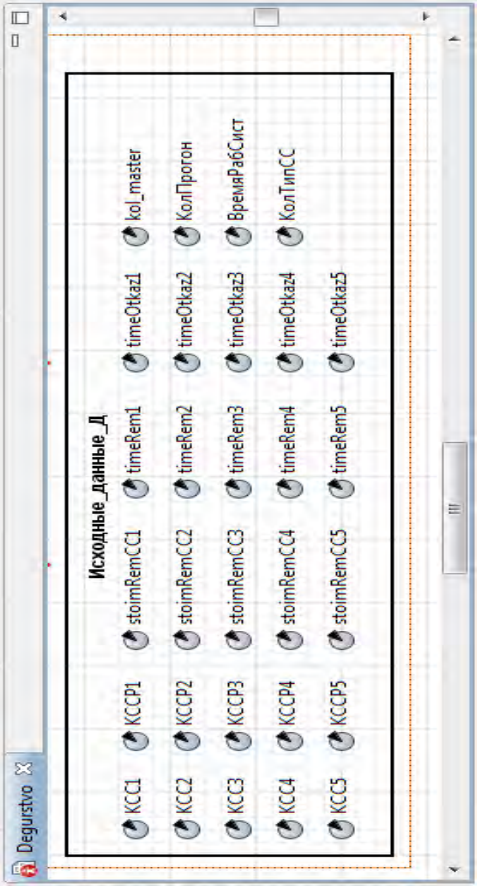


Рис. 5.9. Размещение элементов **Параметр** для ввода данных

Таблица 5.3

Свойства элементов **Параметр** на `Исходные_данные_Д`

Имя	Тип	Значение по умолчанию	Отображать имя
КСС1	int	55	Установить флажок во всех элементах
КСС2	int	100	
КСС3	int	60	
КСС4	int	45	
КСС5	int	60	
КССР1	int	2	
КССР2	int	4	
КССР3	int	4	
КССР4	int	3	
КССР5	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	
КолПрогон	double	1000	
ВремяРабСист	double	1000	

5.1.5.2. Вывод результатов моделирования

Здесь выводятся все результаты моделирования (рис. 5.10). Однако с целью экономии машинного времени, выводятся они по разному. Рассчитанные ранее максимальные доходы от дежурства СС и затраты на содержание резервных СС не выводятся в ходе моделирования. Выводятся только текущие доходы от дежурства СС и текущие затраты на ремонт неисправных СС. Все обработанные результаты выводятся по окончании моделирования. Для организации вывода используется способ **Событие** (см. п. 5.1.7).

Рис. 5.10. Элементы **Переменная**
для вывода результатов моделирования

1. Перетащите элемент **Область просмотра**. Перейдите на панель **Свойства**. В поле **Имя:** введите `текущие_результаты`.

2. Задайте **Выравнивать по:** `Верхн. левому углу`.

3. Выберите масштабирование `Подогнать под окно`.

4. На странице **Местоположение и размер** введите в поля **X:** 40, **Y:** 430, **Ширина:** 730, **Высота:** 350.

5. Перетащите элемент **Прямоугольник**.

6. Перейдите на панель **Свойства**. Введите в поля **X:** 54, **Y:** 466, **Ширина:** 710, **Высота:** 300.

7. Результаты разбиты на две группы: затраты и доходы. Перетащите два элемент **text** и на панели **Свойства** в поле **Текст:** введите `Затраты` и `Доходы` соответственно.

8. В **Палитре** выделите **Основная**. Перетащите, используя копирование, элементы **Переменная**. Разместите их, дайте имена, как показано на рис. 5.10. У всех переменных установите флажки **Отображать имя** и тип double.

5.1.5.3. Событийная часть сегмента Имитация дежурства

Реализация событийной части сегмента показана на рис. 5.11.

1. Перетащите элемент **Скруглённый прямоугольник**. На нём мы разместим все элементы сегмента **Имитация дежурства**.

2. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** 60, **Y:** 50, **Ширина:** 600, **Высота:** 360.

3. Перетащите элемент **Прямоугольник**. На нём мы разместим элементы, непосредственно имитирующие дежурство СС.

4. На странице **Местоположение и размер** введите в поля **X:** 250, **Y:** 60, **Ширина:** 170, **Высота:** 330.

5. Перетащите ещё один элемент **Прямоугольник** для размещения элементов, имитирующих ремонтное подразделение СС.

6. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** 440, **Y:** 60, **Ширина:** 190, **Высота:** 330.

7. Установите, используя копирование, на тип агента **Degyrstvo** последовательно: пять объектов queue, пять объектов delay, один объект queue, один объект delay и соедините их так, как показано на рис. 5.11. При копировании используйте свойства каждого объекта, приведенные в табл. 5.4. Кроме них также свойства, указанные в п. 9.

8. Перетащите три элемента **text** и введите названия в соответствующие поля **Текст:** согласно рис. 5.11.

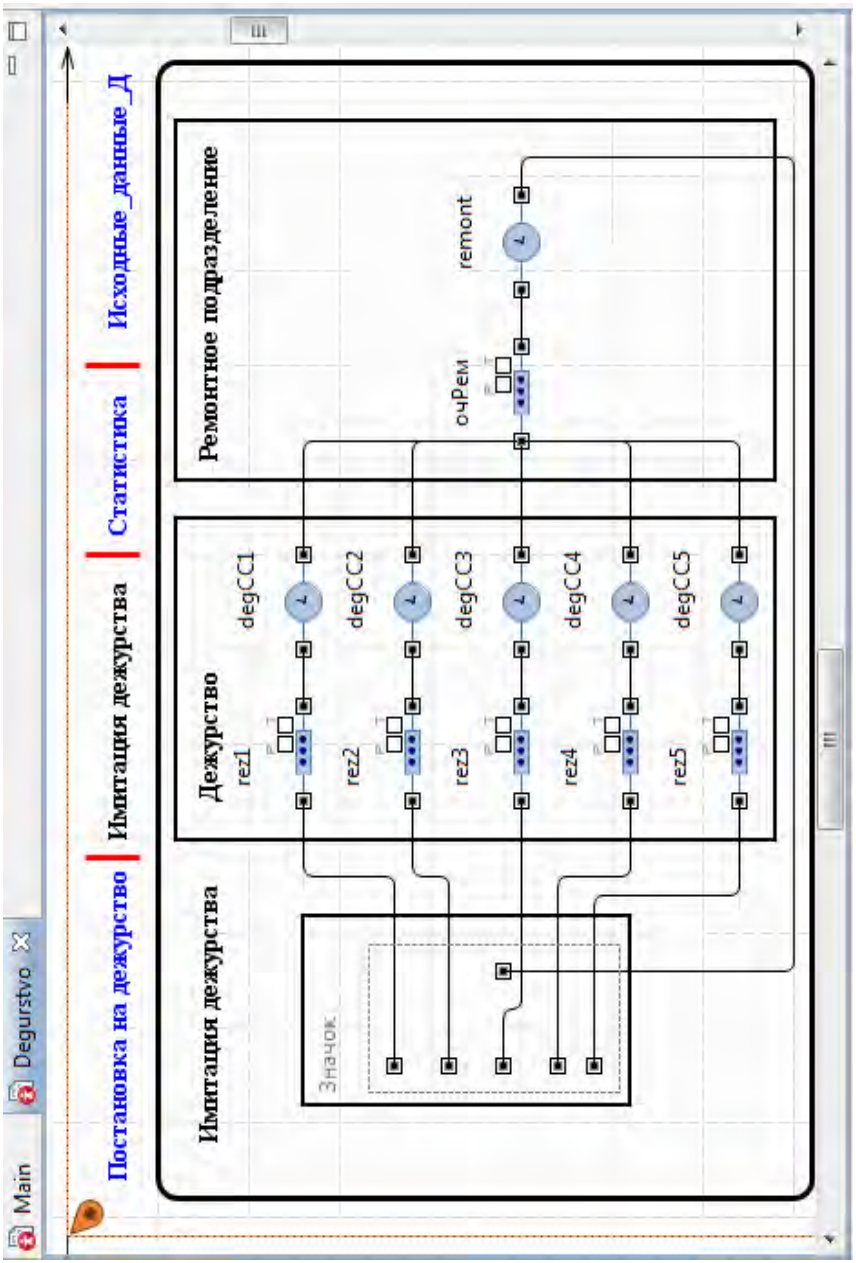


Рис. 5.11. Сегмент **Имитация дежурства**

Таблица 5.4

Объекты сегмента **Имитация дежурства** и их свойства

queue		
Имя	Вместимость	**Действия При выходе**
резерв1	КССР1	`entity.timeOtkaz=1/timeOtkaz1`
резерв2	КССР2	`entity.timeOtkaz=1/timeOtkaz2`
резерв3	КССР3	`entity.timeOtkaz=1/timeOtkaz3`
резерв4	КССР4	`entity.timeOtkaz=1/timeOtkaz4`
резерв5	КССР5	`entity.timeOtkaz=1/timeOtkaz5`
очРем	Максимальная	

delay		
Имя	Время задержки	Вместимость
degCC1	`exponential(entity.timeOtkaz)`	КСС1
degCC2	`exponential(entity.timeOtkaz)`	КСС2
degCC3	`exponential(entity.timeOtkaz)`	КСС3
degCC4	`exponential(entity.timeOtkaz)`	КСС4
degCC5	`exponential(entity.timeOtkaz)`	КСС5
remont	`exponential(entity.timeMeanRem)`	3

Имя	**Действия При выходе**
degCC1	`DoxDegCC1+=(time()-entity.nach1)*main.doxDegCC1;` `entity.timeMeanRem=1/timeRem1;`
degCC2	`DoxDegCC2+=(time()-entity.nach1)*main.doxDegCC2;` `entity.timeMeanRem=1/timeRem2;`
degCC3	`DoxDegCC3+=(time()-entity.nach1)*main.doxDegCC3;` `entity.timeMeanRem=1/timeRem3;`
degCC4	`DoxDegCC4+=(time()-entity.nach1)*main.doxDegCC4;` `entity.timeMeanRem=1/timeRem4;`
degCC5	`DoxDegCC5+=(time()-entity.nach1)*main.doxDegCC5;` `entity.timeMeanRem=1/timeRem5;`

9. Кроме свойств, указанных в табл. 5.4, нужно также:

во всех объектах в поле **Тип заявки:** `Agent` заменить `ComFacility;`

для всех объектов поставить флажки **Включить сбор статистики**;

для всех объектов **delay** degCC1 … degCC5 установить:

Действие при входе `entity.nach1=time();`

для объекта **delay** с именем remont также ввести Java коды в следующие свойства:

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

Действия При выходе

```
if (entity.tipCC == 1)
   {ZatrRemCC1+=((time()-entity.nach)*stoimRemCC1);
    SumZatrRem+=((time()-entity.nach)*stoimRemCC1);}
if (entity.tipCC == 2)
   {ZatrRemCC2+=((time()-entity.nach)*stoimRemCC2);
   SumZatrRem+=((time()-entity.nach)*stoimRemCC2);}
if (entity.tipCC == 3)
   {ZatrRemCC3+=((time()-entity.nach)*stoimRemCC3);
   SumZatrRem+=((time()-entity.nach)*stoimRemCC3);}
if (entity.tipCC == 4)
   {ZatrRemCC4+=(time()-entity.nach)*stoimRemCC4;
  SumZatrRem+=((time()-entity.nach)*stoimRemCC4);}
if (entity.tipCC == 5)
   {ZatrRemCC5+=((time()-entity.nach)*stoimRemCC5);
   SumZatrRem+=((time()-entity.nach)*stoimRemCC5);}
```

5.1.6. Сегмент Статистика

Результаты моделирования выводятся в сегменте **Имитация дежурства**. Тем не менее, организуем вывод результатов моделирования, можно сказать, в более презентабельном виде. Для этого создадим сегмент **Статистика** (рис. 5.12).

1. Создайте область просмотра для размещения элементов сегмента **Статистика**.
2. Перетащите элемент **Область просмотра**.
3. Перейдите на панель **Свойства**.
4. В поле **Имя:** введите `статистика`.
5. Задайте **Выравнивать по:** `Верхн. левому углу`.
6. Выберите режим масштабирования из выпадающего списка **Масштабирование:** `Подогнать под окно`.
7. На странице **Местоположение и размер** введите в поля:

X: `0`, **Y:** `1076`, **Ширина:** `960`, **Высота:** `630`.

8. Перетащите элемент **Прямоугольник**.
9. На странице **Местоположение и размер** введите в поля:

X: `20`, **Y:** `1116`, **Ширина:** `930`, **Высота:** `580`.

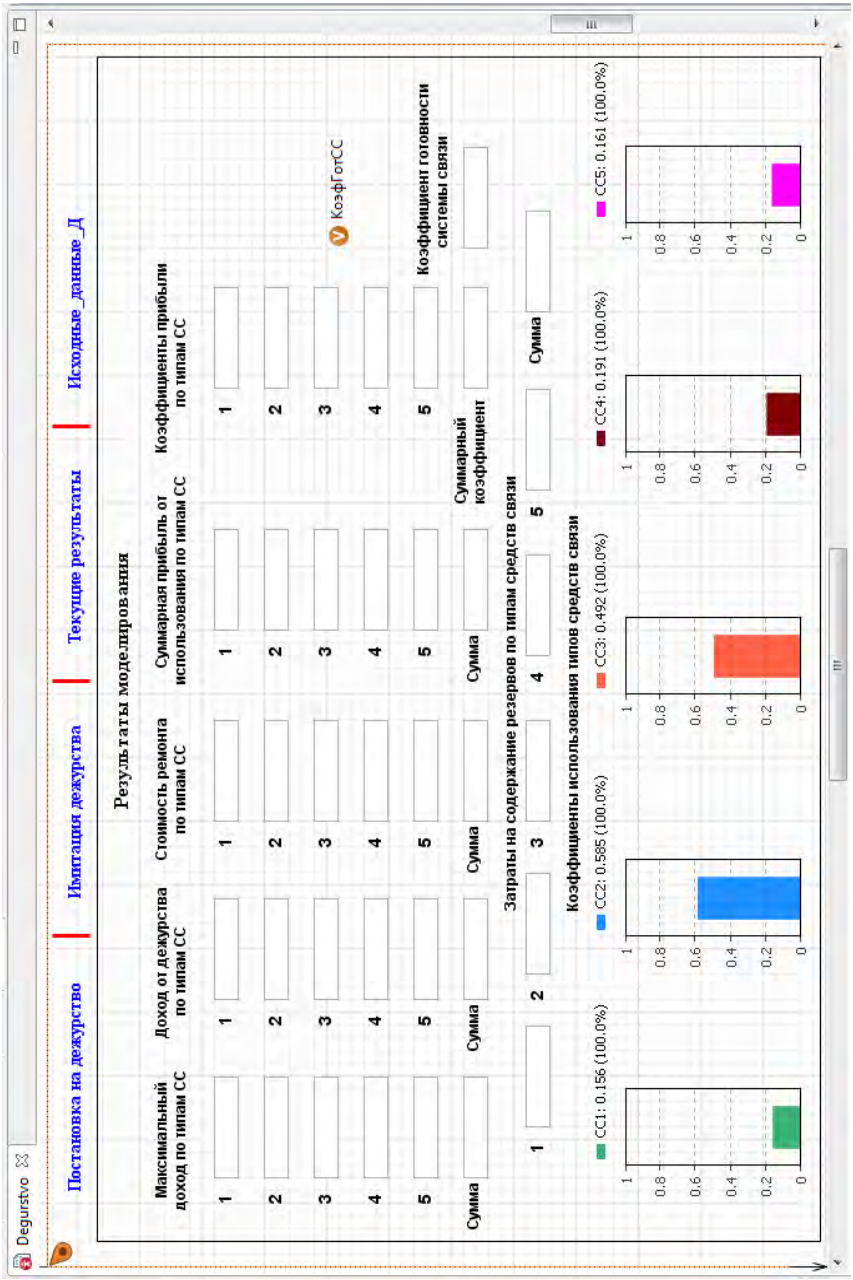


Рис. 5.12. Сегмент **Статистика**

10. Перетащите элемент **text** и в поле **Текст:** введите `Результаты моделирования`. На странице **Местоположение и размер** введите в поля **X:** `360`, **Y:** `1126`.

11. Перетащите еще тринадцать элементов **text**, разместите и введите в соответствующие поля **Текст:** надписи, как на рис. 5.12. Например, при размещении надписи `Максимальный доход по типам СС` укажите в полях **X:** `100`, **Y:** `1160`, при размещении надписи `Коэффициенты использования типов средств связи` укажите в полях **X:** `280`, **Y:** `1490`, а при размещении надписи `Затраты на содержание резервов по типам средств связи` в полях **X:** `280`, **Y:** `1440`.

5.1.6.1. 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. Мы будеи выводить в эти текстовые поля результаты моделирования.

1. Копированием разместите согласно рис. 5.12 элементы **Текстовое поле**. При копировании используйте следующие рекомендации. Перетащите первый элемент **Текстовое поле**. В поле **Имя:** дайте ему имя editbox1.

2. На странице **Внешний вид** в полях **Шрифт:** установите шрифт по своему усмотрению из выпадающего списка или оставьте предложенный, размер шрифта установите 14.

3. На странице **Местоположение и размер** введите в поля **X:** `70`, **Y:** `1210`, **Ширина:** `80`, **Высота:** `20`.

4. Скопируйте editbox1. Вставьте ещё пять таких элементов. При вставке ориентируйтесь на клетки рабочего поля. Например, в данном случае между элементами по вертикали две клетки.

5. Слева от элементов, используя элемент **Текст**, разместите номера типов СС 1…5. Шестому элементу введите имя `Сумма`.

6. Скопируйте только что созданные номера элементов **Текстовое поле** и сами элементы. Вставьте скопированное четыре раза.

7. Вставьте пятый раз. Разместите шесть элементов **Текстовое поле** горизонтально как на рис. 5.12.

8. Перетащите элемент **Текстовое поле** с именем editbox37. С использованием элемента **Текст** введите **Коэффициент готовности системы связи**.

Таблица 5.5

Имена элементов **Текстовое поле**

1	2	3	4	5	Сумма
Максимальный доход по типам СС					
editbox1	editbox2	editbox3	editbox4	editbox5	editbox6
Доход от дежурства по типам СС и всего					
editbox7	editbox8	editbox9	editbox10	editbox11	editbox12
Стоимость ремонта по типам СС и всего					
editbox13	editbox14	editbox15	editbox16	editbox17	editbox18
Суммарная прибыль от использования СС и всего					
editbox19	editbox20	editbox21	editbox22	editbox23	editbox24
Коэффициенты прибыли по типам СС и всего					
editbox25	editbox26	editbox27	editbox28	editbox29	editbox30
Затраты на содержание резервов по типам СС и всего					
editbox31	editbox32	editbox33	editbox34	editbox35	editbox36

5.1.6.2. 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, затем добавленному вторым 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 первый элемент **Диаграмма с накоплением** из палитры **Статистика** и разместите согласно рис. 5.12.
2. На панели **Свойства** установите: **Масштаб:** Фиксированный, **Обновлять данные автоматически**.
3. На странице **Внешний вид** установите:

Направление: `вертикальное`

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

4. На странице **Местоположение и размер** введите в поля **X:** `40`, **Y:** `1510`, **Ширина:** `230`, **Высота:** 180.
5. На странице **Легенда** установите **Высота:** `20`, **Расположение** `сверху`.
6. На страницы **Область диаграммы** введите в поля **Смещение по оси X:** `40`, **Смещение по оси Y:** `10`, **Ширина:** `60`, **Высота:** 140.
7. Щёлкните **Добавить элемент данных**.
8. В поле **Заголовок:** введите `CC1`.
9. В поле **Значение:** введите Java код

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

10. Скопируйте первый элемент **Диаграмма с накоплением**.
11. Вставьте четыре элемента 2…5. После вставки в полях **Заголовок:** вставленных элементов внесите правки для получения имён `CC2, CC3, CC4, CC5` соответственно.
12. В поле **Значение:** вводите Java коды также для элементов 2…5 соответственно:

```
degCC2.statsUtilization.mean()
```

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

На этом построение сегмента **Статистика** завершено.

5.1.7. 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 **Основная** на диаграмму размещения элементов **Переменная для вывода результатов моделирования** (рис. 5.10). Измените его имя на `РезультатыМоделирования`. Нажмите Enter.

2. Установите флажок **Отображать имя**. С помощью выпадающего списка **Тип события:** выберите `По таймауту`.

3. Установите **Режим:** `Срабатывает один раз`.

4. **Время срабатывания (абсолютное)** `1000000`.

5. В поле **Действие** введите Java код, который будет выполняться при появлении этого события.

```
//Расчет результатов по СС1
DoxDegCC1=round((DoxDegCC1/КолПрогон)*100);
DoxDegCC1=DoxDegCC1/100;
ZatrRemCC1=round((ZatrRemCC1/КолПрогон)*100);
ZatrRemCC1=ZatrRemCC1/100;
UbitokCC1=round((1-
degCC1.statsUtilization.mean())*main.ubitokCC1*Время
РабСист*КСС1*100);
UbitokCC1=UbitokCC1/100;
PribCC1=round((DoxDegCC1-
(ZatrResCC1+ZatrRemCC1+UbitokCC1))*100);
PribCC1=PribCC1/100;
koefPribCC1=round((PribCC1/DoxMaxCC1)*1000);
koefPribCC1=koefPribCC1/1000;
//Расчет результатов по СС2
DoxDegCC2=round((DoxDegCC2/КолПрогон)*100);
DoxDegCC2=DoxDegCC2/100;
ZatrRemCC2=round((ZatrRemCC2/КолПрогон)*100);
ZatrRemCC2=ZatrRemCC2/100;
UbitokCC2=(1-
degCC2.statsUtilization.mean())*main.ubitokCC2*Время
РабСист*КСС2;
PribCC2=round((DoxDegCC2-
(ZatrResCC2+ZatrRemCC2+UbitokCC2))*100);
PribCC2=PribCC2/100;
koefPribCC2=round((PribCC2/DoxMaxCC2)*1000);
koefPribCC2=koefPribCC2/1000;
//Расчет результатов по СС3
DoxDegCC3=round((DoxDegCC3/КолПрогон)*100);
DoxDegCC3=DoxDegCC3/100;
ZatrRemCC3=round((ZatrRemCC3/КолПрогон)*100);
ZatrRemCC3=ZatrRemCC3/100;
```

```
UbitokCC3=(1-degCC3.statsUtilization.mean())*
main().ubitokCC3*ВремяРабСист*КСС3;
PribCC3=round((DoxDegCC3-
(ZatrResCC3+ZatrRemCC3+UbitokCC3))*100);
PribCC3=PribCC3/100;
koefPribCC3=round((PribCC3/DoxMaxCC3)*1000);
koefPribCC3=koefPribCC3/1000;
//Расчет результатов по СС4
DoxDegCC4=round((DoxDegCC4/КолПрогон)*100);
DoxDegCC4=DoxDegCC4/100;
ZatrRemCC4=round((ZatrRemCC4/КолПрогон)*100);
ZatrRemCC4=ZatrRemCC4/100;
UbitokCC4=(1-degCC4.statsUtilization.mean())*
main.ubitokCC4*ВремяРабСист*КСС4;
PribCC4=round((DoxDegCC4-
(ZatrResCC4+ZatrRemCC4+UbitokCC4))*100);
PribCC4=PribCC4/100;
koefPribCC4=round((PribCC4/DoxMaxCC4)*1000);
koefPribCC4=koefPribCC4/1000;
//Расчет результатов по СС5
DoxDegCC5=round((DoxDegCC5/КолПрогон)*100);
DoxDegCC5=DoxDegCC5/100;
ZatrRemCC5=round((ZatrRemCC5/КолПрогон)*100);
ZatrRemCC5=ZatrRemCC5/100;
UbitokCC5=(1-degCC5.statsUtilization.mean())*
main.ubitokCC5*ВремяРабСист*КСС5;
PribCC5=round((DoxDegCC5-
(ZatrResCC5+ZatrRemCC5+UbitokCC5))*100);
PribCC5=PribCC5/100;
koefPribCC5=round((PribCC5/DoxMaxCC5)*1000);
koefPribCC5=koefPribCC5/1000;
//Расчет суммарных результатов
SumDoxDeg=DoxDegCC1+DoxDegCC2+DoxDegCC3+
DoxDegCC4+DoxDegCC5;
SumZatrRem=round((SumZatrRem)*100/КолПрогон);
SumZatrRem=SumZatrRem/100;
SumUbitok=UbitokCC1+UbitokCC2+UbitokCC3+
UbitokCC4+UbitokCC5;
SumPribil=round((SumDoxDeg-
(SumZatrRes+SumZatrRem+SumUbitok))*100);
SumPribil=SumPribil/100;
koefPribil=round((SumPribil/SumDoxMax)*1000);
koefPribil=koefPribil/1000;
```

```
//вывод максимального дохода, дохода от дежурства по
типам СС и всего
editbox1.setText(DoxMaxCC1);
editbox7.setText(DoxDegCC1);
editbox2.setText(DoxMaxCC2);
editbox8.setText(DoxDegCC2);
editbox3.setText(DoxMaxCC3);
editbox9.setText(DoxDegCC3);
editbox4.setText(DoxMaxCC4);
editbox10.setText(DoxDegCC4);
editbox5.setText(DoxMaxCC5);
editbox11.setText(DoxDegCC5);
editbox6.setText(SumDoxMax);
editbox12.setText(SumDoxDeg);
//вывод стоимости ремонта по типам СС и всего
editbox13.setText(ZatrRemCC1);
editbox14.setText(ZatrRemCC2);
editbox15.setText(ZatrRemCC3);
editbox16.setText(ZatrRemCC4);
editbox17.setText(ZatrRemCC5);
editbox18.setText(SumZatrRem);
//вывод чистой прибыли от использования по типам СС
и всего
editbox19.setText(PribCC1);
editbox20.setText(PribCC2);
editbox21.setText(PribCC3);
editbox22.setText(PribCC4);
editbox23.setText(PribCC5);
editbox24.setText(SumPribil);
//вывод коэфициентов прибыли по типам СС и всего
editbox25.setText(koefPribCC1);
editbox26.setText(koefPribCC2);
editbox27.setText(koefPribCC3);
editbox28.setText(koefPribCC4);
editbox29.setText(koefPribCC5);
editbox30.setText(koefPribil);
//вывод затрат на содержание резервов по типам СС и
всего
editbox31.setText(ZatrResCC1);
editbox32.setText(ZatrResCC2);
editbox33.setText(ZatrResCC3);
editbox34.setText(ZatrResCC4);
editbox35.setText(ZatrResCC5);
editbox36.setText(SumZatrRes);
```

```
//вывод коэфициента готовности системы связи
КоэфГотСС=(degCC1.statsUtilization.mean()+
degCC2.statsUtilization.mean()+
degCC3.statsUtilization.mean()+
degCC4.statsUtilization.mean()+
degCC5.statsUtilization.mean())/КолТипСС;
editbox37.setText(КоэфГотСС);
```

Из кода следует, что по окончании моделирования, которое длится 1 000 000 единиц модельного времени, сработает метод **событие**, будут рассчитаны и выведены результаты моделирования.

Для округления результатов моделирования (коэффициентов прибыли до трех знаков после запятой, а абсолютных величин прибыли и затрат — до двух знаков) использован метод round(). Предварительно результат умножался на 1000 и 100, а потом делился на эти же величины.

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

```
editbox1.setText(DoxMaxCC1);
```

5.1.8. Переключение между областями просмотра

Мы не акцентировали внимание при построении модели на элементах презентации, которые вы уже видели на некоторых рисунках, для переключения между областями просмотра. Если вы не добавляли их, добавьте и укажите свойства согласно табл. 5.6.

Синим цветом выделены элементы презентации, предназначенные для перехода по щелчку к соответствующим областям просмотра. Чёрным цветом выделены элементы презентации без перехода к каким-либо областям просмотра.

Таблица 5.6

Элемент презентации	Действие по щелчку
Постановка на дежурство (рис. 5.7)	

Имитация дежурства.
Текущие результаты
Статистика
Исходные данные
	`degurstvo.дежурство.navigateTo();`
	
	`degurstvo.статистика.navigateTo();`
	`исходные_данные_ПД.navigateTo();`

Исходные данные ПД (рис. 5.3)	
Постановка на дежурство	`постановка.navigateTo();`

Продолжение табл. 5.6

Элемент презентации	Действие по щелчку
Имитация дежурства (рис. 5.11)	

Постановка на дежурство
Статистика
Исходные данные
	`main.постановка.navigateTo();`
	`статистика.navigateTo();`
	`исходные_данные_Д.navigateTo();`

Исходные данные Д (рис. 5.9)	

Постановка на дежурство
Имитация дежурства
Статистика
	`main.постановка.navigateTo();`
	`дежурство.navigateTo();`
	`статистика.navigateTo();`

Текущие результаты (рис. 5.10)	

Постановка на дежурство
Имитация дежурства
Статистика
	`main.постановка.navigateTo();`
	`дежурство.navigateTo();`
	`статистика.navigateTo();`

Статистика (рис. 5.12	

Постановка на дежурство
Имитация дежурства
Текущие результаты
Исходные данные
	`main.постановка.navigateTo();`
	`дежурство.navigateTo();`
	`текущие_результаты.navigateTo();`
	`исходные_данные_Д.navigateTo();`

5.1.9. Отладка модели

В ходе построения модели вы можете совершить различные ошибки. Мы же остановимся на одной их тех, которая требует хороших знаний или «тонкостей» AnyLogic.

Запустите модель. Ошибка, о которой мы говорим, появится практически сразу после запуска (рис. 5.13 и 5.14).

В **Библиотеке моделирования процессов** реализован pull-протокол передачи заявок между объектами. Суть заключается в том, что следующий объект «вытягивает» заявку из предыдущего объект, если он может её принять. Это работает для объектов, которые могут держать в себе заявку некоторое время. Для примера рассмотрим следующую диаграмму процесса:

```
...-> split -> selectOutput -> queue -> delay ->...
```

В этом случае объекты, которые заявка проходит насквозь (split и selectOutput), просматриваются объектами queue и delay, чтобы оценить возможность прохода заявки.

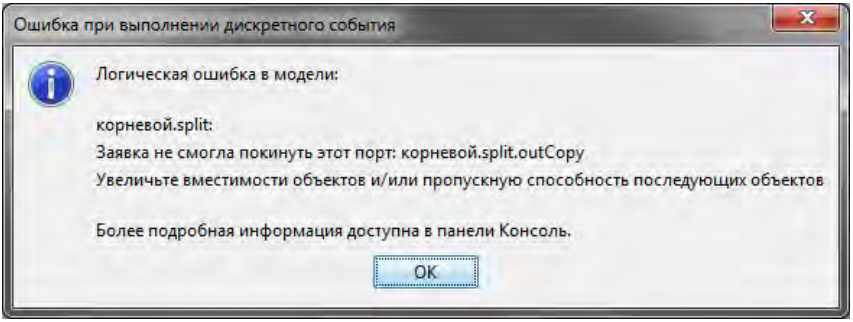


Рис. 5.13. Сообщение об ошибке

Если объекты могут принять заявку, они мгновенно «вытягивают» её, вследствии чего код в поле **При выходе копии** объекта split, который был нами ранее введён (см. рис. 5.5) не выполняется. Это приводит к некорректной маршрутизация заявок в модели.

Поясним это. На рис. 5.14 видно, что 64 заявки, соответствующие типу CC1, проходят не через свой выход true, а через выход false как заявки, соответствующие типу CC5. Это может быть только в том случае, когда в поле entity.tipCC ничего не заносится, то есть при неработающем коде в поле **При выходе копии** объекта split.

Для выполнения некоторых действий заявки, когда она проходит через какое-то место диаграммы процесса, в данном случае перед попаданием в selectOutput, но уже после выхода из split, можно использовать объект **plain Transfer**. В него вы можете вписать код для действий заявки, который был ранее записан в поле **При выходе копии** объекта split.

1. Удалите связь между выходом копии объекта split и входом объекта selectOutput.
2. В **Библиотеке моделирования процессов** откройте **Местоположение и размер**.
3. Перетащите объект **plain Transfer** и соедините его вход с выходом копии объекта split, а выход — со входом объекта selectOutput. Замените тип заявки на ComFacility.
4. Код из поля **При выходе копии** объекта split перепишите в поле **При подходе ко входу:** объекта **plain Transfer**.

Запустите модель. На рис. 5.15 и рис. 5.16 показаны результаты моделирования.

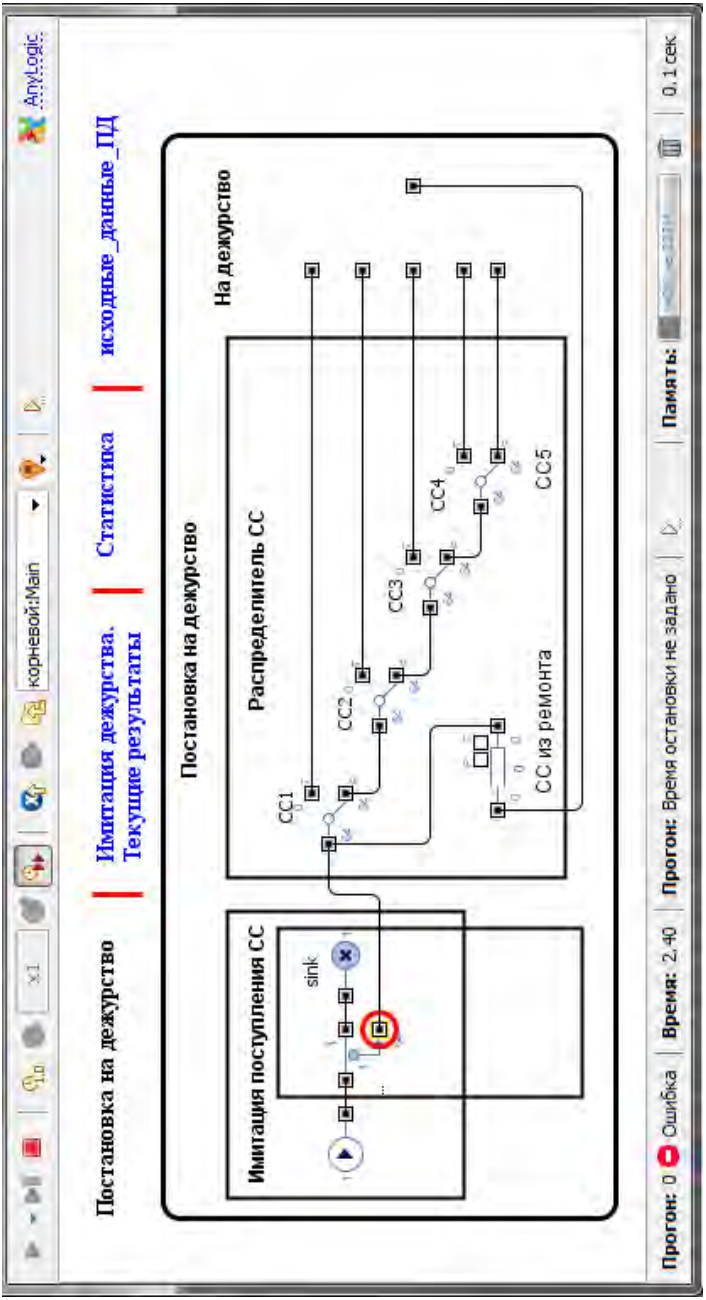


Рис. 5.14.Остановка модели по ошибке

Рис. 5.15. Результаты моделирования

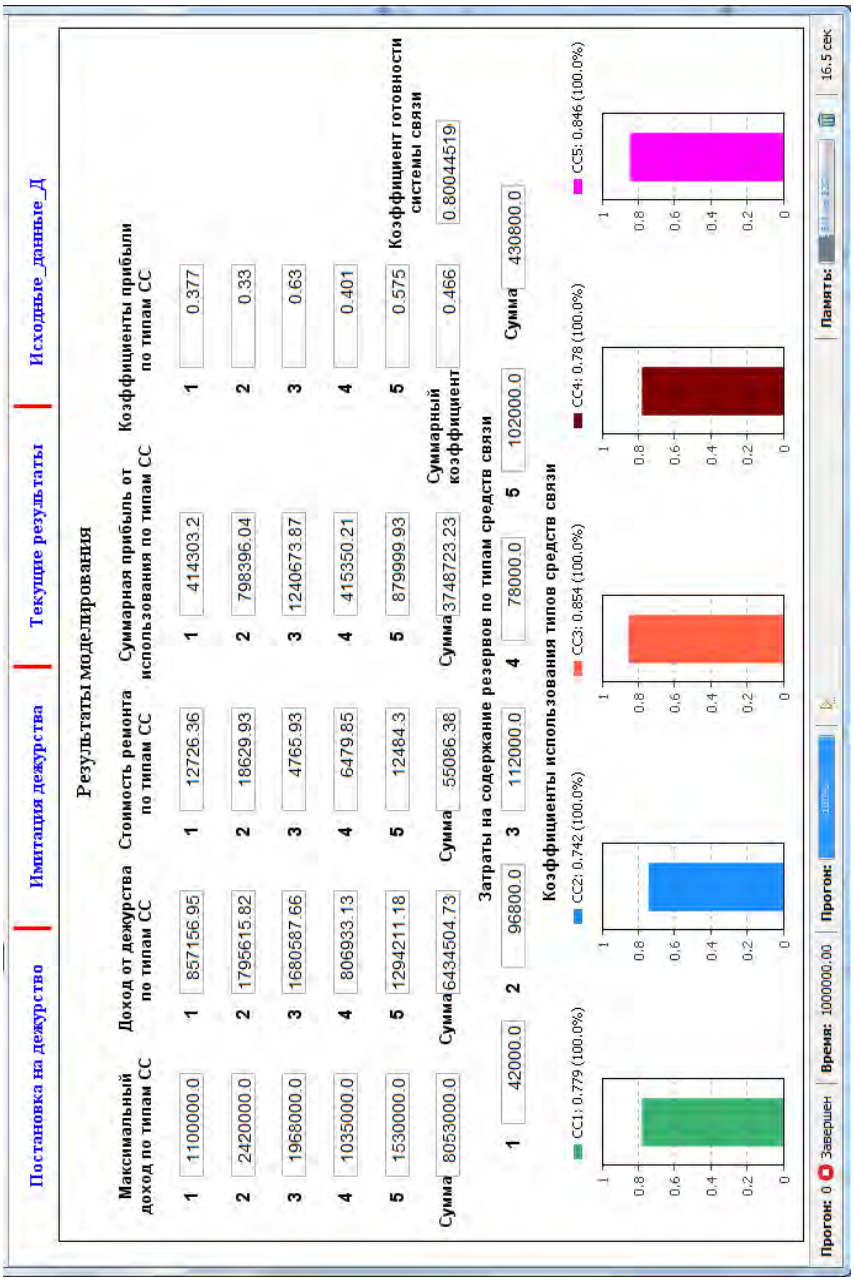


Рис. 5.16. Результаты моделирования в сегменте **Статистика**

5.1.10. 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.

5.1.10.1. 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 доступных в версии AnyLogic University экспериментах оптимизации и варьирования переменных, овладев методиками проведения которых, вы самостоятельно сможете выполнять в AnyLogic Professional и другие эксперименты.

5.1.10.2. Связывание параметров

Начиная создавать модель в AnyLogic, мы ничего не говорили об экспериментах и особенностях их проведения. Поэтому все исходные данные разместили на Исходные_данные_ПД (рис. 5.3) и Исходные_данные_Д (рис. 5.9) так, как нам представлялось удобным для построения модели и управления ею в ходе проведения простого эксперимента.

При проведении эксперимента и наличии в модели, как у нас, вложенных объектов, нужно связывать параметры корневого объекта и вложенных объектов, так как 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, находящемуся ниже уровнем в иерархическом дереве модели*. То есть связываются параметры последовательно от одного уровня к другому, а не через уровень или уровни.

Разместите элементы, как показано на рис. 5.17. Внесите соответствующие изменения в модель. Обратите внимание на различие имён связываемых параметров, например, KCCP_1.

Свяжите параметры корневого типа агента Main с параметрами вложенного объекта типа агента Degyrstvo.

1. Откройте диаграмму типа агента Main.
2. Выберите на диаграмме вложенный объект degyrstvo.
3. Перейдите на панель **Свойства**. В таблице **Параметры** в поле **Значение** введите имя параметра типа агента-владельца Main, значение которого нужно передавать параметру вложенного объекта. В результате у вас должно быть так, как на рис. 5.18.

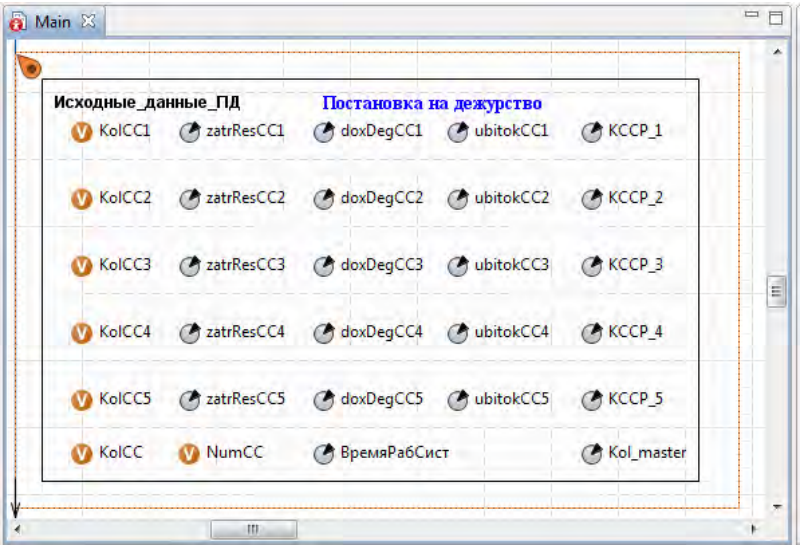


Рис. 5.17. Размещение элементов на **Исходные_данные_ПД**

Свойства

degurstvo - Degurstvo

Имя: degurstvo

Отображать имя | Исключить

Одиночный агент | Популяция агентов

KCC1:	=	55
KCC2:	=	100
KCC3:	=	60
KCC4:	=	45
KCC5:	=	60
KCCP1:	=	KCCP_1
KCCP2:	=	KCCP_2
KCCP3:	=	KCCP_3
KCCP4:	=	KCCP_4
KCCP5:	=	KCCP_5
stoimRemCC1:	=	17
stoimRemCC2:	=	18
stoimRemCC3:	=	16
stoimRemCC4:	=	20
stoimRemCC5:	=	21
timeRem1:	=	6.5
timeRem2:	=	4.2
timeRem3:	=	2.8
timeRem4:	=	3.0
timeRem5:	=	5.5
timeOtkaz1:	=	373
timeOtkaz2:	=	301
timeOtkaz3:	=	482
timeOtkaz4:	=	325
timeOtkaz5:	=	470
kol_master:	=	Kol_master

Рис. 5.18. Фрагмент страницы **Параметры** после связывания параметров

5.1.10.3. Первый эксперимент Оптимизация стохастических моделей

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

Наша модель **Система связи** 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** Щёлкните правой кнопкой мыши элемент модели **Система_связи** и из контекстного меню выберите **Создать/ Эксперимент**. В появившемся диалоговом окне из списка **Тип эксперимента:** выберите **Оптимизация** (рис. 5.19).

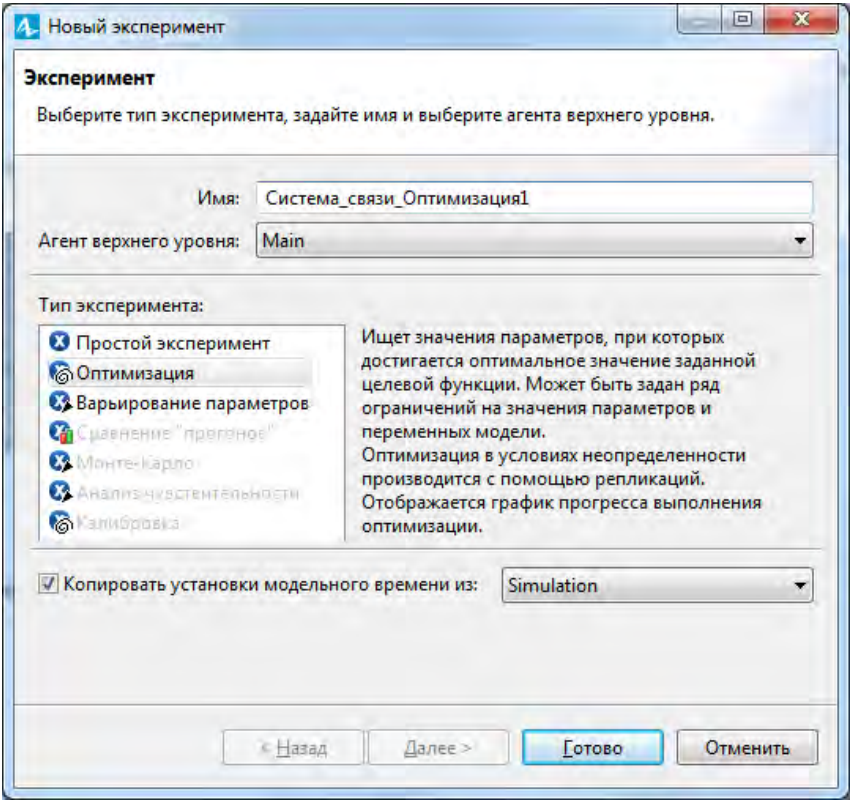


Рис. 5.19. Диалоговое окно **Новый эксперимент**

2. В поле **Имя:** введите имя эксперимента, например, `Система_связи_Оптимизация1`. Имя эксперимента должно начинаться с заглавной буквы — правило названия классов в Java.

3. В поле **Агент верхнего уровня:** выберите Main. Этим действием вы задали корневой (главный) класс эксперимента. Объект этого класса будет играть роль корня иерархического дерева объектов модели, запускаемой оптимизационным экспериментом.

4. 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. В данном случае оставьте, так как есть: `Simulation`.

5. Щёлкните кнопку **Готово**. Появится страница **Основные** панели **Свойства** (рис. 5.20).

6. Установите опцию **максимизировать**.

7. Целевая функция доступна как `root`. Поэтому в поле **Целевая функция** введите `root.degurstvo.koefPribil`.

8. На странице **Случайность** установите **Фиксированное начальное число (воспроизводимые прогоны)**.

9. В поле **Начальное число** введите 5672.

10. 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.

11. Число итераций модели при полном факторном эксперименте, то есть число всех возможных сочетаний факторов, определяется по формуле:

$$I = k_1 \cdot k_2 \cdot \ldots \cdot k_i \cdot \ldots \cdot k_m,$$

где k_i — число уровней i-го фактора, $i = \overline{1, m}$.

12. В нашей модели нужно менять количество резервных средств связи КССР_1…КССР_5 и количество мастеров-ремонтников Kol_master, то есть всего m=6 факторов. Факторы имеют следующие уровни: $k_1 = 3, k_2 = k_3 = k_4 = 6, k_5 = k_6 = 5.$ Тогда число итераций

$$I = k_1 \cdot k_2 \cdot k_3 \cdot k_4 \cdot k_5 \cdot k_6 = 3 \cdot 6 \cdot 6 \cdot 6 \cdot 5 \cdot 5 = 16\,200.$$

Свойства

Система_связи_Оптимизация1 - Оптимизационный эксперимент

Имя: Система_связи_Оптимиз ☐ Исключить

Агент верхнего уровня: Main

Целевая функция: ○ минимизировать ◉ максимизировать

root.degurstvo.koefPribil

☑ Количество итераций: 500

☐ Автоматическая остановка

Максимальный размер памяти: 256 Мб

Создать интерфейс

Параметры

Параметры:

Параметр	Тип	Значение			
		Мин.	Макс.	Шаг	Начальное
zatrResCC1	фиксированный	21			
zatrResCC2	фиксированный	24.2			
zatrResCC3	фиксированный	28			
zatrResCC4	фиксированный	26			
zatrResCC5	фиксированный	25.5			
doxDegCC1	фиксированный	20			
doxDegCC2	фиксированный	24.2			
doxDegCC3	фиксированный	32.8			
doxDegCC4	фиксированный	23			
doxDegCC5	фиксированный	25.5			
ubitokCC1	фиксированный	32			
ubitokCC2	фиксированный	34.2			
ubitokCC3	фиксированный	37			
ubitokCC4	фиксированный	31			
ubitokCC5	фиксированный	32.5			
ВремяРабСист	фиксированный	1000			
KCCP_1	дискретный	1	3	1	
KCCP_2	дискретный	1	6	1	
KCCP_3	дискретный	1	6	1	
KCCP_4	дискретный	1	6	1	
KCCP_5	дискретный	1	5	1	
Kol_master	дискретный	1	5	1	

Рис. 5.20. Вкладка **Основные** оптимизационного эксперимента

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

14. Задайте параметры, значения которых будут меняться. В таблице на рис. 5.21 перечислены все параметры агента верхнего уровня Main.

15. Чтобы разрешить варьирование параметров оптимизатором, перейдите на строку с параметром КССР_1. Щёлкните мышью в ячейке **Тип**. Выберите тип параметра, отличный от значения **фиксированный**. Так как параметр КССР_1 целочисленный типа int, выберите **дискретный**.

16. Задайте диапазон допустимых значений параметра. Для чего введите в ячейку **Мин.** минимальное значение 1, в ячейку **Макс.** максимальное значение, например, для КССР1, 3. Так как параметр дискретный, в ячейке **Шаг** укажите величину шага 1.

17. Задайте так же остальные параметры, как на рис. 5.20.

18. Перейдите на страницу **Репликации** панели **Свойства**.

19. Установите флажок **Использовать репликации**.

20. 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:

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

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 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 ускорен за счёт прекращения поиска неподходящих решений вместо выполнения заданного максимального количества репликаций модели.

21. На странице **Репликации** выберите опцию **Фиксированное количество репликаций** и в поле **Количество репликаций за итерацию:** установите рассчитанное число репликаций (прогонов) модели 4.

22. Вернитесь на страницу **Основные**. Щёлкните кнопку **Создать интерфейс**. После щелчка удаляется содержимое презентации эксперимента и создаётся интерфейс эксперимента заново (рис. 5.21) 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.

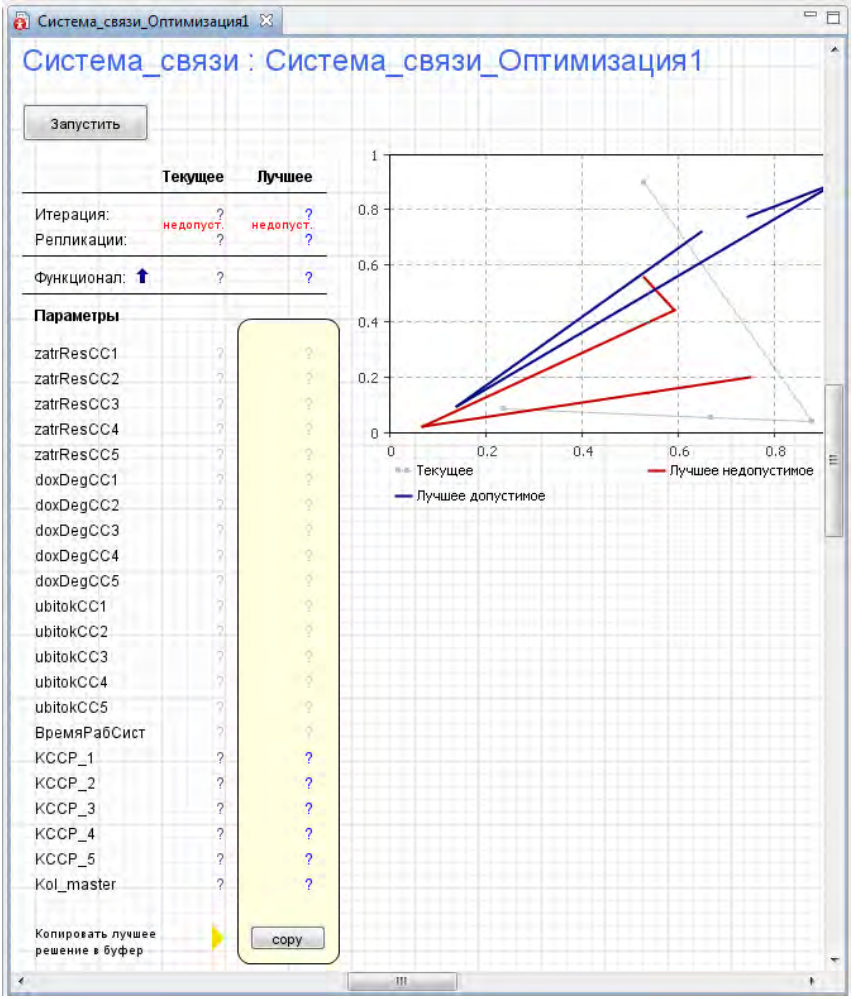


Рис. 5.21. Интерфейс оптимизационного эксперимента

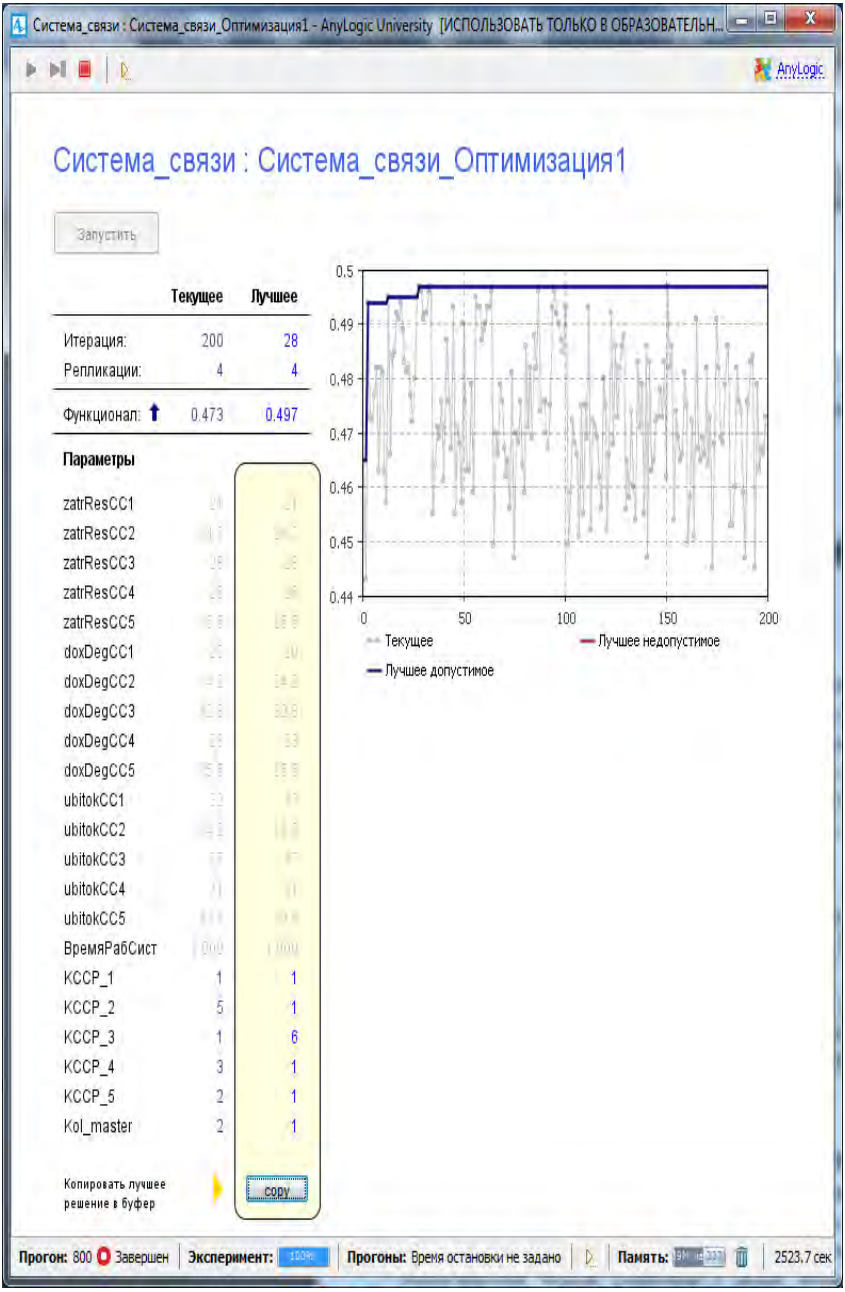


Рис. 5.22. Результаты первого оптимизационного эксперимента

23. В меню запуск выполните **Система_связи / Система_связи_Оптимизация1**.

24. Щёлкните **Запустить**. Начнёт выполняться эксперимент, в ходе которого можно видеть на графике изменения значения целевой функции. После 200*4=800 прогонов (рис. 5.22) эксперимент остановится.

25. Результаты оптимизационного эксперимента приведены на рис. 5.22. Наилучшее значение целевой функции — коэффициент прибыли равен 0,497. Получен он на 28-й итерации при следующих оптимальных значениях параметров: КССР_1 = КССР_2 = КССР_4 = КССР_5 = Kol_master =1, КССР_3 = 6.

Создайте и проведите второй оптимизационный эксперимент при тех же условиях с целью определения коэффициента готовности системы связи.

5.1.10.4. Изменение порядка отображения параметров на странице свойств своего объекта

При размещении параметров для ввода исходных данных и последующем их связывании мы не задумывались над тем, как они будут размещены на страницах свойств и тем более в интерфейсе оптимизационного эксперимента.

Но теперь нас не устраивает то, что оптимизируемые параметры размещены последними в интерфейсе первого оптимизационного эксперимента (рис. 5.20, 5.21, 5.22).

Разместите оптимизируемые параметры в верхних частях страниц свойств соответствующих объектов.

1. На панели свойств агента верхнего уровня Main откройте страницу **Предв. просмотр параметров**.
2. Выделите параметр КССР_1 и, щёлкая по верхней стрелке, расположенной справа от поля параметра, переместите его в верхнюю часть страницы свойств (рис. 5.23).
3. Проделайте то же для параметров КССР_2… КССР_5 и Kol_master. У вас должно быть так, как на рис. 5.24.
4. Закройте панель свойств типа агента Main.
5. Выделите тип агента Degurstvo. На панели свойств откройте страницу **Предв. просмотр параметров**.
6. Переместите последовательно параметры КССР1…КССР5 и kol_master в верхнюю часть страницы свойств.
7. Перейдите к созданию второго оптимизационного эксперимента.

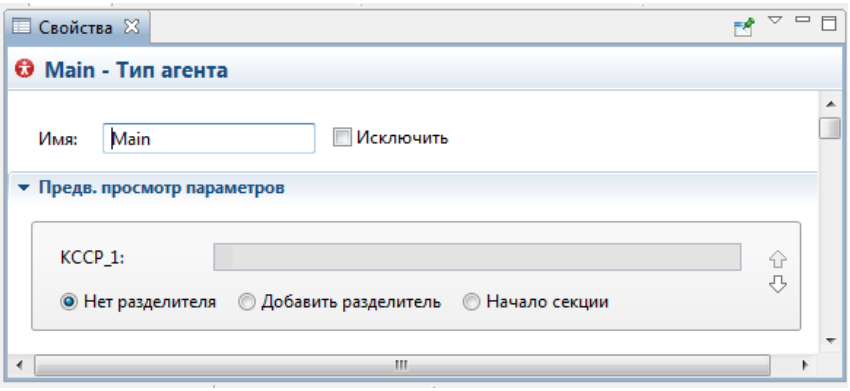


Рис. 5.23. Перемещение параметра на странице предварительного просмотра

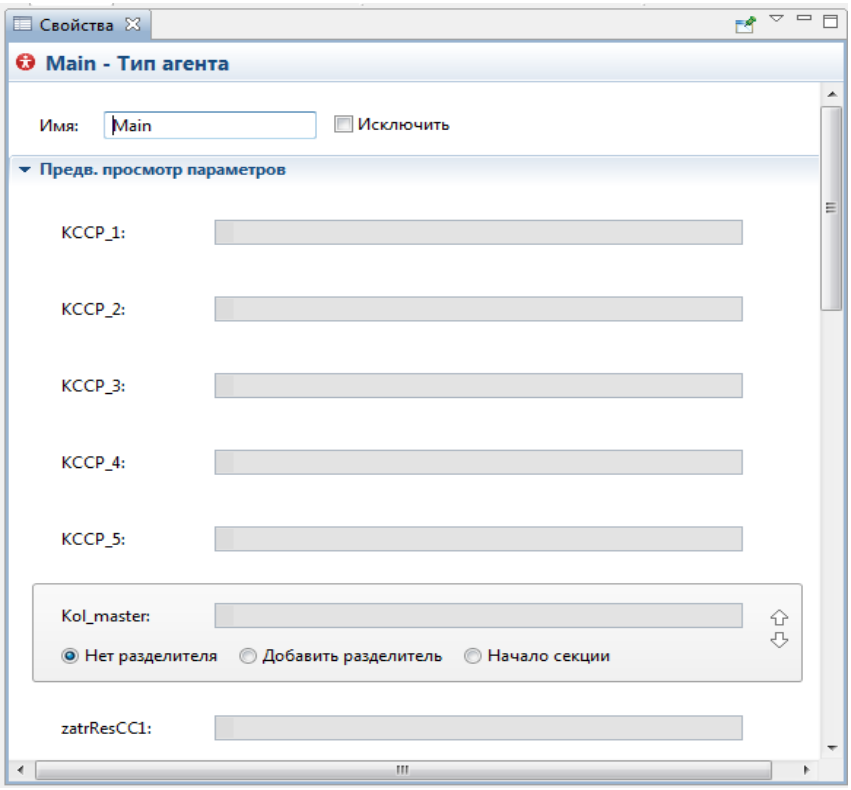


Рис. 5.24. Оптимизируемые параметры перемещены в верхнюю часть страницы свойств типа агента Main

5.1.10.5. Второй эксперимент Оптимизация стохастических моделей

Итак, второй оптимизационный эксперимент проводится при тех же условиях, что и первый оптимизационный эксперимент.

1. В панели **Проект** Щёлкните правой кнопкой мыши элемент модели **Система_связи** и из контекстного меню выберите **Создать/Эксперимент**. В диалоговом окне из списка **Тип эксперимента:** выберите **Оптимизация**.
2. В поле **Имя:** введите `Система_связи_Оптимизация2` имя эксперимента.
3. Установите `максимизировать`. В поле **Целевая функция** введите: `root.degurstvo.КоэфГотСС`
4. На странице **Случайность** установите **Фиксированное начальное число (воспроизводимые прогоны)**.
5. В поле **Начальное число** введите 5672.
6. На странице **Репликации** выберите опцию **Фиксированное количество репликаций** и в поле **Количество репликаций за итерацию:** установите число репликаций (прогонов) модели `4`.
7. Щёлкните **Создать интерфейс**. Обратите внимание**,** что оптимизируемые параметры расположены вверхней части**.** Запустите модель.

В результате второго эксперимента (рис. 5.25) наилучшее значение целевой функции — коэффициент готовности системы связи равен 0,813. Получен он на 35-й итерации при следующих оптимальных значениях параметров: KCCP_1 = KCCP_2 =1, KCCP_3 = KCCP_4 = 6, KCCP_5 = 5, Kol_master = 5.

8. Вернитесь к простому эксперименту. Измените значения KCCP_1…KCCP_5 и Kol_master на **Исходные_данные_ПД** на значения, полученные в первом оптимизационном эксперименте.
9. Запустите простой эксперимент. Вы получите коэффициент прибыли 0,497, т.е. такой же, как и в первом эксперименте. Коэффициент готовности системы связи равен 0,803.
10. Измените значения KCCP_1…KCCP_5 и Kol_master на **Исходные_данные_ПД** на значения, полученные во втором оптимизационном эксперименте.
11. Запустите простой эксперимент.
12. Вы получите коэффициент готовности системы связи о,813, то есть такой же, как и во втором эксперименте. Коэффициент прибыли равен 0,475, то есть уменьшился за счёт увеличения расходов на резервные СС.

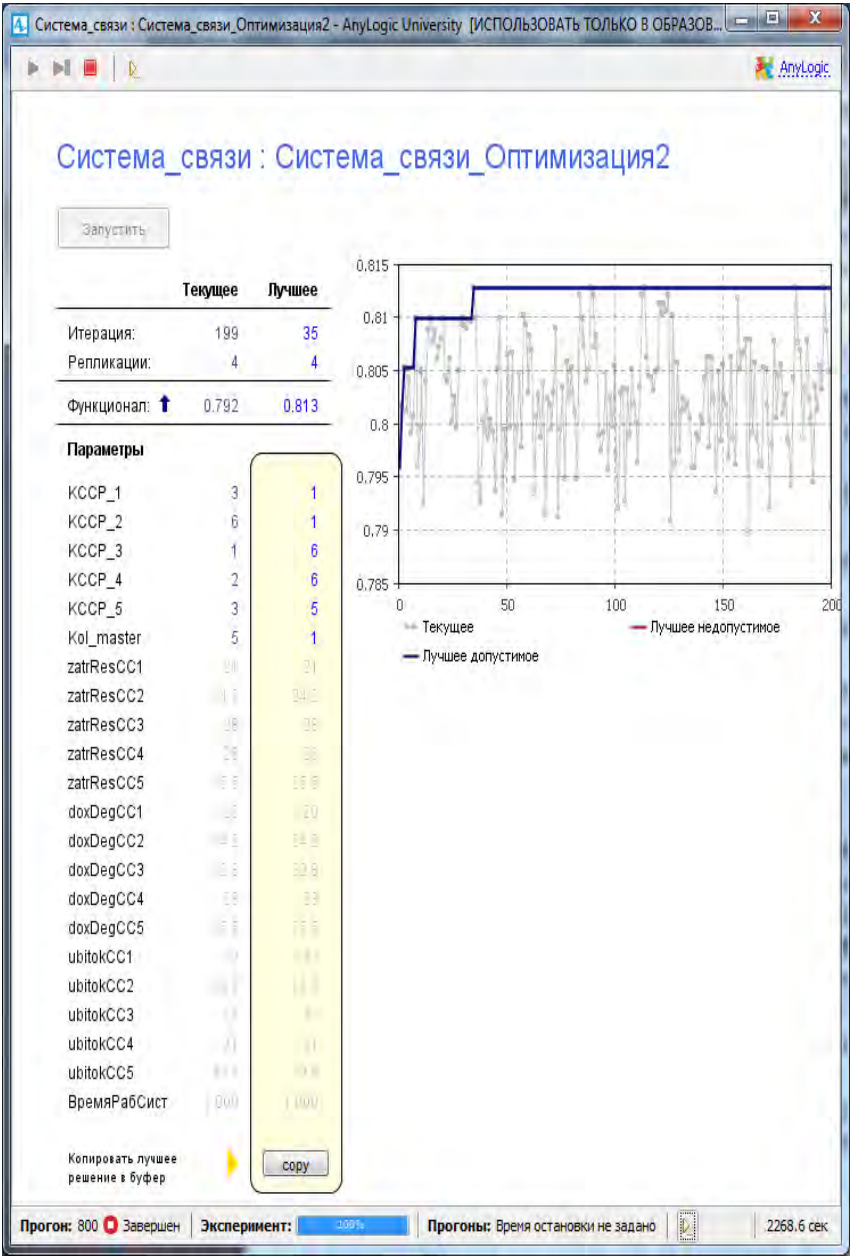


Рис. 5.25. Результаты второго оптимизационного эксперимента

5.1.10.6. Эксперимент Варьирование параметров

Эксперимент **Варьирование параметров** может проводиться только для детерминированных моделей.

Создайте эксперимент **Варьирование параметров** для стохастической модели `Система_связи` с целью наблюдения за изменением коэффициента готовности в зависимости от количества резервных СС и мастеров-ремонтников.

1. В панели **Проект** Щёлкните правой кнопкой мыши элемент модели `Система_связи` и из контекстного меню выберите **Создать эксперимент**.
2. В появившемся диалоговом окне из списка **Тип эксперимента:** выберите **Варьирование параметров**.
3. В поле **Имя** введите имя эксперимента, например, `Система_связи_Вар_параметров`.
4. Щёлкните кнопку **Готово**. Появится страница **Параметры** панели **Свойства** (рис. 5.26).
5. Обратите внимание, что на странице **Параметры** по сравнению с оптимизационным экспериментом отсутствуют опции **минимизировать**, **максимизировать**, **Количество итераций**. Последнее определяется AnyLogic в зависимости от диапазонов и шагов изменения параметров. Не используются и репликации, по
6. Задайте диапазон допустимых значений параметров. Перейдите в таблице на рис. 5.26 на строку с этим параметром. Щёлкните мышью в ячейке **Тип**. Выберите тип параметра, отличный от значения **фиксированный**. Параметр КССР_1 типа int, поэтому он может изменяться в диапазоне. Выберите **Диапазон**. В ячейку **Мин** введите минимальное значение `1`, в ячейку **Макс** — максимальное значение `3`, в ячейке **Шаг** укажите величину шага `1`.
7. Задайте также остальные параметры, как на рис. 5.26.
8. Вернитесь на страницу **Основные** и щёлкните кнопку **Создать интерфейс**.
9. В эксперименте **Варьирование параметров** в отличие от эксперимента **Оптимизация** интерфейс создаёт пользователь. Связано это с тем, что выходными результатами данного эксперимента могут быть любые показатели моделируемой системы.
10. Создайте интерфейс, показанный на рис. 5.27. Автоматически добавлены только параметры эксперимента. Здесь вы видите график, который будет отображать значение коэффициента готовности системы связи для каждой итерации.

Свойства

Система_связи_Вар_параметров...мент варьирования параметров

Имя: Система_связи_Вар_пара ☐ Исключить

Агент верхнего уровня: Main

Максимальный размер памяти: 256 Мб

Создать интерфейс

Параметры

Параметры: ◉ Варьировать в диапазоне ○ Произвольно

Кол-во "прогонов" 10

Параметр	Тип	Значение		
		Мин.	Макс.	Шаг
KCCP_1	Диапазон	1	3	1
KCCP_2	Диапазон	1	6	1
KCCP_3	Диапазон	1	6	1
KCCP_4	Диапазон	1	6	1
KCCP_5	Диапазон	1	5	1
Kol_master	Диапазон	1	5	1
zatrResCC1	Фиксированный	21		
zatrResCC2	Фиксированный	24.2		
zatrResCC3	Фиксированный	28		
zatrResCC4	Фиксированный	26		
zatrResCC5	Фиксированный	25.5		
doxDegCC1	Фиксированный	20		
doxDegCC2	Фиксированный	24.2		
doxDegCC3	Фиксированный	32.8		
doxDegCC4	Фиксированный	23		
doxDegCC5	Фиксированный	25.5		
ubitokCC1	Фиксированный	32		
ubitokCC2	Фиксированный	34.2		
ubitokCC3	Фиксированный	37		
ubitokCC4	Фиксированный	31		
ubitokCC5	Фиксированный	32.5		
ВремяРабСист	Фиксированный	1000		

Рис. 5.26. Страница **Параметры** эксперимента варьирования параметров

11. Перетащите элемент **График** из палитры **Статистика** на диаграмму.

12. На странице **Местоположение и размер** и установите: **X:** `260`, **Y:** `100`, **Ширина:** `510`, **Высота:** `400`, **Цвет фона:** `Нет заливки`, **Цвет границы:** `Нет линии`.

13. Щёлкните **Добавить элемент данных**.

14. Установите опцию **Набор данных**. **Заголовок:** `КоэфГотСС`. **Набор данных:** `dataset`. Установите **Не обновлять данные автоматически**.

15. Перейдите на страницу **Область диаграммы**. Установите: **Смещение по X:** `50`, **Смещение по Y:** `30`, **Ширина:** `450`, **Высота:** `330`.

16. Из палитры **Статистика** перетащите элемент **Набор данных**. Установите опцию **Не обновлять автоматически**.

17. Из палитры **Основная** перетащите элемент **Переменная**. На панели **Свойства** в поле **Имя:** введите `коэфГотСС`. Установите **Уровень доступа:** `public`. **Тип:** `double`.

18. Щёлкните диаграмму интерфейса. На странице **Действия Java** панели **Свойства** и введите коды:

в поле **Действие после прогона модели:**

```
коэфГотСС = root.degurstvo. КоэфГотСС;
```

в поле **Действие после итерации**

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

19. Выполните

Система_связи / Система_связи_Вар_параметров.

20. Щёлкните **Запустить**. Начнет выполняться эксперимент. Во время эксперимента можно видеть на графике изменение значения коэффициента готовности системы связи. Фрагмент результатов выполнения эксперимента **Варьирование параметров** приведен на рис. 5.28.

21. Эксперимент был приостановлен после 301 прогона. По графику можно видеть как изменяется коэффициент готовности системы связи. В данный момент времени коэффициент готовности равен 0,801 при КССР_1=КССР_5 = kol_master =1, КССР_2= КССР_3=5, КССР_4=3.

22. На рис. 5.28 видно, что 301 прогон модели был выполнен за 2029,3 с.

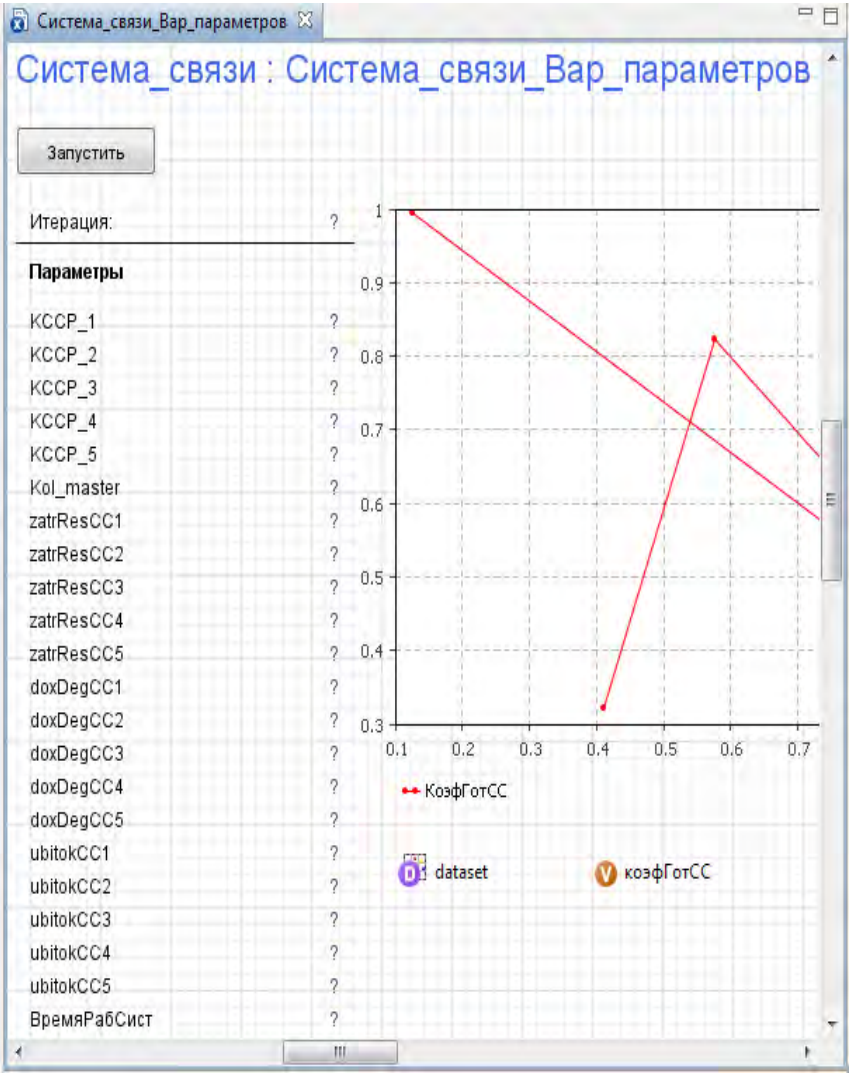


Рис. 5.27. Интерфейс эксперимента варьирования параметров

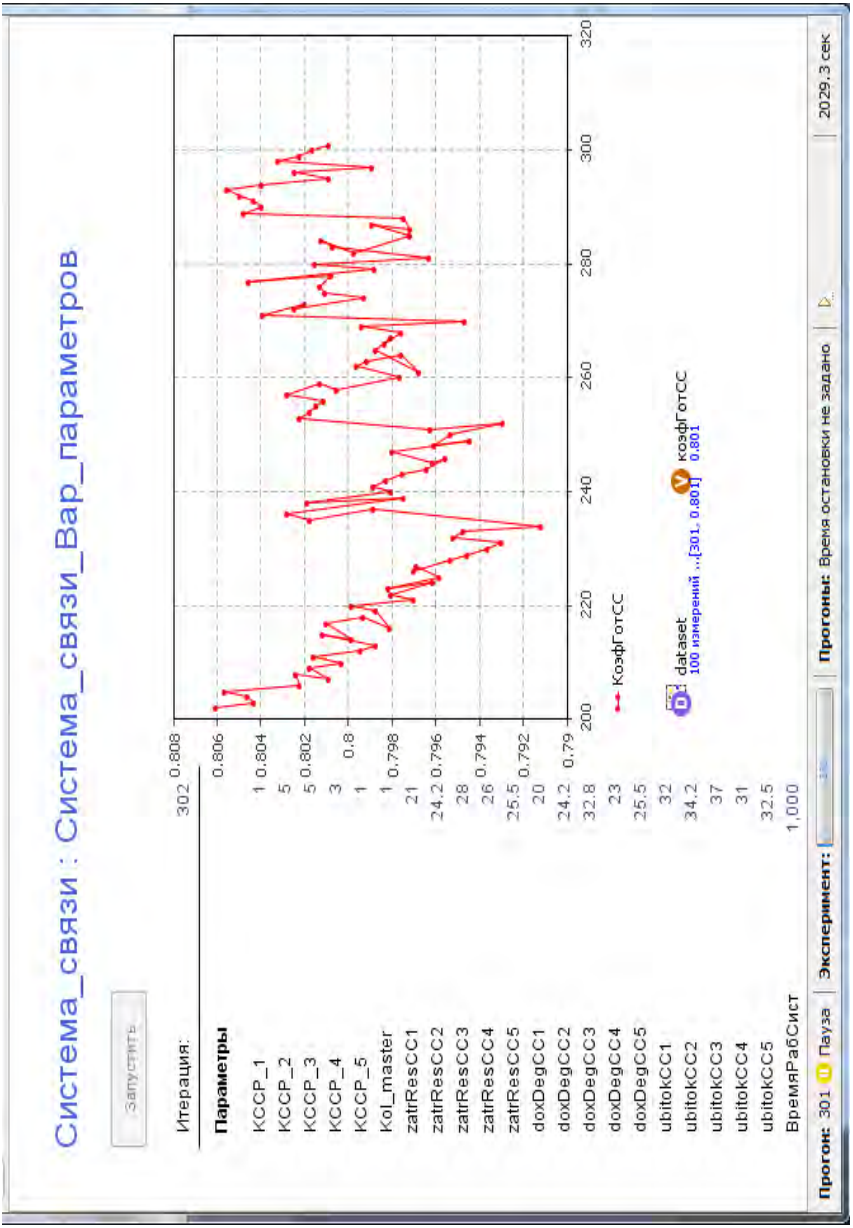


Рис. 5.28. Фрагмент результатов выполнения эксперимента **Варьирование параметров**

5.2. Интерпретация результатов моделирования

В соответствии с постановкой задачи нужно определить такое сочетание количества резервных СС и мастеров-ремонтников, при котором доход от предоставления услуг системой связи будет максимальным.

В GPSS World имеются средства для проведения оптимизационного эксперимента. Однако провести его так, чтобы он был аналогичен оптимизационному эксперименту в AnyLogic и, благодаря этому, можно было бы сравнивать результаты оптимизации, не представляется возможным. Во-первых, число факторов в GPSS World не может быть более пяти. Во-вторых, ремонтное подразделение имитируется МКУ, которое описывается командой STORAGE A. Операнд A этой команды, задающий ёмкость МКУ, должен быть только числом. Факторы же эксперимента обязательно должны быть переменными пользователя и не могут быть на месте операнда A. Отсюда нет возможности изменять в ходе эксперимента количество мастеров-ремонтников. Для изменения количества мастеров-ремонтников такая возможность есть.

Поэтому для достижения цели работы — установления адекватности результатов моделирования, эксперименты проводились в «ручном режиме». Причём, изменялось количество резервных СС только второго типа (ССР2) от 4 до 6 при изменениях количества мастеров-ремонтников от 3 до 5.

Таким образом, было проведено по 9 экспериментов в каждой системе моделирования. GPSS World позволяет проводить сразу все эти девять экспериментов, для чего должен быть написан соответствующий сегмент изменения версий модели. Что и было сделано. В AnyLogic вручную изменялись соответствующие данные, после чего запускалась модель.

Результаты экспериментов представлены в табл. 5.6. Из их сравнения следует, что они адекватны, поскольку отличия незначительны и составляют в основном 0…0,001, 0…0,002.

Что касается выбора оптимального сочетания количества резервных ССР2 и мастеров-ремонтников для условий данных экспериментов, то можно выбрать вариант 7: ССР2 = 4, мастеров-ремонтников = 5.

Таблица 5.6

Показатели функционирования системы связи

Показатели	GPSS World					AnyLogic				
	Типы средств связи									
	СС1	СС2	СС3	СС4	СС5	СС1	СС2	СС3	СС4	СС5
Вариант 1: ССР2 = 4, мастеров-ремонтников = 3										
Коэффициент прибыли по типам СС	0,374	0,327	0,627	0,400	0,573	0,380	0,332	0,633	0,405	0,576
Суммарный коэффициент прибыли	0,460					0,468				
Коэффициент использования по типам СС	0,779	0,741	0,853	0,780	0,846	0,781	0,743	0,856	0,781	0,847
Суммарный коэффициент использования СС	0,799					0,800				
Вариант 2: ССР2 = 5, мастеров-ремонтников = 3										
Коэффициент прибыли по типам СС	0,377	0,336	0,626	0,397	0,571	0,369	0,332	0,624	0,393	0,570
Суммарный коэффициент прибыли	0,462					0,462				
Коэффициент использования по типам СС	0,779	0,749	0,852	0,778	0,845	0,777	0,747	0,851	0,776	0,844
Суммарный коэффициент использования СС	0,801					0,800				
Вариант 3: ССР2 = 6, мастеров-ремонтников = 3										
Коэффициент прибыли по типам СС	0,364	0,330	0,619	0,385	0,564	0,361	0,331	0,620	0,390	0,563
Суммарный коэффициент прибыли	0,453					0,457				
Коэффициент использования по типам СС	0,774	0,751	0,849	0,773	0,842	0,774	0,751	0,850	0,775	0,841
Суммарный коэффициент использования СС	0,797					0,797				

Продолжение табл. 5.6

Показатели функционирования системы связи

Показатели	GPSS World					AnyLogic				
	Типы средств связи									
	СС1	СС2	СС3	СС4	СС5	СС1	СС2	СС3	СС4	СС5
Вариант 4: ССР2 = 4, мастеров-ремонтников = 4										
Коэффициент прибыли по типам СС	0,890	0,895	0,931	0,890	0,911	0,889	0,892	0,931	0,889	0,910
Суммарный коэффициент прибыли	0,903					0,905				
Коэффициент использования по типам СС	0,978	0,977	0,996	0,988	0,995	0,978	0,977	0,996	0,988	0,994
Суммарный коэффициент использования СС	0,987					0,987				
Вариант 5: ССР2 = 5, мастеров-ремонтников = 4										
Коэффициент прибыли по типам СС	0,887	0,892	0,930	0,887	0,909	0,885	0,890	0,930	0,886	0,909
Суммарный коэффициент прибыли	0,901					0,902				
Коэффициент использования по типам СС	0,977	0,981	0,996	0,988	0,994	0,976	0,980	0,995	0,987	0,994
Суммарный коэффициент использования СС	0,987					0,986				
Вариант 6: ССР2 = 6, мастеров-ремонтников = 4										
Коэффициент прибыли по типам СС	0,885	0,890	0,930	0,886	0,909	0,887	0,890	0,930	0,887	0,910
Суммарный коэффициент прибыли	0,900					0,903				
Коэффициент использования по типам СС	0,977	0,984	0,995	0,987	0,994	0,977	0,984	0,996	0,988	0,994
Суммарный коэффициент использования СС	0,987					0,987				

Окончание табл. 5.6

Показатели функционирования системы связи

Показатели	GPSS World					AnyLogic				
	Типы средств связи									
	СС1	СС2	СС3	СС4	СС5	СС1	СС2	СС3	СС4	СС5
Вариант 7: ССР2 = 4, мастеров-ремонтников = 5										
Коэффициент прибыли по типам СС	0,935	0,944	0,939	0,914	0,922	0,936	0,945	0,940	0,915	0,923
Суммарный коэффициент прибыли	0,931					0,934				
Коэффициент использования по типам СС	0,996	0,998	1,000	0,999	1,000	0,996	0,998	1,000	0,999	1,000
Суммарный коэффициент использования СС	0,998					0,998				
Вариант 8: ССР2 = 5, мастеров-ремонтников = 5										
Коэффициент прибыли по типам СС	0,936	0,936	0,939	0,915	0,922	0,936	0,936	0,940	0,915	0,923
Суммарный коэффициент прибыли	0,930					0,932				
Коэффициент использования по типам СС	0,996	0,999	1,000	0,999	1,000	0,996	0,999	1,000	0,999	1,000
Суммарный коэффициент использования СС	0,998					0,998				
Вариант 9: ССР2 = 6, мастеров-ремонтников = 5										
Коэффициент прибыли по типам СС	0,936	0,927	0,939	0,915	0,922	0,936	0,928	0,940	0,915	0,923
Суммарный коэффициент прибыли	0,928					0,929				
Коэффициент использования по типам СС	0,996	0,999	1,000	0,999	1,000	0,996	0,999	1,000	0,999	1,000
Суммарный коэффициент использования СС	0,998					0,998				

ГЛАВА 6. МОДЕЛЬ ФУНКЦИОНИРОВАНИЯ ПРЕДПРИЯТИЯ

6.1. 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 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 вероятностью q_3 % могут быть забракованы m блоков. Вероятности порядковых номеров из $1 ... n_1$ $P_{бл1} ... P_{блn_1}$ соответственно.

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 поступает в отдел приёмки. Время приемки $T_{пр}$ одного изделия случайное. Стоимость приемки одного изделия $C_п$. По результатам приёмки бра-

кується q_4 % изделий, которые направляются вновь на стенд выходного контроля. Принятые приёмкой изделия направляются на склад предприятия.

6.1.1. Исходные данные

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

$C_{k1} = 35; C_{k2} = 32; C_{k3} = 43; C_{k4} = 48;$

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

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

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

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

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

$m = 1; P_{k1} = 1,0; P_{бл1} = 0,25; P_{бл2} = 0,25; P_{бл3} = 0,25; P_{бл4} = 0,25;$

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

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

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

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

6.1.2. 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.

6.1.3. Уяснение задачи на исследование

Предприятие при изготовлении блоков и сборки из них изделий может быть представлено как многофазная многоканальная разомкнутая система массового обслуживания с ожиданием, так как оно имеет все её элементы (рис. 6.1):

поток изготовленных цехами блоков;

очереди блоков на посты контроля и пункты сборки;

очереди изделий на стенды контроля и пункт приемки;

многоканальные устройства обслуживания (посты контроля, стенды выходного контроля, пункты сборки, пункты приёмки);

потоки забракованных блоков;

выходные потоки готовых изделий.

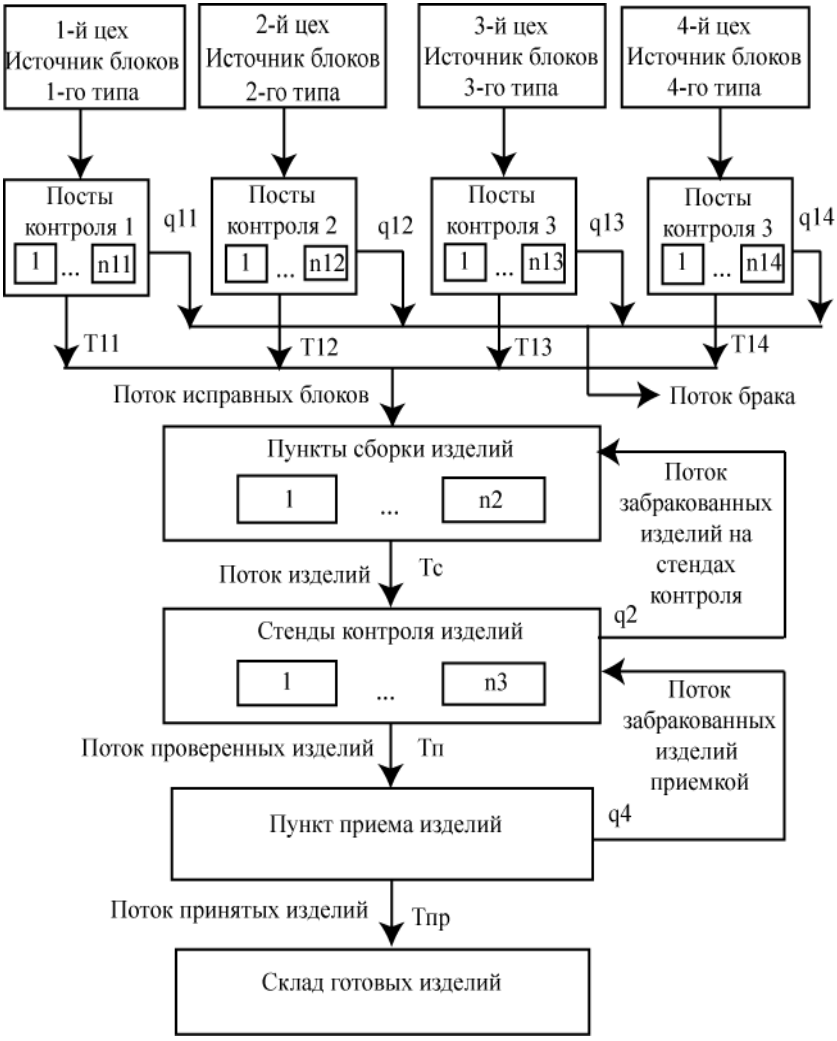


Рис. 6.1. Предприятие как система массового обслуживания

6.2. Модель в AnyLogic

6.2.1. Формализованное описание

Модель, исходя из структуры предприятия (см. рис. 6.1), 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, стендах контроля изделий и на пунктах приема изделий;

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

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

`cost` — стоимость готового изделия.

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.

Стоимость изготовления изделия может возрасти по сравнению с минимальной стоимостью (`CMIN`) за счёт выявления брака и замены блоков. Для фиксации этой стоимости введено поле `cost`.

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

`aveTimeShop1`…`aveTimeShop4` — средние интервалы 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 изделий;

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;

`minCostIzd` — минимальная стоимость одного изделия;

`минСтоимГотИзд` — минимальная стоимость готовых изделий;

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

На текущее модельное время собирается статистика по следующим стоимостным показателям функционирования предприятия:

`costKomplBlock1…costKomplBlock4,costKomplBlock` — стоимости комплектующих блоков 1…4 и суммарная стоимость комплектующих всех блоков;

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

`CostBlock1…CostBlock4` — стоимости изготовления блоков 1…4 с учётом тестирования;

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

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

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

`стоимБракБл` — суммарные затраты на все забракованные блоки;

`стоимГотИзд` — затраты на выпуск готовых изделий;

`costBlockIzd` — стоимость блоков одного изделия;

`costIzd` — стоимость одного изделия;

`времяИзгИзд` — среднее время изготовления одного изделия;

`коэфУвелСтоимИзд` — коэффициент увеличения себестоимости изделия.

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

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

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

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

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

Собирается также статистика о коэффициентах загрузки подразделений предприятия.

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

$$K_{\text{с}} = C_{\max} / C_{\min}.$$

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

$$costBlock1=stKomplBlock1+stIzqBlock1+stTestBlock1;$$
$$costBlock2=stKomplBlock2+stIzqBlock2+stTestBlock2;$$
$$costBlock3=stKomplBlock3+stIzqBlock3+stTestBlock3;$$
$$costBlock4=stKomplBlock4+stIzqBlock4+stTestBlock4;$$

$$costBlockIzd=costBlock1+costBlock2+costBlock3+costBlock4;$$

$$C_{\min\,изд} = costBlockIzd + stSborki + stKontrIzd + stPriemIzd;$$

$$C_{\min} = C_{\min\,изд} \cdot колГотИзд; \quad C_{\max} = стоимГотИзд+стоимБракБл.$$

$$K_c = коэфУвелСтоимИзд = \frac{C_{\max}}{C_{\min}}.$$

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

6.2.2. Ввод исходных данных

Для ввода исходных данных используем элемент **Параметр**.

1. Выполните команду **Файл/Создать/Модель** на панели инструментов.
2. В поле **Имя модели** диалогового окна **Новая модель** введите `Предприятие`.
3. Выберите каталог, в котором будут сохранены файлы модели. Щёлкните кнопку **Готово**.
4. В **Палитре** выделите **Презентация**. Создайте область просмотра `Данные` для размещения элементов исходных данных.
5. Перетащите элемент **Область просмотра**. В поле **Имя:** введите `Данные`.
6. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `0`, **Y:** `1700`, **Ширина:** `760`, **Высота:** `330`.
7. Перетащите элемент **Прямоугольник**. Оставьте имя, предложенное системой.
8. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `10`, **Y:** `1750`, **Ширина:** `740`, **Высота:** `270`.
9. Перетащите элемент **text** и на странице **Основные** панели **Свойства** в поле **Текст:** вместо слова text введите `Исходные данные`.
10. В **Палитре** выделите **Основная**. Перетащите элементы **Параметр** на элемент с именем `Исходные данные`. Разместите

их так, как показано на рис. 6.2. Значения свойств установите согласно табл. 6.1.

Таблица 6.1

Элементы и их свойства					
Параметр			Параметр		
Имя	Тип	Значение по умолчанию	Имя	Тип	Значение по умолчанию
aveTimeShop1	double	19	stKomplBlock1	double	35
aveTimeShop2	double	11	stKomplBlock2	double	32
aveTimeShop3	double	15	stKomplBlock3	double	43
aveTimeShop4	double	18	stKomplBlock4	double	48
stIzgBlock1	double	35	timeTestBlock1	double	12
stIzgBlock2	double	27	timeTestBlock2	double	16
stIzgBlock3	double	36	timeTestBlock3	double	21
stIzgBlock4	double	37	timeTestBlock4	double	17
postKontr1	int	2	procBrakBlock1	double	0.02
postKontr2	int	2	procBrakBlock2	double	0.03
postKontr3	int	2	procBrakBlock3	double	0.04
postKontr4	int	2	procBrakBlock4	double	0.06
stTestBlock1	double	12	kolStendKontrIzd	int	2
stTestBlock2	double	23	timeKontrIzd	double	26
stTestBlock3	double	32	procBrakIzd	double	0.05
stTestBlock4	double	28	stKontrIzd	double	74
kolPunSborki	int	2	verBlock1	double	0.9999
timeSborki	double	22	verBlock2	double	0.999999
stSborki	double	67			
verBlockNum1	double	0.25	timeZamBlock1	double	12
verBlockNum2	double	0.25	timeZamBlock2	double	15
verBlockNum3	double	0.25	timeZamBlock3	double	12
verBlockNum4	double	0.25	timeZamBlock4	double	21
stZamBlock1	double	34	kolPunPriem	int	2
stZamBlock2	double	46	timePriemIzd	double	18
stZamBlock3	double	38	procBrakPriem	double	0.15
stZamBlock4	double	54	stPriemIzd	double	53

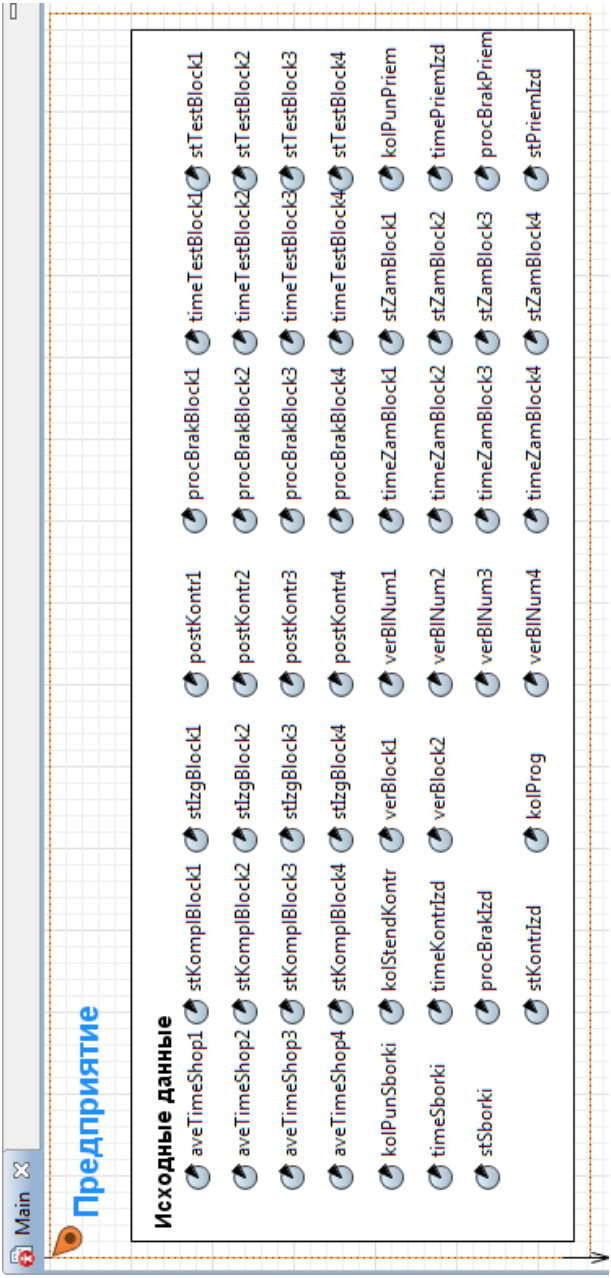


Рис. 6.2. Размещение элементов **Параметр** для ввода исходных данных

6.2.3. Вывод результатов моделирования

Для вывода результатов моделирования используем элемент **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. Создайте область просмотра `Результаты` для размещения элементов **Переменная**.
2. Перетащите элемент **Область просмотра**. В поле **Имя:** введите `Результаты`.
3. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `0`, **Y:** `700`, **Ширина:** `700`, **Высота:** `520`.
4. Перетащите элемент **Скруглённый прямоугольник**. Оставьте имя, предложенное системой.
5. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `10`, **Y:** `750`, **Ширина:** 690, **Высота:** `450`.
6. Перетащите элемент **text** и на панели **Свойства** в поле **Текст:** введите `Данные о количестве подготовленных и забракованных блоков и изделий`.
7. Перетащите ещё элемент **text** и введите `Стоимостные показатели функционирования предприятия`.
8. Перетащите третий элемент **text** и в поле **Текст:** введите `Показатели использования пунктов предприятия`.
9. В **Палитре** выделите **Основная**. Перетащите элементы **Переменная**. Используйте копирование. Разместите их и дайте им имена так, как показано на рис. 6.3. 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 и показателей использования его пунктов — `double`.
10. Выровняйте элементы. Для этого нужно вначале выделить те элементы, которые вы хотите выровнять, а затем открыть правым щелчком мыши по выделенным фигурам контекстное меню и выбрать нужную команду выравнивания из подменю **Выравнивание**. Не поддерживается выравнивание элементов диаграмм состояний и диаграмм действий, потому что при выравнивании таких элементов может нарушиться логика модели.

Рис. 6.3. Размещение элементов **Переменная** для вывода результатов моделирования

6.2.4. Построение событийной части модели

В событийную часть модели включим указанные ранее сегменты согласно представлению предприятия как системы массового обслуживания (см. рис. 6.1).

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 активного объекта элемент *область просмотра*. В дальнейщем после построения всех сегментов событийной части модели организуем переключение между областями просмотра.

Создайте область просмотра для размещения сегментов событийной части модели на диаграмме класса Main.

1. В **Палитре** выделите **Презентация**. Перетащите элемент **Область просмотра**.
2. Перейдите на страницу **Основные панели** Свойства.
3. В поле **Имя:** введите `Mainview`.
4. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `0`, **Y:** `0`, **Ширина:** `990`, **Высота:** `390`.
5. Перетащите элемент **Скруглённый прямоугольник**. В нём мы разместим все сегменты модели. Оставьте имя, предложенное системой.
6. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `10`, **Y:** `60`, **Ширина:** 970, **Высота:** `320`.
7. Перетащите элементы **Скруглённый прямоугольник**, укажите имена сегментов и свойства согласно табл. 6.2. Шрифт, цвет и т.д. выберите по своему усмотрению.

Таблица 6.2

Сегмент	Свойства			
	X:	**Y:**	**Ширина:**	**Высота:**
Цеха	20	80	140	270
Посты контроля блоков	170	80	160	270
Пункты сборки изделий	340	80	120	270
Стенды контроля изделий	470	80	150	270
Пункты приёма изделий	630	80	170	270
Склад готовых изделий	810	80	160	130
Склад бракованных изделий	810	220	160	130

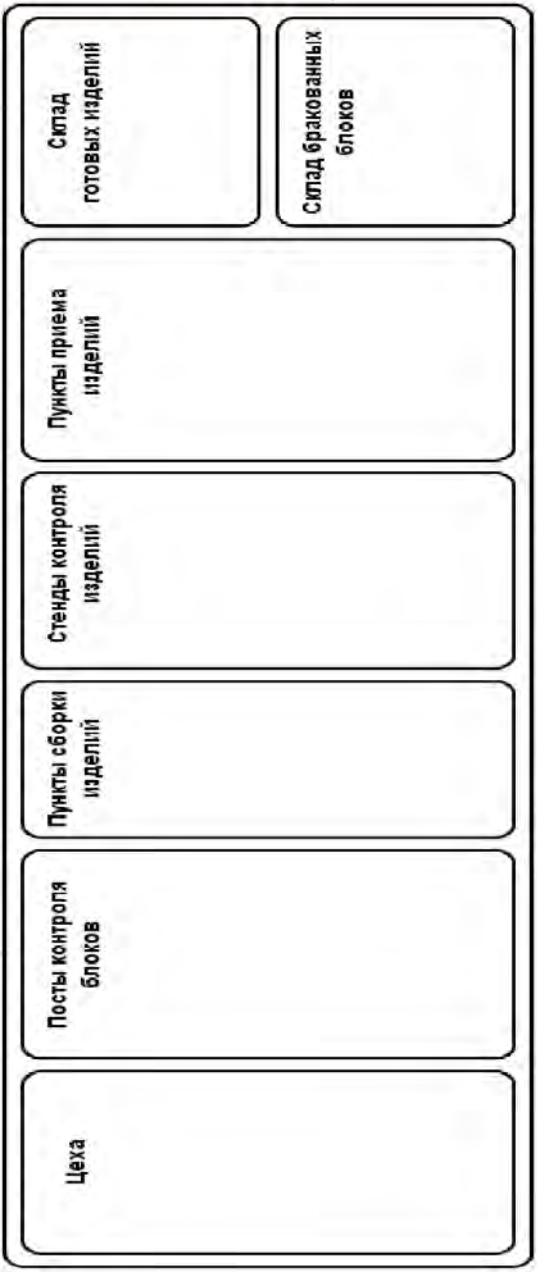


Рис. 6.4. Размещение сегментов модели

6.2.4.1. Имитация работы цехов предприятия

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

1. Из **Библиотеки моделирования процессов** перетащите объект **source** на агент Main и разместите в скругленном прямоугольнике с именем `Цеха`.
2. Для записи и хранения параметров блоков и изделий в дополнительные поля заявок нужно создать нестандартный тип заявки. Создайте тип заявки `Product`.
3. В панели **Проект** щёлкните правой кнопкой мыши элемент модели верхнего уровня дерева и выберите в меню **Создать Java класс**.
4. Появится диалоговое окно **Новый Java класс**. В поле **Имя:** введите имя нового класса `Product`.
5. В поле **Базовый класс:** выберите из выпадающего списка `Entity` в качестве базового класса. Щёлкните кнопку **Далее**.
6. 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;
double cost;
```

7. Оставьте выбранными флажки **Создать конструктор** и **Создать метод toString()**.
8. Щёлкните кнопку **Готово**. Появится редактор кода и автоматически созданный код вашего Java класса. Закройте код.
9. Щёлкните правой кнопкой мыши в панели **Проект** только что созданный Java класс и в контекстном меню выберите **Преобразовать Java класс в тип агента**.

10. Появится окно с параметрами типа заявок `Product`.

11. Выделите объект **source**. На странице **Основные** панели **Свойства** установите свойства согласно табл. 6.3.

Таблица 6.3

Свойства объектов source

Имя	Свойства	Значения

`Цех1`	Отображать имя
	Тип заявки
	Прибывают согласно
	Время между прибытиями
	Новая заявка
	Действия
	При выходе:
		Установите флажок
		`Product`
		Времени между прибытиями
		`exponential(1/aveTimeShop1)`
		`Product`
		`if (a==0){`
		`costBlock1=stKomplBlock1+`
		`stIzgBlock1+stTestBlock1;`
		`costBlock2=stKomplBlock2+`
		`stIzgBlock2+stTestBlock2;`
		`costBlock3=stKomplBlock3+`
		`stIzgBlock3+stTestBlock3;`
		`costBlock4=stKomplBlock4+`
		`stIzgBlock4+stTestBlock4;`
		`costBlockIzd=costBlock1+`
		`costBlock2+costBlock3+`
		`costBlock4;`
		`minCostIzd=costBlockIzd+`
		`stSborki+`
		`stKontrIzd+stPriemIzd;`
		`a=1;}`
		`kolIzgBlock1++;`
		`entity.numBlock = 1;`
		`entity.timeSbor =`
		`exponential(1/timeSborki);`
		`costKomplBlock1 +=`
		`stKomplBlock1;`
		`costKomplBlock +=`
		`stKomplBlock1;`
		`costIzgBlock1 +=`
		`stIzgBlock1;`
		`sumCostBlock1 +=`
		`(stKomplBlock1+stIzgBlock1);`
		`costIzgBlock += stIzgBlock1;`
		`sumCostBlock +=`
		`(stKomplBlock1+stIzgBlock1);`

Продолжение табл. 6.3

Имя	Свойства	Значения

цех2	Отображать имя
	Тип заявки
	Прибывают согласно
	Время между прибытиями
	Новая заявка
	Действия
	При выходе:
		Установите флажок
		Product
		Времени между прибытиями
		exponential(1/aveTimeShop2)
		Product
		
		kolIzgBlock2++;
		entity.numBlock = 2;
		costKomplBlock2 +=
		stKomplBlock2;
		costKomplBlock +=
		stKomplBlock2;
		costIzgBlock2 +=
		stIzgBlock2;
		sumCostBlock2 +=
		(stKomplBlock2+stIzgBlock2);
		costIzgBlock +=
		stIzgBlock2;
		sumCostBlock +=
		(stKomplBlock2+stIzgBlock2);

цех3	Отображать имя
	Тип заявки
	Прибывают согласно
	Время между прибытиями
	Новая заявка
	Действия
	При выходе:
		Установите флажок
		Product
		Времени между прибытиями
		exponential(1/aveTimeShop3)
		Product
		
		kolIzgBlock3++;
		entity.numBlock = 3;
		costKomplBlock3 +=
		stKomplBlock3;
		costKomplBlock +=
		stKomplBlock3;
		costIzgBlock3 +=
		stIzgBlock3;
		sumCostBlock3 +=
		(stKomplBlock3+stIzgBlock3);
		costIzgBlock +=
		stIzgBlock3;
		sumCostBlock +=
		(stKomplBlock3+stIzgBlock3);

Окончание табл. 6.3

Имя	Свойства	Значения

цех4	Отображать имя
	Тип заявки
	Прибывают согласно
	Время между прибытиями
	Новая заявка
	Действия
	При выходе:
		Установите флажок
		Product
		Времени между прибытиями
		exponential(1/aveTimeShop4)
		Product
		
		kolIzgBlock4++;
		entity.numBlock = 4;
		costKomplBlock4 +=
		stKomplBlock4;
		costKomplBlock +=
		stKomplBlock4;
		costIzgBlock4 +=
		stIzgBlock4;
		sumCostBlock4 +=
		(stKomplBlock4+stIzgBlock4);
		costIzgBlock +=
		stIzgBlock4;
		sumCostBlock +=
		(stKomplBlock4+stIzgBlock4);

12. Щёлкните выделенный **source** правой кнопкой мыши и в контекстном меню выберите **Копировать**.

13. Вставьте в скругленный прямоугольник с именем **Цеха** еще три объекта source. Разместите их вертикально один под другим. Во время вставки имена объектов будут изменяться: цех2, цех3, цех4. Однако остальные свойства останутся такими же, как и у объекта цех1.

14. Поэтому, последовательно выделяя второй, третий и четвертый объекты source, скорректируйте их свойства также согласно табл. 6.3.

6.2.4.2. 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 объектов, имитирующих работу постов контроля блоков, создайте новый тип агента **Тест**.

1. На панели **Проект** щёлкните Main правой кнопкой мыши и выберите из контекстного меню **Создать/Тип агента**. Откроется окно **Шаг 1. Создание нового типа агента**.

2. В поле **Имя:** задайте имя нового типа агента `Тест`.

3. Щёлкните кнопку **Готово**.

Создайте область просмотра на диаграмме агента **Тест** для размещения объектов сегмента **Посты контроля блоков**.

1. В **Палитре** выделите **Презентация**. Перетащите элемент **Область просмотра**.

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

3. В поле **Имя:** введите `Kontr1`.

4. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `0`, **Y:** `0`, **Ширина:** `590`, **Высота:** `390`.

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 экземпляр нового типа агента **Тест**.

1. Из Палитры **Основная** перетащите элемент **Порт** и разместите сверху в крайнем левом ряду (рис. 6.5).

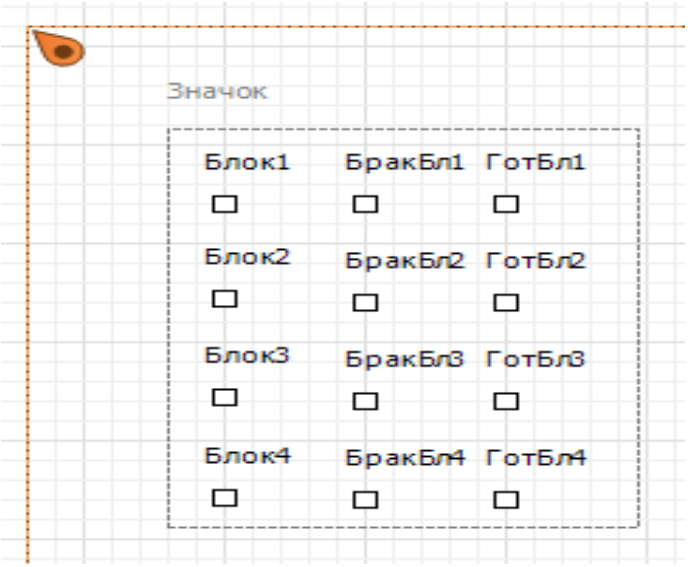


Рис. 6.5. Размещение элементов **Порт** на экземпляре агента **Тест**

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. 6.5). 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. 6.5).

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. 6.5). 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. 6.5).

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. 6.5). 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 агента Main.

15. На панели **Проект** выделите **Тест**, перетащите на диаграмму Main экземпляр агента Тест, разместите и соедините так, как на рис. 6.6. Порты БракБл1…БракБл4 соедините с объектом sink сегмента **Склад бракованных блоков**, предварительно разместив его там.

16. Экземпляр агента **Тест** создан. Возвратитесь на диаграмму агента **Тест**.

Для имитации работы постов контроля блоков одного типа (одного цеха) нам потребуются объекты:

queue — имитация склада изготовленных цехом блоков;

delay — имитация времени тестирования блока;

selectOutPut — имитация процесса браковки блоков.

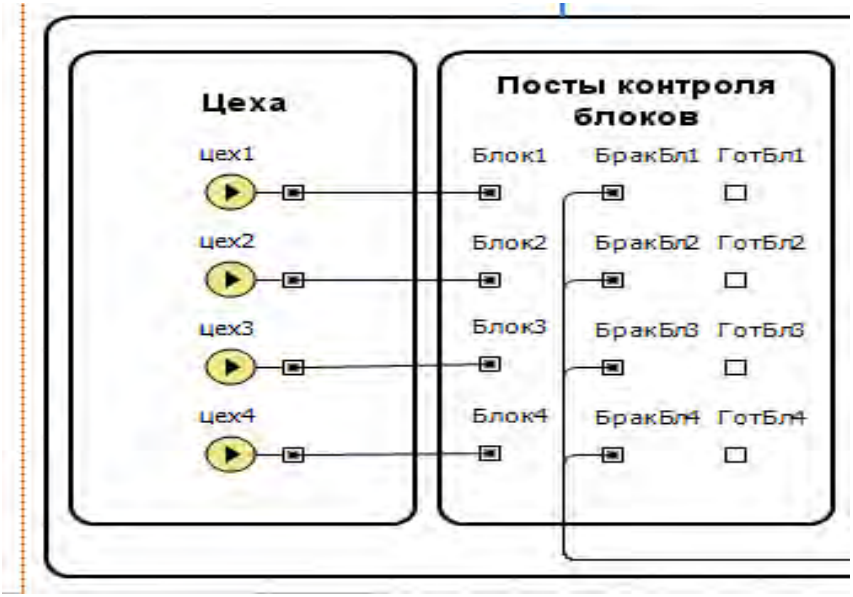


Рис. 6.6. Добавлен элемент диаграммы класса Test

1. Перетащите элемент **Скруглённый прямоугольник**. В нём мы разместим все объекты сегмента имитации работы постов контроля блоков. Оставьте имя, предложенное системой.

2. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `10`, **Y:** `50`, **Ширина:** 570, **Высота:** `330`.

Добавьте на диаграмму агента **Тест** объекты класса Queue.

1. Из **Библиотеки моделирования процессов** перетащите объект queue и разместите слева сверху, как на рис. 6.7.

2. На странице **Основные** панели **Свойства** замените имя queue именем `склИзгБл1` (склад изготовленных блоков цеха 1).

3. В поле **Тип заявки:** `Entity` замените `Product`.

4. Установите **Вместимость:** `Максимальная`.

5. Скопируйте объект с именем `склИзгБл1`.

6. Вставьте и поместите три объекта класса Queue (см. рис. 6.7).

Добавьте на диаграмму класса Test объекты класса Delay.

1. Из библиотеки Enterprise Library перетащите один объект delay и поместите справа рядом с объектом `склИзгБл1`, как на рис. 6.7.

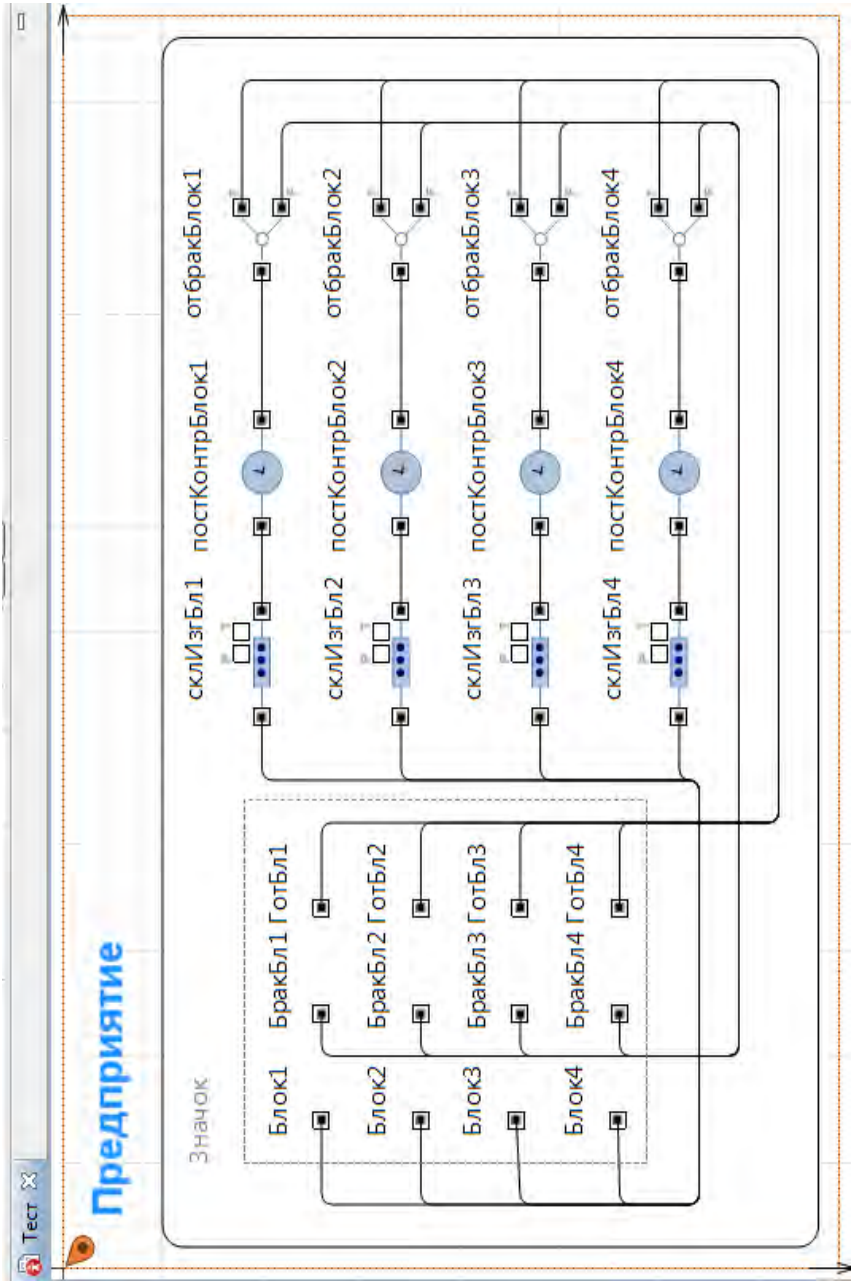


Рис. 6.7. Размещение элементов на диаграмме агента Тест

2. На странице **Основные** панели **Свойства** замените имя delay именем `постКонтрБлок1` (посты контроля блоков цеха 1).

3. В поле **Тип заявки:** `Agent` замените `Product`.

4. Введите в поле **Время задержки:** `exponential (1/main.timeTestBlock1)`.

5. Введите `main.postKontr1` в поле **Вместимость:**.

6. **Действия При выходе:**

```
main.kolTestBlock1++;
main.sumCostBlock1 += main.stTestBlock1;
main.costTestBlock1 += main.stTestBlock1;
main.costTestBlock += main.stTestBlock1;
main.sumCostBlock += main.stTestBlock1;
```

7. Скопируйте объект с именем `постКонтрБлок1`.

8. Вставьте и разместите три объекта delay (см. рис. 6.7).

9. Последовательно выделите и внесите правки в их свойства (табл. 6.4). Внесите правки и в свойство **Действия При выходе**.

Таблица 6.4

Имя	Время задержки	Вместимость
постКонтрБлок2	`exponential (1/`**`main`**`.timeTestBlock2)`	**`main.`**`postKontr2`
постКонтрБлок3	`exponential (1/`**`main`**`.timeTestBlock3)`	**`main.`**`postKontr3`
постКонтрБлок4	`exponential (1/`**`main`**`.timeTestBlock4)`	**`main.`**`postKontr4`

Добавьте на диаграмму **Тест** объекты класса SelectOutput.

1. Перетащите объект selectOutput и разместите справа рядом с объектом `постКонтрБлок1`, как на рис. 6.7.

2. На странице **Основные** панели **Свойства** замените имя selectOutPut именем `ОтбракБлок1` (отбраковка блоков цеха 1).

3. Установите свойства объекта согласно табл. 6.5.

4. Скопируйте объект с именем `ОтбракБлок1`.

5. Вставьте три объекта класса SelectOutput (см. рис. 6.7).

6. Последовательно выделите вставленные объекты и скорректируйте значения их свойств согласно табл. 6.5.

7. Соедините входы и выходы объектов диаграммы агента **Тест** согласно рис. 6.7.

Код в свойство **Действия При выходе (true)** введен для учёта, например, для цеха 1:

gotBlock1 — количества готовых блоков цеха 1.

Код в свойстве **Действия При выходе (false)** обеспечивает счёт количества brakBlock1 забракованных блоков цеха 1.

Таблица 6.5

Свойства	Значение

Имя
Тип заявки
Выход true выбирается
Вероятность:
Действия
При выходе (true)
Действия
При выходе (false)
	`ОтбракБлок1`
	`Product`
	Заданной вероятностью
	`1-`**`main`**`.procBrakBlock1`
	`main``.gotBlock1++;`
	`main``.brakBlock1++;`
	`entity.sign1 = 1;`
	`entity.numBlBrak1 = 1;`

Имя
Тип заявки
Выход true выбирается
Вероятность:
Действия
При выходе (true)
Действия
При выходе (false)
	`ОтбракБлок2`
	`Product`
	Заданной вероятностью
	`1-`**`main`**`.procBrakBlock2`
	`main``.gotBlock2++;`
	`main``.brakBlock2++;`
	`entity.sign1 = 1;`
	`entity.numBlBrak2 = 1;`

Имя
Тип заявки
Выход true выбирается
Вероятность:
Действия
При выходе (true)
Действия
При выходе (false)
	`ОтбракБлок3`
	`Product`
	Заданной вероятностью
	`1-`**`main`**`.procBrakBlock3`
	`main``.gotBlock3++;`
	`main``.brakBlock3++;`
	`entity.sign1 = 1;`
	`entity.numBlBrak3 = 1;`

Имя
Тип заявки
Выход true выбирается
Вероятность:
Действия
При выходе (true)
Действия
При выходе (false)
	`ОтбракБлок4`
	`Product`
	Заданной вероятностью
	`1-`**`main`**`.procBrakBlock4`
	`main``.gotBlock4++;`
	`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 … entity.numBlock4. Это нужно для раздельного счёта забракованных блоков.

Перейдите на диаграмму агента Main.

6.2.4.3. 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 объектов имитации работы пунктов сборки изделий создайте новый тип агента **Сборка**.

1. На панели **Проект** щёлкните Main правой кнопкой мыши и выберите из контекстного меню **Создать/Тип агента**.
2. Откроется окно **Шаг 1. Создание нового типа агента**.
3. В поле **Имя:** задайте имя `Сборка`.
4. Щёлкните кнопку **Готово**.

Создайте область просмотра на диаграмме агента **Сборка** для размещения объектов сегмента **Пункты сборки изделий**.

1. В **Палитре** выделите **Презентация**. Перетащите элемент **Область просмотра**.
2. Перейдите на страницу **Основные панели** Свойства.
3. В поле **Имя:** введите `Sbor`.
4. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `0`, **Y:** `0`, **Ширина:** `820`, **Высота:** `550`.
5. Перетащите элемент **Скруглённый прямоугольник**. В нём мы разместим все элементы сегмента. Оставьте имя, предложенное системой.
6. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `10`, **Y:** `60`, **Ширина:** 800, **Высота:** `480`.

Согласно назначению пунктов сборки изделий для связи их с другими сегментами модели необходимо иметь:

четыре порта для приёма готовых блоков с постов контроля;

порт — для отправки собранных изделий на стенды контроля;

порт для приёма бракованных изделий с целью определения и замены в них забракованных блоков;

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. 6.8).

2. На странице **Основные** панели **Свойства** имя port замените именем `ГотБлок1`.

3. Скопируйте элемент **Порт** с именем `ГотБлок1`.

4. Вставьте три элемента **Порт** (см. рис. 6.8). 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. 6.8.

6. Вернитесь на диаграмму агента Main.

7. На панели **Проект** выделите **Сборка**, перетащите на диаграмму агента Main экземпляр сборка, разместите и соедините так, как на рис. 6.9. Порт БракБл соедините с объектом sink сегмента **Склад бракованных блоков**.

8. Экземпляр агента **Сборка** создан. Возвратитесь на диаграмму агента **Сборка**.

Для реализации сегмента имитации работы пунктов сборки изделий нам потребуются объекты, но прежде мы рассмотрим вариант построения этого сегмента.

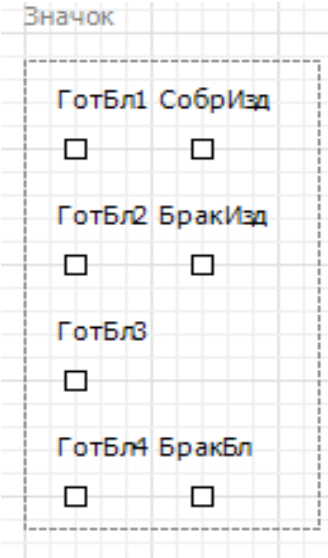


Рис. 6.8. Размещение элементов **Порт** на диагрпмме агента **Сборка**

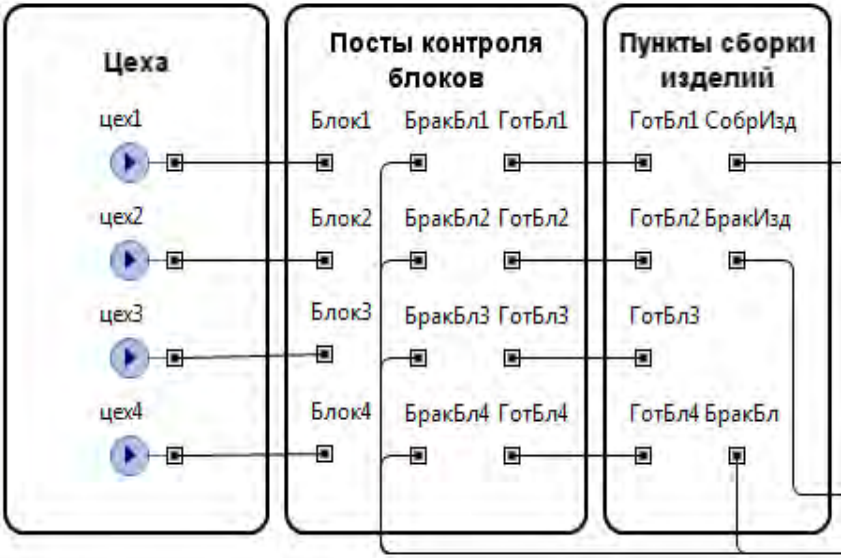


Рис. 6.9. Добавлен экземпляр агента **Сборка**

Сборка изделия начинается тогда, когда будут готовы четыре блока (по одному каждого из четырех типов). Полагаем, что время готовности блоков различное. Значит, нужно использовать такие объекты AnyLogic, которые предназначены для синхронизации движения заявок (блоков). Объекты класса Match предназначены именно для синхронизации движения двух заявок.

После готовности четырех блоков для дальнейщей имитации процесса сборки нужно эти четыре заявки объединить в одну, и интерпретировать её как изделие. В AnyLogic имеются объекты некоторых классов, которые позволяют осуществить нужное нам объединение. Применим объекты 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 объект queue.

Для имитации непосредственно процесса сборки следует взять объект delay, а для разделения потока изделий на собранные первично и на изделия с заменными блоками после браковки — объект selectOutPut.

Реализуйте часть сегмента имитации работы пунктов сборки.

1. Перетащите элемент **Прямоугольник**. Оставьте имя, предложенное системой.

2. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `320`, **Y:** `80`, **Ширина:** 340, **Высота:** `230`.

3. Из **Библиотеки моделирования процессов** перетащите четыре объекта match и три объекта combine. Расположите и соедините их так, как на рис. 6.10.

4. Перетащите объекты queue, delay, selectOutPut, разместите, соедините и дайте им имена также согласно рис. 6.10.

Поочередно выделите каждый из этих объектов и на страницах **Основные** панели **Свойства** установите для:

match…match3:

Типы заявок: Вход1, Вход2: `Product, Product`

Условия соответствия `true`

Максимальная вместимость 1

Максимальная вместимость 2

combine…combine2:

Типы заявок: Вход1, Вход2, Выход: `Product, Product, Product`

Объединенная заявка entity1

queue:

Тип заявки: `Product`

Максимальная вместимость

selectOutPut:

Тип заявки: `Product`

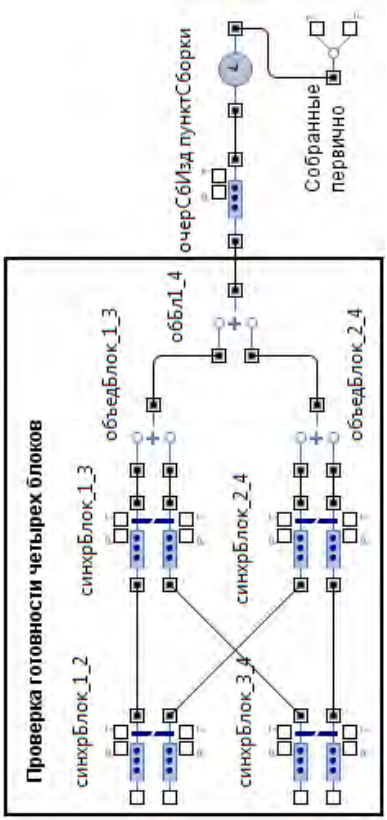
Выход true выбирается При выполнении условия

Условие `entity.sign1 == 0`

Объекты match и match1 обеспечивают синхронизацию движения блоков 1 и 2, 3 и 4 соответственно. А объекты match2 и match3 — блоков 1 и 3, 2 и 4 соответственно. Таким образом, обеспечивается синхронизация движения четырех блоков.

Указание свойства entity1 в объектах 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 неё ранее было определено значение поля `entity.timeSbor` (см. табл. 6.3).

Изделия после элемента `пунктСборки` разделяются на два потока: собранные изделия первично (`entity.sign1=0`) и изделия с заменёнными блоками (`entity.sign1=2`).



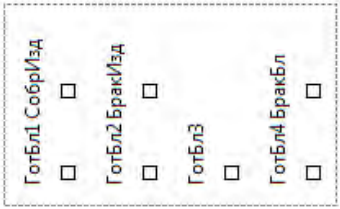


Рис. 6.10. Фрагмент сегмента имитации работы пункта сборки

Разделение, осуществляемое объектом selectOutPut по условию `entity.sign1==0,` нужно для учёта количества изделий с заменёнными блоками и количества заменённых блоков.

Свойства объекта delay установите согласно рис. табл. 6.6.

Таблица 6.6

Имя	Свойства	Значения

пунктборки	Отображать имя
	Тип заявки
	Задержка задаётся
	Время задержки
	Вместимость
	Действия
	При выходе
		Установите флажок
		Product
		Определённое время
		entity.timeSbor
		main.kolPunSborki
		
		if (entity.sign1 == 0)
		{main.kolSobrIzd++;
		main.costSborIzd +=
		main.stSborki;
		entity.cost += main.stSborki;}
		if (entity.sign1 == 2)
		{if (entity.numBlBrak1== 1)
		{main.zamBlock1++;
		entity.cost +=
		main.stZamBlock1;
		main.costSborIzd +=
		main.stZamBlock1;}
		if (entity.numBlBrak2 == 1)
		{main.zamBlock2++;
		entity.cost +=
		main.stZamBlock2;
		main.costSborIzd +=
		main.stZamBlock2;}
		if (entity.numBlBrak3 == 1)
		{main.zamBlock3++;
		entity.cost +=
		main.stZamBlock3;
		main.costSborIzd +=
		main.stZamBlock3;}
		if (entity.numBlBrak4 == 1)
		{main.zamBlock4++;
		entity.cost +=
		main.stZamBlock4;
		main.costSborIzd +=
		main.stZamBlock4;}
		main.kolZamIzd++;}

Продолжим построение сегмента имитации работы пунктов сборки изделий (рис. 6.11).

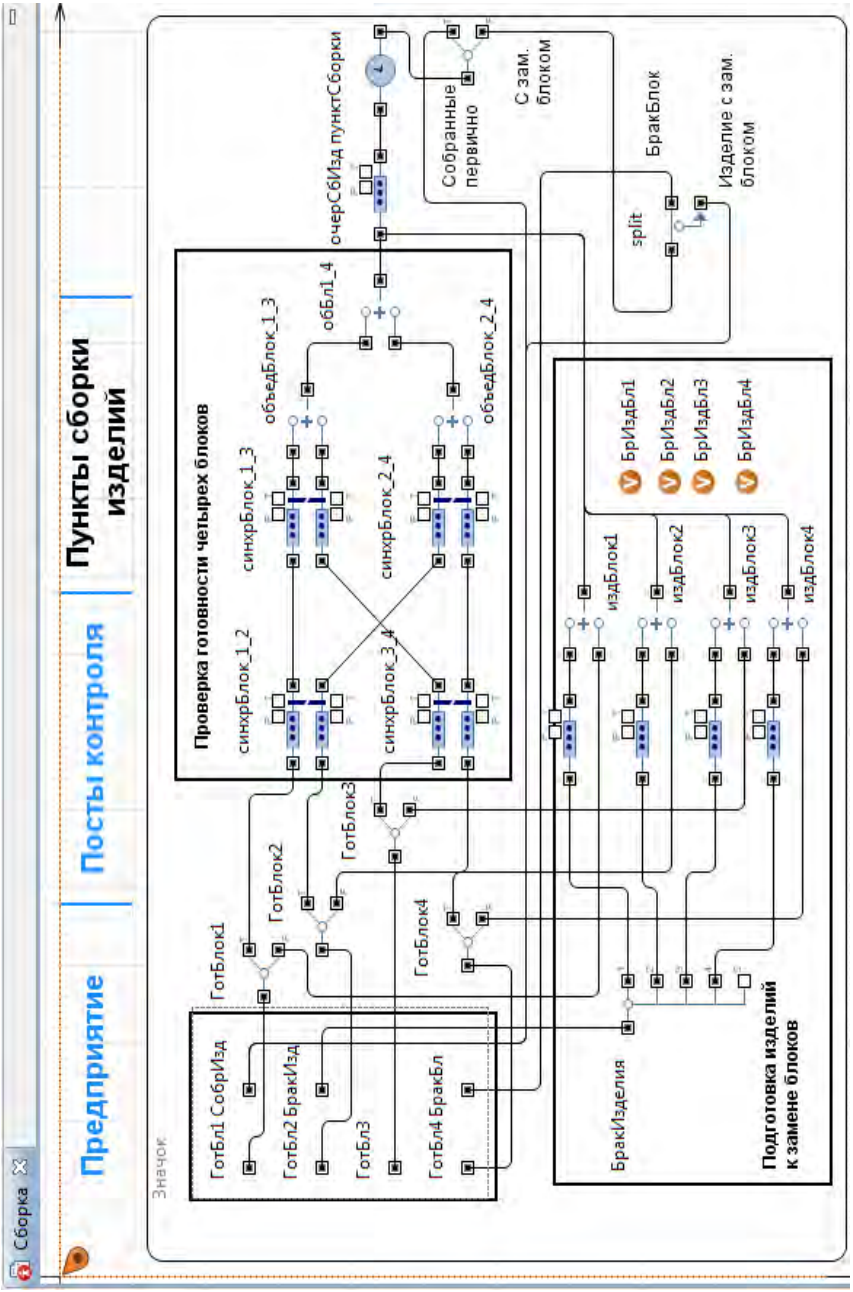


Рис. 6.11. Сегмент имитации работы пунктов сборки изделий

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 приёма изделий. Это мы сделаем (запишем) позже при построении этих сегментов. Для определения, какой из блоков 1 … 4 забракован в изделии, нужно использовать объект selectOutput5. Этот объект в отличие от объекта selectOutput позволяет проверять четыре условия и в зависимости от результата проверки направляет заявку на один из пяти выходов (можно создать объект selectOutput(**N**) на один вход и N выходов, использовав один объект exit и N объектов enter).

Готовые блоки поступают через порты ГотБл1 … ГотБл4. Поэтому их надо было бы соединить с соответствующими входами объектов 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 объектов selectOutPut. Каждый порт ГотБл1 … ГотБл4 соединить с входом соответствующего объекта selectOutPut. Выходы true этих объектов соединить с соответсвующими входами объектов 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 объектами combine. Выходы объекта selectOutput5 соединить с входами delay, а выход каждого из них — с первыми входами объектов combine.

С выходов объектов 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.

Собранные первично изделия с выхода true объекта selectOutPut направляются на стенды контроля, поэтому этот выход нужно соединить с портом `СобрИзд`.

Изделие с заменённым 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 объект split. Один выход этого объекта соедините с портом `БракБл`, а другой — с портом `СобрИзд`.

Выполните изложенные рекомендации практически. Перетащите четыре объекта класса SelectOutPut (или перетащите один объект, а остальные три скопируйте), один объект класса SelectOutPut5, четыре объекта классов Queue и Combine, один объект класса Split и четыре простых переменных. Разместите, дайте им имена и соедините так, как на рис. 6.11.

Поочередно выделите новые объекты и установите их свойства согласно табл. 6.7.

Таблица 6.7

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.numBlBrak2 == 1`
	`entity.numBlBrak3 == 1`
	`entity.numBlBrak4 == 1`

Продолжение табл. 6.7

Свойства	Значение
match	

Имя
Типы заявок: Вход1, Вход2:
Условие соответствия
Максимальная вместимость 1
Максимальная вместимость 2
	`синхрБлок_1_2`
	`Product, Product`
	`true`
	Установить флажок
	Установить флажок

Имя
Типы заявок: Вход1, Вход2:
Условие соответствия
Максимальная вместимость 1
Максимальная вместимость 2
	`синхрБлок_3_4`
	`Product, Product`
	`true`
	Установить флажок
	Установить флажок

Имя
Типы заявок: Вход1, Вход2:
Условие соответствия
Максимальная вместимость 1
Максимальная вместимость 2
	`синхрБлок_1_3`
	`Product, Product`
	`true`
	Установить флажок
	Установить флажок

Имя
Типы заявок: Вход1, Вход2:
Условие соответствия
Максимальная вместимость 1
Максимальная вместимость 2
	`синхрБлок_2_4`
	`Product, Product`
	`true`
	Установить флажок
	Установить флажок

combine	

Имя
Типы заявок: Вход1,
Вход2, Выход
Действие при входе 1
 	
Объединенная заявка
Действия При выходе
	`издБлок1`
	
	`Product, Product, Product`
	`if (entity.numBlBrak1 == 1)`
	`БрИздБл1 ++;`
	`entity1`
	`if (entity.numBlBrak1 == 1)`
	`БрИздБл1 --;`
	`entity.cost +=`
	`main.stZamBlock1;`
	`entity.timeSbor =`
	`exponential(1/main.`
	`timeZamBlock1);`

Имя
Типы заявок: Вход1,
Вход2, Выход
Действие при входе 1
	`издБлок2`
	
	`Product, Product, Product`
	`if (entity.numBlBrak2 == 1)`

Объединенная заявка **Действия При выходе**	БрИздБл2 ++; entity1 **if** (entity.numBlBrak2 == 1) БрИздБл2 --; entity.cost += **main**.stZamBlock2; entity.timeSbor = exponential(1/**main**. timeZamBlock2);
Имя Типы заявок: Вход1, Вход2, Выход Действие при входе 1 Объединенная заявка **Действия При выходе**	издБлок3 Product, Product, Product **if** (entity.numBlBrak3 == 1) БрИздБл3 ++; entity1 **if** (entity.numBlBrak3 == 1) БрИздБл3 --; entity.cost += **main**.stZamBlock3; entity.timeSbor = exponential(1/**main**. timeZamBlock3);
Имя Типы заявок: Вход1, Вход2, Выход Действие при входе 1 Объединенная заявка **Действия При выходе**	издБлок4 Product, Product, Product **if** (entity.numBlBrak4 == 1) БрИздБл4 ++; entity1 **if** (entity.numBlBrak4 == 1) БрИздБл4 --; entity.cost += **main**.stZamBlock4; entity.timeSbor = exponential(1/**main**. timeZamBlock4);
split	
Типы заявок: Количество копий Новая заявка (копия) **Действия При выходе** копии	Product, Product 1 Product entity.sign1 = 0; entity.cost = original.cost;

6.2.4.4. Имитация работы стендов контроля изделий

Стенды контроля изделий предназначены для приёма первично собранных изделий и изделий после замены забракованных блоков, непосредственно процесса контроля изделий, отправки прошедших контроль изделий на пункты приёма, забракованных изделий — на пункты сборки, а также для приёма забракованных изделий с пунктов приёма изделий.

На стендах контроля изделий, вследствие недостатка ресурсов, будет создаваться очередь, для имитации которой используйте объект queue.

Для имитации непосредственно процесса контроля изделий, то есть затрат времени на его проведение, используйте объект delay.

По результатам контроля некоторые изделия будут признаны браком. Для отбраковки изделий нужно применить объект selectOutput.

1. Перетащите объекты queue, delay и selectOutput на прямоугольник с именем **Стенды контроля изделий** диаграммы Main.
2. Разместите и соедините их согласно рис. 6.12.
3. Выделите объект queue и установите на странице **Основные** панели **Свойства**:

Имя: `очерСтенКонтр`

Тип заявки: `Product`

Максимальная вместимость Установите флажок

4. Выделите объект delay и установите его свойства:

Тип заявки: `Product`

Тип `Определённое время`

Время задержки `exponential( 1/timeKontrIzd )`

Вместимость `kolStendKontr`

Действия При выходе:

```
testSobrIzd++;
entity.cost += stKontrIzd;
entity.numBlBrak1 = 0;
entity.numBlBrak2 = 0;
entity.numBlBrak3 = 0;
entity.numBlBrak4 = 0;
```

Замечание. Объект `delay2` с нулевым временем задержки. Если его не включить, объект БракИзделия на диаграмме агента **Сборка** работать согласно замыслу разработчика не будет.

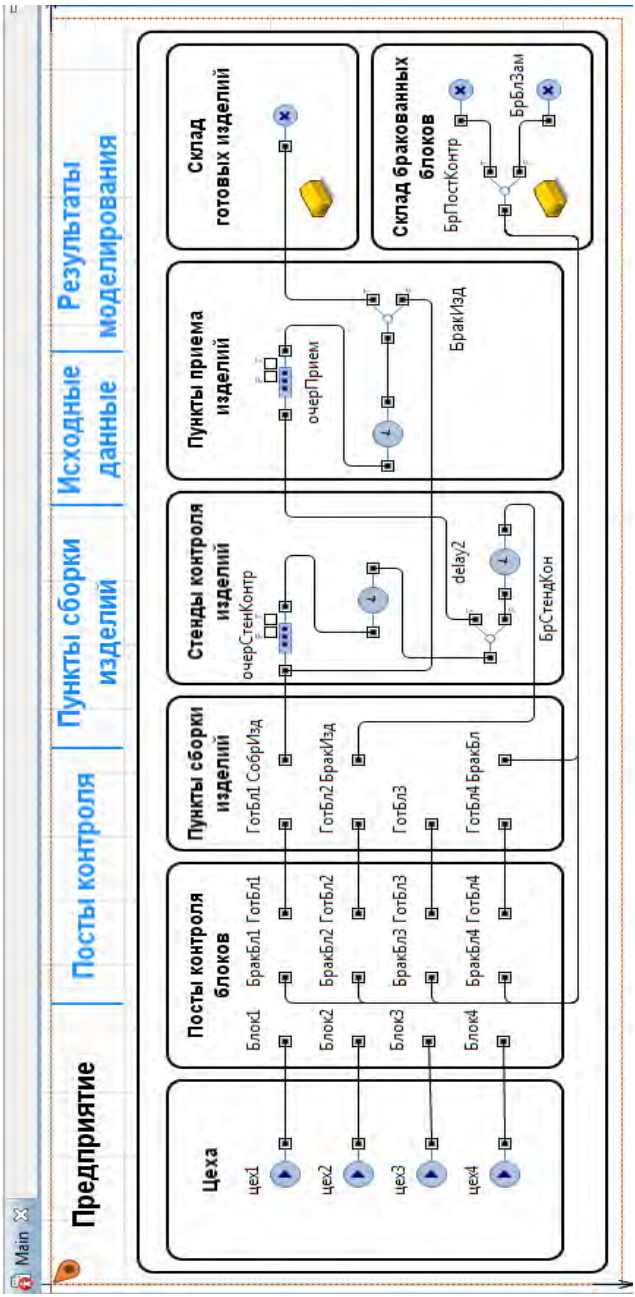


Рис. 6.12. Диаграмма агента Main с элементами всех сегментов

5. Выделите объект selectOutput и установите его свойства:

Имя: `БрСтендКон`

Тип заявки: `Product`

Выход true выбирается `С заданной вероятностью`

Вероятность[0..1] `1-procBrakIzd`

Действия При выходе (true) `costTestIzd += stKontrIzd;`

Действия При выходе (false)

```
double a = 0;
int numBlock = 0;
entity.sign1 = 2;
a = uniform();
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 **Действия При выходе (false)** объекта 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.

6.2.4.5. Имитация работы пунктов приёма изделий

Пункты приёма изделий предназначены для приёма прошедших стенды контроля изделий, непосредственно приёма изделий, отправки прошедших приём изделий на склад готовых изделий, а забракованных изделий — на стенды контроля.

На пунктах приёма будет создаваться очередь, для имитации которой используйте объект queue.

Для имитации непосредственно процесса приёма изделий используйте элемент delay.

По результатам контроля некоторые изделия будут признаны браком. Для отбраковки воспользуйтесь объектом selectOutput.

1. Перетащите объекты queue, delay и selectOutput на прямоугольник с **Пункты приема изделий** диаграммы класса Main.
2. Разместите и соедините их согласно рис. 6.12.
3. Выделите объект queue и установите на странице **Основные** панели **Свойства**:

Имя: `очерПрием`

Тип заявки: `Product`

Максимальная вместимость Установите флажок

4. Выделите объект delay и установите его свойства:

Тип заявки: `Product`

Тип `Определённое время`

Время задержки `exponential( 1/timePriemIzd )`

Вместимость `kolPunPriem`

Действия При выходе

```
kolPriemIzd++;
costPriemkiIzd += stPriemIzd;
entity.cost += stPriemIzd;
```

5. Выделите объект selectOutput и установите его свойства:

Тип заявки: `Product`

Выход true выбирается `С заданной вероятностью`

Вероятность[0..1] `1-procBrakPriem`

Действия При выходе (false)

```
entity.sign1 = 2;
brakPriemIzd++;
```

Код свойства **Действия При выходе** `объекта` delay введен для счета количества `kolPriemIzd` принятых всего изделий.

Код свойства **Действия При выходе (false)** объекта selectOutput считает количество `brakPriemIzd` забракованных изделий и полю `entity.sign1` присваивает 2 — признак брака в изделии.

6.2.4.6. Имитация склада готовых изделий

Для имитации склада готовых изделий используйте объект sink со следующими свойствами:

Имя: `ГотИзделия`

Тип заявки: `Product`

Действия При выходе:

```
колГотИзд ++;
времяИзгИзд = time()/колГотИзд;
стоимГотИзд += entity.cost;
минСтоимГотИзд +=minCostIzd;
коэфУвелСтоимИзд =
(стоимГотИзд+стоимБракБл)/минСтоимГотИзд;
ostGotBlock1 = gotBlock1 - колГотИзд - zamBlock1;
ostGotBlock2 = gotBlock2 - колГотИзд - zamBlock2;
ostGotBlock3 = gotBlock3 - колГотИзд - zamBlock3;
ostGotBlock4 = gotBlock4 - колГотИзд - zamBlock4;
```

Код свойства **Действия При выходе** введен для счёта количества `kolGotIzd` готовых изделий, их стоимости `cost-GotIzd` и коэффициента `koefIncrCostIzd` увеличения себестоимости изделия вследствие наличия брака.

Кроме того, ведется счёт готовых для сборки блоков, то есть изготовленных цехами и проверенных на постах контроля, но не использованных для сборки изделий блоков по типам `ostGot-Block1… ostGotBlock4` на текущее модельное время.

6.2.4.7. Имитация склада бракованных блоков

Ранее (см. п. 6.2.4.2) для имитации склада бракованных блоков было рекомендовано использовать один объект sink.

Предположим, что требуется вести раздельный учёт забракованных блоков на постах контроля цехов и забракованных на стендах контроля и пунктах приёма изделий. Для разделения потока брака дополнительному полю `entity.sign1` присваивались признаки 1 или 2.

1. Перетащите объекты selectOutput и sink на прямоугольник с именем **Склад бракованных блоков** диаграммы агента Main.
2. Разместите и соедините их согласно рис. 6.12.
3. Поочередно выделите объекты имитации склада бракованных блоков, начиная с объекта selectOutput, и установите свойства согласно табл. 6.8.

Коды в свойстве **Действия При выходе:** обоих объектов sink одинаковые. Они предназначены для счёта:

стоимости `costzbrakBlock` забракованных блоков не по типам, а в целом всех блоков;

количества забракованных блоков по типам `allBrakBlock1… allBrakBlock4`.

4. Из палитры **Картинки** перетащите на сегменты имитации складов готовых изделий и бракованных блоков картинки **Склад**.

Таблица 6.8

Свойства	Значение
selectOutput	
Тип заявки: Выход true выбирается Условие	Product При выполнении условия entity.sign1 == 1
sink	
Имя Тип заявки: **Действия** **При выходе:**	БрПостКонтр Product **if** (entity.numBlBrak1 == 1) { стоимБракБл += costBlock1; allBrakBlock1++;} **if** (entity.numBlBrak2 == 1) { стоимБракБл += costBlock2 ; allBrakBlock2++;} **if** (entity.numBlBrak3 == 1) { стоимБракБл += costBlock3; allBrakBlock3++;} **if** (entity.numBlBrak4 == 1) { стоимБракБл += costBlock4; allBrakBlock4++;}
Имя Тип заявки: **Действия** **При выходе:**	БрБлЗам Product **if** (entity.numBlBrak1 == 1) { стоимБракБл += costBlock1; allBrakBlock1++;} **if** (entity.numBlBrak2 == 1) { стоимБракБл += costBlock2; allBrakBlock2++;} **if** (entity.numBlBrak3 == 1) { стоимБракБл += costBlock3; allBrakBlock3++;} **if** (entity.numBlBrak4 == 1) { стоимБракБл += costBlock4; allBrakBlock4++;}

6.2.4.8. Организация перек между областями просмотра

Добавьте свои собственные элементы презентации на ранее созданные области просмотра Mainview, Data, Result, Kont1 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.

Для этого разместите на каждой области просмотра элементы презентации, показанные на рис. 6.13.

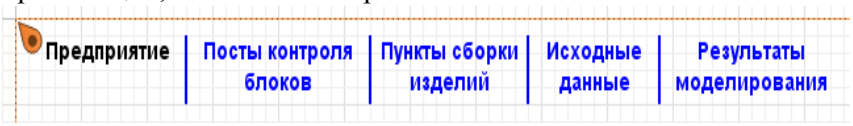


Рис. 6.13. Элементы презентации для переключения

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. 6.13). На панели **Свойства** установите в поле **Цвет:** black, выберите выравнивание текста и шрифт.

2. Перетащите второй элемент **text**, разместите и введите в поле **Текст:** `Посты контроля блоков`. На панели **Свойства** установите в поле **Цвет:** blue, выберите выравнивание текста и шрифт. Раскройте **Специфические** и в поле **Действие по щелчку:** введите код: `тест.Kontr1.navigateTo();`

Замечание. При выборе цветов 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`. Раскройте **Специфические**. В поле **Действие по щелчку:** введите: `сборка.Sbor.navigateTo();`

4. Перетащите четвертый элемент **text**. Разместите и введите в поле **Текст:** `Исходные данные`. В поле **Действие по щелчку:** на странице **Специфические** введите: `Данные.navigateTo();`

5. Перетащите пятый элемент **text**, разместите и введите в поле **Текст:** `Результаты моделирования`. На панели **Свойства** выделите **Специфические** и в поле **Действие по щелчку:** введите: `Результаты.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 согласно рис. 6.13. Шрифт, цвет и его размеры установите по своему усмотрению.

2. Последовательно переходите на остальные области просмотра, вставьте скопированные элементы презентации и внесите правки в их свойства согласно табл. 6.9.

Таблица 6.9

Элемент презентации	Действие по щелчку
Data	
Предприятие	`Mainview.navigateTo();`
Посты контроля блоков	`тест.Kontr1.navigateTo();`
Пункты сборки изделий	`сборка.Sbor.navigateTo();`
Исходные данные	
Результаты моделирования	`Результаты.navigateTo();`
Result	
Предприятие	`Mainview.navigateTo();`
Посты контроля блоков	`тест.Kontr1.navigateTo();`
Пункты сборки изделий	`сборка.Sbor.navigateTo();`
Исходные данные	`Данные.navigateTo();`
Результаты моделирования	
Kontrol	
Предприятие	`main.Mainview.navigateTo();`
Посты контроля блоков	
Пункты сборки изделий	`main.сборка.Sbor.navigateTo();`
Исходные данные	`main.Данные.navigateTo();`
Результаты моделирования	`main.Результаты.navigateTo();`
Sbor	
Предприятие	`main.Mainview.navigateTo();`
Посты контроля блоков	`main.тест.Kontr1.navigateTo();`
Пункты сборки изделий	
Исходные данные	`main.Данные.navigateTo();`
Результаты моделирования	`main.Результаты.navigateTo();`

Построение модели для проведения простого эксперимента завершено. Теперь отладьте модель, используя возможности AnyLogic по обнаружению ошибок. Если вы всё выполнили корректно, то получите результаты, показанные на рис. 6.14.

Замечание. При каждом запуске модели результаты будут повторяться, если используются только распределения и случайные функции из AnyLogic. Например, если вместо функции uniform(1) использовать функцию random(), которая является функцией Java, результаты моделирования воспроизводиться не будут.

Рис. 6.14. Результаты моделирования согласно постановке задачи

6.3. Интерпретация результатов моделирования

Мы провели эксперименты с моделями, в AnyLogic увеличив модельное время в 1000 раз. При этом начальное значение генераторов случайных чисел установили 82874.

Эксперименты проводились при времени работы предприятия 40 часов (2400 единиц модельного времени).

Результаты экспериментов приведены в виде различных показателей функционирования предприятия в табл. 6.10. В ней приняты те же обозначения (идентификаторы) показателей, что и в модели AnyLogic.

Расшифруем некоторые составные части идентификаторов:

ПК1…ПК4 — пункты контроля блоков 1…4 соответственно;

ПС — пункты сборки изделий;

СВК — стенды выходного контроля изделий;

ПП — пункт приёмки изделий.

Сравнительная оценка, напоминаем, также проводилась между результатами, полученными в GPSS World и AnyLogic 7. Она свидетельствует о том, что результаты моделирования, полученные в различных средах, адекватны.

Различие мало, например, для коэффициента увеличения стоимости изделия оно составляет 0,004…0,005. Коэффициенты увеличения стоимости изделий, полученные в AnyLogic 7 и AnyLogic 6, практически во всех экспериментах совпадают.

Коэффициенты использования различных подразделений предприятия по производству изделий также отличаются на тысячные доли 0,003…0,007 или они одинаковые. Например, наибольшее отличие коэффициента использования ПК3 (коэфИспПК3) составляет 0,007 в первом эксперименте. Во втором и третьем экспериментах он равен 1. Увеличение числа ПК3 в четвёртом эксперименте уменьшает коэфИспПК3, но при этом коэффициент использования пунктов сборки (коэфИспПС) равен 1. Следовательно необходимо увеличить число пунктов сборки для увеличения производства изделий.

В целях уменьшения затрат на производство изделий целесообразно уменьшить число ПК1, поскольку они загружены примерно на треть (0,315…0,398) в первых трёх экспериментах и на две третьих (0,598) в последнем четвёртом эксперименте. Также можно уменьшить число ПК4.

Таблица 6.10

Показатели функционирования предприятия

Показатели	GPSS World	AnyLogic6	AnyLogic7
1) Согласно постановке задачи			
колГотИзд	121,628	122,337	121,721
	$\Delta_{11} = \lvert 121,628 - 121,721 \rvert = 0,093$		
стоимГотИзд	74410	75151	74763
минСтоимГотИзд	70787	71200	70841
	$\Delta_{12} = \lvert 70787 - 70841 \rvert = 54$		
стоимБракБл	3099	3122	3079
коэфУвелСтоимИзд	1,095	1,099	1,099
	$\Delta_{13} = \lvert 1,095 - 1,099 \rvert = 0,004$		
времяИзгИзд	19,732	19,618	19,717
коэфИспПК1	0,315	0,316	0,315
коэфИспПК2	0,726	0,728	0,729
коэфИспПК3	0,696	0,699	0,703
коэфИспПК4	0,472	0,474	0,475
коэфИспПС	0,581	0,586	0,583
коэфИспСВК	0,545	0,548	0,545
коэфИспПП	0,539	0,539	0,536
2) aveTimeShop1=15, aveTimeShop2=10, aveTimeShop3=10, aveTimeShop4=15			
колГотИзд	147,909	147,954	148,039
	$\Delta_{21} = \lvert 147,909 - 148,039 \rvert = 0,130$		
стоимГотИзд	90487	90902	90941
минСтоимГотИзд	86083	86109	86158
	$\Delta_{22} = \lvert 86083 - 86158 \rvert = 75$		
стоимБракБл	3848	3824	3844
коэфУвелСтоимИзд	1,096	1,1	1,1
	$\Delta_{23} = \lvert 1,096 - 1,100 \rvert = 0,004$		
времяИзгИзд	16,226	16,221	16,212
коэфИспПК1	0,398	0,4	0,401
коэфИспПК2	0,800	0,804	0,801
коэфИспПК3	1,0	1,0	1,0
коэфИспПК4	0,566	0,563	0,566
коэфИспПС	0,705	0,705	0,704
коэфИспСВК	0,660	0,662	0,665
коэфИспПП	0,654	0,654	0,655

Окончание табл. 6.10

Показатели функционирования предприятия

Показатели	GPSS World	AnyLogic6	AnyLogic7
3) aveTimeShop4=10			
колГотИзд	153,743	154,327	154,004
	$\Delta_{31} = \lvert 153,743 - 154,004 \rvert = 0,261$		
стоимГотИзд	94050	94815	94614
минСтоимГотИзд	89478	89818	89630
	$\Delta_{32} = \lvert 89478 - 89630 \rvert = 152$		
стоимБракБл	4412	4450	4432
коэфУвелСтоимИзд	1,100	1,105	1,105
	$\Delta_{33} = \lvert 1,100 - 1,105 \rvert = 0,005$		
времяИзгИзд	15,610	15,551	15,584
коэфИспПК1	0,398	0,399	0,398
коэфИспПК2	0,800	0,797	0,803
коэфИспПК3	1,0	1,0	1,0
коэфИспПК4	0,850	0,852	0,851
коэфИспПС	0,733	0,737	0,737
коэфИспСВК	0,691	0,691	0,688
коэфИспПП	0,678	0,681	0,682
4) aveTimeShop1=10, postKontr3=3			
колГотИзд	209,234	209,086	209,596
	$\Delta_{41} = \lvert 209,234 - 209,596 \rvert = 0,362$		
стоимГотИзд	127880	128328	128659
минСтоимГотИзд	121774	121688	121984
	$\Delta_{42} = \lvert 121744 - 121984 \rvert = 240$		
стоимБракБл	4952	4848	4880
коэфУвелСтоимИзд	1,091	1,094	1,095
	$\Delta_{43} = \lvert 1,091 - 1,095 \rvert = 0,004$		
времяИзгИзд	11,470	11,478	11,451
коэфИспПК1	0,600	0,599	0,598
коэфИспПК2	0,799	0,801	0,799
коэфИспПК3	0,701	0,700	0,698
коэфИспПК4	0,845	0,853	0,847
коэфИспПС	1,0	1,0	1,0
коэфИспСВК	0,936	0,936	0,937
коэфИспПП	0,923	0,923	0,927

ГЛАВА 7. МОДЕЛЬ ФУНКЦИОНИРОВАНИЯ ТЕРМИНАЛА

7.1. Постановка задачи

Общий вид терминала показан на рис. 7.1. Территория терминала обозначена A, примыкающая городская территория — B.

1. Автомобиль (транспортное средство), груженный или порожний, попадает в порт по дороге общего пользования C. В случае отсутствия мест на парковке D терминала, дорога становится накопительным буфером (очередь с дисциплиной FIFO).

2. Если имеется свободное место, автомобиль въезжает на парковку, водитель выходит и с документами идет в офис E. Процедура парковки занимает около 2 мин.

3. В офисе водитель дожидается своей очереди на обслуживание у одного из окошек. Дождавшись, он оформляет документы на въезд. Получив их, он возвращается к своему автомобилю. Оформление документов занимает, вместе с ходьбой, около 10 мин. Одновременно на терминал отсылается заявка на обслуживание данного автомобиля.

4. Если ворота F имеют свободную полосу, автомобиль подъезжает на полосу досмотра. Здесь у него проверяют разрешение на въезд и проводят физический досмотр контейнера (пломб, наличия повреждений, отсутствия посторонних лиц и пр.). Досмотр занимает 2 мин.

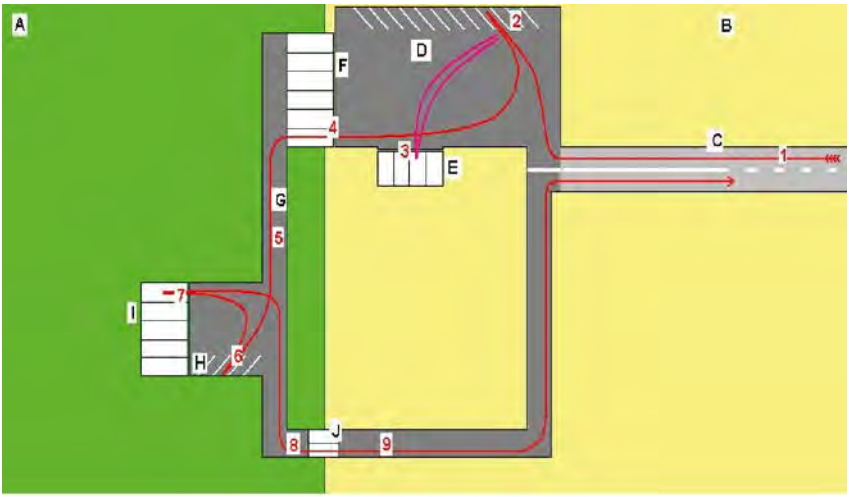


Рис. 7.1. Схема терминала

5. Автомобиль следует на оперативную парковку H, расположенную рядом с зоной погрузки-разгрузки I. Среднее время движения 2 мин. Этот участок дороги внутри терминала может использоваться как накопительный буфер, если нет свободных мест на парковке у зоны погрузки.

6. Автомобиль становится на парковку H и ждет своей очереди на погрузку (момента выполнения заявки на его обслуживание, отправленной на шаге 3). Среднее время выполнения заявки составляет 10 мин.

Когда со стороны терминала готово транспортное средство для его погрузки-разгрузки (заявка на обслуживание автомобиля выполнена), и имеется свободная ячейка для обработки автомобиля в зоне H, автомобиль подъезжает к свободной ячейке для погрузки. Среднее время движения 2 мин. Если заявка была выполнена до приезда автомобиля и имеется свободная ячейка, автомобиль может прямо подъехать к ячейке, минуя парковку 6. Среднее время обслуживания автомобиля 5 мин.

7. Обслуженный автомобиль по терминальному проезду подъезжает к выездным воротам терминала J. Среднее время движения 2 мин.

8. После осмотра автомобиль покидает терминал. Среднее время осмотра 2 мин.

Необходимо разработать имитационную модель и промоделировать функционирование терминала в течение 8 ч.

Определить:

количество обработанных автомобилей;

среднее время обработки одного автомобиля;

коэффициент обработки автомобилей терминалом;

показатели использования элементов терминала.

В модели автомобили следует представить заявками. Все остальные элементы терминала (парковку D, офис E, полосы у ворот F и J, места у зон I и Z) — многоканальными устройствами (МКУ). Дадим МКУ имена согласно постановке задачи, добавив знак подчеркивания, например, D_.

Введем масштабирование: 1 единица модельного времени соответствует 1 мин, то есть, например, время моделирования равно 8*60 = 480 единиц модельного времени.

В постановке задачи на разработку модели указаны средние значения времени поступления автомобилей и их обработки.

Примем, что интервалы времени во всех случаях распределены по экспоненциальному закону.

Декомпозиция терминала и состав сегментов модели определяются разработчиком. Введем следующие сегменты:

ввод исходных данных;

событийная часть модели;

вычисление результатов моделирования.

Как видно, в данном случае можно обойтись без разделения событийной части модели на сегменты.

7.2. Модель в AnyLogic

7.2.1. Исходные данные и результаты моделирования

1. Для ввода исходных данных используем элемент **Параметр**. Выполните команду **Файл/Создать/Модель** на панели инструментов.
2. В поле **Имя модели** диалогового окна **Новая модель** введите `Терминал`. Выберите каталог, в котором будут сохранены файлы модели. Щёлкните кнопку **Готово**.

Создадим две области просмотра. Первую для ввода исходных данных и вывода результатов моделирования, вторую — для размещения событийной части модели.

Создайте вторую область просмотра для размещения объектов модели на агенте Main.

1. Из палитры **Презентация** перетащите элемент **Область просмотра**.
2. На странице **Основные** панели **Свойства** в поле **Имя:** введите `Модель_Терминал`.
3. Задайте расположение области просмотра относительно её якоря с помощью элемента управления **Выравнивать по:** `Верхн. левому углу`.
4. Выберите режим масштабирования из выпадающего списка **Масштабирование:** `Подогнать под окно`.
5. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `0`, **Y:** `750`, **Ширина:** `560`, **Высота:** `330`.
6. Из палитры **Презентация** перетащите элемент **Скруглённый прямоугольник**. Оставьте имя, предложенное системой.
7. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `20`, **Y:** `760`, **Ширина:** 530, **Высота:** `310`. Задайте, как будет располагаться область просмотра относительно её

якоря, с помощью элемента управления **Выравнивать по:** `Верхн. левому углу`.

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

9. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `0`, **Y:** `0`, **Ширина:** `680`, **Высота:** `410`.

10. Из палитры **Презентация** перетащите элемент **Прямоугольник**. Оставьте имя, предложенное системой.

11. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `30`, **Y:** `40`, **Ширина:** 640, **Высота:** `350`.

12. Из палитры **Презентация** перетащите второй элемент **Область просмотра**.

13. Перейдите на страницу **Основные** панели **Свойства**. В поле **Имя:** введите `Данные`.

14. Задайте, как будет располагаться

15. Перетащите элемент **text** и на странице **Текст** панели **Свойства** в поле вместо слова text введите `Исходные данные`.

16. Перетащите второй элемент **text** и на странице **Текст** панели **Свойства** в поле вместо слова text введите `Результаты моделирования`.

17. В **Палитре** выделите **Основная**. Перетащите элементы **Параметр** на элементы с именами **Исходные данные и Результаты моделирования**. Разместите их и дайте имена так, как показано на рис. 7.2.

18. Значения свойств установите согласно табл. 7.1. Имена параметров (в том числе и знак подчёркивания) приняты те же, что и в GPSS-модели.

Таблица 7.1

Имя	Тип	Значение по умолчанию	Имя	Тип	Значение по умолчанию
D_	int	10	timeD	double	2
E_	int	5	timeE	double	10
F_	int	5	timeF	double	2
I_	int	7	timeI	double	5
ZP_	int	2	timeZ	double	10
J_	int	7	timeJ	double	2
			timeA	double	9
			timeFH	double	2
			timeIJ	double	2

19. Для вывода результатов моделирования используем элемент **Переменная**. В **Палитре** выделите **Основная**. Перетащите элементы **Переменная**. Разместите их и дайте им имена так, как показано на рис. 7.2. Тип переменных `double`.

В GPSS World коэффициенты использования элементов терминала, длины к ним очередей определяются системой автоматически и выводятся в стандартном отчёте.

Коэффициенты использования элементов терминала (в нашей модели их будут имитировать объекты delay) автоматически определяются и в AnyLogic. Тем не менее, для удобства их чтения в ходе и по окончании моделирования мы ввели переменные **KoefIsp_E**, **KoefIsp_F**, **KoefIsp_Z**, **KoefIsp_I**, **KoefIsp_J**.

Для определения максимальных длин очередей к этим элементам терминала нами также введены переменные **очередь_E**, **очередь_F**, **очередь_Z**, **очередь_I**, **очередь_J**.

Замечание. При построении событийной части модели нужно устанавливать флажок **Включить сбор статистики** в свойствах этих элементов на странице **Специфические** панели **Свойства**.

Переменные **KolObrCar**, **TimeObr**, **KoefIsp** имеют тот же смысл, что и в GPSS-модели.

Переменная kolJ введена для счета числа обработанных транспортов. Переменная TimeSum введена для счета суммарного времени обработки всех транспортов. По значениям этих переменных рассчитывается среднее время обработки TimeObr транспорта.

Далее для определения количественных значений каждой из этих переменных мы напишем соответствующие Java-коды.

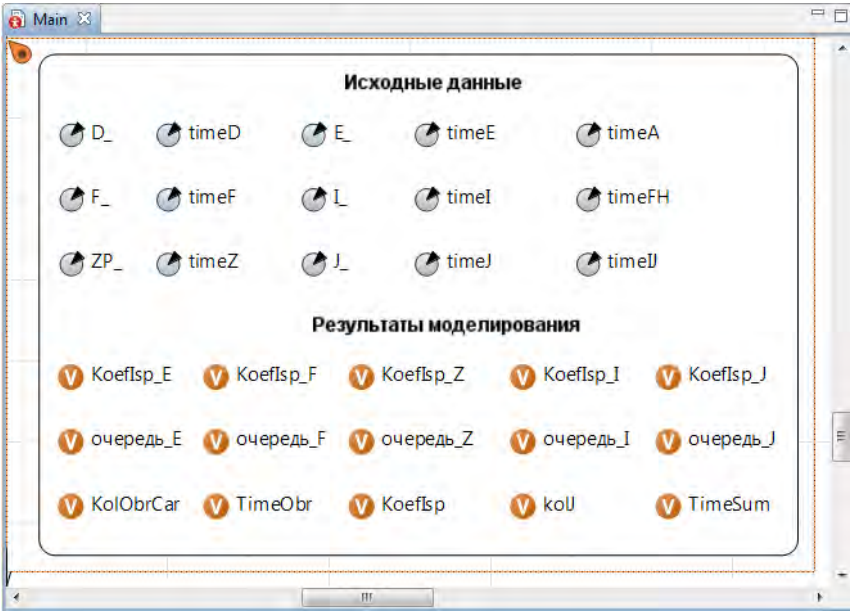


Рис. 7.2. Размещение элементов для ввода исходных данных и вывода результатов моделирования

7.2.2. Событийная часть модели

Объекты событийной части модели показаны на рис. 7.3.

Для построения использованы следующие двадцать объектов **Библиотеки моделирования процессов** (в скобках после имени каждого объекта указано их количество в событийной части модели): source (один); queue (8); delay (8); split (1); match (1); sink.

Создадим событийную часть модели. Для неё мы уже создали область просмотра с именем **Модель_Терминал** и перетащили элемент **Прямоугольник**, в котором и разместим все элементы.

1. Перетащите объект source. Установите его свойства согласно табл. 7.2.
2. Создайте нестандартный тип заявки `Car` с полями:

```
long id;
double vxod;
```

Дополнительное поле `id` нестандартного класса заявки `Car` введено для проверки объектом match условия объединения двух заявок в одну. Поле `vxod` предназначено для записи времени входа заявки в модель.

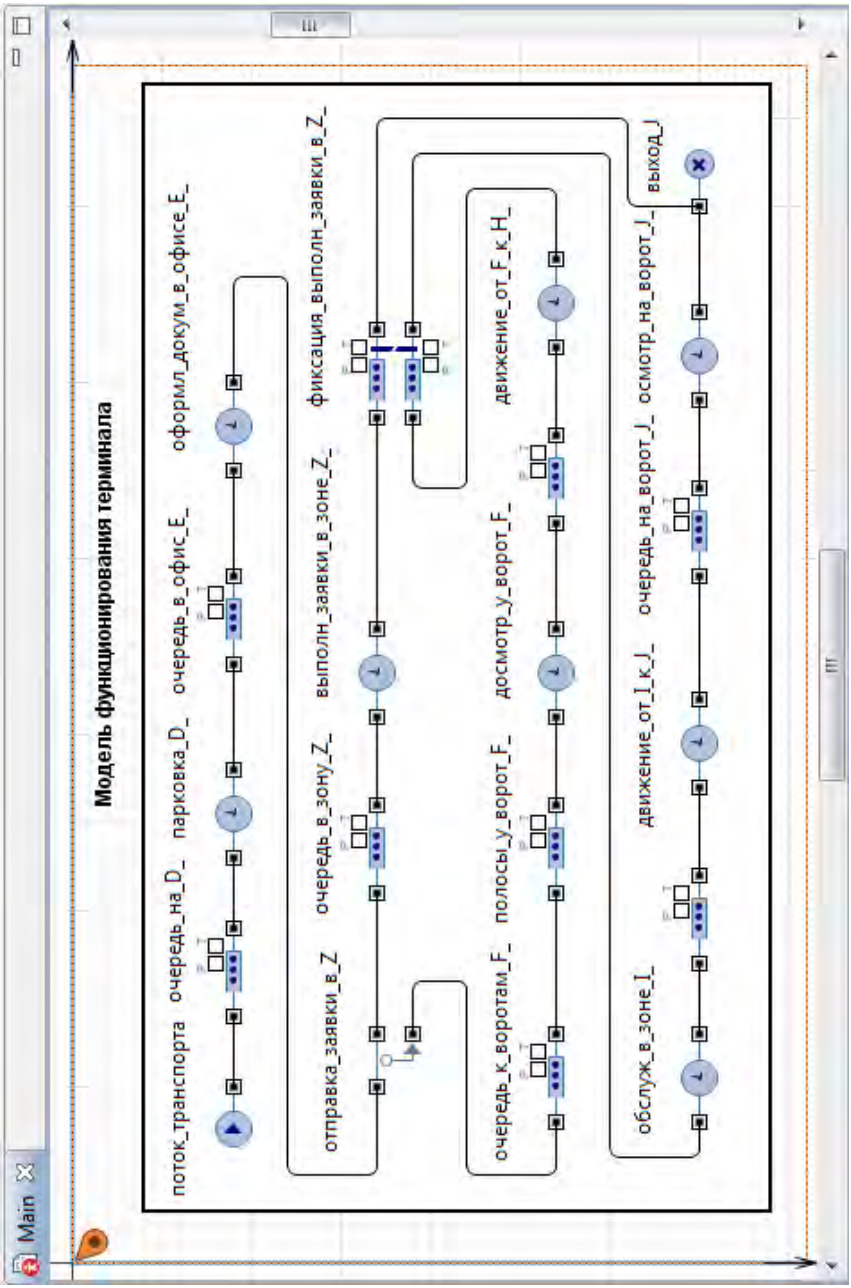


Рис. 7.3. Событийная часть модели **Терминал**

Перетащите остальные объекты, соедините так, как показано на рис. 7.3. Установите их свойства согласно табл. 7.3.

Таблица 7.2

Свойства объекта source

Имя	Свойства	Значения
поток_транс-порта	Отображать имя Тип заявки Прибывают согласно Время между прибытиями Новая заявка Действия При выходе	Установите флажок Car Времени между прибытиями exponential(1/timeA) Car entity.id = поток_транспорта.count(); entity.vxod = time();

Таблица 7.3

Свойства объектов событийной части модели

Свойства	Значение
Имя Отображать имя Тип заявки Максимальная вместимость Включить сбор статистики	очередь_на_D_ Установить флажок Car Установить флажок Установить флажок
Имя Отображать имя Тип заявки Тип Время задержки Вместимость Включить сбор статистики	парковка_D_ Установить флажок Car Определённое время exponential(1/timeD) D_ Установить флажок
Имя Отображать имя Тип заявки Вместимость Действия При подходе к выходу Включить сбор статистики	очередь_в_офис_E_ Установить флажок Car 10 **if** (очередь_E < очередь_в_офис_E_.size()) очередь_E = очередь_в_офис_E_.size(); Установить флажок

Имя Отображать имя Тип заявки Тип Время задержки Вместимость	оформл_докум_в_офисе_E_ Установить флажок Car Определённое время exponential(1/timeE) E_

Имя
Отображать имя
Тип заявки
Количество копий
Новая заявка (копия)
Действия
При выходе копии
	отправка_заявки_в_Z
	Установить флажок
	Car
	1
	Car
	entity.id = original.id;
	entity.vxod = original.vxod;

Имя Отображать имя Тип заявки Вместимость Действия При подходе к выходу Включить сбор статистики	очередь_в_зону_Z_ Установить флажок Car 100 **if** (очередь_Z < оче- редь_в_зону_Z_.size()) очередь_Z = очередь_в_зону_Z_.size(); Установить флажок
Имя Отображать имя Тип заявки Тип Время задержки Вместимость Включить сбор статистики	выполн_заявки_в_зоне_Z_ Установить флажок Car Определённое время exponential(1/timeZ) ZP_ Установить флажок
Имя Отображать имя Тип заявки Условие соответствия Вместимость 1 Вместимость 2 Действия При выходе 1 Действия При выходе 2	фиксация_выполн_заявки_в_Z_ Установить флажок Car entity1.id == entity2.id 100 100 entity.id = 0; **if** (очередь_I < фиксация _выполн_заявки_в_Z_.size2()) очередь_I = фиксация _выполн_заявки_в_Z_.size2();

Имя Отображать имя Тип заявки Максимальная вместимость	очередь_к_воротам_F_ Установить флажок Car Установить флажок
Имя Отображать имя Тип заявки Вместимость Действия При подходе к выходу	полосы_у_ворот_F_ Установить флажок Car F_ **if** (очередь_F < поло- сы_у_ворот_F_.size()) очередь_F = полосы_у_ворот_F_.size();
Имя Отображать имя Тип заявки Тип Время задержки Вместимость	досмотр_у_ворот_F_ Установить флажок Car Определённое время exponential(1/timeF) F_
Имя Тип заявки Вместимость	queue1 Сбросить флажок Car 100
Имя Отображать имя Тип заявки Тип Время задержки Вместимость	движение_от_F_к_H_ Установить флажок Car Определённое время exponential(1/timeFH) 10
Имя Отображать имя Тип заявки Тип Время задержки Вместимость	обслуж_в_зоне_I_ Установить флажок Car Определённое время exponential(1/timeI) I_
Имя Отображать имя Тип заявки Вместимость	queue Сбросить флажок Car 100
Имя Отображать имя Тип заявки	движение_от_I_к_J_ Установить флажок Car

Окончание табл. 7.3

Тип Время задержки Вместимость Включить сбор статистики	Определённое время exponential(1/timeIJ) 10 Установить флажок
Имя Отображать имя Тип заявки Вместимость Действие при подходе к выходу Включить сбор статистики	очередь_на_ворот_J_ Установить флажок Car 10 **if** (очередь_J < очередь_на_ворот_J_.size()) очередь_J = очередь_на_ворот_J_.size(); Установить флажок
Имя Отображать имя Тип заявки Тип Время задержки Вместимость	осмотр_на_ворот_J_ Установить флажок Car Определённое время exponential(1/timeJ) J_

Поясним необходимость и целесообразность применения некоторых объектов. В модели заявка имитирует транспорт. Объект split предназначен для расщепления одной заявки в нашей модели на две заявки. Одна поступает, как документ, в зону Z, а вторая, как транспорт, продолжает движение. Когда в зоне Z необходимые для обработки транспорта действия выполнены (подготовлены документы), объект match фиксирует этот момент, то есть синхронизирует дальнейшее движение транспорта. Вторая заявка направляется в объект sink и уничтожается. Как известно, объединить две заявки в одну (а значит не использовать объект sink) можно с помощью объекта combine. Однако в нашем случае combine использовать нельзя, так как он не проверяет выполнение у заявок условий, необходимых для их уничтожения. Поэтому могут быть объединены различные заявки. Объект match проверяет условия объединения, установленные в нашем случае разработчиком модели в дополнительном поле id нестандартного класса заявки Car (`entity1.id == entity2.id`), именно двух заявок в одну. То есть синхронизации движения заявки как документа и заявки как автомобиля при применения объекта combine не обеспечивается.

В табл. 7.4 указаны свойства объекта sink.

Таблица 7.4

Свойства	Значение
Имя	выход_J
Отображать имя	Установить флажок
Тип заявки	Car

Действия При входе	`if (entity.id != 0){`
	`kolJ ++ ;`
	`KolObrCar = kolJ/10000;`
	`KoefIsp = kolJ/поток_транспорта.count();`
	`TimeSum += (time() - entity.vxod);`
	`TimeObr = TimeSum / kolJ;`
	`KoefIsp_E = оформл_докум_в_офисе_E_.statsUtilization.mean();`
	`KoefIsp_F = досмотр_у_ворот_F_.statsUtilization.mean();`
	`KoefIsp_Z = выполн_заявки_в_зоне_Z_.statsUtilization.mean();`
	`KoefIsp_I = обслуж_в_зоне_I_.statsUtilization.mean();`
	`KoefIsp_J = осмотр_на_ворот_J_.statsUtilization.mean();}`

Остановимся на коде свойства **Действия При входе**. Заявки с выхода 1 объекта match можно было бы не направлять на этот объект sink, а добавить еще объект sink и уничтожать их там. Но у нас ознакомительная версия, которая не позволяет иметь в модели на диаграмме одного класса больше двадцати объектов (в образовательной версии это возможно). Нам не хватило ровно одного объекта, поэтому пришлось поступить так (размещать объекты на разных диаграммах ради этого мы посчитали нецелесообразным). Для реализации такого приёма на выходе 1 объекта match `entity.id = 0`, то есть поле `id` обнуляется и все заявки с таким полем игнорируются. Но в количестве заявок, вошедших в объект sink, они учитываются. Поэтому пришлось для счета количества обработанных транспортов ввести переменную `kolJ`, хотя имеется стандартная функция `count()`, которая возвращает количество заявок, вошедших в данный объект. Любых заявок, без какого-либо их разделения.

7.2.3. Результаты моделирования

Результаты проведенных нами на модели экспериментов сведены в табл. 7.5. В этот раз мы выполнили по одному эксперименту на каждой модели. В GPSS-модели начальные числа генераторов случайных чисел устанавливали произвольно, а в AnyLoic-модели установили число 1876. В GPSS World выполнили 10000 прогонов, а в AnyLogic увеличили время моделирования в 10000 раз, то есть моделировали в течение 4800000 мин.

Таблица 7.5

Показатели	Системы моделирования		
	GPSS World	AnyLogic6	AnyLogic7
KolObrCar	53,31	53,41	53,294
TimeObr	35,485	36,109	36,053
KoefIsp	1,000	1,000	1,000
KoefIsp_E	0,222	0,222	0,222
KoefIsp_F	0,044	0,044	0,044
KoefIsp_Z	0,556	0,556	0,554
KoefIsp_I	0,079	0,080	0,062
KoefIsp_J	0,032	0,032	0,032
очередь_E	5	5	8
очередь_F	1	1	1
очередь_Z	21	20	16
очередь_I	22	22	17
очередь_J	1	1	1
Машинное время	25 мин 34 с	25 с	37 с

Как видно, результаты моделирования идентичны. Кроме машинного времени, которое в GPSS World составляет 25 мин 34 с, то есть более чем в 40 раз больше, чем в AnyLogic7. Различие в количестве обработанных в течение 8 часов автомобилей (KolObrCar) и среднего времени обработки одного автомобиля (TimeObr) незначительно. Коэффициенты использования элементов терминала и максимальные длины очередей к ним одинаковые.

7.3. Эксперименты

Далее поступим так. Проведем эксперименты с моделями в GPSS World и AnyLogic. В GPSS World это будет дисперсионный анализ, а в AnyLogic — оптимизация стохастических моделей. В результате мы получим оптимальные значения факторов и показатели функционирования терминала, которые и оценим.

7.3.1. Первый оптимизационный эксперимент в AnyLogic

Создайте первый эксперимент **Оптимизация стохастических моделей** в AnyLogic. Цель эксперимента: определение минимального времени обработки одного автомобиля в зависимости от тех же факторов и их значений, что и в отсеивающем эксперименте.

1. Откройте в AnyLogic модель **Терминал**.
2. В панели **Проект** щёлкните правой кнопкой мыши элемент модели **Терминал** и из меню выберите **Создать/ Эксперимент**.
3. В появившемся диалоговом окне из списка **Тип эксперимента:** выберите **Оптимизация**.
4. В поле **Имя** введите имя эксперимента, например, `Опт-Терминал1`.
5. В поле **Агент верхнего уровня:** выберите Main. Этим выбором вы задали корневой (главный) агент эксперимента. Тип этого агента будет играть роль корня дерева объектов модели, запускаемой экспериментом.
6. Если вы хотите применить к создаваемому эксперименту временные установки другого эксперимента, оставьте установленным флажок **Копировать установки модельного времени из:** и выберите эксперимент из выпадающего списка.
7. Установите опцию **минимизировать**.
8. В поле **Целевая функция** введите `root.TimeObr`, так как корневой агент модели доступен здесь как root.
9. На странице **Случайность** выберите опцию **Фиксированное начальное число (воспроизводимые прогоны)**. В поле начальное число введите 1687. В этом случае при каждом запуске модели генератор случайных чисел будет инициализироваться этим числом. Поэтому прогоны будут воспроизводимыми при очередном запуске модели, что полезно при отладке модели.
10. 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.
11. В нашей модели нужно менять факторы timeA, timeE, timeF, timeI, timeZ, timeFH. Примем, что каждый фактор имеет два уровня k=2, m=6, тогда число итераций $I = k^m = 2^6 = 64$.
12. В поле **Количество итераций:** установите 64. Задайте параметры, значения которых будут меняться. В таблице на рис. 7.9 перечислены часть параметров корневого агента Main.

13. Однако они будут расположены не так, как на рис. 7.9. Можно сказать, будут «неудобно» расположены. Они будут разбросаны и некоторые могут оказаться даже внизу таблицы. Для перемещения оптимизируемых параметров в верхнюю часть таблицы используйте методику, описанную в п. 5.1.10.4.

14. Чтобы разрешить варьирование параметров оптимизатором, перейдите на строку с параметром timeE. Щёлкните мышью в ячейке **Тип**. Выберите тип параметра, отличный от значения **фиксированный**. Чтобы параметры изменялись точно так же, как и в отсеивающем эксперименте GPSS World, то есть имели два уровня (нижний и верхний или минимальное и максимальное значения), выберите тип **дискретный**.

15. Задайте диапазон допустимых значений параметра. Для чего введите в ячейку **Мин** минимальное значение 10, в ячейку **Макс** максимальное значение 15. Так как параметр дискретный, в ячейке **Шаг** укажите величину шага 5.

16. Задайте так же остальные варьируемые параметры эксперимента, как на рис. 7.9.

17. Перейдите на страницу **Репликации** панели **Свойства**.

18. Установите флажок **Использовать репликации**.

19. Число репликаций (прогонов) в одной итерации (наблюдении) может быть фиксированным или переменным. *Фиксированное число репликаций*, например, при доверительной вероятности $\alpha = 0,95$, точности $\varepsilon = 0,01$ и стандартном отклонении $\sigma = 0,1$ может быть определено по формуле:

$$N = t_\alpha^2 \frac{\sigma^2}{\varepsilon^2} = 1,96^2 \frac{0,1^2}{0,01^2} = 3,8416 \cdot 100 \approx 400,$$

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

20. Выберите опцию **Фиксированное количество репликаций** и в соответствующем поле установите `400`.

21. Щёлкните **Создать интерфейс**.

После щелчка удаляется презентация эксперимента и создаётся интерфейс эксперимента заново 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 будут видны знаки вопросов напротив оптимизационных параметров.

22. В меню запуск выполните **Терминал/ОптТерминал1.**

Свойства

ОптТерминал1 - Оптимизационный эксперимент

Имя: ОптТерминал1 Исклю

Агент верхнего уровня: Main

Целевая функция: минимизировать максимизиро

root.TimeObr

Количество итераций: 64

Автоматическая остановка

Максимальный размер памяти: 256 Мб

Создать интерфейс

Параметры

Параметры:

Параметр	Тип	Значение		
		Мин.	Макс.	Шаг
timeE	дискретный	10	15	5
timeA	дискретный	10	20	10
timeF	дискретный	2	4	2
timeI	дискретный	5	10	5
timeFH	дискретный	2	4	2
timeZ	дискретный	10	15	5
D_	фиксированный	10		
timeD	фиксированный	2		

Рис. 7.9. Страница **Основные** панели **Свойства** эксперимента

23. Щёлкните **Запустить оптимизацию**. Начнет выполняться эксперимент. Во время эксперимента можно видеть на графике изменение значения целевой функции (вертикальная ось). После выполнения 400*64=25600 прогонов (рис. 7.10) эксперимент остановится.

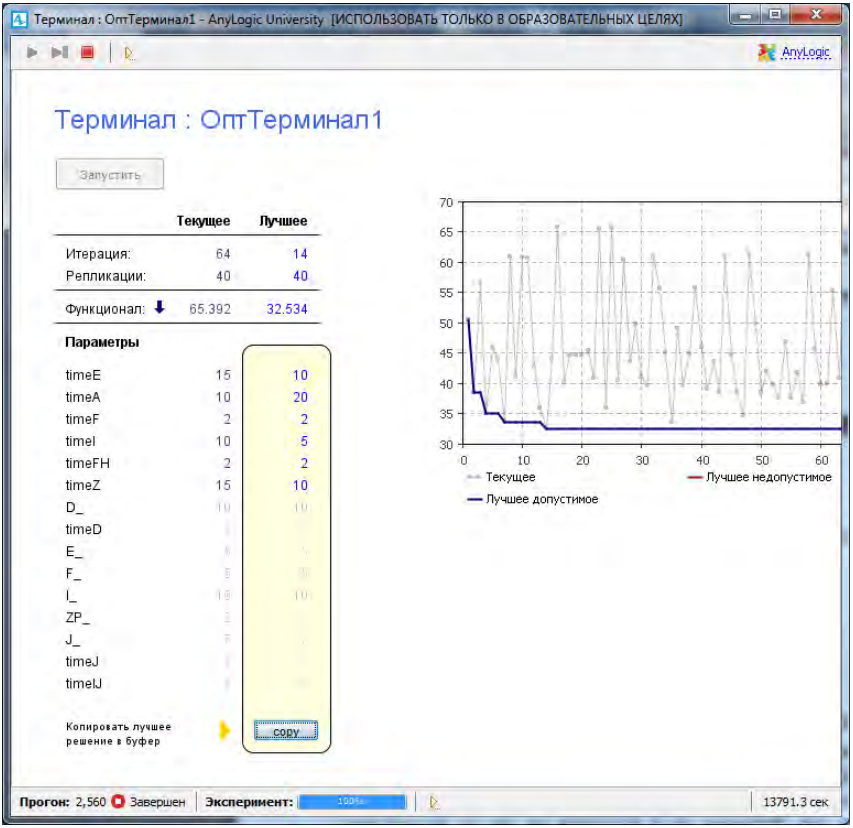


Рис. 7.10. Результаты первого оптимизационного эксперимента

7.3.2. Второй оптимизационный эксперимент в AnyLogic

Создайте второй эксперимент **Оптимизация стохастических моделей**. Цель эксперимента: определение максимального количества обработанных автомобилей в зависимости от тех же факторов и их значений, что и в первом эксперименте.

1. Откройте в AnyLogic модель **Терминал**.
2. В панели **Проект** щёлкните правой кнопкой мыши элемент модели Terminal и из контекстного меню выберите **Создать/ Эксперимент**. В появившемся диалоговом окне из списка **Тип эксперимента:** выберите **Оптимизация**.
3. В поле **Имя** введите имя эксперимента, например, `ОптТерминал2`.
4. Выполните пп. 5…8 первого эксперимента.

5. Установите опцию **максимизировать**.

6. В поле **Целевая функция** введите `root.KolObrCar`.

7. Выполните пп. 11…19 первого эксперимента.

8. Так как в данном эксперименте определяется максимальное количество обработанных автомобилей, то доверительную вероятность оставьте прежней $\alpha = 0,95$, а точность и стандартное отклонение достаточно задать равными 1, то есть $\varepsilon = 1$ и $\sigma = 1$. Тогда фиксированное число репликаций будет равно:

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

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

10. Щёлкните **Создать интерфейс**.

11. В меню запуск выполните **Терминал/ОптТерминал2.**

12. Щёлкните **Запустить оптимизацию**. После выполнения 4*64=256 прогонов (рис. 7.11) эксперимент остановится. На второй итерации (втором наблюдении) четвертой репликации (прогоне) получено оптимальное значение функционала: 48,129.

Перейдем к рассмотрению результатов экспериментов.

7.4. Интерпретация результатов экспериментов

Результаты экспериментов с имитационной моделью функционирования терминала сведены в табл. 7.7 и 7.8.

Таблица 7.7

Первый эксперимент			
Показатели, параметры	GPSS World	AnyLogic6	AnyLogic7
TimeObr	**31,844**	**31,799**	**32,534**
timeA	**20**	**20**	**20**
timeE	**10**	**10**	**10**
timeF	**2**	**2**	**2**
timeI	**5**	**5**	**5**
timeZ	**10**	**10**	**10**
timeFH	**2**	**2**	**2**
TimeObr	31,605	31,799	34,527
timeA	20	20	20
timeE	10	10	10
timeF	4	2	4
timeI	5	5	5
timeZ	10	10	10
timeFH	4	2	4

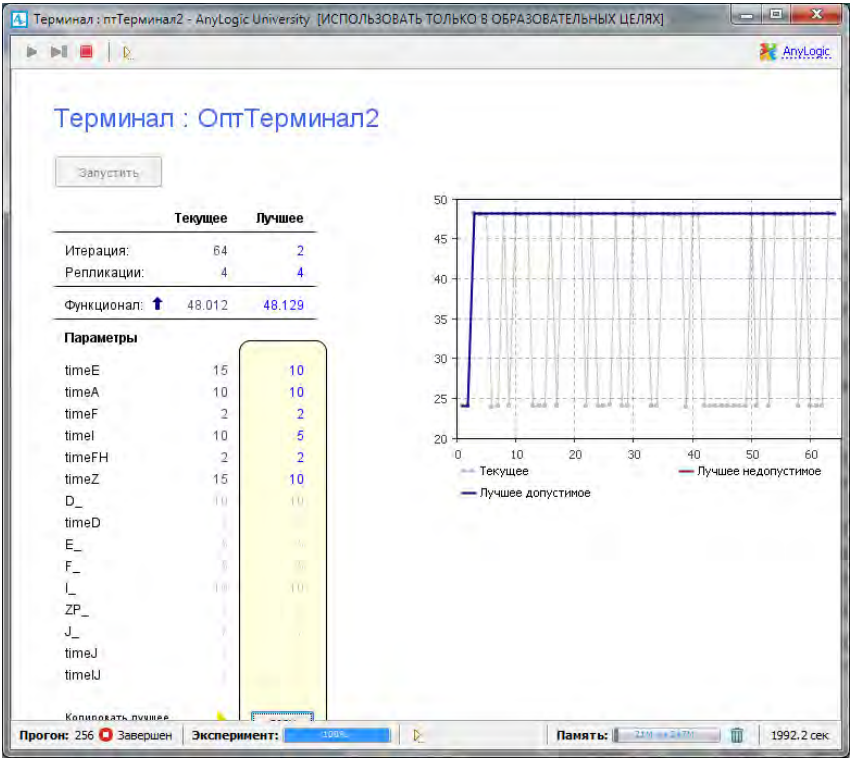


Рис. 7.11. Результаты второго оптимизационного эксперимента

Значения факторов (параметров) в отсеивающих экспериментах GPSS World и оптимизационных экспериментах AnyLogic изменялись одинаково. Значит, в отчёте GPSS World об отсеивающем эксперименте можно найти значение целевой функции, рассчитанное по параметрам, которые определены AnyLogic в эксперименте как оптимальные. Показатели и параметры при таком подходе к сравнительной оценке результатов моделирования в табл. 7.7 и 7.8 выделены жирным шрифтом.

В первом эксперименте минимальное время обработки одного автомобиля, определенное оптимизатором AnyLogic, составляет **TimeObr = 32,534** мин (см. рис. 7.10) при следующих значениях параметров: **timeA = 20, timeE = 10, timeF = 2, timeI = 5, timeZ = 10, timeFH = 2.**

По значениям оптимальных параметров находим в отчёте GPSS World (см. рис. 7.7) **TimeObr = 31,844** мин. Как видим, отличие составляет **0,045** мин.

Таблица 7.8

Второй эксперимент			
Показатели, параметры	GPSS World	AnyLogic6	AnyLogic7
KolObrCar	**48,049**	**48,08**	**48,129**
timeA	**10**	**10**	**10**
timeE	**10**	**10**	**10**
timeF	**2**	**2**	**2**
timeI	**5**	**5**	**5**
timeZ	**10**	**10**	**10**
timeFH	**2**	**2**	**2**
KolObrCar	48,30	48,08	47,957
timeA	10	10	10
timeE	15	10	15
timeF	4	2	4
timeI	5	5	5
timeZ	15	10	15
timeFH	4	2	4

Во втором оптимизационном эксперименте максимальное количество обработанных автомобилей, определенное AnyLogic, составляет **KolObrCar = 48,129** (см. рис. 7.11) при следующих значениях параметров:

timeA = 10, timeE = 10, timeF = 2, timeI = 5, timeZ = 10, timeFH = 2.

По значениям оптимальных параметров находим в отчёте GPSS World (см. рис. 7.8) **KolObrCar = 48,049**. Как видим, результаты практически одинаковы, если учесть измерение показателя целыми значениями (разница 0,08).

Теперь попробуем в отчётах GPSS World найти значения целевой функции, которые превышают значения, полученные при оптимальных параметрах AnyLogic. Такие значения нашлись, и также включены в табл. 7.7 и 7.8 (не выделены жирным шрифтом). Но и они свидетельствуют о близости полученных результатов моделирования в GPSS World и AnyLogic (48,300 – 47,957 = 0,343).

ГЛАВА 8. МОДЕЛЬ ПРЕДОСТАВЛЕНИЯ РЕМОНТНЫХ УСЛУГ

8.1. Постановка задачи

В фирму предоставления ремонтных услуг поступают заявки n типов с вероятностями p1, p2, …, pn соответственно. Интервалы времени Tп между двумя очередными поступлениями одного типа заявок случайные. Каждый любой тип заявки может требовать одного из a1, a2, …, ak видов ремонта с вероятностями pa1, pa2, …, pak соответственно.

В фирме имеются n1, n2, …, nn мастеров для выполнения заявок каждого типа соответственно. Мастера n1 выполняют заявки первого типа. Если их нет и мастера n2, …, nn групп заняты, они выполняют заявки этих типов. При этом поступающие заявки первого типа ожидают их освобождения. Мастера n2 выполняют заявки второго типа. Если их нет и мастера n3, n4, …, nn групп заняты, они выполняют заявки этих типов. При этом поступающие заявки второго типа ожидают их освобождения. Аналогичные обязанности и у мастеров остальных групп. Только мастера nn выполняют заявки одного n-го типа.

Время выполнения заявки n-го типа 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 заявок между группами мастеров осуществляется d диспетчерами. Время, затрачиваемое одним диспетчером на одну заявку, T1, случайное. Диспетчерами не принимаются к ремонту q заявок всех типов.

8.1.1. Исходные данные

exponential(Tп) = exponential (30); n = 4;
p1 = 0.2, p2 = 0.3, p3 = 0.25, p4 = 0.25;
p11 = 0.5, p12 = 0.25, p13 = 0.25;
n1 = 2; T11 = 30; T12 = 40; T13 = 50;
n2 = 1; T21 = 20; T22 = 30; T23 = 40;
n3 = 1; T31 = 15; T32 = 25; T33 = 35;
n4 = 1; T41 = 25; T42 = 35; T43 = 45;
d = 2; normal(T1, To1) = normal(15, 2); q = 2 %.

Интервалы времени между поступлениями заявок и время выполнения заявок распределены по экспоненциальному закону.

Время обслуживания одной заявки диспетчером подчинено нормальному закону.

8.1.2. Задание на исследование

Разработать имитационную модель предоставления ремонтных услуг. Исследовать зависимость количества выполненных заявок и вероятностей выполнения заявок всех типов от интервала Тп поступления их в ремонт и вероятностей p1, p2, p3, p4.

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 фирмы предоставления ремонтных услуг.

8.1.3. Формализованное описание модели

Уясним задачу на разработку модели, предварительно представив структуру фирмы предоставления ремонтных услуг (рис. 8.1) как СМО.

Рис. 8.1. Фирма предоставления ремонтных услуг как СМО

Фирма предоставления ремонтных услуг представляет собой многофазную многоканальную систему массового обслуживания разомкнутого типа с отказами.

Исходя из структуры, модель предоставления ремонтных услуг должна состоять из следующих сегментов:

ввода исходных данных;

источника заявок;

диспетчеров;

мастеров;

учёта выполненных ремонтов.

Заявки на ремонт должны иметь следующие параметры (поля):

`типЗ` — код типа заявки;

`видР` — вид ремонта;

`времяР` — время выполнения одного вида ремонта;

Как уже отмечалось, интервалы между соседними заявками подчинены экспоненциальному закону. Принято, что за время `Тр` от каждого источника поступает одна заявка. Тогда средний интервал поступления заявок равен `Тр/n`. Поэтому вместо `n` объектов имитации источников заявок будем использовать один.

Код типа заявки определяется в виде чисел 1, 2, 3, 4, так как `n=4`. Код вида ремонта определяется также числами 1…3 соответственно. Для этого используются следующие исходные данные:

`p1` … `p4` — вероятности поступления заявок 1…4 типов соответственно;

`p11` … `p43` — вероятности поступления заявок 1…4 типов с видами 1…3 ремонтов соответственно.

Коды типа заявки и вида ремонта записываются в поля `типЗ` и `видР` соответственно.

По этим кодам определяется среднее время вида ремонта и заносится в поле `времяР`.

В процессе выполнения модели накапливаются следующие статистические данные:

`постЗаявТип1` … `постЗаявТип1`, `постЗаявТип` — количество поступивших заявок 1…4 типов и заявок всех типов;

`выпЗаявТип1` … `выпЗаявТип1`, `выпЗаявТип` — количество выполненных заявок 1…4 типов и заявок всех типов;

`выпРемВида11` … `выпРемВида43` — количество выполненных заявок 1…4 типов с видами 1…3 ремонтов соответственно.

Поскольку эти данные накапливаются за все прогоны модели, то для получения средних значений они делятся на количество прогонов `колПрог`. Например, `выпЗаявТип1=выпЗаявТип1/колПрог`.

По этим же статистическим данным рассчитываются:

`верВыпЗаяв1 … верВыпЗаяв4, верВыпЗаяв` — вероятности выполнения заявок 1…4 типов и заявок всех типов.

Например, `верВыпЗаяв1=выпЗаявТип1/постЗаявТип1`.

8.2. Модель в AnyLogic

8.2.1. Ввод исходных данных

Элементы для ввода исходных данных разместим на агенте верхнего уровня Main.

1. Выполните команду **Файл/Создать/Модель** на панели инструментов. Откроется диалоговое окно **Новая модель**.
2. В поле **Имя модели** диалогового окна **Новая модель** введите `Рем_услуги`. Выберите каталог, в котором будут сохранены файлы модели.
3. Щёлкните **Готово**.
4. Создайте область просмотра для размещения исходных данных на агенте Main. Из палитры **Презентация** перетащите элемент **Область просмотра**. На странице **Основные** панели **Свойства** в поле **Имя:** введите `Данные`.
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. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `0`, **Y:** `650`, **Ширина:** `610`, **Высота:** `590`.
8. Из палитры **Презентация** перетащите элемент **Скруглённый прямоугольник**. Оставьте имя, предложенное системой. В прямоугольнике мы разместим элементы для ввода исходных данных и вывода результатов моделирования.
9. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `10`, **Y:** `670`, **Ширина:** 590, **Высота:** `560`.
10. Перетащите элемент **text** и на странице **Текст** панели **Свойства** вместо слова **text:** введите `Исходные данные`.

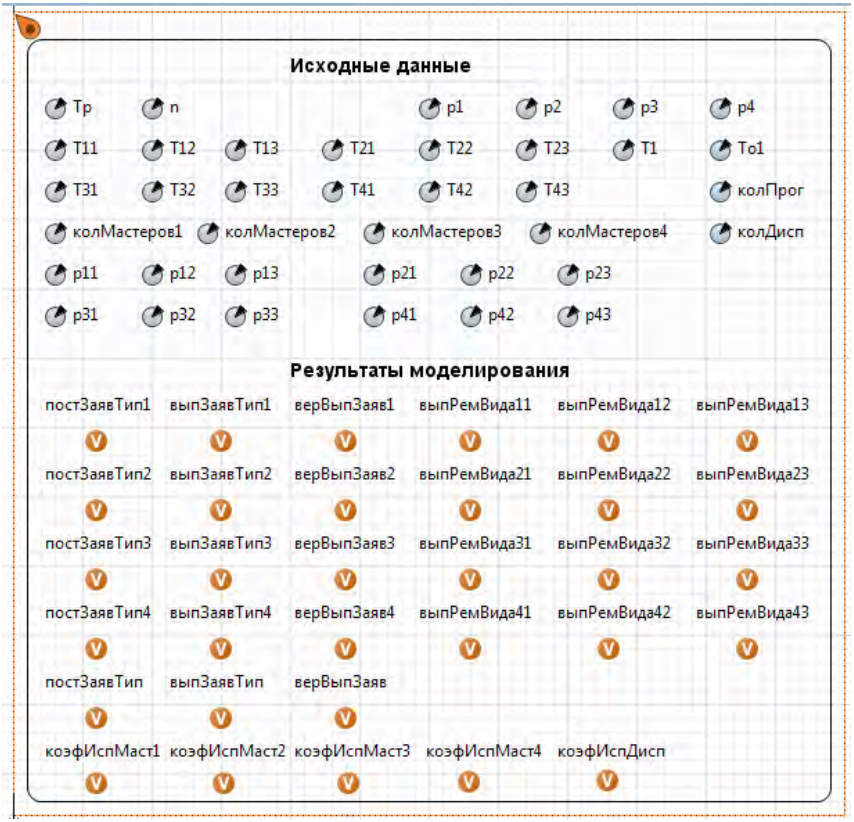


Рис. 8.2. Размещение элементов **Параметр** и **Переменная**

11. В **Палитре** выделите **Основная**. Перетащите элементы **Параметр** на элемент **Скругленныё прямоугольник**. Разместите их так, как показано на рис. 8.2. Используйте копирование.

12. Значения свойств установите согласно табл. 8.1. Во всех элементах оставьте установленным флажок **Отображать имя**.

13. Тип элементов с именами `колДисп`, `колМастеров1`…`колМастеров4` установите `int`. Тип остальных элементов — `double`.

Имена параметров оставлены практически такими же, как в постановке задачи на разработку имитационной модели. По рис. 8.2 нельзя определить, например, в Т11 русская буква Т или английская. Поэтому принимайте решение сами. Тем не менее, лучше все имена давать на одном языке.

Таблица 8.1

Элементы и их свойства			
Параметр		Параметр	
Имя	Значение по умолчанию	Имя	Значение по умолчанию
Tp n	30 4	T1 To1	15 2
T11 T12 T13	30 40 50	p11 p12 p13	0.50 0.75 1.00
T21 T22 T23	20 30 40	p21 p22 p23	0.50 0.75 1.00
T31 T32 T33	15 25 35	p31 p32 p33	0.50 0.75 1.00
T41 T42 T43	25 35 45	p41 p42 p43	0.50 0.75 1.00
колПрог	1000	колДисп	2
колМастеров1 колМастеров2	2 1	колМастеров3 колМастеров4	1 1

8.2.2. Вывод результатов моделирования

1. На **Область просмотра** мы уже перетащили **Скругленный прямоугольник**. На нём мы будем также размещать, как отмечалось ранее, элементы для вывода результатов моделирования.

2. Перетащите на него элемент **text** и на странице **Текст:** панели **Свойства** вместо слова `text` введите `Результаты моделирования`. Поместите этот текст посредине в нижней части элемента **Скругленный прямоугольник**.

3. Из палитры **Основная** перетащите элементы **Переменная**. Разместите их и дайте им имена согласно рис. 8.2. Тип всех переменных double.

8.2.3. Построение событийной части модели

Строить событийную часть модели будем последовательной реализацией средствами AnyLogic выделенных ранее сегментов (см. п. 8.1.4):

источники заявок;

диспетчеры;

мастера;

учёт выполненных заявок.

1. Создайте область просмотра для размещения элементов модели на агенте Main. Из палитры **Презентация** перетащите элемент **Область просмотра**. Перейдите на страницу **Основные** панели **Свойства**. В поле **Имя** введите `МодРемУслуги`.

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

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

4. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `0`, **Y:** `0`, **Ширина:** `1240`, **Высота:** `460`.

5. Из палитры **Презентация** перетащите элемент **Прямоугольник**. Оставьте имя, предложенное системой. В прямоугольнике мы разместим объект source для имитации поступления заявок.

6. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `20`, **Y:** `120`, **Ширина:** 100, **Высота:** `140`.

7. Перетащите элемент **text** на прямоугольник и на странице **Текст** панели **Свойства** введите `Источники заявок`.

8. Перетащите ещё один элемент **Прямоугольник**. Оставьте имя, предложенное системой. В этом прямоугольнике мы разместим объекты сегмента **Диспетчеры** для имитации работы диспетчеров.

9. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `130`, **Y:** `30`, **Ширина:** 590, **Высота:** `410`.

10. Перетащите элемент **text** на прямоугольник и на странице **Текст** панели **Свойства** введите `Диспетчеры`.

На рис. 8.3 показаны объекты двух сегментов: **Источники заявок** и **Диспетчеры**. Приступим к их построению.

8.2.3.1. Сегмент Источники заявок

1. Из **Библиотеки моделирования процессов** перетащите объект source на прямоугольник с названием **Источники заявок** (см. рис. 8.3). Создайте новый тип агента `Заявка`.

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

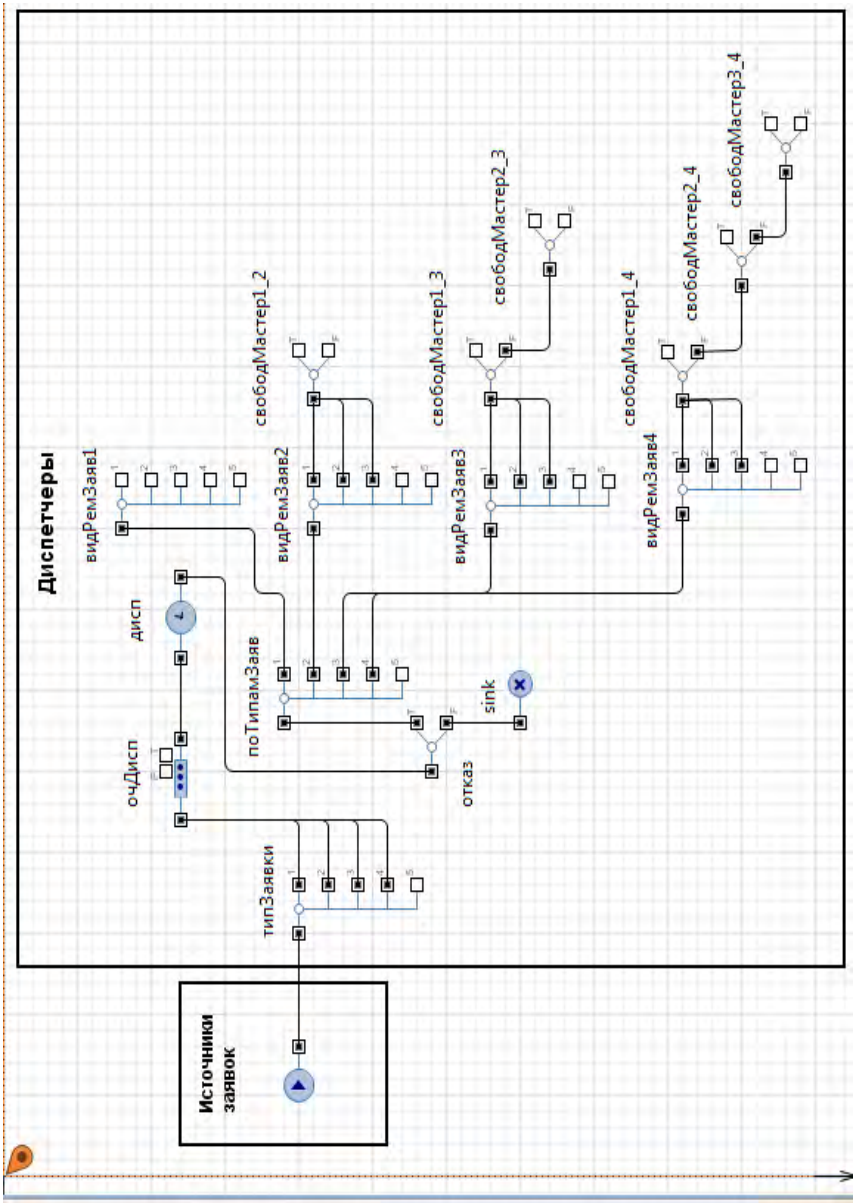


Рис. 8.3. Сегменты **Источники заявок** и **Диспетчеры**

3. Появится диалоговое окно **Новый Java класс**. В поле **Имя:** введите имя нового типа агента `Заявка`.

4. В поле **Базовый класс:** выберите из выпадающего списка `Entity` в качестве базового класса. Щёлкните **Далее**.

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

```
double типЗ;
double видР;
double времяР;
```

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

7. Щёлкните кнопку **Готово**. Выделите правой кнопкой мыши созданный **Java класс** `Заявка` и выполните в контекстном меню **Преобразовать Java класс в тип агента**.

8. Выделите объект **source**. На странице **Основные** панели **Свойства** уберите флажок **Отображать имя**. В полях **Тип заявки:** и **Новая заявка** `Agent` замените `Заявка`.

Установите:

Прибывают согласно `Времени между прибытиями`

Время между прибытиями `exponential(1/(Tp/n))`

Действия При выходе: `entity.типЗ=uniform();`
`entity.видР=uniform();`

Java-кодом в поля `entity.типЗ` и `entity.видР` заносятся равномерно распределённые случайные числа. Они нужны далее для розыгрыша кодов типов заявок и кодов видов ремонта.

8.2.3.2. Сегмент Диспетчеры

Сегмент **Диспетчеры** предназначен для распределения по группам мастеров заявок согласно их типам и видам ремонта в зависимости от занятости мастеров в текущий момент времени.

Данный сегмент реализуется шестью объектами **selectOutput5**, восемью объектами **selectOutput**, объектами **queue**, **delay** и **sink** (см. рис. 8.3).

1. Перетащите указанные объекты (или, перетащив один, скопируйте остальные, но перед копированием измените свойства, общие для всех копируемых объектов, например, класс заявки `Заявка`) из **Основной библиотеки** на диаграмму класса Main. Соедините их так, как показано на рис. 8.3.

2. Установите свойства объектов согласно табл. 8.2.

Таблица 8.2

Свойства	Значение

Имя
Использовать
Условие 1
Действия
При выходе 1
 	
Условие 2
Действия
При выходе 2
 	
Условие 3
Действия
При выходе 3
 	
Условие 4
Действия
При выходе 4
	`типЗаявки`
	Условия
	`entity.типЗ<=p1`
	
	`entity.типЗ=1;`
	`постЗаявТип1++;`
	`постЗаявТип++;`
	`entity.типЗ<=p2`
	`entity.типЗ=2;`
	`постЗаявТип2++;`
	`постЗаявТип++;`
	`entity.типЗ<=p3`
	`entity.типЗ=3;`
	`постЗаявТип3++;`
	`постЗаявТип++;`
	`entity.типЗ<=p4`
	`entity.типЗ=4;`
	`постЗаявТип4++;`
	`постЗаявТип++;`

Имя
Выход true выбирается
Вероятность
	`отказ`
	С заданной вероятностью
	`0.98`

Имя
Использовать
Условие 1
Условие 2
Условие 3
Условие 4
	`поТипамЗаяв`
	Условия
	`entity.типЗ==1`
	`entity.типЗ==2`
	`entity.типЗ==3`
	`entity.типЗ==4`

Имя
Использовать
Условие 1
Действия
При выходе 1
 	
Условие 2
Действия
При выходе 2
Условие 3
Действия
При выходе 3
	`видРемЗаяв1`
	Условия
	`entity.видР<=p11`
	
	`entity.видР=1;`
	`entity.времяР=exponential(1/T11);`
	`entity. видР<=p12`
	`entity.видР=2;`
	`entity.времяР=exponential(1/T12);`
	`entity.видР=<p13`
	`entity.видР=3;`
	`entity.времяР=exponential(1/T13);`

Продолжение табл. 8.2

Свойства	Значение

Имя
Использовать
Условие 1
Действия
При выходе 1
Условие 2
Действия
При выходе 2
Условие 3
Действия
При выходе 3
	`видРемЗаяв2`
	Условия
	`entity.видР<=р21`
	`entity.видР=1;`
	`entity.времяР=exponential(1/Т21);`
	`entity. видР<=р22`
	`entity.видР=2;`
	`entity.времяР=exponential(1/Т22);`
	`entity.видР=<р23`
	`entity.видР=3;`
	`entity.времяР=exponential(1/Т23);`

Имя
Использовать
Условие 1
Действия
При выходе 1
Условие 2
Действия
При выходе 2
Условие 3
Действия
При выходе 3
	`видРемЗаяв3`
	Условия
	`entity.видР<=р31`
	`entity.видР=1;`
	`entity.времяР=exponential(1/Т31);`
	`entity. видР<=р32`
	`entity.видР=2;`
	`entity.времяР=exponential(1/Т32);`
	`entity.видР=<р33`
	`entity.видР=3;`
	`entity.времяР=exponential(1/Т33);`

Имя
Использовать
Условие 1
Действия
При выходе 1
Условие 2
Действия
При выходе 2
Условие 3
Действия
При выходе 3
	`видРемЗаяв4`
	Условия
	`entity.видР<=р41`
	`entity.видР=1;`
	`entity.времяР=exponential(1/Т41);`
	`entity. видР<=р42`
	`entity.видР=2;`
	`entity.времяР=exponential(1/Т42);`
	`entity.видР=<р43`
	`entity.видР=3;`
	`entity.времяР=exponential(1/Т43);`

Имя
Выход true выбирается
Условие
	`свобМастер1_2`
	При выполнении условия
	
	`(очМастеров1.size()==0)&&`
	`(мастера1.size()<колМастеров1)&&`
	`(мастера2.size()!=0)`

Окончание табл. 8.2

Свойства	Значение
Имя Выход true выбирается Условие	свобМастер1_3 При выполнении условия (очМастеров1.size()==0)&& (мастера1.size()<колМастеров1)&& (мастера3.size()!=0)
Имя Выход true выбирается Условие	свобМастер1_4 При выполнении условия (очМастеров1.size()==0)&& (мастера1.size()<колМастеров1)&& (мастера4.size()!=0)
Имя Выход true выбирается Условие	свобМастер2_3 При выполнении условия (очМастеров2.size()==0)&& (мастера2.size()<колМастеров2)&& (мастера3.size()!=0)
Имя Выход true выбирается Условие	свобМастер2_4 При выполнении условия (очМастеров2.size()==0)&& (мастера2.size()<колМастеров2)&& (мастера4.size()!=0)
Имя Выход true выбирается Условие	свобМастер3_4 При выполнении условия (очМастеров3.size()==0)&& (мастера3.size()<колМастеров3)&& (мастера4.size()!=0)
Имя Макс. вместимость	очДисп Установить флажок
Имя Тип Время задержки Вместимость Действия при выходе	Дисп Определённое время normal(To1,T1) колДисп коэфИспДисп=дисп.statsUtilization.mean();

Объектом типЗаявки разыгрывается код типа заявки. Например, в поступившей заявке entity.типЗ=0.723. Проверяется условие 0: entity.типЗ=0.723<=p1=0.5. Условие 0 не выполняется. Тогда проверяется условие 1: entity.типЗ= 0.723 <=p2=0.75. Условие 1 выполняется. Заявка пропускается на выход 1. При этом выполняется код, записанный в поле **Действия При выходе 1**,

```
entity.типЗ=2;
постЗаявТип2++;
постЗаявТип++;
```

Кроме записи кода 2 в поле `entity.типЗ=2`, учитывается количество поступивших заявок 2 типа и количество всех типов поступивших заявок. Последнее в дальнейщем используется для определения вероятности выполнения заявок.

С выходов 0…3 объекта `типЗаявки` заявки поступают в `очДисп` (объект queue) с максимальной вместимостью, а затем в объект `дисп` (delay), имитирующий время работы одного из диспетчеров с одной заявкой.

Объект `отказ` (selectOutput) предназначен для розыгрыша отказа в принятии заявки с вероятностью q = 2 %. Заявки, получившие отказ, уничтожаются объектом sink.

Принятые к выполнению заявки распределяются по типам объектом `поТипамЗаяв`. С выходов 0…3 этого объекта заявки поступают на объекты `видРемЗаяв1`…`видРемЗаяв4` соответственно. Аналогичным образом как объектом `типЗаявки` этими объектами разыгрываются для заявок 1…4 типов коды видов 1…3 ремонтов.

Функции остальных объектов сегмента **Диспетчеры** рассмотрим в п. 8.1.7.3.

8.2.3.3. Сегмент Мастера

Сегмент **Мастера** предназначен для имитации ожидания освобождения мастеров, непосредственно времени выполнения соответствующего вида ремонта и отправки выполненной заявки в сегмент учёта.

Сегмент построен на четырёх объектах queue и четырёх объектах delay.

1. Из палитры **Презентация** перетащите элемент **Прямоугольник**. Оставьте имя, предложенное системой.
2. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `740`, **Y:** `30`, **Ширина:** 200, **Высота:** `410`.
3. Перетащите элемент **text** на прямоугольник и на странице **Текст** панели **Свойства** в поле вместо text введите `Мастера`.
4. Перетащите указанные объекты из **Библиотеки моделирования процессов** на агент Main. Разместите, дайте имена и соедините их так, как показано на рис. 8.4.

5. У объектов очМастеров1…очМастеров4 укажите максимальную вместимость и тип заявки Заявка.

6. У объектов мастера1…мастера4 установите свойства:

Тип заявки: Заявка

Тип Определённое время

Время задержки entity.времяР

Включить сбор статистики Установите флажок

7. Свойство **Вместимость** у этих же объектов укажите колМастеров1…колМастеров4 соответственно.

8. **Действия При выходе** установите соответственно:

```
коэфИспМаст1=мастера1.statsUtilization.mean();
коэфИспМаст2=мастера2.statsUtilization.mean();
коэфИспМаст3=мастера3.statsUtilization.mean();
коэфИспМаст4=мастера4.statsUtilization.mean();
```

С выходов 1…3 объекта видРемЗаяв1 заявки сразу поступают в объект очМастеров1 (queue).

С выходов объектов видРемЗаяв2…видРемЗаяв4 заявки поступают на объекты свобМастер1_2, свобМастер1_3, свобМастер1_4 соответственно. В принятых именах первая цифра означает группу мастеров, а вторая — тип заявки. Этими объектами проверяются условия. Например, объектом свобМастер1_3 проверяется условие:

```
(очМастеров1.size()==0)&&
(мастера1.size()<колМастеров1)&&(мастера3.size()!=0)
```

Пуста ли очередь мастеров 1 группы? И есть ли свободные мастера 1 группы? И заняты ли мастера 3 группы? Если сложное условие, состоящее из трёх простых условий, выполняется, заявка с выхода true объекта свобМастер1_3 на объект мастера1.

Аналогичные проверки осуществляются в объектах свобМастер1_3, свобМастер1_4.

Если условие не выполняется, то заявка с выходов false поступает в очередь очМастеров2…очМастеров4 соответственно.

Объекты свобМастер2_3, свобМастер2_4 проверяют возможности в текущий момент времени выполнения заявок 3 и 4 типов мастерами 2 группы.

Объект свобМастер3_4 проверяет возможность выполнения заявок 4 типа мастерами 3 группы.

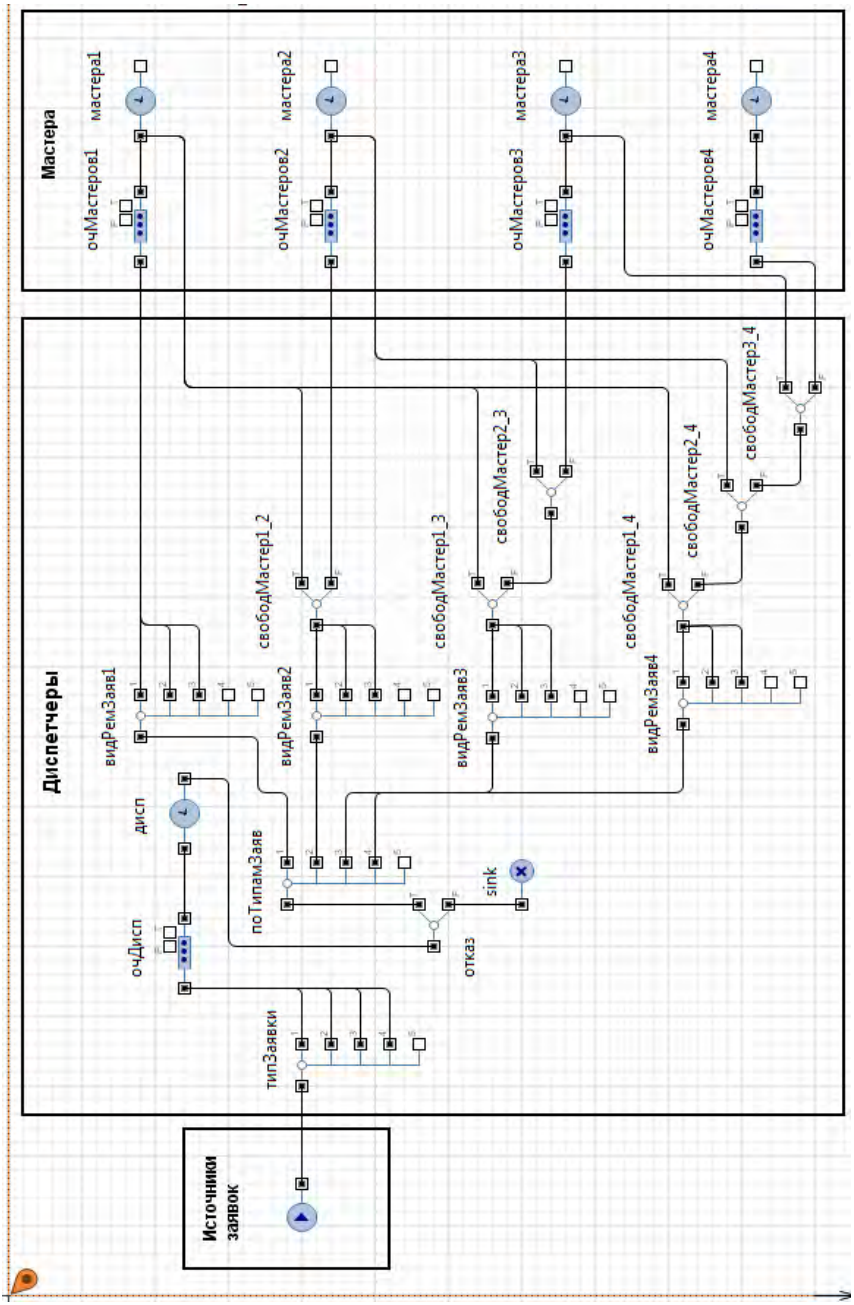


Рис. 8.4. Сегменты **Диспетчеры** и **Мастера**

8.2.3.4. Сегмент Учёт выполненных заявок

Сегмент предназначен для учёта количества выполненных заявок по типам и видам ремонтов, а также для определения вероятности выполнения заявок в целом.

Сегмент построен на пяти объектах selectOutput5 и одном объекте sink.

1. Из палитры **Презентация** перетащите элемент **Прямоугольник**. Оставьте имя, предложенное системой.
2. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `970`, **Y:** `30`, **Ширина:** 250, **Высота:** `410`.
3. Перетащите элемент **text** на прямоугольник и на странице **Текст** панели **Свойства** в поле вместо text введите `Учёт выполненных заявок`.
4. Перетащите указанные элементы на прямоугольник. Разместите, соедините и дайте имена согласно рис. 8.5.

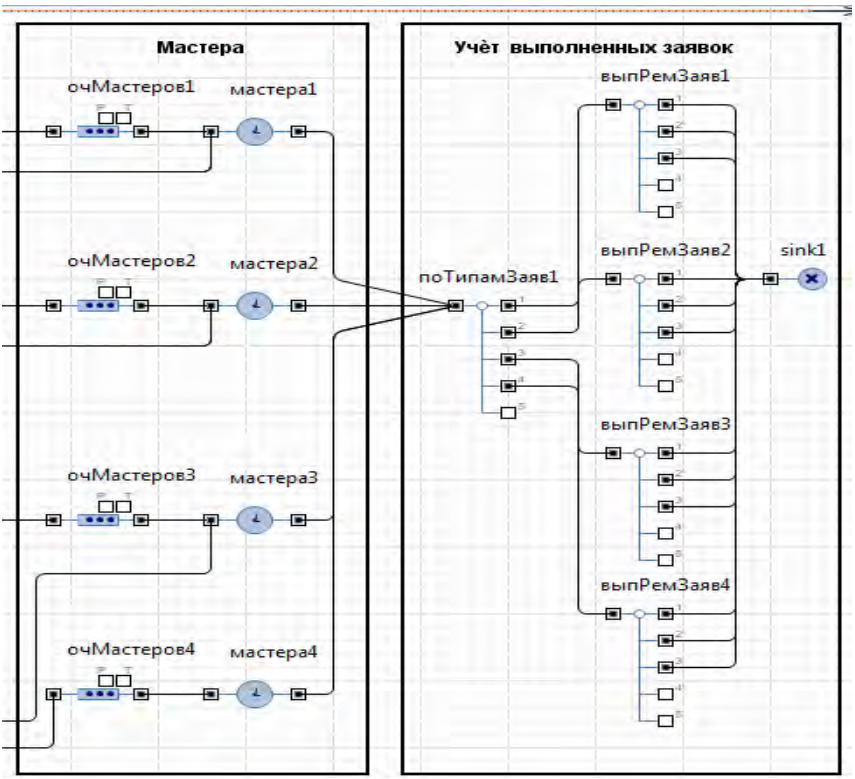


Рис. 8.5. Сегменты **Мастера** и **Учёт выполненых заявок**

5. Свойства элементов установите согласно табл. 8.3.

Таблица 8.3

Свойства	Значение

Имя
Использовать
Условие 1
Действия
При выходе 1
 	
Условие 2
Действия
При выходе 2
 	
Условие 3
Действия
При выходе 3
 	
Условие 4
Действия При выходе 4
	`поТипамЗаяв1`
	Условия
	`entity.типЗ==1`
	`выпЗаявТип1++;`
	`выпЗаявТип++;`
	`верВыпЗаяв1=`
	`выпЗаявТип1/постЗаявТип1;`
	`entity.типЗ==2`
	`выпЗаявТип2++;`
	`выпЗаявТип++;`
	`верВыпЗаяв2=`
	`выпЗаявТип2/постЗаявТип2;`
	`entity.типЗ==3`
	`выпЗаявТип3++;`
	`выпЗаявТип++;`
	`верВыпЗаяв3=`
	`выпЗаявТип3/постЗаявТип3;`
	`entity.типЗ==4`
	`выпЗаявТип4++;`
	`выпЗаявТип++;`
	`верВыпЗаяв4=`
	`выпЗаявТип4/постЗаявТип4;`

Имя
Использовать
Условие 1
Действия При выходе 1
Условие 2
Действия При выходе 2
Условие 3
Действия При выходе 3
	`выпРемЗаяв1`
	Условия
	`entity.видР==1`
	`выпРемВида11++;`
	`entity.видР==2`
	`выпРемВида12++;`
	`entity.видР==3`
	`выпРемВида13++;`

Имя
Использовать
Условие 1
Действия При выходе 1
Условие 2
Действия При выходе 2
Условие 3
Действия При выходе 3
	`выпРемЗаяв2`
	Условия
	`entity.видР==1`
	`выпРемВида21++;`
	`entity.видР==2`
	`выпРемВида22++;`
	`entity.видР==3`
	`выпРемВида23++;`

Окончание табл. 8.3

Свойства	Значение

Имя
Использовать
Условие 1
Действия При выходе 1
Условие 2
Действия При выходе 2
Условие 3
Действия При выходе 3
	`выпРемЗаяв3`
	Условия
	`entity.видР==1`
	`выпРемВида31++;`
	`entity.видР==2`
	`выпРемВида32++;`
	`entity.видР==3`
	`выпРемВида33++;`

Имя
Использовать
Условие 1
Действия При выходе 1
Условие 2
Действия При выходе 2
Условие 3
Действия При выходе 3
	`выпРемЗаяв4`
	Условия
	`entity.видР==1`
	`выпРемВида41++;`
	`entity.видР==2`
	`выпРемВида42++;`
	`entity.видР==3`
	`выпРемВида43++;`

6. Установите значения свойств объекта sink1:

Тип заявки: `Заявка`

Действие при входе `верВыпЗаяв=выпЗаявТип/постЗаявТип`

Объект `поТипамЗаяв1` осуществляет разделение и учёт выполненных заявок по типам, а также рассчитывает вероятности выполнения заявок каждого типа. В количество поступивших заявок, а, следовательно, и в расчёт вероятностей входят и те заявки, которым отказано в обслуживании.

Объекты `выпРемЗаяв1`…`выпРемЗаяв4` учитывают количество видов ремонтов, выполненных по заявкам каждого типа.

Объект sink1 уничтожает поступающие заявки. Введённый в поле **Действие при входе** код рассчитывает вероятность выполнения всех заявок.

8.2.3.5. Отладка модели

Построение модели закончено. Выделите в окне **Проекты** Simulation:Main. На странице **Основные** установите **Фиксированное начальное число (воспроизводимые прогоны)** и **Начальное число:** `892`. Перейдите на страницу **Модельное время**, выберите из списка **Остановить:** `В заданное время`. Введите **Конечное время:** 1440000 (модельное время ↑ в 1000).

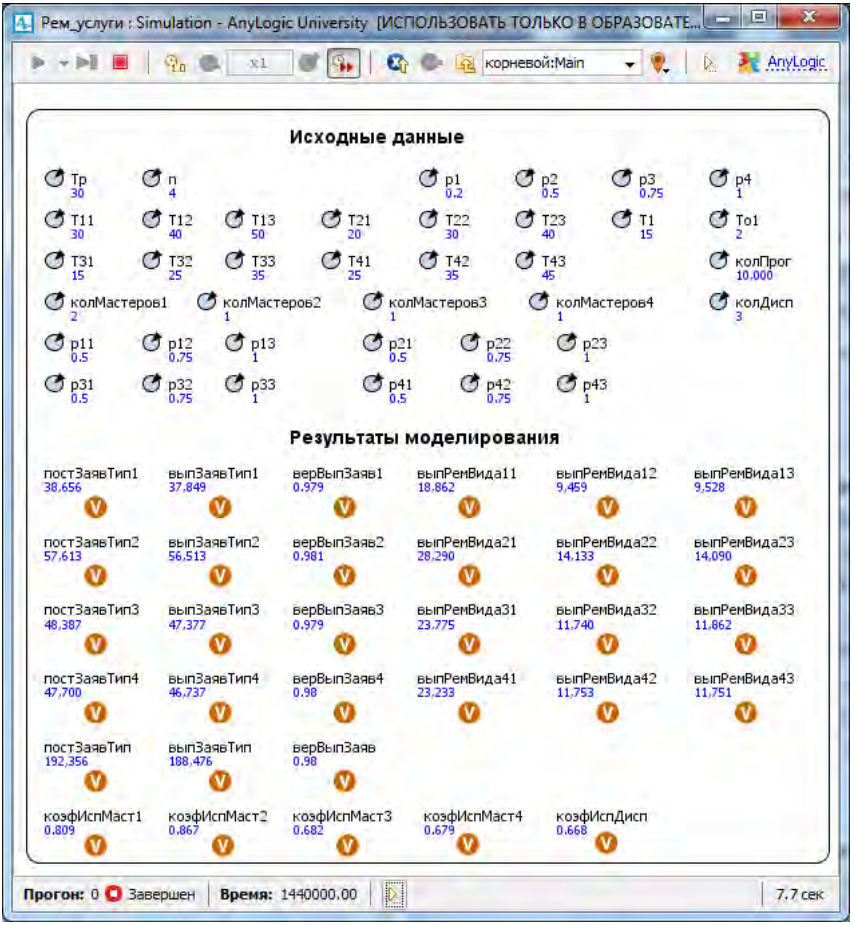


Рис. 8.6. Результаты моделирования

Запустите модель. Если вы всё делали согласно нашим рекомендациям, то ошибок не будет.

По завершении работы модели или в ходе её перейдите на область просмотра **Данные_Результаты**. Поскольку мы для переключения между областями просмотра своего ничего не делали, используйте приём, предлагаемый AnyLogic.

Результаты моделирования при исходных данных согласно постановке задачи должны быть такими как на рис. 8.6. Например, заявок 1 типа выполнено 37,612. Вероятность выполнения составляет 0, 979. Всего выполнено заявок всех типов 188,476 с вероятностью 0,980.

8.3. Интерпретация результатов моделирования

Всего выполнены три группы экспериментов по три эксперимента в каждой группе. Результаты экспериментов каждой группы представлены в табл. 8.4…8.6 соответственно.

В первой группе экспериментов (табл. 8.4) все исходные данные соответствуют постановке задачи, кроме среднего интервала времени поступления заявок, который уменьшен: Тр = 20 мин.

Как и в предыдущих экспериментах сравнивать будем результаты моделирования, полученные в GPSS World и в AnyLogic7.

Например, разница вероятностей выполнения всех заявок составляет Δ_5 = 0,001, коэффициентов использования мастеров всех групп — Δ_6 = 0,005…0,013. Количество выполненных заявок и ремонтов, если считать с точностью до целого, одно и тоже в обеих системах моделирования.

Во второй группе экспериментов (табл. 8.5) все исходные данные соответствуют постановке задачи, то есть средний интервал времени поступления заявок Тр = 30 мин.

Во второй группе разница вероятностей выполнения всех заявок составляет Δ_5 = 0,001, коэффициентов использования мастеров всех групп практически такое же — Δ_6 = 0,001…0,014. Количество выполненных заявок и ремонтов, так же если считать с точностью до целого, одно и тоже.

Аналогичные выводы можно сделать и по третьей группе экспериментов (табл. 8.6), в которой были изменены следующие данные в сторону увеличения количества диспетчеров и мастеров второй и третьей групп: Тр = 20 мин, колДисп = 3, колМастеров2 = 2, колМастеров3 = 2.

Здесь также вероятность выполнения всех заявок отличается незначительно Δ_5 = 0,001, но несколько больше разница между коэффициентами занятости всех групп мастеров-ремонтников: Δ_5 = 0,015…0,026.

Увеличение количества мастеров-ремонтников второй и третьей групп с одного до двух в каждой группе позволило по сравнению со второй группой экспериментов увеличить количество выполненных заявок в абсолютном выражении с 240,075 до 281,239 и относительном выражении в 1,174 раза (281,239: 240,075 = 1,174). При этом незначительно выросла вероятность выполнения заявок: с 0,977 до 0,980. Увеличилась загрузка мастеров-ремонтников всех групп.

Таблица 8.4

Показатели фирмы предоставления ремонтных услуг

Показатели	GPSS World	AnyLogic6	AnyLogic7
Среднее время поступления заявок 20 мин			
Выполнено заявок типа 1	37,879	37,428	37,707
Δ_1 = |37,879-37,707| = 0,172			
ремонтов: вида 11	19,021	18,705	18,697
вида 12	9,461	9,354	9,353
вида 13	9,397	9,369	9,257
Выполнено заявок типа 2	56,297	56,330	56,524
Δ_2 = |56,297-56,524| = 0,227			
ремонтов: вида 21	28,195	28,378	28,517
вида 22	13,900	13,914	14,026
вида 23	14,202	14,038	13,981
Выполнено заявок типа 3	47,075	46,946	47,322
Δ_3 = |47,075-47,322| = 0,247			
ремонтов: вида 31	23,570	23,414	23,784
вида 32	11,831	11,788	11,711
вида 33	11,674	11,744	11,827
Выполнено заявок типа 4	46,857	47,486	47,000
Δ_4 = |46,857-47,000| = 0,143			
ремонтов: вида 41	23,443	23,552	23,527
вида 42	11,811	12,050	11,886
вида 43	11,603	11,884	11,587
Вероятность выполнения всех заявок	0,654	0,655	0,653
Δ_5 = |0,654-0,653| = 0,001			
Коэффициенты использования: мастеров 1	0,832	0,838	0,833
мастеров 2	0,865	0,868	0,866
мастеров 3	0,659	0,668	0,664
мастеров 4	0,637	0,662	0,650
Δ_6 = 0,001…0,013			
Коэффициент использования диспетчеров	1,000	1,000	1,000

Таблица 8.5

Показатели фирмы предоставления ремонтных услуг

Показатели	GPSS World	AnyLogic6	AnyLogic7
Среднее время поступления заявок 20 мин			
Выполнено заявок типа 1	37,490	37,466	37,612
Δ_1 = |37,490-37,612| = 0,192			
ремонтов: вида 11	18,705	18,846	18,893
вида 12	9,325	9,236	9,283
вида 13	9,460	9,384	9,436
Выполнено заявок типа 2	56,597	56,360	56,279
Δ_2 = |56,597-56,279| = 0,318			
ремонтов: вида 21	28,186	27,941	28,014
вида 22	14,341	14,200	14,070
вида 23	14,070	14,219	14,195
Выполнено заявок типа 3	46,811	47,151	46,980
Δ_3 = |46,811-46,980| = 0,169			
ремонтов: вида 31	23,373	27,941	23,680
вида 32	11,842	14,200	11,662
вида 33	11,596	14,219	11,638
Выполнено заявок типа 4	47,223	47,151	46,887
Δ_4 = |47,223-46,887| = 0,336			
ремонтов: вида 41	23,647	23,311	23,445
вида 42	11,844	11,791	11,801
вида 43	11,732	11,807	11,641
Вероятность выполнения всех заявок	0,978	0,978	0,977
Δ_5 = |0,978-0,977| = 0,001			
Коэффициенты использования: мастеров 1	0,828	0,836	0,832
мастеров 2	0,858	0,868	0,864
мастеров 3	0,653	0,667	0,649
мастеров 4	0,632	0,657	0,633
Δ_6 = 0,001…0,014			
Коэффициент использования диспетчеров	1,000	0,999	0,998

Таблица 8.6

Показатели фирмы предоставления ремонтных услуг

Показатели	GPSS World	AnyLogic6	AnyLogic7
Среднее время поступления заявок 20 мин			
Выполнено заявок типа 1	56,592	56,221	55,849
Δ_1 = |56,592-55,849| = 0,743			
ремонтов: вида 11	28,398	28,069	28,075
вида 12	14,158	14,152	13,889
вида 13	14,036	14,000	13,885
Выполнено заявок типа 2	84,227	84,740	84,592
Δ_2 = |84,227-84,592| = 0,355			
ремонтов: вида 21	42,087	42,513	42,411
вида 22	21,196	21,285	21,026
вида 23	20,974	20,942	21,155
Выполнено заявок типа 3	70,679	70,586	70,377
Δ_3 = |70,679-70,377| = 0,302			
ремонтов: вида 31	35,256	35,072	35,147
вида 32	17,671	17,659	17,662
вида 33	17,752	17,855	17,568
Выполнено заявок типа 4	70,244	70,486	70,421
Δ_4 = |70,244-70,421| = 0,177			
ремонтов: вида 41	35,112	34,974	35,344
вида 42	17,757	17,696	17,491
вида 43	17,375	17,816	17,586
Вероятность выполнения всех заявок	0,979	0,980	0,980
Δ_5 = |0,979-0,979| = 0,001			
Коэффициенты использования: мастеров 1	0,932	0,949	0,947
мастеров 2	0,874	0,893	0,891
мастеров 3	0,667	0,646	0,641
мастеров 4	0,772	0,796	0,793
Δ_6 = |0,932-0,947| = 0,015			
Коэффициент использования диспетчеров	0,998	0,999	0,996

ГЛАВА 9. МОДЕЛЬ ФУНКЦИОНИРОВАНИЯ СИСТЕМЫ ВОЗДУШНЫХ ПЕРЕВОЗОК

9.1. Модель в AnyLogic

9.1.1. Постановка задачи

Система авиаперевозок включает два аэропорта. Перевозки выполняются из первого аэропорта во второй и обратно. Время полета между аэропортами распределено по нормальному закону.

Грузы в каждый аэропорт поступают партиями в контейнерах. Количество контейнеров в партии распределено по равномерному закону. Интервалы времени между поступлениями партий грузов распределены по экспоненциальному закону.

Для авиаперевозок используют два типа самолетов А и Б и грузоподъемностями q1 и q2 шт. контейнеров соответственно (q1 < q2). В аэропорту нет фиксированного расписания. Каждый самолет отправляется в полет сразу после его полной загрузки. Время погрузки и время выгрузки одного контейнера распределены по экспоненциальному закону. В первую очередь используются самолеты типа А, а при их отсутствии — типа Б.

9.1.2. Исходные данные

Таблица 9.1

Характеристики	Аэропорты	
	1	2
Количество самолётов типа А, шт.	2	0
Грузоподъёмность самолёта типа А, шт.	50	50
Количество самолётов типа Б, шт.	1	0
Грузоподъёмность самолёта типа Б, шт.	100	100
Минимальное количество контейнеров в партии, шт.	1	1
Максимальное количество контейнеров в партии, шт.	15	22
Средний интервал поступления партий грузов, час	0,5	0,4
Количество одновременно погружаемых контейнеров в самолёт типа А, шт.	2	2
Количество одновременно погружаемых контейнеров в самолёт типа Б, шт.	2	2
Количество одновременно выгружаемых контейнеров из самолёта типа А, шт.	3	2
Количество одновременно выгружаемых контейнеров из самолёта типа Б, шт.	3	2

Окончание табл. 9.1

Характеристики	Аэропорты	
	1	2
Среднее время погрузки контейнера в самолёт типа А, час	0,1	0,2
Среднее время погрузки контейнера в самолёт типа Б, час	0,1	0,1
Среднее время выгрузки контейнера из самолёта типа А, час	0,2	0,2
Среднее я время выгрузки контейнера из самолёта типа Б, час	0,2	0,2
Среднее время полета самолёта типа А из аэропорта 1 в аэропорт 2 и из аэропорта 2 в аэропорт 1, час	3,4	3,6
Стандартное отклонение времени полёта самолёта типа А из аэропорта 1 в аэропорт 2 и из аэропорта 2 в аэропорт 1, час	0,5	0,6
Среднее время полета самолёта типа Б из аэропорта 1 в аэропорт 2 и из аэропорта 2 в аэропорт 1, час	3,2	4,1
Стандартное отклонение времени полёта самолёта типа Б из аэропорта 1 в аэропорт 2 и из аэропорта 2 в аэропорт 1, час	0,5	0,8

9.1.3. Задание на исследование

Построить имитационную модель функционирования системы воздушных перевозок с целью определения следующих её показателей:

коэффициенты доставки грузов самолётами типов А и Б в аэропорты 1 и 2;

коэффициент доставки грузов системой перевозок в целом;

коэффициенты использования самолётов типов А и Б в аэропортах 1 и 2;

коэффициент использования самолётов системой перевозок в целом;

коэффициенты использования средств погрузки, выгрузки в аэропортах 1 и 2 и самолётов при выполнении ими полётов.

Исследовать влияние на показатели функционирования системы воздушных перевозок её характеристик, выявить среди них существенные и несущественные.

Сделать выводы о построении эффективной системы воздушных перевозок и возможных направлениях совершенствования после введения в эксплуатацию.

9.1.4. Формализованное описание модели

Представим систему воздушных перевозок в виде СМО (рис. 9.1).

Система воздушных перевозок представляет собой многофазную многоканальную СМО сложной структуры с различными видами заявок. Модель, исходя из такой структуры, должна состоять из двух частей:

имитация функционирования аэропорта 1;

имитация функционирования аэропорта 2.

В каждую из этих частей модели нужно включить следующие сегменты:

имитация функционирования аэропорта 1:

прибытие самолётов в аэропорт 1, ожидание погрузки;

поступление и учёт грузов в аэропорту 1;

погрузка грузов в аэропорту 1;

полёт из аэропорта 1 в аэропорт 2;

ожидание разгрузки в аэропорту 1;

разгрузка самолётов в аэропорту 1;

имитация функционирования аэропорта 2:

поступление и учёт грузов в аэропорту 2;

ожидание разгрузки в аэропорту 2;

разгрузка самолётов в аэропорту 2;

ожидание погрузки в аэропорту 2;

погрузка грузов в аэропорту 2;

полёт из аэропорта 2 в аэропорт 1.

В модели с точки зрения интерпретации целесообразно рассматривать заявки трёх видов:

заявки как транспортные средства — самолёты;

заявки как поступающие грузы в аэропорт 1;

заявки как поступающие грузы в аэропорт 2.

Заявки как транспортные средства — самолёты должны иметь следующие параметры (поля):

`типТрансп` — код типа транспортного средства — самолёта;

`колГрузоМест` — количество груза (контейнеров) в загруженном транспортном средстве — самолёте;

`врПолёта` — время полёта самолёта из аэропорта отправления в аэропорт назначения;

`разные` — другие характеристики процесса перевозки грузов воздушным транспортом.

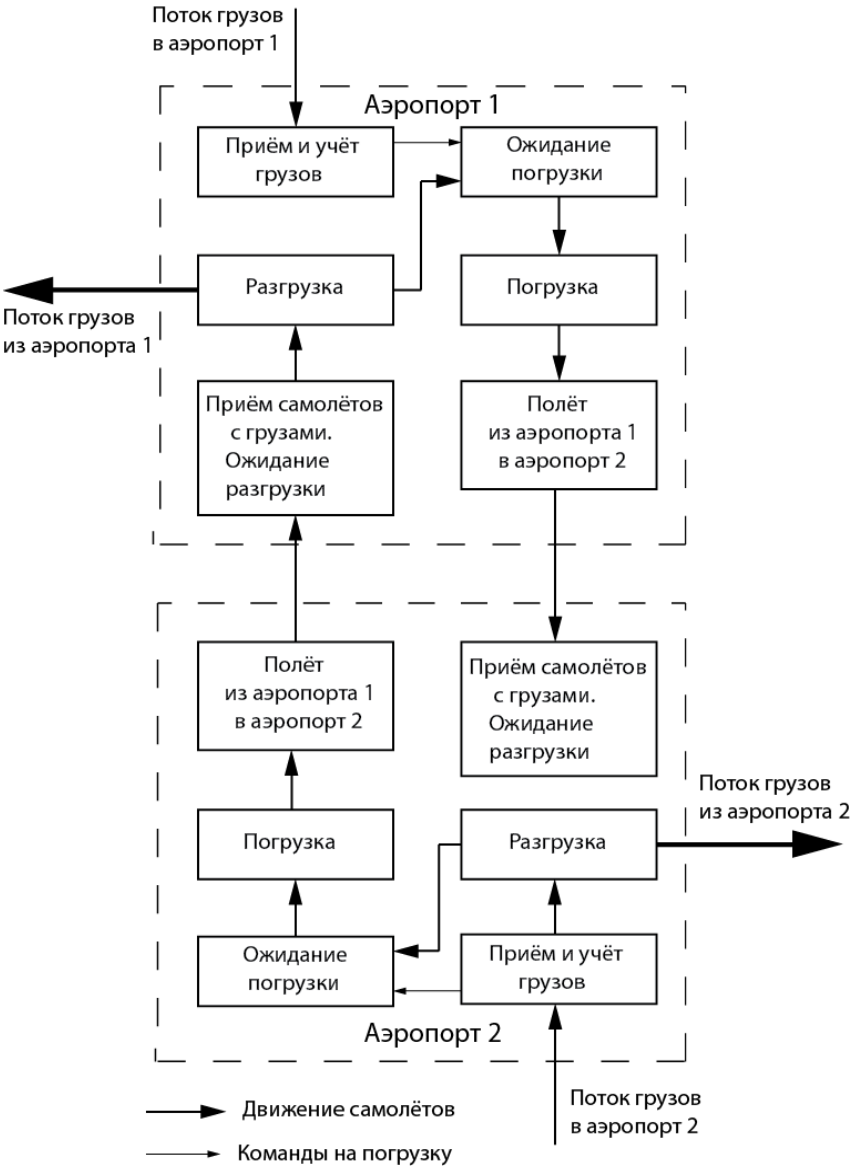


Рис. 9.1. Система воздушных перевозок как СМО

В постановке задачи транспортные средства — самолёты определены двух типов А и Б. Для идентификации этих типов в поле `типТрансп` следует использовать коды 1 и 2 соответственно.

Для фиксации количества поступающих грузов в аэропорты 1 и 2 в соответствующих заявках следует использовать поля `колГрузоМест1` и `колГрузоМест2`.

Для параметров — исходных данных и для показателей функционирования системы воздушных перевозок разработаны идентификаторы (п. 9.1.6 и п. 9.1.7 соответственно).

Показатели системы воздушных перевозок разделены на две группы:

показатели, рассчитываемые системой моделирования встроенными средствами;

показатели, рассчитываемые по формулам разработчика.

В первую группу включены следующие показатели:

`коэфПогр1А`, `коэфПогр1Б`, `коэфПогр2А`, `коэфПогр2Б` — коэффициенты использования средств погрузки при погрузке в самолёты типов А и Б в аэропортах 1 и 2 соответственно;

`коэфРазгр1А`, `коэфРазгр1Б`,`коэфРазгр2А`,`коэфРазгр2Б` — коэффициенты использования средств разгрузки при разгрузке самолётов типов А и Б в аэропортах 1 и 2 соответственно;

`коэфПолётА12`, `коэфПолётБ12`, `коэфПолётА21`, `коэфПолётБ21` — коэффициенты нахождения самолётов типов А и Б в полётах из аэропорта 1 в аэропорт 2 и из аэропорта 2 в аэропорт 1 соответственно.

Вторую группу составляют следующие показатели:

`коэфДост21=достК21/всегоПостК2` — коэффициент доставки грузов из аэропорта 2 в аэропорт 1, `достК21` — количество доставленного груза из аэропорта 2 в аэропорт 1, `всегоПостК2` — количество всего поступившего груза в аэропорт 2;

`коэфДост12=достК12/всегоПостК1` — коэффициент доставки грузов из аэропорта 1 в аэропорт 2, `достК12` — количество доставленного груза из аэропорта 1 в аэропорт 2, `всегоПостК1` — количество всего поступившего груза в аэропорт 1;

`коэфДост=(достК12+достК21)/(всегоПостК1+всегоПостК2)` — коэффициент доставки грузов системой перевозок в целом;

`коэфИспСам1А=коэфПогр1А+коэфРазгр1А+коэфПолётА12` — коэффициент использования самолётов типа А в аэропорту 1;

`коэфИспСам1Б=коэфПогр1Б+коэфРазгр1Б+коэфПолётБ12` — коэффициент использования самолётов типа Б в аэропорту 1;

`коэфИспСам2А`, `коэфИспСам2Б` — коэффициенты использования самолётов типа А и Б в аэропорту 2.

9.1.5. Создание областей просмотра

Создайте следующие области просмотра, на которых вы будете помещать сегменты имитационной модели:

исхДанные;

Результаты;

Аэропорт1;

Аэропорт2.

При необходимости вы можете создать и другие области просмотра в зависимости от возможностей вашего компьютера и монитора. При этом можете пользоваться встроенными средствами перехода к областям просмотра или добавить свои соответствующие элементы управления.

Значения свойств элементов **Область просмотра** установите согласно табл. 9.2.

Таблица 9.2

Свойства	Области просмотра			
Имя:	исхДанные	Результаты	Аэропорт1	Аэропорт1
X:	10	10	10	10
Y:	1860	2600	10	790
Ширина:	620	530	1080	1110
Высота:	490	470	480	490

Для всех областей просмотра оставьте **Выравнивать по:** Верхнему левому углу, установите из списка **Масштабирование:** Подогнать под окно.

9.1.6. Ввод исходных данных

Организуйте ввод исходных данных для модели в одном месте. Для удобства пользования, например, при модификации исходных данных, все их целесообразно разделить на две группы по признаку принадлежности к аэропорту 1 и аэропорту 2.

1. Перетащите элемент **Скруглённый прямоугольник** на элемент **Область просмотра** с именем исхДанные.
2. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** 30, **Y:** 1880, **Ширина:** 580, **Высота:** 450.
3. Перетащите элемент **text** и в поле **Текст:** введите Исходные данные.
4. Из библиотеки **Основная** перетащите элементы **Параметр** и поместите их так, как на рис. 9.2.
5. Значения свойств установите согласно табл. 9.3.

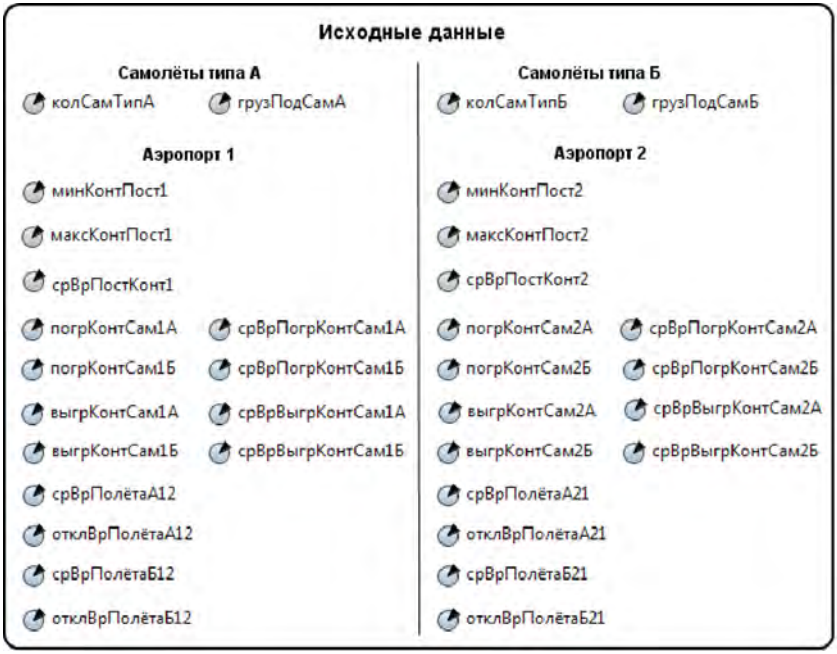


Рис. 9.2. Элементы **Параметр** для ввода исходных данных

Таблица 9.3

Имя	Тип	Значение по умолчанию
Самолёты типа А		
колСамТипА	int	2
грузПодСамА	int	50
Аэропорт 1		
минКонтПост1	int	1
максКонтПост1	int	15
срВрПостКонт1	double	0.5
погрКонтСам1А	int	2
погрКонтСам1Б	int	2
выгрКонтСам1А	int	3
выгрКонтСам1Б	int	3
срВрПогрКонтСам1А	double	0.1
срВрПогрКонтСам1Б	double	0.1
срВрВыгрКонтСам1А	double	0.2
срВрВыгрКонтСам1Б	double	0.2
срВрПолётаА12	double	3.4
отклВрПолётаА12	double	0.5

Продолжение табл. 9.3

Имя	Тип	Значение по умолчанию
срВрПолётаБ12 отклВрПолётаБ12	double double	3.2 0.5
Самолёты типа Б		
колСамТипБ	int	1
грузПодСамБ	int	100
Аэропорт 2		
минКонтПост2 максКонтПост2 срВрПостКонт2 погрКонтСам2А погрКонтСам2Б выгрКонтСам2А выгрКонтСам2Б срВрПогрКонтСам2А срВрПогрКонтСам2Б срВрВыгрКонтСам2А срВрВыгрКонтСам2Б срВрПолётаА12 отклВрПолётаА12 срВрПолётаБ12 отклВрПолётаБ12	int int double int int int int double double double double double double double double	1 22 0.4 2 2 2 2 0.2 0.1 0.2 0.2 3.6 0.6 4.1 0.8

9.1.7. Вывод результатов моделирования

Организуйте вывод результатов моделирования. Их также, как и исходные данные, следует разбить на две группы по признаку принадлежности к соответствующему аэропорту, выделить итоговые показатели. На этой области просмотра можно поместить и вспомогательные переменные, предназначенные для накопления необходимых для расчёта показателей статистических данных.

1. Перетащите элемент **Скруглённый прямоугольник** на элемент **Область просмотра** с именем `Результаты`.

2. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `25`, **Y:** 2610, **Ширина:** `510`, **Высота:** `450`.

3. Перетащите элемент **text** и на сранице **Текст:** введите `Результаты моделирования`.

4. Из библиотеки **Основная** перетащите элементы **Переменная** и поместите их так, как на рис. 9.3. Для всех переменных сохраните предлагаемый тип double, значение по умолчанию — 0.

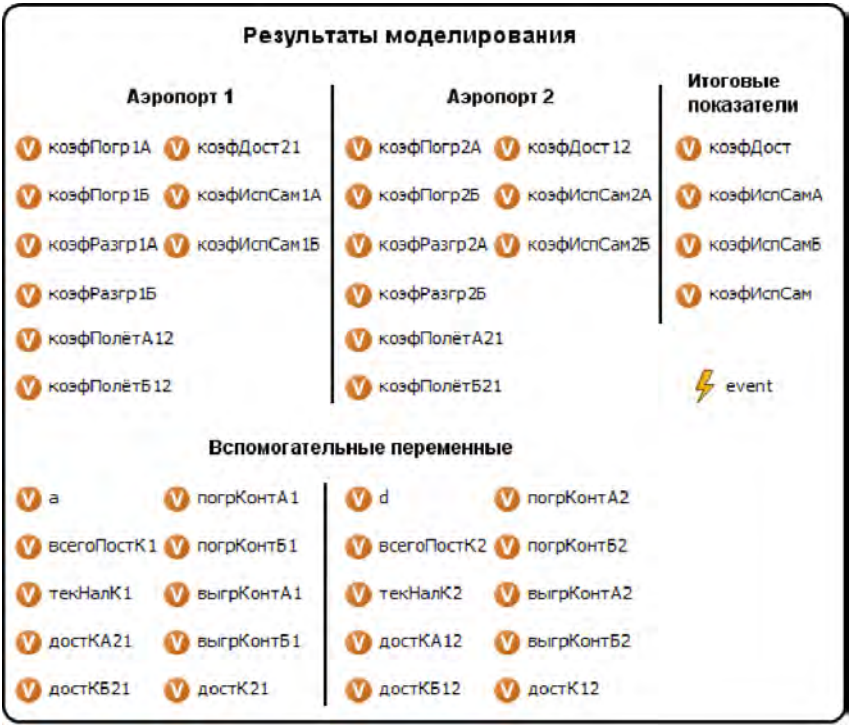


Рис. 9.3. Элементы **Переменная** для вывода результатов моделирования

5. Перетащите ещё элементы **text** и введите пояснения к результатам моделирования как на рис. 9.3.

6. Перетащите элемент **text** и в поле **Текст:** введите `Вспомогательные переменные`.

7. Перетащите элементы **Переменная** и поместите их согласно рис. 9.3. Для всех переменных оставьте предлагаемый тип double, значение по умолчанию — 0.

8. С целью сокращения машинного времени для вывода результатов моделирования используйте способ **Событие** (event). Код, который необходим для обработки статистических данных при наступлении события, вы напишите после построения событийной части модели.

Приступайте к построению событийной части модели, которая в соответствии со структурой системы воздушных перевозок включает имитацию функционирования аэропорта 1 (рис. 9.4) и аэропорта 2 (рис. 9.5).

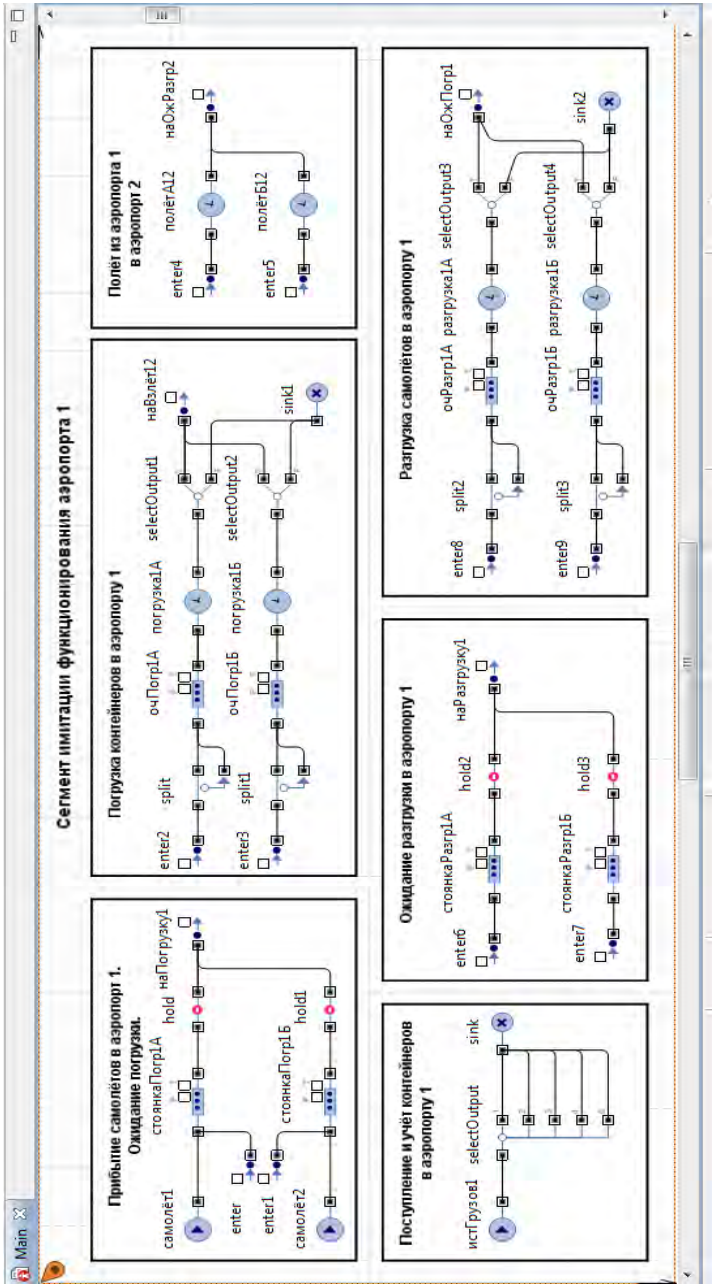


Рис. 9.4. Сегмент имитации функционирования аэропорта 1

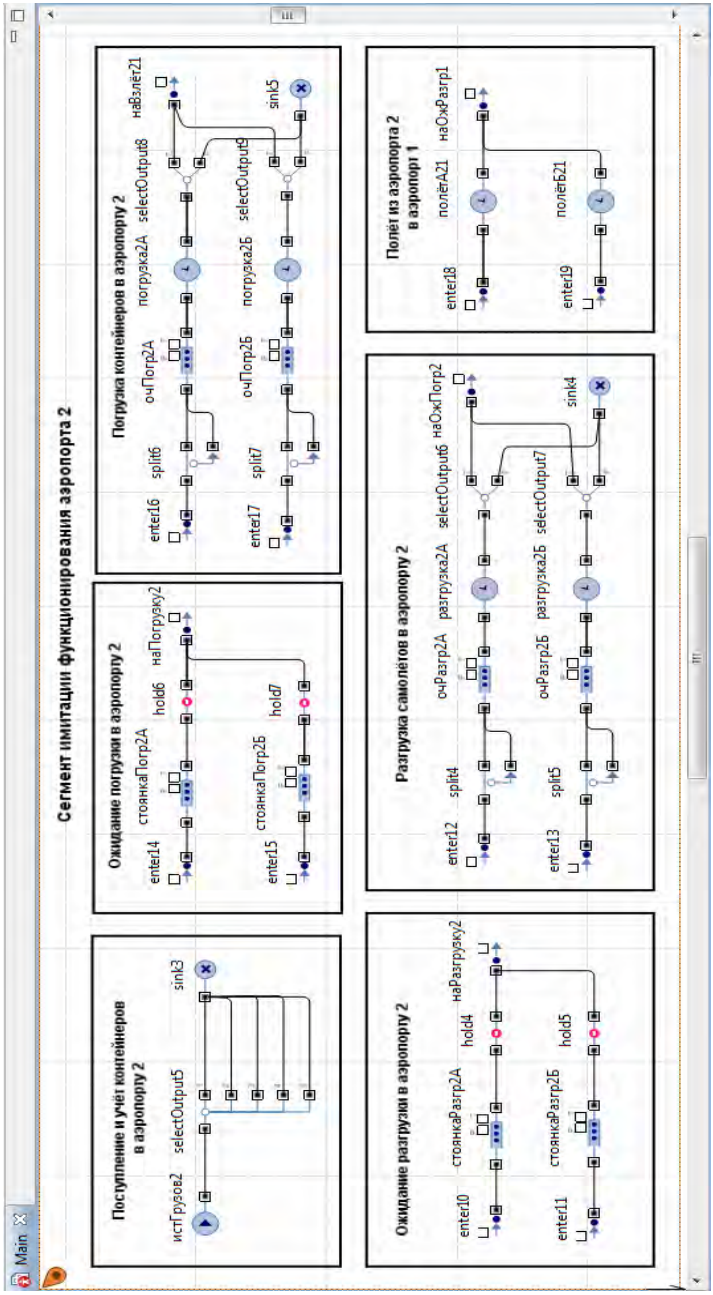


Рис. 9.5. Сегмент имитации функционирования аэропорта 2

9.1.8. Имитация функционирования аэропорта 1

9.1.8.1. Прибытие самолётов в аэропорт 1. Ожидание погрузки

Сегмент предназначен для имитации первоначального (последующего) прибытия самолётов в аэропорт 1, размещения их на стоянках, ожидания и отправки на погрузку (разгрузку) контейнеров.

1. Из палитры **Презентация** перетащите элемент **Прямоугольник**.
2. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `30`, **Y:** `50`, **Ширина:** `310`, **Высота:** `200`.
3. Перетащите элемент **text** и в поле **Текст:** введите `Прибытие самолётов в аэропорт 1. Ожидание погрузки`.
4. Перетащите из **Библиотеки моделирования процессов** по два объекта **source**, **enter**, **queue**, **hold** и один объект **exit**. Поместите и соедините их так, как на рис. 9.6.

При построении модели вам придётся воспользоваться Java-кодом, в котором потребуются дополнительные поля заявок. Для этого вы должны создать нестандартный тип заявки с дополнительными полями для записи и хранения параметров, о которых упоминалось ранее (см. п. 9.1.4).

Создайте тип заявок `ТранспСредство`.

1. Выделите объект **source**.
2. В панели **Проект** щёлкните правой кнопкой мыши элемент модели верхнего уровня дерева и выберите **Создать/Java класс**.
3. Появится диалоговое окно **Новый Java класс**. В поле **Имя:** введите имя нового класса: `ТранспСредство`.
4. В поле **Базовый класс:** выберите из выпадающего списка `Entity` в качестве базового класса. Щёлкните кнопку **Далее**.
5. Появится вторая страница **Мастера создания Java класса**. Добавьте поля Java класса, показанные на рис. 9.7.
6. Оставьте флажки **Создать конструктор** и **Создать метод toString ()**. Щёлкните **Готово**. Закройте редактор кода.
7. Щёлкните правой кнопкой созданный Java класс и из меню выберите **Преобразовать Java класс в тип агента**.
8. Аналогичным образом создайте типы заявок `ГрузАэропорт1` и `ГрузАэропорт2` с дополнительными полями `колГрузоМест1` и `колГрузоМест2` типа `int` соответственно. Эти классы заявок, как вы помните, будут использоваться при имитации поступления партий грузов в аэропорты 1 и 2 соответственно.

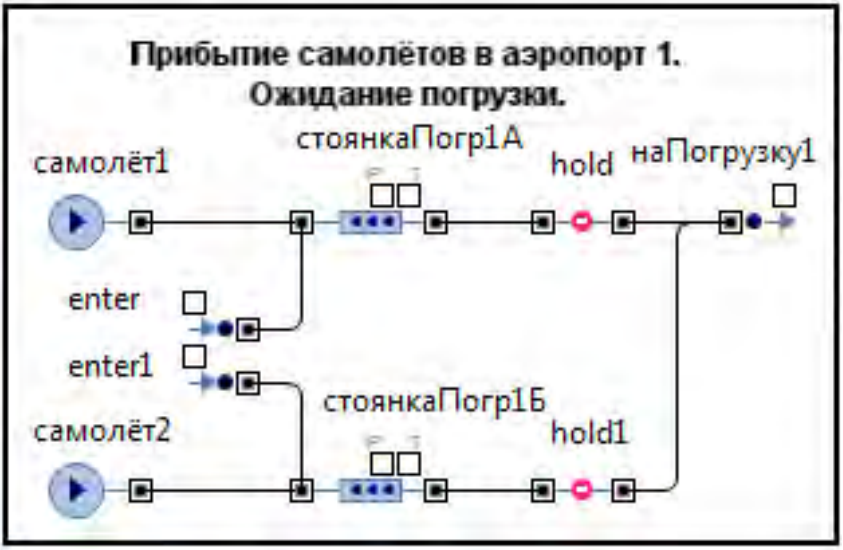


Рис. 9.6. Сегмент **Прибытие самолётов в аэропорт 1**

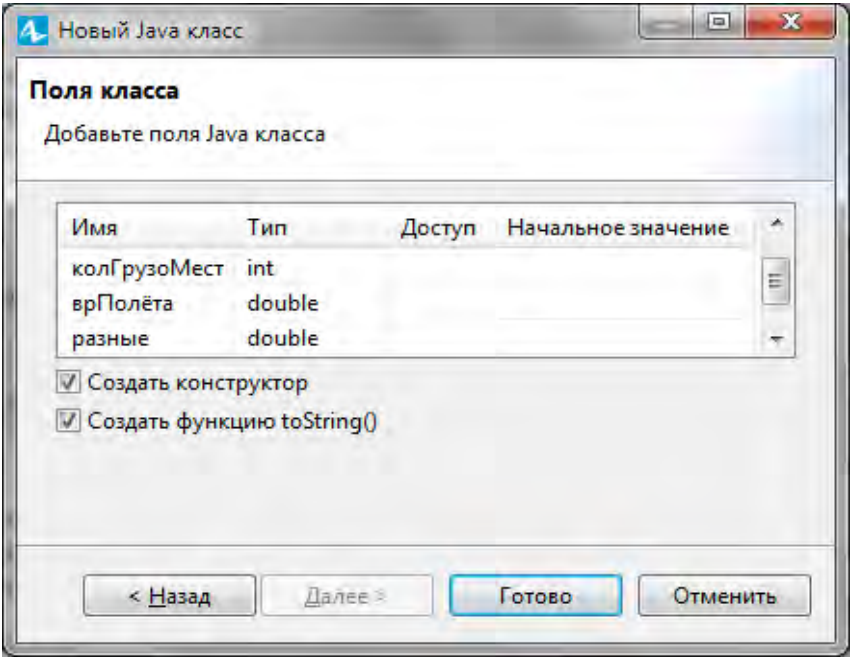


Рис. 9.7. Поля нового Java класса `ТранспСредство`

Продолжите построение сегмента имитации **Прибытие самолётов в аэропорт 1 и ожидания погрузки**.

Поочередно выделите объекты этого сегмента и установите их свойства согласно табл. 9.4.

Таблица 9.4

Свойство	Значения
source	

Имя:
Тип заявки:
Прибывают согласно
Время между прибытиями
Количество заявок, прибывающих за один раз
Ограниченное количество прибытий
Максимальное количество прибытий
Новая заявка
Действия
При выходе:
	`самолёт1`
	`ТранспСредство`
	`Времени между прибытиями`
	`0`
	`колСамТипА`
	`Установить флажок`
	`1`
	`ТранспСредство`
	`entity.типТрансп=1;`
	`entity.колГрузоМест=`
	`грузПодСамА;`

source1	

Имя:
Тип заявки:
Прибывают согласно
Время между прибытиями
Количество заявок, прибывающих за один раз
Ограниченное количество прибытий
Максимальное количество прибытий
Новая заявка
Действия
При выходе:
	`самолёт2`
	`ТранспСредство`
	`Времени между прибытиями`
	`0`
	`колСамТипБ`
	`Установить флажок`
	`1`
	`ТранспСредство`
	`entity.типТрансп=2;`
	`entity.колГрузоМест=`
	`грузПодСамБ;`

queue	

Имя:
Тип заявки:
Вместимость
Включить сбор статистики
	`стоянкаПогр1А`
	`ТранспСредство`
	`колСамТипА`
	Установить флажок

Продолжение табл. 9.4

Свойство	Значения
queue1	

Имя:
Тип заявки:
Вместимость
Включить сбор статистики
	`стоянкаПогр1Б`
	`ТранспСредство`
	`колСамТипБ`
	Установить флажок

hold	
Тип заявки: **Изначально заблокирован**	`ТранспСредство` Установить флажок
hold1	
Тип заявки: **Изначально заблокирован**	`ТранспСредство` Установить флажок
exit	

Имя:
Тип заявки:
Действия
При выходе:
	`наПогрузку1`
	`ТранспСредство`
	`if (entity.типТрансп==1)`
	`{hold.setBlocked(true);`
	`enter2.take(entity);}`
	`else`
	`{hold1.setBlocked(true);`
	`enter3.take(entity);}`

`enter`	
Тип заявки:	`ТранспСредство`
`enter1`	
Тип заявки:	`ТранспСредство`

Остановимся на замысле построения сегмента.

При запуске модели источники `самолёт1` и `самолёт2` генерируют число заявок, равное количеству самолётов типа А и типа Б соответственно. На этом активность источников прекращается.

Эти заявки-самолёты поступают на имитируемые объектами queue стоянки `стоянкаПогр1А` и `стоянкаПогр1Б` соответственно.

Элементы hold и hold1 изначально заблокированы, поэтому заявки-самолёты дальше не проходят.

Элементы hold и hold1 управляются сегментом **Поступление и учёт контейнеров в аэропорту 1**. Как только в аэропорт 1 поступит количество контейнеров, достаточное для полной загрузки самолёта типа А, и этот самолёт имеется на стоянке, а также нет самолётов типа А на погрузке, формируется команда на разблокировывание элемента hold.

Заявка-самолёт поступает на объект наПогрузку1 (объект exit) и далее на сегмент **Погрузка контейнеров в аэропорту 1**.

Если ожидающих погрузку самолётов типа А нет, а количество контейнеров достаточно для полной загрузки самолёта типа Б и он имеется в наличии, то аналогично формируется команда на разблокировывание элемента hold1. Заявка-самолёт также поступает на объект наПогрузку1 (объект exit) и далее на сегмент **Погрузка контейнеров в аэропорту 1**.

Элементы enter и enter1 увеличивают число входов объектов queue и queue1 соответственно. Через них поступают в последующем заявки-самолёты, прибывшие из аэропорта 2, разгруженные и теперь отправленные на стоянки ожидания погрузки в аэропорту 1.

9.1.8.2. Поступление и учёт контейнеров в аэропорту 1

Сегмент **Поступление и учёт контейнеров в аэропорту 1** предназначен для имитации приёма поступающих от источников грузов (контейнеров), учёта их, определения необходимого количества контейнеров для загрузки соответствующего самолёта и формирования команды для отправки этого самолёта на погрузку контейнеров.

Создайте этот сегмент.

1. Из палитры **Презентация** перетащите элемент **Прямоугольник**.
2. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** 30, **Y:** 270, **Ширина:** 220, **Высота:** 200.
3. Перетащите элемент **text** и в поле **Текст:** введите Поступление и учёт контейнеров в аэропорту 1.
4. Перетащите из **Библиотеки моделирования процессов** по одному объекту **source**, **selectOutput5** (имя selectOutput) и **sink**. Поместите и соедините их так, как на рис. 9.8.

Установите свойства объектов согласно табл. 9.5.

Объект source с именем истГрузов1 генерирует заявку-партию класса ГрузАэропорт1 поступивших контейнеров, число которых в партии распределено по равномерному закону. Заявка-партия поступает на объект selectOutput.

В условии 1 проверяется наличие числа контейнеров, достаточного для полной загрузки самолёта типа А. Если число контейнеров недостаточно, заявка-партия уничтожается. Так продолжается до тех пор, пока не выполнится условие 1.

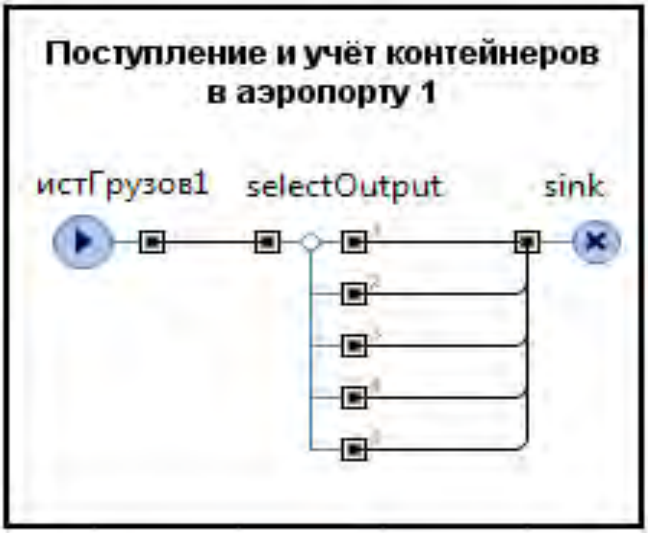


Рис. 9.8. Сегмент **Поступление и учёт контейнеров в аэропорту 1**

Таблица 9.5

Свойство	Значения
source	

Имя:
Тип заявки:
Прибывают согласно
Время между прибытиями
Количество заявок, прибывающих за один раз
Новая заявка
Действия При выходе
	`истГрузов1`
	`ГрузАэропорт1`
	`Времени между прибытиями`
	`exponential(1/срВрПостКонт1)`
	
	`1`
	`ГрузАэропорт1`
	`a=(uniform_`
	`discr(минКонтПост1,`
	`максКонтПост1));`
	`всегоПостК1+=a;`
	`текНалКонт1+=a;`

selectOutput	
Тип заявки: **Использовать:**	`ГрузАэропорт1` `Условия`
Условие 1	`текНалК1<грузПодСамА`
Условие 2	`стоянкаПогр1А.size()>0&&` `погрузка1А.size()==0`
Действия При выходе 2	`текНалК1-=грузПодСамА;` `hold.setBlocked(false);`
Условие 3	`текНалК1<грузПодСамБ`

Окончание табл. 9.5

Свойство	Объекты
Условие 4	`стоянкаПогр1Б.size()>0&&` `погрузка1Б.size()==0`
Действия При выходе 4	`текНалК1-=грузПодСамБ;` `hold1.setBlocked(false);`
sink	
Тип заявки:	`ГрузАэропорт1`

После выполнения условия 1 проверяется условие 2: наличие самолётов типа А на стоянке ожидания погрузки и незанятость пунктов погрузки.

При выполнении условия 2 формируется команда на разблокировывание элемента hold (см. п. 9.1.8.2) и самолёт типа А отправляется на погрузку.

Если условие 2 не выполняется, например, при отсутствии свободного самолёта типа А, проверяется условие 3.

Если условие 3 не выполняется, то есть недостаточно контейнеров для полной загрузки самолёта типа Б, заявка-партия уничтожается.

Если условие 3 выполняется (поступило число контейнеров, достаточное для полной загрузки самолёта типа Б), проверяется условие 4: наличие самолётов типа Б на стоянке ожидания погрузки и незанятость пунктов погрузки.

Если условие 4 не выполняется, заявка-партия уничтожается.

При выполнении условия 4 формируется команда на разблокировывание элемента hold1 (см. п. 9.1.8.2) и самолёт типа Б отправляется на погрузку.

Описанная последовательность проверок производится каждый раз при появлении в модели очередной заявки-партии. Все заявки типа `ГрузАэропорт1` выводятся из модели.

9.1.8.3. Погрузка контейнеров в аэропорту 1

Сегмент **Погрузка контейнеров в аэропорту 1** предназначен для имитации погрузки в самолёты и отправки загруженных самолётов в полёт в аэропорт назначения.

Создайте сегмент **Погрузка контейнеров в аэропорту 1**.

1. Из палитры **Презентация** перетащите элемент **Прямоугольник**.
2. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `350`, **Y:** `50`, **Ширина:** `460`, **Высота:** `200`.

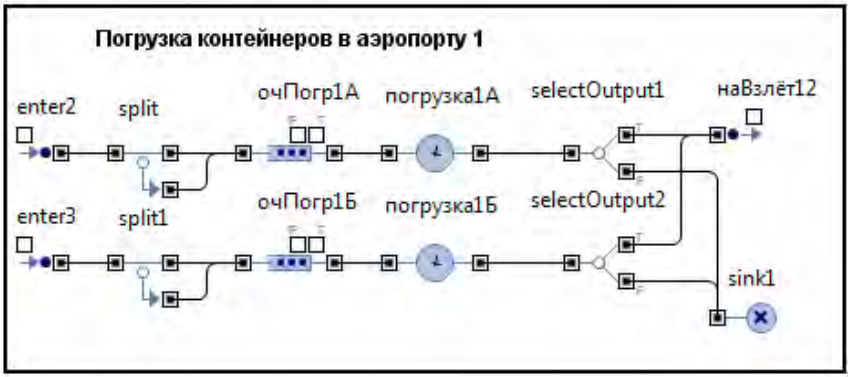


Рис. 9.9. Сегмент **Погрузка контейнеров в аэропорту 1**

3. Перетащите элемент **text** и в поле **Текст:** введите `Погрузка контейнеров в аэропорту 1`.

4. Перетащите из **Библиотеки моделирования процессов** по два объекта **enter**, **split, queue**, **delay, selectOutput** и по одному объекту **exit** и **sink**. Поместите и соедините их так, как на рис. 9.9.

5. Установите свойства объектов согласно табл. 9.6.

Таблица 9.6

Свойство	Значения
enter2	
Тип заявки:	`ТранспСредство`
enter3	
Тип заявки:	`ТранспСредство`
split	

Типы заявок:
Оригинал, Копия
 	
Количество копий
Новая заявка (копия)
Действия
При выходе копии:
	
	`ТранспСредство,`
	`ТранспСредство`
	`entity.колГрузоМест-1`
	`ТранспСредство`
	`entity.типТрансп=`
	`original.типТрансп;`
	`entity.колГрузоМест=`
	`original.колГрузоМест;`
	`entity.tPolet=`
	`original.tPolet;`
	`entity.разные`
	`=original.разные;`

Продолжение табл. 9.6

Свойство	Значения
split1	
Типы заявок: **Оригинал, Копия** **Количество копий** **Новая заявка (копия)** **Действия** **При выходе копии:**	 ТранспСредство, ТранспСредство entity.колГрузоМест-1 ТранспСредство entity.типТрансп= original.типТрансп; entity.колГрузоМест= original.колГрузоМест; entity.врПолёта= original.врПолёта; entity.разные =original.разные;
Имя: **Тип заявки:** **Максимальная вместимость** **Действия При выходе** **Включить сбор статистики**	очПогр1А ТранспСредство Установить флажок entity.разные= срВрПогрКонтСам1А; Установить флажок
queue1	
Имя: **Тип заявки:** **Максимальная вместимость** **Действия При выходе** **Включить сбор статистики**	очПогр1Б ТранспСредство Установить флажок entity.разные= срВрПогрКонтСам1Б; Установить флажок
delay	
Имя: **Тип заявки:** **Задержка задаётся** **Время задержки** **Вместимость** **Действия При подходе к выходу** **Включить сбор статистики**	погрузка1А ТранспСредство Определённое время exponential (1/entity.разные) погрКонтСам1А погрКонтА1++; Установить флажок
delay1	
Имя: **Тип заявки:** **Задержка задаётся**	погрузка1Б ТранспСредство Определённое время

Окончание табл. 9.6

Время задержки
 	
Вместимость
Действия При подходе к выходу
Включить сбор статистики
	`exponential (1/entity.разные)`
	`погрКонтСам1Б`
	`погрКонтБ1++;`
	Установить флажок

selectOutput1	

Тип заявки:
Выход true выбирается
Условие
 	
Действия При выходе (true)
	`ТранспСредство`
	При выполнении условия
	`entity.колГрузоМест ==погрКонтА1`
	`entity.врПолёта= normal(отклВрПолётаА12, срВрПолётаА12);`
	`погрКонтА1=0;`

selectOutput2	

Тип заявки:
Выход true выбирается
Условие
 	
Действия При выходе (true)
	`ТранспСредство`
	При выполнении условия
	`entity.колГрузоМест ==погрКонтБ1`
	`entity.врПолёта= normal(отклВрПолётаБ12, срВрПолётаБ12);`
	`погрКонтБ1=0;`

exit	

Имя:
Тип заявки:
Действия При выходе
	`наВзлёт12`
	`ТранспСредство`
	`if (entity.типТрансп==1) enter4.take(entity);`
	`else enter5.take(entity);`

sink	
Тип заявки:	`ТранспСредство`

Предположим, что из сегмента ожидания погрузки через объект enter2 поступила заявка-самолёт типа А в объект split. Объектом split заявка размножается на число заявок, равное количеству контейнеров, которые должны быть погружены в самолёт. Заявка-оригинал из модели не выводится.

Таким образом, далее каждая заявка интерпретируется как заявка-контейнер. Тем не менее, каждой копии присваиваются значения полей оригинала, так как после погрузки все заявки-контейнеры, кроме последней, будут выведены из модели.

Заявки-контейнеры занимают очередь к объекту погрузка1А, имитирующему непосредственно погрузку контейнеров в самолёт типа А в аэропорту 1. При покидании очереди выполняется код entity.разные=срВрПогрКонтСам1А, записывающий в поле с именем разные заявки-контейнера среднее время погрузки одного контейнера.

После объекта погрузка1А, на выходе которого ведётся счёт погруженных контейнеров (погрКонтА1++), заявки-контейнеры входят в объект selectOutput1.

Этот объект проверяет условие (entity.колГрузоМест ==погрКонтБ1): полная ли загрузка самолёта? При выполнении этого условия, а оно будет выполнено тогда, когда будет загружен последний контейнер, последняя заявка теперь уже в качестве заявки-самолёта поступит в объект наВзлёт (exit).

Из этого объекта заявка-самолёт типа А с полным грузом поступит в сегмент имитации полёта из аэропорта 1 в аэропорт 2.

Аналогичным образом имитируется погрузка в самолёт типа Б. Имитация начинается с поступления заявки-самолёта через объект enter3 в объект split1.

9.1.8.4. Полёт из аэропорта 1 в аэропорт 2

Сегмент **Полёт из аэропорта 1 в аэропорт 2** предназначен для имитации полёта самолётов с грузом из аэропорта 1 в аэропорт 2.

Создайте сегмент.

1. Из палитры **Презентация** перетащите элемент **Прямоугольник**.
2. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** 830, **Y:** 50, **Ширина:** 240, **Высота:** 200.
3. Перетащите элемент **text** и в поле **Текст:** введите Полёт из аэропорта 1 в аэропорт 2.
4. Перетащите из **Библиотеки моделирования процессов** по два объекта **enter**, **delay** и один объект **exit**. Поместите и соедините их так, как на рис. 9.10.
5. Установите свойства объектов согласно табл. 9.7.

Предположим, что поступила заявка-самолёт типа А в объект enter4 и далее в объект с именем полётА12 (delay). Идентификатор полётА12 означает, что объект имитирует непосредственно полёт из аэропорта 1 в аэропорт 2. В аэропорту 2 заявка-самолёт входит в сегмент ожидания разгрузки.

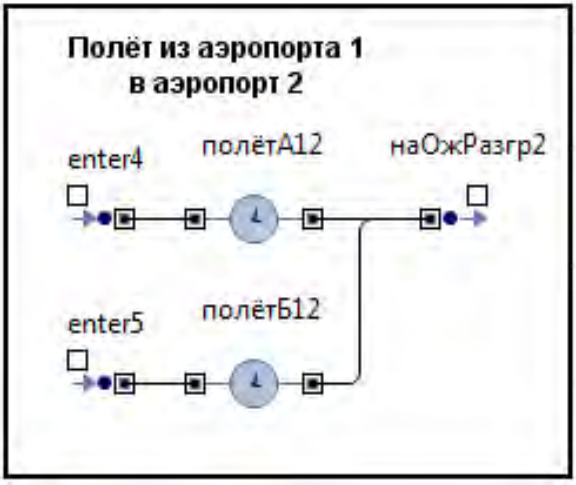


Рис. 9.10. Сегмент **Полёт из аэропорта 1 в аэропорт 2**

Таблица 9.7

Свойство	Значения
enter4	
Тип заявки:	`ТранспСредство`
enter5	
Тип заявки:	`ТранспСредство`
delay	

Имя:
Тип заявки:
Задержка задаётся
Время задержки
Вместимость
Включить сбор статистики
	`полётА12`
	`ТранспСредство`
	`Определённое время`
	`entity.врПолёта`
	`колСамТипА`
	Установить флажок

delay1	

Имя:
Тип заявки:
Задержка задаётся
Время задержки
Вместимость
Включить сбор статистики
	`ПолётБ12`
	`ТранспСредство`
	`Определённое время`
	`entity.врПолёта`
	`колСамТипБ`
	Установить флажок

exit	

Действия
При выходе:
	`if (entity.типТрансп==1)`
	`enter10.take(entity);`
	`else enter11.take(entity);`

9.1.8.5. Ожидание разгрузки в аэропорту 1

Сегмент **Ожидание разгрузки в аэропорту 1** предназначен для имитации ожидания разгрузки самолётов, прибывающих с грузом из аэропорта 2.

Создайте сегмент.

1. Из палитры **Презентация** перетащите элемент **Прямоугольник**.
2. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `270`, **Y:** 270, **Ширина:** `310`, **Высота:** `200`.
3. Перетащите элемент **text** и в поле **Текст:** введите `Ожидание разгрузки в аэропорту 1`.
4. Перетащите из **Библиотеки моделирования процессов** по два объекта **enter**, **queue, hold** и один объект **exit**. Поместите и соедините их так, как на рис. 9.11.
5. Установите свойства объектов согласно табл. 9.8.

Предположим, что поступила заявка-самолёт из аэропорта 2 в объект enter6. Если средства разгрузки свободны, то есть выполняется условие `(разгрузка1А.size()==0)`, разблокировывается объект hold2 и заявка-самолёт входит в объект `наРазгрузку1` и далее в сегмент имитации разгрузки.

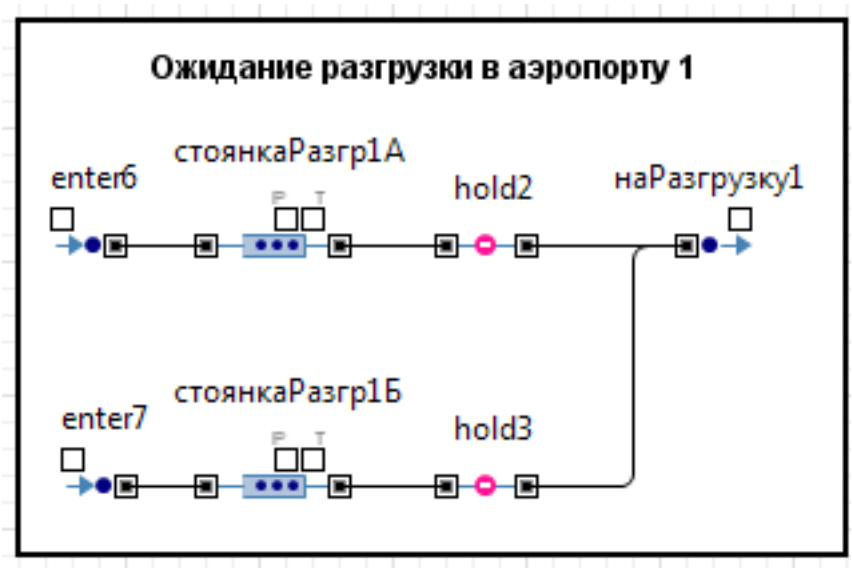


Рис. 9.11. Сегмент **Ожидание разгрузки в аэропорту 1**

Таблица 9.8

Свойство	Значения
enter6	

Тип заявки:
Действие при входе
	`ТранспСредство`
	`if (разгрузка1А.size()==0)`
	`hold2.setBlocked(false);`

enter7	

Тип заявки:
Действие при входе
	`ТранспСредство`
	`if (разгрузка1Б.size()==0)`
	`hold3.setBlocked(false);`

queue	

Имя:
Тип заявки:
Вместимость
Включить сбор статистики
	`стоянкаРазгр1А`
	`ТранспСредство`
	`колСамТипА`
	Установить флажок

queue1	

Имя:
Тип заявки:
Вместимость
Включить сбор статистики
	`стоянкаРазгр1Б`
	`ТранспСредство`
	`колСамТипБ`
	Установить флажок

hold2	
Тип заявки: **Изначально заблокирован**	`ТранспСредство` Установить флажок
hold3	
Тип заявки: **Изначально заблокирован**	`ТранспСредство` Установить флажок
exit	

Имя:
Действия
При выходе:
	`наРазгрузку1`
	`if (entity.типТрансп==1)`
	`{hold2.setBlocked(true);`
	`enter8.take(entity);}`
	`else`
	`{hold3.setBlocked(true);`
	`enter9.take(entity);}`

При выходе из объекта `наРазгрузку1` блокируется объект hold2 кодом `hold2.setBlocked(true)`, так как теперь средства разгрузки самолётов типа А аэропорта 1 заняты.

Если поступает заявка-самолёт типа Б, то она входит в сегмент через объект enter7. Имитация ожидания разгрузки заявкой-самолётом типа Б производится аналогично.

9.1.8.6. Разгрузка самолётов в аэропорту 1

Сегмент **Разгрузка самолётов в аэропорту 1** предназначен для имитации разгрузки самолётов, прибывающих с грузом из аэропорта 2.

Создайте сегмент.

1. Из палитры **Презентация** перетащите элемент **Прямоугольник**.
2. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `600`, **Y:** 270, **Ширина:** `470`, **Высота:** `200`.
3. Перетащите элемент **text** и в поле **Текст:** введите `Разгрузка самолётов в аэропорту 1`.
4. Перетащите по два объекта **enter**, **split**, **queue, delay**, selectOutput и по одному объекту **exit** и **sink**. Поместите и соедините их так, как на рис. 9.12.
5. Установите свойства объектов согласно табл. 9.9.

Предположим, что из сегмента ожидания разгрузки через объект enter8 поступила заявка-самолёт типа А в объект split2. Объектом split2 заявка размножается на число заявок, равное количеству контейнеров, которые должны быть выгружены из самолёта. Заявка-оригинал из модели не выводится.

Таким образом, далее каждая заявка интерпретируется как заявка-контейнер. Тем не менее, каждой копии присваиваются значения полей оригинала, так как после выгрузки все заявки-контейнеры, кроме последней, будут выведены из модели. Однако неизвестно какая из заявок будет последней — оригинал или копия. Поэтому и присваиваются копиям значения полей оригинала.

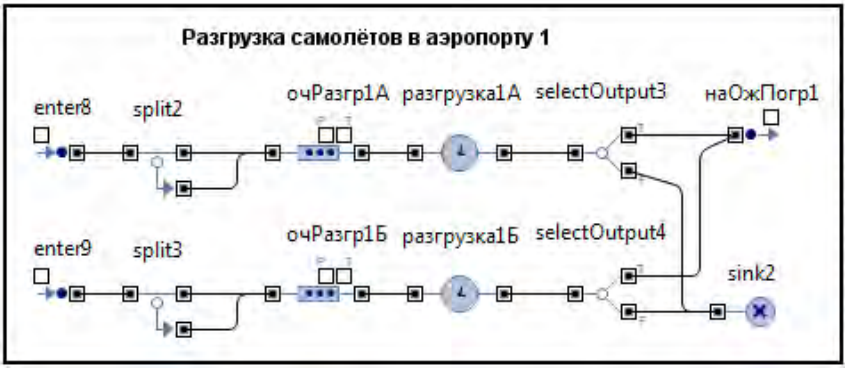


Рис. 9.12. Сегмент **Разгрузка самолётов в аэропорту 1**

Таблица 9.9

Свойство	Значения
enter8	
Тип заявки:	ТранспСредство
enter9	
Тип заявки:	ТранспСредство
split2	
Типы заявок: **Оригинал, Копия** **Количество копий** **Новая заявка (копия)** **Действия** **При выходе копии:**	 ТранспСредство, ТранспСредство entity.колГрузоМест-1 ТранспСредство entity.типТрансп= original.типТрансп; entity.колГрузоМест= original.колГрузоМест; entity.врПолёта= original.врПолёта; entity.разные= original.разные;
split3	
Типы заявок: **Оригинал, Копия** **Количество копий** **Новая заявка (копия)** **Действия** **При выходе копии:**	 ТранспСредство, ТранспСредство entity.колГрузоМест-1 ТранспСредство entity.типТрансп= original.типТрансп; entity.колГрузоМест= original.колГрузоМест; entity.врПолёта= original.врПолёта; entity.разные =original.разные;
queue	
Имя: **Тип заявки:** **Максимальная вместимость** **Действия** **При выходе:** **Включить сбор статистики**	очРазгр1А ТранспСредство Установить флажок entity.разные= срВрВыгрКонтСам1А Установить флажок

Продолжение табл. 9.9

Свойство	Значения
queue1	
Имя: **Тип заявки:** **Максимальная вместимость** **Действия При выходе** **Включить сбор статистики**	очРазгр1Б ТранспСредство Установить флажок entity.разные= срВрВыгрКонтСам1Б Установить флажок
delay	
Имя: **Тип заявки:** **Задержка задаётся** **Время задержки** **Вместимость** **Действия При подходе к выходу** **Включить сбор статистики**	разгрузка1А ТранспСредство Определённое время exponential (1/entity.разные) выгрКонтСам1А выгрКонтА1++; Установить флажок
delay1	
Имя: **Тип заявки:** **Задержка задаётся** **Время задержки** **Вместимость** **Действия При подходе к выходу** **Включить сбор статистики**	разгрузка1Б ТранспСредство Определённое время exponential (1/entity.разные) выгрКонтСам1Б выгрКонтБ1++; Установить флажок
selectOutput3	
Тип заявки: **Выход true выбирается** **Условие** **Действия** **При выходе (true):**	ТранспСредство При выполнении условия entity.колГрузоМест== выгрКонтА1 выгрКонтА1=0; достКА21+= entity.колГрузоМест; hold2.setBlocked(**false**);
selectOutput4	
Тип заявки: **Выход true выбирается** **Условие** **Действия** **При выходе (true):**	ТранспСредство При выполнении условия entity.колГрузоМест== выгрКонтБ1 выгрКонтБ1=0; достКБ21+=

Окончание табл. 9.9

	`entity.колГрузоМест;` `hold3.setBlocked(false);`
exit	

Имя:
Тип заявки:
Действия
При выходе
	`наОжПогр1`
	`ТранспСредство`
	`достК21+=`
	`entity.колГрузоМест;`
	`if (entity.типТрансп==1)`
	`enter.take(entity);`
	`else enter1.take(entity);`

sink2	
Тип заявки:	`ТранспСредство`

Заявки-контейнеры занимают очередь к объекту `разгрузка1А`, имитирующему непосредственно выгрузку контейнеров из самолёта типа А в аэропорту 1. При покидании очереди выполняется код `entity.разные=срВрВыгрКонтСам1А`, записывающий в поле с именем `разные` заявки-контейнера среднее время выгрузки одного контейнера.

После объекта `разгрузка1А`, на выходе которого ведётся счёт выгруженных контейнеров (`выгрКонтА1++)`, заявки-контейнеры входят в объект selectOutput3.

Этот объект проверяет условие (`entity.колГрузоМест ==выгрКонтА1)`: полная ли выгрузка самолёта? При выполнении этого условия, а оно будет выполнено тогда, когда будет выгружен последний контейнер, последняя заявка теперь уже в качестве заявки-самолёта поступит в объект `наОжПогр1` (exit).

При выходе из объекта selectOutput3 переменной `выгрКонтА1` присваивается значение 0, так она должна участвовать в очередной имитации выгрузки. Кроме того, разблокировывается объект hold2, так как теперь средства выгрузки аэропорта 1 самолётов типа А свободны

Из объекта `наОжПогр1` (exit) заявка-самолёт типа А поступит в сегмент имитации ожидания погрузки самолётов типа А аэропорта 1.

Аналогичным образом имитируется выгрузка из самолёта типа Б. Имитация начинается с поступления заявки-самолёта через объект enter9 в объект split3.

9.1.9. Имитация функционирования аэропорта 2

Сегменты, имитирующие функционирование аэропорта 2, построены в основном также, как и сегменты аэропорта 1, поэтому будут отмечены лишь некоторые особенности.

9.1.9.1. Поступление и учёт контейнеров в аэропорту 2

Сегмент **Поступление и учёт контейнеров в аэропорту 2** предназначен для имитации приёма поступающих от источников грузов (контейнеров), учёта их, определения необходимого количества контейнеров для загрузки соответствующего самолёта и формирования команды для отправки этого самолёта на погрузку контейнеров (при наличии самолёта).

1. Из палитры **Презентация** перетащите элемент **Прямоугольник**.
2. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `30`, **Y:** 830, **Ширина:** `290`, **Высота:** `190`.
3. Перетащите элемент **text** и в поле **Текст:** введите `Поступление и учёт контейнеров в аэропорту 2`.
4. Перетащите из **Основной библиотеки** по одному `объекту` **source**, **selectOutput5** (имя selectOutput5) и **sink**. Поместите и соедините их так, как на рис. 9.13.

Установите свойства объектов согласно табл. 9.10.

Рис. 9.13. Сегмент **Поступление и учёт контейнеров в аэропорту 2**

Таблица 9.10

Свойство	Значения
source	

Имя:
Тип заявки:
Прибывают согласно
Время между прибытиями
 	
Количество заявок, прибывающих за один раз
Новая заявка
Действия
При выходе:
	`истГрузов2`
	`ГрузАэропорт2`
	`Времени между прибытиями`
	`exponential`
	`(1/срВрПостКонт2)`
	
	`1`
	`ГрузАэропорт2`
	`d=(uniform_discr(минКонтПост2,`
	`макcКонтПост2));`
	`всегоПостК2+=d;`
	`текНалК2+=d;`

selectOutput5	
Тип заявки: **Использовать:**	`ГрузАэропорт2` `Условия`
Условие 1	`текНалК2<грузПодСамА`
Условие 2	`стоянкаПогр2А.size()>0&&` `погрузка2А.size()==0`
Действия **При выходе 2**	`текНалК2-=грузПодСамА;` `hold6.setBlocked(false);`
Условие 3	`текНалК2<грузПодСамБ`
Условие 4	`стоянкаПогр2Б.size()>0&&` `погрузка2Б.size()==0`
Действия **При выходе 4**	`текНалК2-=грузПодСамБ;` `hold7.setBlocked(false);`
sink	
Тип заявки:	`ГрузАэропорт2`

Объект source с именем `истГрузов2` генерирует заявку-партию класса `ГрузАэропорт2` поступивших контейнеров. Заявка-партия поступает на объект selectOutput5.

Описанная в п. 9.1.8.2 последовательность проверок условий производится и в данном сегменте каждый раз при появлении в модели очередной заявки-партии. Все заявки класса `ГрузАэропорт2` также выводятся из модели.

В данном сегменте, как и в п. 9.1.8.2, при выполнении подобных условий, только относительно аэропорта 2, формируются команды на отправку самолётов на погрузку.

9.1.9.2. Ожидание разгрузки в аэропорту 2

Сегмент **Ожидание разгрузки в аэропорту 2** предназначен для имитации ожидания разгрузки самолётов, прибывших из аэропорта 1.

Создайте сегмент.

1. Из палитры **Презентация** перетащите элемент **Прямоугольник**.
2. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `30`, **Y:** 1040, **Ширина:** `310`, **Высота:** `220`.
3. Перетащите элемент **text** и в поле **Текст:** введите `Ожидание разгрузки в аэропорту 2`.
4. Перетащите из **Библиотеки моделирования процессов** по два объекта **enter**, **queue, hold** и один объект **exit**. Поместите и соедините их так, как на рис. 9.14.
5. Установите свойства объектов согласно табл. 9.11

Предположим, что поступила заявка-самолёт в объект enter11. В нём проверяется условие: средства разгрузки свободны `(разгрузка2Б.size()==0)` и на стоянке нет ожидающих разгрузку самолётов? Если да, то разблокировывается объект hold5 и заявка-самолёт типа Б входит в объект `наРазгрузку2` и далее в сегмент имитации разгрузки аэропорта 2.

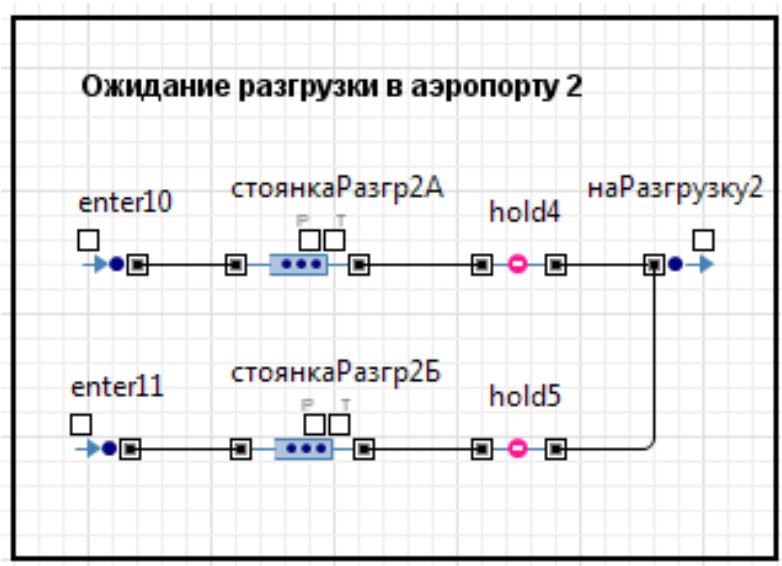


Рис. 9.14. Сегмент **Ожидание разгрузки в аэропорту 2**

Таблица 9.11

Свойство	Объекты
enter10	
Тип заявки: **Действие при входе**	ТранспСредство **if** (стоянкаРазгр2А.size()==0 &&разгрузка2А.size()==0) hold4.setBlocked(**false**);
enter11	
Тип заявки: **Действие при входе**	ТранспСредство **if** (стоянкаРазгр2Б.size()==0 &&разгрузка2Б.size()==0) hold5.setBlocked(**false**);
queue	
Имя: **Тип заявки:** **Вместимость** **Включить сбор статистики**	стоянкаРазгр2А ТранспСредство колСамТипА Установить флажок
queue1	
Имя: **Тип заявки:** **Вместимость** **Включить сбор статистики**	стоянкаРазгр2Б ТранспСредство колСамТипБ Установить флажок
hold4	
Тип заявки: **Изначально заблокирован**	ТранспСредство Установить флажок
hold5	
Тип заявки: **Изначально заблокирован**	ТранспСредство Установить флажок
exit	
Имя: **Действия При выходе**	наРазгрузку2 **if** (entity.типТрансп==1) {hold4.setBlocked(**true**); enter12.take(entity);} **else** {hold5.setBlocked(**true**); enter13.take(entity);}

9.1.9.3. Разгрузка самолётов в аэропорту 2

Сегмент **Разгрузка самолётов в аэропорту 2** предназначен для имитации разгрузки самолётов, прибывающих из аэропорта 1.

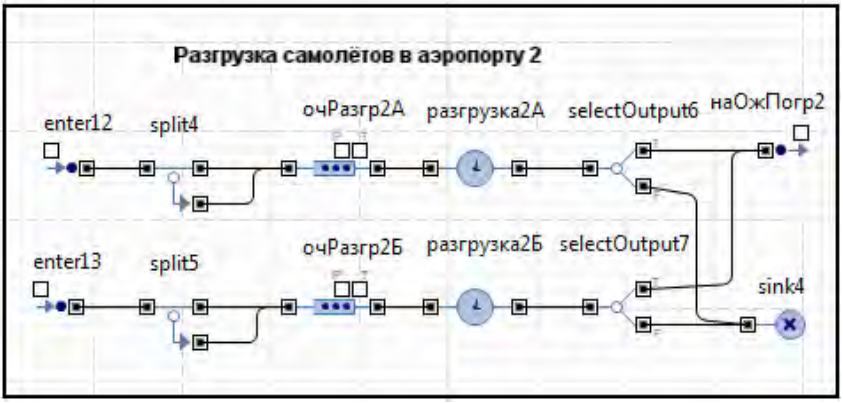


Рис. 9.15. Сегмент **Разгрузка самолётов в аэропорту 2**

Создайте сегмент.

1. Из палитры **Презентация** перетащите элемент **Прямоугольник**.
2. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `360`, **Y:** 1040, **Ширина:** `470`, **Высота:** `220`.
3. Перетащите элемент **text** и в поле **Текст:** введите `Разгрузка самолётов в аэропорту 2`.
4. Перетащите по два объекта **enter**, **split**, **queue, delay**, **selectOutput** и по одному объекту **exit** и **sink**. Поместите и соедините их так, как на рис. 9.15.
5. Установите свойства объектов согласно табл. 9.12.

Предположим, что из сегмента ожидания разгрузки (п. 9.1.9.2) через объект enter13 поступила заявка-самолёт типа Б в объект split5. Объектом split5 заявка размножается на число заявок, равное количеству контейнеров, которые должны быть выгружены из самолёта. Заявка-оригинал из модели не выводится. Поэтому количество копий на 1 меньше, чем количество выгружаемых контейнеров (`entity.колГрузоМест-1`).

Таким образом, также, как и в соответствующем сегменте аэропорта 1, далее каждая заявка интерпретируется как заявка-контейнер. Тем не менее, каждой копии присваиваются значения полей оригинала, так как после выгрузки все заявки-контейнеры, кроме последней, будут выведены из модели. Однако неизвестно какая из заявок будет последней — оригинал или копия. Поэтому также и присваиваются копиям значения полей оригинала.

Таблица 9.12

Свойство	Значения
enter12	
Тип заявки:	ТранспСредство
enter13	
Тип заявки:	ТранспСредство
split4	
Типы заявок: **Оригинал, Копия** **Количество копий** **Новая заявка (копия)** **Действия** **При выходе** **копии**	 ТранспСредство, ТранспСредство entity.колГрузоМест-1 ТранспСредство entity.типТрансп= original.типТрансп; entity.колГрузоМест= original.колГрузоМест; entity.врПолёта= original.врПолёта; entity.разные = original.разные;
split5	
Типы заявок: **Оригинал, Копия** **Количество копий** **Новая заявка (копия)** **Действия** **При выходе копии:**	ТранспСредство, ТранспСредство entity.колГрузоМест-1 ТранспСредство entity.типТрансп= original.типТрансп; entity.колГрузоМест= original.колГрузоМест; entity.врПолёта= original.врПолёта; entity.разные= original.разные;
queue	
Имя: **Тип заявки:** **Максимальная вместимость** **Действия** **При выходе:** **Включить сбор статистики**	очРазгр2А ТранспСредство Установить флажок entity.разные= срВрВыгрКонтСам2А; Установить флажок

Продолжение табл. 9.12

Объекты	Свойства
queue1	
Имя: **Тип заявки:** **Максимальная вместимость** **Действия** **При выходе:** **Включить сбор статистики**	очРазгр2Б ТранспСредство Установить флажок entity.разные= срВрВыгрКонтСам2Б; Установить флажок
delay	
Имя: **Тип заявки:** **Задержка задаётся** **Время задержки** **Вместимость** **Действия При подходе к выходу** **Включить сбор статистики**	разгрузка2А ТранспСредство Определённое время exponential (1/entity.разные) выгрКонтСам2А выгрКонтА2++; Установить флажок
delay1	
Имя: **Тип заявки:** **Задержка задаётся** **Время задержки** **Вместимость** **Действия При подходе к выходу** **Включить сбор статистики**	разгрузка2Б ТранспСредство Определённое время exponential (1/entity.разные) выгрКонтСам2Б выгрКонтБ2++; Установить флажок
selectOutput6	
Тип заявки: **Выход true выбирается** **Условие** **Действия При выходе (true)**	ТранспСредство При выполнении условия entity.колГрузоМест== выгрКонтА2 выгрКонтА2=0; достКА12+= entity.колГрузоМест; hold4.setBlocked(**false**);
selectOutput7	
Тип заявки: **Выход true выбирается** **Условие** **Действия При выходе (true)**	ТранспСредство При выполнении условия entity.колГрузоМест== выгрКонтБ2 выгрКонтБ2=0;

Окончание табл. 9.12

Свойства	Значения

	`достКБ12+=`
	`entity.колГрузоМест;`
	`hold5.setBlocked(false);`

exit	

Имя:
Тип заявки:
Действия
При выходе:
	`наОжПогр2`
	`ТранспСредство`
	`достК12+=`
	`entity.колГрузоМест;`
	`if (entity.типТрансп==1)`
	`enter14.take(entity);`
	`else enter15.take(entity);`

sink4	
Тип заявки:	`ТранспСредство`

9.1.9.4. Ожидание погрузки в аэропорту 2

Сегмент **Ожидание погрузки в аэропорту 2** предназначен для имитации ожидания погрузки самолётов, прибывших из аэропорта 1, после разгрузки в аэропорту 2 .

Создайте сегмент.

1. Из палитры **Презентация** перетащите элемент **Прямоугольник**.
2. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `340`, **Y:** 830, **Ширина:** `290`, **Высота:** `190`.
3. Перетащите элемент **text** и в поле **Текст:** введите `Ожидание погрузки в аэропорту 2`.
4. Перетащите из **Библиотеки моделирования процессов** по два `объекта` **enter**, **queue, hold** и один объект **exit**. Поместите и соедините их так, как на рис. 9.16.
5. Установите свойства объектов согласно табл. 9.13.

Данный сегмент отличается от аналогичного сегмента аэропорта 1 тем, что он не предназначен в том числе и для первичного приёма самолётов. Заявки-самолёты поступают на имитируемые объектами queue стоянки `стоянкаПогр2А` и `стоянкаПогр2Б` соответственно только после разгрузки.

Элементы hold6 и hold7 также изначально заблокированы, поэтому заявки-самолёты дальше стоянок не проходят. Элементы hold6 и hold7 также управляются сегментом **Поступление и учёт контейнеров в аэропорту 2**.

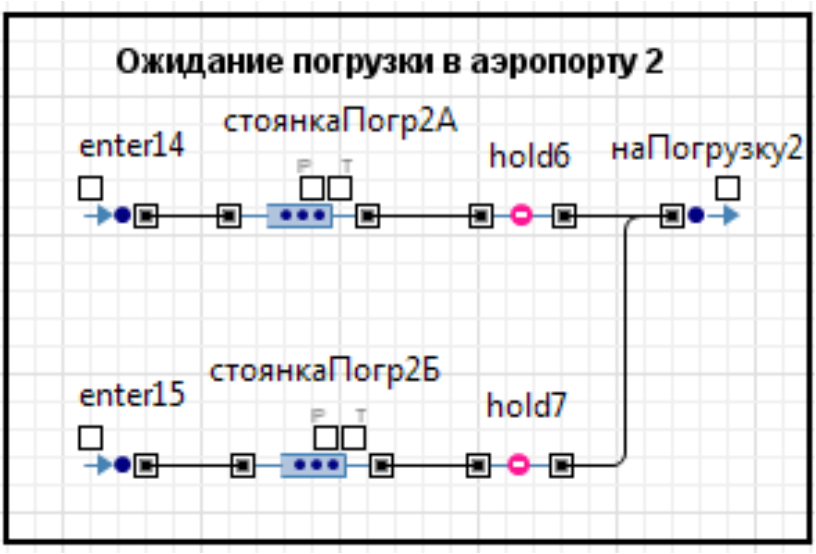


Рис. 9.16. Сегмент **Ожидание погрузки в аэропорту 2**

Таблица 9.13

Свойство	Значения
enter14	
Тип заявки:	`ТранспСредство`
enter15	
Тип заявки:	`ТранспСредство`
queue	

Имя:
Тип заявки:
Вместимость
Включить сбор статистики
	`стоянкаПогр2А`
	`ТранспСредство`
	`колСамТипА`
	Установить флажок

queue1	

Имя:
Тип заявки:
Вместимость
Включить сбор статистики
	`стоянкаПогр2Б`
	`ТранспСредство`
	`колСамТипБ`
	Установить флажок

hold6	
Тип заявки: **Изначально заблокирован**	`ТранспСредство` Установить флажок
hold7	
Тип заявки: **Изначально заблокирован**	`ТранспСредство` Установить флажок

Окончание табл. 9.13

Свойство	Значения
exit	

Имя:
Действия
При выходе:
	наПогрузку2
	`if (entity.типТрансп==1)`
	`{hold6.setBlocked(true);`
	`enter16.take(entity);}`
	`else`
	`{hold7.setBlocked(true);`
	`enter17.take(entity);}`

9.1.9.5. Погрузка контейнеров в аэропорту 2

Сегмент **Погрузка контейнеров в аэропорту 2** предназначен для имитации погрузки в самолёты и отправки загруженных самолётов в полёт в аэропорт назначения.

Создайте сегмент **Погрузка контейнеров в аэропорту 2**.

1. Из палитры **Презентация** перетащите элемент **Прямоугольник**.
2. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `350`, **Y:** `50`, **Ширина:** `460`, **Высота:** `200`.
3. Перетащите элемент **text** и в поле **Текст:** введите `Погрузка контейнеров в аэропорту 1`.
4. Перетащите по два объекта **enter**, **split, queue**, **delay, selectOutput** и по одному объекту **exit** и **sink**. Поместите и соедините их так, как на рис. 9.17.
5. Установите свойства объектов согласно табл. 9.14.

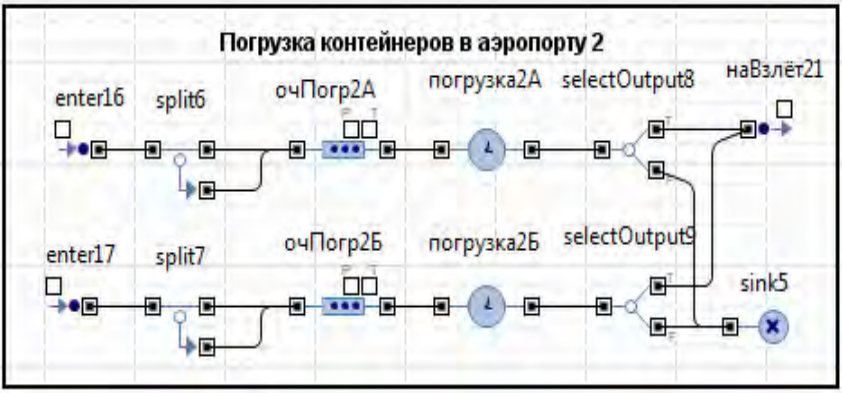


Рис. 9.17. Сегмент **Погрузка контейнеров в аэропорту 2**

Таблица 9.14

Свойство	Значения
enter16	
Тип заявки:	ТранспСредство
enter17	
Тип заявки:	ТранспСредство
split6	
Типы заявок: **Оригинал, Копия** **Количество копий** **Новая заявка (копия)** **Действия** **При выходе копии**	 ТранспСредство, ТранспСредство entity.колГрузоМест-1 ТранспСредство entity.типТрансп= original.типТрансп; entity.колГрузоМест= original.колГрузоМест; entity.tPolet= original.tPolet; entity.разные= original.разные;
split7	
Типы заявок: **Оригинал, Копия** **Количество копий** **Новая заявка (копия)** **Действия** **При выходе копии**	 ТранспСредство, ТранспСредство entity.колГрузоМест-1 ТранспСредство entity.типТрансп= original.типТрансп; entity.колГрузоМест= original.колГрузоМест; entity.врПолёта= original.врПолёта; entity.разные= original.разные;
queue6	
Имя: **Тип заявки:** **Максимальная вместимость** **Действия При выходе** **Включить сбор статистики**	очПогр2А ТранспСредство Установить флажок entity.разные= срВрПогрКонтСам2А; Установить флажок

Продолжение табл.9.14

Свойство	Значения
queue7	
Имя: **Тип заявки:** **Максимальная вместимость** **Действия При выходе** **Включить сбор статистики**	очПогр2Б ТранспСредство Установить флажок entity.разные= срВрПогрКонтСам2Б; Установить флажок
delay	
Имя: **Тип заявки:** **Задержка задаётся** **Время задержки** **Вместимость** **Действия При подходе к выходу** **Включить сбор статистики**	Погрузка2А ТранспСредство Определённое время exponential (1/entity.разные) погрКонтСам2А погрКонтА2++; Установить флажок
delay1	
Имя: **Тип заявки:** **Задержка задаётся** **Время задержки** **Вместимость** **Действия При подходе к выходу** **Включить сбор статистики**	Погрузка2Б ТранспСредство Определённое время exponential (1/entity.разные) погрКонтСам2Б погрКонтБ2++; Установить флажок
selectOutput8	
Тип заявки: **Выход true выбирается** **Условие** **Действия При выходе (true)**	ТранспСредство При выполнении условия entity.колГрузоМест ==погрКонтА2 entity.врПолёта= normal(отклВрПолётаА21, срВрПолётаА21); погрКонтА2=0;
selectOutput9	
Тип заявки: **Выход true выбирается** **Условие**	ТранспСредство При выполнении условия entity.колГрузоМест ==погрКонтБ2

Окончание табл.9.14

Свойство	Значения

Действия При выходе (true)	`entity.врПолёта=`
	`normal(отклВрПолётаБ21,`
	`срВрПолётаБ21);`
	`погрКонтБ2=0;`

exit	

Имя:
Тип заявки:
Действия При выходе
	`наВзлёт21`
	`ТранспСредство`
	`if (entity.типТрансп==1)`
	`enter18.take(entity);`
	`else`
	`enter19.take(entity);`

sink	
Тип заявки:	`ТранспСредство`

9.1.9.6. Полёт из аэропорта 2 в аэропорт 1

Сегмент **Полёт из аэропорта 2 в аэропорт 1** предназначен для имитации полёта самолётов с грузом из аэропорта 2 в аэропорт 1.

1. Из палитры **Презентация** перетащите элемент **Прямоугольник**.
2. На странице **Местоположение и размер** панели **Свойства** введите в поля **X:** `850`, **Y:** 1040, **Ширина:** `250`, **Высота:** `220`.
3. Перетащите элемент **text** и в поле **Текст:** введите `Полёт из аэропорта 2 в аэропорт 1`.

Рис. 9.18. Сегмент **Полёт из аэропорта 2 в аэропорт 1**

4. Перетащите из **Библиотеки моделирования процессов** по два `объекта` **enter**, **delay** и один объект **exit**. Поместите и соедините их так, как на рис. 9.18.

5. Установите свойства объектов согласно табл. 9.15.

Таблица 9.15

Свойство	Объекты
enter18	
Тип заявки:	`ТранспСредство`
enter19	
Тип заявки:	`ТранспСредство`
delay	

Имя:
Тип заявки:
Задержка задаётся
Время задержки
Вместимость
Включить сбор статистики
	`полётА21`
	`ТранспСредство`
	`Определённое время`
	`entity.врПолёта`
	`колСамТипА`
	Установить флажок

delay1	

Имя:
Тип заявки:
Задержка задаётся
Время задержки
Вместимость
Включить сбор статистики
	`ПолётБ21`
	`ТранспСредство`
	`Определённое время`
	`entity.врПолёта`
	`колСамТипБ`
	Установить флажок

exit	

Имя:
Действия При выходе
	`наОжРазгр1`
	`if (entity.типТрансп==1)`
	`enter6.take(entity);`
	`else enter7.take(entity);`

9.1.9.7. Вывод результатов моделирования с использованием способа Событие

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

5. Перетащите элемент **Событие** из палитры **Модель** на область просмотра **Результаты** как на рис. 9.3. Измените его имя на `ОбрабРезультМодел`. Нажмите **Enter**.

6. Установите флажок **Отображать имя**.

7. С помощью выпадающего списка **Тип события:** выберите **По таймауту**.

8. Установите **Режим:** Срабатывает один раз.

9. **Время срабатывания (абсолютное)** 720000.

10. В поле **Действие** введите Java код, который будет выполняться при появлении этого события.

```
коэфДост21=достК21/всегоПостК2;
коэфДост12=достК12/всегоПостК1;
коэфПогр1А=погрузка1А.statsUtilization.mean();
коэфПогр1Б=погрузка1Б.statsUtilization.mean();
коэфПогр2А=погрузка2А.statsUtilization.mean();
коэфПогр2Б=погрузка2Б.statsUtilization.mean();
коэфРазгр1А=разгрузка1А.statsUtilization.mean();
коэфРазгр1Б=разгрузка1Б.statsUtilization.mean();
коэфРазгр2А=разгрузка2А.statsUtilization.mean();
коэфРазгр2Б=разгрузка2Б.statsUtilization.mean();
коэфПолётА12=полётА12.statsUtilization.mean();
коэфПолётБ12=полётБ12.statsUtilization.mean();
коэфИспСам1А=коэфПогр1А+коэфРазгр1А+коэфПолётА12;
коэфИспСам1Б=коэфПогр1Б+коэфРазгр1Б+коэфПолётБ12;
коэфПолётА21=полётА21.statsUtilization.mean();
коэфПолётБ21=полётБ21.statsUtilization.mean();
коэфИспСам2А=коэфПогр2А+коэфРазгр2А+коэфПолётА21;
коэфИспСам2Б=коэфПогр2Б+коэфРазгр2Б+коэфПолётБ21;
коэфИспСамА=(коэфИспСам1А+коэфИспСам2А)/2;
коэфИспСамБ=(коэфИспСам1Б+коэфИспСам2Б)/2;
коэфДост=(достК12+достК21)/(всегоПостК1+всегоПостК2);
коэфИспСам=(коэфИспСамА+коэфИспСамБ)/2;
```

9.1.10. Запуск и отладка модели

Прежде чем запустить модель:

1. В окне **Проекты** выделите **ВоздушныеПеревозки**.

2. На странице **Основные** в поле **Единицы модельного времени:** установите часы.

3. В окне **Проекты** выделите **Simulation: Main**.

4. На странице **Случайность** установите **Фиксированное начальное число (воспроизводимые «прогоны»)**.

5. В поле **Начальное число:** введите 15657.

6. Перейдите на страницу **Модельное время**. В поле **Остановить:** выберите **В заданное время**.

7. В поле **Конечное время:** введите 720000. Время моделирования увеличено в 1000 раз по числу прогонов модели в GPSS World. Таким образом, моделируется функционирование системы воздушных перевозок в течение 720 часов (30-ти суток).

Рис. 9.19. Результаты моделирования

8. Запустите модель. Если появятся ошибки, исправьте их. При правильном построении модели вы получите результаты, показанные на рис. 9.19.

Во время выполнения модели не все результаты моделирования видны, так как они рассчитываются и выводятся после окончания заданного времени моделирования.

Видно, что коэффициент доставки системы воздушных перевозок при принятых её характеристиках и потоках грузов составляет `коэфДост` = 0,31 при среднем коэффициенте использования самолётов обоих типов `коэфИспСам` = 0,638. Коэффициент использования самолёта типа Б (0,481) ниже `коэфИспСамА = 0,795`.

Машинное время выполнения модели 467,2 сек.

ГЛАВА 10. МОДЕЛЬ ОБРАБОТКИ ДОКУМЕНТОВ В ОРГАНИЗАЦИИ

10.1. Постановка задачи

Для приёма и обработки документов в организации назначена группа в составе трёх сотрудников. Ожидаемая интенсивность потока документов — 15 документов в час. Среднее время обработки одного документа одним сотрудником — $t_{\text{обс}} = 12$ мин. Каждый сотрудник может принимать документы из любой организации. Освободившийся сотрудник обрабатывает последний из поступивших документов. Поступающие документы должны обрабатываться с вероятностью не менее 0,95.

Определить, достаточно ли назначенной группы из трёх сотрудников для выполнения поставленной задачи.

10.2. Аналитическое решение задачи

Группа сотрудников работает как СМО, состоящая из трёх каналов, с отказами, без очереди. Поток документов с интенсивностью $\lambda = 15\dfrac{1}{\text{час}}$ можно считать простейшим, так как он суммарный от нескольких организаций. Интенсивность обслуживания $\mu = \dfrac{1}{t_{\text{обс}}} = \dfrac{60}{12} = 5\dfrac{1}{\text{час}}$. Закон распределения неизвестен, но это несущественно, так как показано, что для систем с отказами он может быть произвольным.

Граф состояний СМО — это схема «гибели и размножения». Для неё имеются готовые выражения для предельных вероятностей состояний системы:

$$P_1 = \frac{\rho}{1!}P_0,\ P_2 = \frac{\rho^2}{2!}P_0, \ldots, P_n = \frac{\rho^n}{n!}P_0,\ P_{n+1} = \frac{\rho^{n+1}}{nn!}P_0, \ldots, P_{n+m} = \frac{\rho^{n+m}}{n^m n!}P_0,$$

$$P_0 = \left(1 + \frac{\rho}{1!} + \ldots + \frac{\rho^n}{n!} + \frac{\rho^{n+1}}{nn!} + \frac{\rho^{n+2}}{n^2 n!} + \ldots + \frac{\rho^{n+m}}{n^m n!}\right)^{-1}.$$

Отношение $\rho = \lambda/\mu$ называют *приведенной интенсивностью потока документов (заявок)*. Физический смысл её следующий: величина ρ представляет собой среднее число заявок, приходящих в СМО за среднее время обслуживания одной заявки.

В задаче $\rho = \frac{\lambda}{\mu} = \frac{15}{5} = 3.$

В рассматриваемой СМО отказ наступает при занятости всех трёх каналов, то есть при $P_{\text{отк}} = P_3$. Тогда:

$$P_0 = \left(1 + \frac{3}{1} + \frac{3^2}{2!} + \frac{3^3}{3!}\right)^{-1} = 0,077,\ P_3 = \frac{3^3}{3!} P_0 = 4,5 \cdot 0,077 = 0,346.$$

Так как вероятность отказа в обработке документов составляет более 0,34 (0,346), то необходимо увеличить количество сотрудников группы. Увеличим состав группы в два раза, то есть СМО будет иметь теперь шесть каналов, и рассчитаем $P_{\text{отк}}$:

$$P_0 = \left(1 + \frac{3}{1} + \frac{3^2}{2!} + \frac{3^3}{3!} + \frac{3^4}{4!} + \frac{3^5}{5!} + \frac{3^6}{6!}\right)^{-1} = 0,051,$$

$$P_6 = \frac{3^6}{6!} P_0 = \frac{729}{720} \cdot \frac{1}{19,412} = 1,012 \cdot 0,051 = 0,052.$$

Теперь $P_{\text{обс}} = 1 - P_{\text{отк}} = 1 - 0,052 \approx 0,95.$

Таким образом, только группа из шести сотрудников сможет обрабатывать поступающие документы с вероятностью 0,95.

10.3. Решение задачи в AnyLogic

Создайте модель **Обработка_документов**.

1. Выполните команду **Файл/Создать/Модель** на панели инструментов. Откроется диалоговое окно **Новая модель**.
2. В поле **Имя модели** диалогового окна **Новая модель** введите `Обработка_документов`. Выберите каталог, в котором будут сохранены файлы модели. Щёлкните **Готово**.
3. Объекты и элементы модели **Обработка_документов** показаны на рис. 10.1. Перетащите их на агента Main, разместите, соедините и установите значения свойств согласно табл. 10.1.

Для ввода исходных данных используйте элемент **Параметр,** тип первых двух double, а третьего — int:

`срИнтПост` — средний интервал поступления документов, по умолчанию — `4`;

`срВрОбр` — среднее время обработки документа, по умолчанию — `12`;

`колСотруд` — количество сотрудников, по умолчанию — `3`.

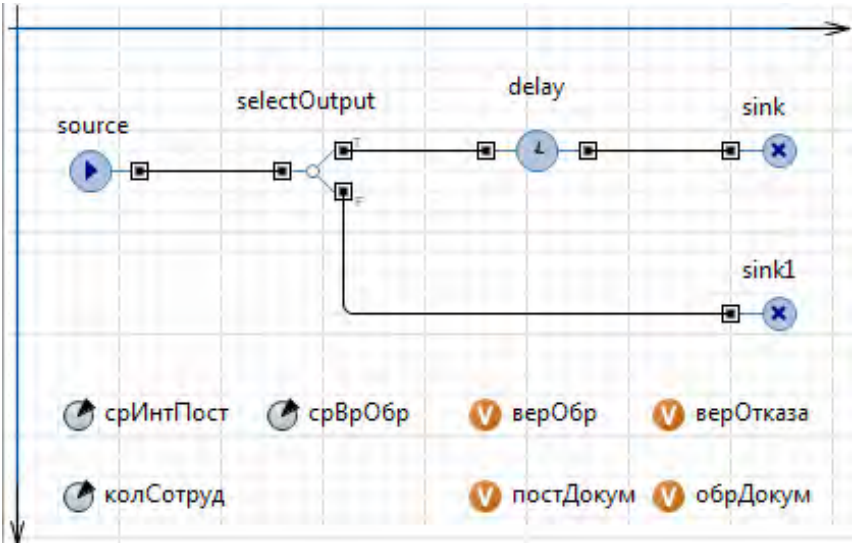


Рис. 10.1. Объекты и элементы модели **ОбрДокументов**

Таблица 10.1

Свойства	Значение

Имя
Тип заявки
Прибывают согласно
Время между прибытиями
Действия
При выходе:
	`source`
	`Entity`
	Времени между прибытиями
	`exponential(1/срИнтПост)`
	
	`постДокум++;`

Имя
Выход true выбирается
Условие
	`selectOutput`
	При выполнении условия
	`delay.size()<колСотруд`

Имя
Тип
Время задержки
Вместимость
Включить сбор статистики
	`delay`
	Определённое время
	`exponential(1/срВрОбр)`
	`колСотруд`
	Установить флажок

Имя
Действие
При входе:
	`sink`
	`обрДокум++;`
	`верОбр=обрДокум/постДокум;`
	`верОтказа=1-верОбр;`

Для вывода результатов моделирования используются элементы **Переменная**, тип которых double:

`постДокум` — количество поступивших документов;
`обрДокум` — количество обработанных документов;
`верОбр` — вероятность обработки документов;
`верОтказа` — вероятность не обработки документов.

AnyLogic-модель построена.

Выделите в окне **Проекты** Simulation:Main.

На странице **Модельное время**, выберите из списка **Остановить:** `В заданное время`. Введите **Конечное время:** 600000 (модельное время увеличено в 10000). **Режим выполнения:** `Виртуальное время (максимальная скорость)`

На странице **Случайность** установите **Фиксированное начальное число (воспроизводимые прогоны)** и **Начальное число:** `1055`.

Запустите модель. Вы должны получить результаты, приведенные на рис. 10.2.

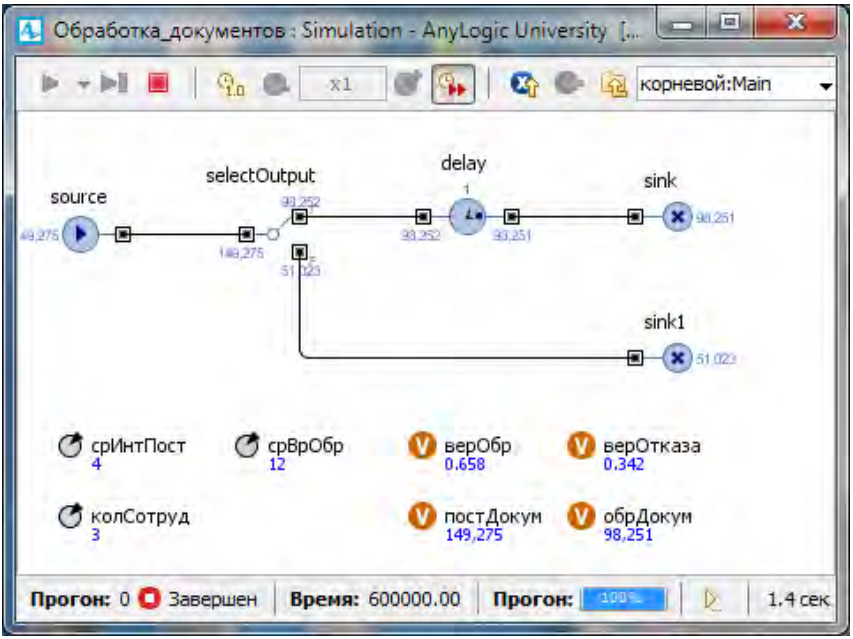


Рис. 10.2. Результаты решения задачи в AnyLogic7

Вероятность не обработки всех документов верОтказа=0,342 (0,345 в AnyLogic6), то есть отличается от полученного аналитическим путём решения на 0,004 (0,001). Хотя это отличие можно отнести на счёт округления до трёх знаков после запятой.

Теперь измените количество сотрудников с трёх на шесть. Для этого выделите элемент **Параметр** с именем колСотруд и установите по умолчанию 6. Всё остальные данные оставьте без изменения. Запустите модель. Вероятность не обработки документов верОтказа=0,052 (0,051), то есть отличается от полученного аналитическим путём решения на 0,002 (0,001).

Сравнительную оценку можно было бы провести и при проведении расчётов с большим числом знаков после запятой, то есть с большей точностью.

10.4. Решение задачи в GPSS World

Программа с комментариями GPSS-модели обработки документов приведена ниже.

```
; Модель обработки документов в организации
T1   EQU  4 ; Среднее время поступления документов
T2   EQU  12 ; Среднее время обработки одного документа
Sotr STORAGE   3 ; Количество сотрудников
VrMod  EQU    60 ; Время моделирования
; Сегмент имитации обработки документов
     GENERATE  (Exponential(1053,0,T1)); Источники
документов
Met1 GATE SNF  Sotr,Met2 ; Не заняты ли сотрудники?
     ENTER  Sotr        ; Нет, тогда занять
     ADVANCE(Exponential(1053,0,T2))  ; обработка
     LEAVE  Sotr        ; Освободить сотрудника
Met3 TERMINATE   ; Учёт обработанных документов
Met2 TERMINATE   ; Учёт необработанных документов
; Сегмент задания времени моделирования и расчёта
результатов
     GENERATE  VrMod
     TEST E TG1,1,Met4  ; Если TG1=1, то расчет
; Вероятностей
     SAVEVALUE VerObr,(N$Met3/N$Met1); обработки
     SAVEVALUE VerOtk,(1-X$VerObr)   ; необработки
Met4 TERMINATE 1
     START  10000  ; задание количества прогонов
```

При трёх сотрудниках в группе обработки документов вероятность необработки (VEROTK) такая же, как и при аналитическом решении задачи, что видно из фрагмента отчёта:

```
STORAGE CAP.REM.MIN.MAX. ENTRIES AVL. AVE.C. UTIL.
 SOTR    3   2   0   3    98161   1   1.963  0.654
SAVEVALUE                RETRY       VALUE
 VEROBR                    0          0.654
 VEROTK                    0          0.346
```

При шести сотрудниках результаты моделирования также совпадают с результатами аналитического решения задачи.

Таким образом, в обеих системах имитационного моделирования получены одинаковые результаты, которые совпадают с достаточно высокой точностью с результатами аналитического решения задачи.

Замечание. Данная задача приводится автором на занятиях обучаемым и как поучительный пример для принятия решений на практике по причине, которая заключается в следующем.

Вполне логичным представляется первичное назначение группы сотрудников для обработки документов. Действительно, за один час в организацию поступают 15 документов. И три сотрудника также могут обработать за один час 15 документов. Каждый сотрудник по 5 документов.

Но элемент случайного их поступления, а также принятые условия, что обрабатывается свободным сотрудником последний поступивший документ, и вероятность обработки должна быть не менее 0,95, вносят свои коррективы, в которых мы убедились, решая эту задачу и аналитически, и в трёх системах моделирования.

ГЛАВА 11. РЕШЕНИЕ ОБРАТНЫХ ЗАДАЧ В ANYLOGIC

Решение обратных задач в GPSS World подробно рассмотрено в [ ]. Приступим к решению этих же задач в AnyLogic.

11.1. Определение среднего времени обработки группы запросов сервером

1. В главе 1 была сохранена модель **Сервер**. Откройте её и сохраните с именем **Сервер Обратная задача**.
2. Удалите из модели объекты презентации и вывода статистики. Она должна иметь объекты, показанные на рис. 11.1.
3. Добавьте два элемента **Параметр** и один элемент **Переменная**. Дайте имена, как на рис. 11.1.
4. Тип элемента **количествоЗапросов** `int`. В поле **Значение по умолчанию:** введите `29`. Это целая часть количества обработанных запросов при решении прямой задачи.
5. Элемент **коэффициент** как и в GPSS-модели предназначен для учёта дробной части количества обработанных запросов (см. п. 1.3). Его тип `double`. **Значение по умолчанию:** `1`.
6. Тип элемента **времяВыполнения** `double`.
7. Выделите объект sink и в поле **Действие при входе:** замените имеющийся там Java код следующим кодом:

```
if( sink.in.count() == количествоЗапросов ){
   времяВыполнения = time();
   stopSimulation();}
```

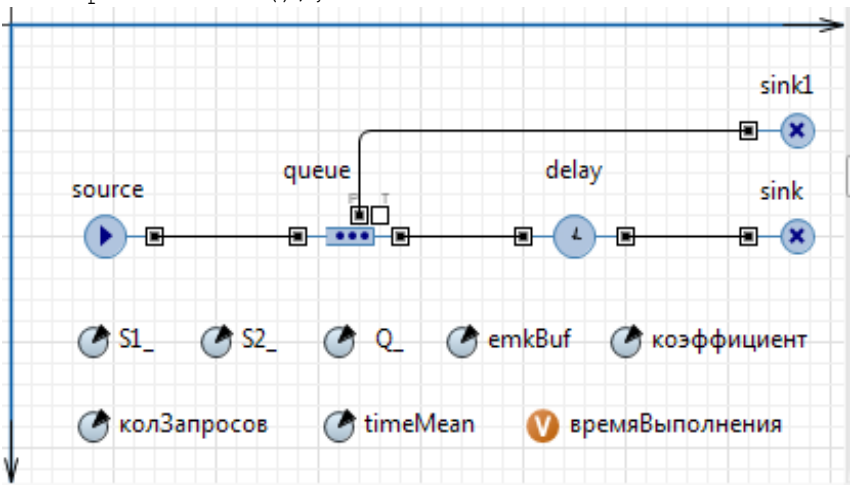


Рис. 11.1. Простой эксперимент **Сервер Обратная задача**

Этот код сравнивает заданное количество запросов в группе (в данном случае 29) с обработанным количеством запросов. При равенстве заданных и обработанных запросов переменной **времяВыполнения** присваивается величина текущего модельного времени, то есть время обработки заданной группы запросов в одном прогоне модели, и простой эксперимент останавливается.

Для выполнения заданного количества прогонов создайте эксперимент варьирования параметров.

1. В панели **Проект** щёлкните правой кнопкой мыши элемент модели и из контекстного меню выберите **Создать эксперимент**.
2. В появившемся диалоговом окне из списка **Тип эксперимента:** выберите **Варьирование параметров**.
3. Оставьте имя эксперимента, рекомендованное системой.
4. Щёлкните кнопку **Готово**. Появится панель **Свойства** (рис. 11.2). Установите свойства согласно рис. 11.2.

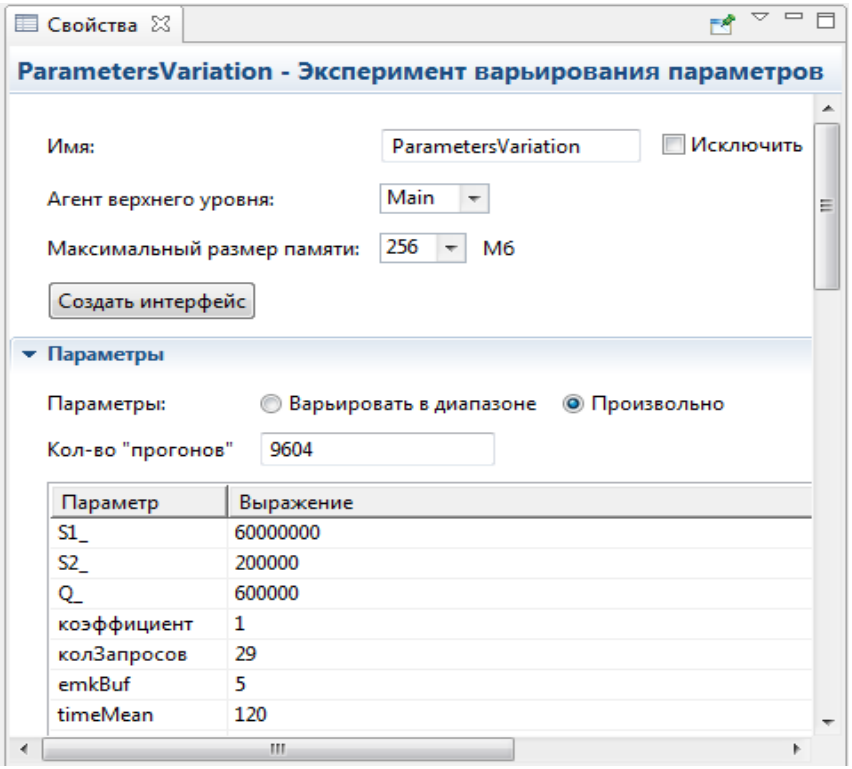


Рис. 11.2. Страница **Эксперимента варьирования параметров**

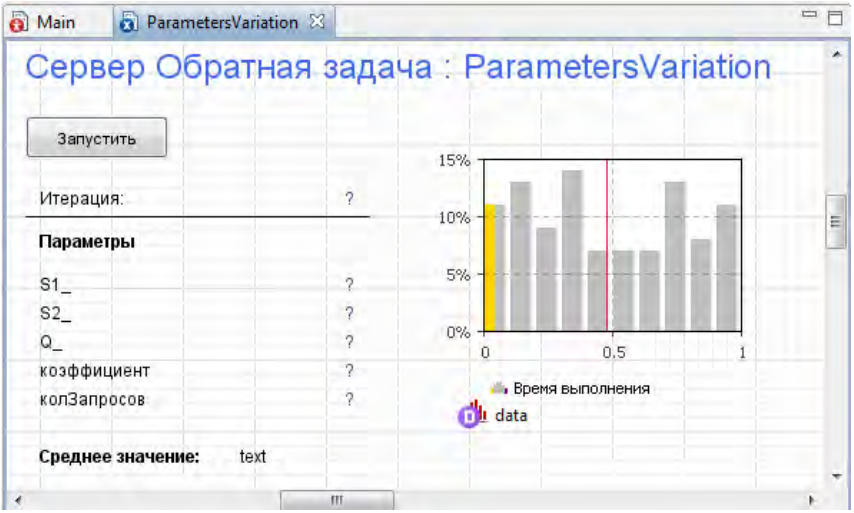


Рис. 11.3. Интерфейс **Эксперимента варьирования параметров**

5. На странице **Действия Java** в поле **Код инициализации эксперимента:** введите `data.reset();`. А в поле **Действие после прогона модели:** `data.add(time()/коэффициент);`

6. В эксперименте **Варьирование параметров,** как вы помните, в отличие от эксперимента **Оптимизация** интерфейс создает пользователь.

7. Создайте интерфейс (рис. 11.3). Начните с нажатия **Создать интерфейс** (см. рис. 11.2).

8. На рис. 11.3 вы видите гистограмму, которая будет отображать время выполнения группы запросов. Перетащите элемент **Гистограмма** из палитры **Статистика** на диаграмму агента Main. На странице **Основные** установите флажок **Отображать среднее**. Оставьте флажок **Отображать плотность вероятности**.

9. Щёлкните **Добавить данные**.

10. Установите **Заголовок:** `Время выполнения`. **Данные:** `data`. **Цвет плотности вер-ти:** `silver`. **Цвет линии среднего:** `crimson`. Оставьте **Не обновлять данные автоматически**.

11. Перетащите элемент **Данные гистограммы**. В поле **Имя:** введите `data`. Остальные свойства оставьте установленными по умолчанию.

12. Добавьте элемент text и на странице **Текст** введите `Среднее значение:`.

13. Добавьте второй элемент text. Оставьте имя, предложенное системой. На странице **Текст:** введите `data.mean()`.

14. Построение **Эксперимента варьирование параметров** завершено. Запустите его. При наличии ошибок, устраните их.

Далее проведём по четыре эксперимента в AnyLogic и GPSS World. Результаты одного из экспериментов при количестве запросов 2916 показаны на рис. 11.4. Результаты всех экспериментов сведены в табл. 11.1.

Для первого эксперимента исходные данные были введены в ходе предыдущих построений. Для экспериментов 2…4 данные вводятся из табл. 11.1. Например, для второго эксперимента значение по умолчанию параметра **колЗапросов** заменяется на `291`, а значение по умолчанию параметра **коэффициент** учёта дробной части — на 10 на диаграмме простого эксперимента.

По результатам экспериментов видно, что по мере учёта дробной части разница между средними временами обработки группы запросов сервером, полученными в GPSS-модели и AnyLogic-модели уменьшается: $\Delta_1 = 192{,}101 \ldots \Delta_4 = 1{,}056$.

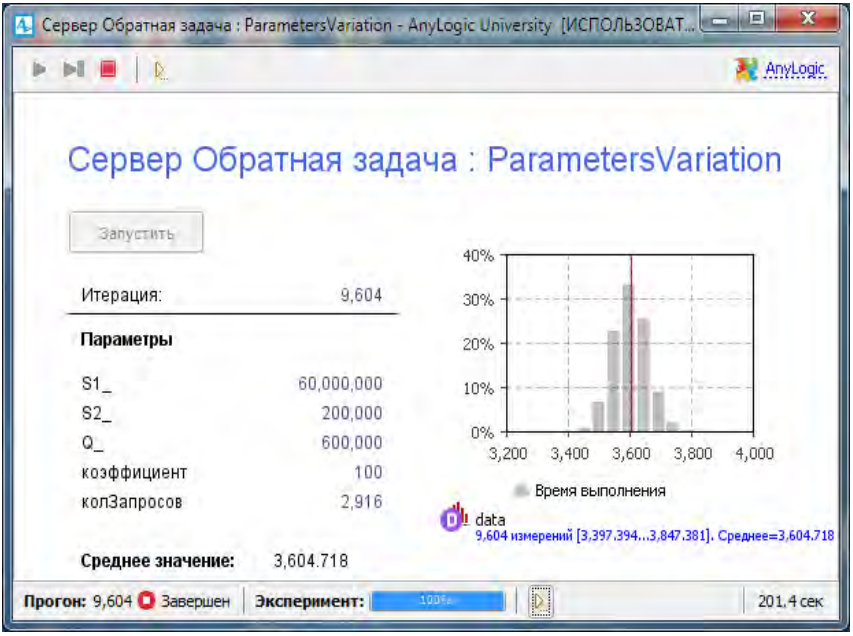


Рис. 11.4. Вариант результатов решения обратной задачи

Таблица 11.1

Параметры и показатели	GPSS World	AnyLogic6	AnyLogic7
Эксперимент 1			
Количество запросов	29	29	29
Коэффициент учёта дробной части	1	1	1
Машинное время выполнения модели, с	2	1,3	6,7
Модельное время выполнения группы запросов	3589,688	3781,755	3789,765
$\Delta_{11} = \lvert 3589,688 - 3781,789 \rvert = 192,101$, $\Delta_{12} = \lvert 3781,789 - 3600 \rvert = 181,789$			
Эксперимент 2			
Количество запросов	291	291	291
Коэффициент учёта дробной части	10	10	10
Машинное время выполнения модели, с	24	7	15
Модельное время выполнения группы запросов	3596,113	3614,877	3616,752
$\Delta_{21} = \lvert 3596,113 - 3614,877 \rvert = 20,639$, $\Delta_{22} = \lvert 3600 - 3614,877 \rvert = 10,877$			
Эксперимент 3			
Количество запросов	2916	2916	2916
Коэффициент учёта дробной части	100	100	100
Машинное время выполнения модели	4 мин 48 с	44 с	201,4 с
Модельное время выполнения группы запросов	3603,039	3604,676	3604,718
$\Delta_{31} = \lvert 3603,089 - 3604,718 \rvert = 1,629$, $\Delta_{32} = \lvert 3600 - 3604,718 \rvert = 4,718$			
Эксперимент 4			
Количество запросов	29161	29161	29161
Коэффициент учёта дробной части	1000	1000	1000
Машинное время выполнения модели	40 мин 48 с	11 мин 17 с	1186,7 с
Модельное время выполнения группы запросов	3604,526	3603,182	3603,470
$\Delta_{41} = \lvert 3604,526 - 3603,470 \rvert = 1,056$, $\Delta_{42} = \lvert 3600 - 3603,470 \rvert = 3,47$			

При этом среднее время в трёх моделях приближается к времени моделирования (3600) в прямой задаче: 181,789 … 3,47 единиц модельного времени. То что оно приближается «справа», объясняется определением одного прогона модели обработкой заданного количества запросов, а не временем моделирования.

Замечание. При проведении экспериментов время выполнения группы запросов, полученное в AnyLogic-модели, может отличаться, так как используются уникальные прогоны (см. рис. 11.2).

11.2. Определение среднего времени изготовления деталей

В главе 2 мы рассмотрели методику решения обратной задачи в GPSS World и прямой задачи в AnyLogic. Теперь решим обратную задачу и в AnyLogic.

1. Откройте модель **Изготовление_в_цехе_деталей**. Сохраните с именем **Изготовление_в_цехе_деталей Обратная задача**.
2. Добавьте на агента Main два элемента **Параметр** и один элемент **Переменная**.
3. Тип **Параметра количествоДеталей** `int`. В поле **Значение по умолчанию:** введите `9017`. Это целая и дробная части количества изготовленных деталей при решении прямой задачи, записанные как целое число.
4. **Параметр коэффициент** как и в GPSS-модели предназначен для учёта дробной части количества обработанных запросов. Его тип `double`. Введите в поле **Значение по умолчанию:** `1000`.
5. Тип элемента **времяИзготовления** `double`.
6. Перейдите на агента **Kontrol**. Выделите объект sink сегмента **Склад готовых изделий** и в поле **Действие при входе:** введите вместо имеющегося там кода следующий Java код:

```
if(склГотДет.in.count() == main.количествоДеталей ){
  main.времяИзготовления=time();
    stopSimulation();}
```

Для выполнения заданного количества прогонов также как и в предыдущем случае создайте эксперимент варьирования параметров.

1. В панели **Проект** щёлкните правой кнопкой мыши элемент модели и из контекстного меню выберите **Создать эксперимент**.
2. В появившемся диалоговом окне из списка **Тип эксперимента:** выберите **Варьирование параметров**.

3. В поле **Имя:** введите:

`Изготовление_в_цехе_деталей_Обратная_задача`.

4. Щёлкните кнопку **Готово**. Появится страница **Основные** панели **Свойства** (см. рис. 11.2). Установите свойства согласно рис. 11.2, кроме количества прогонов. В соответствующее поле введите `16641`.

5. Перейдите на страницу **Действия Java**.

6. В поле **Код инициализации эксперимента:** введите `data.reset();`.

7. В поле **Действие после прогона модели:**

```
data.add( time()/коэффициент );
```

7. Перейдите на страницу **Модельное время**. В поле **Остановить** выберите из списка `Нет`.

8. Создайте интерфейс (см. рис. 11.3). Начните с нажатия **Создать интерфейс** (см. рис. 11.2).

9. Перетащите элемент **Гистограмма** из палитры **Статистика** на агента Main эксперимента. На странице **Основные** установите флажок **Отображать среднее**. Оставьте флажок **Отображать плотность вероятности**.

10. Щёлкните **Добавить данные**.

11. Установите **Заголовок:** `Время выполнения`. **Данные:** `data`. **Цвет плотности вер-ти:** `silver`. **Цвет линии среднего:** `red`. Оставьте **Не обновлять данные автоматически**.

12. Перетащите элемент **Данные гистограммы**. В поле **Имя:** введите `data`. Остальные свойства оставьте установленными по умолчанию.

13. Добавьте элемент text и на странице **Текст** введите `Среднее значение:`.

14. Добавьте второй элемент text. Оставьте имя, предложенное системой. На странице **Текст:** введите `data.mean()`.

15. Построение **Эксперимента варьирование параметров** завершено. Запустите его. При наличии ошибок, устраните их.

При построении модели для решения обратной задачи мы ввели данные для проведения эксперимента с учётом трёх знаков после запятой.

Проведём эксперимент в AnyLogic. Результат решения обратной задачи показан на рис. 11.5.

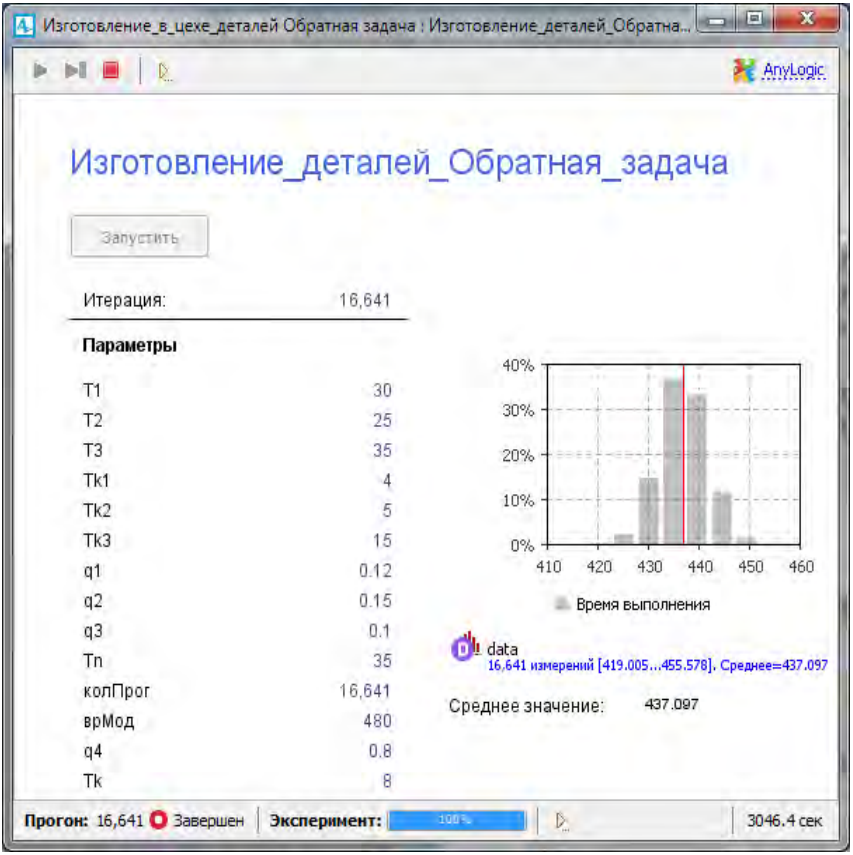


Рис. 11.5. Среднее время изготовления заданного количества деталей

В табл. 11.2 сведены результаты экспериментов решения обратной задачи в GPSS World и AnyLogic. Они свидетельствуют об их адекватности, так как $\Delta_{11} = 0,66$. Если, конечно, исследователя устраивает величина такой разницы, как Δ_{11}.

Разница в модельном времени изготовления партии деталей составляет $\Delta_{12} = 42,903$. Это свидетельствует о том, что для большего приближения к 480 единицам модельного времени нужно увеличивать учёт дробной части количества изготовленных деталей, то есть учитывать четвёртый знак после запятой и т.д.

Машинное время выполнения модели в GPSS World примерно в три раза больше, чем в AnyLogic7, и равно 2 час 23 мин 48 сек.

Таблица 11.2

Параметры и показатели	GPSS World	AnyLogic6	AnyLogic7
Количество деталей	9017	9017	9017
Коэффициент учёта дробной части	1000	1000	1000
Машинное время выполнения модели	2 ч 23 мин 48 с	30 мин 54 с	3046,4 с
Модельное время изготовления партии деталей	436,50	437,16	437,097
$\Delta_{11} = \lvert 436,500 - 437,097 \rvert = 0,597,\ \Delta_{12} = \lvert 480 - 437,097 \rvert = 42,903$			

Обратим внимание ещё раз на то, что время изготовления при повторных запусках эксперимента может отличаться, так как выбраны были уникальные прогоны модели.

Замечание. Таким же способом, то есть заданием количества прогонов в AnyLogic как и в GPSS World можно решать и прямые задачи. Следует иметь в виду, что при таком проведении эксперимента доступна только та информация, вывод которой будет вами организован на интерфейс эксперимента.

ГЛАВА 12. ЗАДАНИЯ НА ПРОЕКТИРОВАНИЕ

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 моделирования необходимо получить с точностью $\varepsilon = 0{,}1$ и доверительной вероятностью $\alpha = 0{,}95$.

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 направлению 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. Сообщения 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 Тс случайное.

После сборки изделие поступает на один из n3 стендов выходного контроля. На одном стенде одновременно проверяется только одно изделие. Время проверки 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 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 проверки 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 приемки. Время приемки Тпр одного изделия случайное.

По результатам приемки бракуется q4 % изделий, которые направляются вновь на стенд выходного контроля.

Принятые 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 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 Tc случайное.

После сборки изделие поступает на один из n3 стендов выходного контроля. На одном стенде одновременно проверяется одно изделие. Время проверки Tп 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 Tз 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 приемки. Время приемки Тпр одного изделия случайное. По результатам приемки бракуется q4 % изделий, которые направляются вновь на стенд выходного контроля и снова проверяются.

Принятые 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 сложностями 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 ИМ функционирования ВККС. Исследовать влияние емкостей входных накопителей, интервалов времени, вероятностей, Q и Vп на вероятности передачи сообщений 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 $\varepsilon = 0{,}01$ и доверительной вероятностью $\alpha = 0{,}99$.

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 Pc 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 ε = 0,01 и доверительной вероятностью α = 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 Т21, Т22, …, Т2n2 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 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 $\varepsilon = 0{,}01$ и доверительной вероятностью $\alpha = 0{,}99$.

Сделать выводы о загруженности средств связи, мастеров по промежуткам ΔT 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 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.

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 эксперимента и количество телефонных аппаратов. Результаты моделирования необходимо получить с точностью $\varepsilon = 0{,}01$ и доверительной вероятностью $\alpha = 0{,}99$.

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 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 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 средними значениями 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 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 19

Постановка задачи

Изготовление в цехе детали начинается через случайное время Тн. Выполнению основных 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 основных 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 Т1, Т2, …, Тn 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: Т11, Т12, Т13 – для СС первого типа, Т21, Т22, Т23 – для СС второго типа, …, Тn1, Тn2, …, Тnn – 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: 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 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;

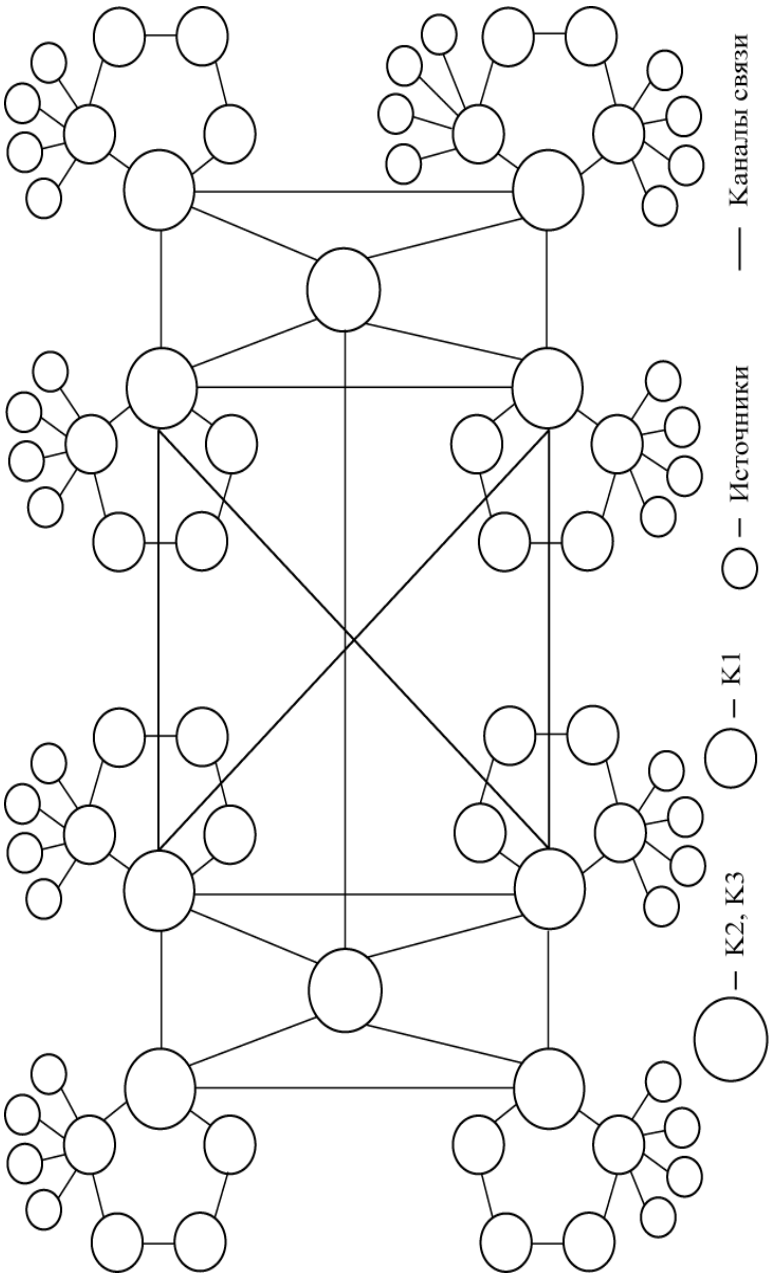


Рис. 12.1. Вариант структуры сети передачи данных

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:

$P_{\text{пр}} = \dfrac{S_{\text{пол}}}{S_{\text{отп}}}$ — вероятность пропускной способности сети, $S_{\text{пол}}$ — количество полученных адресатами сообщений, $S_{\text{отп}}$ — количество отправленных сообщений (поступивших на входы сети);

$t_{\text{ср}} = \dfrac{T_{\text{пер}}}{S_{\text{пол}}}$ — 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. 12.2.

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. 12.2. Алгоритм имитационной модели (начало, лист 1)

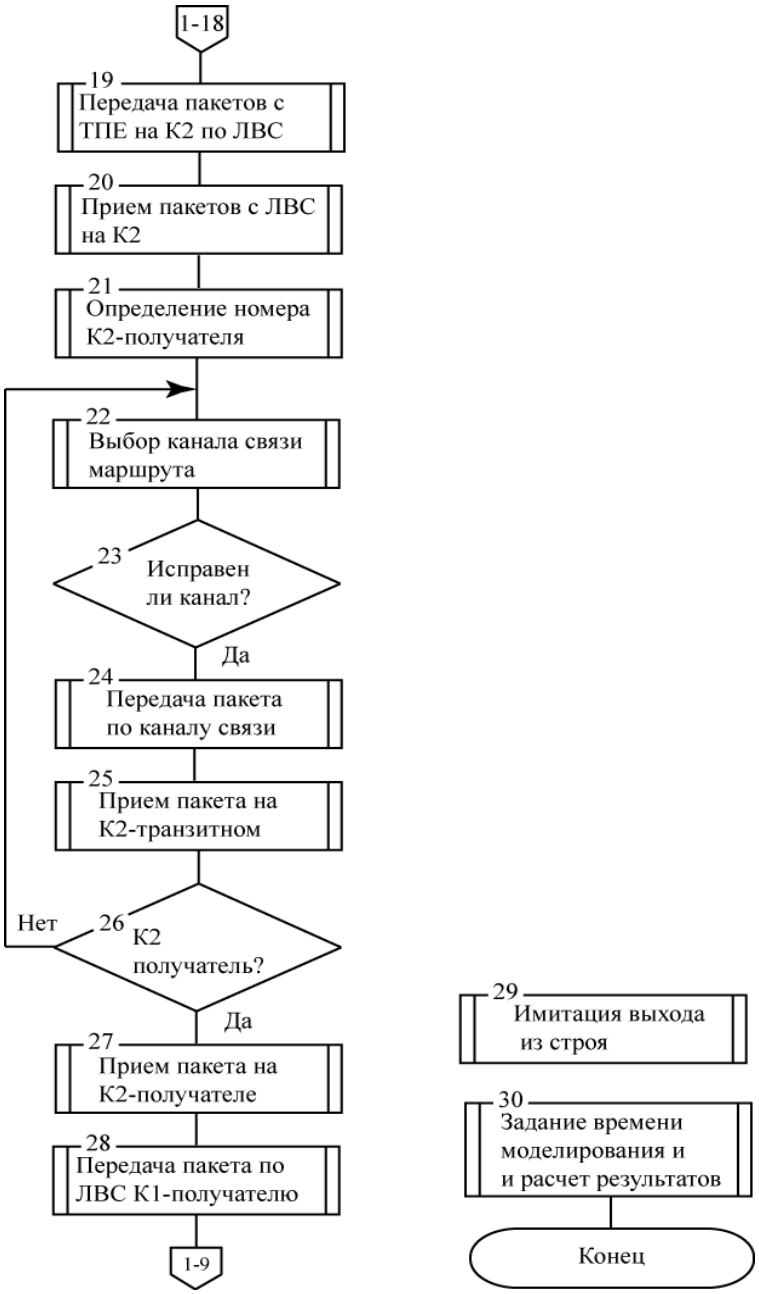


Рис. 12.2. Алгоритм имитационной модели (окончание, лист 2)

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. 12.1) согласно алгоритму, представленному на рис. 12.2. 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

Целью настоящего учебного пособия являлось приобретение навыков разработки постановок задач на концептуальное проектирование, освоение техники ИМ, планирования, проведения и обработки данных компьютерных экспериментов для принятия проектных решений. ИМ выбрано потому как мировая практика научных исследований свидетельствует о том, что методы ИМ занимают около 70 % в общем объёме исследовательского инструментария. В настоящее время в ИМ выделяют три подхода: системной динамики, дискретно-событийный и агентный. Из этих подходов в рамках указанных дисциплин изучается дискретно-событийный подход, обеспечивающий универсальность и эффективность ИМ. Он ориентирован на исследование широкого класса сложных систем, представимых в виде систем массового обслуживания, в том числе и систем военного назначения.

При изучении в рамках различных дисциплин ИМ, а также в практике создания моделей неизбежно возникает вопрос о выборе среды разработки, адекватности систем моделирования: будут ли реализованы так же все функции моделируемой системы? Будут ли получены одинаковые результаты моделирования?

Вспомним Р. Шеннона [24]: «Подобно всем мощным средствам, существенно зависящим от искусства их применения, имитационное моделирование способно дать либо очень хорошие, либо очень плохие результаты».

В пособии на детально разработанных и реализованных средствами GPSS World и AnyLogic моделях объектов с разнородными протекающими в них процессами помимо основной цели также демонстрируется достаточная адекватность GPSS World и AnyLogic относительно результатов моделирования.

В задачу автора не входило дать кардинальную оценку, какая система ИМ и в каких случаях предпочтительна. Читатель вправе сделать это самостоятельно, исходя из целей создания своей имитационной модели проектируемой системы и опираясь на приведенные в пособии результаты.

СПИСОК ЛИТЕРАТУРЫ

1. Боев В. Д. Об адекватности систем имитационного моделирования GPSS World и AnyLogic. Часть 1 // Прикладная информатика. № 6 (30). 2010. С. 69-82.

2. Боев В. Д. Об адекватности систем имитационного моделирования GPSS World и AnyLogic. Часть 2 // Прикладная информатика. № 4 (34). 2011. С. 50-62.

3. Боев В. Д. Некоторые аспекты адекватности систем имитационного моделирования дискретно-событийных процессов: Статья — В сб. докладов Пятой Всероссийской конференции «Имитационное моделирование. Теория и практика» ИММОД-2011 — СПб.: ЦТСиР, 2011.

4. Боев В. Д. Исследование адекватности GPSS World и AnyLogic при моделировании дискретно-событийных процессов: Монография — www.xjtek.ru, 2011.

5. Боев В. Д., Рыжиков Д. М. Имитационная модель процессов изготовления электромеханических модулей: Статья — В сб. докладов Пятой Всероссийской конференции «Имитационное моделирование. Теория и практика» ИММОД-2011 — СПб.: ЦТСиР, 2011.

6. Боев В. Д., Сыпченко Р. П. Компьютерное моделирование. Элементы теории и практики: Учеб. пособие. — СПб.: ВАС, 2009.

7. Боев В. Д., Сыпченко Р. П. Компьютерное моделирование: Курс лекций. — ИНТУИТ, 2010.

8. Боев В. Д., Кирик Д. И., Сыпченко Р. П. Компьютерное моделирование: Пособие по курсовому и дипломному проектированию. — СПб.: ВАС, 2011.

9. Боев В. Д. Моделирование систем. Инструментальные средства GPSS World: Учеб. пособие. — СПб.: БХВ-Петербург, 2004.

10. 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.

11. 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.

12. 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.

13. Боев В. Д. Концептуальное проектирование систем в AnyLogic и GPSS World. — ИНТУИТ.ru, 2013, 436 с. ISBN 978-5-9556-0146-5.

14. Боев В. Д., Моисеев Р. А. Имитационная модель самоорганизующейся сети связи. — Инфокоммуникационные технологии в инновациях, медико-биологических и технических науках: сборник научных трудов Пятого международного научного конгресса «Нейробиотелеком-2012». — СПб.: Политехника, 2012. С. 50-55.

15. Боев В. Д., Мякотин А. В., Тарасов О. М. Оптимизация топологии сети связи объединения (соединения) по минимальности суммарного расстояния между полевым подвижным узлом связи и корреспондирующими узлами // Информационный сборник № 6 Академии военных наук. — СПб., 2011.

16. Боев В. Д., Кирик Д. И., Сыпченко Р. П. Компьютерное моделирование: Пособие по курсовому и дипломному проектированию. — www.xjtek.ru, 2011.

17. Боев В. Д. Модель бизнес-процесса и особенности ее реализации в системе моделирования: Статья. — В сб. докладов конференции «Имитационное моделирование. Теория и практика» ИММОД-2005 — СПб.: ФГУП ЦНИИТС, 2005.

18. Боев В. Д. Решение в системе моделирования прямой и обратной задач: Статья. — В сб. докладов конференции «Имитационное моделирование. Теория и практика» ИММОД-2005 — СПб.: ФГУП ЦНИИТС, 2005.

19. Боев В. Д., Моисеев Р.А. Некоторые классы типовых объектов сетей связи в AnyLogic. Материалы Всероссийской конференции ИММОД-2013 — Казань, 2013.

20. Боев В. Д. Концептуальное проектирование систем в AnyLogic 7 и GPSS World. — ИНТУИТ.ru, 2013, 513 с.

21. Девятков В. В. Методология и технология имитационных исследований сложных систем: современное состояние и перспективы развития: Монография/ В.В. Девятков - М.: Вузовский учебник: ИНФРА-М, 2013. - 448 с. ISBN 978-5-9558-0338-8

22. Девятков В. В. Мир имитационного моделирования: взгляд из России // Прикладная информатика. № 4 (34). 2011. С. 9-29.

23. Лоу А., Кельтон Д. Имитационное моделирование. — СПб.: Питер, БХВ-Петербург, 2004.

24. Скаткова Н. А., Воронин Д. Ю., Ткаченко К. С. Дискриминационный анализ систем имитационного моделирования с использованием версионно-модельной избыточности: Статья. — Радиоэлектронные компьютерные системы, 2010, № 7 (48).

25. Шеннон Р. Имитационное моделирование — искусство и наука. — М.: Мир, 1978.

26. Шрайбер Т. Моделирование на GPSS. — М.:Машиностроение, 1980.

27. The AnyLogic Company anylogic.ru.

ПРИЛОЖЕНИЕ 1

Объекты Библиотеки моделирования процессов

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, позволяя пользователю самому решить, что следует сделать 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.
	SelectOutput5	Объект направляет входящие заявки в один из пяти выходных портов в зависимости от выполнения заданных (детерминистических или заданных с помощью вероятностей) условий.
	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.
	MoveTo	Перемещает заявку в новое место сети.
	RestrictedAreaStart	Обозначает вход в область процесса, в которой одновременно может находиться ограниченное количество заявок.
	RestrictedAreaEnd	Обозначает выход из области процесса, в которой может находиться только ограниченное количество заявок.

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	Добавляет заявки к содержимому поступающей *заявки-контейнера*.
	Dropoff	Удаляет избранные заявки из поступающей заявки-контейнера и пересылает их далее.
	Assembler	Осуществляет сборку одной новой заявки из определенного числа заявок, пришедших из различных источников (до 5).

Обработка

	Delay	Задерживает заявки на заданный период времени.

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.
	ResourceTask	Позволяет конфигурировать собственную задачу для ресурсов, которую нельзя задать стандартными параметрами аварий, обслуживания, перерывов.
	ResourceTaskStart	Задает начало отдельной диаграммы процесса, моделирующей процесс выполнения задачи ресурсами (обычно это процесс подготовки ресурсов).
	ResourceTaskEnd	Задает конец отдельной диаграммы процесса, моделирующей процесс выполнения задачи для ресурсов (обычно это процесс завершения задачи).
	ResourceSendTo	Посылает (перемещает) указанные движущиеся/переносные сетевые ресурсы из их текущего местоположения в заданный узел.

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.

Измерение времени

	TimeMeasureStart	**TimeMeasureStart** вместе с **TimeMeasureEnd** составляет пару объектов, позволяющую измерять время, проведенное заявками между двумя точками диаграммы процесса. Обычно с их помощью измеряется время нахождения заявки в системе или длительность пребывания заявки в каком-то подпроцессе. **TimeMeasureStart** задает начальную точку, он запоминает момент времени, в который заявка проходит через этот объект.
	TimeMeasureEnd	**TimeMeasureEnd** вычисляет для каждой поступившей в него заявки разность между текущим моментом времени и моментом, запомненным объектом **TimeMeasureStart** , на который ссылается этот объект.

Моделирование зон хранения и складов

	RackSystem	Моделирует зону хранения, состоящую из набора стеллажей и проходов между ними (моделируемыми с помощью объектов **PalletRack**), предоставляющий централизованный доступ и управление этими стеллажами.
	RackPick	Извлекает заявку из ячейки стеллажа (**PalletRack**) или зоны хранения (**RackSystem**) и перемещает ее в заданный узел сети.
	RackStore	Помещает заявку в ячейку заданного стеллажа (**PalletRack**) или зоны хранения (**RackSystem**).

Дополнительные

	PML Settings	Задает дополнительные настройки, относящиеся к блокам Библиотеки Моделирования Процессов.
	Wait	Этот блок похож на блок **Queue** с одним исключением: он поддерживает изъятие в ручном режиме (нужно вызвать методы free(), или freeAll()). Этот блок нет определенного порядка (кроме случаев, когда включено вытеснение).
	SelectOutputIn	Вместе с блоком **SelectOutputOut** действуют как две половинки большого блока **SelectOutput** с множеством выходов.
	SelectOutputOut	Вместе с блоком **SelectOutputIn** действуют как две половинки большого блока **SelectOutput** с множеством выходов.
	PlainTransfer	Блок, в который Вы можете вписать код для действий заявки, когда она проходит через какое-то место диаграммы процесса.

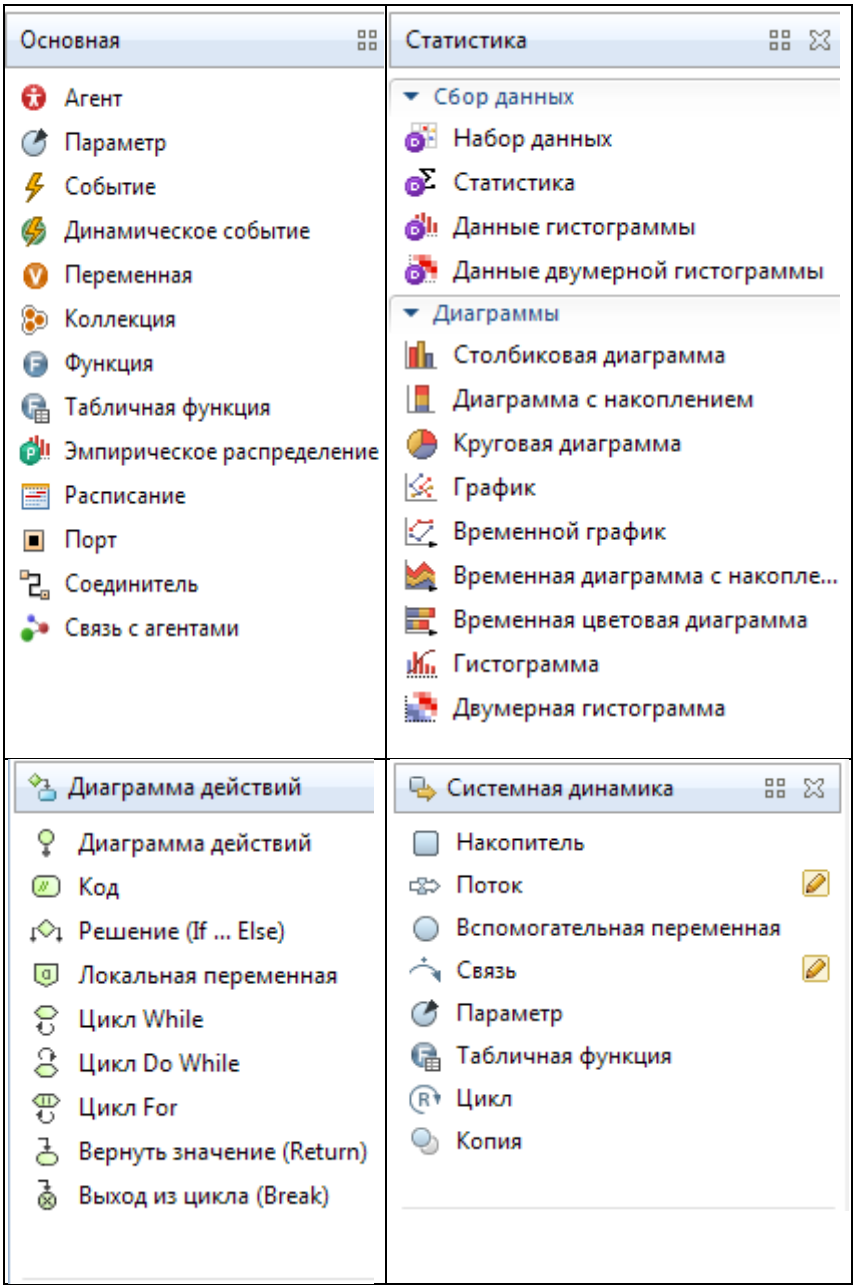
ПРИЛОЖЕНИЕ 2

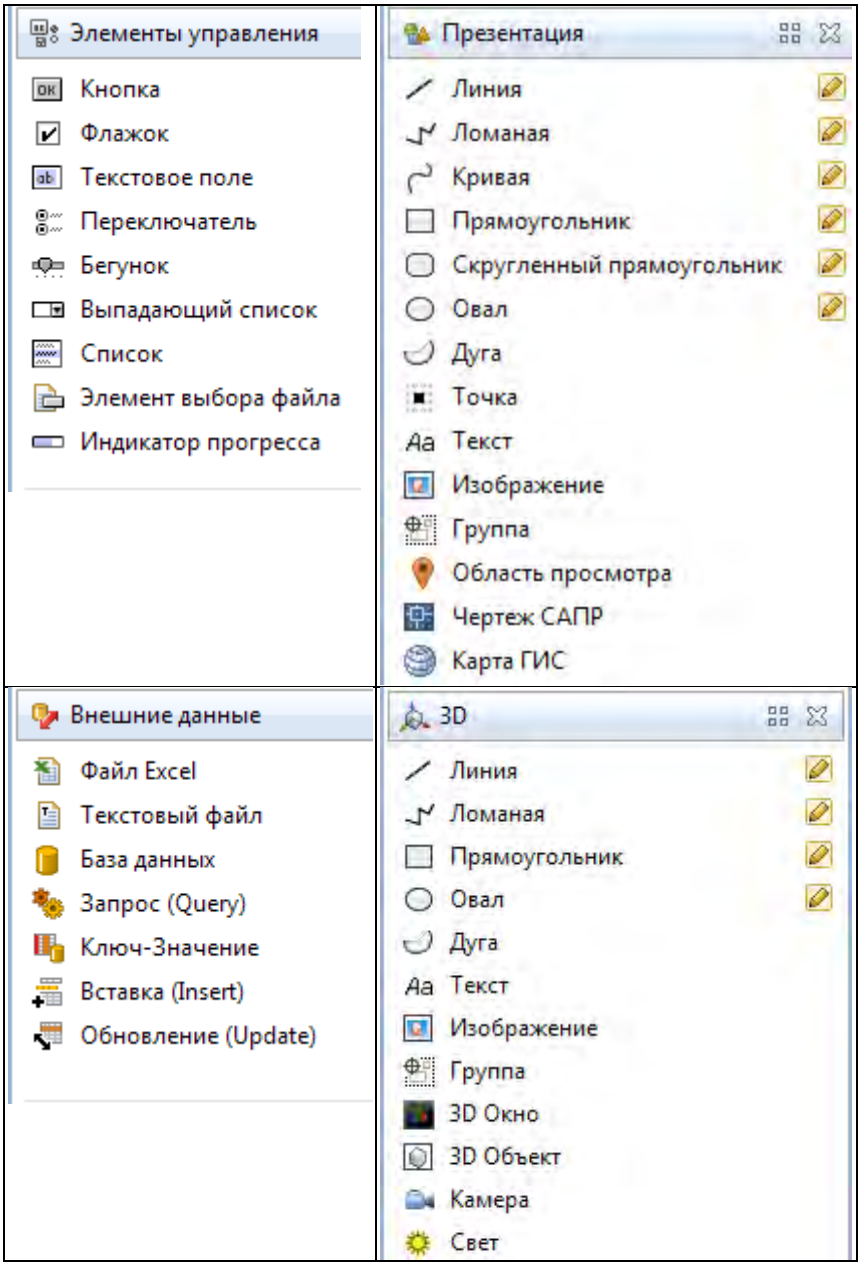
Палитры

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

- **Основная** - Палитра содержит основные элементы, с помощью которых Вы можете задать динамику модели,ее структуру и данные.
- **Системная динамика**- Палитра содержит элементы, часто используемые в системной динамике: элементы диаграммы потоков и накопителей, а также параметр, соединитель и табличную функцию.
- **Диаграмма состояний**- Палитра содержит блоки *диаграмм состояний* -диаграмм, позволяющих графически задавать поведение объекта.
- **Диаграмма действий**- Палитра содержит блоки *диаграмм действий* -структурированных блок-схем, позволяющих задавать алгоритмы визуально.
- **Статистика**- Палитра содержит элементы, используемые для сбора, анализа и отображения результатов моделирования.
- **Презентация**- Палитра содержит элементы, используемые для рисования презентаций моделей:примитивные фигуры, с помощью которых Вы можете рисовать сложные презентации.
- **3D**- Палитра содержит элементы, необходимые для создания сцены трехмерной анимации, а также набор фигур, с помощью которых Вы можете рисовать трехмерные анимации Ваших моделей.
- **Элементы управления**- Палитра содержит элементы управления, с помощью которых Вы можете сделать анимации Ваших моделей интерактивными.
- **Внешние данные**- Палитра содержит инструменты для работы с внешними данными -базами данных и текстовыми файлами.
- **Картинки** - Палитра содержит набор картинок наиболее часто моделируемых объектов: человек, медсестра, врач, грузовик, фура, погрузчик, склад,завод и т. д.
- **3D Объекты** - Палитра содержит набор трехмерных изображений наиболее часто моделируемых объектов: человек, медсестра, врач, грузовик, фура, погрузчик, поддон и т.д.

Элементы, блоки, изображения палитр приведены ниже.


Основная
Агент
Параметр
Событие
Динамическое событие
Переменная
Коллекция
Функция
Табличная функция
Эмпирическое распределение
Расписание
Порт
Соединитель
Связь с агентами
Статистика
Сбор данных
Набор данных
Статистика
Данные гистограммы
Данные двумерной гистограммы
Диаграммы
Столбиковая диаграмма
Диаграмма с накоплением
Круговая диаграмма
График
Временной график
Временная диаграмма с накопле...
Временная цветовая диаграмма
Гистограмма
Двумерная гистограмма
Диаграмма действий
Диаграмма действий
Код
Решение (If ... Else)
Локальная переменная
Цикл While
Цикл Do While
Цикл For
Вернуть значение (Return)
Выход из цикла (Break)
Системная динамика
Накопитель
Поток
Вспомогательная переменная
Связь
Параметр
Табличная функция
Цикл
Копия


Элементы управления
Кнопка
Флажок
Текстовое поле
Переключатель
Бегунок
Выпадающий список
Список
Элемент выбора файла
Индикатор прогресса
Презентация
Линия
Ломаная
Кривая
Прямоугольник
Скругленный прямоугольник
Овал
Дуга
Точка
Текст
Изображение
Группа
Область просмотра
Чертеж САПР
Карта ГИС
Внешние данные
Файл Excel
Текстовый файл
База данных
Запрос (Query)
Ключ-Значение
Вставка (Insert)
Обновление (Update)
3D
Линия
Ломаная
Прямоугольник
Овал
Дуга
Текст
Изображение
Группа
3D Окно
3D Объект
Камера
Свет

ПРИЛОЖЕНИЕ 3

Методика разработки моделей в AnyLogic:

постановка задачи;

формализованное описание;

выделение сегментов модели и их размещение по областям просмотра;

организация ввода исходных данных;

организация вывода результатов моделирования;

построение событийной части модели;

проведение экспериментов;

интерпретация полученных результатов;

концептуальные выводы.