

Boxoft Image To PDF Demo. Purchase from [www.Boxoft.com](http://www.Boxoft.com)

# Концептуальное проектирование систем в AnyLogic и GPSS World

2-е издание, исправленное

Боев В.Д.

Национальный Открытый Университет "ИНТУИТ"

2016

# Концептуальное проектирование систем в AnyLogic и GPSS World

2-е издание, исправленное

Боев В.Д.

Национальный Открытый Университет "ИНТУИТ"

2016

Концептуальное проектирование систем в AnyLogic и GPSS World/ В.Д. Боеv - М.: Национальный Открытый Университет "ИНГУИТ", 2016

Курс предназначен для проведения практических занятий 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 AnyLogic и GPSS World. Приводятся сравнительные оценки результатов моделирования разнородных дискретных процессов, полученных на моделях одной и той же проектируемой системы в GPSS World и AnyLogic. Интерпретируются результаты моделирования. Делаются концептуальные выводы о целесообразности показателей и параметров проектируемых систем.

(c) ООО "ИНГУИТ.РУ", 2013-2016

(c) Боеv В.Д., 2013-2016

## Введение

"Концепция проекта — это модель проекта. В ней должны быть на должном уровне обобщения выделены ключевые факторы (даны ответы на ключевые вопросы). Тебе, в процессе написания концепции (подготовки модели) должно стать понятно "что делать?" и "на каком основании это делать?" — ты должен почувствовать конструкцию процесса и понять, как его шевелить" (Клейн ссылка: [www.zen.ru](http://www.zen.ru) - <http://www.zen.ru>).

Пособие предназначено для проведения практических занятий по дисциплинам "Проектирование и моделирование систем", "Моделирование", "Компьютерное моделирование" с использованием систем имитационного моделирования.

При имитационном моделировании (ИМ) дискретных процессов в современной практике в качестве инструментального средства получила широкое распространение система общепринятого назначения GPSS World, являющаяся последним современным представителем семейства языков моделирования GPSS. Идеи, заложенные создателем GPSS Джоном Гордоном, использовались во многих последующих специализированных языках и средах ИМ.

В течение полувека, подвергаясь непрерывной эволюции, язык GPSS находил и находит применение в новых типах компьютеров и операционных систем. К началу XXI в. было создано около десятка версий языка GPSS. GPSS World, благодаря своим возможностям, наличию литературы и бесплатной студенческой (учебной) версии, широко распространена в нашей стране и за рубежом.

В последние годы наряду с ней применяется отечественная система моделирования AnyLogic (новая версия 6). AnyLogic разработана компанией XJTechnologies 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.

Необходимым условием для оценки возможностей новой системы моделирования является её способность воспроизводить модели одинаковых процессов с не меньшей эффективностью, чем это сделано с помощью других систем, в данном случае GPSS World. Эффективность (точность и достоверность) получаемых результатов GPSS World подтверждена многолетней практикой использования при проектировании сложных систем. AnyLogic существует уже более 10 лет, успешно применяется в различных предметных областях, в бизнес-среде. В статьях [1, 2, 3] и монографии [4] показана адекватность AnyLogic классическому инструменту дискретно-событийного моделирования GPSS World. Поэтому в пособии вопросы анализа и проектирования систем рассматриваются на примерах использования AnyLogic и GPSS World как одних из универсальных средств моделирования проектных решений. Изложение ведётся согласно подходу, который можно назвать "обучение на примерах" или "делай как мы".

В работе не ставилась задача дать всестороннюю оценку достоинствам и недостаткам обеих систем ИМ. Подготовленный читатель может, исходя из стоящих перед ним целей моделирования проектируемой системы, сформулировать оценки такого плана. Основное внимание сосредоточено на сравнительной оценке результатов моделирования, полученных на моделях одной и той же проектируемой системы. Адекватность достигалась также стремлением к идентичной реализации всех функций проектируемой системы при построении моделей средствами GPSS World и AnyLogic. Последнее обстоятельство привело авторов 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.

В приложении приводятся условные обозначения и функции объектов Основной библиотеки и элементов палитр AnyLogic.

Большое спасибо сотрудникам фирмы "Экс Джей Текнолоджис" П. А. Лебедеву, С. А. Суслову за плодотворное сотрудничество и рекомендации по построению моделей, а также руководству фирмы за предоставленную версию AnyLogic 6.

За допечатную подготовку автор благодарен Д. В. Боеву.

# Модель обработки запросов сервером

## Модель в GPSS World

При имитационном моделировании с использованием специальных инструментальных средств, например, GPSS World, в общем случае решаются две задачи. Назовем их прямой и обратной.

Прямая задача заключается в нахождении оценки математического ожидания какого-либо показателя моделируемой системы при заданном времени ее функционирования.

Обратная задача состоит в определении оценки математического ожидания времени функционирования системы, за которое какой-либо её показатель достигает заданного значения.

Решение этих задач, особенно обратной задачи, имеет свои особенности. Рассмотрим эти особенности далее на примере. Начнём построение GPSS-моделей с прямой задачи.

## Решение прямой задачи

### Постановка задачи

Сервер обрабатывает запросы, поступающие с автоматизированных рабочих мест (АРМ) с интервалами, распределенными по показательному закону со средним значением  $T_1 = 2$  мин. Сервер имеет входной буфер ёмкостью 5 запросов.

Вычислительная сложность запросов подчинена нормальному закону с математическим ожиданием  $S_1 = 6 \cdot 10^7$  оп и среднеквадратическим отклонением  $S_2 = 2 \cdot 10^5$  оп. Производительность сервера  $Q = 6 \cdot 10^5$  оп/с. В случае полной занятости входного буфера поступающий запрос теряется.

Построить имитационную модель обработки запросов сервером для

определения оценки математического ожидания количества запросов (далее - количества запросов), обработанных сервером за время функционирования  $T = 1$  час, и оценки математического ожидания вероятности обработки запросов (далее - вероятности обработки запросов).

### Уяснение задачи моделирования

Сервер представляет собой однофазную систему массового обслуживания разомкнутого типа с ожиданием и с отказами.

В модели для имитации источника запросов следует использовать блок GENERATE, для имитации сервера как одноканального устройства - блоки SEIZE и RELEASE, для имитации буфера - QUEUE и DEPART, обработки запросов - ADVANCE.

В модели должны быть следующие элементы:

- задание исходных данных;
- описание арифметических выражений;
- сегмент имитации поступления и обработки запросов;
- сегмент задания времени моделирования и расчета результатов моделирования.

Серверу дадим имя Server. Для вывода из модели транзактов, имитирующих обработанные и потерянные запросы, используем блоки TERMINATE с метками ObrZap и PotZap соответственно. Для счета количества всех запросов используем метку KolZap.

Выберем масштаб: 1 единице масштабного времени соответствует 1 с. Так как среднее значение интервалов поступления запросов  $T_1 = 2$  мин, то теперь это будет 120 ед. мод. времени.

Рассчитаем количество прогонов, которые нужно выполнить в каждом наблюдении, т. е. проведем так называемое тактическое планирование эксперимента. Пусть результаты моделирования (вероятность обработки запросов) нужно получить с доверительной вероятностью  $\alpha = 0,95$  и

точностью  $\varepsilon = 0,01$ . Расчет проведем для худшего случая, т. е. при вероятности  $\rho = 0,5$ , так как до эксперимента  $\rho$  неизвестно:

$$N = t_{\alpha}^2 \cdot \frac{\rho \cdot (1 - \rho)}{\varepsilon^2} = 1,96^2 \cdot \frac{0,5 \cdot (1 - 0,5)}{0,01^2} = 3,8416 \cdot \frac{0,25}{0,0001} = 9604$$

### Блок-диаграмма модели

Построим блок-диаграмму модели для решения прямой задачи, т. е. сегмент имитации поступления и обработки запросов и сегмент задания времени моделирования и расчета результатов моделирования ([Рис. 1.1](#)).

Блок-диаграмма представляет собой набор стандартных блоков. Она строится так. Из множества блоков выбирают нужные, и далее выстраивают их в диаграмму для того, чтобы в процессе функционирования модели они как бы взаимодействовали друг с другом. Диаграмма сопровождается необходимыми комментариями. Использование блоков при построении моделей зависит от логических схем работы реальных систем, моделируемых на ЭВМ.

Теперь приступим к написанию программы модели.

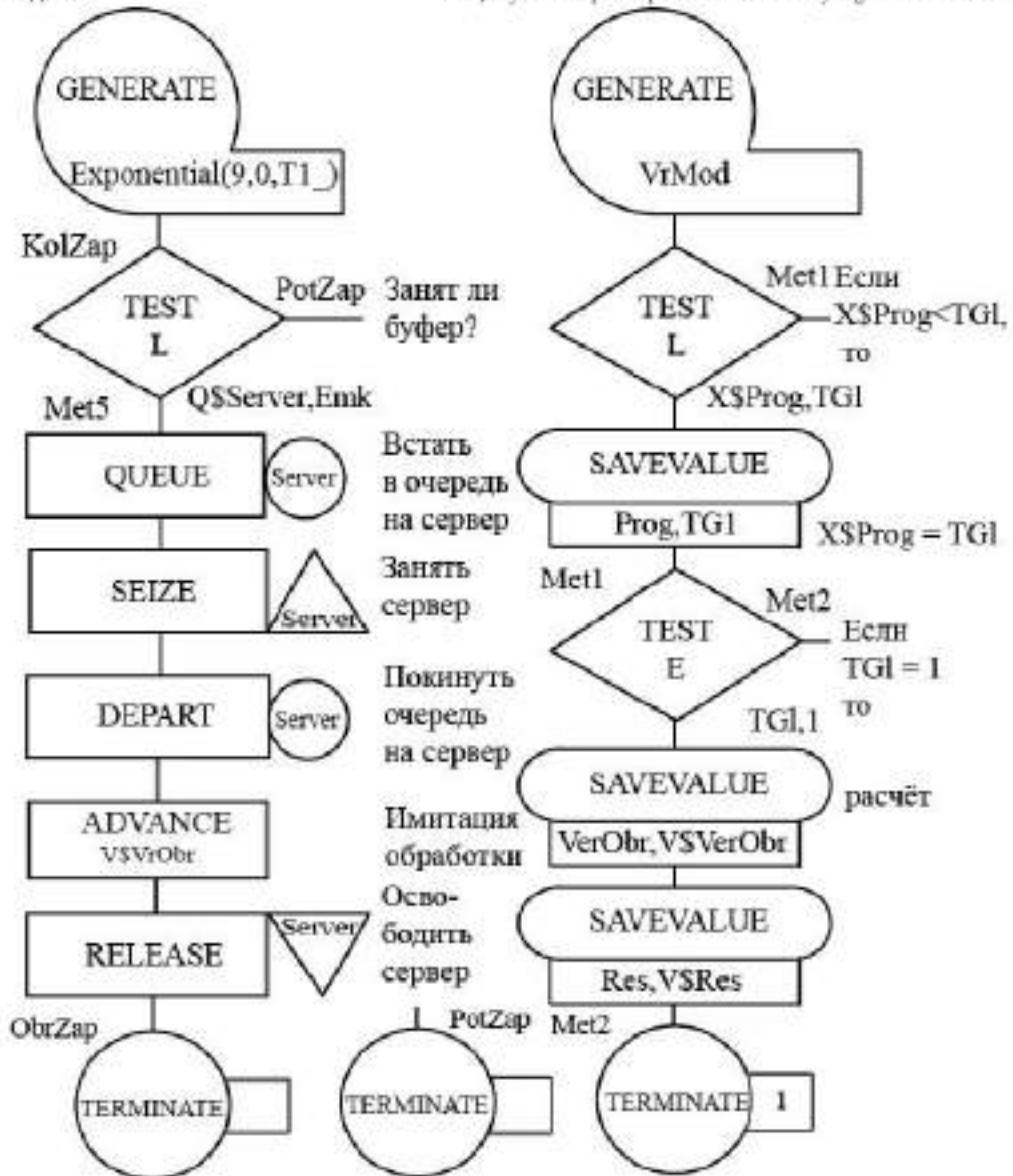


Рис. 1.1. Блок-диаграмма модели

### Программа модели

Для задания исходных данных используем переменные пользователя. Они задаются с помощью команды **EQU**. Переменным пользователя

даны такие же имена, как и в постановке задачи, но добавлен знак подчеркивания. Например, `T1_`, `S1_` и т. д. Время моделирования зададим переменной пользователя `VrMod`.

Арифметическая переменная для расчета времени обработки `VerObr` запроса на сервере:

```
VerObr VARIABLE (Normal(2,(S1_#Koef),(S2_#Koef)))/Q_
```

Переменная пользователя `Koef` введена для удобства изменения (пропорционального изменения) характеристик нормального закона распределения, которому подчиняется вычислительная сложность запросов. Особенно целесообразно использование этой переменной при проведении экспериментов.

Вероятность обработки `VerObr` запросов на сервере будем определять как отношение количества обработанных `N$ObrZap` запросов к количеству всего поступивших `N$KolZap` запросов:

```
VerObr VARIABLE N$ObrZap/N$KolZap
```

В арифметическом выражении `VerObr`, например, `N$ObrZap` - системный числовой атрибут - количество транзактов, вошедших в блок с меткой `ObrZap`, а `N$KolZap` - количество транзактов, вошедших в блок с меткой `KolZap`.

Количество обработанных запросов определяется арифметическим выражением:

```
Res VARIABLE INT(N$ObrZap/X$Prog)
```

Все необходимое для написания программы модели имеется. Напишем программу модели для решения прямой задачи.

; Обработка запросов сервером. Прямая задача

; Задание исходных данных

```
T1_ EQU 120 ; Средний интервал поступления запросов, с
S1_ EQU 6000000 ; Среднее значение вычислительной сложности эпсилон
S2_ EQU 200000 ; Стандартное отклонение вычислительной сложности эпсилон
Q_ EQU 600000 ; Средняя производительность сервера, оп/с
```

```

Emk EQU 5 ; Ёмкость входного буфера
Koef EQU 1 ; Коэффициент изменения характеристик нормального р
VrObr VARIABLE (Normal(9,(S1_#Koef),(S2_#Koef))/Q_
VerObr VARIABLE N$ObrZap/N$KolZap
Res VARIABLE INT(N$ObrZap/X$Prog)
VrMod EQU 3600 ; Время моделирования, 1 ед. мод. времени = 1 с.
; Сегмент имитации обработки запросов
GENERATE (Exponential(2,0,T1_)) ; Источник запросов
KolZap TEST L Q$Server,Emk,PotZap ; Занят ли буфер? QUEUE Serve
SEIZE Server ; Занять сервер
DEPART Server ; Покинуть очередь к серверу
ADVANCE V$VrObr ; Имитация обработки запроса
SAVEVALUE SumTime+,M1; Время обработки всех запросов
RELEASE Server ; Освободить сервер
ObrZap TERMINATE ; Обработанные запросы
PotZap TERMINATE ; Потерянные запросы
; Сегмент задания времени моделирования и расчета результатов
GENERATE VrMod
TEST L X$Prog,TG1,Met1 ; Если X$Prog < TG1,
SAVEVALUE Prog,TG1 ; то X$Prog = TG1
Met1 TEST E TG1,1,Met2 ; Если TG1 = 1, то
SAVEVALUE VerObr,V$VerObr ; расчет и сохранение в ячейке VerObi
SAVEVALUE Res,V$Res ; числа обработанных запросов
SAVEVALUE TimeMean,(X$SumTime/N$ObrZap)
Met2 TERMINATE 1
START 9604 ; Количество прогонов модели

```

При расчете количества обработанных запросов Res в арифметическом выражении N\$ObrZap/X\$Prog используется число прогонов. В арифметическом выражении указано не явное число прогонов, а в виде содержимого ячейки X\$Prog. Число прогонов заносится предварительно в эту ячейку по завершении первого прогона модели, но до того момента, когда из счетчика завершений TG1 будет вычтена первая единица. В этом случае арифметическое выражение не зависит от числа прогонов, которое может меняться на различных этапах создания и эксплуатации модели, в том числе и в зависимости от исходных данных, а также от точности и достоверности результатов моделирования. Поскольку количество обработанных запросов не может быть дробным числом, то для получения целого числа, записываемого в

ячейку `Res`, используется процедура `INT` из встроенной библиотеки.

Среднее время `X$TimeMean` обработки одного запроса определяется как отношение суммарного времени `X$SumTime` к количеству обработанных запросов `N$ObjZap`. В данной модели можно определять `X$TimeMean` как сумму средних времен обработки одного транзакта на сервере и среднего времени задержки в очереди.

Для уменьшения машинного времени расчет искомых показателей производится не после каждого прогона, а после завершения последнего прогона, т. е. когда содержимое счетчика завершений будет равно единице (`TG1 = 1`).

Ввод текста программы модели, исправление ошибок и проведение моделирования

1. Запустите GPSS World.
2. Закройте окно напоминания о необходимости обновления "Заметок". Откроется главное меню.
3. Для ввода текста GPSS World имеет текстовый редактор. Откройте окно текстового редактора. Для этого выберите в меню `File / New` и в появившемся меню выберите `Model`. Нажмите `Ok`.
4. Наберите текст программы модели, т. е. создайте объект "Модель". При вводе текста следует использовать клавишу `[Tab]`. Например, после набора `T1_` нужно нажать клавишу `[Tab]`. Интервалы табуляции установлены по умолчанию.
5. После ввода программы модели создайте объект "Процесс моделирования", представляющий собой оттранслированный объект "Модель". Для трансляции выберите `Command / Create Simulation`. По этой команде транслятор GPSS проверяет программу модели на наличие синтаксических ошибок.
6. При наличии синтаксических ошибок система в окне `JOURNAL` выдаст список сообщений об ошибках трансляции. Перейдите к п.
7. При отсутствии ошибок в окне `JOURNAL` появится сообщение `Model Translation Begun. Ready.` Перейдите к п. 9.
7. Исправьте ошибки. Для поиска ошибок в тексте программы модели и их исправления используйте команду `Search / Next`

Error, предварительно перейдя из окна JOURNAL в текст программы модели. При первом выполнении этой команды курсор мыши помещается в строке текста модели с ошибкой. После исправления первой ошибки вновь используйте команду Search / Next Error и т.д. столько раз, сколько ошибок в тексте программы.

8. После исправления ошибок перейдите к п.5.
9. Сохраните модель. Для этого выберите в главном меню File / Save As. Дайте модели имя Модель процессов изготовления изделий и нажмите Ok.
10. Откликом в данной модели является вероятность обработки запросов сервером. Ранее вы рассчитали количество прогонов модели для худшего случая при точности  $\varepsilon = 0,01$ , и доверительной вероятности  $\alpha = 0,95 : N = 9604$ .
11. Запустите модель. Для этого в главном меню выберите Command / Start и в диалоговом окне вместо 1 наберите 9604. Нажмите Ok.
12. При наличии логических ошибок для их поиска в тексте программы модели и исправления используйте команду Search / Goto Line столько раз, сколько ошибок указано в окне JOURNAL. Там же (в окне JOURNAL) указаны номера строк с ошибками.
13. При отсутствии логических ошибок в модели по окончании её работы система GPSS World автоматически создает стандартный отчет, который появится в окне Report. В окне будут содержаться следующие данные:
  - об именах объектов модели;
  - блоках модели;
  - ОКУ и МКУ;
  - очередях;
  - сохраняемых величинах.

В результате решения прямой задачи получим, что за один час сервером будет обработано  $N = 29$  запросов, а вероятность обработки составит  $VerObr = 0,97$ . Если не использовать процедуру INT - выделения целого числа с отбрасыванием дробной части, будет обработано 29,161 запроса. Среднее время обработки одного запроса составит TimeMean

= 255,262.

## Дисперсионный анализ (отсеивающий эксперимент)

Сущность этого эксперимента состоит в проведении многофакторного дисперсионного анализа с целью выявления степени влияния различных факторов и их комбинаций (взаимодействий) на значение целевой функции (функции отклика, представленной в виде уравнения регрессии).

Для каждого фактора необходимо выбрать два уровня - нижний и верхний. Рекомендуется выбирать уровни, значительно отстоящие друг от друга. Это необходимо для получения также значительно отличающихся откликов.

В условиях прямой задачи требуется исследовать зависимость вероятности обработки запросов от трех факторов, например, при следующих их минимальных и максимальных значениях ([Табл. 1.1](#)):

Таблица 1.1.

Уровни факторов	Факторы		
	T1_	с Koeff Q_	оп/с
Нижний	60	0.5	300000
Верхний	180	1.5	700000

Для проведения дисперсионного анализа нужно воспользоваться созданным объектом "Модель". В программе модели удалите последнюю строку.

Откройте модель Прямая задача. Выберите *Edit / Insert Experiment / Screening ...* (Правка / Вставить эксперимент / Отсеивающий ...).

Откроется диалоговое окно *Screening Experiment Generator* (Генератор отсеивающего эксперимента) ([Рис. 1.2](#)).

Приступите к заполнению полей диалогового окна.

В поля **Experiment Name** (Имя эксперимента) и **Run Procedure Name** (Имя процедуры запуска) введите, например, **Dis\_Server** и **Dis\_Server\_Run** соответственно (Рис. 1.3).

Имена эксперименту и процедуре запуска эксперимента даёт пользователь.

Дальше расположена группа полей **Factors** (Факторы). В рассматриваемом примере определяется вероятность обработки запросов, поступающих на сервер. Факторы, влияние которых необходимо исследовать, были определены нами ранее (см. табл. 1.1).

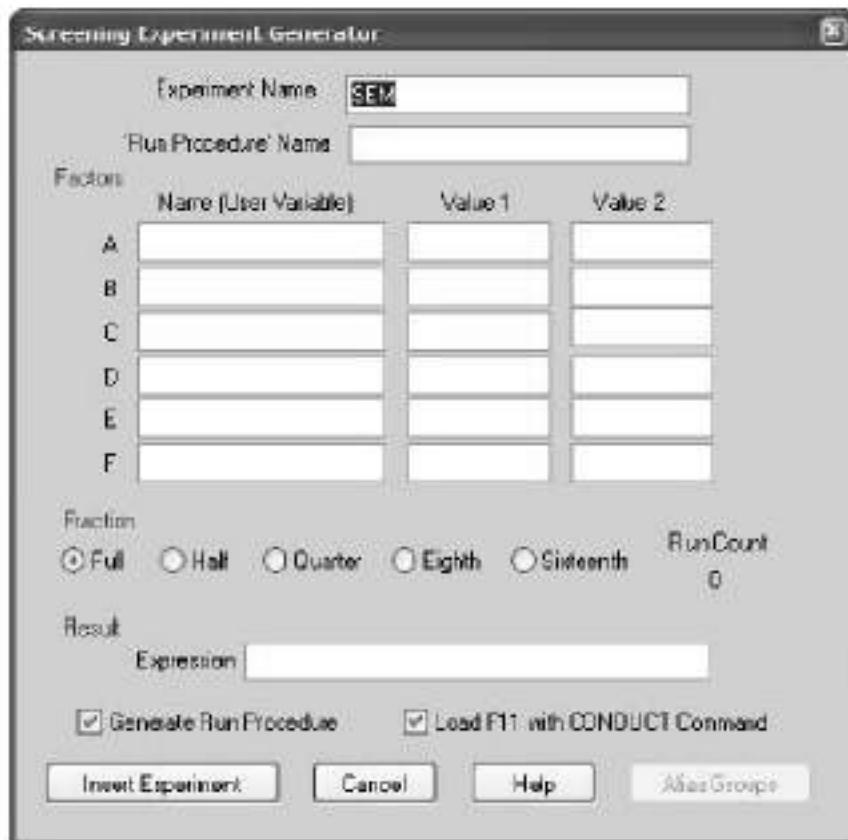


Рис. 1.2. Диалоговое окно (незаполненное) Screening Experiment Generator (Генератор отсеивающего эксперимента)

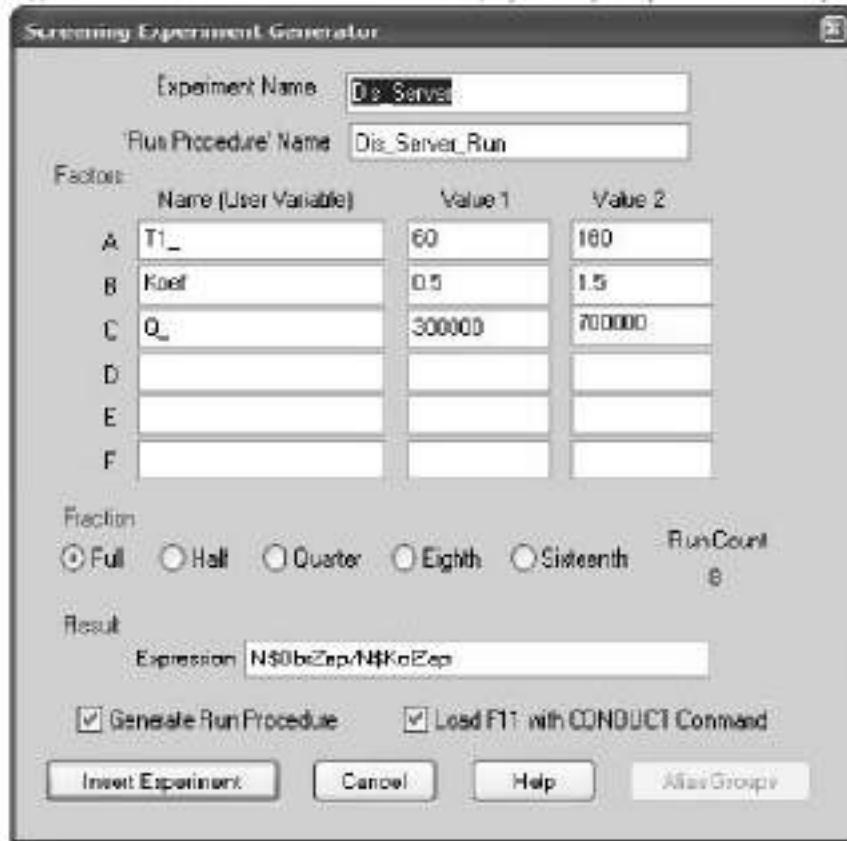


Рис. 1.3. Диалоговое окно (заполненное) Screening Experiment Generator (Генератор отсеивающего эксперимента)

В GPSS World максимальное количество факторов, влияние которых на функцию отклика можно исследовать посредством дисперсионного анализа, равно шести.

Введите ранее выбранные факторы, начиная с фактора А. В поле *Name (User Variable)* (Имя (Переменная пользователя)) введите имя фактора, в поля *Value1* и *Value2* - его нижний и верхний уровни соответственно. После ввода всех факторов для дальнейшей работы будем иметь факторы А, В и С.

Ниже идет группа *Fraction* (Часть полного эксперимента). Эксперимент, проводимый в GPSS World, может быть полным факторным экспериментом (ПФЭ) или дробным факторным

экспериментом (ДФЭ). Группа Fraction (Часть дробного эксперимента) позволяет это задавать, т. е. позволяет провести стратегическое планирование эксперимента, цель которого, как вам известно, является определение количества наблюдений и сочетаний уровней факторов в них для получения наиболее полной и достоверной информации о поведении системы.

Установка ПФЭ соответствует кнопка Full, для ДФЭ в 1/2 от ПФЭ - Half, в 1/4 - Quarter, в 1/8 - Eight, в 1/16 - Sixteen.

Установите пока Half (1/2). Справа под Run Count появится число 4, так как  $2^2 = 4$ . Это количество наблюдений, которое необходимо сделать. Количество прогонов в каждом наблюдении будет указано позже.

В поле Expression (Выражение) группы Result (Результат) введите выражение, по которому вычисляется вероятность обработки запросов: NS0br2ap/NSKolZap.

После группы Result (Результат) расположены два флажка, позволяющие выбирать опции.

При выборе опции Generate Run Procedure вместе с экспериментом создается стандартная процедура запуска, которую пользователь может корректировать согласно своим требованиям.

Выбор второй опции Load Fil with CONDUCT Command закрепляет команду CONDUCT за функциональной клавишей F11. Тогда после создания объекта "Процесс моделирования" для запуска эксперимента нужно только нажать функциональную клавишу F11. Выберите обе опции.

Перед созданием эксперимента необходимо изучить группы смешивания с целью осуществления стратегического планирования эксперимента. Для этого нужно нажать кнопку Alias Groups (Группы смешивания). Появится диалоговое окно Alias Groups (Группы смешивания) (Рис. 1.4).

При изучении групп смешивания необходимо вначале найти

отсутствующие факторы, а затем факторы, которые неразличимы, так как находятся в одной группе смешивания. Например, взаимодействие факторов А и В - АВ.



Рис. 1.4. Диалоговое окно Alias Groups (Группы смешивания)

Из рис. 1.4 видно, что отсутствующих факторов нет. Факторы А, В и С находятся в различных группах смешивания по два фактора в каждом. Невозможно будет судить об эффектах, т. е. о влиянии на отклик взаимодействий двух факторов. В некоторых случаях этого будет достаточно.

Нажмите кнопку Cancel (Отмена).

В диалоговом окне Screening Experiment Generator (Генератор отсеивающего эксперимента) в группе Fraction (Часть дробного эксперимента) установите Full (ПФЭ). Под Run Count появится число 8.

Обратите внимание, что кнопка Alias Groups (Группы смешивания) при установке полного факторного эксперимента Full (ПФЭ) не будет активной.

Теперь необходимо создать Plus - операторы и вставить их в нижнюю часть модели Прямая задача. Для этого нажмите кнопку Insert Experiment (Вставить эксперимент), расположенную в левой нижней части диалогового окна Screening Experiment Generator (Генератор отсеивающего эксперимента).

Так как была выбрана опция Generate Run Procedure, то создана стандартная процедура запуска с именем Dis\_Server\_Run. Появится ее диалоговое окно, дающее возможность пользователю изменить процедуру запуска согласно своим требованиям (Рис. 1.5).



Рис. 1.5. Диалоговое окно стандартной процедуры запуска



Рис. 1.6. Условия стандартной процедуры запуска по умолчанию

Перейдите, пользуясь клавишами вверх-вниз, в конец процедуры запуска. Там в разделе Set up your own run conditions (Задайте свои условия наблюдения) имеются две команды

START, между которыми находится команда RESET (Рис. 1.6).

Поясним назначение этих команд.

Первой командой START

```
DoCommand("START 100,NP"); /*Get past the Startup Period. */
```

определяется количество прогонов в неустоявшемся режиме.

Подразумевается, что если моделирование выполняется долго, то система приходит в стационарное состояние. Сколько времени следует вести моделирование, чтобы достичь стационарного состояния? Часто ответ на этот вопрос можно получить из опыта экспериментирования с моделью. Команда RESET служит для этого. Она сбрасывает в ноль накопленную на неустоявшемся режиме статистику без удаления транзактов из процесса моделирования.

Второй командой START

```
DoCommand("START 1000,NP"); /*Run the Simulation. */
```

определяется количество прогонов в наблюдении, т. е. количество прогонов, которое было определено ранее при тактическом планировании эксперимента: N = 9604. Измените 1000 на 9604 (Рис. 1.7).

Корректировка процедуры запуска возможна до и после того, как она будет добавлена к объекту "Модель".



Рис. 1.7. Условия стандартной процедуры запуска после корректировки

После корректировки нажмите **Ok**.

Сгенерированный Plus - эксперимент представлен ниже. Изучите его. Это необходимо для создания собственных экспериментов, отличающихся от стандартных экспериментов GPSS World.

В начале автоматически сгенерированного эксперимента определяется и инициализируется в неопределенное состояние (**UNSPECIFIED**) матрица результатов. Далее имеются Plus - операторы, которые для каждого из наблюдений определяют сочетания уровней факторов. В рассматриваемом примере таких сочетаний восемь.

Plus - эксперимент содержит также вызов Plus - процедуры запуска. Процедура запуска осуществляет связь между генерируемым экспериментом и процессом моделирования. Она вызывается столько раз, сколько требуется сделать наблюдений. Так как процедура запуска вызывается Plus - экспериментом, ей разрешается вызывать библиотечную процедуру **DoCommand** и, следовательно, выполнять **RMULT**, **CLEAR**, **RESET** и многие другие команды GPSS. Поэтому все команды, необходимые для определения условий наблюдения, например, обнуление сохраняемых ячеек, следует помещать в процедуру запуска.

Для сохранения матрицы результатов при обнулении переменных перед очередным наблюдением используется команда CLEAR OFF. Для изменения начального числа генератора случайных чисел в каждом наблюдении процедуре передается номер запуска.

```
*****
* Dis_Server *
* Факторный отсеивающий эксперимент *
*****
Dis_Server_Results MATRIX ,2,2,2
INITIAL Dis_Server_Results,UNSPECIFIED
Dis_Server_NextRunNumber EQU 0
EXPERIMENT Dis_Server() BEGIN

/* Наблюдение 1 */
T1_ = 60;
Koef = 0.5;
Q_ = 300000;
IF (StringCompare(DataType(Dis_Server_Results[1,1,1]),
"UNSPECIFIED")'E'0)
THEN BEGIN
/* Установить начальное значение переменной количества наблюдений
Dis_Server_NextRunNumber = 1;
/* Записать данные наблюдения и запустить процесс моделирования*/
Dis_Server_GetResult();
Dis_Server_Results[1,1,1] = NSObrZap/N$KolZap;
END;
/* Наблюдение 2 */
T1_ = 60;
Koef = 0.5;
Q_ = 700000;
IF (StringCompare(DataType(Dis_Server_Results[1,1,2]),
"UNSPECIFIED")'E'0)
THEN BEGIN
/* Записать данные наблюдения и запустить процесс моделирования */
Dis_Server_GetResult();
Dis_Server_Results[1,1,2] = NSObrZap/N$KolZap;
END;
/* Наблюдения 3 - 7 для краткости пропущены */
```

```

/* Наблюдение 8 */
T1_ = 180;
Koef = 1.5;
Q_ = 700000;
IF (StringCompare(DataType(Dis_Server_Results[2,2,2]),
    "UNSPECIFIED")'E'0)
THEN BEGIN
/* Записать данные наблюдения и запустить процесс моделирования */
Dis_Server_GetResult();
Dis_Server_Results[2,2,2] = N$ObrZap/N$KoIZap;
END;
/* Эффекты смешивания в дробном факторном эксперименте */
SE_Effects(Dis_Server_Results,"I");
END;
*****  

* Процедура запуска наблюдения *
*****  

PROCEDURE Dis_Server_GetResult() BEGIN
/* Выполнить указанное число прогонов и записать результаты. */
/* Факторы для этого наблюдения уже были определены. */
TEMPORARY CurrentYield>ShowString>CommandString;
/* Вызов процедуры запуска */
Dis_Server_Run(Dis_Server_NextRunNumber);
CurrentYield = N$ObrZap/N$KoIZap;
ShowString = PolyCatenate("Run ",String(Dis_Server_NextRunNumber),". ");
ShowString = PolyCatenate(ShowString," Yield=",String(CurrentYield),". ");
ShowString = PolyCatenate(ShowString," T1_=",String(T1_),";");
ShowString = PolyCatenate(ShowString," Koef=",String(Koef),";");
ShowString = PolyCatenate(ShowString," Q_=",String(Q_),";");
CommandString = PolyCatenate("SHOW """,ShowString,"""", " ");
DoCommand(CommandString);
Dis_Server_NextRunNumber = Dis_Server_NextRunNumber + 1;
RETURN CurrentYield;
END;
*****  

* Процедура запуска *
*****  

PROCEDURE Dis_Server_Run(Run_Number) BEGIN
DoCommand("CLEAR OFF"); /* Использовать OFF для сохранения

```

```

/* Увеличьте число команд RMULT, если у вас большее число ГСЧ. */
/* Задать новые случайные числа всем потокам случайных чисел. */
TEMPORARY CommandString;
/* Вычислить, прежде чем перейти к DoCommand.*/
CommandString = Catenate("RMULT ",Run_Number#111);
/* DoCommand контролирует строку в глобальном контексте.*/
DoCommand(CommandString);
/* Установить собственные условия наблюдения.*/
DoCommand("START 100,NP"); /* Пройти неустоявшийся режим.*/
DoCommand("RESET"); /* Начать период измерений.*/
DoCommand("START 9604,NP"); /* Провести моделирование.*/
END;

```

Проведем эксперимент. Для вызова эксперимента предназначена команда CONDUCT. Однако за функциональной клавишей [F11] была закреплена соответствующая команда CONDUCT (Edit / Settings / Function Keys (Правка / Настройки / Функциональные клавиши)).

Проведите трансляцию, т. е. создайте объект "Процесс моделирования", для чего нажмите [Ctrl]+[Alt]+[S] или выполните команду Command / Create Simulation (Команда / Создать процесс моделирования).

При отсутствии ошибок в сгенерированном эксперименте в окне Journal (Журнал) появится сообщение (Рис. 1.8), свидетельствующее об отсутствии ошибок. Нажмите функциональную клавишу [F11]. Эксперимент начинает работать.

Замечание. Во время эксперимента доступна только команда HALT. Остальные команды неактивны, т. е. процесс моделирования можно только остановить и потом продолжить, но просмотреть его с использованием меню, вызываемого командой WINDOW / SIMULATION WINDOW и другими командами, нельзя.

В ходе выполнения сгенерированного эксперимента автоматически создается отчет, который по готовности записывается в окно Journal (Журнал) объекта "Процесс моделирования". Фрагмент отчета для четырех наблюдений (Run1 ... Run4) показан на рис. 1.9. В отчете

содержатся Yield - целевая функция и значения факторов, при которых получение значение целевой функции.

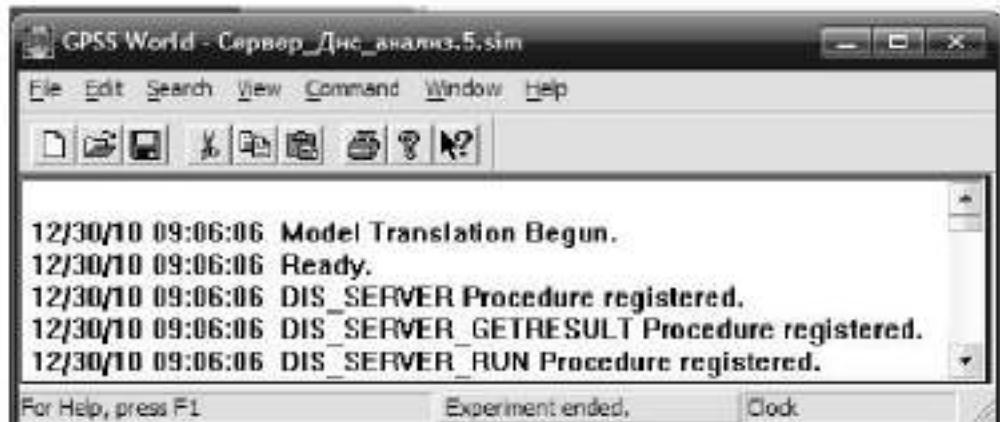


Рис. 1.8. Окно Journal (Журнал) с сообщением об успешном создании объекта 'Процесс моделирования'

GPSS World - Сервер_Дис_анализ.5.sim						
File Edit Search View Command Window Help						
A set of toolbar icons for file operations like Open, Save, Print, etc.						
12/30/10 09:06:34	Alias Group	Effect	Sum of Squares	Degrees of Freedom	F-for Only Main Effects	Critical Value of F (p=.05)
12/30/10 09:06:34	A	0.333	0.221	1	10.600	7.71
12/30/10 09:06:34	B	-0.333	0.221	1	10.598	7.71
12/30/10 09:06:34	AB	0.128				
12/30/10 09:06:34	C	0.263	0.138	1	6.603	7.71
12/30/10 09:06:34	AC	-0.064				
12/30/10 09:06:34	BC	0.067				
12/30/10 09:06:34	ABC	0.120				
12/30/10 09:06:34	Error		0.084	4		
12/30/10 09:06:34	Total		0.664	7		
12/30/10 09:06:34	Grand Mean		0.731			
12/30/10 09:06:34	Experiment ended.					
For Help, press F1	Experiment ended.			Clock		

Рис. 1.9. Окно Journal (Журнал) с отчетами по каждому наблюдению

Так как эксперимент включает 8 наблюдений по 9604 прогонов в

каждом из них, то будет выдано 8 отчетов (на рис. 1.9 в целях сокращения показаны только первые четыре отчета). Окончательные результаты моделирования после статистической обработки будут выведены в виде таблицы Anova (Рис. 1.10).

В таблице каждый фактор и взаимодействие факторов представлены отдельной строкой. В каждой строке для всех эффектов указаны коэффициенты, с которыми они входят в целевую функцию (столбец Effect), а для главных эффектов (A, B, C) - суммы квадратов отклонений - столбец Sum of Squares.

В столбце Degrees of Freedom приведены степени свободы соответствующих измерений.

В столбце F-for Only Main Effects - вычисленные значения F-статистик для главных эффектов, а в столбце Critical Value of F ( $p=0,5$ ) - соответствующие критические значения F-распределения для уровня значимости 50%.

В строке Error показаны остаточная составляющая дисперсии и соответствующая степень свободы.

В строке Total - общая сумма квадратов ошибок по всему эксперименту.

В строке Grand - среднее значение результата исследования (в примере - вероятности) по данным всего эксперимента.

GPSS World [Server_Процессы_запросы2.htm - JOURNAL]						
File Edit Search View Command Window Help						
		Alias	Effect	Sum of Squares	Degrees of Freedom	F-for Only Main Effects
05/07/08 08:54:23		Group				Critical Value of F (p=.05)
05/07/08 08:54:23	A		0.241	0.116	1	355.658
05/07/08 08:54:23	B		-0.241	0.116	1	356.055
05/07/08 08:54:23	AB		-0.005			
05/07/08 08:54:23	C		0.184	0.068	1	267.873
05/07/08 08:54:23	AC		0.004			
05/07/08 08:54:23	BC		-0.003			
05/07/08 08:54:23	ABC		0.025			
05/07/08 08:54:23	Error			0.001	4	
05/07/08 08:54:23	Total			0.301	7	
05/07/08 08:54:23	Grand Mean			0.481		
05/07/08 08:54:24						
05/07/08 08:54:24 Experiment ended.						
For Help, press F1		Experiment ended.			Clock	

Рис. 1.10. Результаты дисперсионного анализа

Чем больше значение F-статистики (F-for Only Main Effects), тем сильнее эффект. Эффект, а, следовательно, и фактор, считается значимым, если превышает критическое значение (Critical Value of F (p=0,5)).

В данном примере факторы А и В являются значимыми, так как их F-статистики больше критического значения, равного 7,71. Обратите внимание, что эффекты факторов А и В противоположны.

Таким образом, по результатам моделирования можно сделать вывод, что при данном потоке и характеристиках сервера вероятность обработки запросов в среднем составляет 0,731, т. е. вероятность потерь запросов составляет 0,269. Для уменьшения потерь запросов нужно продолжить исследование каждого значимого фактора А и В.

Для того чтобы определить среднее количество запросов, которые будут обработаны при таких же значениях факторов, проведите ещё один дисперсионный анализ.

Для этого удалите генерированный эксперимент из программы модели. Затем выберите *Edit / Insert Experiment / Screening ...* (Правка / Вставить эксперимент / Отсеивающий ...). Всё, что вы ранее вводили ([см. рис. 1.3](#)), останется неизменным. Вам нужно будет только заменить в поле *Expression* (Выражение) группы *Result* (Результат) выражение, по которому вычисляется вероятность обработки запросов, выражением для расчёта количества обработанных запросов:  $N\$ObrZap/X\$Prog$ .

После замены вставьте эксперимент и выполните его. Вы получите, что среднее количество обработанных запросов составит 25,889, а все факторы будут несущественными.

## Решение обратной задачи

Для решения обратной задачи возьмем количество запросов, ожидаемое время обработки которых нужно определить,  $N = 29$  - результат решения прямой задачи.

Программа модели приведена ниже.

```
; Обработка запросов сервером. Обратная задача
; Задание исходных данных
T1_ EQU 120 ; Средний интервал поступления запросов, с
S1_ EQU 60000000 ; Среднее значение вычислительной сложности зк
S2_ EQU 200000 ; Стандартное отклонение вычислительной сложнос
Q_ EQU 600000 ; Средняя производительность сервера, оп/с
Emk EQU 5 ; Ёмкость входного буфера
Koef EQU 1 ; Коэффициент изменения характеристик нормального р
Koef1 EQU 1 ; Коэффициент учета дробной части
N_ EQU 29 ; Количество запросов
; Сегмент имитации обработки запросов
GENERATE (Exponential(1,0,T1_)) ; Источник запросов
KolZap TEST L Q$Server,Emk,PotZap ; Занят ли буфер? QUEUE Serve
SEIZE Server ; Занять сервер
DEPART Server ; Покинуть очередь к серверу
ADVANCE ((Normal(2,(S1_#Koef),(S2_#Koef))/Q_) ; Имитация обрабо
RELEASE Server ; Освободить сервер
```

TRANSFER ,ObrZap ; Запрос отправляется в сегмент завершения мод  
 PotZap TERMINATE ; Потерянные запросы  
 ; Сегмент организации завершения моделирования и расчета результатов  
 ObrZap TEST L X\$Prog,TG1,Met1 ; Если X\$Prog < TG1,  
 SAVEVALUE Prog,TG1 ; то X\$Prog = TG1  
 SAVEVALUE NZap,0 ; Обнуление счетчика обработанных запросов  
 Met1 SAVEVALUE NZap+,1 ; Счет количества обработанных запросов  
 TEST E X\$NZap,N\_Ter1 ; Если X\$NZap = N\_, то  
 TEST E TG1,1,Met2 ; если TG1 = 1, то  
 SAVEVALUE VerObr,(N\$ObrZap/N\$KolZap) ; расчет и сохранение в ячейку  
 SAVEVALUE TimeNZap,((AC1-X\$AC2)/(X\$Prog#Koeff))  
 ; расчет и сохранение в ячейку TimeNZap времени обработки запросов  
 SAVEVALUE AC2,AC1 ; Запомнить абсолютное модельное время в ячейку  
 Met2 SAVEVALUE NZap,0 ; Обнуление счетчика обработанных запросов  
 TERMINATE 1  
 Ter1 TERMINATE  
 START 1000,NP ; Прогоны до установившегося режима  
 RESET ; Сброс накопленной статистики  
 START 9604 ; Количество прогонов модели

При решении обратной задачи один прогон определяется заданным количеством запросов  $N_$ , которые нужно обработать сервером, а не временем моделирования. Для этого организован счетчик обработанных запросов в виде сохраняемой ячейки  $X$NZap$ . Как только содержимое  $X$NZap = N_$ , из счетчика завершений вычитается единица. Таким образом, фиксируется один прогон модели. После этого ячейка  $X$NZap$  обнуляется и начинается очередной прогон.

Для расчета времени обработки заданного количества запросов используется арифметическое выражение  $(AC1-X$AC2)/X$Prog$ . В состав этого выражения входит абсолютное модельное время  $AC1$  и опять количество прогонов. Запоминается количество прогонов также как и при решении прямой задачи.

Кроме этого, в арифметическом выражении есть сохраняемая ячейка  $X$AC2$ . Дело в том, что команда **RESET** не влияет на абсолютное модельное время  $AC1$ . Время же выполнения 1000 прогонов до установившегося режима не должно участвовать в расчёте. Поэтому оно

запоминается, а затем вычитается из абсолютного модельного времени выполнения  $1000 + 9604 = 10604$  прогонов. Количество прогонов до установившегося режима может быть и другим.

В результате моделирования получим среднее время обработки 29 запросов 3579,401 с.

Фрагмент из отчета моделирования приведен ниже:

SAVEVALUE	RETRY	VALUE
PROG	0	9604.000
NZAP	0	0
VEROBR	0	0.971
TIMENZAP	0	3579.401

А почему не 3600 сек? Ведь это же время моделирования было задано при решении прямой задачи? Потому что мы отбросили дробную часть, т. е. взяли 29, а не 29,161. Как же поступить, чтобы учесть и отброшенную дробную часть? Ведь в счётчике фиксируются обработанные запросы только целыми числами, а не дробными?

Для учёта десятых долей дробной части зададим  $N_ = 291$ , т. е. увеличим в 10 раз. Это нужно учесть и в арифметическом выражении:  $((AC1-X$AC2) / (X$Prog#Koef1))$ . Переменной пользователя Koef1 задается значение 10. По завершении моделирования получим 3594,826 с. Этот результат уже ближе к 3600.

Для учёта сотых долей дробной части установим  $N_ = 2916$ , а Koef1 = 100. Получим 3602,099.

Вероятность обработки запросов в обоих случаях практически одна и та же, т. е. 0,971. Однако время моделирования существенно возрастает: 2 с, 21 с и 3 мин 34 с соответственно, т. е. более чем в 10 и 100 раз.

В примере обратной задачи также показано, что арифметические выражения можно не описывать отдельно до блоковой части программы вместе с заданием исходных данных (как в программе модели прямой задачи), а сразу записывать в соответствующих блоках, заключив в скобки (скобки можно и не ставить, но лучше это делать).

Например (см. сегмент организации завершения моделирования и расчета результатов):

```
ADVANCE ((Normal(2,(S1_#Koef),(S2_#Koef))/Q_) ; Розыгрыш врем
SAVEVALUE VerObr,(N$ObrZap/N$KolZap)) ; Расчет вероятности обр
SAVEVALUE TimeNZap,((AC1-X$AC2)/(X$Prog#Koef1)) ; Расчет средне
```

## Модель в AnyLogic

### Постановка задачи

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

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

Как уже отмечалось, сервер представляет собой однофазную систему массового обслуживания разомкнутого типа с ограниченной входной емкостью. Версия 6 AnyLogic при создании подобных простейших моделей предоставляет возможность использования шаблонов моделей. То есть выполнение первых одних и тех же шагов можно перепоручить Мастеру создания модели. Все, что нужно пользователю, это указать, какой метод моделирования будете применять, и выбрать те опции, которые нужны в модели. После этого Мастер автоматически создаст простейшую модель. Далее можно продолжить ее разработку, изменяя свойства объектов модели и добавляя при необходимости другие объекты.

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

## Создание диаграммы процесса

1. Выполните Файл/Создать/Модель  на панели инструментов. Появится диалоговое окно Новая модель (Рис. 1.11).
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, открываясь нажатием кнопки Выбрать...
4. Введите Java пакет: server.
5. Будет создана следующая папка:  
G:\BOEV\Server\Server.adb



Рис. 1.11. Диалоговое окно Новая модель

4. Щелкните кнопку Далее. Откроется вторая страница Мастера создания модели (Рис. 1.12).
5. Здесь будет предложено выбрать шаблон, на базе которого будете разрабатывать модель. Поскольку мы хотим создать новую дискретно-событийную модель не "с нуля", установите флажок Использовать шаблон модели и выберите Дискретно-

событийное моделирование в расположеннном ниже списке.

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

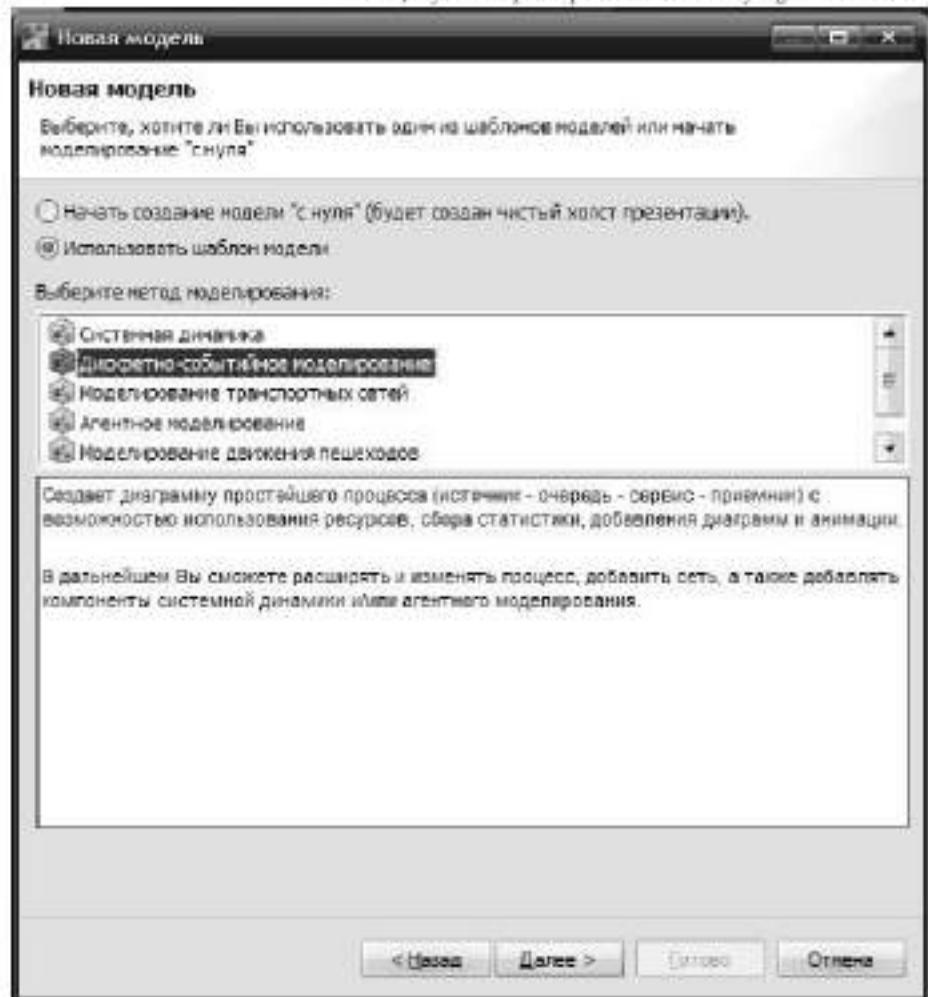


Рис. 1.12. Вторая страница диалогового окна Новая модель

7. Вы создали новую модель. Далее познакомимся с пользовательским интерфейсом AnyLogic ([Рис. 1.13](#)).
8. В левой части рабочей области находится панель Проект. Панель Проект обеспечивает навигацию по элементам моделей, открытых в текущий момент времени. Поскольку модель организована иерархически, то она отображается в виде дерева. Сама модель образует верхний уровень дерева. Эксперименты, классы активных объектов и Java классы образуют следующий уровень. Элементы, входящие в состав активных объектов, вложены в соответствующую подветвь дерева класса активного

объекта и т. д.

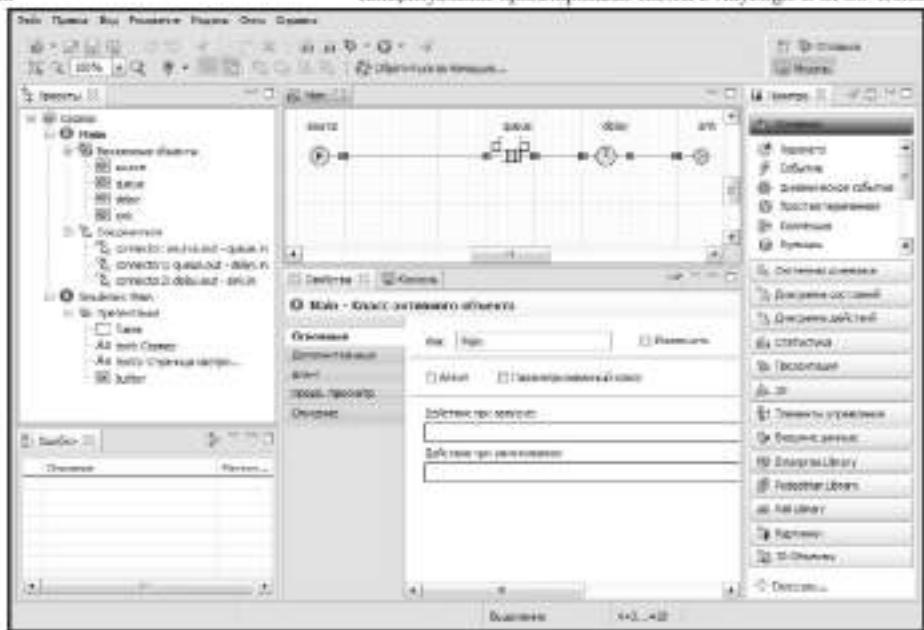


Рис. 1.13. Интерфейс AnyLogic

9. В правой рабочей области будет отображаться панель Палитра, а внизу в средней части интерфейса - панель Свойства. Панель Палитра содержит разделенные по категориям элементы, которые могут быть добавлены на диаграмму класса активного объекта или эксперимента. Панель Свойства используется для просмотра и изменения свойств выбранного в данный момент элемента (или элементов) модели.
10. В центре рабочей области AnyLogic находится графический редактор диаграммы класса активного объекта Main.
11. В левой нижней части расположена панель Ошибки. Она отображает ошибки в модели и помогает их локализовать.

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

**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 блоков (в AnyLogic это блоки библиотеки Enterprise Library), задающих последовательность операций, которые будут производиться над проходящими по диаграмме процесса заявками.

Обратите внимание на диаграмму класса Main. Вы увидите, что диаграмма нашего простейшего процесса ([Рис. 1.14](#)) была автоматически создана Мастером создания модели, поскольку такая модель является ничем иным, как простейшей системой массового обслуживания, наиболее часто используемой в качестве отправной точки создания дискретно-событийных моделей. Поэтому она и выбрана в качестве базового шаблона при разработке дискретно-событийных моделей.

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

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

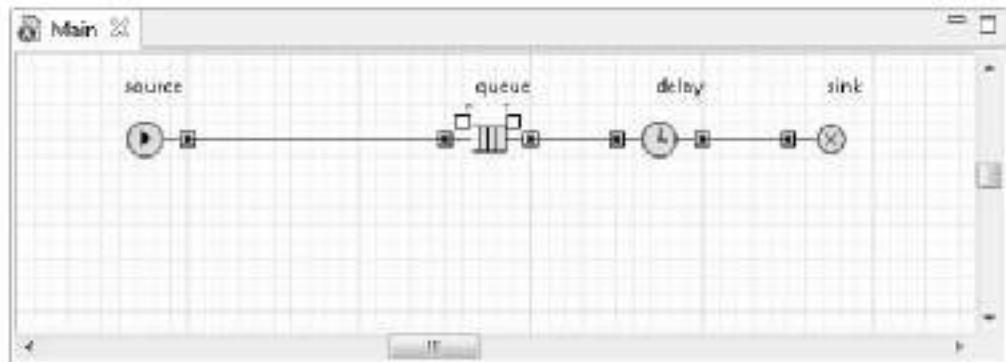


Рис. 1.14. Диаграмма простейшей системы массового обслуживания

## 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, блоки в диаграмме процесса обычно располагаются цепочкой слева направо, представляя собой последовательную очередь 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.

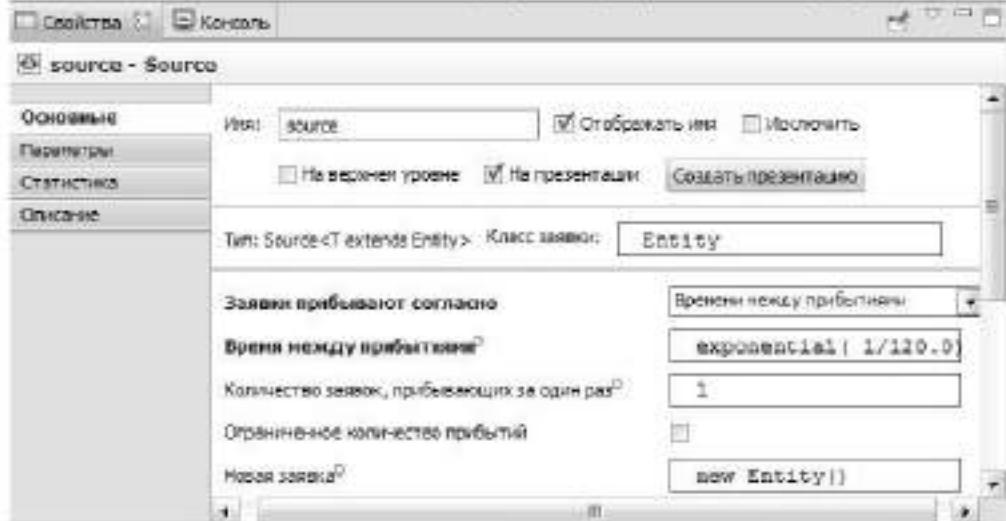


Рис. 1.15. Свойства объекта source

В нашем случае объект (элемент) 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.

Выделите объект source. В выпадающем списке Заявки прибывают согласно укажите, что запросы поступают согласно Времени между прибытиями (Рис. 1.15). 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 моделировать очередь запросов, ждущих освобождения сервера.

Измените свойства объекта `queue` ([Рис. 1.16](#)).

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.

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

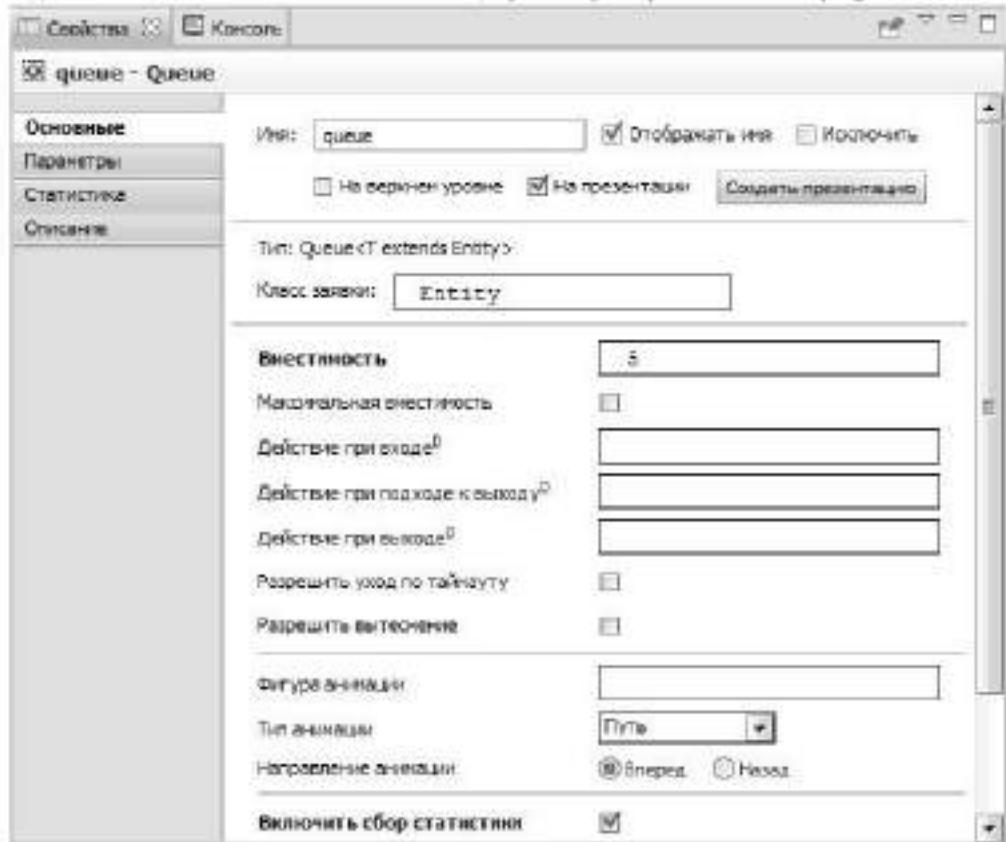


Рис. 1.16. Свойства объекта queue

Измените свойства объекта delay (Рис. 1.17).

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.

Последним в диаграмме нашей дискретно-событийной модели находится объект sink. Этот объект уничтожает поступившие заявки.

Обычно он используется в качестве конечной точки потока заявок (и диаграммы процесса соответственно). В нашем случае он выводит из модели обработанные сервером запросы.

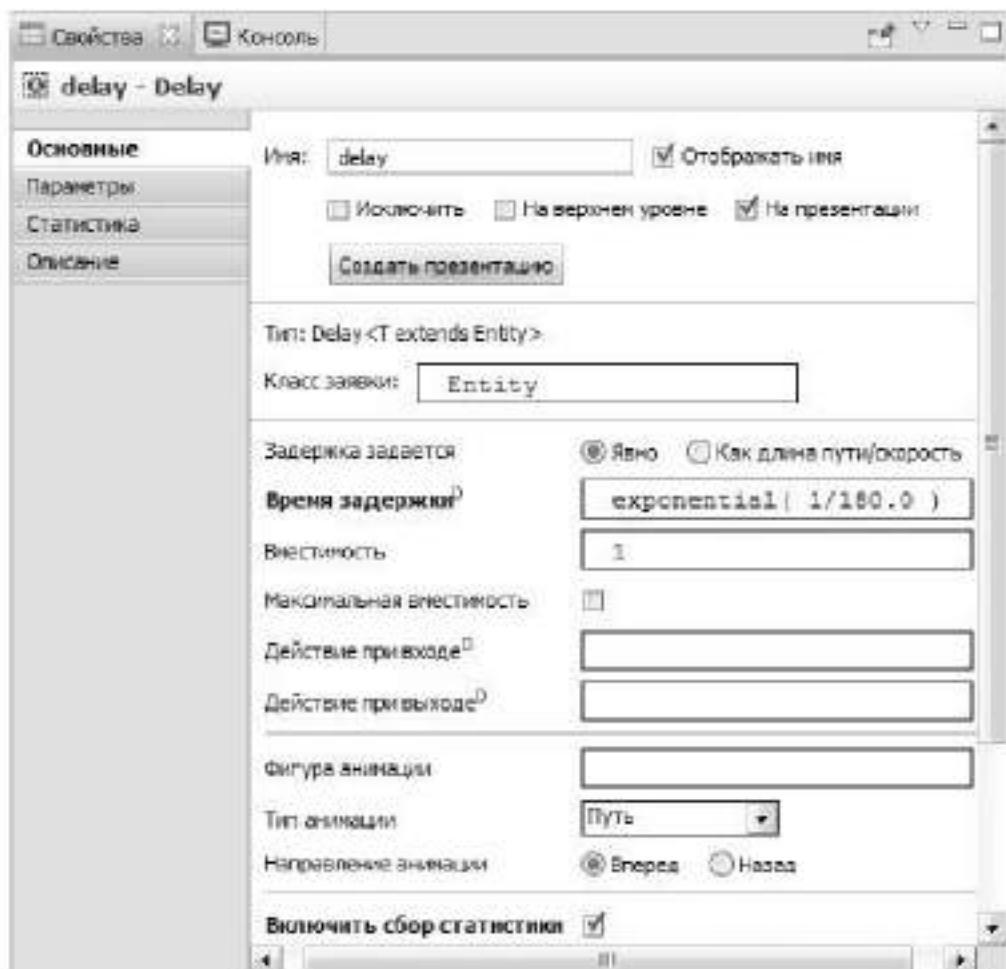


Рис. 1.17. Свойства объекта delay

### 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.



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. Тем не менее, оставьте время остановки модели не заданным.

1. В панели Проект выделите эксперимент *Simulation:Main*.
2. На странице Основные панели Свойства выберите опцию Фиксированное начальное число (воспроизводимые прогоны).
3. В поле Начальное число: установите 9 (такое же, как и в GPSS-программе).
4. На странице Модельное время панели Свойства оставьте все по умолчанию.

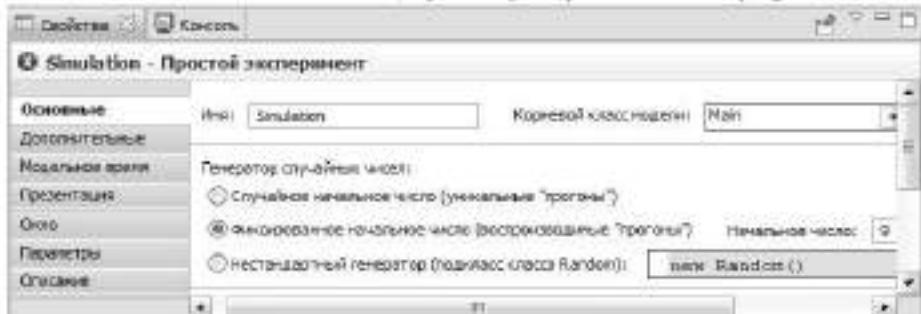


Рис. 1.18. Страница Основные панели Свойства



Рис. 1.19. Установка единиц модельного времени

5. В панели Проект, выделите Server (Рис. 1.19).
6. На странице Основные панели Свойства из выпадающего списка Единицы модельного времени: выберите секунды.

### 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.

После исправления ошибок и построения модели, запустите её:

- Щелкните мышью кнопку панели инструментов  Запустить (или нажмите F5) и выберите из открывшегося списка эксперимент, который вы хотите запустить. Эксперимент этой модели будет называться *Server/Simulation* (Рис. 1.20).
- В дальнейшем нажатием кнопки Запустить (или кнопки F5) будет запускаться тот эксперимент, который запускался вами в последний раз. Чтобы выбрать другой эксперимент, вам 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).

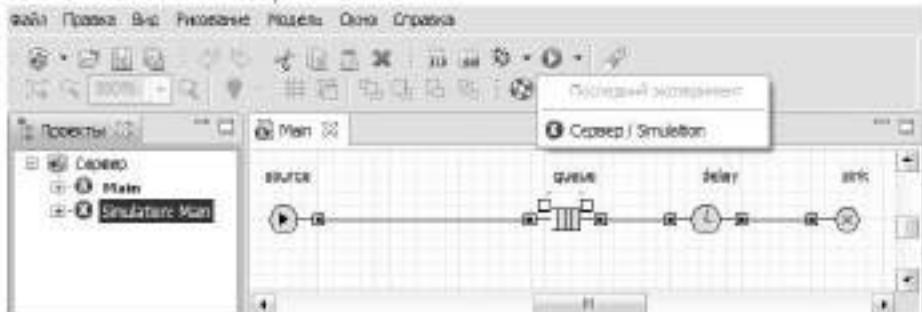


Рис. 1.20. Выбор эксперимента

- После запуска модели вы увидите окно презентации этой модели (Рис. 1.21). В нем будет отображена презентация запущенного эксперимента. AnyLogic автоматически помещает на презентацию каждого простого эксперимента заголовок и кнопку, позволяющую запустить модель и перейти на презентацию, нарисованную вами для главного класса активного объекта этого эксперимента (*Main*).
- Щелкните данную кнопку. Этим щелчком вы запустите модель и перейдете к презентации корневого класса активного объекта запущенного эксперимента. Для каждой модели, созданной в Enterprise Library, автоматически создается блок-схема с наглядной визуализацией процесса, с помощью которой вы можете изучать

текущее состояние модели, например, длину очереди, количество обработанных запросов и так далее (Рис. 1.22).

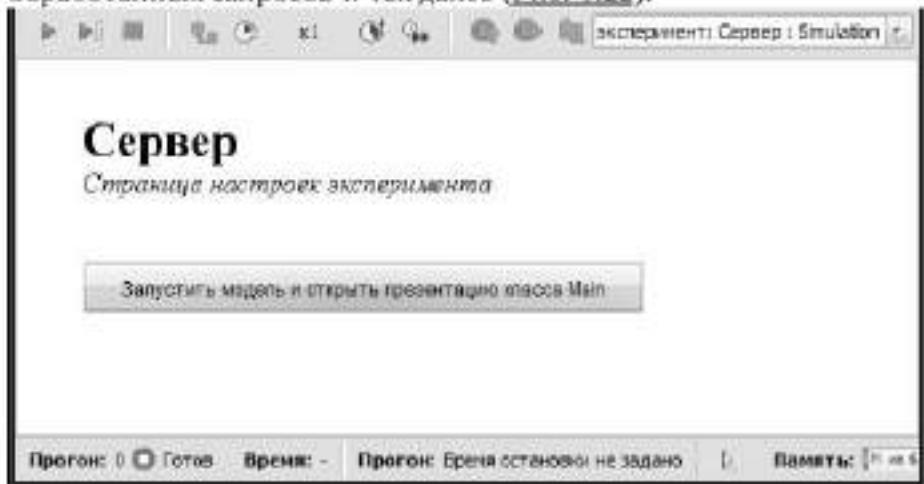


Рис. 1.21. Окно презентации модели

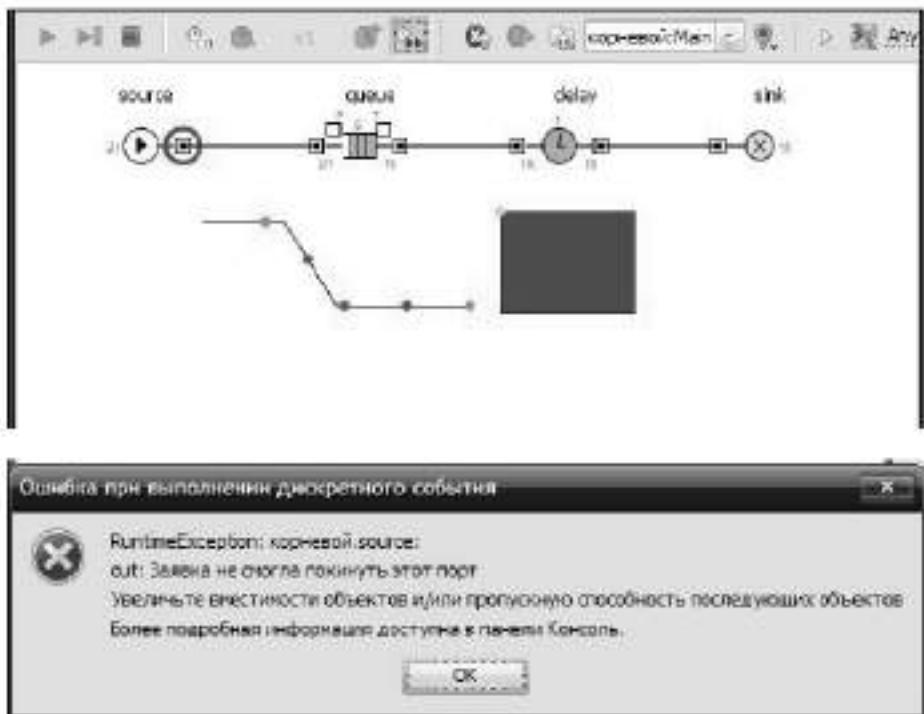


Рис. 1.22. Моделирование остановилось с ошибкой

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. 1.22](#)). Ошибка означает, что запрос не может войти в блок queue, так как его ёмкость, равная 5, заполнена.
6. Нажмите OK. Далее измените свойства объекта queue, т. е. увеличьте длину очереди (см. [Рис. 1.16](#)). 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. Снова запустите модель.
7. Вы можете изменить скорость выполнения модели с помощью кнопок Замедлить и Ускорить панели инструментов.
8. Вы можете следить за состоянием любого блока диаграммы процесса во время выполнения модели с помощью окна инспектора этого объекта. Чтобы открыть окно инспектора, щелкните мышью по значку нужного блока. Окно инспектора, подведя курсор, можно перемещать в нужное вам место. Также, подведя курсор к правому нижнему углу окна инспектора, можно изменять его размеры.

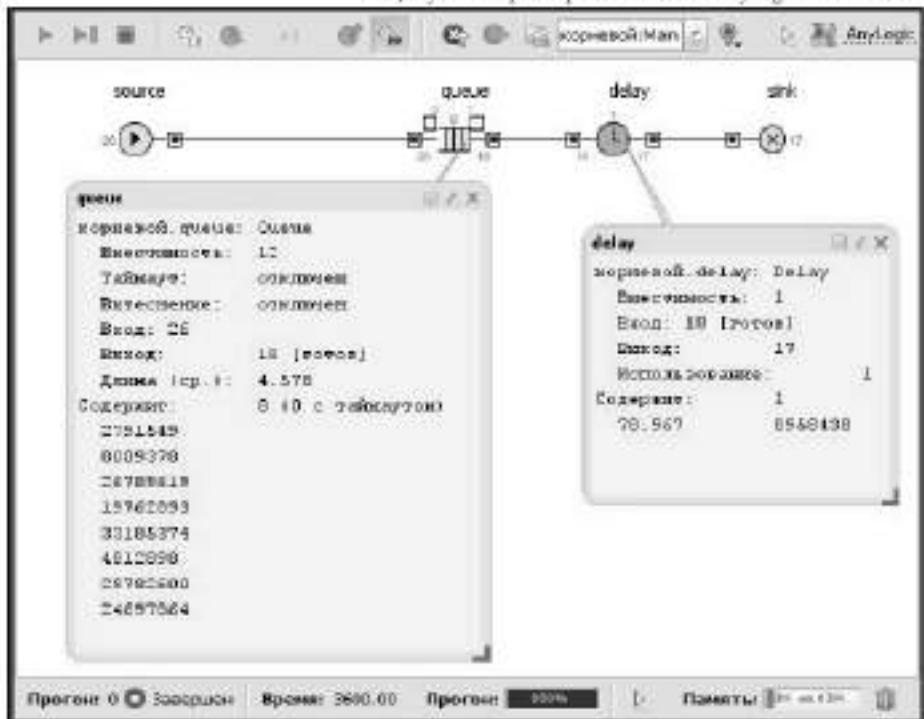


Рис. 1.23. Окно инспекта

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, средняя длина очереди и т. д. Такая же информация содержится в инспекте и для объекта `delay` (Рис. 1.23).
10. Когда вы захотите остановить выполнение модели, щелкните мышью кнопку Прекратить выполнение ■ панели управления окна презентации.
11. Для предотвращения остановок модели по ранее указанной ошибке - 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 не выполнили условий, указанных в постановке задачи, поэтому к выполнению их вернемся позже.

## Создание анимации модели

Можно было наблюдать, анализировать и интерпретировать работу запущенной модели с помощью визуализированной диаграммы процесса (см. [Рис. 1.22](#), [Рис. 1.23](#)). Однако удобнее иметь более наглядную визуализацию с помощью анимации. В этом примере мы хотим создать визуализированный процесс поступления запросов на сервер и обработки запросов сервером.

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

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

Нарисуйте прямоугольник, который будет обозначать на анимации сервер.

1. Вначале откройте закладку Презентация панели Палитра ([Рис. 1.24](#)). Чтобы открыть какую-либо закладку панели Палитра (в дальнейшем палитра), нужно щелкнуть мышью по заголовку этой палитры.

Замечание. Можно открыть сразу все палитры, щелкнув мышью кнопку Развернуть все палитры.

2. Палитра Презентация ([Рис. 1.25](#)) содержит в качестве элементов примитивные фигуры, используемые для рисования презентаций моделей. Это линия, ломаная, кривая, прямоугольник, овал, дуга, точка, изображение, кнопка, флагок и др. Имеются также элементы управления, с помощью которых вы можете сделать ваши презентации интерактивными.
3. Выделите щелчком мыши элемент Прямоугольник на палитре Презентация и перетащите его на диаграмму класса активного

объекта. Поместите элемент Прямоугольник так, как показано на Рис. 1.26.

4. Давайте сделаем так, что цвет этого прямоугольника будет меняться в зависимости от того, обрабатывает ли сервер в данный момент времени запрос или нет.
5. Для этого выделите нарисованную нами фигуру на диаграмме и перейдите на страницу Динамические панели свойств (Рис. 1.27). Здесь вы увидите список полей, в которых задаются значения динамических свойств фигуры.

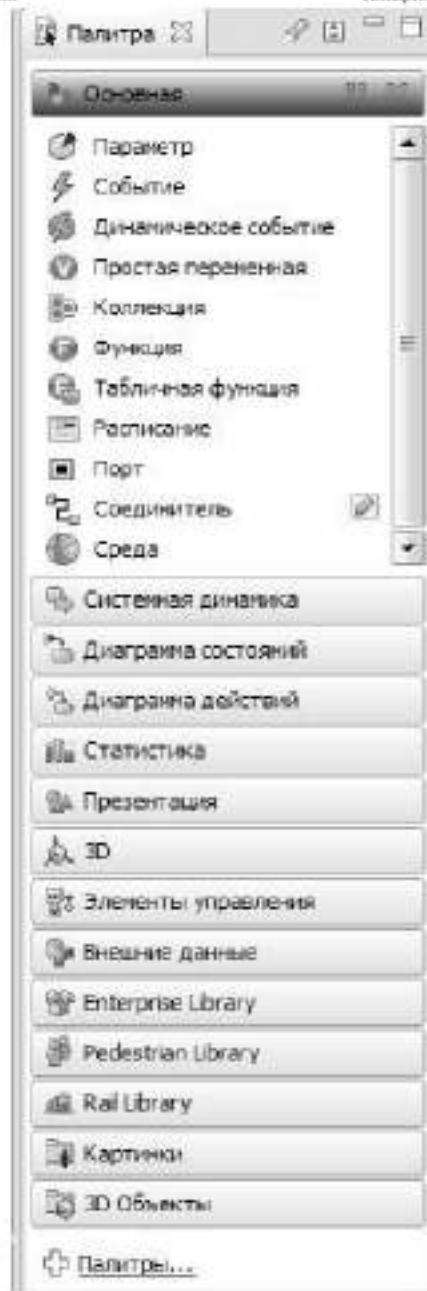


Рис. 1.24. Панель Палитра

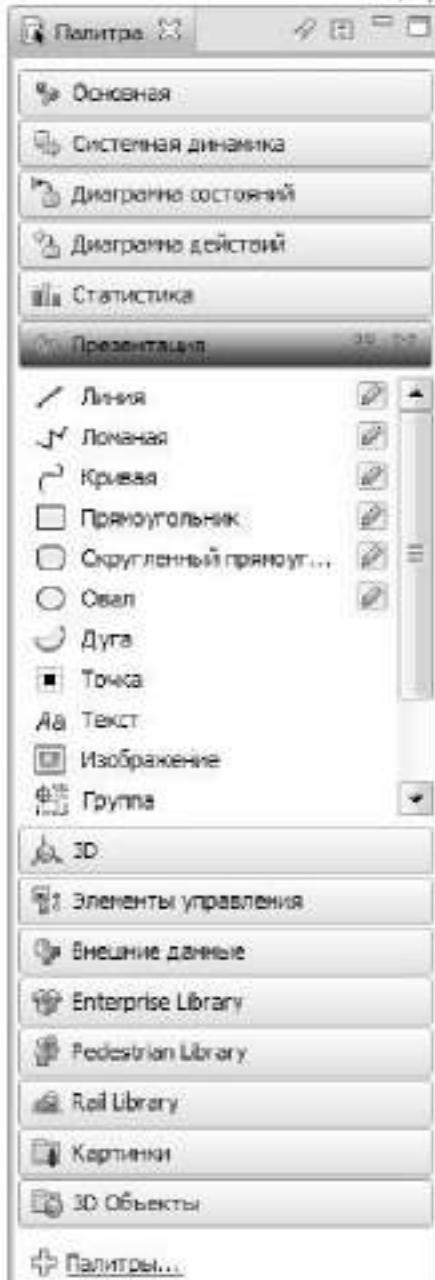


Рис. 1.25. Палитра Презентация

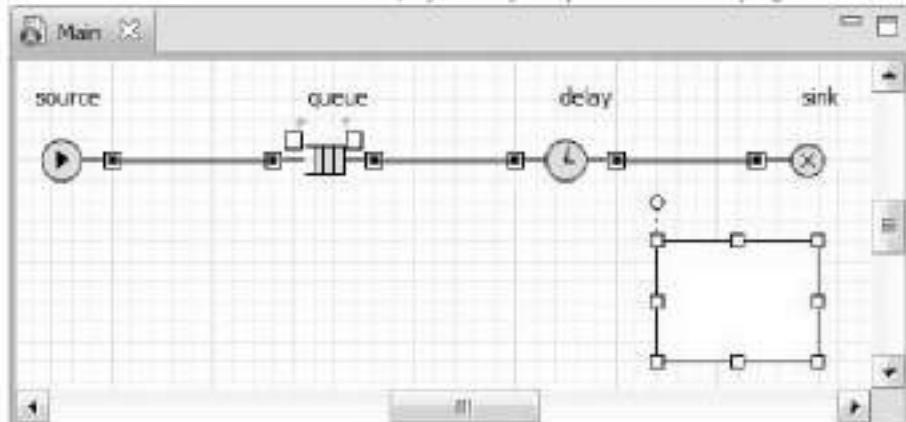
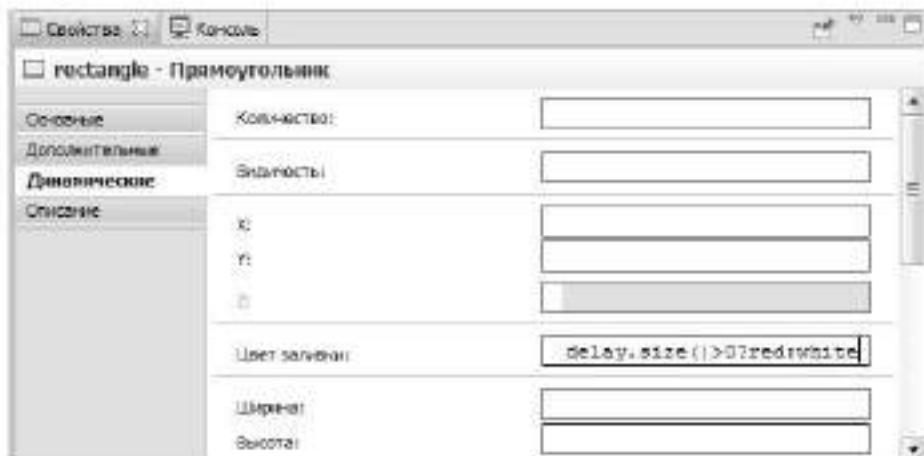


Рис. 1.26. Элемент Прямоугольник на диаграмме

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

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

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



### Рис. 1.27. Страница Динамические панели свойств

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

6. Нарисуйте ломаную, которая будет обозначать на анимации очередь к серверу ([Рис. 1.28](#)). Чтобы нарисовать ломаную, сделайте двойной щелчок мышью по элементу Ломаная в палитре (при этом его значок должен поменяться на этот: ). Теперь вы можете рисовать ломаную точка за точкой, последовательно щелкнув мышью в тех точках диаграммы, куда вы хотите поместить вершины ломаной. Чтобы завершить рисование, добавьте последнюю точку ломаной двойным щелчком мыши.

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

7. Теперь мы должны задать созданные анимационные объекты в качестве анимационных фигур блоков диаграммы нашего процесса. Задайте ломаную линию в качестве фигуры анимации очереди. На странице свойств объекта `queue`, введите `polyline` в поле **Фигура анимации** ([Рис. 1.29](#)).

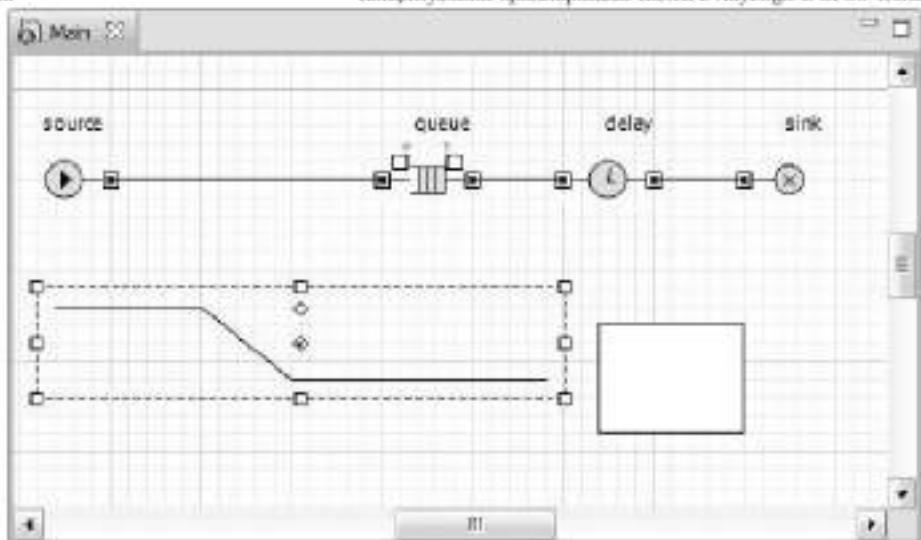


Рис. 1.28. Ломаная

8. Задайте прямоугольник в качестве фигуры анимации сервера. Введите в поле Фигура анимации имя нашего прямоугольника: `rectangle` (Рис. 1.30). Выберите из выпадающего списка Тип анимации Одиночная. Большинство объектов Enterprise Library поддерживает несколько анимационных стилей. Например, очередь может отображаться в виде линии, упорядоченного или неупорядоченного набора элементов. В нашем случае, если сервер будет занят, то мы будем показывать в фигуре сервера обрабатываемого в нем запроса, а поскольку единовременно наш сервер не обрабатывает больше одного запроса, то мы и выбираем тип анимации Одиночная.

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

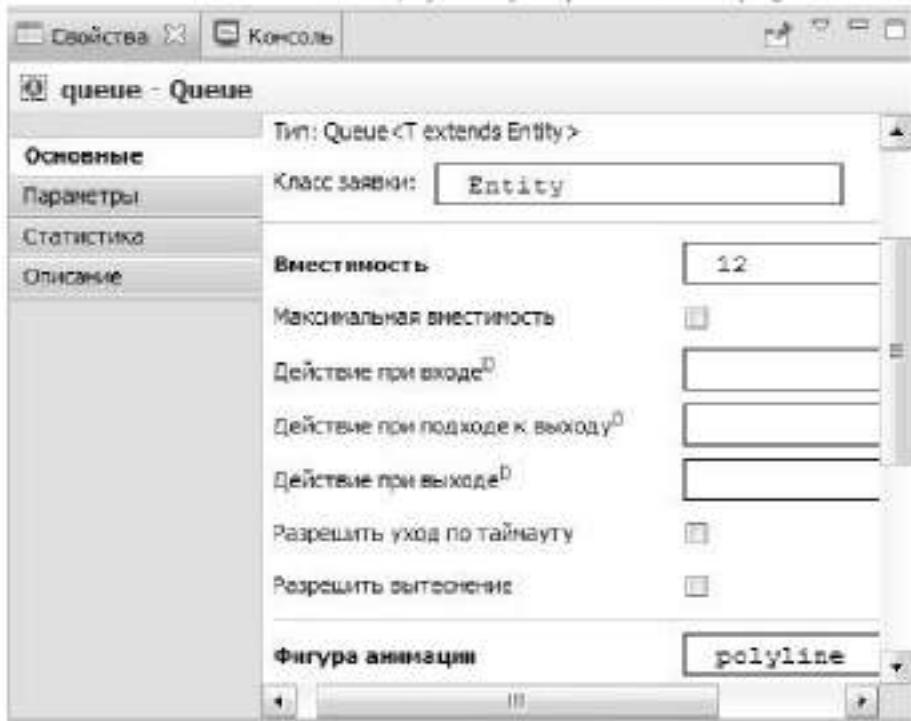


Рис. 1.29. Задание ломаной в качестве фигуры анимации очереди

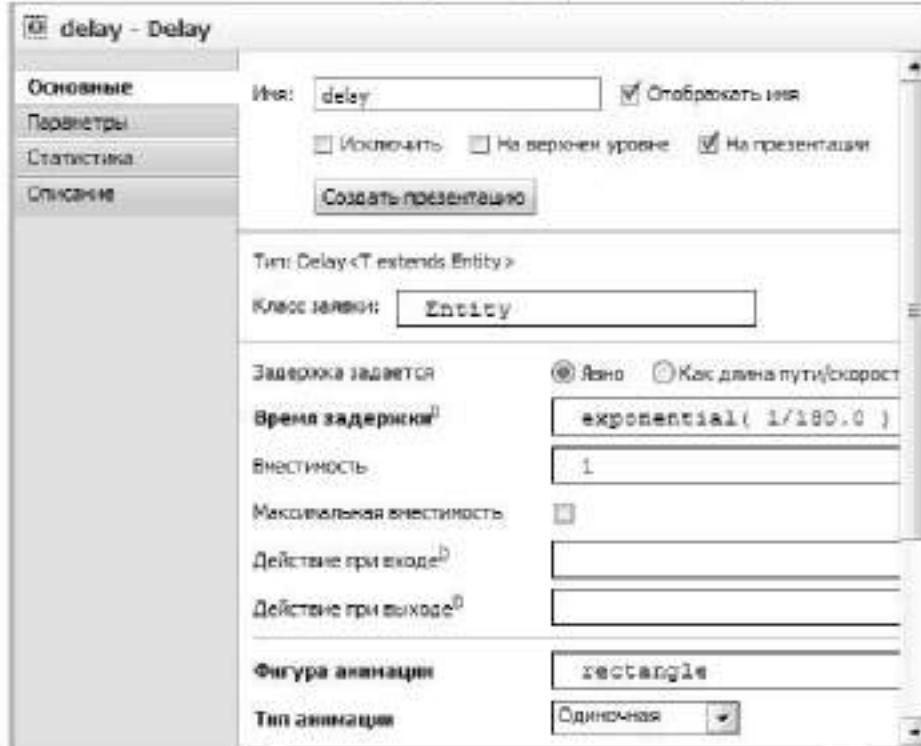


Рис. 1.30. Задание прямоугольника в качестве фигуры анимации сервера

9. Запустите модель. Вы увидите, что у модели теперь есть простейшая анимация - сервер и очередь запросов к нему (Рис. 1.31). Цвет фигуры сервера будет меняться в зависимости от того, обрабатывается ли запрос в данный момент времени или нет.

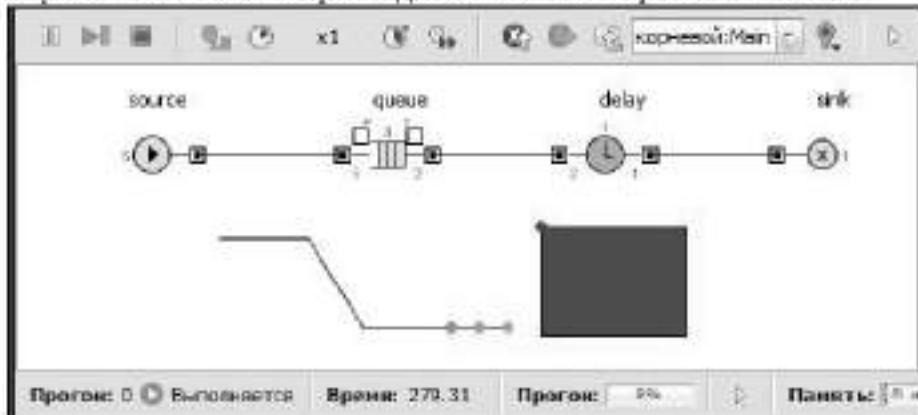


Рис. 1.31. Анимация модели

## 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.
3. Перейдите на страницу Основные панели Свойства. Щелкните мышью кнопку Добавить элемент данных. После щелчка появится секция свойств того элемента данных (`chart` - Столбиковая диаграмма), который будет отображаться на этой диаграмме (Рис. 1.33).

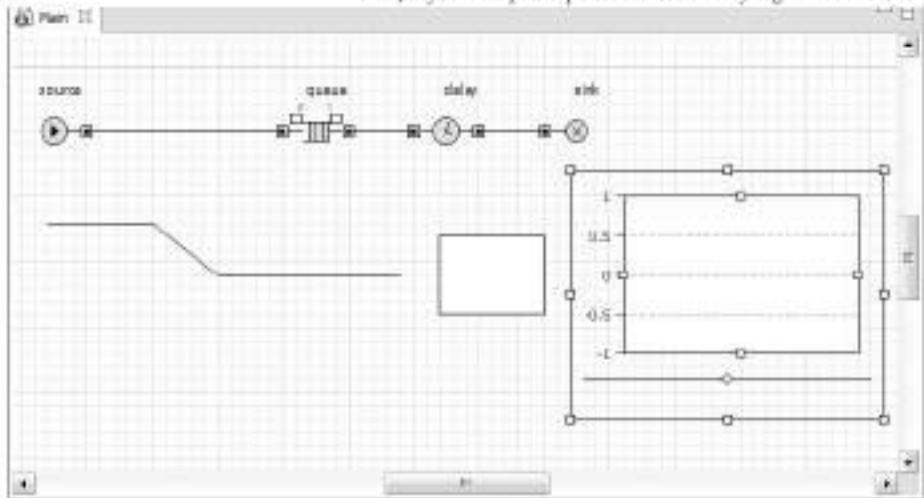


Рис. 1.32. Элемент Столбиковая диаграмма на диаграмме класса

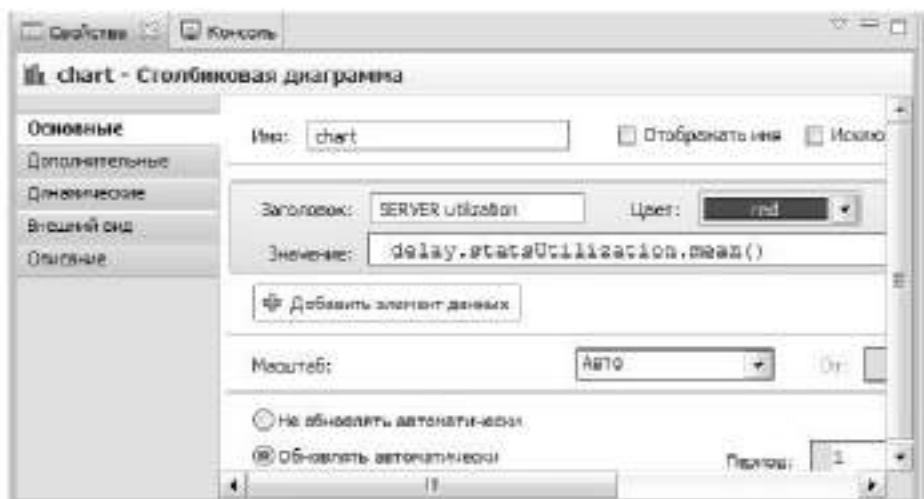


Рис. 1.33. Страница Основные панели Свойства

4. Измените Заголовок на SERVER utilization.
5. Введите `delay.statsUtilization.mean()` в поле Значение. Здесь `delay` - это имя нашего объекта `delay`. У каждого объекта `delay` есть встроенный набор данных `statsUtilization`, занимающийся сбором статистики использования этого объекта. Функция `mean()` возвращает среднее из всех измеренных этим набором данных значений. Вы

можете использовать и другие методы сбора статистики, такие, как `min()` или `max()`. Полный список методов можно найти на странице документации этого класса набора данных: `StatisticsContinuous` (на английском языке).

- Перейдите на страницу Внешний вид (Рис. 1.34).

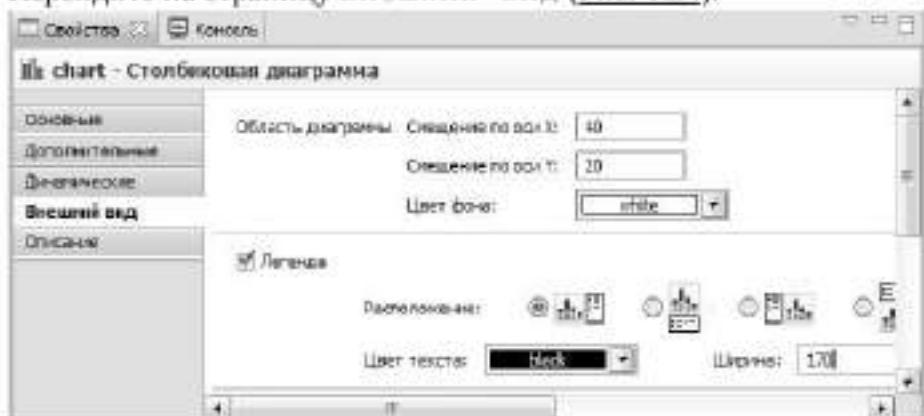


Рис. 1.34. Страница Внешний вид панели Свойства

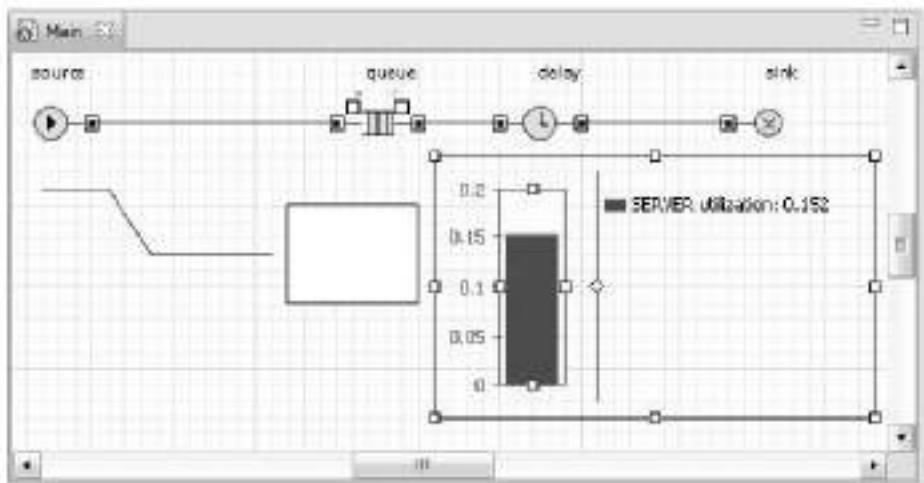


Рис. 1.35. Изменённый вид столбиковой диаграммы

- Выберите первую опцию из набора кнопок Расположение, чтобы изменить расположение легенды относительно диаграммы (мы хотим, чтобы она отображалась справа). Размер диаграммы в графическом редакторе измените так, чтобы она приняла вид, показанный на Рис. 1.35.

8. Аналогичным образом добавьте еще одну столбиковой диаграмму для отображения средней длины очереди. Заголовок и Значение измените так, как показано на Рис. 1.36.

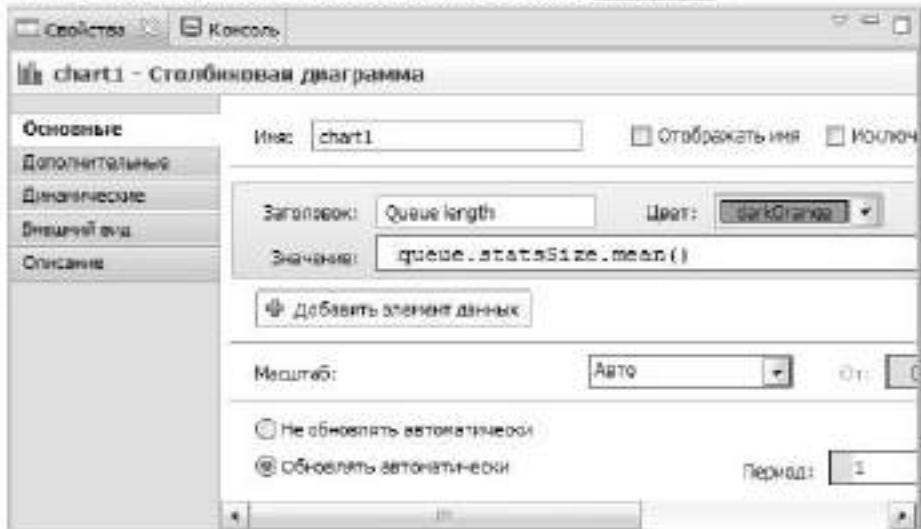


Рис. 1.36. Страница Основные панели Свойства

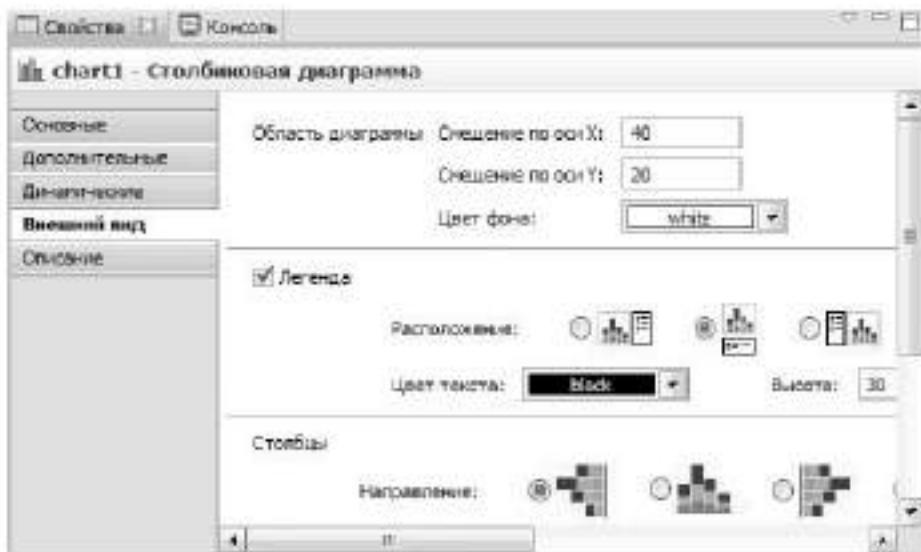


Рис. 1.37. Страница Внешний вид панели Свойства

9. Здесь `queuee` - 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.

- Перейдите на страницу Внешний вид панели Свойства и выберите в секции свойств Направление первую опцию ([Рис. 1.37](#)), чтобы столбцы во второй столбиковой диаграмме, расположенной горизонтально, росли влево ([Рис. 1.38](#)).

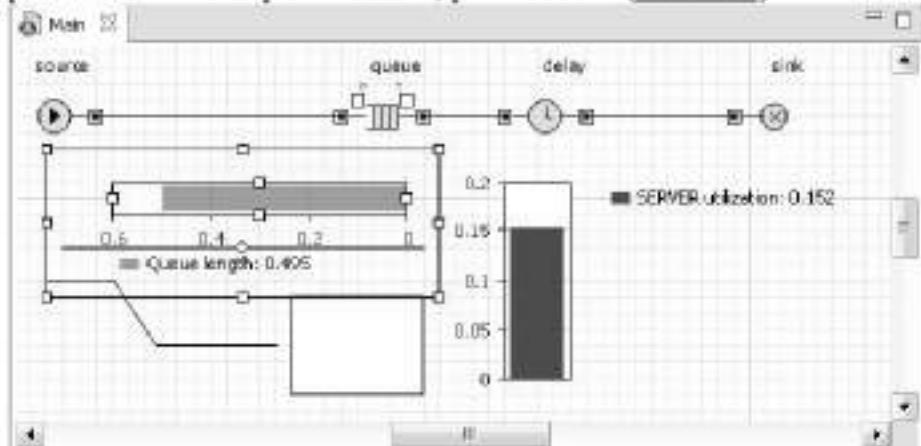


Рис. 1.38. Добавлена вторая столбиковая диаграмма

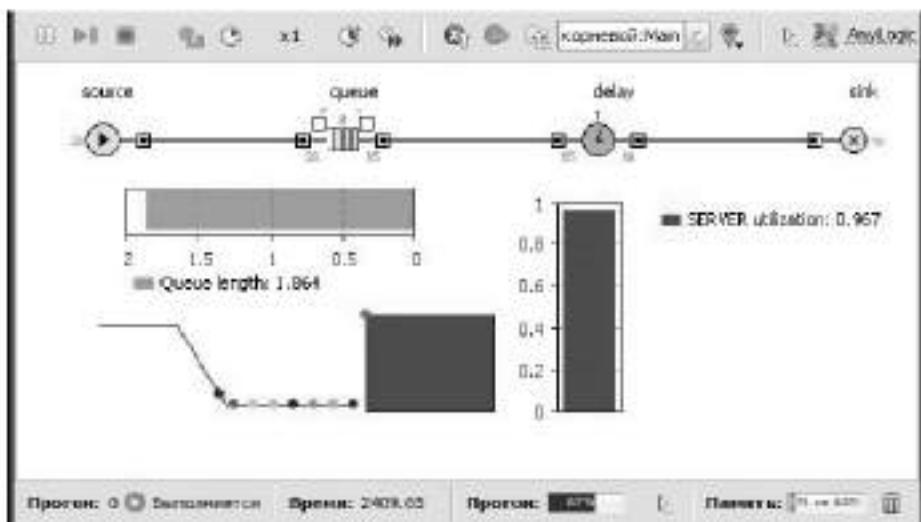


Рис. 1.39. Наблюдение за моделью с двумя столбиковыми диаграммами

11. Запустите модель с двумя столбиковыми диаграммами и понаблюдайте за ее работой (Рис. 1.39).

## Уточнение модели согласно ёмкости входного буфера

На Рис. 1.39 (снимок сделан, в случайный момент времени, равный 2409,65 единицам) видно, что длина очереди равна 8 запросам при установленной максимальной длине 12. 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.

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 принятия 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 странице Основные панели Свойства измените Вместимость с 15 на 5 запросов.
2. На этой же странице установите Разрешить вытеснение.
3. Для уничтожения потерянных запросов вследствие полного заполнения накопителя необходимо добавить второй объект `sink`. Откройте в Палитре библиотеку Enterprise Library и перетащите блок `sink` на диаграмму ([Рис. 1.40](#)).

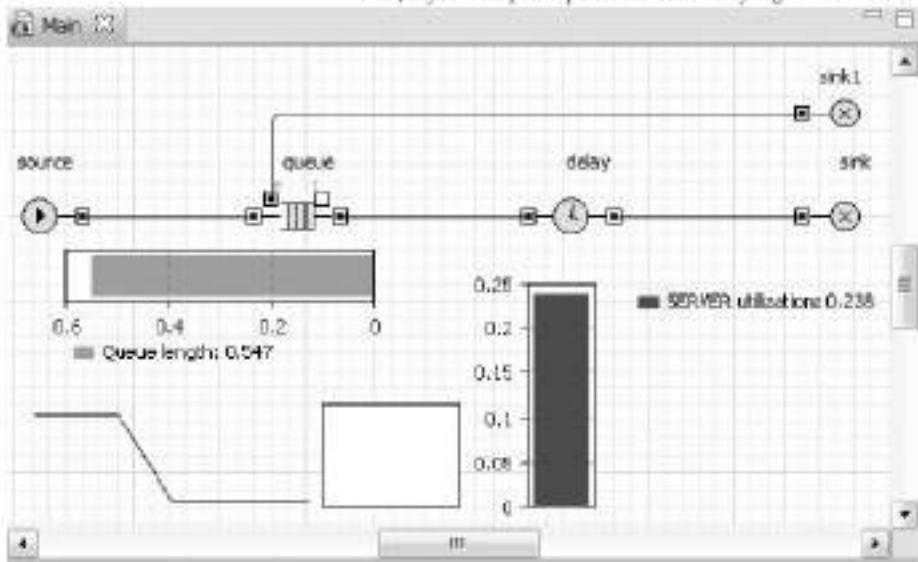


Рис. 1.40. Уточненная модель

4. 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 щелкните в тех местах диаграммы, где вы хотите поместить точки изгиба соединителя. Завершите процесс соединения двойным щелчком мыши по второму порту.
5. После двойного щелчка по второму порту вы увидите, что появится соединитель. Если выделить его мышью, то 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.
6. Запустите уточненную модель и понаблюдайте за ее работой. Сравните Рис. 1.41 с Рис. 1.22. На Рис. 1.41 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.

Приступим к реализации этих требований. Но предварительно сохраним модель с именами Сервер Прямая задача и Сервер Обратная задача. К последней мы вернёмся в главе 10.

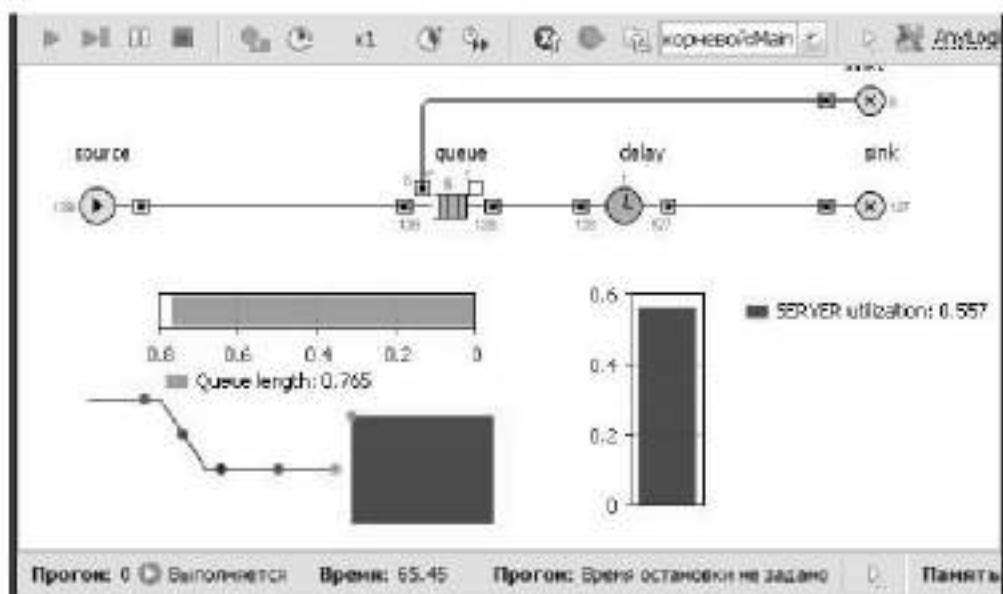


Рис. 1.41. Работа модели согласно ёмкости входного буфера

## Сбор статистики по показателям обработки запросов

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

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

сервером.

Математическое ожидание или среднее время обработки одного запроса определяется как отношение суммарного времени обработки п. запросов к их количеству, т. е. к п. 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 (выхода в блок `sink`). Тогда

$$\text{time_obrabi} = \text{time_vixod} - \text{time_vxod}$$

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 (выходе в блок `sink`). Для этого также введем дополнительные поля: `col_vxod` - количество поступивших всего запросов, `col_vixod` - количество обработанных сервером запросов. Тогда

$$\text{ver_obrabi} = \text{col_vixod} / \text{col_vxod}$$

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

### Создание нестандартного класса заявок

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

Создайте класс заявок `Inquiry`.

1. В панели Проект щелкните правой кнопкой мыши элемент модели верхнего уровня дерева и выберите Создать/Java класс из контекстного меню.
2. Появится диалоговое окно Новый Java класс (Рис. 1.42). В

поле Имя : введите имя нового класса: Inquiry.

3. В поле Базовый класс: выберите из выпадающего списка Entity в качестве базового класса. Щелкните кнопку Далее.

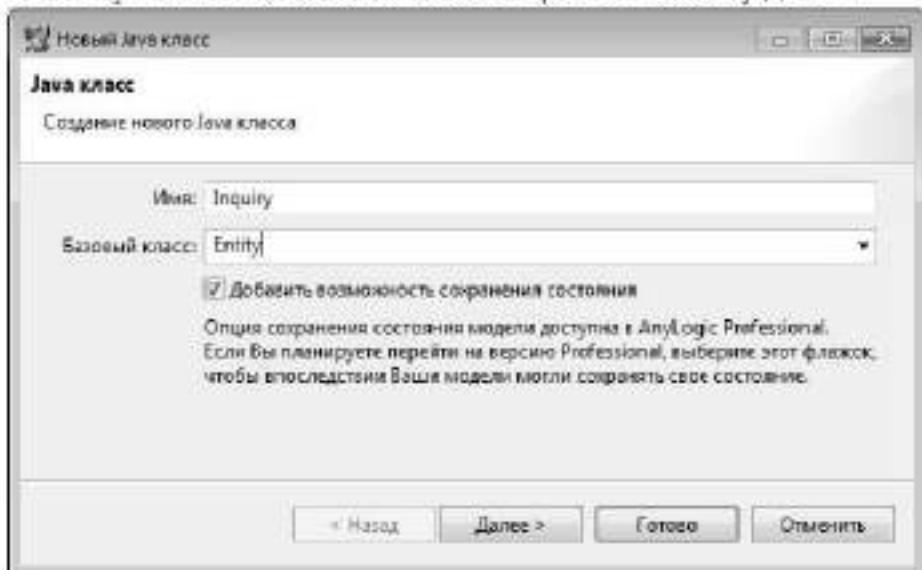


Рис. 1.42. Диалоговое окно Мастера создания Новый Java класс

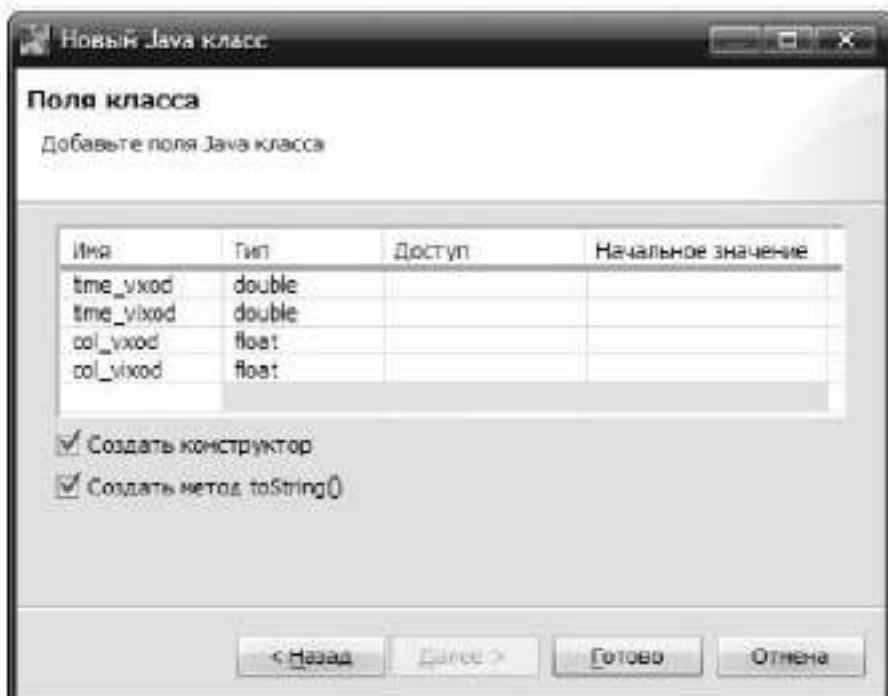
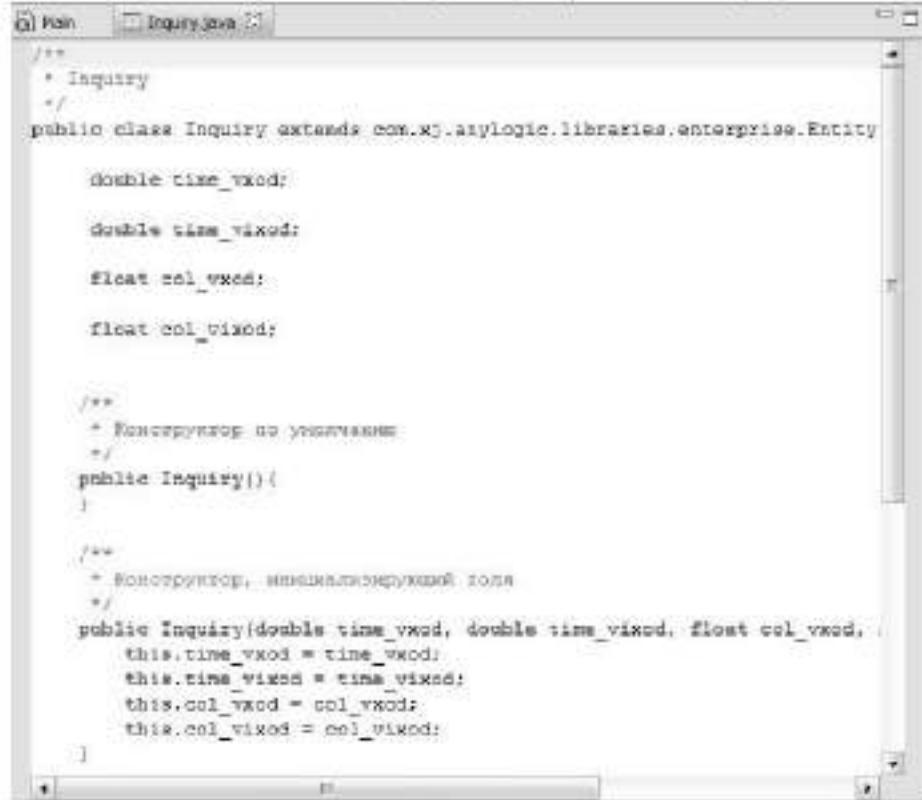


Рис. 1.43. Вторая страница Мастера создания Новый Java класс

4. Появится вторая страница Мастера создания Java класса ([Рис. 1.43](#)). Добавьте поля Java класса: `time_vxod` типа `double`, `time_vixod` типа `double`, `col_vxod` типа `float`, `col_vixod` типа `float`. Типы полей выбираются из выпадающего списка. Начальные значения всех параметров, поскольку не указаны, по умолчанию будут установлены равными нулю.
5. Оставьте выбранными флагки Создать конструктор и Создать метод `toString ()`. Тогда у класса будут созданы сразу два конструктора: один, по умолчанию, без параметров, и второй, с параметрами, инициализирующими поля класса. Эти конструкторы используются объектами, создающими новые заявки, такие, как `Source`.
6. Щелкните кнопку Готово. Вы увидите редактор кода, в котором будет показан автоматически созданный код вашего Java класса ([Рис. 1.44](#)). Изучите этот код и выясните, как можно самостоятельно добавлять в класс заявки новые поля и методы. Закройте редактор, щелкнув крестик в закладке рядом с его названием.



```

(Q) Main Inquiry.java  Inquiry.java [1]
//+
* Inquiry
*/
public class Inquiry extends com.xj.anylogic.libraries.enterprise.Entity

    double time_vxod;
    double time_vixod;
    float col_vxod;
    float col_vixod;

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

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

```

Рис. 1.44. Окно редактора кода созданного нестандартного класса

### Добавление элементов статистики

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

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

2. Задайте свойства элемента ([Рис. 1.45](#)):

- измените Имя: на time\_obrabotki;
- сделайте Кол-во интервалов: равным 50;
- задайте Нач. размер интервала: 0.01.

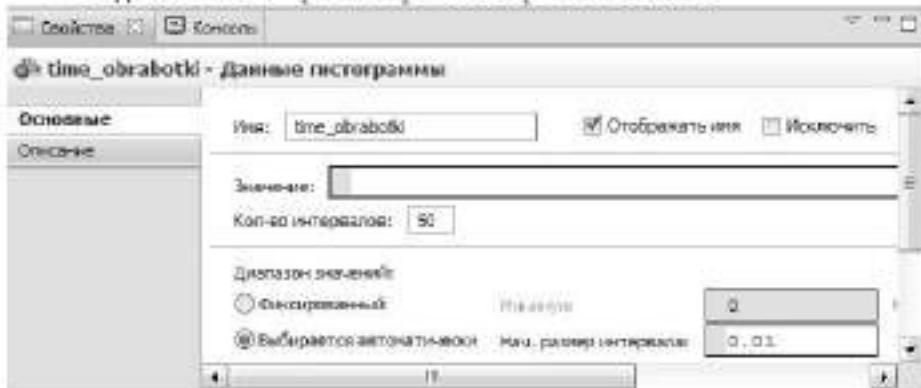


Рис. 1.45. Элемент сбора статистики о времени обработки запросов

Добавьте еще элемент сбора статистики для определения вероятности обработанных запросов.

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

2. Задайте свойства элемента ([Рис. 1.46](#), [Рис. 1.47](#)):

- измените Имя: на ver\_obrabotki;
- сделайте Кол-во интервалов: равным 50;
- задайте Нач. размер интервала: 0.01.

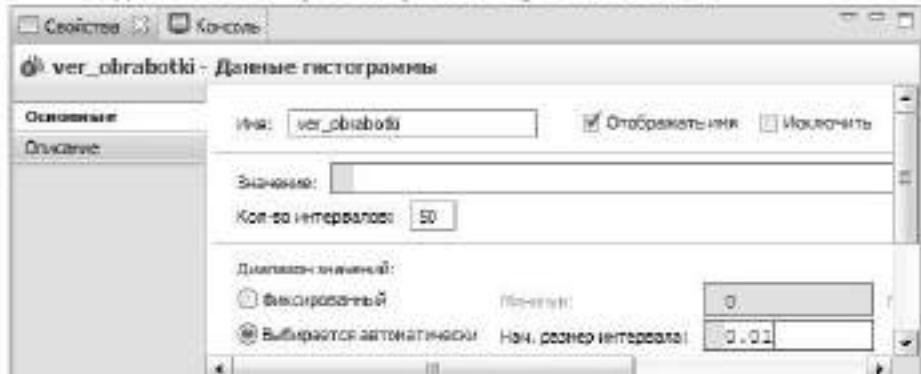


Рис. 1.46. Элемент сбора статистики о вероятности обработки

## запросов

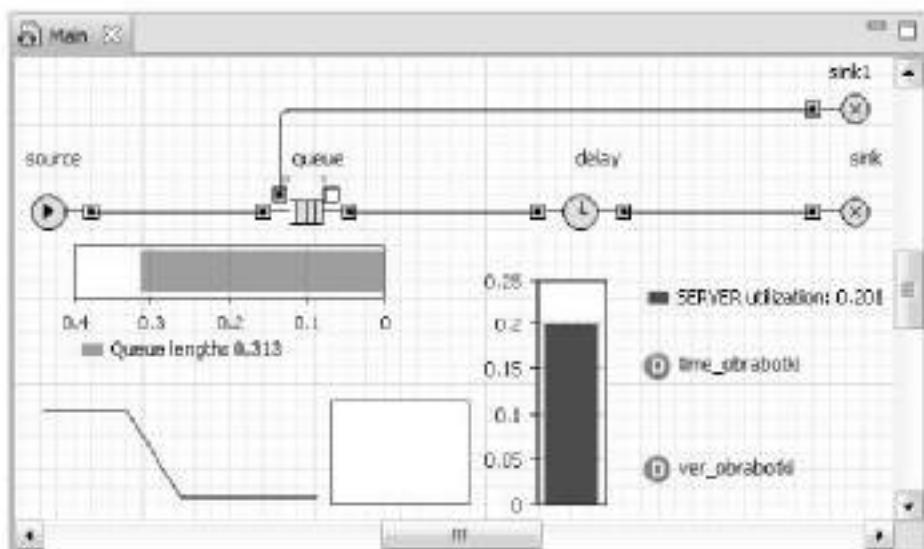


Рис. 1.47. Диаграмма после добавления элементов сбора статистики

## 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 класса заявки в коде динамических параметров объектов потоковой диаграммы, вам нужно указать имя нестандартного класса заявки в качестве Класса заявки этого объекта. В нашей потоковой диаграмме с учетом блока *source* всего пять объектов. Измените их свойства.

**1. Измените свойства объекта *source* (Рис. 1.48):**

- введите *Inquiry* в поле Класс заявки:. Это позволит

напрямую обращаться к полям класса заявки Inquiry в коде динамических параметров этого объекта;

- введите new Inquiry() в поле Новая заявка. Теперь этот объект будет создавать заявки нашего типа Inquiry;
- введите entity.time\_vxod-time(); в поле Действие при выходе. Код будет сохранять время создания заявки-запроса в переменной time\_vxod нашего класса заявки Inquiry.

Функция time() возвращает текущее значение модельного времени.

## 2. Измените свойства объекта queue:

- введите Inquiry в поле Класс заявки:.

## 3. Измените свойства объекта delay:

- введите Inquiry в поле Класс заявки:.

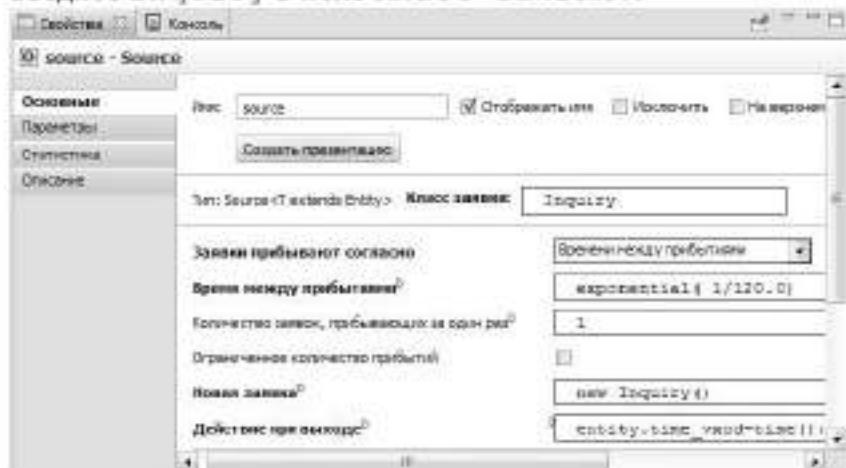


Рис. 1.48. Объект source с измененными свойствами

## 4. Измените свойства объекта sink1:

- введите Inquiry в поле Класс заявки:.

## 5. Измените свойства объекта sink (Рис. 1.49):

- введите Inquiry в поле Класс заявки:;
- введите в поле Действие при входе:

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

Этот код добавляет время обработки одного запроса в объект сбора данных гистограммы `time_обработки`. 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_обработки.add(entity.col_vixod/entity.col_vxod);
```

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

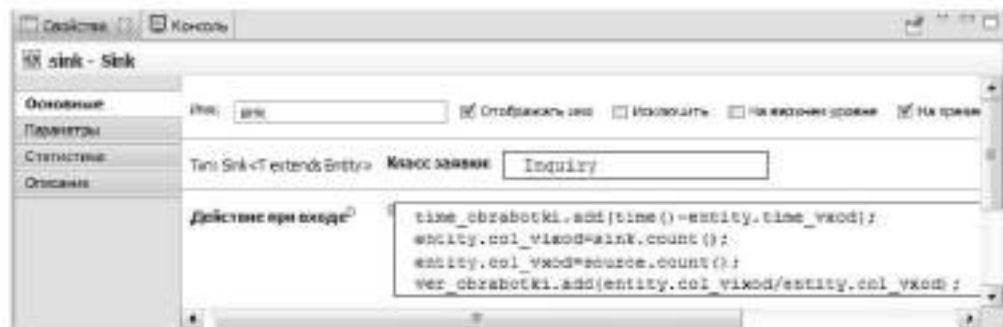


Рис. 1.49. Объект `sink` с измененными параметрами

#### Удаление и добавление новых полей класса заявок

Обратите внимание, что вам не пришлось использовать поле `time_vixod`, так как вместо него была использована функция

`time()`, возвращающая, как вам уже известно, текущее значение модельного времени.

Удалите поле `time_vxod`.

1. В поле Проект дважды щелкните кнопку Inquiry. Откроется редактор кода (см. [рис. 1.44](#)).
2. Удалите обычным образом все, что касается `time_vxod`. Получите код, представленный на [рис. 1.50](#).
3. Закройте редактор кода.



The screenshot shows a Java code editor window with the title bar "Main Inquiry.java". The code is as follows:

```

/*
 * Inquiry
 */
public class Inquiry extends com.xj.anylogic.libraries.enterprise.Enti
{
    double time_vxod;
    float col_vxod;
    float col_vxod;

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

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

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

```

Рис. 1.50. Редактор кода после удаления поля time\_vixod

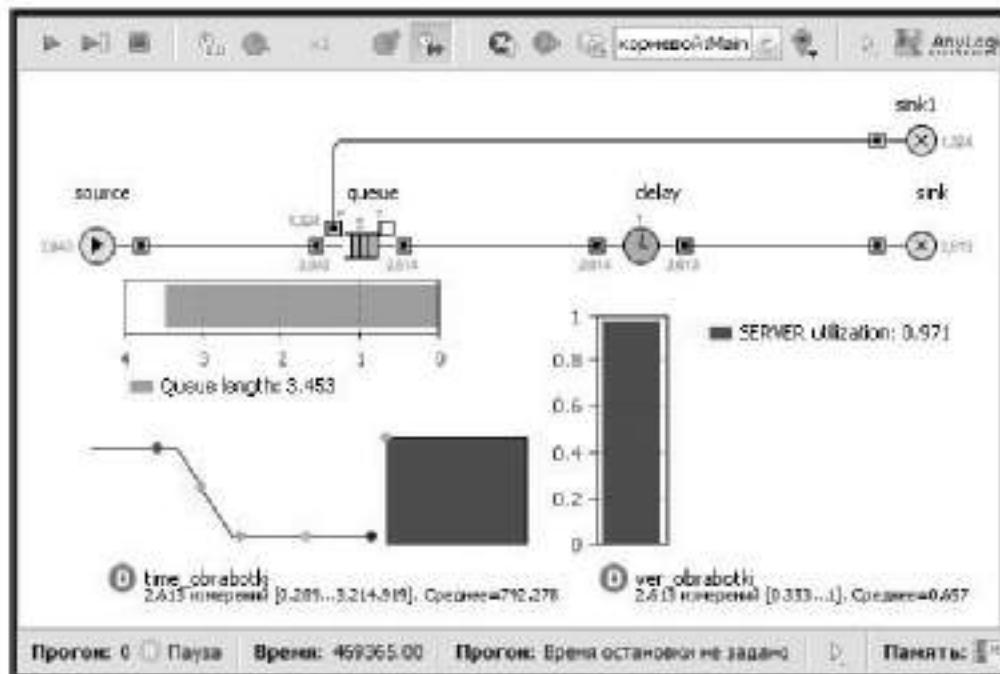


Рис. 1.51. Фрагмент работы модели

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

Итак, все условия постановки задачи выполнены. Запустите модель. Выберите виртуальное время. Наблюдайте за работой модели (Рис. 1.51).

## 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 сверху объекта `delay`, чтобы было видно, к какому объекту относится параметр.
3. Перейдите на страницу Основные панели Свойства (Рис. 1.52).

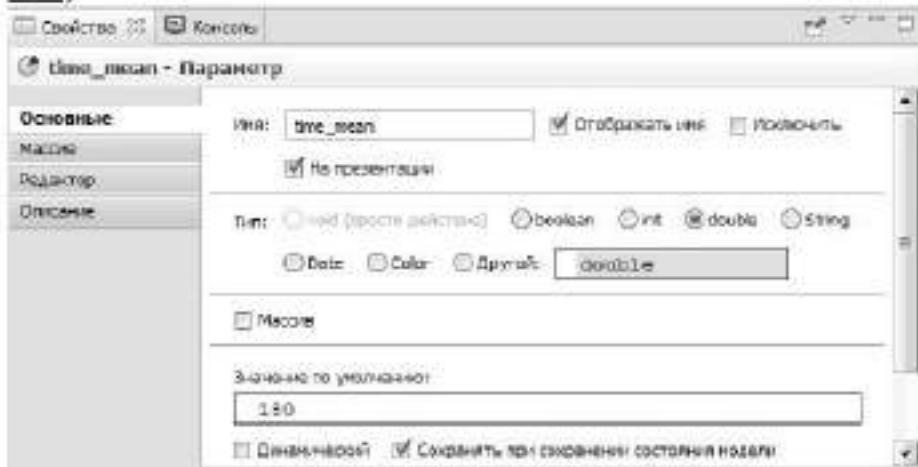


Рис. 1.52. Окно установки свойств элемента Параметр

4. В поле Имя введите имя параметра `time_mean` (среднее время). По этому имени параметр будет доступен из кода.
5. Задайте тип параметра `double`.
6. В поле Значение по умолчанию установите 180. Если значение не будет задано явно, то по правилам Java оно будет равно нулю.
7. Выделите объект `delay`.
8. На странице Основные панели Свойства в поле Время задержки вместо выражения `exponential(1/180.0)` введите `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. 1.54).
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. 1.53).
4. Установите флажок Связать с: и в активизированное поле введите `time_mean`.

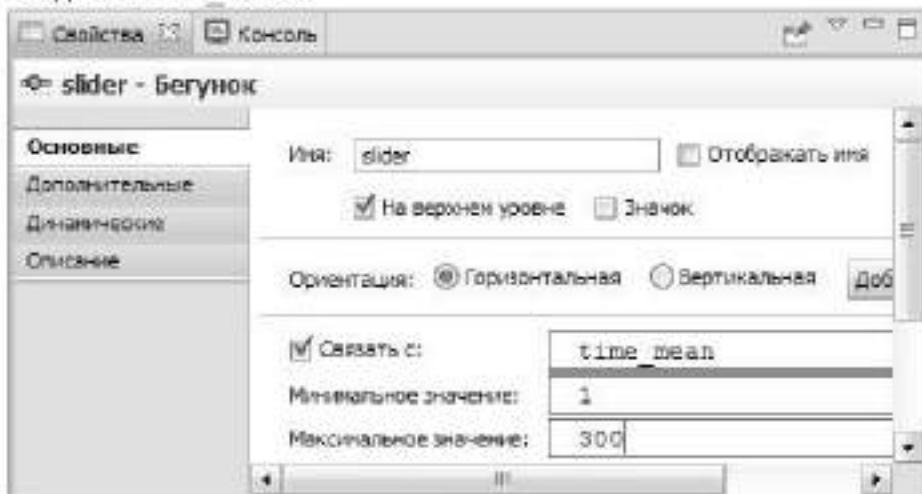


Рис. 1.53. Окно установки свойств элемента управления Бегунок

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. 1.54).

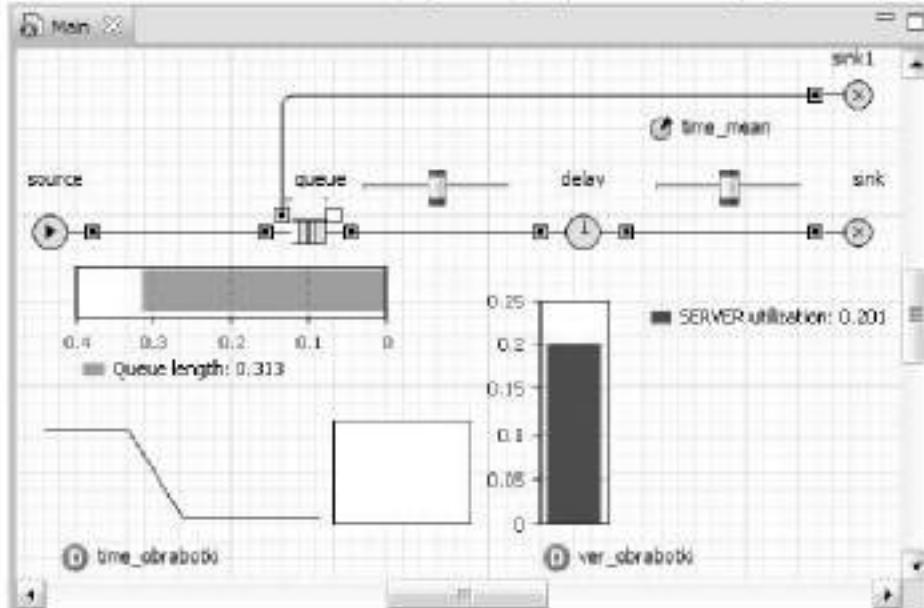


Рис. 1.54. На диаграмму добавлены Параметр и элементы управления

2. Поместите бегунок над 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.
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.

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

Изменение значения в окне инспектора поддерживается для следующих элементов:

- простая переменная;
- параметр;
- накопитель.

И для следующих типов:

- численные;
- логический (boolean);
- текстовый (String).

7. Удалите элемент Бегунок для Параметра `time_mean`.
8. Запустите модель и приостановите её.
9. Щелкните по значку Параметра `time_mean`.
10. Перейдите в режим редактирования (Рис. 1.55).
11. Введите новое значение: 240.0.

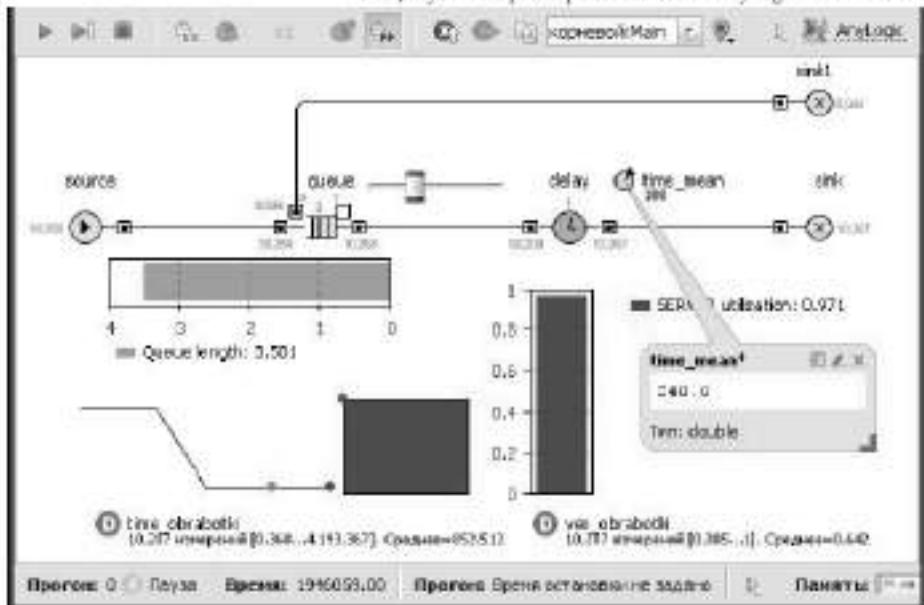


Рис. 1.55. Ввод нового значения параметра в окно инспектора

## Добавление гистограмм

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

1. Перетащите элемент Гистограмма из палитры Статистика в то место графического редактора, куда хотите ее поместить.
2. Укажите, какой элемент сбора данных хранит данные, которые вы хотите отображать на гистограмме: щелкните мышью кнопку Добавить данные и введите в поле Данные имя соответствующего элемента: `time_obrabotki` (Рис. 1.56). Установите Отображать среднее.
3. В поле Заголовок: введите `Histogram Time  
obrabotki`.
4. Запустите модель. Фрагмент работы показан на рис. 1.57.

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

## 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 распределенным по показательному (экспоненциальному) закону со средним значением  $T_2 = 3$  мин.

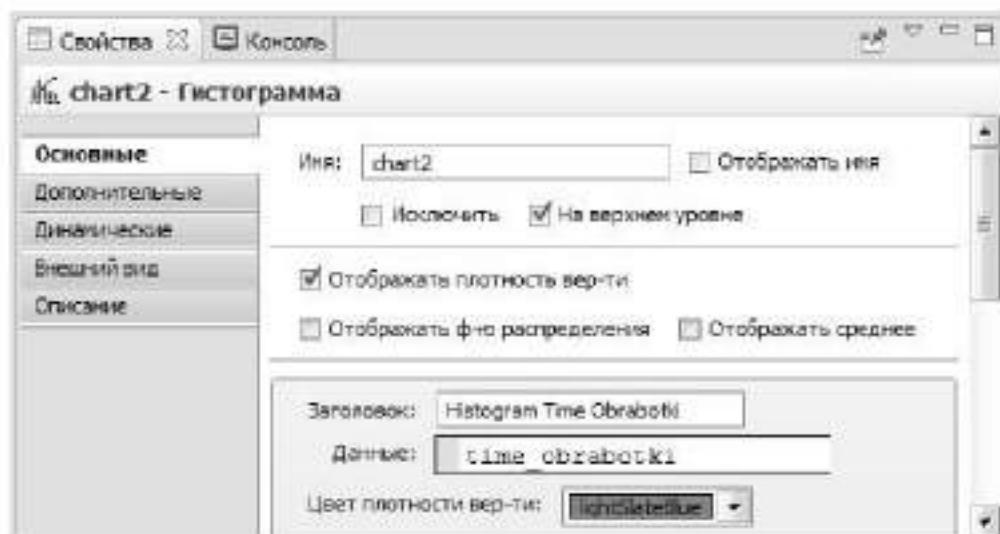


Рис. 1.56. Окно установки свойств элемента Гистограмма

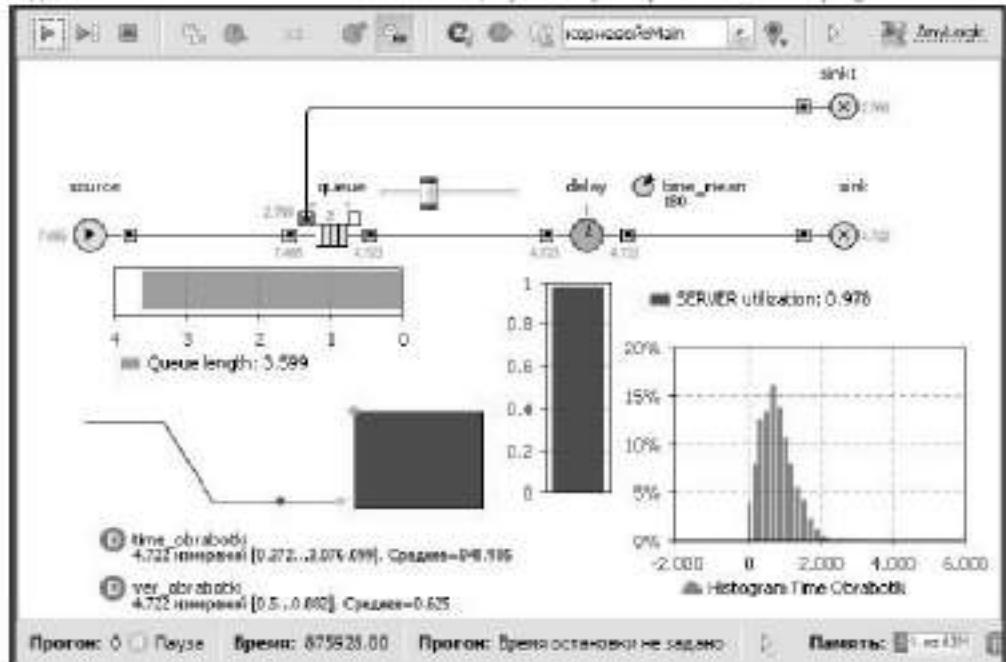


Рис. 1.57. Фрагмент работы модели с элементом управления и гистограммой

Однако в модели, реализованной средствами GPSS World (п. 1.1), 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

$$\text{VrObr VARIABLE (Normal(2,(S1\_Koef),(S2\_Koef))/Q}$$

Кроме того, в этой 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`.
2. Из палитры Основная перетащите три элемента Параметр на диаграмму класса Main (Рис. 1.58).

3. В поле Имя каждого из элементов введите S1\_, S2\_ и Q\_ соответственно.
4. Выберите Тип double.
5. В поле Значение по умолчанию каждого из элементов введите 60000000, 200000 и 600000 соответственно.
6. Перетащите элемент Простая переменная.
7. В поле Имя укажите KolZap.
8. Выделите объект delay.
9. В поле Время задержки вместо `exponential(1/time_mean)` введите: `(normal(S2_,S1_))/Q_`
10. Выделите объект sink. В поле Действие при входе к имеющемуся там коду добавьте код:

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

В GPSS-модели количество прогонов равнялось 9604. 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 9604 раз, то для получения среднего значения это количество нужно разделить на 9604, что и предусмотрено в коде.

11. Показатели моделируемой системы нужно определить в течение 3600 с, поэтому время моделирования в AnyLogic составит 34574400 единиц.
12. В панели Проект выделите Simulation. На странице Модельное время в поле Установить выберите В заданное время.
13. В поле Конечное время установите 34574400.
14. Запустите модель и дождитесь окончания моделирования.

Результаты моделирования приведены на Рис. 1.59.

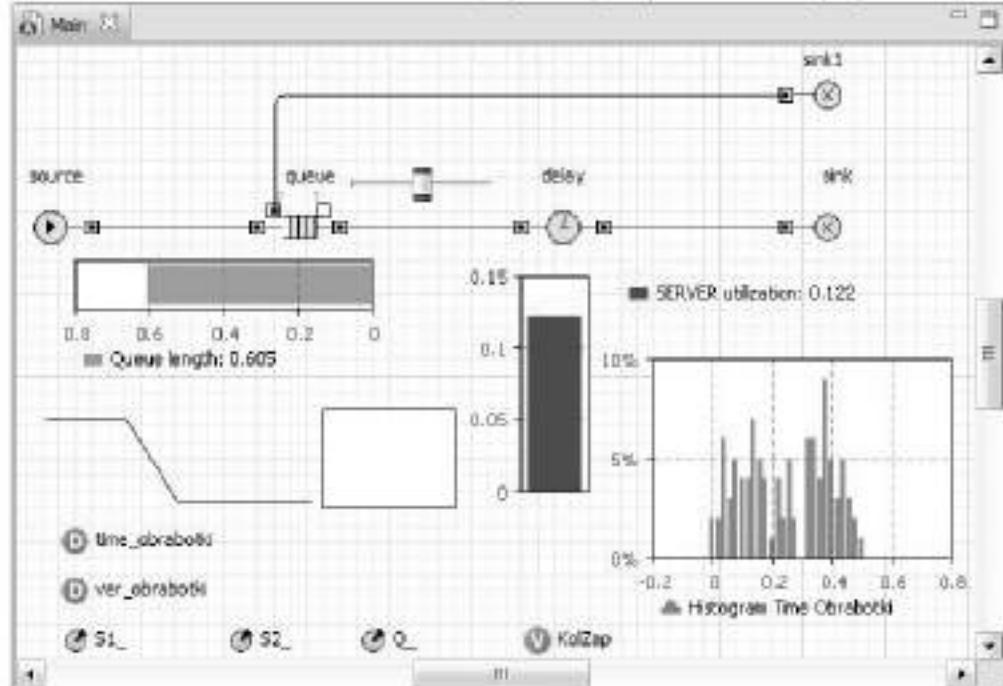


Рис. 1.58. Элементы AnyLogic-модели, соответствующие постановке GPSS-модели

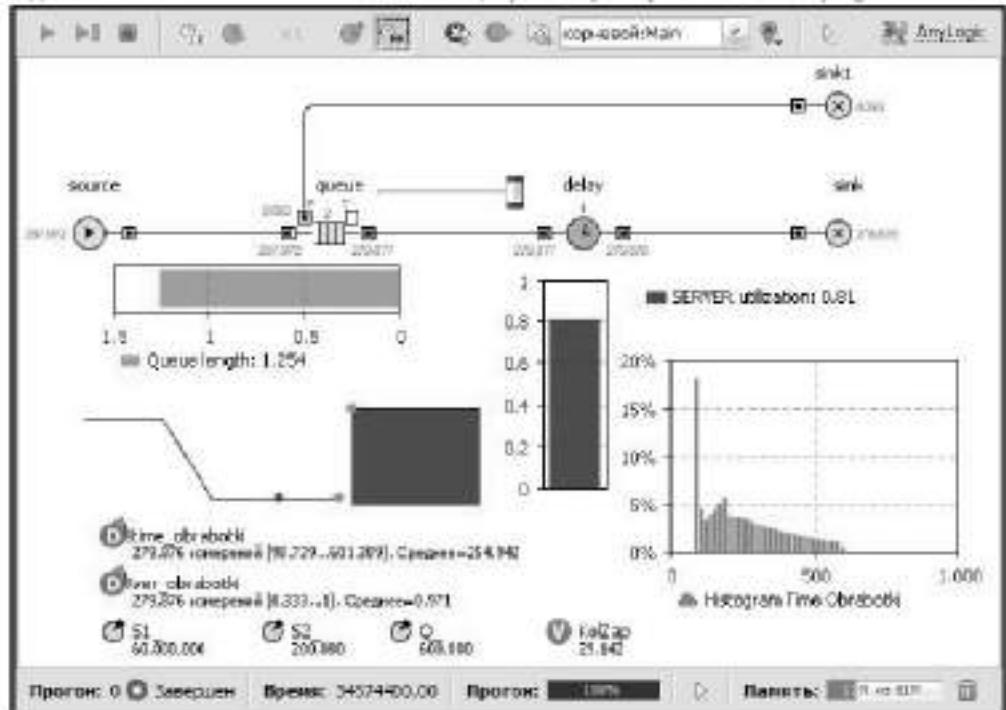


Рис. 1.59. Результаты моделирования обработки данных сервером

## Интерпретация результатов решения прямой задачи

Для проведения исследований на модели сделайте ещё несколько дополнений и изменений.

1. Из палитры Основная перетащите три элемента Параметр на диаграмму класса Main. Разместите их в один ряд с параметрами S1\_, S2\_ и др.
2. В поле имя первого параметра введите timeMean - среднее время поступления запросов. 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.
5. Выделите второй элемент Параметр. В поле имя введите

- kolProg - количество прогонов модели. Оставьте тип **double**.
6. В поле Значение по умолчанию введите 9604.
  7. Выделите объект **sink**.
  8. В поле Действие при входе в имеющемь там коде замените последнюю строку следующей:

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

На рис. 1.59 количество обработанных запросов KoZap сервером выдаётся с дробной частью. Теперь, вследствие применения процедуры **round**, количество запросов будет целым (Рис. 1.60).

Также при изменении количества запросов не надо буде искать код для требуемой корректировки. Однако всё-таки потребуется соответствующим образом изменить модельное время. Например, если потребуется выполнять с моделью 1000 прогонов, то модельное время следует установить равным  $3600 * 1000 = 3600000$ . Для этого нужно в окне Проекты выделить **Simulation** и на странице Модельное время в поле Конечное время: ввести 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 10.
11. Выделите объект **queue**. В поле Вместимость введите **emkBuf**.

Проведём несколько экспериментов в AnyLogic и GPSS World.

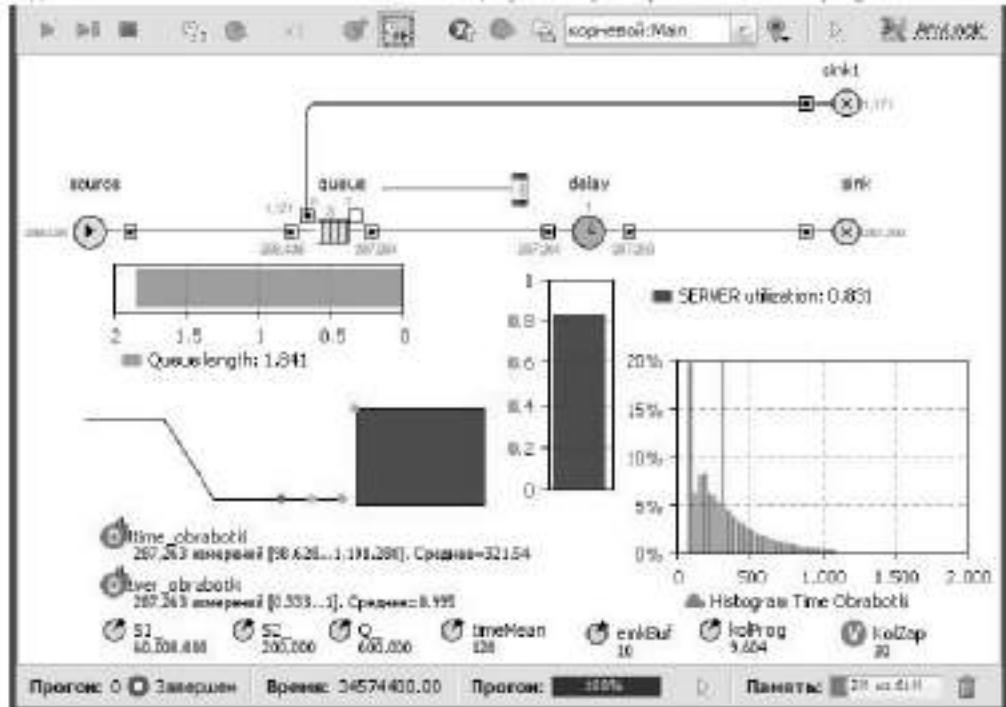


Рис. 1.60. Результаты моделирования при emkBuf = 10 сообщений

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\_Prog* сервера.

Результаты экспериментов - решения прямой задачи в GPSS World и AnyLogic приведены в Табл. 1.2. Из сравнительного анализа полученных результатов следует, что они или равны, или отличаются незначительно.

Разница между количеством обработанных сервером запросов составляет  $\Delta_1 = |29,161 - 29,142| = 0,019$  в первом эксперименте, во втором и пятом  $\Delta_2 = 1$  (после округления до целого), а среднее время обработки одного запроса -  $\Delta_3 = 0,190 \dots 0,556$ . Вероятности обработки запросов отличаются на  $\Delta_4 = 0,001$  в трёх экспериментах, а в трёх равны. Коэффициенты

использования (загрузки) серверов или равны или отличаются на  $\Delta_5 = 0,001 \dots 0,002$ .

Машинное время выполнения модели в GPSS World составляет около 30 с, а в AnyLogic примерно 120 с.

Таблица 1.2.

Показатели	GPSS World	AnyLogic
<b>timeMean = 120, emkBuf = 5</b>		
Количество обработанных запросов	29,161	29,142
Вероятность обработки запросов	0,97	0,971
Среднее время обработки одного запроса	255,262	254,942
Коэффициент использования сервера	0,81	0,81
<b>timeMean = 120, emkBuf = 10</b>		
Количество обработанных запросов	29	30
Вероятность обработки запросов	0,995	0,995
Среднее время обработки одного запроса	328,328	321,54
Коэффициент использования сервера	0,833	0,831
<b>timeMean = 40, emkBuf = 5</b>		
Количество обработанных запросов	36	36
Вероятность обработки запросов	0,4	0,399
Среднее время обработки одного запроса	555,385	555,166
Коэффициент использования сервера	1	1
<b>timeMean = 40, emkBuf = 10</b>		
Количество обработанных запросов	36	36
Вероятность обработки запросов	0,399	0,399
Среднее время обработки одного запроса	1055,335	1055,108
Коэффициент использования сервера	1	1
<b>timeMean = 40, emkBuf = 10, Q_ = 1000000</b>		
Количество обработанных запросов	59	60
Вероятность обработки запросов	0,666	0,665
Среднее время обработки одного запроса	591,86	591,304

Коэффициент использования сервера	1	1
<b>timeMean = 40, emkBuf = 15, Q_ = 2000000</b>		
Количество обработанных запросов	90	90
Вероятность обработки запросов	1	1
Среднее время обработки одного запроса	74,859	75,049
Коэффициент использования сервера	0,751	0,75

# Модель процесса изготовления в цехе деталей

## Модель в GPSS World

### Решение прямой задачи

#### Постановка задачи

Изготовление в цехе детали начинается через случайное время  $T_n$ . 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  $T_1, T_2, \dots, T_n$  соответственно. После каждой операции в течение времени  $T_{x1}, T_{x2}, \dots, T_{xn}$  следует контроль. Время выполнения операций и контроля - случайное. Контроль не проходят  $q_1, q_2, \dots, q_n$  % деталей соответственно.

Забракованные детали поступают на пункт окончательного контроля и проходят на нем проверку в течение времени, распределённого по экспоненциальному закону со средним значением  $T_x$ . 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.

## Исходные данные

$n_1 = 6$ ;  $Exponential(T_n) = Exponential(30)$ ;  $q_1 = 12\%$ ,  $q_2 = 15\%$ ;  
 $n = 3$ ;  $Exponential(T_1) = Exponential(30)$ ;  $q_3 = 10\%$ ,  $q_4 = 80\%$ ;  
 $Exponential(T_2) = Exponential(25)$ ;  $Exponential(T_3) = Exponential(35)$ ;  
 $Exponential(T_{k1}) = Exponential(4)$ ;  $Exponential(T_{k2}) = Exponential(5)$ ;  
 $Exponential(T_{k3}) = Exponential(15)$ ;  $Exponential(T_k) = Exponential(8)$ .

### 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  $\alpha = 0,99$ .

### 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.

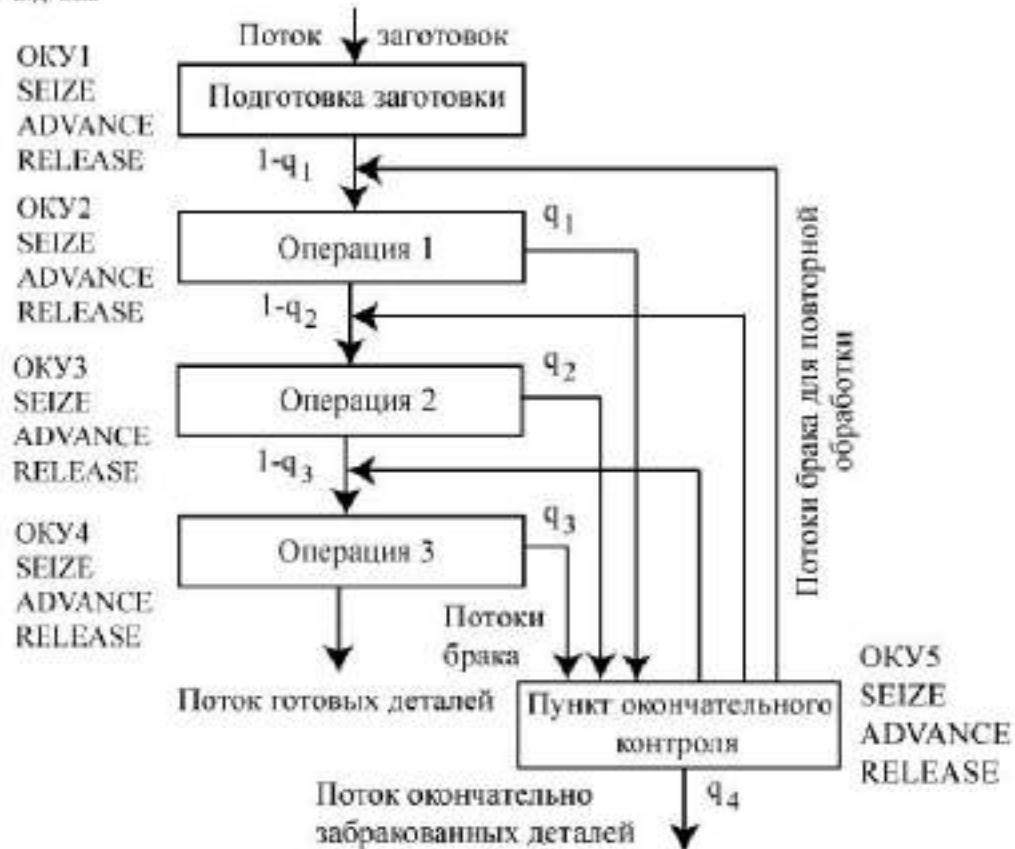


Рис. 2.1. Цех как система массового обслуживания

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:

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

Программа модели прямой задачи приведена ниже.

### Программа модели

```
;Модель процесса изготовления деталей. Прямая задача
; Задание исходных данных
TimeMod EQU 480; Время моделирования, 1 ед. мод. вр. = 1 мин
; Среднее время
Tn_ EQU 35 ; между поступлениями заготовок
T1 EQU 30 ; выполнения 1-й операции, мин
T2 EQU 25 ; выполнения 2-й операции, мин
T3 EQU 35 ; выполнения 3-й операции, мин
Tk1 EQU 4 ; контроля после 1-й операции, мин
Tk2 EQU 5 ; контроля после 2-й операции, мин
Tk3 EQU 15 ; контроля после 3-й операции, мин
Tk EQU 8 ; окончательного контроля, мин
q1_ EQU .12 ; Доля брака после 1-й операции
q2_ EQU .15 ; Доля брака после 2-й операции
q3_ EQU .10 ; Доля брака после 3-й операции
q4_ EQU .80 ; Доля окончательного брака
; Описание функции времени подготовки заготовок
Pod FUNCTION RN10,D6
.05,10/.18,14/.34,21/.56,22/.85,28/1,25
;Сегмент имитации изготовления деталей
GENERATE (Exponential(23,0,Tn_)) ; Источник заготовок
; Подготовка заготовок для деталей
QUEUE Pod ; Встать в очередь
SEIZE Pod ; Начать подготовку заготовки
DEPART Pod ; Покинуть очередь
ADVANCE (Exponential(34,0,FN$Pod)); Подготовка
RELEASE Pod ; Закончить подготовку заготовки
; Имитация выполнения 1-й операции
DCount ASSIGN 1,1 ; Код 1 - проходит первый раз
ASSIGN 2,1 ; Код 1 в P2-признак 1-й операции
Oper1 QUEUE P2 ; Встать в очередь
```

SEIZE Konveer1 ; Начать 1-ю операцию  
 DEPART P2 ; Покинуть очередь  
 ADVANCE (Exponential(23,0,T1)); 1-я операция  
 RELEASE Konveer1 ; Закончить 1-ю операцию  
 ADVANCE (Exponential(23,0,Tk1)); Контроль 1-й операции  
 TRANSFER q1\_„Sboi ; Брак на пункт контроля  
 ; Имитация выполнения 2-й операции  
 ASSIGN 2,2 ; Код 2 в Р2-признак 2-й операции  
 Oper2 QUEUE P2 ; Встать в очередь  
 SEIZE Konveer2 ; Начать вторую операцию  
 DEPART P2 ; Покинуть очередь  
 ADVANCE (Exponential(23,0,T2)) ; 2-я операция  
 RELEASE Konveer2 ; Закончить 2-ю операцию  
 ADVANCE (Exponential(23,0,Tk2)) ; Контроль 2-й операции  
 TRANSFER q2\_„Sboi ; Брак на пункт контроля  
 ; Имитация выполнения 3-й операции  
 ASSIGN 2,3 ; Код 3 в Р2-признак 3-й операции  
 Oper3 QUEUE P2 ; Встать в очередь  
 SEIZE Konveer3 ; Начать третью операцию  
 DEPART P2 ; Покинуть очередь  
 ADVANCE (Exponential(23,0,T3)) ; 3-я операция  
 RELEASE Konveer3 ; Закончить 3-ю операцию  
 ADVANCE (Exponential(23,0,Tk3)); Контроль 3-й операции  
 TRANSFER q3\_„Sboi ; Брак на пункт контроля  
 EndOper1 TERMINATE ; Счёт готовых деталей  
 ; Сегмент имитации работы пункта контроля  
 Sboi TEST E P1,1,EndOper ; Если второй раз, то в окончательный брак  
 QUEUE Kont ; В очередь на пункт контроля  
 SEIZE Kont ; Занять пункт контроля  
 DEPART Kont ; Покинуть  
 ADVANCE (Exponential(23,0,Tk)); Окончательный контроль  
 RELEASE Kont ; Освободить пункт контроля  
 TRANSFER q4\_„EndOper ; В окончательный брак  
 ASSIGN 1,2 ; Код 2 в Р1-деталь пойдёт второй раз  
 Met1 TRANSFER ,(Met1+P2)  
 TRANSFER ,Oper1 ; Повторно на 1-ю операцию  
 TRANSFER ,Oper2 ; Повторно на 2-ю операцию  
 TRANSFER ,Oper3 ; Повторно на 3-ю операцию  
 EndOper TERMINATE ; Счет брака

```

; Сегмент задания времени моделирования и расчета результатов модел
GENERATE TimeMod ; Время моделирования
TEST L X$Prog,TG1,Met11 ; Если условие выполняется, то
SAVEVALUE Prog,TG1 ; X$Prog=TG1 содержимому счетчика завершения
Met11 TEST E TG1,1,Met12 ; Если содержимое счетчика равно 1, то ре
SAVEVALUE NDet,(N$EndOper1/X$Prog) ; Количество готовых деталей
SAVEVALUE Brak,(N$EndOper/X$Prog) ; Количество забракованных деталей
SAVEVALUE DoljaBrak,(X$Brak/(X$Brak+X$NDet))
; Общая доля брака
SAVEVALUE DoljaDet,(X$NDet/(X$Brak+X$NDet))
; Доля готовых деталей
SAVEVALUE NDet,(INT(X$NDet)) ; Количество готовых деталей (целочисленное)
SAVEVALUE Brak,(INT(X$Brak)) ; Количество забракованных деталей
SAVEVALUE SDet,((AC1-X$AC2)/N$EndOper1) ; Среднее время изготовления
SAVEVALUE AC2,AC1
Met12 TERMINATE 1
START 1000,NP ; Число предварительных прогонов
RESET ; Сброс статистики
START 16641 ; Число основных прогонов

```

Замечание. В программе при обращении несколько раз к встроенному генератору экспоненциально распределенных случайных чисел взято одно и тоже начальное число 23, хотя рекомендуется брать различные начальные числа. Сделано это для чистоты эксперимента - сравнения в последующем результатов моделирования GPSS World с результатами AnyLogic.

Программа модели имеет достаточно подробный комментарий. Поэтому остановимся лишь на некоторых её особенностях.

Для задания исходных данных - времени подготовки заготовок - использована дискретная функция Pod. Это позволяет сократить программу по сравнению с тем, если применять команду EQU. Кроме того, упрощается событийная часть модели, так как в блоке имитации подготовки заготовок

ADVANCE (Exponential(23,0,FN\$Pod)); Имитация подготовки заготовок достаточно указать только ссылку FN\$Pod на функцию.

Коды 1 и 2, записываемые в параметр 1 транзакта, служат признаками не прохождения деталью контроля первый и второй раз соответственно. Признак 2 является основанием отправки детали в брак и исключения её из производства.

Начало сегмента задания времени моделирования и расчёта результатов моделирования построено аналогично этому же сегменту модели п. 1.1.

Для счёта количества готовых и забракованных деталей введены метки EndOper1 и EndOper2 соответственно. Поскольку эти количества накапливаются за все прогоны, то для получения средних значений они делятся на число прогонов X\$Prog, округляются до целого процедурой INT и заносятся в сохраняемые ячейки NDet и Brak соответственно. Далее эти средние значения используются для вычисления относительных долей готовых DoljaDet и забракованных DoljaBrak деталей.

Среднее время SDet изготовления одной детали определяется как отношение абсолютного модельного времени AC1 к количеству подготовленных деталей за все прогоны, т.е. к N\$EndOper1.

Это было бы правильно, если бы не было предварительных прогонов модели. Модельное время этих прогонов не должно учитываться при определении среднего времени изготовления детали. Его нужно запомнить по завершении предварительных прогонов. Для этого введена команда

`SAVEVALUE AC2,AC1 ; Время предварительных прогонов`

После окончания основных прогонов производится расчёт:

`SAVEVALUE SDet,((AC1-X$AC2)/N$EndOper1) ; Среднее время изго`

### Проведение исследований

Полагаем, что вы ввели программу модели, исправили ошибки и выполнили указанное количество прогонов модели. Фрагмент отчёта приведен ниже.

SAVEVALUE	RETRY	VALUE
PROG	0	16641.000
NDET	0	9.000
BRAK	0	3.000
DOLJABRAK	0	0.279
DOLJADET	0	0.721
SDET	0	48.559

В результате решения прямой задачи получим, что за 8 часов цехом будет изготовлено NDet = 9 деталей, относительная доля готовых деталей составит DoljaDet = 0,721, а среднее время изготовления одной детали SDet = 48,559 мин. При этом будет забраковано Brak = 3 детали, относительная доля которых составит DoljaBrak = 0,2279.

Фрагмент отчёта, если не использовать процедуру INT, т.е. не округлять до целого количество изготовленных и количество забракованных деталей:

SAVEVALUE	RETRY	VALUE
PROG	0	16641.000
NDET	0	9.885
BRAK	0	3.821
DOLJABRAK	0	0.279
DOLJADET	0	0.721
SDET	0	48.559

За 8 часов цехом будет изготовлено Ndet = 9,885 деталей, относительная доля готовых деталей DoljaDet = 0,721 и среднее время изготовления одной детали SDet = 48,559 мин останутся такими же. При этом будет забраковано Brak = 3,821 деталей, относительная доля которых от общего количества готовых и забракованных DoljaBrak = 0,279 также не изменится.

Замечание. Если вы вместо 1000 укажите 100 предварительных прогонов и запустите модель, то получите тот же результат.

## Решение обратной задачи

Целью обратной задачи является определение среднего времени на изготовление какого-то количества деталей. Для проверки работоспособности модели возьмём количество деталей, полученных в результате решения прямой задачи, т.е. Det = 9.

### Особенности построения программы модели

Модель для решения обратной задачи приведена ниже.

; Модель процессса изготвления деталей. Обратная задача

...

; Имитация выполнения 3-й операции

ASSIGN 2,3 ; Код 3 в Р2-признак 3-й операции

Oper3 QUEUE P2 ; Встать в очередь

SEIZE Konveer3 ; Начать третью операцию

DEPART P2 ; Покинуть очередь

ADVANCE (Exponential(23,0,T3)) ; 3-я операция

RELEASE Konveer3 ; Закончить 3-ю операцию

ADVANCE (Exponential(23,0,Tk3)); Контроль 3-й операции

TRANSFER q3\_,Sboi ; Брак на пункт контроля

TRANSFER ,Met2 ; Готовые детали

; Сегмент имитации работы пункта контроля

Sboi QUEUE Kont ; В очередь на пункт контроля

SEIZE Kont ; Занять пункт контроля

DEPART Kont ; Покинуть очередь на пункт контроля

ADVANCE (Exponential(23,0,Tk)); Окончательный контроль

RELEASE Kont ; Освободить пункт контроля

TRANSFER q4\_,EndOper ; В окончательный брак

TESTE P1,1,EndOper ; Если второй раз, то в окончательный брак

ASSIGN 1,2 ; Код 2 в Р1-деталь пойдёт второй раз

Met1 TRANSFER ,(Met1+P2)

TRANSFER ,Oper1 ; Повторно на 1-ю операцию

TRANSFER ,Oper2 ; Повторно на 2-ю операцию

TRANSFER ,Oper3 ; Повторно на 3-ю операцию

EndOper TERMINATE ; Счет брака

; Сегмент завершения моделирования и расчета результатов

```

Met2 TEST L X$Prog,TG1,Met3 ; Если условие выполняется, то
    SAVEVALUE Prog,TG1 ; X$Prog=TG1 счетчику завершений
Met3 SAVEVALUE NDet+,1 ; Счет количества готовых деталей
TEST E X$NDet,Det,Ter1 ; Если готово Det деталей, зафиксировать одн
    TEST E TG1,1,Met4 ; Если в счетчике завершений 1, то расчет резуль
    SAVEVALUE Brak,(N$EndOper/X$Prog) ; Количество забракованных д
    SAVEVALUE DoljaBrak,(X$Brak/(X$Brak+Det); Общая доля брака
    SAVEVALUE DoljaDet,(Det/(X$Brak+Det)) ; Доля готовых деталей
    SAVEVALUE Brak,(INT(X$Brak)) ; Количество забракованных деталей
    SAVEVALUE TDet,(((AC1-X$AC2)/X$Prog)/60); Среднее время изгото
    SAVEVALUE SDet,((X$TDet/Det)#60); Среднее время изготовления одн
    SAVEVALUE AC2,AC1 ; Время предварительных прогонов
    SAVEVALUE X$Prog,0 ; Обнуление ячейки X$Prog
Met4 SAVEVALUE NDet,0 ; Обнуление X$NDet
TERMINATE 1 ; Из счётчика завершений минус 1
Ter1 TERMINATE ; Вывод вспомогательных транзактов
START 1000,NP ; Число предварительных прогонов модели
RESET ; Сброс статистики
START 16641 ; Число основных прогонов модели

```

В приведенной программе модели для решения обратной задачи в целях сокращения исключён текст до части, имитирующей выполнение 3-й операции, так как он такой же, как и в программе модели для решения прямой задачи. Единственное отличие состоит в том, что в исходные данные следует добавить команду для задания количества деталей, которые нужно изготовить:

Det EQU 9 ; Количество деталей, которые нужно изготовить

Округление до целого количества забракованных деталей производится после расчёта доли готовых и забракованных деталей. Если иначе, то расчёт долей будет некорректным.

В программе модели прямой задачи годные детали после выполнения 3-й операции подсчитывались. Имитирующие их транзакты уничтожались:

EndOper1 TERMINATE ; Счёт готовых деталей

В программе модели обратной задачи годные детали нужно отправить в

сегмент организации завершения моделирования. Для этого приведенный только что блок счёта готовых деталей заменён следующим (в тексте программы выше он выделен жирным):

**TRANSFER ,Met2 ; Готовые детали**

Сегмент имитации работы пункта контроля остаётся неизменным. Остановимся на сегменте завершения моделирования и расчета результатов.

Сохраняемая ячейка **NDet** служит для счёта текущего количества изготовленных деталей. Как только выполняется условие **X\$NDet = Det**, фиксируется один прогон модели. **Det** - переменная пользователя, которой задаётся количество деталей, время подготовки которых нужно определить.

Вам уже известно, что модельное время предварительных прогонов не должно учитываться при расчёте среднего времени изготовления 9 деталей. Поэтому оно запоминается в сохраняемой ячейке **X\$AC2**, а при расчёте вычитается из **AC1**:

**SAVEVALUE TDet,((AC1-X\$AC2)/X\$Prog)/60 ; Среднее время изготов**

### Проведение исследований

Фрагмент отчёта приведен ниже:

SAVEVALUE	RETRY	VALUE
PROG	0	16641,000
BRAK	0	3.000
DOLJABRAK	0	0.279
DOLJADET	0	0.721
TDET	0	7.266
SDET	0	48,443

В результате решения обратной задачи получим, что **Det = 9** деталей будут изготовлены цехом за **TDet = 7,266** часа, относительная доля которых составит **DoljaDet = 0,721**, а среднее время изготовления одной детали **SDet = 48,443** мин. При этом будет забраковано

*BRAK* = 3 детали, относительная доля которых составит *DoljaBRAK* = 0,279.

Замечание. Если вы замените 1000 предварительных прогонов на 100 и запустите модель, то получите:

SAVEVALUE	RETRY	VALUE
PROG	0	16641.000
BRAK	0	3.000
DOLJABRAK	0	0.278
DOLJADET	0	0.722
TDET	0	7.256
SDET	0	48.374

Видно, что результаты моделирования изменились незначительно. Но если эти изменения для вас несущественны, можно оставить указанное число предварительных прогонов модели в дальнейших исследованиях.

### Проведение экспериментов

В главе 1 был проведен дисперсионный анализ с имитационной моделью, предназначенный для решения прямой задачи. Здесь же мы проведём дисперсионный анализ с моделью, решающей обратную задачу.

Исследовать влияние качества  $q_1, q_2, q_3, q_4$  выполнения операций на время изготовления D деталей. Значения уровней факторов приведены в табл. 2.2.

Таблица 2.2.

Уровни факторов	Факторы			
	$q_1$	$q_2$	$q_3$	$q_4$
Нижний	0,1	0,15	0,1	0,2
Верхний	0,25	0,35	0,2	0,9

Результаты моделирования необходимо получить с точностью  $\varepsilon = 1$  мин и доверительной вероятностью  $\alpha = 0,99$ . Но поскольку остается условие определения относительного числа подготовленных деталей,

то количество прогонов остаётся равным 16641, что и в модели для решения прямой задачи.

Некоторые особенности построения для такого случая модели были изложены в п. 2.1.2.1. Однако показатели, которые там определялись в дисперсионном анализе, не были временными. А это является существенным в построении сегмента организации завершения моделирования.

Поскольку результатом моделирования является оценка математического ожидания времени TDet изготовления Det деталей, то в ее вычислении используется абсолютное модельное время AC1 (системный числовой атрибут). При проведении дисперсионного анализа встроенный генератор эксперимента имеет две команды START, а между ними - команда RESET. Команда RESET не влияет на абсолютное модельное время. Поэтому AC1 будет суммой абсолютного модельного времени предварительных прогонов до установившегося режима, обозначим его AC2, и абсолютного модельного времени, пусть AC3, основных прогонов, в ходе которых собирается интересующая нас статистика. Нам для расчетов нужно AC3. Для его получения в программу введены строки:

```
SAVEVALUE AC3,(AC1-X$AC2)
SAVEVALUE AC2,AC1
```

После предварительных прогонов в ячейке X\$Prog сохранится указанное в первой команде START количество прогонов. Эта ячейка используется в первой строке рассматриваемого сегмента и её содержимое должно быть равным нулю. В противном случае модель будет работать неверно. Для предотвращения ошибки введена строка:

```
SAVEVALUE X$Prog,0
```

Замечание. Если предварительное число прогонов модели меньше числа основных прогонов, то можно обойтись и без этой команды.

Проведите эксперимент. Как проводится дисперсионный анализ, вы уже знаете. Теме не менее, установите необходимые данные эксперимента согласно рис. 2.2. Укажите число предварительных

прогонов 1000 вместо 100, установленных по умолчанию.

Результаты эксперимента показаны на [рис. 2.3](#). Видно, что все четыре фактора существенные. Наибольшее влияние на функцию отклика оказывает фактор **B**, что вполне логично, так как из первых трёх он имеет наибольший верхний уровень, т. е. наибольшую долю брака. Наименьшую значимость имеет фактор **C**, что также объясняется более низким значением доли брака.

Измените непосредственно в процедуре запуска генерированного Plus-эксперимента количество предварительных прогонов с 1000 на 100 и запустите модель. Вы получите практически те же самые результаты, например, Grand Mean = 9,006 вместо 9,017.

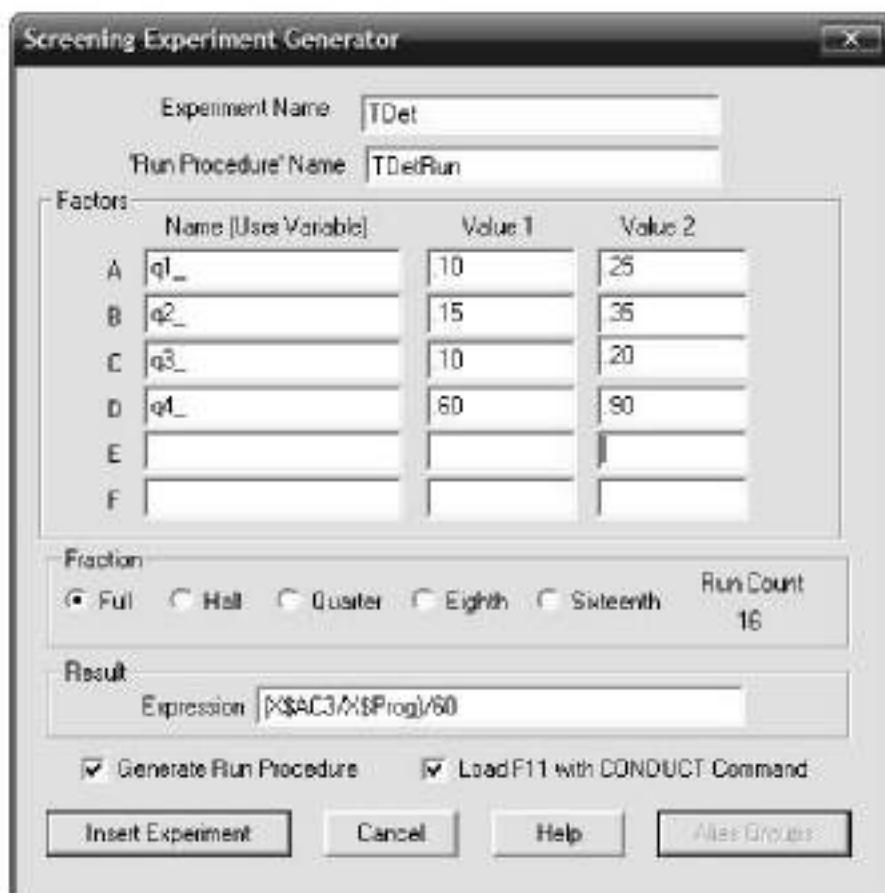


Рис. 2.2. Диалоговое окно (заполненное) Screening Experiment Generator

01/12/11 10:17:09	Alias Group	Effect	Sum of Squares	Degrees of Freedom	F-for Only Main Effects	Critical Value of F (p=.05)
01/12/11 10:17:09						
01/12/11 10:17:09	A	1.373	7.540	1	78.503	4.84
01/12/11 10:17:09	B	2.025	16.420	1	170.153	4.84
01/12/11 10:17:09	AB	0.140				
01/12/11 10:17:09	C	0.885	3.135	1	32.635	4.84
01/12/11 10:17:09	AC	0.077				
01/12/11 10:17:09	BC	0.107				
01/12/11 10:17:09	ABC	0.002				
01/12/11 10:17:09	D	1.400	7.839	1	81.611	4.84
01/12/11 10:17:09	AD	0.265				
01/12/11 10:17:09	BD	0.358				
01/12/11 10:17:09	ABD	0.046				
01/12/11 10:17:09	CD	0.161				
01/12/11 10:17:09	ACD	0.019				
01/12/11 10:17:09	BCD	0.006				
01/12/11 10:17:09	ABCD	-0.009				
01/12/11 10:17:09	Error		1.057	11		
01/12/11 10:17:09	Total		35.989	15		
01/12/11 10:17:09	Grand Mean		9.017			
01/12/11 10:17:09						

Рис. 2.3. Результаты отсеивающего эксперимента

## 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.

## Исходные данные. Использование массивов

Для ввода исходных данных используем элементы Параметр. Идентификаторы возьмём те же (кроме времени моделирования и количества прогонов модели), что и в GPSS-модели, только без знака подчёркивания.

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. Щелкните кнопку Далее.
3. На открывшейся второй странице Мастера создания модели выберите Начать создание модели "с нуля". Щелкните кнопку Далее.
4. В Палитре выделите Презентация.
5. Перетащите элемент Скруглённый прямоугольник в нужное место.
6. Перетащите элемент text и на странице Основные панели Свойства в поле Текст: введите Исходные данные.
7. В Палитре выделите Основная. Перетащите элементы Параметр на элемент с именем Исходные данные. Разместите их так, как показано на рис. 2.4. Значения свойств установите согласно табл. 2.3. На рис. 2.4, как вы, наверное, уже заметили, два элемента Параметр отличаются от остальных. Они используются для ввода данных табл. табл. 2.1 как одномерных массивов.



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	врMod	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 КолVarЗаг.
  3. Установите Тип размерности: Диапазон.
  4. В открывшееся поле Диапазон: введите 1-10.
  5. Щёлкните Готово.

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

1. Из Палитры Основные перетащите элемент Параметр.
2. На странице Основные панели Свойства в поле Имя: введите верVarЗаг.
3. Установите флажок Массив. Рядом появится значок (...). Щёлкните по нему.
4. Откроется страница свойств Массив. В окошке Возможные размерности: выделите КолVarЗаг.
5. Щёлкните по кнопке Размерность КолVarЗаг появится в окошке Выбранные размерности.
6. Вернитесь на страницу Основные панели Свойства.
7. В поле Значение по умолчанию: введите:

{0.05,0.18,0.34,0.56,0.85,1.0,0,0,0,0}

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.

1. Аналогичным образом создайте второй массив с именемсрBrПодгЗаг для среднего времени подготовки заготовки.
2. В поле Значение по умолчанию: введите:

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

В событийную (функциональную) часть модели включим указанные ранее сегменты. Поскольку построение модели это итерационный процесс, то размещение сегментов и объектов AlyLogic 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.

#### 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 и разместите, где считаете нужным. На странице Дополнительные панели Свойства введите в поле Ширина: 240, в поле Высота: 150.
2. Перетащите элемент text и на странице Основные панели Свойства в поле Текст: введите Подготовка заготовки ([Рис. 2.5](#)).
3. В Палитре выделите Enterprise Library. Перетащите объект source на диаграмму класса 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 щелкните 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. Щелкните кнопку Далее.

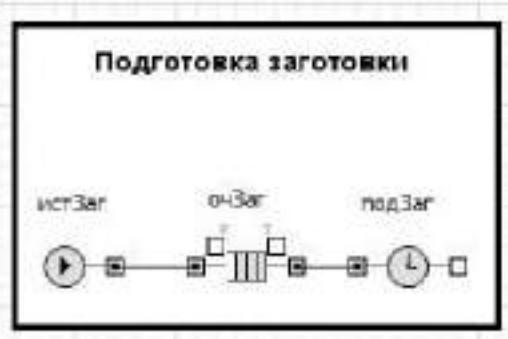


Рис. 2.5. Сегмент Подготовка заготовки

- 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.

- Оставьте выбранными флажки `Создать конструктор` и `Создать метод toString()`.
- Щелкните кнопку `Готово`. Появится редактор кода и автоматически созданный код вашего Java класса. Закройте код.
- Из Палитры Основная перетащите на сегмент Исходные данные элемент `Простая переменная`. На странице Основные панели Свойства дайте Имя: `b`, установите Тип: `double`.
- Выделите объект `source`. На странице Основные панели Свойства установите свойства согласно [рис. 2.6](#).

В коде, приведенном ниже, который вы ввели в поле свойства `Действие 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);
```

13. Из библиотеки Enterprise Library перетащите объекты queue и delay на диаграмму класса Main, разместите в прямоугольнике с именем Подготовка заготовки и соедините как на [рис. 2.5](#).



Рис. 2.6. Элемент source с установленными свойствами

14. Выделите объект queue и на странице Основные панели Свойства установите свойства:

  - Имя: очЗаг
  - Класс заявки: Detail

15. Максимальная вместимость установить флажок.

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

  - Имя: подЗаг
  - Класс заявки: Detail
  - Задержка задаётся Явно
  - Время задержки exponential (1/entity.Tn1)
  - Вместимость 1
  - Действие при выходе entity.a = 1;

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. В Палитре выделите Презентация. Перетащите три элемента Прямоугольник и разместите так, как на [рис. 2.7](#). На странице Дополнительные панели Свойства для каждого прямоугольника введите в поле Ширина: 240, в поле 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. 2.7](#)).
3. В Палитре выделите Enterprise Library. Перетащите для каждого сегмента два объекта queue, два объекта delay и один объект selectOutput на диаграмму класса Main, разместите в прямоугольниках и соедините так, как показано на [рис. 2.7](#).
4. Выделите поочередно объекты, начиная с объекта queue сегмента 1, и на странице Основные панели Свойства установите свойства согласно [рис. 2.7](#) и [табл. 2.4](#).

На сегменте Операция 1 поясним принятые имена объектов: очOp1 - очередь на операцию 1; выпOp1 - имитация выполнения операции 1; очКонOp1 - очередь на контроль после операции 1; конOp11 - имитация контроля после операции 1; конOp12 - розыгрыш результата контроля.

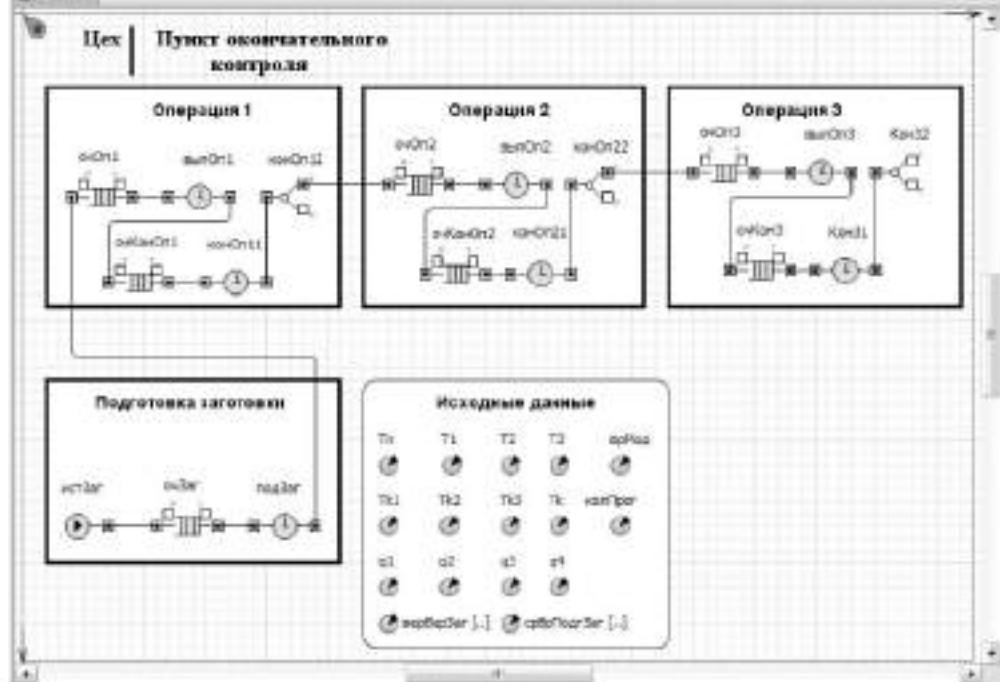


Рис. 2.7. Добавлены сегменты Операция 1, Операция 2, Операция 3

Таблица 2.4.

Объект	Свойства	Значения
Сегмент Операция 1		
queue	имя	очOp1
delay	Класс заявки	Detail
	Максимальная вместимость	Установите флажок
queue1	имя	выOp1
delay	Класс заявки	Detail
	Задержка задаётся	Явно
	Время задержки	exponential (1/T1)
	Вместимость	1
queue1	имя	очКонOp1
	Класс заявки	Detail
	Максимальная вместимость	Установите флажок

	имя	конОп11
delay1	Класс заявки	Detail
	Задержка задаётся	Явно
	Время задержки	exponential (1/Tk1)
	Вместимость	1
	Действие при выходе	entity.n = 1;
selectOutput	имя	конОп12
	Класс заявки	Detail
	Выход true выбирается	С заданной вероятностью
	Вероятность [0..1]	1-q1
Сегмент Операция 2		
queue	имя	очОп2
	Класс заявки	Detail
	Максимальная вместимость	Установите флажок
	имя	выпОп2
delay	Класс заявки	Detail
	Задержка задаётся	Явно
	Время задержки	exponential (1/T2)
	Вместимость	1
queue1	имя	очКонОп2
	Класс заявки	Detail
	Максимальная вместимость	Установите флажок
	имя	конОп21
delay1	Класс заявки	Detail
	Задержка задаётся	Явно
	Время задержки	exponential (1/Tk2)
	Вместимость	1
	Действие при выходе	entity.n = 2
selectOutput	имя	конОп22
	Класс заявки	Detail
	Выход true выбирается	С заданной вероятностью

	Вероятность [0..1]	1 - q2
Сегмент Операция 3		
queue	имя	очOp3
	Класс заявки	Detail
	Максимальная вместимость	Установите флажок
delay	имя	выпOp3
	Класс заявки	Detail
	Задержка задаётся	Явно
	Время задержки	exponential (1/T3)
	Вместимость	1
queue1	имя	очKonOp3
	Класс заявки	Detail
	Максимальная вместимость	Установите флажок
delay1	имя	конOp31
	Класс заявки	Detail
	Задержка задаётся	Явно
	Время задержки	exponential (1/Tk3)
	Вместимость	1
	Действие при выходе	entity.n = 3
selectOutput	имя	конOp32
	Класс заявки	Detail
	Выход true выбирается	С заданной вероятностью
	Вероятность [0..1]	1 - q3

Создание нового класса активного объекта

На рис. 2.7 показаны четыре функциональных сегмента модели, которые построены с использованием 18 объектов AnyLogic. Если вы начнёте создавать очередной сегмент, то после перетаскивания второго объекта появится сообщение: Ограничение ознакомительной версии: нельзя создавать более 20 вложенных объектов. Поэтому остальные сегменты модели нам нужно разместить на втором активном объекте. Создайте

этот объект.

1. На панели Проект щелкните Main правой кнопкой мыши и выберите из контекстного меню Создать / Класс активного объекта.
2. Откроется окно Новый класс активного объекта.
3. В поле Имя : задайте имя нового класса Kontrol.
4. Если нужно, в поле Описание : введите описание сущности, моделируемой этим классом.
5. Щелкните кнопку Готово.

#### Создание элемента нового класса активного объекта

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 элемент нового класса активного объекта Kontrol.

1. Из Палитры Основная перетащите элемент Порт и разместите сверху в левом ряду ([рис. 2.8](#)).
2. На странице Основные панели Свойства имя port замените именем наOp1.
3. Скопируйте элемент Порт с именем наOp1.
4. Вставьте два элемента Порт ([см. рис. 2.8](#)). При вставке последовательно будут изменяться их имена: наOp2, наOp3.
5. Из Палитры Основная перетащите элемент Порт и разместите сверху в правом ряду ([см. рис. 2.8](#)).

## Значок

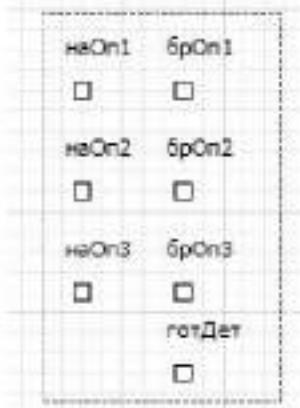


Рис. 2.8. Порты

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. 2.8](#)). При вставке последовательно будут изменяться их имена: `броп2`, `броп3`.
9. Из Палитры Основная перетащите элемент Порт и разместите внизу в правом ряду (см. [рис. 2.8](#)).
10. На странице Основные панели Свойства имя `port` замените именем `готдат`.
11. По мере размещения элементов Порт они автоматически будут объединяться прямоугольником (с пунктирными линиями) и появится надпись Значок.
12. Возвратитесь на диаграмму класса `Main`.
13. На панели Проект выделите `Kontrol`, перетащите на диаграмму класса `Main` объект класса `Kontrol`, разместите как на [рис. 2.9](#).
14. Объект активного класса `Kontrol` создан. На странице Основные панели Свойства уберите флагок Отображать имя и в поле Имя: введите `на_контроль`.
15. В Палитре выделите Презентация. Перетащите элемент Прямоугольник и разместите так, как на [рис. 2.9](#). На странице

Дополнительные панели Свойства введите в поле Ширина: 140, в поле Высота: 220.

16. 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 `конOp12`, `конOp22`, `конOp32` с портами `брOp1`, `брOp2`, `брOp3` соответственно.
2. Соедините порты `наOp1`, `наOp2`, `наOp3` с входами объектов `очOp1`, `очOp2`, `очOp3` соответственно.
3. Соедините выход `T` (`true`) объекта `конOp32` сегмента Операция 3 с портом `готДат`.

На [рис. 2.9](#) вы видите ещё объекты. Переидём к размещению их на диаграмме класса `Main`.



Рис. 2.9. Объект Main с сегментами модели и элементом объекта Kontrol

#### 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 6 AnyLogic предоставляет в распоряжение пользователей специальный элемент для решения этой проблемы - область просмотра. С помощью этого элемента вы можете выделить на диаграмме активного объекта некоторые области, содержащие логически обособленные группы элементов или участки диаграммы. Задав такие области, вы сможете легко переключаться между ними во время выполнения модели с помощью специальных средств навигации, что позволит быстро переходить к тому или иному участку диаграммы.

активного объекта. При этом в окне презентации запущенной модели будут отображаться те элементы активного объекта, которые попали в заданную вами ранее и сделанную в текущий момент активной область просмотра.

Используем две области просмотра. В первой области просмотра разместим объекты первых пяти сегментов (см. [рис. 2.9](#)), во второй - сегменты Пункт окончательного контроля, Склад готовых деталей и Склад бракованных деталей.

Первая область просмотра будет на диаграмме класса Main, а вторая - на новом классе активного объекта Kontrol.

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

1. В Палитре выделите Презентация. Перетащите элемент Область просмотра в нужное место.
2. Вы увидите на диаграмме значок якоря этой области просмотра . Чтобы в дальнейшем изменить свойства этой области, Вам нужно будет выделить этот значок мышью.
3. Перейдите на страницу Основные панели Свойства.
4. В поле Имя : введите zek.
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. Сбросьте флажок исключить, если он был установлен.

#### Переключение между областями просмотра

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

Чтобы перейти к другой области просмотра в режиме создания модели:

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

Чтобы перейти к другой области просмотра в режиме выполнения модели:

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

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

1. В Палитре выделите Презентация. Перетащите элемент `text`, разместите и введите в поле Текст: Цех, как на [рис. 2.9](#).
2. Перетащите второй элемент `text`, разместите и введите в поле Текст: Пункт окончательного контроля.
3. На панели Свойства выделите Динамические и в поле Действие по щелчку: введите следующий Java код:

```
на_контроль.kontr.navigateTo();
```

Во введённом коде `на_контроль` - имя элемента нового класса активного объекта `Kontrol`, а `kontr` - имя области просмотра, которую мы создадим позднее на новом активном объекте `Kontrol`.

На [рис. 2.10](#) показан активный объект `Kontrol` с размещёнными на

нём тремя сегментами модели. Создадим эти сегменты.

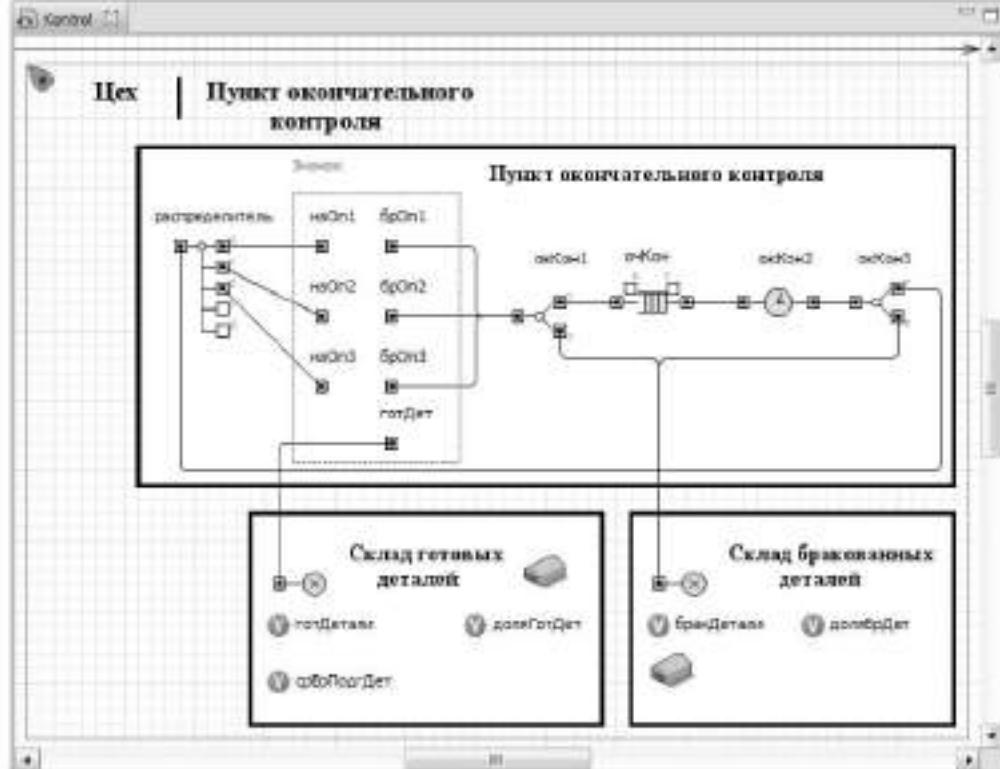


Рис. 2.10. Объект Kontrol с размещёнными на нём тремя сегментами

#### Пункт окончательного контроля

1. Из Презентации перетащите три элемента Прямоугольник и разместите так, как на рис. 2.10. На странице Дополнительные панели Свойства для верхнего прямоугольника введите в поле Ширина: 580, в поле Высота: 240. Для нижних прямоугольников: 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. 2.10).
3. Из Основной библиотеки (Enterprise Library) перетащите два объекта selectOutput, объект queue, объект delay и один

объект `selectOutput5` на диаграмму класса `Kontrol`, разместите в верхнем прямоугольнике и соедините так, как показано на [рис. 2.10](#). Порты `брOp1`, `брOp2` и `брOp3` соединяются с входом объекта `selectOutput`. Выход `T (true)` объекта `окKon3` соединяется с входом объекта `selectOutput`.

- Выделите поочередно объекты, начиная с левого объекта `selectOutput`, и на странице Основные панели Свойства установите свойства согласно [рис. 2.10](#) и [табл. 2.5](#). Во всех объектах должен быть установлен флажок Отображать имя и На презентации.

Таблица 2.5.

Объект	Свойства	Значения
<code>selectOutput</code>	имя	<code>окKon1</code>
	Класс заявки	<code>Detail</code>
	Выход <code>true</code> выбирается	При выполнении условия
	Условие	<code>entity.a &lt; 2</code>
<code>queue</code>	имя	<code>очKon</code>
	Класс заявки	<code>Detail</code>
	Максимальная вместимость	Установите флажок
<code>delay</code>	имя	<code>окKon2</code>
	Класс заявки	<code>Detail</code>
	Задержка задаётся	Явно
<code>selectOutput</code>	Время задержки	<code>exponential (1/get_Main().Tk)</code>
	Вместимость	1
<code>selectOutput</code>	имя	<code>окKon3</code>
	Класс заявки	<code>Detail</code>
	Выход <code>true</code> выбирается	С заданной вероятностью
	Вероятность [0..1]	<code>1-get_Main().q4</code>

selectOut при 5	Действие при выходе(true)	entity.a=2
	имя	распределитель
	Класс заявки	Detail
	Использовать:	Условия
	Условие 0	entity.n==1
	Условие 1	entity.n==2
	Условие 2	entity.n==3

Склад готовых деталей. Вывод результатов моделирования

1. Из библиотеки Основная перетащите на левый нижний прямоугольник три элемента Простая переменная. На странице Основные панели Свойства в поле Имя: каждого элемента введите соответствующие имена, показанные на [рис. 2.10](#). Установите Тип: double.
2. Из Основной библиотеки перетащите объект sink.
3. На странице Основные панели Свойства установите следующие свойства:
  - Имя: склГотДет
  - Отображать имя сбросьте флажок;
  - Класс заявки: Detail
  - Действие при входе

готДетали = склГотДет.count()/get\_Main().колПрог;  
 доляГотДет = готДетали/(готДетали + бракДетали);  
 срВрПодгДет = (get\_Main().врМод\*get\_Main().колПрог)/склГс

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. 2.10](#).

1. Из библиотеки Основная перетащите на правый нижний прямоугольник два элемента Простая переменная. На странице Основные панели Свойства в поле Имя: каждого элемента введите соответствующие имена, показанные на рис. 2.10. Установите Тип: double.
2. Из Enterprise Library перетащите объект sink.
3. На странице Основные панели Свойства установите следующие свойства:
  - Имя: склБракДет
  - Отображать имя сбросьте флагок;
  - Класс заявки: Detail
  - Действие при входе

бракДетали = склБракДет.соин()/.get\_Main().колПрог;  
доляБрДет = бракДетали/(готДетали+бракДетали);

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

4. Из библиотеки Картинки перетащите картинку Склад и разместите как на рис. 2.10.

Так как все исходные данные размещены на диаграмме класса Main, то ссылка на них из диаграммы класса Kontrol, производится, например, так: get\_Main().колПрог;

#### 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 Дополнительные панели Свойства введите в поля X : 30, Y : 10, Ширина: 670, в поле Высота: 480.
7. Сбросьте флагок Исключить, если он был установлен.
8. Перетащите элемент `text`, разместите и введите в поле Текст: Цех, как на [рис. 2.10](#).
9. На панели Свойства выделите Динамические и в поле Действие по щелчку: введите следующий Java код:

```
get_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.

## Проведение исследований в AnyLogic

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

1. Из Презентации перетащите элемент Скругленный прямоугольник на диаграмму класса `Kontrol` и разместите как на [рис. 2.11](#). Можно было бы всё это сделать на диаграмме класса `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 Простая

переменная. Разместите и дайте имя коэфИспПодЗаг, как на рис. 2.11. Оставьте тип double.

4. Перетащите ещё один элемент Простая переменная и дайте ему имя коэфИспВыпОп1. Оставьте тип double.
5. Сююрийте элемент с именем коэфИспВыпОп1. Вставьте ещё два таких элемента, которым системой будут присвоены имена коэфИспВыпОп2 и коэфИспВыпОп3 (см. рис. 2.11).
6. Выделите объект подЗаг. В поле Действие при выходе добавьте код:

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

7. Выделите объект выпОп1. В поле Действие при выходе введите код:

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

8. Выделите объект выпОп2. В поле Действие при выходе введите код:

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

9. Выделите объект выпОп3. В поле Действие при выходе введите код:

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

10. Перейдите на диаграмму класса Main. В панели Проекты выделите Simulation:Main.

В AnyLogic начальное число генератора случайных чисел устанавливается один раз перед запуском модели.

Нужно в панели Проект щёлкнуть Эксперимент. На странице Основные установить Фиксированное начальное число (воспроизводимые прогоны). В соответствующее поле

ввести начальное число.

В GPSS World при обращении к датчикам случайных чисел (в нашей модели распределенных по экспоненциальному закону) можно указывать различные начальные числа, или те же, что и в GPSS World.

Чтобы не связывать разработчиков моделей выполнением условия одинаковости начальных чисел датчиков случайных чисел, в AnyLogic используем опцию Случайное начальное число (уникальные прогоны), а в GPSS-модели произвольным образом будем устанавливать различные начальные числа во всех обращениях к генераторам случайных чисел в каждом эксперименте. Это вполне корректно, так как в [1, 2, 3, 4] показано, что и при условии неодинаковости начальных чисел генераторов тенденция изменения количественных показателей, а значит и величины различия результатов моделирования в GPSS World и AnyLogic практически одни и те же.

11. На странице Модельное время в поле Остановить: введите 7987 680.0 ( $480 * 16\ 641 = 7\ 987\ 680.0$ ).

## Интерпретация результатов моделирования

Мы провели эксперименты с моделями прямых задач, выполнив в каждом эксперименте в GPSS World 16 641 прогонов, а в AnyLogic увеличив модельное время в 16 641 раз.

Всего выполнено 8 экспериментов, результаты которых сведены в [табл. 2.6](#). Первый эксперимент соответствует постановке задачи. Результаты первого эксперимента представлены на [рис. 2.11](#) 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  $T_p = 35$  до  $T_p = 30$ , 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 AnyLogic, 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 0,010 ... 0,185.

Коэффициенты 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 9,885 (9,882), а среднее время изготовления одной детали 48,559 (48,573), то в последнем восьмом эксперименте цель была достигнута - 21,161 (21,250) и 22,683 (22,588) соответственно.

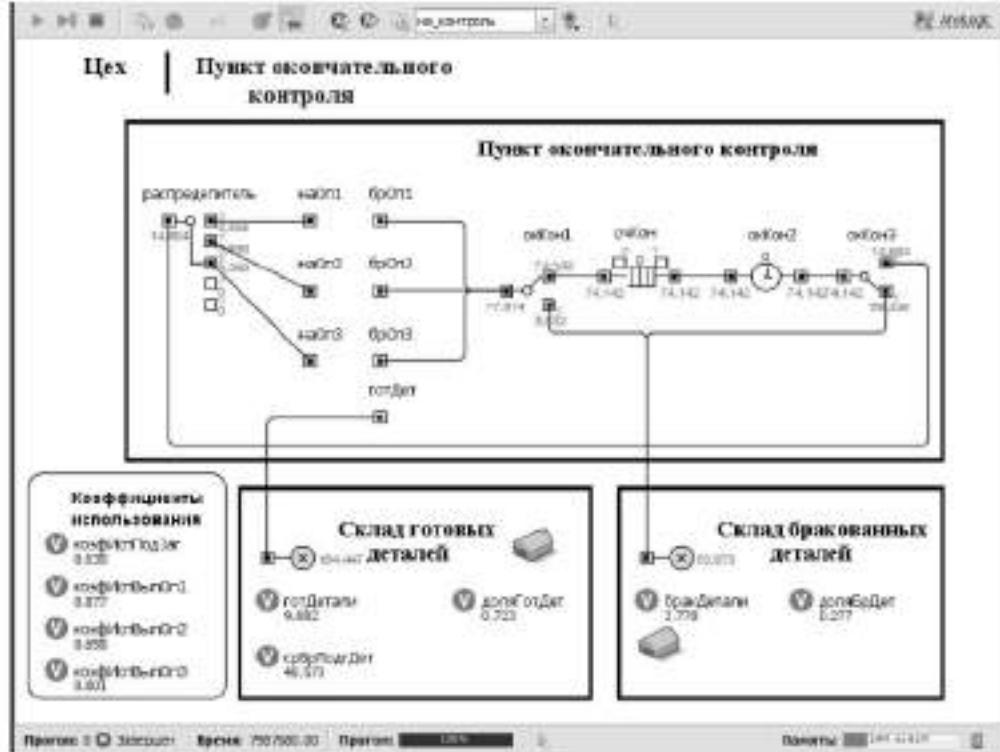


Рис. 2.11. Вариант результатов моделирования

Таблица 2.6. Показатели функционирования цеха

Показатели	GPSS World	AnyLogic
<b>1) согласно постановке задачи</b>		
гот.детали	9,885	9,882
долг.отдат	0,721	0,723
брак.детали	3,778	3,778
долг.бр.дет	0,279	0,277
срвр.подг.дет	48,559	48,573
коэф.исп.подзаг	0,639	0,638
коэф.исп.вып.оп1	0,879	0,877
коэф.исп.вып.оп2	0,662	0,658
коэф.исп.вып.оп3	0,801	0,801
Δ долг.отдат	0,002	

$\Delta$ доляБрДет	0,002
$\Delta$ срВрПодгДет	0,014
2) Tn = 30	
готДетали	11,284
доляГотДет	0,722
бракДетали	4,338
доляБрДет	0,278
срВрПодгДет	42,538
коэфИспПодЗаг	0,749
коэфИспВыпОп1	1,000
коэфИспВыпОп2	0,752
коэфИспВыпОп3	0,915
$\Delta$ доляГотДет	0,001
$\Delta$ доляБрДет	0,001
$\Delta$ срВрПодгДет	0,185
3) T1 = 25, T3 = 30	
готДетали	11,552
доляГотДет	0,722
бракДетали	4,459
доляБрДет	0,278
срВрПодгДет	41,553
коэфИспПодЗаг	0,748
коэфИспВыпОп1	0,852
коэфИспВыпОп2	0,774
коэфИспВыпОп3	0,802
$\Delta$ доляГотДет	0,002
$\Delta$ доляБрДет	0,002
$\Delta$ срВрПодгДет	0,011
4) Tn = 25, T1 = 20, T2 = 15, T3 = 25	
готДетали	13,845
доляГотДет	0,723

бракДетали	5,314	5,305
доляБрДет	0,277	0,276
срВрПодгДет	34,669	34,532
коэфИспПодЗаг	0,895	0,899
коэфИспВыпОп1	0,817	0,822
коэфИспВыпОп2	0,555	0,558
коэфИспВыпОп3	0,801	0,803
Δ доляГотДет	0,001	
Δ доляБрДет	0,001	
Δ срВрПодгДет	0,037	
5) Тn = 20, Тз = 20		
готДетали	15,453	15,499
доляГотДет	0,723	0,724
бракДетали	5,932	5,915
доляБрДет	0,277	0,276
срВрПодгДет	31,063	30,969
коэфИспПодЗаг	1,000	1,000
коэфИспВыпОп1	0,911	0,913
коэфИспВыпОп2	0,619	0,621
коэфИспВыпОп3	0,716	0,718
Δ доляГотДет	0,001	
Δ доляБрДет	0,001	
Δ срВрПодгДет	0,094	
6) q3 = 0,05		
готДетали	16,200	16,206
доляГотДет	0,755	0,757
бракДетали	5,246	5,1921
доляБрДет	0,245	0,243
срВрПодгДет	29,629	29,619
коэфИспПодЗаг	1,000	1,000
коэфИспВыпОп1	0,916	0,911

коэфИспВыпОп2	0,622	0,620
коэфИспВыпОп3	0,710	0,711
Δ доляБрДет	0,002	
Δ доляГотДет	0,002	
Δ срВрПодгДет	0,010	
7) q1 = 0,05, q2 = 0,05		
готДетали	18,903	18,856
доляГотДет	0,883	0,883
бракДетали	2,504	2,491
доляБрДет	0,117	0,117
срВрПодгДет	25,393	25,456
коэфИспПодЗаг	1,000	1,000
коэфИспВыпОп1	0,901	0,896
коэфИспВыпОп2	0,650	0,646
коэфИспВыпОп3	0,828	0,829
Δ доляГотДет	0	
Δ доляБрДет	0	
Δ срВрПодгДет	0,059	
8) T1=15, срВрПодгЗаг={7,11,18,19,23,22,0,0,0,0}		
готДетали	21,161	21,250
доляГотДет	0,883	0,884
бракДетали	2,801	2,785
доляБрДет	0,117	0,116
срВрПодгДет	22,683	22,588
коэфИспПодЗаг	0,939	0,940
коэфИспВыпОп1	0,756	0,760
коэфИспВыпОп2	0,725	0,729
коэфИспВыпОп3	0,929	0,932
Δ доляГотДет	0,001	
Δ доляБрДет	0,001	
Δ срВрПодгДет	0,095	

Но уже по второму эксперименту видно, что коэфИспВыпОп1=1, а коэфИспВыпОп3=0,916, то есть приближается 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.

## Модель функционирования направления связи

### 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  $T_1 = 3$  мин и  $T_2 = 4$  мин. При нормальной работе сообщения передаются по основному каналу. Время передачи одного сообщения распределено по экспоненциальному закону со средним значением  $T_3 = 2$  мин.

В основном канале происходят сбои через интервалы времени, распределенные по экспоненциальному закону со средним значением  $T_4 = 15$  мин. Если сбой происходит во время передачи, то сообщение теряется. За время  $T_5 = 5$  с запускается резервный канал, который передает сообщения, начиная с очередного. Время передачи одного сообщения распределено по экспоненциальному закону со средним значением  $T_6 = 3$  мин.

Основной канал восстанавливается. Время восстановления канала подчинено экспонциальному закону со средним значением  $T_7 = 2$  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.

### Модель направления связи в GPSS World

В модели сообщения следует представлять транзактами, основной и резервный канал - одноканальными устройствами (ОКУ), входной буфер (накопитель) - списком пользователя. В списке пользователя следует использовать дисциплину обслуживания FIFO.

Для ввода исходных данных целесообразно использовать переменные пользователя. В этом случае можно проводить при необходимости встроенными средствами GPSS World дисперсионный и оптимизирующий эксперименты.

Так как сообщения имеют одинаковые приоритеты, то для моделирования ОКУ нужно использовать блоки SEIZE и RELEASE. Моделирование отказов основного канала нужно произвести блоками FUNAVAIL и FAVAIL, а не блоками PREEMPT и RETURN в режиме абсолютного захвата. Тогда статистика ОКУ не будет искажена.

Введем масштабирование: 1 единица модельного времени соответствует 1 с, то есть, например, время моделирования равно 2 часам, тогда  $2 \cdot 60 \cdot 60 = 7200$  единиц модельного времени. Аналогично  $T_1 = 120$ ,  $T_2 = 240$  и т.д.

Декомпозиция системы и состав сегментов модели определяются разработчиком. Введем следующие сегменты:

- ввода исходных данных и описания арифметических выражений;
- имитации сообщений потока 1;
- имитации сообщений потока 2;
- имитации работы буфера и основного канала;
- имитации работы резервного канала;
- имитации выхода из строя основного канала;
- задания времени моделирования и вычисления результатов моделирования.

Ниже приводится программа модели.

```
; Модель функционирования направления связи
;Задание исходных данных
Emk EQU 5 ; Емкость накопителя
VrMod EQU 7200 ; Время моделирования
```

T1 EQU 180 ; Среднее время поступления сообщений потока 1  
 T2 EQU 240 ; Среднее время поступления сообщений потока 2  
 T3 EQU 120 ; Среднее время передачи по OsnK  
 T4 EQU 900 ; Средний интервал времени выхода из строя OsnK  
 T5 EQU 10 ; Время включения Resk  
 T6 EQU 180 ; Среднее время передачи по ResK  
 T7 EQU 120 ; Среднее время восстановления OsnK  
 ; Описание арифметических выражений  
 ; Вероятность передачи сообщений потока 1  
 Ver1 VARIABLE (N\$Term1+N\$Term3)/N\$Soob1  
 ; Вероятность передачи сообщений потока 2  
 Ver2 VARIABLE (N\$Term2+N\$Term4)/N\$Soob2  
 ; Вероятность передачи сообщений потоков 1 и 2  
 Ver VARIABLE (V\$Ver1+V\$Ver2)/2  
 ; Сегмент имитации сообщений потока 1  
 GENERATE (Exponential(12,0,T1)) ; Генератор сообщений потока 1  
 Soob1 ASSIGN 1,1 ; Код 1 в Р1 - сообщения потока 1  
 TRANSFER ,Nakop ; Направить на OsnK  
 ; Сегмент имитации сообщений потока 2  
 GENERATE (Exponential(15,0,T2)) ; Генератор  
 сообщений потока 2  
 Soob2 ASSIGN 1,2 ; Код 2 в Р1 - сообщения потока 2  
 ; Сегмент имитации работы накопителя и OsnK  
 Nakop GATE FV OsnK,KRes ; Доступен ли OsnK? Если нет, на Resk  
 GATE NU OsnK,Spis ; Свободен ли OsnK? Если  
 нет, в накопитель  
 Prov3 SEIZE OsnK ; Занять OsnK  
 ADVANCE (Exponential(11,0,T3)) ; Обслуживание  
 RELEASE OsnK ; Освободить OsnK  
 UNLINK Nak,Prov3,1 ; Вывод из накопителя  
 одного транзакта на OsnK  
 TEST E P1,1,Term2 ; Сообщение потока 1 или  
 потока 2 передано по OsnK?  
 Term1 TERMINATE ; Счет переданных сообщений потока 1 по OsnK  
 Term2 TERMINATE ; Счет переданных сообщений потока 2 по OsnK  
 ; Список пользователя Nak  
 Spis TEST L CHSNak,Emk,Term7 ; Есть ли место в накопителе?  
 LINK Nak,FIFO ; Если да, поместить  
 сообщение в накопитель

Term7 TEST E P1,1,Term6 ; Сообщение потока 1 или потока 2 было пс  
 Term5 TERMINATE ; Счет потерянных сообщений потока 1  
 Term6 TERMINATE ; Счет потерянных сообщений потока 2  
 ; Сегмент имитации работы Resk  
 KRes GATE NU ResK,Spis ; Свободен ли Resk? Нет, сообщение в нах  
     TEST E Kont,1,Prov1 ; Включить Resk  
 ADVANCE T5 ; Включение Resk  
 SAVEVALUE Kont,0  
 Prov1 SEIZE ResK ; Занять Resk  
 ADVANCE (Exponential(12,0,T6)); Передача  
 RELEASE ResK ; Освободить Resk  
 GATE FNV OsnK,Prov2 ; Доступен ли OsnK?  
 UNLINK Nak,Prov1,1 ; Если нет, из буфера  
 сообщение на Resk  
 Prov2 TEST E P1,1,Term4 ; Сообщение потока 1 или 2 передано по Res  
 Term3 TERMINATE ; Счет переданных сообщений потока 1  
 Term4 TERMINATE ; Счет переданных сообщений потока 2  
 ; Сегмент имитации выхода из строя OsnK  
     GENERATE „,1  
 Term8 ADVANCE (Exponential(12,0,T4)); Расчет времени до следующего  
     FUNAVAIL OsnK ; Выход из строя OsnK  
     SAVEVALUE Kont,1  
 ASSIGN 1,(Exponential(12,0,T7)); Расчет  
 времени восстановления OsnK  
 ADVANCE P1 ; Имитация восстановления OsnK  
 SAVEVALUE VrOtk+,P1 ; Учет времени восстановления OsnK  
 FAVAIL OsnK ; OsnK в доступное состояние  
 UNLINK Nak,Prov3,1 ; Сообщение на OsnK  
     TRANSFER ,Term8  
 ; Сегмент задания времени моделирования  
 ; и вычисления результатов моделирования  
     GENERATE VrMod  
 TEST L X\$Prog,TG1,Met3  
     SAVEVALUE Prog,TG1  
 Met3 TEST E TG1,1,Met4  
 SAVEVALUE Ver1,V\$Ver1 ; Вероятность передачи  
 сообщений потока 1  
     SAVEVALUE Ver2,V\$Ver2 ; Вероятность передачи сообщений поток  
     SAVEVALUE Ver,V\$Ver ; Вероятность передачи сообщений напр

```
SAVEVALUE VOtk,(1-V$Ver) ; Вероятность отказа в передаче сооѓ
SAVEVALUE VerOtk,((AC1-X$VrOtk)/AC1) ; Вероятность безотказн
Met4 TERMINATE 1
START 10000
```

Для определения вероятности безотказной работы суммируется в ячейке *X\$VrOtk* время отказов направления связи, которое затем вычитается из абсолютного модельного времени *AC1*, а полученная разность делится на *AC1*.

## Модель направления связи в AnyLogic

Поскольку методика построения модели в AnyLogic существенным образом отличается от методики построения в GPSS World, выделим в модели функционирования направления связи следующие сегменты:

- исходные данные;
- источники сообщений;
- буфер, основной и резервный каналы связи;
- имитатор отказов основного канала;
- результаты моделирования.

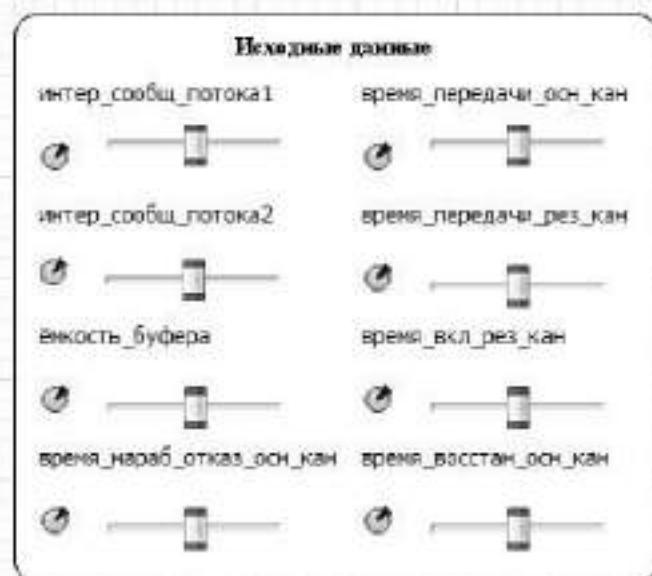
## Исходные данные

Для ввода исходных данных используем элементы Параметр и Бегунок. В данной модели мы оставим элемент Бегунок, хотя знаем, что можно обойтись и без него.

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. Щелкните кнопку Далее.
3. На открывшейся второй странице Мастера создания модели выберите Начать создание модели "с нуля". Щелкните кнопку Далее.

4. Полагаем, что все сегменты модели мы сможем разместить так, что они будут видны в ходе работы модели. В Палитре выделите Презентация.
5. Перетащите элемент Скругленный прямоугольник в нужное место для размещения элементов исходных данных.
6. Перетащите элемент *text* и на странице Основные панели Свойства в поле Текст: введите Исходные данные.
7. В Палитре выделите Основная. Перетащите элементы Параметр на элемент с именем Исходные данные. Разместите их и дайте имена так, как показано на [рис. 3.1](#). Значения свойств установите согласно [табл. 3.1](#).
8. В Палитре выделите Элементы управления. Перетащите элементы Бегунок на элемент с именем Исходные данные. Разместите их так, как показано на [рис. 3.1](#). Значения свойств установите согласно [табл. 3.2](#).

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	double	180
интер_сообщ_потока2	double	240
ёмкость_буфера	int	5
время_передачи_осн_кан	double	120
время_передачи_рез_кан	double	180
время_вкл_рез_кан	double	10
время_нараб_отказ_осн_кан	double	900
время_восстан_осн_кан	double	120

Таблица 3.2.

Бегунок		
Связать с	Минимальное значение	Максимальное значение
интер_сообщ_потока1	1	300
интер_сообщ_потока2	1	300
ёмкость_буфера	1	25
время_передачи_осн_кан	1	200
время_передачи_рез_кан	1	300
время_вкл_рез_кан	0,1	20
время_нараб_отказ_осн_кан	1	2000
время_восстан_осн_кан	1	500

## Вывод результатов моделирования

Для вывода результатов моделирования используем элемент Простая переменная.

1. В Палитре выделите Презентация. Перетащите элемент Скруглённый прямоугольник для размещения элементов Простая переменная.
2. Перетащите элемент text и на странице Основные панели

Свойства в поле Текст: введите Результаты моделирования.

3. В Палитре выделите Основная. Перетащите элементы Простая переменная. Разместите их и дайте им имена так, как показано на [рис. 3.2](#). Тип всех переменных double, кроме переменной - текущей емкости буфера. Её тип - int.

## 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. Перетащите также три элемента text и на странице Основные панели Свойства в поле Текст: каждого элемента введите названия элементов, показанные на [рис. 3.3](#).

### Результаты моделирования

всего_сообщ_поступило	поступило_сообщ_потока1	поступило_сообщ_потока2
всего_потеряно_сообщ	потеряно_сообщ_потока1	потеряно_сообщ_потока2
всего_передано_сообщ	передана_сообщ_потока1	передано_сообщ_потока2
вероятность_передачи_сообщ		текущая_емкость_буфера
вероятность_передачи_сообщ_потока1		вероятность_потери_сообщ
вероятность_передачи_сообщ_потока2		коэф_использов_осн_кан

Рис. 3.2. Размещение элементов для вывода результатов моделирования

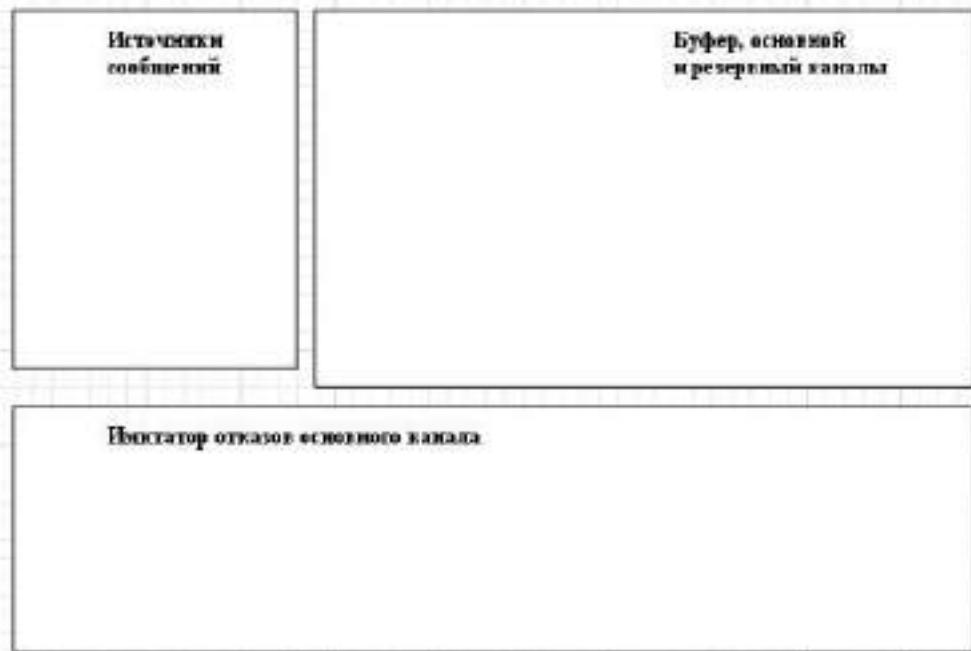


Рис. 3.3. Элементы для размещения сегментов событийной части

### 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 Enterprise Library.
2. Перетащите два объекта source на диаграмму класса Main и разместите в прямоугольнике с именем Источники сообщений.
3. Для записи и хранения параметров сообщений в дополнительные поля заявок нужно создать нестандартный класс заявки. Создайте класс заявки Message.
4. В панели Проект щелкните правой кнопкой мыши элемент модели верхнего уровня дерева и выберите Создать Java

класс.

- Появится диалоговое окно Новый Java класс. В поле Имя: введите имя нового класса *Message*.
- В поле Базовый класс: выберите из выпадающего списка *Entity* в качестве базового класса. Щелкните кнопку Далее.
- 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 numPort;
```

- Оставьте выбранными флагки Создать конструктор и Создать метод *toString()*.
- Щелкните кнопку Готово. Появится редактор кода и автоматически созданный код вашего Java класса. Закройте код.
- Выделите последовательно объекты *source*. На странице Основные панели Свойства установите как в табл. 3.3.

### Буфер, основной и резервный каналы

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

- В Палитре выделите Основная библиотека.
- Перетащите на диаграмму класса *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
	Отображать имя	Установите флагок
	Класс заявки	<i>Message</i>
	Заявки прибывают согласно	Времени между прибытиями

	Время между прибытиями	<code>exponential(1/ интер_сообщ_потока1)</code>
Поток_1	Количество заявок, прибывающих за один раз	1
	Новая заявка	<code>new Message() entity.numPotok = 1; поступило_сообщ_потока1 ++;</code>
	Действие при выходе	<code>всего_сообщ_поступило ++;</code>
	Отображать имя	Установите флагок
	Класс заявки	<code>Message</code>
	Заявки прибывают согласно	Времени между прибытиями
	Время между прибытиями	<code>exponential(1/ интер_сообщ_потока2)</code>
Поток_2	Количество заявок, прибывающих за один раз	1
	Новая заявка	<code>new Message() entity.numPotok = 2; поступило_сообщ_потока2 ++;</code>
	Действие при выходе	

- Эти объекты вам известны, кроме объекта `hold` из библиотеки Enterprise Library. Он блокирует/разблокирует поток заявок на определенном участке блок-схемы. Если объект находится в заблокированном состоянии, то заявки не будут поступать на его входной порт, и будут ждать, пока объект не будет разблокирован. Состоянием объекта можно управлять программно с помощью метода `setBlocked()`. Блокирует входной порт, если в качестве значения аргумента передано `true`, и разблокирует его при передаче аргумента `false`.
- Метод `isBlocked()` возвращает `true`, если входной порт заблокирован. Если порт не заблокирован - возвращает `false`.

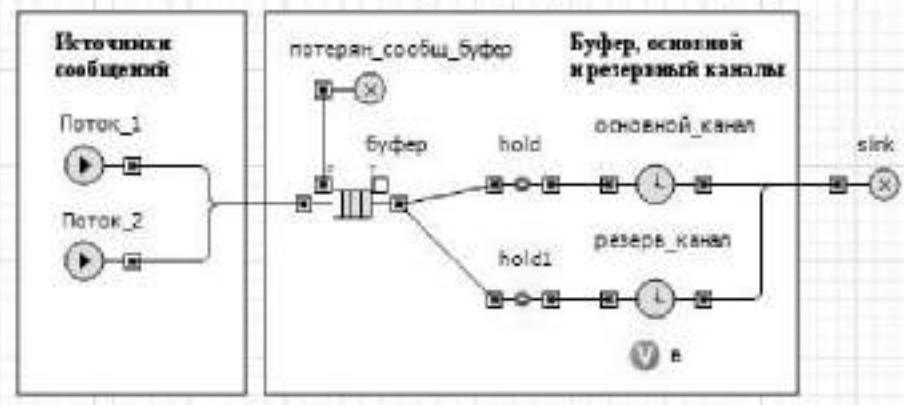


Рис. 3.4. Объекты двух сегментов модели

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

Таблица 3.4.

Свойства	Значение
Имя	Буфер
Отображать имя	Установить флажок
Класс заявки	Message
Вместимость	ёмкость_буфера
Действие при входе	текущая_емкость_буфера++;
Действие при выходе	текущая_емкость_буфера--;
Разрешить вытеснение	Установить флажок
Имя	hold1
Отображать имя	Установить флажок
Класс заявки	Message
Изначально заблокирован	Установить флажок

Имя	hold
Класс заявки	Message
Имя	потер_сообщ_буфер
Отображать имя	Установить флагок
Класс заявки	Message
Действие при входе	<pre>if(entity.numPotok == 1) потеряно_сообщ_потока1++; if(entity.numPotok == 2) потеряно_сообщ_потока2++; всего_потеряно_сообщ++;</pre>
Имя	основной_канал
Отображать имя	Установить флагок
Класс заявки	Message
Задержка задается	Явно
Время задержки	$\text{exponential}(1/\text{время\_передачи\_осн\_кан})$
Вместимость	1
Включить сбор статистики	Установить флагок
Имя	резерв_канал
Отображать имя	Установить флагок
Класс заявки	Message
Вместимость	1
Задержка задается	Явно
Время задержки	$\text{exponential}(1/\text{время\_передачи\_рез\_кан})$

Вместимость	1
Действие при входе	<pre> if (a==0)     в = время_передачи_рез_кан; if (a==1)     {в = время_передачи_рез_кан +      время_вкл_рез_кан;     a=0;} </pre>
Действие при выходе	<pre> if (hold.isBlocked()== true)     hold1.setBlocked(false); </pre>
Включить сбор статистики	Установить флагок
Имя	sink
Отображать имя	Установить флагок
Класс заявки	Message
Действие при входе	<pre> if (entity.numPotok == 1)     (передано_сообщ_потока1++;     вероятность_передачи_сообщ_потока1 =     передано_сообщ_потока1 /поступило_сообщ_потока1;} if (entity.numPotok == 2)     (передано_сообщ_потока2++;     вероятность_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; </pre>

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 объекты, соедините их как на [рис. 3.5](#).
2. Последовательно выделите и установите свойства объектов согласно [табл. 3.5](#).

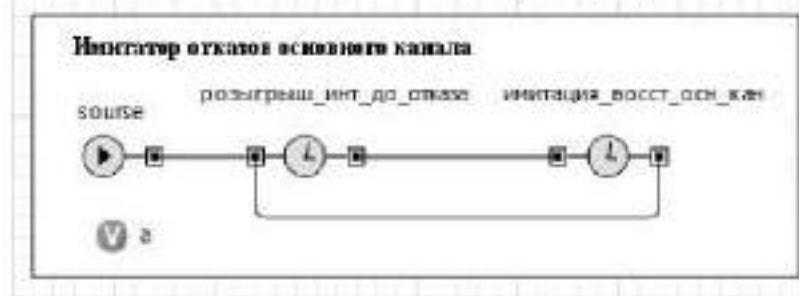


Рис. 3.5. Сегмент имитации отказов основного канала связи

Таблица 3.5.

Свойства	Значение
Имя	source
Отображать имя	Установить флагок
Класс заявки	Entity
Заявки прибывают согласно	Интенсивности
Интенсивность прибытия	1
Ограниченнное количество прибытий	Установить флагок
Количество заявок, прибывающих за один раз	1
Имя	розыгрыш_инт_до_отказа
Отображать имя	Установить флагок
Класс заявки	Entity
Задержка задается	Явно
Время задержки	exponential (1/время_нараб_отказ_осн_кан)
Вместимость	1
Действие при выходе	<pre> hold.setBlocked(true); if (основной_канал.size()!=0) {основной_канал.remove((Message) основной_канал.get(0)); всего_потеряно_сообщ++; } </pre>

	hold1.setBlocked(false);
Включить сбор статистики	Установить флагок имитация_восст_осн_кан
Имя	Entity
Класс заявки	Entity
Задержка задается	Явно
Время задержки	exponential (1/время_восстан_осн_кан )
Вместимость	1
Действие при выходе	hold.setBlocked(false); hold1.setBlocked(true);
Включить сбор статистики	Установить флагок

Обратим внимание на переменные *a* и *v*, 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.

## Интерпретация результатов моделирования

Модели построены. Тем не менее, для удобства чтения результатов моделирования сделайте следующие дополнения.

1. Из палитры Основная перетащите три элемента Простая переменная на диаграмму класса Main в скруглённый прямоугольник с именем Результаты моделирования.
2. Разместите их в один столбец с простыми переменными вероятность\_потери\_сообщ, коэф\_использ\_осн\_кан (см. [рис. 3.2](#)).
3. Выделите первый элемент Простая переменная. В поле имя введите коэф\_использ\_осн\_кан. Оставьте тип double.
4. Выделите второй элемент. В поле имя введите коэф\_использ\_резерв\_кан. Оставьте тип double.
5. Выделите третий элемент. В поле имя введите коэф\_использ\_резерв\_кан. Оставьте тип double.
6. Выделите второй элемент Параметр. В поле имя введите коэф\_безотк\_раб\_осн\_кан. Оставьте тип double.
7. Выделите объект основной\_канал. В поле Действие при выходе введите код:

```
коэф_использ_осн_кан=
основной_канал.statsUtilization.mean()
```

8. Выделите объект резерв\_канал. В поле Действие при выходе к имеющемуся там коду добавьте внизу строку:

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

9. Выделите объект имитация\_восст\_осн\_кан. В поле Действие при выходе к имеющемуся там коду добавьте строку:

```
коэф_безотк_раб_осн_кан=
1-имитация_восст_осн_кан.statsUtilization.mean()
```

## 10. Во всех объектах установите флагок Сбор статистики.

Теперь вы сможете снимать значения этих коэффициентов, не отыскивая каждый раз нужные объекты и не используя при этом окна инспектора. Проведите моделирование в GPSS World и AnyLogic 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 10 000 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](#) в первых трёх экспериментах вероятность передачи сообщений отличается на 0,015 ... 0,022. В следующих пяти экспериментах вероятности передачи сообщений, полученные в GPSS World и AnyLogic, отличаются на 0,002 ... 0,006, то есть на порядок меньше. Причём во всех экспериментах это отличие в большую сторону в AnyLogic-модели на  $\Delta_1 \dots \Delta_5$ .

**Таблица 3.6. Показатели функционирования направления связи**

Показатели	GPSS World	AnyLogic
1) объем_буфера = 5		
вероятность_передачи_сообщ	0,752	0,773
вероятность_передачи_сообщ_потока1	0,752	0,772
вероятность_передачи_сообщ_потока2	0,753	0,773
$\Delta_1$ вероятности_передачи_сообщ	$\Delta_1 = 0,022$	
вероятность_потери_сообщ	0,248	0,227
коэф_использ_осн_кан	0,777	0,757
коэф_использ_рез_кан	0,152	0,222
сум_коэф_использ_кан	0,929	0,979
2) объем_буфера = 10		
вероятность_передачи_сообщ	0,829	0,861
вероятность_передачи_сообщ_потока1	0,829	0,861
вероятность_передачи_сообщ_потока2	0,829	0,862
$\Delta_2$ вероятности_передачи_сообщ	$\Delta_2 = 0,032$	
вероятность_потери_сообщ	0,171	0,139
коэф_использ_осн_кан	0,861	0,841
коэф_использ_рез_кан	0,157	0,25
сум_коэф_использ_кан	1,018	1,091
3) интер_сообщ_потока1 = 90, интер_сообщ_потока2 = 120		
вероятность_передачи_сообщ	0,438	0,454
вероятность_передачи_сообщ_потока1	0,438	0,454
вероятность_передачи_сообщ_потока2	0,438	0,453
$\Delta_3$ вероятности_передачи_сообщ	$\Delta_3 = 0,015$	
вероятность_потери_сообщ	0,562	0,546
коэф_использ_осн_кан	0,882	0,882
коэф_использ_рез_кан	0,209	0,266
сум_коэф_использ_кан	1,091	1,148
4) время_передачи_осн_кан = 60, время_нараб_отказ_осн_кан = 5000		
вероятность_передачи_сообщ	0,844	0,848

вероятность_передачи_сообщ_потока1	0,844	0,848
вероятность_передачи_сообщ_потока2	0,845	0,848
$\Delta_4$ вероятности_передачи_сообщ	$\Delta_1 = 0,004$	
вероятность_потери_сообщ	0,156	0,152
коэф_использ_осн_кан	0,971	0,97
коэф_использ_рез_кан	0,043	0,058
сум_коэф_использ_кан	1,014	1,028

5) время\_передачи\_рез\_кан = 90, время\_восстан\_осн\_кан = 60

вероятность_передачи_сообщ	0,851	0,857
вероятность_передачи_сообщ_потока1	0,851	0,857
вероятность_передачи_сообщ_потока2	0,851	0,857

$\Delta_5$  вероятности\_передачи\_сообщ

$\Delta_1 = 0,006$		
вероятность_потери_сообщ	0,149	0,143
коэф_использ_осн_кан	0,982	0,98
коэф_использ_рез_кан	0,017	0,029
сум_коэф_использ_кан	0,999	1,009

6) интер\_сообщ\_потока1 = 45, интер\_сообщ\_потока2 = 60

вероятность_передачи_сообщ	0,43	0,432
вероятность_передачи_сообщ_потока1	0,431	0,432
вероятность_передачи_сообщ_потока2	0,429	0,431

$\Delta_6$  вероятности\_передачи\_сообщ

$\Delta_2 = 0,002$		
вероятность_потери_сообщ	0,57	0,568
коэф_использ_осн_кан	0,988	0,988
коэф_использ_рез_кан	0,024	0,029
сум_коэф_использ_кан	1,012	1,017

7) время\_передачи\_осн\_кан = 30, время\_передачи\_рез\_кан = 45

вероятность_передачи_сообщ	0,85	0,853
вероятность_передачи_сообщ_потока1	0,851	0,853
вероятность_передачи_сообщ_потока2	0,85	0,853

$\Delta_7$  вероятности\_передачи\_сообщ

$\Delta_3 = 0,003$

вероятность_потери_сообщ	0,15	0,147
коэф_использ_осн_кан	0,982	0,982
коэф_использ_рез_кан	0,015	0,021
сум_коэф_использ_кан	0,997	1,003
8) время_вкл_рез_кан = 1, время_восстан_осн_кан = 30		
вероятность_передачи_сообщ	0,851	0,855
вероятность_передачи_сообщ_потока1	0,851	0,855
вероятность_передачи_сообщ_потока2	0,851	0,855
$\Delta_5$ вероятности_передачи_сообщ	$\Delta_1 = 0,004$	
вероятность_потери_сообщ	0,149	0,145
коэф_использ_осн_кан	0,988	0,987
коэф_использ_рез_кан	0,008	0,015
сум_коэф_использ_кан	0,996	1,002

Коэффициенты использования основного канала в первых двух экспериментах в GPSS World-модели отличаются в большую сторону на 0,02, а в остальных - на 0,0 ... 0,002.

Коэффициент использования резервного канала и в целом суммарный коэффициент использования обоих каналов больше в AnyLogic-модели на 0,005 ... 0,093 и 0,006 ... 0,073 соответственно. Тем не менее, как видно, превышение вероятности передачи сообщений у AnyLogic-модели примерно в два раза ниже, чем превышение суммарного коэффициента использования обоих каналов.

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 0,752 (0,772) до 0,829 (0,861). Уменьшение интервалов (увеличение интенсивности) поступления сообщений потоков 1 и 2 в два раза (90 и 120) снижает вероятности передачи сообщений с 0,829 (0,861) до 0,438 (0,434). 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 0,438 (0,434) до 0,844 (0,848).

Машинное время выполнения модели в обеих системах практически одинаковое и составляет 5...7 сек (в AnyLogic в виртуальном режиме).

# Модель функционирования сети связи

## Модель в AnyLogic

### 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  $T_1, T_2, \dots, T_{n_1}$ .

Сообщения могут быть  $n_2$  категорий с вероятностями их появления  $p_{k1}, p_{k2}, \dots, p_{kn_2}$  ( $p_{k1}, p_{k2}, \dots, p_{kn_2} = 1$ ) и вычислительными сложностями обработки  $S_1, S_2, \dots, S_{n_2}$  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 буфера, 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  $V_{ni}$  бит/с,  $i = \overline{1, k}$ .

ВК и основные каналы связи имеют конечную надёжность. Интервалы времени  $T_{\text{отВК}}$  и  $T_{\text{отK}_1}, T_{\text{отK}_2}, \dots, T_{\text{отK}_n}$  между отказами ВК и каналов связи случайные. Длительности восстановления ВК и каналов связи  $T_{\text{вВК}}$  и  $T_{\text{вK}_1}, T_{\text{вK}_2}, \dots, T_{\text{вK}_n}$  также случайные.

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

## Исходные данные

```

 $n_1 = 6; k = 4;$ 
 $\text{exponential}(T_1) = \text{exponential}(T_2) = \dots = \text{exponential}(T_6) = \text{exponential}(30);$ 
 $n_2 = 4; p_{k1} = 0, 3; p_{k2} = 0, 2; p_{k3} = 0, 2; p_{k4} = 0, 3;$ 
 $Q = 40000 \text{ оп/с}; L_1 = 5000000;$ 
 $\text{normal}(S_1, S_{o1}) = \text{normal}(53000, 6100);$ 
 $\dots$ 
 $\text{normal}(S_3, S_{o3}) = \text{normal}(66000, 7000);$ 
 $\text{normal}(S_4, S_{o4}) = \text{normal}(50000, 500);$ 
 $\text{exponential}(T_{\text{отK}_1}) = \dots = \text{exponential}(T_{\text{отK}_2}) = \text{exponential}(360);$ 
 $\text{exponential}(T_{\text{вK}_1}) = \dots = \text{exponential}(T_{\text{вK}_2}) = \text{exponential}(3.2);$ 
 $\text{exponential}(T_{\text{отВК}}) = \text{exponential}(3600), \text{exponential}(T_{\text{вВК}}) = \text{exponential}(3.7);$ 
 $V_{ni} = 5000 \text{ бит/с}, n_3 = 1, L_{2i} = 250000, i = 1..4.$ 

```

## 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.

## 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](#)).

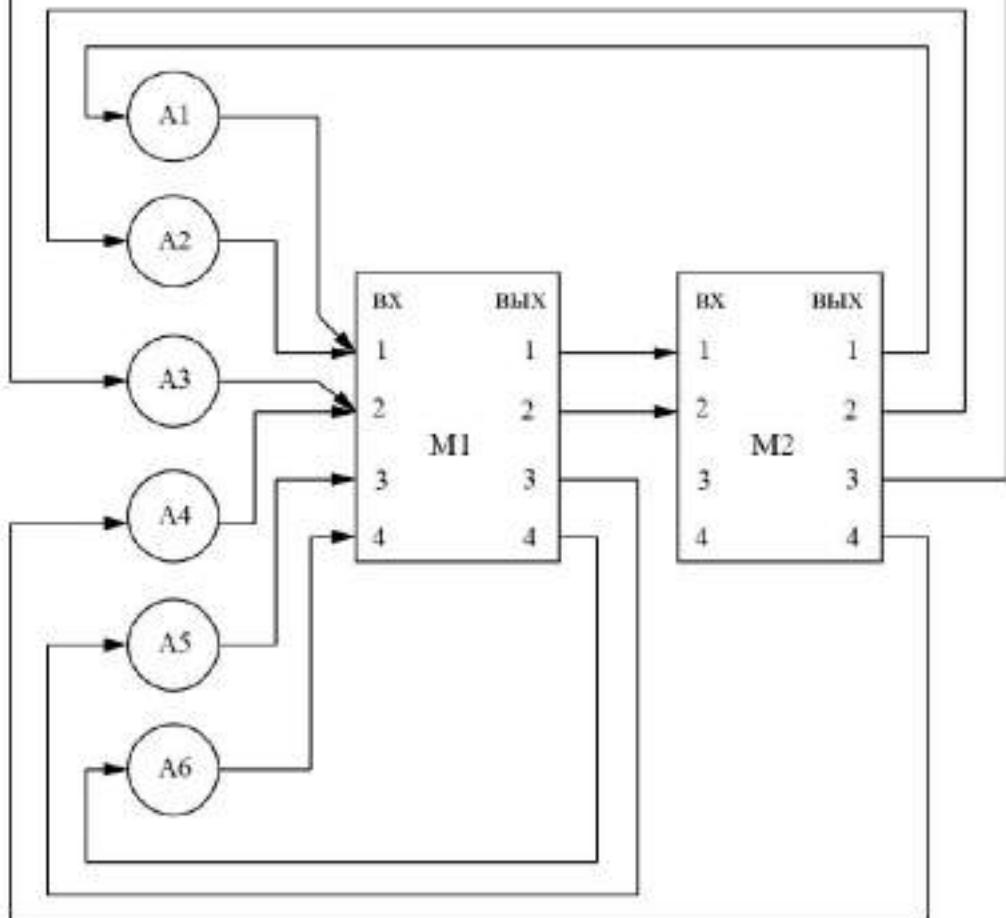


Рис. 4.1. Вариант схемы сети связи

Сообщения абонентов А1 и А2 поступают на вход 1 маршрутизатора М1, абонентов 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](#)).

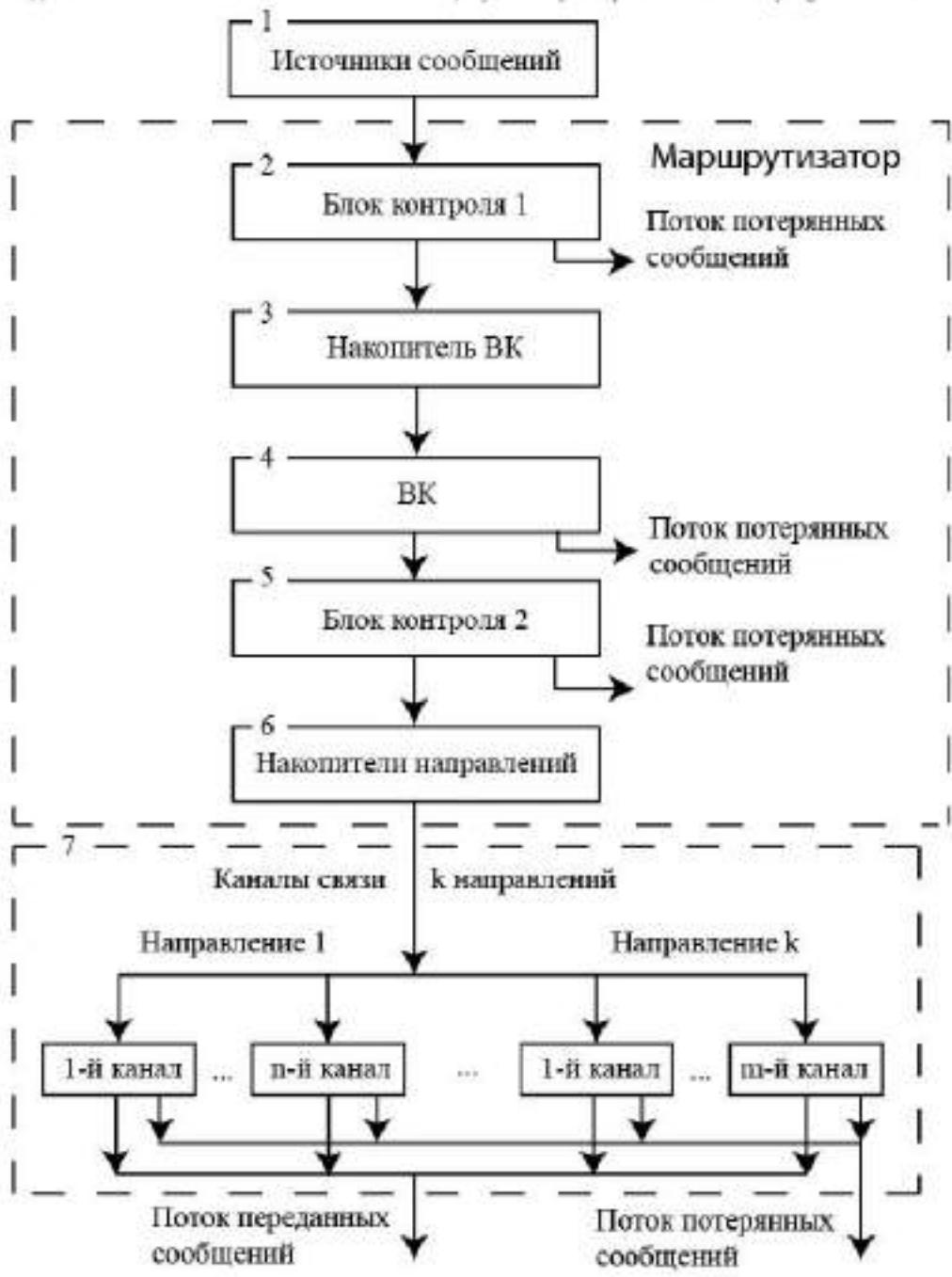


Рис. 4.2. Маршрутизатор как система массового обслуживания

В маршрутизаторе выделены блок контроля 1, накопитель (буфер) ВК,

непосредственно ВК, блок контроля 2, на входе которых 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;

- *skorPerekAn* - среднее время передачи сообщений по каналу связи, бит/с;
- *skorPerekAnR* - 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;
- *всегоОтпрKat1*, *всегоОтпрKat2*, *всегоОтпрKat3*, *всегоОтпрKat4*, *всегоОтпр* - количество отправленных в сети сообщений первой, второй, третьей, четвёртой и всех категорий соответственно;
- *всегоПолKat1*, *всегоПолKat2*, *всегоПолKat3*, *всегоПолKat4*, *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:

- `timeOtkVK` - среднее время между отказами ВК;
- `timeVosstVK` - среднее время восстановления ВК;
- `timeOtkKan` - среднее время между отказами канала связи;
- `timeVosstKan` - среднее время восстановления канала связи.

## Создание новых классов активных объектов

Построение модели начнём с создания новых классов активных объектов. 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 элементов, будем строить сеть на корневом объекте Main.

Создайте три новых класса активных объектов: 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.
3. Щелкните кнопку Далее. Откроется вторая страница Мастера создания модели. Выберите Начать создание модели "с нуля". Щелкните кнопку Далее.
4. На панели Проекты щёлкните правой кнопкой Main и выберите из контекстного меню Создать/Класс активного объекта.
5. Откроется окно Новый класс активного объекта.
6. Введите в поле Имя: имя нового класса Абонент.
7. Если нужно, в поле Описание: введите описание назначения класса Абонент. Щелкните Готово. При построении даже таких моделей это целесообразно делать.
8. Также создайте классы Канал и Маршрутизатор.

## Создание областей просмотра

Последовательно переходите на классы объектов 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:	облСеть	облАбонент	облМарш	облКанал
X:	0	0	0	0
Y:	0	0	0	0
Ширина:	770	810	740	400
Высота:	470	360	470	340
Имя:	viewData	viewData	viewData	viewData
X:	870	0	0	0
Y:	600	700	550	700
Ширина:	470	450	330	400
Высота:	620	420	250	180
Имя:	viewResult	viewResult		
X:	0	0		
Y:	600	1200		
Ширина:	442	470		
Высота:	820	432		

Приступим к созданию сегментов сети связи.

## Сегмент Абонент

### Исходные данные

1. Откройте объект Абонент.
2. Перейдите на область просмотра viewData.

- Перетащите из палитры Презентация элемент Скругленный прямоугольник. На странице Основные оставьте Имя:, предложенное системой.
- Перейдите на страницу Дополнительные панели Свойства. Введите в поля X: 5, Y: 745, Ширина: 435, Высота: 365.
- Перетащите два элемента text и на странице Основные панели Свойства в поле Текст: введите Исходные данные и Результаты соответственно (рис. 4.3).



Рис. 4.3. Размещение элементов Параметр и Простая переменная

- Переход между областями просмотра в ходе моделирования организуйте позднее.
- Перетащите из палитры Основная элементы Простая переменная, разместите и дайте им имена согласно рис. 4.3.

Тип всех простых переменных, кроме `tekEmkBuferVx`, установите `double`. Тип `tekEmkBuferVx` - `int`. Простые переменные предназначены для промежуточных вычислений и сбора статистических данных за одного абонента.

- 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
<code>kolAbonent</code>			6
<code>timeAbonent</code>			30
<code>kolProg</code>			100
<code>numAbonent</code>			1
<code>emkBuferVx</code>			80000
<code>verKat</code>	Установить флагки	KoIKat	{0.3, 0.5, 0.7, 1.0}
<code>dlKat</code>		KoIKat	{53000, 86000, 66000, 50000}
<code>dlKatO</code>		KoIKat	{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 `KoIKat`.

- Щёлкните правой кнопкой мыши в панели Проекты и в контекстном меню выберите Размерность.
- В открывшемся окне Размерность в поле Имя: введите `KoIKat`. Если введёте `kolKat`, в окне Ошибки появится сообщение: По соглашению, имена типов Java обычно начинаются с заглавной буквы.
- Установите Тип размерности: Диапазон.
- В открывшееся поле Диапазон: введите 1-4.
- Щёлкните Готово.

Теперь создайте массив `verKat`.

1. Выделите элемент Параметр (см. [рис. 4.3](#)).
2. На странице Основные панели Свойства установите флајок Массив. Рядом появится значок { ... }. Щёлкните по нему.
3. Откроется страница свойств Массив. В окошке Возможные размерности: выделите `KolKat`.
4. Щёлкните кнопку Размерность `KolKat` появится в окошке Выбранные размерности.
5. Вернитесь на страницу Основные панели Свойства.
6. В поле Значение по умолчанию: введите:  
`{0.3, 0.5, 0.7, 1.0}`
7. Создайте также массивы `dlKat`, `dlKat0` одинаковой размерности `KolKat`. Данные в поле По умолчанию введите из [табл. 4.2](#).

Для получения стандартной статистической информации о времени передачи сообщений по каждому источнику 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.

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.

1. В Палитре выделите Презентация. Перетащите элемент Скруглённый прямоугольник в область просмотра облАбонент.
2. На странице Основные в поле Имя: оставьте имя, предложенное системой. Сбросьте флажок Отображать имя.
3. Перейдите на страницу Дополнительные панели Свойства. Введите в поля X: 13, Y: 1250, Ширина: 447, Высота: 371.

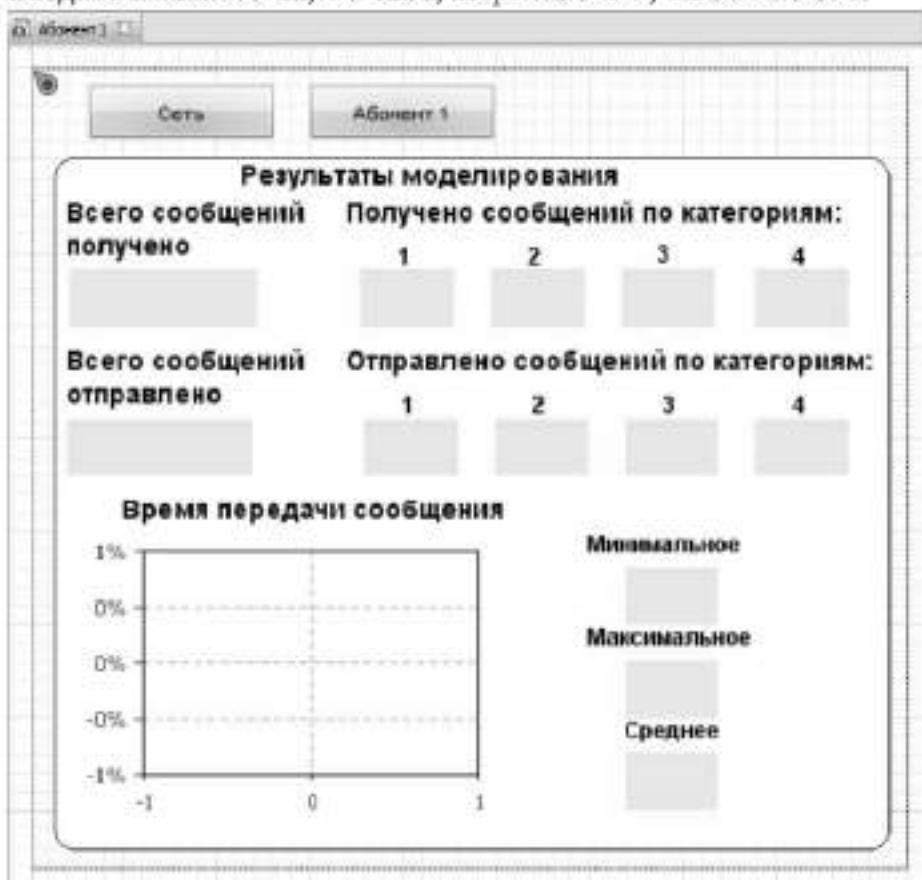


Рис. 4.4. Размещение элементов для вывода результатов

## Моделирования

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.

Всего сообщений получено	Получено сообщений по категориям			
editbox X: 21 Y: 1310	editbox1 X: 177 Y: 1310	editbox2 X: 247 Y: 1310	editbox3 X: 317 Y: 1310	editbox4 X: 389 Y: 1310
Всего сообщений отправлено	Отправлено сообщений по категориям			
editbox5 X: 19 Y: 1390	editbox6 X: 179 Y: 1390	editbox7 X: 249 Y: 1390	editbox8 X: 319 Y: 1390	editbox9 X: 389 Y: 1390
Время передачи сообщения				
Минимальное	Максимальное	Среднее		
editbox13 X: 319 Y: 1470	editbox12 X: 319 Y: 1520	editbox15 X: 319 Y: 1570		
chart-Гистограмма				

Х: 19

Y: 1450

Ширина: Высота: 160

240

3. На странице Основные панели Свойства в поле Имя: оставьте имя, предложенное системой. Сбросьте флагок 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) согласно [рис. 4.4](#).

И снова, прежде чем перейти к построению событийной части сегмента Абонент, разместим на корневом объекте 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.

#### Показатели качества обслуживания сети связи

1. Откройте корневой объект Main.
2. Перейдите на область просмотра ViewData.
3. Перетащите или скопируйте элементы Простая переменная, разместите и дайте имена согласно [рис. 4.5](#).

Сеть					
<input checked="" type="checkbox"/> всегоПолКат1	<input checked="" type="checkbox"/> всегоПолКат3	<input checked="" type="checkbox"/> всегоОтпрКат1	<input checked="" type="checkbox"/> всегоОтпрКат3		
<input checked="" type="checkbox"/> всегоПолКат2	<input checked="" type="checkbox"/> всегоПолКат4	<input checked="" type="checkbox"/> всегоОтпрКат2	<input checked="" type="checkbox"/> всегоОтпрКат4		
<input checked="" type="checkbox"/> всегоПол		<input checked="" type="checkbox"/> всегоОтпр		<input checked="" type="checkbox"/> коэфПропСпособ	
<input checked="" type="checkbox"/> k1	<input checked="" type="checkbox"/> k2	<input checked="" type="checkbox"/> k3	<input checked="" type="checkbox"/> k4	<input checked="" type="checkbox"/> g	<input checked="" type="checkbox"/> врПередСообщ
<input checked="" type="checkbox"/> d1	<input checked="" type="checkbox"/> d2	<input checked="" type="checkbox"/> d3	<input checked="" type="checkbox"/> d4	<input checked="" type="checkbox"/> f	
<b>Количество отправленных сообщений абонент - абонент</b>					
<input checked="" type="checkbox"/> отпр11	<input checked="" type="checkbox"/> отпр12	<input checked="" type="checkbox"/> отпр13	<input checked="" type="checkbox"/> отпр14	<input checked="" type="checkbox"/> отпр15	<input checked="" type="checkbox"/> отпр16
<input checked="" type="checkbox"/> отпр21	<input checked="" type="checkbox"/> отпр22	<input checked="" type="checkbox"/> отпр23	<input checked="" type="checkbox"/> отпр24	<input checked="" type="checkbox"/> отпр25	<input checked="" type="checkbox"/> отпр26
<input checked="" type="checkbox"/> отпр31	<input checked="" type="checkbox"/> отпр32	<input checked="" type="checkbox"/> отпр33	<input checked="" type="checkbox"/> отпр34	<input checked="" type="checkbox"/> отпр35	<input checked="" type="checkbox"/> отпр36
<input checked="" type="checkbox"/> отпр41	<input checked="" type="checkbox"/> отпр42	<input checked="" type="checkbox"/> отпр43	<input checked="" type="checkbox"/> отпр44	<input checked="" type="checkbox"/> отпр45	<input checked="" type="checkbox"/> отпр46
<input checked="" type="checkbox"/> отпр51	<input checked="" type="checkbox"/> отпр52	<input checked="" type="checkbox"/> отпр53	<input checked="" type="checkbox"/> отпр54	<input checked="" type="checkbox"/> отпр55	<input checked="" type="checkbox"/> отпр56
<input checked="" type="checkbox"/> отпр61	<input checked="" type="checkbox"/> отпр62	<input checked="" type="checkbox"/> отпр63	<input checked="" type="checkbox"/> отпр64	<input checked="" type="checkbox"/> отпр65	<input checked="" type="checkbox"/> отпр66
<b>Коэффициенты пропускной способности абонент - абонент</b>					
<input checked="" type="checkbox"/> кПрСп12	<input checked="" type="checkbox"/> кПрСп13	<input checked="" type="checkbox"/> кПрСп14	<input checked="" type="checkbox"/> кПрСп15	<input checked="" type="checkbox"/> кПрСп16	
<input checked="" type="checkbox"/> кПрСп21	<input checked="" type="checkbox"/> кПрСп22	<input checked="" type="checkbox"/> кПрСп23	<input checked="" type="checkbox"/> кПрСп24	<input checked="" type="checkbox"/> кПрСп25	<input checked="" type="checkbox"/> кПрСп26
<input checked="" type="checkbox"/> кПрСп31	<input checked="" type="checkbox"/> кПрСп32	<input checked="" type="checkbox"/> кПрСп33	<input checked="" type="checkbox"/> кПрСп34	<input checked="" type="checkbox"/> кПрСп35	<input checked="" type="checkbox"/> кПрСп36
<input checked="" type="checkbox"/> кПрСп41	<input checked="" type="checkbox"/> кПрСп42	<input checked="" type="checkbox"/> кПрСп43	<input checked="" type="checkbox"/> кПрСп44	<input checked="" type="checkbox"/> кПрСп45	<input checked="" type="checkbox"/> кПрСп46
<input checked="" type="checkbox"/> кПрСп51	<input checked="" type="checkbox"/> кПрСп52	<input checked="" type="checkbox"/> кПрСп53	<input checked="" type="checkbox"/> кПрСп54	<input checked="" type="checkbox"/> кПрСп55	<input checked="" type="checkbox"/> кПрСп56
<input checked="" type="checkbox"/> кПрСп61	<input checked="" type="checkbox"/> кПрСп62	<input checked="" type="checkbox"/> кПрСп63	<input checked="" type="checkbox"/> кПрСп64	<input checked="" type="checkbox"/> кПрСп65	<input checked="" type="checkbox"/> кПрСп66

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

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.

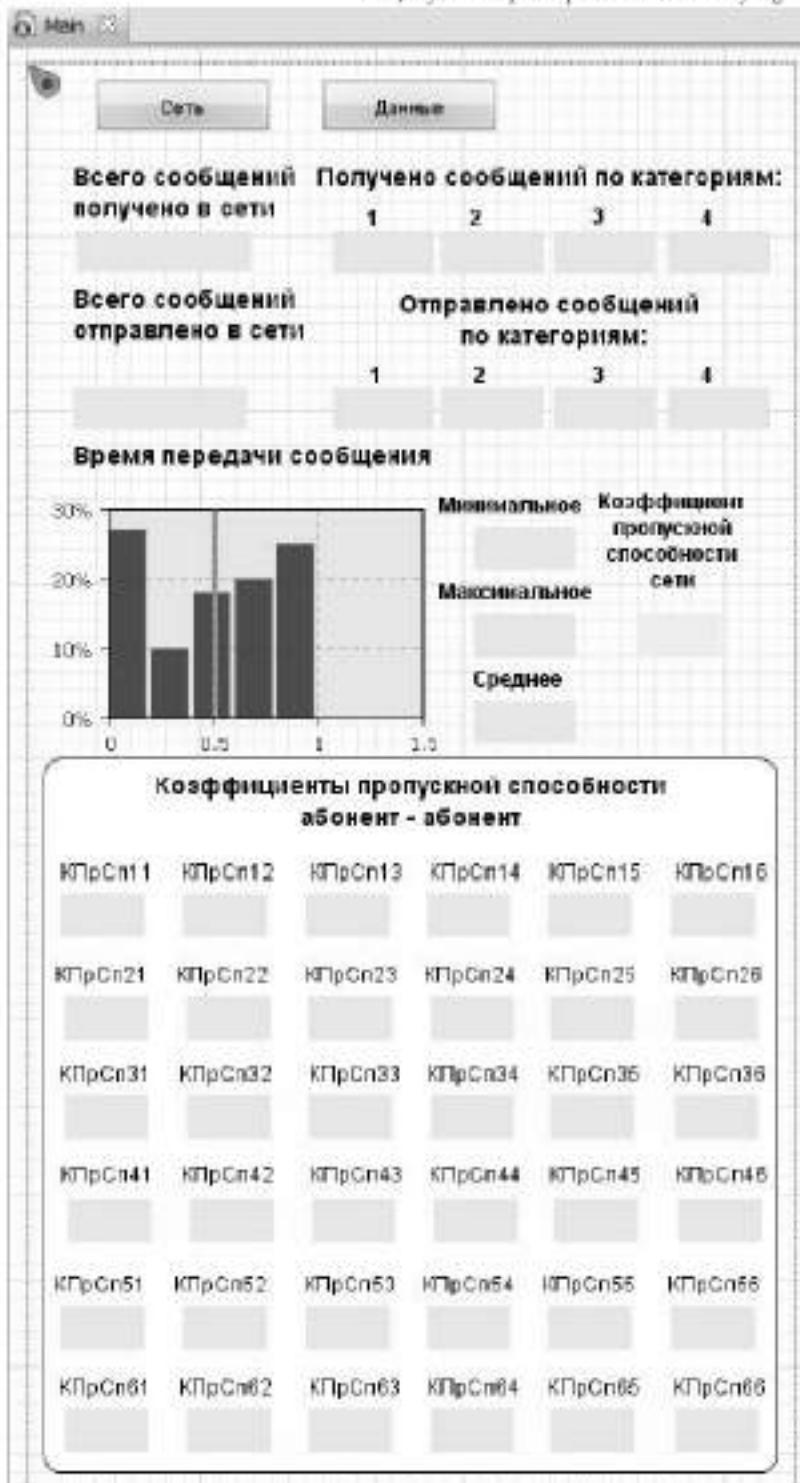
5. Выровняйте элементы. Выделите нужные элементы, щелкните правой кнопкой мышки, выберите Выравнивание.
6. Перетащите элемент Данные гистограммы с палитры Статистика.
7. Задайте свойства элемента:
  - измените Имя: на врПередСообщ;
  - сделайте Количество интервалов: равным 10;
  - задайте Нач. размер интервалов: 0.01.

Теперь организуем вывод в более презентабельном виде. На [рис. 4.6](#) показаны элементы для вывода 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.
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.



### Рис. 4.6. Вывод показателей качества обслуживания сети

13. На странице Основные в поле Имя: оставьте имя, предложенное системой. Сбросьте флажок Отображать имя.
14. Перейдите на страницу Дополнительные 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 КПрСпл11. Сбросьте флажок Отображать имя.
17. Перетащите элемент text. В поле Текст: введите КПрСпл11.
18. Скопируйте элементы text и Текстовое поле. Вставьте их пять раз согласно [рис. 4.6](#). При этом в полях Имя: последовательно меняются имена КПрСпл12... КПрСпл16. Но поле Текст: останется неизменным, т.е. КПрСпл11. 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.

### 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. Перейдите на страницу Дополнительные 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 Основной библиотеки (Enterprise Library) объект `source`. Поместите его так же, как на рис. 4.8. Остальные объекты блока Абонент1 вы будете размещать по мере необходимости.

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

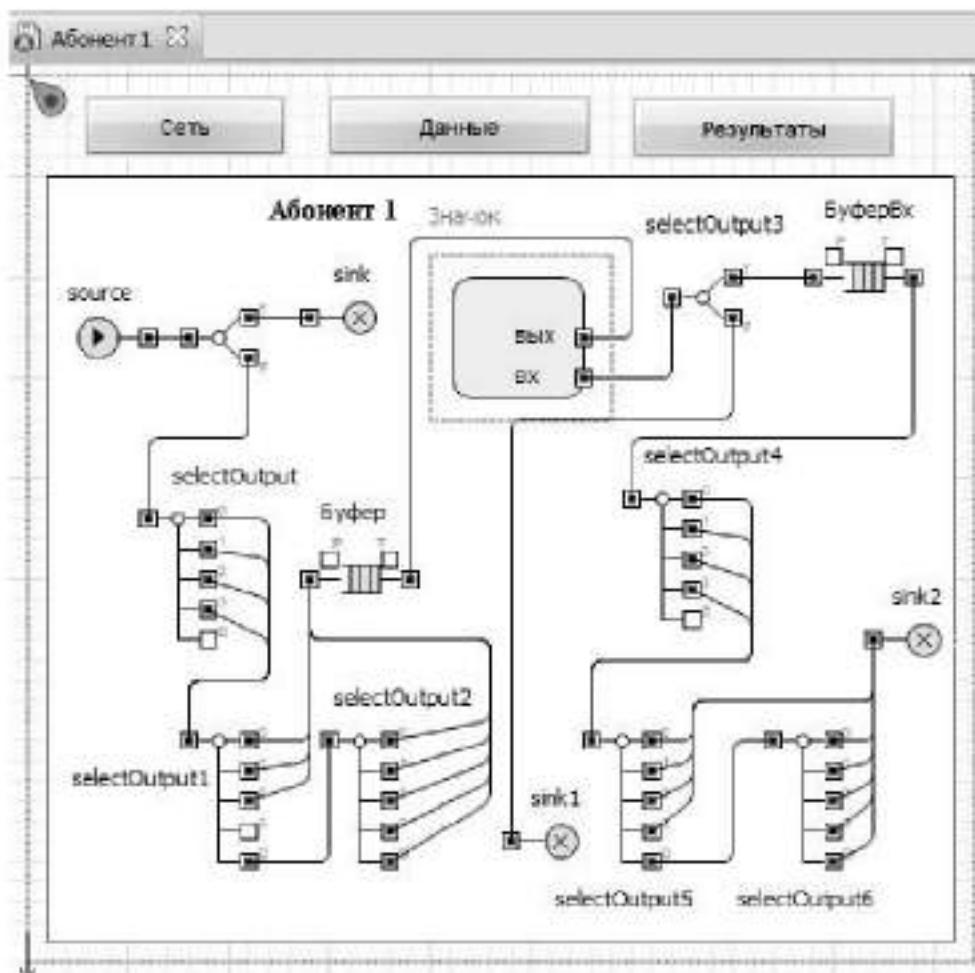


Рис. 4.8. Реализация блока Абонент1 средствами AnyLogic

Создайте класс заявки Message.

1. Выделите объект source.
2. В панели Проект щелкните 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 в качестве базового класса (в ранних версиях - anylogic.libraries.enterprise.Entity). Щелкните кнопку Далее.
5. Появится вторая страница Мастера создания Java класса. Добавьте поля Java класса, показанные на рис. 4.9.

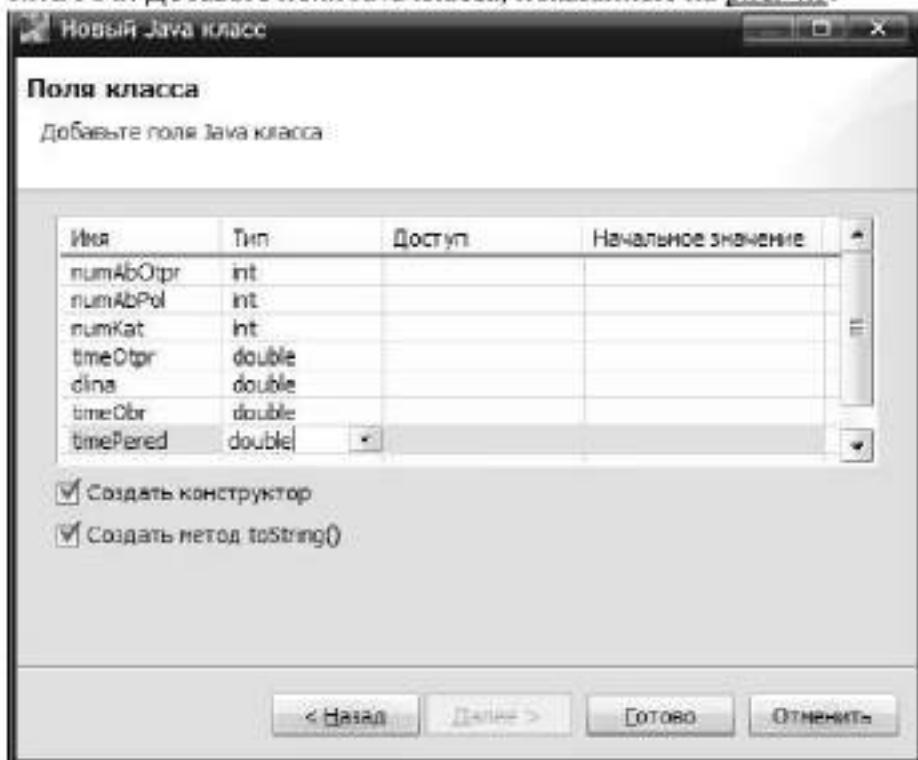


Рис. 4.9. Дополнительные поля класса заявок Message

6. Оставьте выбранными флагги Создать конструктор и

Создать метод `toString ()`.

7. Щелкните Готово. Закройте редактор кода.

Продолжим построение блока Абонент1.

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

- Класс заявки: `Message`
- Заявки прибывают согласно Времени между прибытиями
- Время между прибытиями `exponential(1/timeAbonent)`
- Количество заявок, прибывающих за один раз 1
- Новая заявка `new Message ()`
- Действие 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.timeOtr=time();
entity.numAbOtr=kolAbonent;
```

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 `timeOtr` заносится время отправления сообщения.

Замечание. Основная библиотека (Enterprise Library) содержит объекты `TimeMeasureStart` и `TimeMeasureEnd`, позволяющие измерять время прохождения заявки между двумя

заданными точками диаграммы процесса. TimeMeasureStart запоминает момент времени, в который заявка проходит через этот объект. TimeMeasureEnd вычисляет для поступившей в него заявки разность между текущим временем и запомненным объектом TimeMeasureStart, на который ссылается этот объект. Эта разность добавляется в элементы сбора статистики, встроенные в объект TimeMeasureEnd.

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 Т объекта selectOutput.
3. Перетащите объект selectOutput5. Данный объект предназначен для розыгрыша категорий отправляемых сообщений. Его вход соедините с выходом F объекта selectOutput. На странице Основные 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.

Таблица 4.4.

Свойство	selectOutput
Класс заявки:	Message
Выход true выбирается	При выполнении условия
Условие	entity.numAb0ptr==entity.numAbPol
Класс заявки:	Message
Использовать:	Условия a=uniform();

	<pre> f++; kolOptr=f/kolProg; editbox5.setText(kolOptr, true); get_Main().f++; get_Main().всегоОтпр=get_Main().f/ kolProg; get_Main().editbox5.setText(get_Mai) всегоОтпр, true); </pre>
<b>Условие 0</b>	<pre> a&lt;=verKat.get(1) entity.numKat=1; b=normal(dlKat0.get(entity.numKat), dlKat.get(entity.numKat)); entity.dlina=(int)(b); d1++; kolOptrKat1=d1/kolProg; editbox6.setText(kolOptrKat1, true) get_Main().d1++; get_Main().всегоОтпрKat1=get_Main() get_Main().editbox6.setText(get_Mai true); </pre>
<b>Условие 1</b>	<pre> a&lt;=ver verKat.get(2) entity.numKat=2; b=normal(dlKat0.get(entity.numKat), dlKat.get(entity.numKat)); entity.dlina = (int)(b); d2++; kolOptrKat2=d2/kolProg; editbox7.setText(kolOptrKat2, true) get_Main().d2++; get_Main().всегоОтпрKat2=get_Main() get_Main().editbox7.setText(get_Mai true); </pre>
<b>Условие 2</b>	<pre> a&lt;=verKat.get(3) entity.numKat=3; </pre>

```
b=normal(dlKat0.get(entity.numKat),
dlKat.get(entity.numKat));
entity.dlina=(int)(b);
d3++;
kolOtpKat3=d3/kolProg;
editbox8.setText(kolOtpKat3, true)
get_Main().d3++;
get_Main().всегоOтпрKat3=get_Main()
get_Main().editbox8.setText(get_Main(),
true);
```

**Условие 3** a<=verKat.get(4)

```
entity.numKat=4;
b=normal(dlKat0.get(entity.numKat),
dlKat.get(entity.numKat));
entity.dlina=(int)(b);
```

**Действие при выходе 3** d4++;
kolOtpKat4=d4/kolProg;

```
editbox9.setText(kolOtpKat4, true)
get_Main().d4++;
get_Main().всегоOтпрKat4=get_Main()
get_Main().editbox9.setText(get_Main(),
true);
```

<b>Свойство</b>	selectOutput1
-----------------	---------------

Класс заявки: Message

Использовать: Условия

**Условие 0** entity.numAbPol==1

**Условие 1** entity.numAbPol==2

**Действие при** отпрAb2++;

**выходе 1** get\_Main().отпр12=отпрAb2;

**Условие 2** entity.numAbPol==3

**Действие при** отпрAb3++;

**выходе 2** get\_Main().отпр13=отпрAb3;

**Условие 3** entity.numAbOtpri==entity.numAbPol

<b>Свойство</b>	selectOutput2
-----------------	---------------

**Класс заявки:** Message

**Использовать:** Условия

**Условие 0** entity.numAbPol==4

**Действие при выходе 0** отправ4++;

**выходе 0** get\_Main().отпр14=отпрA64;

**Условие 1** entity.numAbPol==5

**Действие при выходе 1** отправ5++;

**выходе 1** get\_Main().отпр15=отпрA65;

**Условие 2** entity.numAbPol==6

**Действие при выходе 2** отправ6++;

**выходе 2** get\_Main().отпр16=отпрA66;

- Перетащите объект queue. Укажите его свойства: Имя: Буфер. Класс заявки: Message. Вместимость 100. Соедините выходы 0...3 selectOutput1 с входом элемента Буфер (см. [рис. 4.8](#)).
- Соедините выход по умолчанию selectOutput1 с входом selectOutput2. Через этот выход будут проходить на selectOutput2 сообщения, адресованные абонентам 4, 5 и 6.
- 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.

- Из палитры Презентация перетащите элемент Скруглённый прямоугольник.
- Перейдите на страницу Дополнительные панели Свойства. Введите в поля X: 210, Y: 100, Ширина: 65, Высота: 60.
- Из палитры Основная перетащите два элемента порт. Разместите их как на [рис. 4.8](#). В поле Имя: замените предложенное системой на вх и вых.
- Обратите внимание на то, чтобы у элементов Скруглённый прямоугольник и порт был установлен флажок На

верхнем уровне. У остальных элементов сегмента Абонент1 этот флагок должен быть сброшенным.

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

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:
  - Класс заявки: Message
  - Выход true выбирается при выполнении условия  $emkBuferVx - tekEmkBuferVx >= entity.dlina$
5. Перетащите объект queue. Укажите его свойства:
  - Имя: БуферВх
  - Класс заявки: Message
  - Вместимость emkBuferVx
  - Действие при  $tekEmkBuferVx + entity.dlina;$  входе
  - Действие при выходе  $tekEmkBuferVx - entity.dlina;$

Соедините его вход с выходом T объекта selectOutput3.

6. Перетащите объект selectOutput5 (имя selectOutput4). Он предназначен для разделения потока полученных сообщений по категориям. На странице Основные 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 количества полученных сообщений за один прогон,

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

7. Соедините выход элемента БуферВх с выходом объекта selectOutput4.

Таблица 4.5.

Свойство	selectOutput4
Класс заявки: Message	
Использовать: Условия	
	<pre> g++; kolPol=g/kolProg; editbox.setText(kolPol, true); </pre>
Действие при выходе:	<pre> get_Main().g++; get_Main().всегоПол=get_Main().g/ kolProg; get_Main().editbox.setText(get_Main() true); </pre>
Условие 0	<pre> entity.numKat==1 k1++; kolPolKat1=k1/kolProg; editbox1.setText(kolPolKat1, true); get_Main().k1++; get_Main().всегоПолKat1=get_Main().k1; get_Main().editbox1.setText(get_Main() true); </pre>
Действие при выходе 0	
Условие 1	<pre> entity.numKat==2 k2++; kolPolKat2=k2/kolProg; editbox2.setText(kolPolKat2, true); get_Main().k2++; get_Main().всегоПолKat2=get_Main().k2; get_Main().editbox2.setText(get_Main() true); </pre>
Действие при выходе 1	
Условие 2	<pre> entity.numKat==3 k3++; </pre>

	kolPolKat3=k3/kolProg; editbox3.setText(kolPolKat3, true); get_Main().k3++; get_Main().всегоПолKat3=get_Main(). get_Main().editbox3.setText(get_Main(). true);
Условие 3	entity.numKat==4 k4++; kolPolKat4=k4/kolProg; editbox4.setText(kolPolKat4, true); get_Main().k4++; get_Main().всегоПолKat4=get_Main(). get_Main().editbox4.setText(get_Main(). true);
Свойство	selectOutput5
Класс заявки: Message	
Использовать: Условия	
Условие 0	entity.numAbOtpr==1
Условие 1	entity.numAbOtpr==2 отAб2++; get_Main().кПрСп21=отAб2/get_Main() отпр21; get_Main().КПрСп21.setText(get_Main(). true);
Условие 2	entity.numAbOtpr==3 отAб3++; get_Main().кПрСп31=отAб3/get_Main() отпр31; get_Main().КПрСп31.setText(get_Main(). true);
Условие 3	entity.numAbOtpr==4 отAб4++; get_Main().кПрСп41=отAб4/get_Main()
Действие при выходе 1	
Действие при выходе 2	
Действие при выходе 3	

выходе 3	get_Main().КПрСп41.setText(get_Main().true);
Свойство	selectOutput6
Класс заявки:	Message
Использовать:	Условия
Условие 0	entity.numAb0ptr==5 отAб5++; get_Main().кПрСп51=отAб5/get_Main(). отпр51; get_Main().КПрСп51.setText(get_Main(). true);
Условие 1	entity.numAb0ptr==6 отAб6++; get_Main().кПрСп61=отAб6/get_Main(). отпр61; get_Main().КПрСп61.setText(get_Main(). true);

8. 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.
9. Перетащите объект sink2. Его вход соедините с выходами 0...3 объекта selectOutput5 и выходами объекта selectOutput6.
10. Установите свойства объекта sink2 согласно табл. 4.6. Код рассчитывает коэффициент пропускной способности сети связи и время передачи одного сообщения.

Таблица 4.6.

Свойство	sink2
Класс заявки:	Message

```

editbox11.setText(врПередачи.mean(), true)
editbox12.setText(врПередачи.max(), true);
editbox13.setText(врПередачи.min(), true);
get_Main().коэфПропСпособ=get_Main();
всегоПол/get_Main().всегоОтпр;
Действие при входе: get_Main().editbox10.setText(get_Main().коэфПропСпособ, true);
get_Main().врПередСооб.add(time()-entity.t);
get_Main().editbox11.setText(get_Main().врПередачи.mean(), true);
get_Main().editbox12.setText(get_Main().врПередачи.max(), true);
get_Main().editbox13.setText(get_Main().врПередачи.min(), true);

```

## Сегмент Маршрутизатор

### Исходные данные

1. Откройте объект 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. Перейдите на страницу Дополнительные панели Свойства. Введите в поля X: 10, Y: 600, Ширина: 314, Высота: 190.
5. Перетащите элемент `text` и на странице Основные панели Свойства в поле Текст: введите Исходные данные ([рис. 4.9.](#)).
6. Перетащите элементы Параметр и Простая переменная. Разместите их и дайте имена согласно [рис. 4.10.](#)
7. Значения по умолчанию элементов Параметр установите согласно [табл. 4.7.](#)
8. Тип `timeOfkBK`, `timeVosstBK`, `proizvod` установите `double`, остальных параметров и простых переменных - `int`.

double, остальных параметров и простых переменных - int.

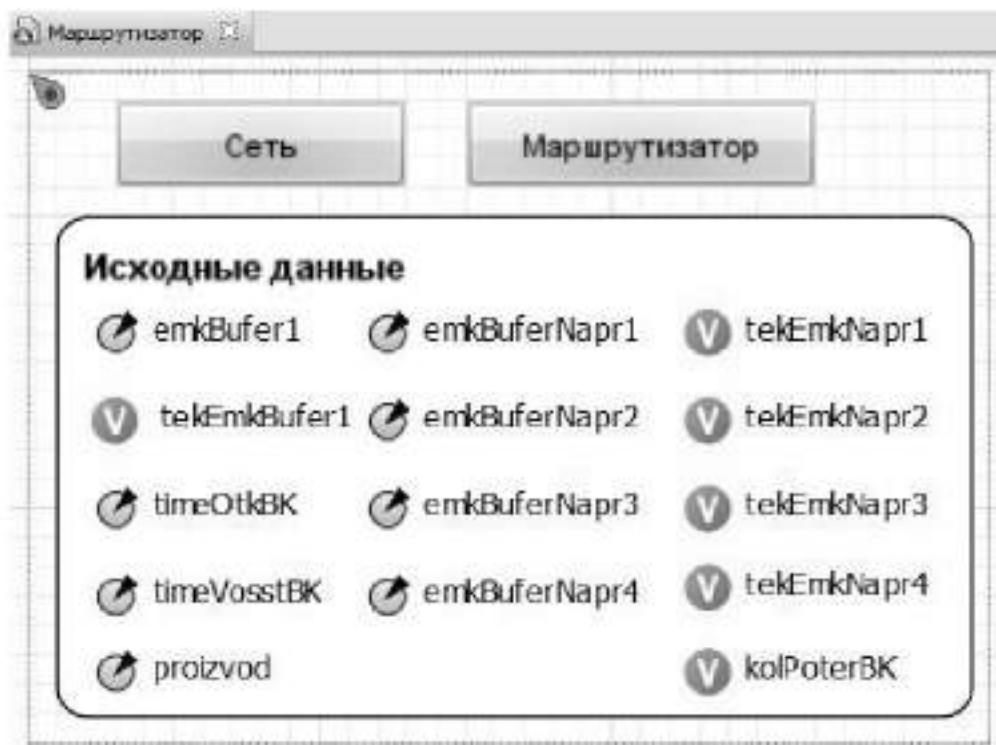


Рис. 4.10. Исходные данные сегмента Маршрутизатор

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

### 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.

Приступим к построению событийной части сегмента Маршрутизатор.

#### 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. Перейдите на страницу Дополнительные панели Свойства. Введите в поля X: 26, Y: 50, Ширина: 154, Высота: 150.
3. Перетащите объекты `selectOutput`, `hold` и `sink` на диаграмму класса Маршрутизатор, разместите и соедините так, как показано на [рис. 4.11](#).
4. Перетащите элемент `text` и на странице Основные панели Свойства в поле Текст: введите Блок контроля 1.
5. Выделите объект `selectOutput`.
6. В поле Имя: вместо `selectOutput` введите `blokKontrol_1`.
7. В поле Класс заявки: `Entity` замените `Message`.

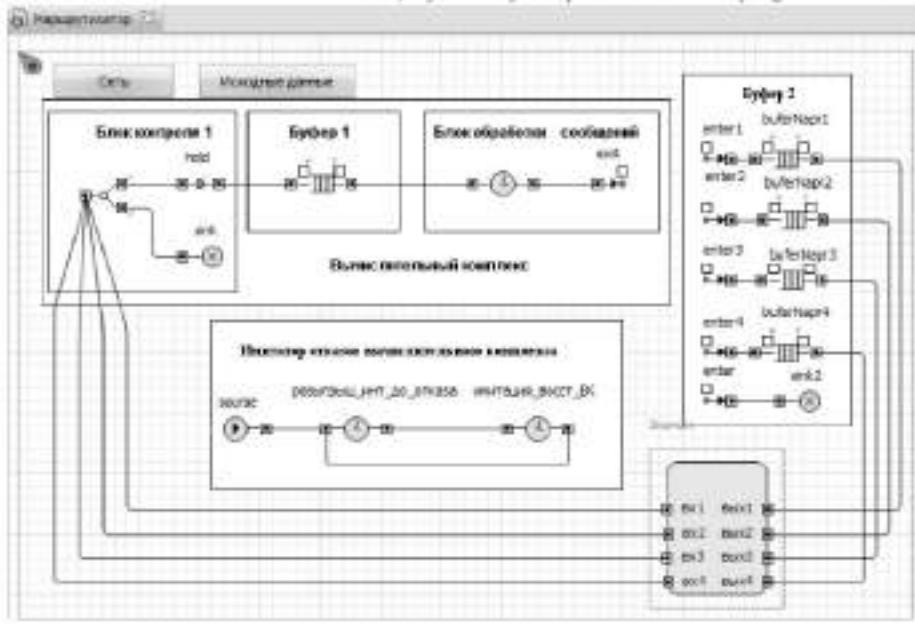


Рис. 4.11. Событийная часть сегмента Маршрутизатор



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 Класс заявки: Entity 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 Класс заявки: Entity замените Message.

### 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. Перейдите на страницу Дополнительные 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 устанавливайте флажок Отображать имя.

4. Перетащите элемент **text** и на странице Основные панели Свойства в поле Текст : введите Буфер 1.

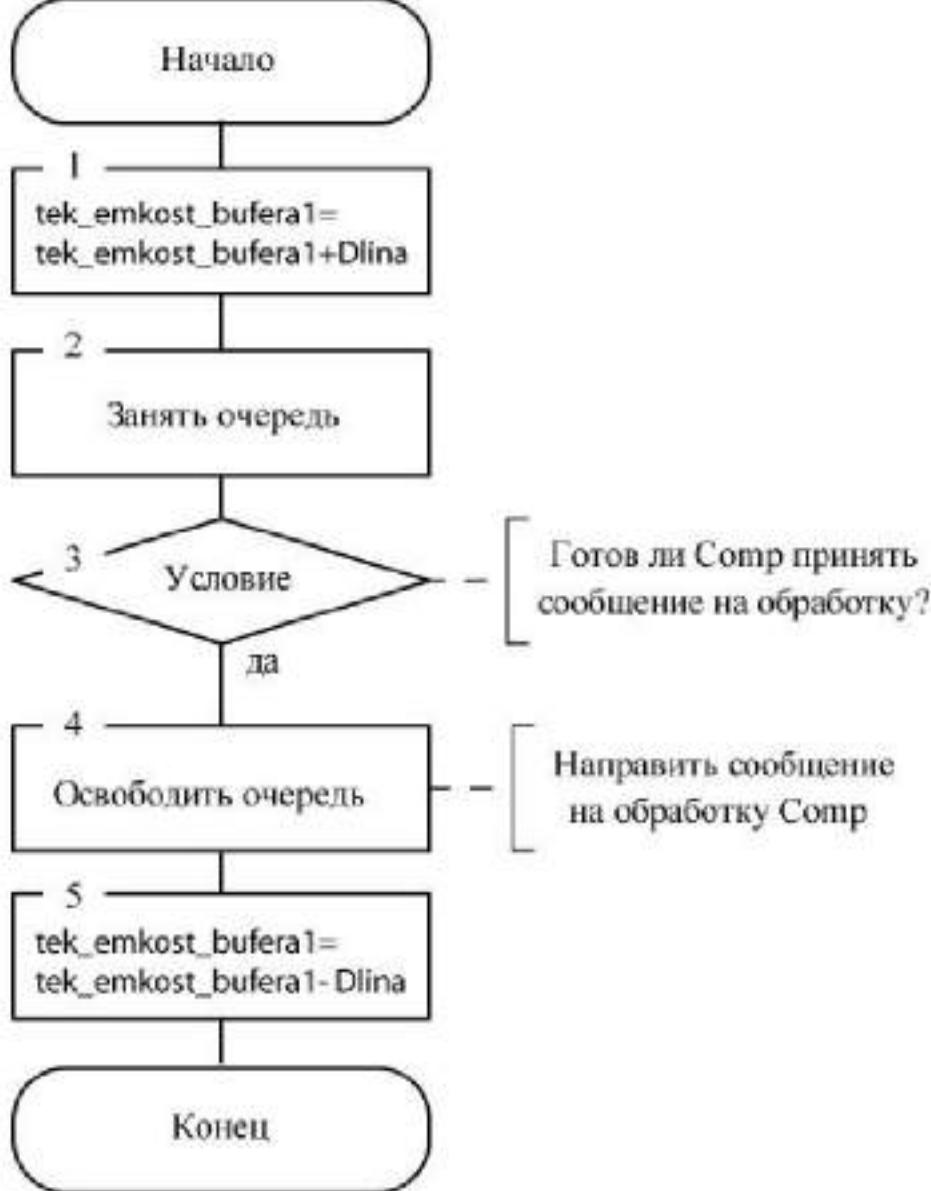


Рис. 4.13. Алгоритм работы блока Буфер 1

- Выделите объект **queue**,
- В поле Имя : вместо **queue** введите **bufer1**.
- В поле Класс заявки: **Entity** замените **Message**.

8. В поле Вместимость введите emkBufer1.
9. При помещении сообщения в буфер его текущая емкость увеличивается, поэтому в поле Действие при входе введите:

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

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

```
tekEmkBufer1 = -entity.dlina;
```

11. Поставьте флагок Включить сбор статистики.

#### 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. Не устанавливайте флажок Отображать имя.

3. Перейдите на страницу Дополнительные панели Свойства. Введите в поля X: 336, Y: 50, Ширина: 194, Высота: 100.
4. Перетащите элемент text и на странице Основные панели Свойства в поле Текст: введите 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 Класс заявки: Entity замените Message.
8. Задержка задаётся установите Явно.
9. В поле Время задержки введите:

`exponential(1/entity.timeObr)`

10. Оставьте Вместимость 1.
11. Действие при входе `entity.timeObr = entity.dлина/proизвод;`
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 устанавливайте флажок Отображать имя.
15. Перейдите на страницу Дополнительные панели Свойства. Введите в поля X: 20, Y: 40, Ширина: 520, Высота: 170.
16. Перетащите элемент text и на странице Основные панели Свойства в поле Текст: введите Вычислительный комплекс.

## 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](#).

Вначале определяется номер направления, по которому должно быть передано поступившее сообщение.



Рис. 4.15. Алгоритм работы блока Блок контроля 2

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 Класс заявки: `Entity` замените `Message`.
3. В поле Действие 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 определенным образом, например, таблицей маршрутизации. В данном случае он настраивается программным путем так, что сообщения первого и второго отправителей передаются по первому направлению, третьего и четвертого 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](#)).

## 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 Rectangle. Не устанавливайте флажок Отображать имя.
3. Перейдите на страницу Дополнительные 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 `bufNapr1`, вход которого соединён с выходом `enter1` и установите значения свойств.
7. В поле Класс заявки: Entity замените Message.
8. Вместимость `emkBufNapr1`
9. В поле Действие при входе введите:

```
tekEmkNapr1 += emity.dlina;
```

10. В поле Действие при выходе введите:

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

11. Установите флагок Включить сбор статистики.
12. Выделите элемент buferNapr2, вход которого соединен с выходом enter2 и установите значения свойств.
13. В поле Класс заявки: Entity замените Message.
14. Вместимость emkBuferNapr2
15. В поле Действие при входе введите:

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

16. В поле Действие при выходе введите:

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

17. Установите флагок Включить сбор статистики.
18. Выделите элемент buferNapr3, вход которого соединен с выходом enter3 и установите значения свойств.
19. В поле Класс заявки: Entity замените Message.
20. Вместимость emkBuferNapr3
21. В поле Действие при входе введите:

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

22. В поле Действие при выходе введите:

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

23. Установите флагок Включить сбор статистики.
24. Выделите элемент buferNapr4, вход которого соединен с выходом enter4 и установите значения свойств.
25. В поле Класс заявки: Entity замените Message.
26. Вместимость emkBuferNapr4
27. В поле Действие при входе введите:

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

28. В поле Действие при выходе введите:

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

## 29. Установите флажок Включить сбор статистики.

### 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 Дополнительные 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).

### 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 Дополнительные 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:

- Заявки прибывают согласно Интенсивности
- Интенсивность прибытия 1
- Количество заявок, прибывающих за один раз 1
- Ограниченнное количество прибытий 1
- Максимальное количество прибытий 1

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

- Имя: `розыгрыш_инт_до_отказа`
- Задержка задаётся Явно
- Время задержки `exponential(1/timeOtKBK)`
- Вместимость 1
- Действие 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_BK`
- Задержка задаётся Явно
- Время задержки `exponential(1/timeVosstBK)`
- Вместимость 1
- Действие при выходе

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

## 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.

### Исходные данные

1. Откройте объект Канал. Перейдите на область просмотра `viewData`.
2. Перетащите элемент Скруглённый прямоугольник.
3. На странице Дополнительные панели Свойства введите в поля X : 10, Y : 752, Ширина : 377, Высота : 118.
4. Перетащите элементы Параметр и Простая 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
<code>skorPerek</code>	<code>double</code>	5000
<code>skorPerekR</code>	<code>double</code>	5000
<code>timeOtkKan</code>	<code>double</code>	360
<code>timeVosstKan</code>	<code>double</code>	3,2
<code>timeBklResK</code>	<code>double</code>	0,1
<code>всего_потеряно_сообщ</code>	<code>int</code>	0

### Событийная часть сегмента Каналы

1. Перейдите на область просмотра обл Каналы.
2. Перетащите элемент Прямоугольник.
3. На странице Дополнительные панели Свойства введите в поля X : 186, Y : 50, Ширина : 200, Высота : 135.
4. Перетащите два элемента `hold`, разместите как на [рис. 4.17](#).
5. Оставьте имена, предложенные системой, и флагок Отображать имя.
6. В поле Класс заявки: Entity замените Message.



Рис. 4.16. Исходные данные сегмента Канал

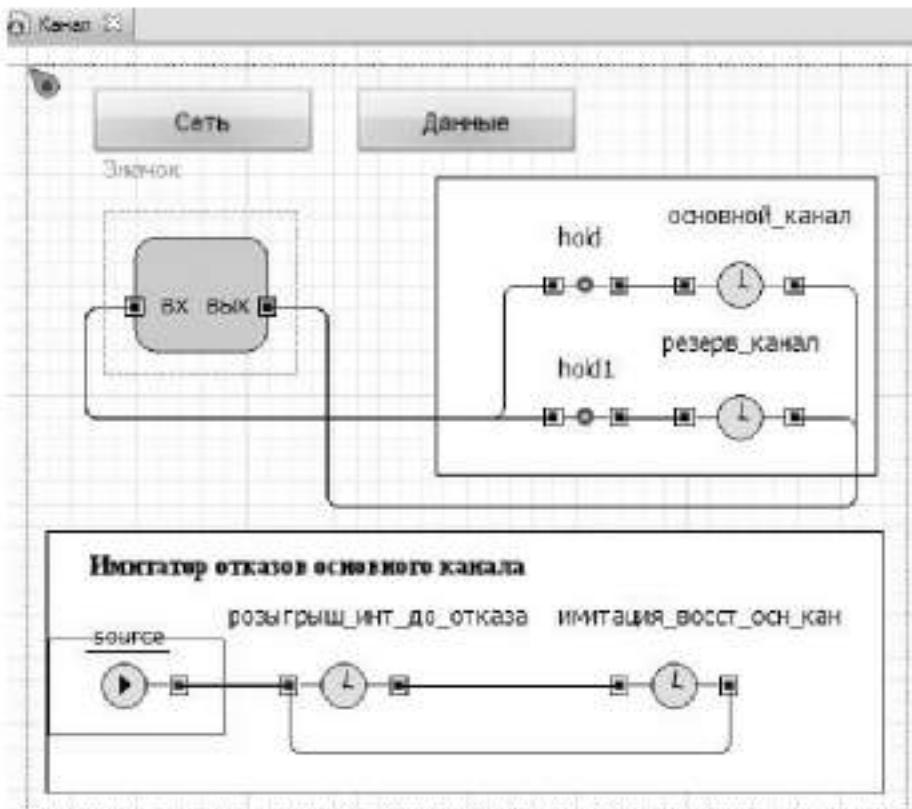


Рис. 4.17. Объекты событийной части сегмента Канал

7. У элемента `hold1` установите флагок Изначально заблокирован.
8. Перетащите объект `delay`, поместите сверху (см. рис. 4.16). В поле Имя: введите основной\_канал.
9. Установите флагок Отображать имя.
10. В поле Класс заявки: Entity замените Message. Задержка задаётся установите Явно.
11. В поле Время задержки введите

```
exponential(1/entity.timePered)
```

12. В поле Вместимость введите 1.
13. Действие при входе  
`entity.timePered=entity.dlina/ skorPeredKan;`
14. Установите флагок Включить сбор статистики.
15. Скопируйте объект `delay`. Вставьте его один раз. При этом изменится имя на основной\_канал, а остальные свойства останутся неизменными.
16. Замените имя основной\_канал на резерв\_канал.
17. В поле Действие при входе замените имеющийся там код следующим:

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

18. Действие при выходе

```
if (hold.isBlocked()== true)
    hold1.setBlocked(false);
```

19. Соедините объекты `hold` и `delay` согласно рис. 4.17.

## Организация входного и выходного портов

1. Перетащите элемент Скруглённый прямоугольник.
2. На странице Дополнительные 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 флагки Отображать имя.
4. Обратите также внимание на то, чтобы у элементов Скруглённый прямоугольник и Порт был установлен флагок На верхнем уровне. У остальных элементов сегмента Канал этот флагок должен быть сброшенным.
5. Соедините выходы элементов основной\_канал и резерв\_канал с портом вых, а порт вх - с входами элементов hold и hold1.

### Имитатор отказов каналов связи

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

1. Перетащите элемент Прямоугольник.
2. На странице Дополнительные 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 флагок
  - Заявки прибывают согласно Интенсивности
  - Интенсивность прибытия 1
  - Количество заявок, прибывающих за один раз 1
  - Ограниченнное количество прибытий установите флагок
  - Максимальное количество прибытий 1
  - Действие при выходе
5. Выделите первый объект delay и установите значения его свойств:

- Имя: `розыг_инт_до_отказа`
- Отображать имя установите флајок
- Задержка задаётся Явно
- Время задержки `exponential(1/timeOtkKan)`
- Вместимость 1
- Действие при выходе

```

hold.setBlocked(true);
if (основной_канал.size()!=0) {
    основной_канал.remove((Message)
    основной_канал.get(0));
    всего_потеряно_сообщ++;
}
hold1.setBlocked(false);
a=1;

```

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

- Имя имитация\_восст\_осн\_кан
- Отображать имя установите флајок
- Задержка задаётся Явно
- Время задержки `exponential(1/timeVosstKan)`
- Вместимость 1
- Действие при выходе

```

hold.setBlocked(false);
hold1.setBlocked(true);

```

## Построение модели сети связи

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

1. Перейдите 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 активные объекты 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](#).
11. Выходы объектов Абонент1...Абонент6 соедините с соответствующими входами объектов Канал1...Канал6.
12. Из окна Проекты перетащите объект Маршрутизатор. В поле Имя : сделайте маршрут1.
13. Соедините выходы первого и второго абонентов с вх1, выходы третьего и четвёртого - с вх2, пятого - с вх3, шестого - с вх4 объекта маршрут1.

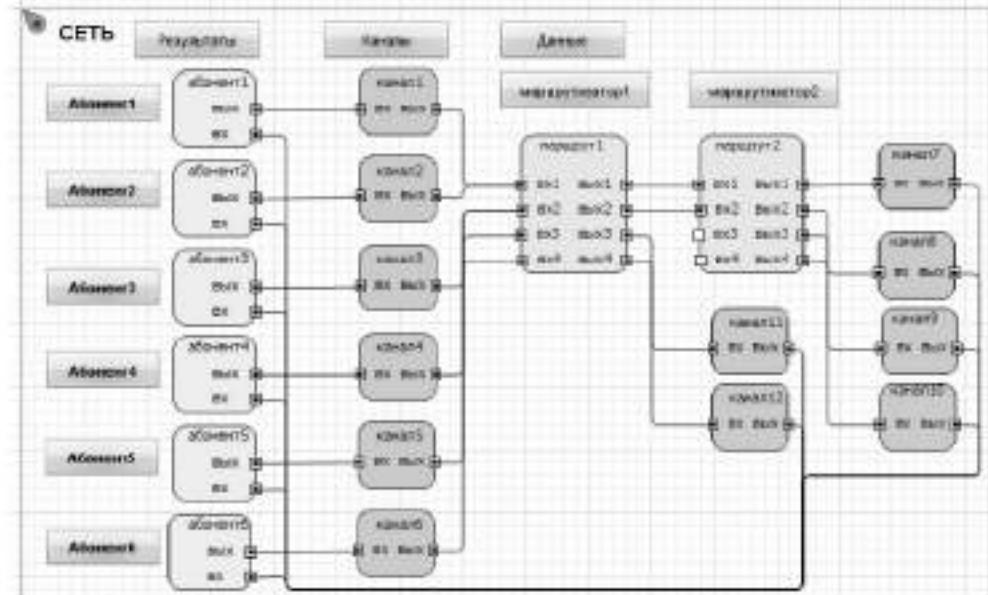


Рис. 4.18. Элементы модели функционирования сети связи

Таблица 4.9.

## Абонент2

Свойство	selectOutput
Класс заявки:	Message
Выход true выбирается	При выполнении условия
Условие	<code>entity.numAbOptr==entity.numAbPol numAbonent 2</code>
Свойство	selectOutput1
Класс заявки:	Message
Использовать:	Условия
Условие 0	<code>entity.numAbPol==1</code>
Действие при выходе 0	<code>отпрAb1++; get_Main().отпр21=отпрAb1;</code>
Условие 1	<code>entity.numAbPol==2</code>
Условие 2	<code>entity.numAbPol==3</code>

<b>Действие при выходе 2</b>	отпрAб3++; get_Main().отпр23=отпрAб3;
<b>Условие 3</b>	entity.numAbOptr==entity.numAbPol
<b>Свойство</b>	selectOutput2
<b>Класс заявки:</b>	Message
<b>Использовать:</b>	Условия
<b>Условие 0</b>	entity.numAbPol==4
<b>Действие при выходе 0</b>	отпрAб4++; get_Main().отпр24=отпрAб4;
<b>Условие 1</b>	entity.numAbPol==5
<b>Действие при выходе 1</b>	отпрAб5++; get_Main().отпр25=отпрAб5;
<b>Условие 2</b>	entity.numAbPol==6
<b>Действие при выходе 2</b>	отпрAб6++; get_Main().отпр26=отпрAб6;
<b>Свойство</b>	selectOutput5
<b>Класс заявки:</b>	Message
<b>Использовать:</b>	Условия
<b>Условие 0</b>	entity.numAbOptr==1
<b>Действие при выходе 0</b>	отAб1++; get_Main().кПрCп12=отAб1/get_Main(). отпр12; get_Main().КПрCп12.setText(get_Main().к true);
<b>Условие 1</b>	entity.numAbOptr==2
<b>Условие 2</b>	entity.numAbOptr==3
<b>Действие при выходе 2</b>	отAб3++; get_Main().кПрCп32=отAб3/get_Main(). отпр32; get_Main().КПрCп32.setText(get_Main().к true);
<b>Условие 3</b>	entity.numAbOptr==4
	отAб4++;

Действие при выходе 3	<pre>get_Main().КПрСп42=отАб4/get_Main(); отпр42; get_Main().КПрСп42.setText(get_Main().к true);</pre>
Свойство	selectOutput
Класс заявки:	Message
Использовать:	Условия
Условие 0	<pre>entity.numAb0ptr==5 отАб5++; get_Main().КПрСп52=отАб5/get_Main(); отпр52; get_Main().КПрСп52.setText(get_Main().к true);</pre>
Действие при выходе 0	
Условие 1	<pre>entity.numAb0ptr==6 отАб6++; get_Main().КПрСп62=отАб6/get_Main(); отпр62; get_Main().КПрСп62.setText(get_Main().к true);</pre>
Действие при выходе 1	

**Абонент3**

Свойство	selectOutput
Класс заявки:	Message
Выход true выбирается	При выполнении условия
Условие	<pre>entity.numAb0ptr==entity.numAbPol numAbonent 3</pre>
Свойство	selectOutput1
Класс заявки:	Message
Использовать:	Условия
Условие 0	<pre>entity.numAbPol==1 отпрАб1++; get_Main().отпр31=отпрАб1;</pre>
Действие при выходе 0	
Условие 1	<pre>entity.numAbPol==2 отпрАб2++; get_Main().отпр32=отпрАб2;</pre>
Действие при выходе 1	

<b>Действие при выходе 2</b>	отпрAб2++; get_Main().отпр32=отпрAб2;
<b>Условие 2</b>	entity.numAbPol==3
<b>Условие 3</b>	entity.numAb0ptr==entity.numAbPol
<b>Свойство</b>	selectOutput2
<b>Класс заявки:</b>	Message
<b>Использовать:</b>	Условия
<b>Условие 0</b>	entity.numAbPol==4
<b>Действие при выходе 0</b>	отпрAб4++; get_Main().отпр34=отпрAб4;
<b>Условие 1</b>	entity.numAbPol==5
<b>Действие при выходе 1</b>	отпрAб5++; get_Main().отпр35=отпрAб5;
<b>Условие 2</b>	entity.numAbPol==6
<b>Действие при выходе 2</b>	отпрAб6++; get_Main().отпр36=отпрAб6;
<b>Свойство</b>	selectOutput5
<b>Класс заявки:</b>	Message
<b>Использовать:</b>	Условия
<b>Условие 0</b>	entity.numAb0ptr==1 отAб1++;
<b>Действие при выходе 0</b>	get_Main().кПрCп13=отAб1/get_Main(). отпр13; get_Main().КПрCп13.setText(get_Main().к true);
<b>Условие 1</b>	entity.numAb0ptr==2 отAб2++;
<b>Действие при выходе 1</b>	get_Main().кПрCп23=отAб2/get_Main(). отпр23; get_Main().КПрCп23.setText(get_Main().к true);
<b>Условие 2</b>	entity.numAb0ptr==3
<b>Условие 3</b>	entity.numAb0ptr==4

<b>Действие при выходе 3</b>	<code>отAб4++;</code>
	<code>get_Main().кПрСп43=отAб4/get_Main(). отпр43;</code>
	<code>get_Main().КПрСп43.setText(get_Main().к true);</code>
<b>Свойство</b>	<code>selectOutput6</code>
<b>Класс заявки:</b>	<code>Message</code>
<b>Использовать:</b>	<code>Условия</code>
<b>Условие 0</b>	<code>entity.numAbOptr==5</code>
<b>Действие при выходе 0</b>	<code>отAб5++;</code>
	<code>get_Main().кПрСп53=отAб5/get_Main(). отпр53;</code>
	<code>get_Main().КПрСп53.setText(get_Main().к true);</code>
<b>Условие 1</b>	<code>entity.numAbOptr==6</code>
<b>Действие при выходе 1</b>	<code>отAб6++;</code>
	<code>get_Main().кПрСп63=отAб6/get_Main(). отпр63;</code>
	<code>get_Main().КПрСп63.setText(get_Main().к true);</code>
<b>Абонент4</b>	
<b>Свойство</b>	<code>selectOutput</code>
<b>Класс заявки:</b>	<code>Message</code>
<b>Выход true выбирается</b>	При выполнении условия
<b>Условие</b>	<code>entity.numAbOptr==entity.numAbPol numAbonent 4</code>
<b>Свойство</b>	<code>selectOutput1</code>
<b>Класс заявки:</b>	<code>Message</code>
<b>Использовать:</b>	<code>Условия</code>
<b>Условие 0</b>	<code>entity.numAbPol==1</code>
<b>Действие при выходе 0</b>	<code>отпрAб1++;</code>
	<code>get_Main().отпр41=отпрAб1;</code>

<b>Условие 1</b>	entity.numAbPol==2
<b>Действие при выходе 2</b>	отпрAб2++; get_Main().отпр42=отпрAб2;
<b>Условие 2</b>	entity.numAbPol==3
<b>Действие при выходе 2</b>	отпрAб3++; get_Main().отпр43=отпрAб3;
<b>Условие 3</b>	entity.numAbOptr==entity.numAbPol
<b>Свойство</b>	selectOutput2
<b>Класс заявки:</b>	Message
<b>Использовать:</b>	Условия
<b>Условие 0</b>	entity.numAbPol==4
<b>Условие 1</b>	entity.numAbPol==5
<b>Действие при выходе 1</b>	отпрAб5++; get_Main().отпр45=отпрAб5;
<b>Условие 2</b>	entity.numAbPol==6
<b>Действие при выходе 2</b>	отпрAб6++; get_Main().отпр46=отпрAб6;
<b>Свойство</b>	selectOutput5
<b>Класс заявки:</b>	Message
<b>Использовать:</b>	Условия
<b>Условие 0</b>	entity.numAbOptr==1  отAб1++; get_Main().КПрCн14=отAб1/get_Main(). отпр14; get_Main().КПрCн14.setText{get_Main().к true};
<b>Действие при выходе 0</b>	
<b>Условие 1</b>	entity.numAbOptr==2  отAб2++; get_Main().КПрCн24=отAб2/get_Main(). отпр24; get_Main().КПрCн24.setText{get_Main().к true};
<b>Действие при выходе 1</b>	
<b>Условие 2</b>	entity.numAbOptr==3

<b>Действие при выходе 2</b>	отAб3++;
	get_Main().кПрCп34=отAб3/get_Main(). отпр34; get_Main().КПрCп34.setText(get_Main().к true);
<b>Условие 3</b>	entity.numAbOptr==4
<b>Свойство</b>	selectOutput6
<b>Класс заявки:</b>	Message
<b>Использовать:</b>	Условия
<b>Условие 0</b>	entity.numAbOptr==5
<b>Действие при выходе 0</b>	отAб5++;
	get_Main().кПрCп54=отAб5/get_Main(). отпр54; get_Main().КПрCп54.setText(get_Main().к true);
<b>Условие 1</b>	entity.numAbOptr==6
<b>Действие при выходе 1</b>	отAб6++;
	get_Main().кПрCп64=отAб6/get_Main().отп get_Main().КПрCп64.setText(get_Main().к true);

**Абонент5**

<b>Свойство</b>	selectOutput
<b>Класс заявки:</b>	Message
<b>Выход true выбирается</b>	При выполнении условия
<b>Условие</b>	entity.numAbOptr==entity.numAbPol numAbonent 5
<b>Свойство</b>	selectOutput1
<b>Класс заявки:</b>	Message
<b>Использовать:</b>	Условия
<b>Условие 0</b>	entity.numAbPol==1
<b>Действие при</b>	отпрAб1++;

	get_Main().отпр51=отпрAб1;
Условие 1	entity.numAbPol==2
Действие при выходе 2	отпрAб2++; get_Main().отпр52=отпрAб2;
Условие 2	entity.numAbPol==3
Действие при выходе 2	отпрAб3++; get_Main().отпр53=отпрAб3;
Условие 3	entity.numAbOptr==entity.numAbPol
Свойство	selectOutput2
Класс заявки:	Message
Использовать:	Условия
Условие 0	entity.numAbPol==4
Действие при выходе 0	отпрAб4++; get_Main().отпр54=отпрAб4;
Условие 1	entity.numAbPol==5
Условие 2	entity.numAbPol==6
Действие при выходе 2	отпрAб6++; get_Main().отпр56=отпрAб6;
Свойство	selectOutput5
Класс заявки:	Message
Использовать:	Условия
Условие 0	entity.numAbOptr==1 отAб1++;
Действие при выходе 0	get_Main().КПрCп15=отAб1/get_Main().отпр15; get_Main().КПрCп15.setText(get_Main().к true);
Условие 1	entity.numAbOptr==2 отAб2++;
Действие при выходе 1	get_Main().КПрCп25=отAб2/get_Main().отпр25; get_Main().КПрCп25.setText(get_Main().к

	true);
Условие 2	entity.numAb0ptr==3 отAb3++; get_Main().КПрСп35=отAb3/get_Main(). отпр35; get_Main().КПрСп35.setText(get_Main().к true);
Действие при выходе 2	
Условие 3	entity.numAb0ptr==4 отAb4++; get_Main().КПрСп45=отAb4/get_Main(). отпр45; get_Main().КПрСп45.setText(get_Main().к true);
Действие при выходе 3	
Свойство	selectOutput6
Класс заявки:	Message
Использовать:	Условия
Условие 0	entity.numAb0ptr==5
Условие 1	entity.numAb0ptr==6 отAb6++; get_Main().КПрСп65=отAb6/get_Main(). отпр65; get_Main().КПрСп65.setText(get_Main().к true);
Действие при выходе 1	

**Абонент6**

Свойство	selectOutput
Класс заявки:	Message
Выход true выбирается	При выполнении условия
Условие	entity.numAb0ptr==entity.numAbPol numAbonent 6
Свойство	selectOutput1
Класс заявки:	Message
Использовать:	Условия

<b>Условие 0</b>	entity.numAbPol==1
<b>Действие при выходе 0</b>	отпрAб1++; get_Main().отпр61=отпрAб1;
<b>Условие 1</b>	entity.numAbPol==2
<b>Действие при выходе 2</b>	отпрAб2++; get_Main().отпр62=отпрAб2;
<b>Условие 2</b>	entity.numAbPol==3
<b>Действие при выходе 2</b>	отпрAб3++; get_Main().отпр63=отпрAб3;
<b>Условие 3</b>	entity.numAb0ptr==entity.numAbPol
<b>Свойство</b>	selectOutput2
<b>Класс заявки:</b>	Message
<b>Использовать:</b>	Условия
<b>Условие 0</b>	entity.numAbPol==4
<b>Действие при выходе 0</b>	отпрAб4++; get_Main().отпр64=отпрAб4;
<b>Условие 1</b>	entity.numAbPol==5
<b>Действие при выходе 1</b>	отпрAб5++; get_Main().отпр65=отпрAб5;
<b>Условие 2</b>	entity.numAbPol==6
<b>Свойство</b>	selectOutput5
<b>Класс заявки:</b>	Message
<b>Использовать:</b>	Условия
<b>Условие 0</b>	entity.numAb0ptr==1
<b>Действие при выходе 0</b>	отAб1++; get_Main().кПрCп16=отAб1/get_Main().отп get_Main().КПрCп16.setText(get_Main().к true);
<b>Условие 1</b>	entity.numAb0ptr==2
<b>Действие при выходе 1</b>	отAб2++; get_Main().кПрCп26=отAб2/get_Main().отп get_Main().КПрCп26.setText(get_Main().к true);

	true);
Условие 2	entity.numAbOptr==3 отAб3++; get_Main().кПрСп35=отAб3/get_Main(). отпр35; get_Main().КПрСп35.setText(get_Main().к true);
Действие при выходе 2	
Условие 3	entity.numAbOptr==4 отAб3++; get_Main().кПрСп36=отAб3/get_Main(). отпр36; get_Main().КПрСп36.setText(get_Main().к true);
Действие при выходе 3	
Свойство	selectOutput6
Класс заявки:	Message
Использовать:	Условия
Условие 0	entity.numAbOptr==5 отAб5++; get_Main().кПрСп56=отAб5/get_Main(). отпр56; get_Main().КПрСп56.setText(get_Main().к true);
Действие при выходе 0	
Условие 1	entity.numAbOptr==6

Для того чтобы связь была между всеми абонентами и они могли бы обмениваться сообщениями, нам потребуется ещё один маршрутизатор. Однако мы не можем использовать второй элемент этого же класса, так как программно он настроен именно на наш вариант организации связи.

1. Создайте ещё класс активного объекта Маршрутизатор1.
2. Откройте объект Маршрутизатор.
3. Выделите все объекты и скопируйте их.
4. Перейдите на Маршрутизатор1 и вставьте скопированные объекты.

5. Выделите элемент `exit` и в поле Действие при выходе замените имеющийся там код следующим кодом:

```

int i;
i=entity.numAbPo;
{
    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 соответственно.
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.

## 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 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:.
4. Перетащите элемент `button`. Укажите свойства:
  - Метка: Результаты
  - Действие: `viewData.navigateTo()`
5. Перетащите элемент `button`. Укажите свойства:
  - Метка: маршрут1
  - Действие: `маршрут1.облМарш.navigateTo()`
6. Скопируйте кнопку маршрут1. Скорректируйте свойства:
  - Метка: маршрут2
  - Действие: `маршрут2.облМарш.navigateTo()`

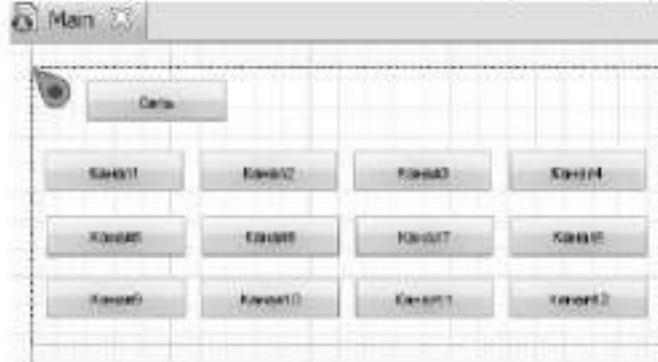


Рис. 4.19. Область просмотра облКан

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 Дополнительные 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. Сюпируйте кнопку Канал1. Вставьте одиннадцать раз. Последовательно откройте и внесите соответствующие правки в поля 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 классах активных объектов 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
	облАбонент	Действие:	get_Main().облСеть.navigate()
	viewData	Метка:	Исходные данные
	viewData	Действие:	viewData.navigate()
Канал	облКан	Метка:	Сеть
	облКан	Действие:	get_Main().облСеть.navigate()
	viewData	Метка:	Данные
	viewData	Действие:	viewData.navigate()
Маршрутизатор	облМарш	Метка:	Сеть
	облМарш	Действие:	get_Main().облСеть.navigate()
	viewData	Метка:	Данные
		Действие:	viewData.navigate()
		Метка:	Сеть
		Действие:	viewData.navigate()

		Действие: <code>get_Main().облСеть.пакеты</code>
	viewData	Метка: Маршрутизатор
		Действие: <code>облМарш.navigateTo()</code>
	облМарш	Метка: Сеть
		Действие: <code>get_Main().облСеть.пакеты</code>
	облМарш	Метка: Данные
Маршрутизатор1		Действие: <code>viewData.navigateTo()</code>
	viewData	Метка: Сеть
		Действие: <code>get_Main().облСеть.пакеты</code>
	viewData	Метка: Маршрутизатор
		Действие: <code>облМарш.navigateTo()</code>

## 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 Основные установите Фиксированное начальное число (воспроизводимые "прогоны").
5. В поле Начальное число: введите 897.
6. Перейдите на страницу Модельное время. В поле Остановить: выберите В заданное время.
7. В поле Конечное время: введите 3600000.0. Время моделирования увеличено в 1000 раз по числу прогонов модели.
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, а получено всего абонентами - 490,035. Если разделить количество полученных сообщений на количество 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](#).

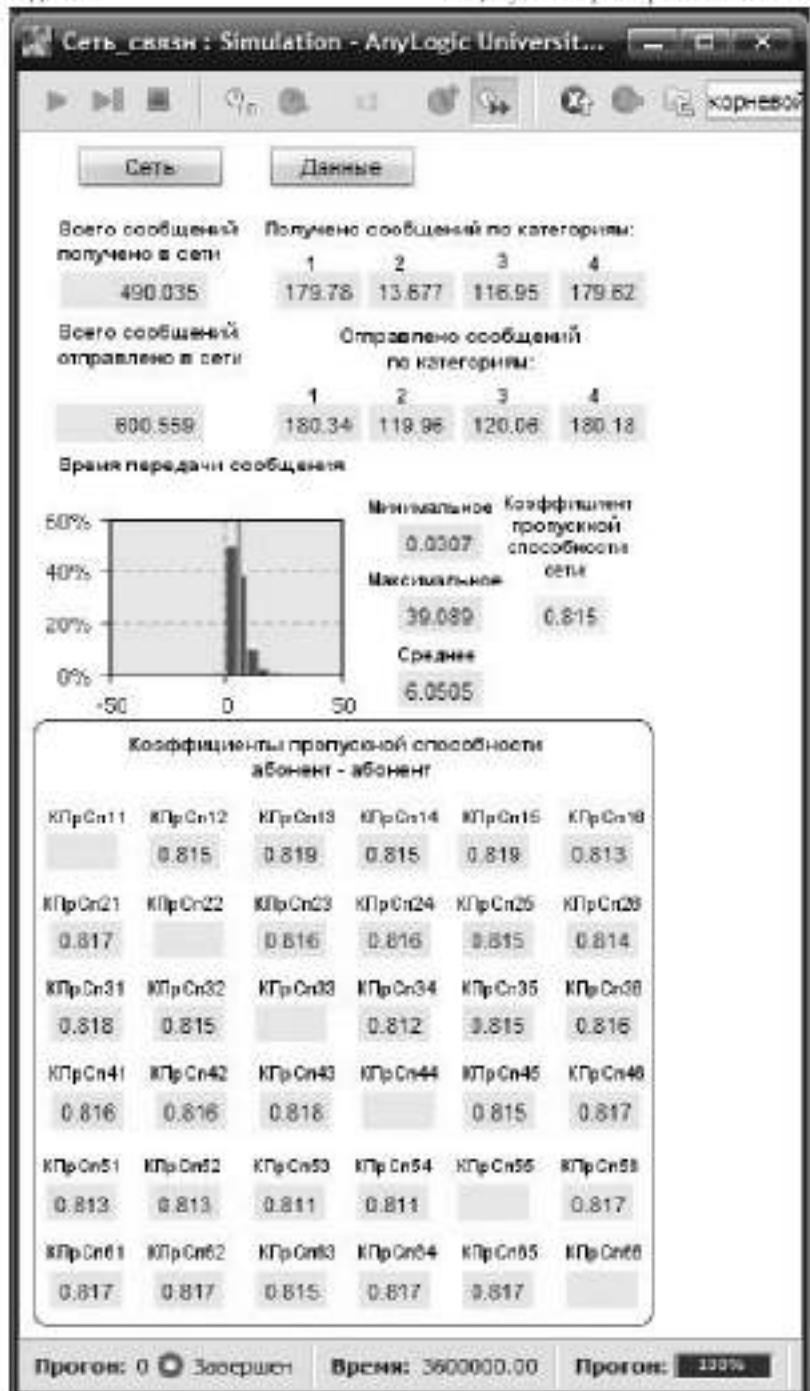


Рис. 4.20. Результаты моделирования

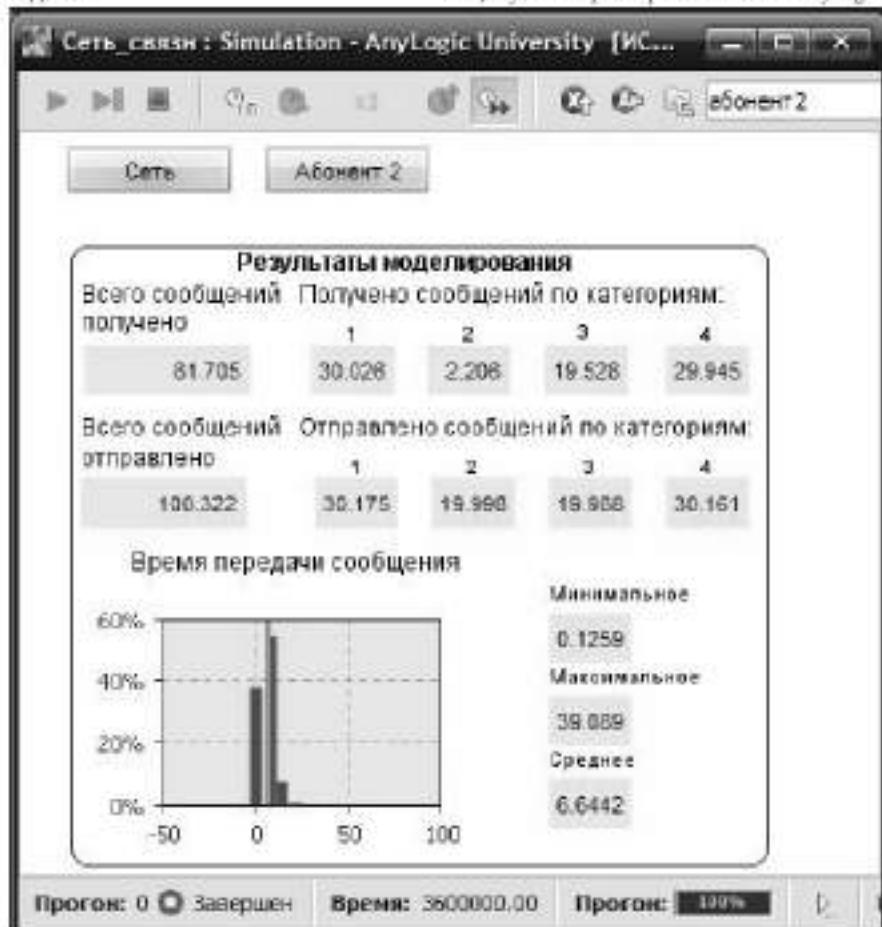


Рис. 4.21. Результаты моделирования по абоненту 2

## Модель в GPSS World

### Состав GPSS-модели

Модель в GPSS World разработана согласно постановке и включает следующие элементы:

- задание исходных данных:
  - определение матриц;
  - описание функций, задающих исходные данные;

- арифметические выражения;
- сегмент имитации поступления сообщений от источников:
  - розыгрыш абонентов-получателей сообщений;
  - розыгрыш категории сообщения и счёта сообщений;
  - розыгрыши характеристик сообщений;
- сегмент имитации работы основных каналов 1-6;
- сегмент имитации работы маршрутизатора 1;
- сегмент имитации работы маршрутизатора 2;
- сегмент имитации работы основных каналов 7-10;
- сегмент имитации работы основных каналов 11-12;
- сегмент имитации получения сообщений;
- сегмент имитации работы резервных каналов 13-18;
- сегмент имитации работы резервных каналов 19-24;
- сегмент имитации работы резервных каналов 19-24;
- сегмент имитации отказов ВК1;
- сегмент имитации отказов ВК2;
- сегмент имитации отказов каналов связи 1-6;
- сегмент имитации отказов каналов связи 7-12;
- сегмент счёта переданных и потерянных сообщений и расчёт вероятностей передачи сообщений;
- задание времени моделирования и расчёт результатов.

## GPSS-программа

Ввод данных в виде массивов организован таким образом, во-первых, чтобы в программе было меньшее количество строк, во-вторых, чтобы удобнее было использовать эти данные при написании программы модели и, в-третьих, чтобы при изменении числа элементов массива требовалась бы незначительная коррекция. Для организации такого ввода использованы функции. Например, функцией `S` задаются средние вычислительные сложности (длины) сообщений. Выборка соответствующей средней длины в зависимости от категории осуществляется по аргументу, представляющему собой параметр транзакта `P1`. В этот параметр заранее посредством розыгрыша заносится код категории.

Сообщения имитируются транзактами с параметрами, необходимыми согласно логике работы модели. Каналы связи и ВК имитируются

одноканальными устройствами (ОКУ): основные каналы связи - ОКУ1...ОКУ12, резервные каналы - ОКУ13...ОКУ24, ВК1 - ОКУ25, ВК2 - ОКУ26.

Для одновременной проверки выполнения каких-либо условий используются булевые переменные. Например, булева переменная Prov2 проверяет исправность ВК1 и наличие свободной ёмкости входного буфера, достаточной для размещения поступившего сообщения. Эти проверки можно выполнить и по отдельности, но использование булевой переменной более эффективно. Булева переменная Prov1 используется для выделения сообщений, адресованных абонентам 5 и 6.

Количество отправленных сообщений каждым абонентом каждому абоненту сети накапливается в матрице Отр, а количество полученных сообщений каждым абонентом от каждого абонента сети - в матрице Pol. По эти статистическим данным рассчитываются коэффициенты пропускной способности абонент-абонент и записываются в матрицу KPS.

Коэффициент пропускной способности сети (КПС) связи в целом сохраняется в ячейке VPerS. А среднее время передачи одного сообщения - в ячейке TimeS.

GPSS-программа приведена ниже.

; Модель функционирования сети связи

Определение матриц

Отр MATRIX ,6,6 ; Матрица для записи отправленных сообщений

Pol MATRIX ,6,6 ; Матрица для записи полученных сообщений

KPS MATRIX ,6,6 ; Матрица для записи коэффициентов пропускной с:

; Задание исходных данных

VrMod EQU 3600 ; Время моделирования,1 ед. мод. вр.=1с

KolAb EQU 6 ; Количество абонентов сообщений

Q\_ EQU 40000 ; Производительность ВК, оп/с

V\_ EQU 40000 ; Скорость передачи, бит/с

MaxKat EQU 4 ; Количество категорий сообщений

NKan EQU 12 ; Количество каналов передачи данных

KolNapr EQU 4 ; Количество направлений в маршрутизаторе

KolBK EQU 2 ; Количество ВК  
 T5 EQU 0.1 ; Время включения резервного канала  
 EmkBK1 EQU 5000000 ; Ёмкость входного буфера BK1  
 EmkBK2 EQU 5000000 ; Ёмкость входного буфера BK2  
 : Описание функций, задающих исходные данные  
 NaprM1 FUNCTION P\$NumPol,D6 ; Таблица адресов в маршрутизатор  
 1,1/2,1/3,2/4,2/5,3/6,4  
 NaprM2 FUNCTION P\$NumPol,D4 ; Таблица адресов в маршрутизатор  
 1,1/2,2/3,3/4,4  
 TOtkBK FUNCTION P4,D2 ; Среднее время между отказами ВК  
 1,3600/2,3600  
 TVosstBK FUNCTION P4,D2; Среднее время восстановления ВК  
 1,3.7/2,3.7  
 TOtkKan FUNCTION P4,D12 ; Среднее время между отказами каналов  
 1,3600/2,3600/3,3600/4,3600/5,3600/6,3600/7,3600/  
 8,3600/9,3600/10,3600/11,3600/12,3600  
 TVosstKan FUNCTION P4,D12 ; Среднее время восстановления каналов  
 1,3.2/2,3.2/3,3.2/4,3.2/5,3.2/6,3.2/7,3.2/8,3.2/  
 9,3.2/10,3.2/11,3.2/12,3.2  
 TimeAb FUNCTION P\$NumOtr,D6 ; Среднее время передачи сообщений  
 1,30/2,30/3,30/4,30/5,30/6,30  
 Kat FUNCTION RN897,D4 ; Вероятности видов категорий  
 3,1/5,2/7,3/1,4  
 S\_ FUNCTION P1,D4 ; Средние вычислительные сложности сообщений  
 1,53000/2,86000/3,66000/4,50000  
 So\_ FUNCTION P1,D4 ; Среднеквадратические отклонения вычислений  
 1,6100/2,5000/3,7000/4,500  
 EmkNaprM1 FUNCTION P\$NumNapr,D4 ; Ёмкости накопителей напротив  
 1,250000/2,250000/3,250000/4,250000  
 EmkNaprM2 FUNCTION P\$NumNapr,D4 ; Ёмкости накопителей напротив  
 5,250000/6,250000/7,250000/8,250000  
 BufAb FUNCTION P\$NumPol,D6 ; Ёмкости входных буферов абонентов  
 1,80000/2,80000/3,80000/4,80000/5,80000/6,80000  
 Prov1 BVARIABLE (P\$NumPol'E'5)'OR'(P\$NumPol'E'6)  
 Prov2 BVARIABLE (FV\*NumBK1)'AND'(P2'LE'(EmkBK1-Q\*NumBK1))  
 Prov3 BVARIABLE (FV\*NumBK2)'AND'(P2'LE'(EmkBK2-Q\*NumBK2))  
 Арифметические выражения вычисления  
 DL VARIABLE INT(NORMAL(897,FN\$\$S\_,FN\$\$So\_)) ; Длин (вычислить)  
 VrPer VARIABLE (P2/V\_) ; Времени передачи сообщения

VrObr VARIABLE P2/Q\_ ; Времени обработки сообщения  
; Сегмент имитации поступления сообщений от источников  
GENERATE ..KolAb ; Число транзактов по числу абонентов  
SAVEVALUE Num+,1;Нумерация абонента-отправителя  
ASSIGN NumOtpr,XSNum ; Номер абонента-отправителя в NumOtpr  
Soob ADVANCE (Exponential(897,0,FN\$TimeAb)) ; Время отправления (SPLIT 1,Soob ; Создание двойника сообщения  
Розыгрыш абонентов-получателей сообщений  
Met1 SAVEVALUE VerAb,(RN897/1000); Получение PPCЧ  
Met2 ASSIGN NumPol+,1 ; В PSNumPol номер абонента-получателя сообщения TEST LE X\$VerAb,(PSNumPol#(1/KolAb)),Met2  
TEST NE PSNumOtpr,P\$NumPol,Met6 ; Отправитель не себе ли отправил  
Розыгрыш категории сообщения и счёта сообщений  
Met4 ASSIGN 1,FNSKat ; В P1 код категории и счет сообщений всех категорий  
ASSIGN TimeVxod,AC1 ; Запись времени входа сообщения в сеть  
ASSIGN 10,(200+P1) ; В P10 номера X для счета отправленных сообщений  
ASSIGN 11,(P10+MaxKat) ; В P11 номера X для счета полученных сообщений  
ASSIGN 12,(P11+MaxKat) ; В P12 номера X для счета потерянных сообщений  
ASSIGN 13,(P12+MaxKat) ; В P13 номера X для записи КПС полученных сообщений  
ASSIGN 14,(P13+MaxKat) ; В P14 номера X для записи КПС потерянных сообщений  
MSAVEVALUE \*10+,1 ; Счет отправленных сообщений по категориям  
MSAVEVALUE Otpr+,P\$NumOtpr,P\$NumPol,1 ; Запись в матрицу количества сообщений  
Розыгрыш характеристик сообщений  
Met02 ASSIGN 2,V\$DL ; Занесение в P2 длины (вычислительной) строки сообщения  
ASSIGN 3,V\$VrPer ; Занесение в P3 времени передачи сообщения  
ASSIGN 8,V\$VrObr ; Занесение в P8 времени обработки сообщения  
; Сегмент имитации работы основных каналов 1-6  
OsnK1 GATE FV P\$NumOtpr,ResK1 ; Исправлен ли канал?  
ASSIGN Vsp1,P\$NumOtpr ; Запись в параметр Vsp1 номера абонента  
GATE NU P\$Vsp1;Свободен канал с номером в P\$Vsp1?  
SEIZE P\$Vsp1; Занять канал с номером в P\$Vsp1  
ADVANCE P3 ; Имитация передачи сообщения  
RELEASE P\$Vsp1 ; Освободить канал с номером в P\$Vsp1  
; Сегмент имитации работы маршрутизатора 1  
Met11 ASSIGN NumBK1,(2#N Kan+1); Запись в параметр номера маршрута  
TEST E BV\$Prov2,1,Met5 ; Исправлен ли BK1 и есть ли место в его буфере  
QUEUE PSNumBK1,P2 ; Если есть, поместить в буфер BK1  
Met14 SEIZE PSNumBK1 ; Занять BK1  
DEPART PSNumBK1,P2 ; Освободить буфер BK1

ADVANCE P8 ; Имитация обработки сообщения  
 ASSIGN NumNapr,FNSNaprM1 ; Определение номера направления м;  
 RELEASE P\$NumBK1 ; Освобождение BK1  
 TEST LE P2,(FN\$EmkNaprM1-Q\*NumNapr),Met5; Есть ли место в буфере  
 QUEUE P\$NumNapr,P2 ; Поместить в буфер направления  
 TEST NE PSNumPol5,Met21 ; Выделение сообщения для абонента 5  
 TEST NE PSNumPol6,Met21 ; Выделение сообщения для абонента 6  
 TRANSFER ,Met15 ; Отправить на маршрутизатор 2 сообщения для а  
 Met21 DEPART PSNumNapr,P2 ; Покинуть буфер направления маршрут  
 TRANSFER ,Met16 ; Отправить на каналы сообщения для абонентов !  
 Met15 DEPART PSNumNapr,P2 ; Покинуть буфер направления маршрут  
 ; Сегмент имитации работы маршрутизатора 2  
 ASSIGN NumBK2,(2#N Kan+2)  
 TEST E BV\$Prov3,1,Met5 ; Если маршрутизатор 2 исправен и есть мес-  
 QUEUE P\$NumBK2,P2 ; поместить в буфер маршрутизатора 2  
 SEIZE P\$NumBK2 ; Занять BK2  
 DEPART PSNumBK2,P2 ; Покинуть буфер маршрутизатора 2  
 ADVANCE P8 ; Имитация обработки сообщения  
 RELEASE P\$NumBK2 ; Освобождение BK2  
 TEST E P\$NumPol1,Met24  
 ; Определить номер направления маршрутизатора 2  
 ASSIGN NumNapr,(KolNapr+1)  
 TEST LE P2,(FN\$EmkNaprM2-Q\*NumNapr),Met5  
 TRANSFER ,Met25  
 Met24 TEST E PSNumPol2,Met26  
 ASSIGN NumNapr,(KolNapr+2) ; Определить номер направления мар-  
 TEST LE P2,(FN\$EmkNaprM2-Q\*NumNapr),Met5  
 TRANSFER ,Met25  
 Met26 TEST E PSNumPol3,Met27  
 ASSIGN NumNapr,(KolNapr+3) ; Определить номер направления мар-  
 TEST LE P2,(FN\$EmkNaprM2-Q\*NumNapr),Met5  
 TRANSFER ,Met25  
 Met27 TEST E PSNumPol4,Met24  
 ASSIGN NumNapr,(KolNapr+4) ; Определить номер направления мар-  
 TEST LE P2,(FN\$EmkNaprM2-Q\*NumNapr),Met5  
 Met25 QUEUE PSNumNapr,P2 ; Поместить в буфер направления марш-  
 ; Сегмент имитации работы основных каналов 7-10  
 ASSIGN Vsp1,(KolAb+PSNumPol); Определение каналов 7-10  
 DEPART PSNumNapr,P2 ; Покинуть буфер направления маршрутизатор

GATE FV P\$Vsp1,ResK2 ; Если исправен основной канал,  
 GATE NU P\$Vsp1 ; и свободен, то  
 SEIZE P\$Vsp1 ; занять его  
 ADVANCE P3 ; Передача сообщения  
 RELEASE P\$Vsp1 ; Освободить канал  
 TRANSFER ,Met17 ; Отправить для имитации получения сообщения  
 ; Сегмент имитации работы основных каналов 11-12  
 Met16 ASSIGN Vsp1,(KolAb+P\$NumPol); Определение каналов 11-12  
 GATE FV P\$Vsp1,ResK2 ; Если исправен основной канал,  
 GATE NU P\$Vsp1 ; и свободен, то  
 SEIZE P\$Vsp1 ; занять его  
 Met20 ADVANCE P3 ; Передача сообщения  
 RELEASE P\$Vsp1 ; Освободить канал  
 ; Сегмент имитации получения сообщений  
 Met17 ASSIGN Vsp3,(2#KolNapr+P\$NumPol); Запись в Vsp3 номера буфера  
 TEST LE P2,(FNS\$BufAb-Q\*Vsp3),Met5 ; Есть ли место в буфере абонента  
 QUEUE P\$Vsp3,P2 ; Поместить сообщение в буфер  
 DEPART P\$Vsp3,P2 ; Освободить буфер  
 TRANSFER ,Met10 ; Отправить для счёта полученных сообщений  
 ; Сегмент имитации работы резервных каналов 13-18  
 ResK1 ASSIGN Vsp1,(NKan+P\$NumOptr)  
 GATE NU P\$Vsp1 ; Свободен ли резервный канал?  
 TESTE X\*Vsp1,1,Met7 ; Включить резервный канал  
 ADVANCE T5 ; Включение резервного канала  
 SAVEVALUE P\$Vsp1,0 ; Признак того, что резервный канал включен  
 Met7 SEIZE P\$Vsp1 ; Занять резервный канал  
 ADVANCE P3 ; Передача  
 RELEASE P\$Vsp1 ; Освободить резервный канал  
 TRANSFER ,Met11 ; Отправить на маршрутизатор 1  
 ; Сегмент имитации работы резервных каналов 19-24  
 ResK2 ASSIGN Vsp1,(NKan+KolAb+P\$NumPol)  
 TEST NE P\$NumPol,5,Met22 ; Выделение сообщения для абонента 5  
 TEST NE P\$NumPol,6,Met22 ; Выделение сообщения для абонента 6  
 Met22 GATE NU P\$Vsp1 ; Свободен ли резервный канал?  
 TESTE X\*Vsp1,1,Met8 ; Включить резервный канал  
 ADVANCE T5 ; Включение резервного канала  
 SAVEVALUE P\$Vsp1,0 ; Признак включения ResK  
 Met8 SEIZE P\$Vsp1 ; Занять резервный канал  
 ADVANCE P3 ; Передача

RELEASE P\$Vsp1 ; Освободить резервный канал  
 TRANSFER ,Met17 ; Отправить для имитации получения сообщения  
 ; Сегмент имитации отказов BK1  
 GENERATE „1  
 ASSIGN 4,(2#N Kan+1) ; Номер BK в P4  
 Met50 ADVANCE (Exponential(897,0,FN\$TOtkBK)) ; Розыгрыш времени  
 GATE FV P4, Met50  
 FUNAVAIL P4,RE, Met117 ; Перевод BK в неисправное состояние  
 ADVANCE (Exponential(897,0,FN\$TVossiBK)) ; Имитация восстановле  
 FAVAIL P4 ; Перевод BK в исправное состояние  
 TRANSFER ,Met50 ; Отправить для розыгрыша очередного отказа  
 Met117 RELEASE PSNumBK1 ; Освобождение BK прерванным сообщ  
 TRANSFER ,Met5 ; Отправить для счета потерь  
 ; Сегмент имитации отказов BK2  
 GENERATE „1  
 ASSIGN 4,(2#N Kan+2) ; Номер BK в P4  
 Met49 ADVANCE (Exponential(897,0,FN\$TOtkBK)) ; Розыгрыш времени  
 GATE FV P4, Met49  
 FUNAVAIL P4,RE, Met115 ; Перевод BK в неисправное состояние  
 ADVANCE (Exponential(897,0,FN\$TVossiBK)) ; Имитация восстановле  
 FAVAIL P4 ; Перевод BK в исправное состояние  
 TRANSFER ,Met49 ; Отправить для розыгрыша очередного отказа  
 Met115 RELEASE PSNumBK2 ; Освобождение BK прерванным сообщ  
 TRANSFER ,Met5 ; Отправить для счета потерь  
 ; Сегмент имитации отказов каналов связи 1-6  
 GENERATE „,KolAb ; Число транзактов - по числу каналов связи  
 SAVEVALUE NumKan+,1 ; Записать в X\$NumCan последовательно 1,  
 ASSIGN 4,X\$NumKan ; Записать в P4 последовательно 1, 2, ..., KolAb  
 Met19 ADVANCE (Exponential(897,0,FN\$TOtkKan)) ; Розыгрыш времен  
 GATE FV P4, Met19  
 FUNAVAIL P4,RE, Met112 ; Перевод канала в неисправное состояние  
 ASSIGN NumKan,(N Kan+P4)  
 SAVEVALUE P\$NumKan,1 ; Признак включения резервного канала  
 ADVANCE (Exponential(47,0,FN\$TVossiKan)) ; Имитация восстановлен  
 FAVAIL P4 ; Перевод в исправное состояние  
 TRANSFER ,Met19 ; Отправить для розыгрыша очередного отказа  
 Met112 RELEASE PSVsp1 ; Освобождение канала с номером в P\$Vsp  
 TRANSFER ,Met5 ; Отправить для счета потерь  
 ; Сегмент имитации отказов каналов связи 7-12

```

GENERATE „KolAb ; Число транзактов - по числу каналов связи
SAVEVALUE NumKan+,1 ; Записать в X$NumCan последовательно 1, :
ASSIGN 4,(X$NumKan+KolAb) ; Записать в P4 последовательно KolA
Met23 ADVANCE (Exponential(897,0,FN$TOtkKan)) ; Розыгрыш временя
GATE FV P4, Met23
FUNAVAIL P4,RE,Met113 ; Перевод канала в неисправное состояние
ASSIGN NumKan,(NKan+P4)
SAVEVALUE P$NumKan,1 ; Признак включения резервного канала
ADVANCE (Exponential(47,0,FN$TVosstKan)) ; Имитация восстановления
FAVAIL P4 ; Перевод в исправное состояние
TRANSFER ,Met23 ; Отправить для розыгрыша очередного отказа
Met113 RELEASE PSVsp1 ; Освобождение канала с номером в P$Vsp
TRANSFER ,Met5 ; Отправить для счета потерь
Met5 TERMINATE ; Уничтожение сообщений, отправленных самим же
; Сегмент счета переданных и потерянных сообщений и расчет вероятности
Met10 SAVEVALUE *11+,1 ; Счет в ячейке с номером в P11 полученных
SAVEVALUE *13,(X*11/X*10) ; Расчет и сохранение в ячейке с номером
MSAVEVALUE Pol+,PSNumPol,P$NumOtrr,1 ; Счёт и запись в матрицу
MSAVEVALUE KPS,P$NumOtrr,P$NumPol, (MX$Po)/(PSNumPol,PSNumKan)
TERMINATE
Met5 SAVEVALUE *12+,1 ; Счет в ячейке с номером в P12 потерянных
TERMINATE
; Задание времени моделирования и расчёт результатов
GENERATE VrMod ; Задание времени моделирования
TEST L X$Prog,TG1, Met30 ; Если X$Prog < TG1,
SAVEVALUE Prog,TG1 ; то X$Prog = TG1
Met30 TEST E TG1,1, Met32 ; Если TG1=1, расчёт и сохранение результата
SAVEVALUE TimeS,(X$TimeSum/N$Met10) ; Расчёт и сохранение в ячейку
SAVEVALUE VPerS,(N$Met10/N$Met4) ; Расчет и сохранение в ячейку
SAVEVALUE VPotS,(N$Met5/N$Met4) ; Расчет и сохранение в ячейку
Met32 TERMINATE 1
START 1000

```

## Интерпретация результатов моделирования

Всего выполнено 6 экспериментов. Изменение параметров сети связи проводилось так же, 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 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 ( $\Delta_{11} \dots \Delta_{61}$ ) отличаются на 0...0,001, а среднее время передачи одного сообщения ( $\Delta_{12} \dots \Delta_{62}$ ) - на 0,075...1,085 с. Коэффициенты загрузки вычислительных комплексов также отличаются на 0...0,020, а коэффициенты загрузки основных каналов - на 0,000...0,001.

В четвёртом и пятом экспериментах в связи с уменьшением ёмкостей входных буферов BK1 и BK2 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 5 мин.

Таблица 4.11. Показатели функционирования сети связи

Показатели	GPSS World	AnyLogic
1) Согласно постановке задачи		
Коэффициент пропускной способности сети связи	0,816	0,815
Среднее время передачи одного сообщения	$\Delta_{12} =  0,816 - 0,815  = 0,001$ 6,127	6,052
Коэффициент загрузки BK1	$\Delta_{12} =  6,127 - 6,052  = 0,075$ 0,255	0,255

Коэффициент загрузки BK2	0,17	0,17
Средний коэффициент загрузки основных каналов	0,042	0,042
Среднее количество полученных / отправленных сообщений	489,930/600,263	490,195/601,092
2) emkBuferVx = ... = emkBuferVx = 70000		
Коэффициент пропускной способности сети связи	0,741	0,741
Среднее время передачи одного сообщения	$\Delta_{21} =  0,741 - 0,741  = 0,000$ 6,127	5,867
Коэффициент загрузки BK1	0,255	0,255
Коэффициент загрузки BK2	0,17	0,17
Средний коэффициент загрузки основных каналов	0,043	0,043
Среднее количество полученных / отправленных сообщений	444,728/600,263	444,863/600,347
3) emkBuferVx = ... = emkBuferVx = 60000		
Коэффициент пропускной способности сети связи	0,599	0,599
Среднее время передачи одного сообщения	$\Delta_{31} =  0,599 - 0,599  = 0,000$ 6,132	5,66
Коэффициент загрузки BK1	0,255	0,256
Коэффициент загрузки BK2	0,17	0,17
Средний коэффициент загрузки основных каналов	0,043	0,043
Среднее количество полученных / отправленных сообщений	359,640/600,584	360,251/601,104
4) emkBufer1 = 4 000 000		
Коэффициент пропускной способности сети связи	0,599	0,6
Среднее время передачи одного	$\Delta_{41} =  0,599 - 0,600  = 0,001$ 6,132	5,65

	$\Delta_{42} =   6,132 - 5,650   = 0,482$	
Коэффициент загрузки BK1	0,255	0,251
Коэффициент загрузки BK2	0,17	0,169
Средний коэффициент загрузки основных каналов	0,043	0,042
Среднее количество полученных / отправленных сообщений	359,640/600,584	359,651/599,302
5) emkBufer1 = 3 000 000		
Коэффициент пропускной способности сети связи	0,599	0,598
Среднее время передачи одного сообщения	$\Delta_{51} =   0,599 - 0,598   = 0,001$ 6,132	5,648
Коэффициент загрузки BK1	0,255	0,256
Коэффициент загрузки BK2	0,17	0,17
Средний коэффициент загрузки основных каналов	0,043	0,044
Среднее количество полученных / отправленных сообщений	359,640/600,584	360,114/601,495
6) emkBuferVx = ... = emkBuferVx = 100000, emkBufer1 = 6 000 000		
Коэффициент пропускной способности сети связи	0,996	0,996
Среднее время передачи одного сообщения	$\Delta_{61} =   0,996 - 0,996   = 0,000$ 6,155	7,237
Коэффициент загрузки BK1	0,362	0,382
Коэффициент загрузки BK2	0,238	0,254
Средний коэффициент загрузки основных каналов	0,064	0,064
Среднее количество полученных / отправленных сообщений	848,296/851,388	896,072/899,333

# Модель функционирования системы связи

## Модель в AnyLogic

### Постановка задачи

На дежурстве находятся  $n_1$  средств связи (СС)  $n_2$  типов ( $n_{21} + n_{22} + \dots + n_{2n_2} = n_2$ ) в течение  $n_3$  часов.

Каждое СС может в любой момент времени выйти из строя. Интервалы времени  $T_{21}, T_{22}, \dots, T_{2n_2}$  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 -  $n_4$ .

Ремонт неисправных СС производят  $n_5$  мастеров. Время  $T_1, T_2, \dots, T_{n_2}$  ремонта случайное и зависит от типа СС, но не зависит от того, какой мастер это СС ремонтирует.

Прибыль от СС, находящихся на дежурстве, составляет  $S_1$  денежных единиц в час. Потасовой убыток при отсутствии на дежурстве одного СС -  $S_2, \dots, S_{2n_2}$  денежных единиц в час. Оплата мастера за ремонт неисправного СС -  $S_{31}, S_{32}, \dots, S_{3n_2}$  денежных единиц в час соответственно.

Затраты на содержание одного резервного СС составляют  $S_4$  денежных единиц в час.

### 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.

## 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;
- timeMeanOtkaz - 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:

- КСС1 ... КСС5 - количество СС первого ... пятого типов соответственно;
- КССР1 ... КССР5 - количество резервных СС первого ... пятого типов соответственно.

По этим же данным определяются количества всех СС по типам КолКСС1 ... КолКСС5, а также общее количество СС всех типов КолКСС.

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

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

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

Параметры 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:

- $PribCC1 \dots PribCC5, SumPribil$  - абсолютные величины ожидаемой прибыли по каждому типу СС и в целом;
- $KoefPribCC1 \dots KoefPribCC5, KoefPribil$  - 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)*  
get_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())*  
get_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+ZatrResCC1+UbitokCC1).

Относительный коэффициент прибыли равен:

KoefPribCC1=PribCC1/DoxMaxCC1.

Показатели в целом за систему связи:

SumPrib=SumDoxDeg-(SumZatrRes+SumZatrRem+SumUbitok),

KoefPrib=SumPrib/SumDoxMax,

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

## 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. Задайте имя новой модели. В поле Имя модели введите ComSystem. Выберите каталог для сохранения файлов модели.
3. Щелкните кнопку Далее. Откроется вторая страница Мастера создания модели. Выберите Начать создание модели "с нуля". Щелкните кнопку Далее.

### Область просмотра

Используем три области просмотра. В первой области просмотра разместим объекты и элементы сегмента Постановка на дежурство, во второй - сегмента Имитация дежурства СС и

сегмента имитации функционирования ремонтного подразделения, в третьей - сегмента Статистика.

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

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

1. В Палитре выделите Презентация. Перетащите элемент Область просмотра.
2. Перейдите на страницу Основные панели Свойства.
3. В поле Имя : введите Postanovka.
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. Перейдите на страницу Дополнительные панели Свойства. Введите в поля X : 0, Y : 0, Ширина : 700, Высота : 970.
7. Перетащите элемент Скруглённый прямоугольник. Оставьте имя, предложенное системой. В нём мы разместим объекты сегмента Постановка на дежурство.
8. Перейдите на страницу Дополнительные панели Свойства. Введите в поля X : 38, Y : 62, Ширина : 642, Высота : 268.

### Ввод исходных данных

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

Организуйте ввод исходных данных для сегмента Постановка на дежурство.

1. Перетащите элемент Прямоугольник на элемент Область

просмотра, если вы хотите видеть данные в ходе моделирования. Если нет, поместите этот элемент вне области просмотра.

2. Перейдите на страницу Дополнительные панели Свойства. Введите в поля X : 20, Y : 350, Ширина : 390, Высота : 300.
3. Перетащите элемент `text` и на странице Основные панели Свойства в поле Текст: введите `Initial_data_PD` (здесь PD - постановка на дежурство).
4. В Палитре выделите Основная. Перетащите элементы Параметр и Простая переменная на элемент с именем `Initial_data_PD` и разместите их так, как показано на [рис. 5.3](#).
5. Значения свойств установите согласно [табл. 5.1](#).

Простые переменные с именами `Ko1CC1..Ko1CC5` - количество СС по типам, а `Ko1CC` - количество СС всех типов. Эти переменные мы будем вычислять по исходным значениям `KCC1 .. KCC5` и `KCCP1 .. KCCP5` и использовать при генерации равного значению `Ko1CC` заявок, имитирующих СС. Количество СС изменять можно только перед началом моделирования, так как заявки, имитирующие СС, генерируются только один раз.

#### Initial\_data\_PD

<input checked="" type="checkbox"/> Ko1CC1	<input checked="" type="checkbox"/> zatrResCC1	<input checked="" type="checkbox"/> doxDegCC1	<input checked="" type="checkbox"/> ubitokCC1
<input checked="" type="checkbox"/> Ko1CC2	<input checked="" type="checkbox"/> zatrResCC2	<input checked="" type="checkbox"/> doxDegCC2	<input checked="" type="checkbox"/> ubitokCC2
<input checked="" type="checkbox"/> Ko1CC3	<input checked="" type="checkbox"/> zatrResCC3	<input checked="" type="checkbox"/> doxDegCC3	<input checked="" type="checkbox"/> ubitokCC3
<input checked="" type="checkbox"/> Ko1CC4	<input checked="" type="checkbox"/> zatrResCC4	<input checked="" type="checkbox"/> doxDegCC4	<input checked="" type="checkbox"/> ubitokCC4
<input checked="" type="checkbox"/> Ko1CC5	<input checked="" type="checkbox"/> zatrResCC5	<input checked="" type="checkbox"/> doxDegCC5	<input checked="" type="checkbox"/> ubitokCC5
<input checked="" type="checkbox"/> Ko1CC	<input checked="" type="checkbox"/> NumCC		

## Рис. 5.3. Размещение элементов Параметр и Простая переменная

## Имитация поступления средств связи

- На Область просмотра мы уже перетащили Скругленный прямоугольник. На нём мы будем размещать, как отмечалось ранее, объекты сегмента Постановка на дежурство.
- Перетащите на него элемент `text` и на странице Основные панели Свойства в поле Текст: введите Постановка на дежурство. Поместите этот текст посередине в верхней части элемента Скругленный прямоугольник.
- Перетащите элемент Прямоугольник на Область просмотра. Перейдите на страницу Дополнительные панели Свойства. Введите в поля X: 50, Y: 100, Ширина: 160, Высота: 140.
- Перетащите элемент `text` на Прямоугольник и на странице Основные панели Свойства в поле Текст: введите Имитация поступления СС.
- Перетащите объект `source` на Прямоугольник. Для записи и хранения параметров СС в дополнительные поля заявок необходимо создать нестандартный класс заявки. Создайте класс заявки `ComFacility`.

Таблица 5.1. Свойства элементов на Initial\_data\_PD

Имя	Тип	Значение по умолчанию	Отображать имя
KolCC1	int	0	
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	Установить флагок во всех элементах
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	
ВремяРабСист	double 1000	

6. В панели Проект щелкните правой кнопкой мыши элемент модели верхнего уровня дерева и выберите Создать/Java класс.
7. Появится диалоговое окно Новый Java класс. В поле Имя: введите имя нового класса ComFacility.
8. В поле Базовый класс: выберите из выпадающего списка Entity в качестве базового класса. Щелкните кнопку Далее.
9. Появится вторая страница Мастера создания Java класса. Добавьте следующие поля Java класса:

```
int tipCC;
double timeMeanRem;
double nach;
double nach1;
double timeMeanOtkaz;
```

10. Оставьте выбранными флагки Создать конструктор и Создать метод `toString ()`.
11. Щелкните кнопку Готово. Вы увидите редактор кода и в автоматически созданный код вашего Java класса. Закройте код.
12. Выделите объект `source`. На странице Основные панели

**Свойства** уберите флажок Отображать имя. В полях Класс заявки: и Новая заявка Entity замените ComFacility.

Установите:

- Заявки прибывают согласно Интенсивности.
- Интенсивность прибытия 1
- Количество заявок, прибывающих за один раз 1
- Ограниченнное количество прибытий установите флажок
- Максимальное количество прибытий 1

В поле Действие при выходе введите Java код:

```
KoICC1=degystvo.KCC1+degystvo.KCCP1;
degystvo.DoxMaxCC1=
round((degystvo.KCC1*doxDegCC1)*ВремяРабСист*100);
degystvo.DoxMaxCC1=degystvo.DoxMaxCC1/100;
degystvo.ZatrResCC1=
round((degystvo.KCCP1*zatrResCC1)*ВремяРабСист*100);
degystvo.ZatrResCC1=degystvo.ZatrResCC1/100;
KoICC2=degystvo.KCC2+degystvo.KCCP2;
degystvo.DoxMaxCC2=
round((degystvo.KCC2*doxDegCC2)*ВремяРабСист*100);
degystvo.DoxMaxCC2=degystvo.DoxMaxCC2/100;
degystvo.ZatrResCC2=
round((degystvo.KCCP2*zatrResCC2)*ВремяРабСист*100);
degystvo.ZatrResCC2=degystvo.ZatrResCC2/100;
KoICC3=degystvo.KCC3+degystvo.KCCP3;
degystvo.DoxMaxCC3=
round((degystvo.KCC3*doxDegCC3)*ВремяРабСист*100);
degystvo.DoxMaxCC3=degystvo.DoxMaxCC3/100;
degystvo.ZatrResCC3=
round((degystvo.KCCP3*zatrResCC3)*ВремяРабСист*100);
degystvo.ZatrResCC3=degystvo.ZatrResCC3/100;
KoICC4=degystvo.KCC4+degystvo.KCCP4;
degystvo.DoxMaxCC4=
round((degystvo.KCC4*doxDegCC4)*ВремяРабСист*100);
degystvo.DoxMaxCC4=degystvo.DoxMaxCC4/100;
```

```

degystvo.ZatrResCC4=
round((degystvo.KCCP4*zatrResCC4)*ВремяРабСист*100);
degystvo.ZatrResCC4=degystvo.ZatrResCC4/100;
KoICC5=degystvo.KCC5+degystvo.KCCP5;
degystvo.DoxMaxCC5=
round((degystvo.KCC5*doxDegCC5)*ВремяРабСист*100);
degystvo.DoxMaxCC5=degystvo.DoxMaxCC5/100;
KoICC=KoICC1+KoICC2+KoICC3+KoICC4+KoICC5;
degystvo.ZatrResCC5=
round((degystvo.KCCP5*zatrResCC5)*ВремяРабСист*100);
degystvo.ZatrResCC5=degystvo.ZatrResCC5/100;
degystvo.SumDoxMax=degystvo.DoxMaxCC1+
degystvo.DoxMaxCC2+degystvo.DoxMaxCC3+
degystvo.DoxMaxCC4+degystvo.DoxMaxCC5;
degystvo.SumZatrRes=degystvo.ZatrResCC1+
degystvo.ZatrResCC2+degystvo.ZatrResCC3+
degystvo.ZatrResCC4+degystvo.ZatrResCC5;

```

Введённым кодом определяется количество СС всех типов, включая и резервные средства связи. Эти данные необходимы в последующем в объекте `split`. Количество СС как исходные данные будут размещены на активном классе `Degystvo`, который мы создадим позже, поэтому в коде используется доступ к ним в виде, например, `degystvo.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 расчётом 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;
```

13. Добавьте объекты `split` и `sink` (Рис. 5.4).

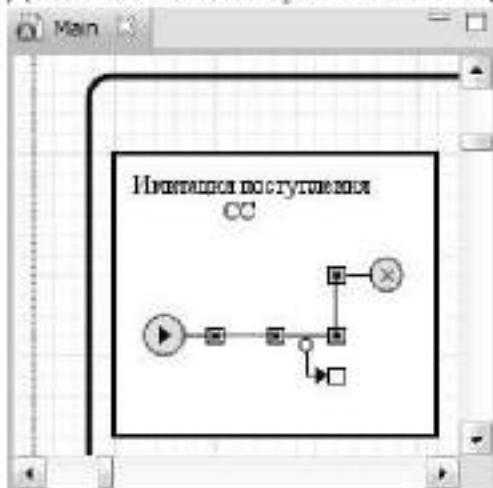


Рис. 5.4. Добавлены объекты `split` и `sink`

14. Выделите объект `split`. Установите свойства как на рис. 5.5. Так как копии не унаследуют свойств от оригинала, поэтому в поле **Действие при выходе** копии введите Java код:

```
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;
```

В поле entity.tipCC запоминается соответствующий код типа СС, например, entity.tipCC = 5, который необходим для нормального функционирования модели, то есть отличия СС по типам.

15. Объект sink уничтожает заявку-оригинал.



Рис. 5.5. Элемент source с установленными свойствами

### 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 Enterprise Library на диаграмму класса Main. Соедините их так, как показано на рис. 5.6.
2. Установите на странице Основные панели Свойства свойства объектов selectOutput согласно табл. 5.2(при использовании

объекта `selectOutput5` условия разделения СС по типам останутся такими же).

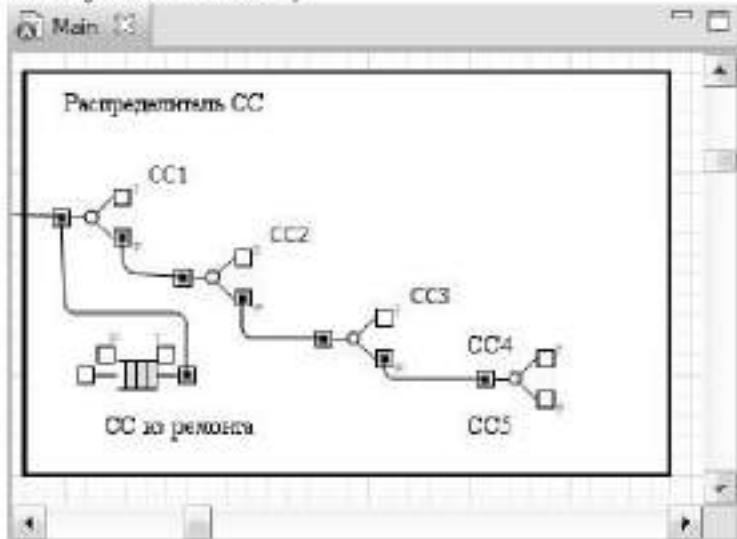


Рис. 5.6. Добавлены объекты `SelectOutput` и `queue`

Таблица 5.2.

Имя	Отображать имя	Класс заявки:	Выход <code>true</code> выбирается	Условие
CC1		ComFacility		
CC2 Установите флагки		ComFacility	При выполнении условия	entity.tipCC=
CC3		ComFacility		entity.tipCC=
CC4		ComFacility		entity.tipCC=

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. Установите Вместимость 100 и флажок Включить сбор статистики.
6. Перетащите из палитры Презентация три элемента `text` и в соответствующих полях Текст: введите текст, как на [рис. 5.6](#).

## Создание нового класса 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 выполнить "связь" между сегментом Постановка на дежурство и сегментом Имитация дежурства, создадим новый класс активного объекта.

1. На панели Проект щелкните правой кнопкой мыши Main, с которым вы работаете в данный момент, и выберите Создать / Класс активного объекта из контекстного меню.
2. Откроется окно Новый класс активного объекта.
3. Задайте в поле Имя: имя нового класса Degurstvo.
4. Если нужно, в поле Описание: введите описание сущности, моделируемой этим классом.
5. Щелкните кнопку Готово.

Создайте область просмотра на диаграмме класса Degurstvo для размещения элементов сегмента Имитация дежурства и текущих результатов моделирования.

6. В Палитре выделите Презентация. Перетащите элемент Область просмотра.
7. Перейдите на страницу Основные панели Свойства.
8. В поле Имя: введите degur.
9. Перейдите на страницу Дополнительные панели Свойства. Введите в поля X: 40, Y: 0, Ширина: 730, Высота: 730.
10. Задайте, как будет располагаться область просмотра относительно ее якоря, с помощью элемента управления Выравнивать по: Верхн. левому углу.
11. Выберите режим масштабирования из выпадающего списка Масштабирование: Подогнать под окно.



Рис. 5.7. Сегмент Постановка на дежурство

### Создание элемента нового класса активного объекта

Созданный новый класс активного объекта `Degyurstvo` 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. Установите только флагки На верхнем уровне и На презентации.
2. Перейдите на страницу Дополнительные панели Свойства. Введите в поля X: 110, Y: 150, Ширина: 100, Высота: 140.
3. Перетащите шесть элементов Порт  из палитры Основная и разместите так, как показано на [рис. 5.8](#). Автоматически они будут объединены прямоугольником (с пунктирными линиями) и появится надпись Значок.
4. Возвратитесь на диаграмму класса Main.

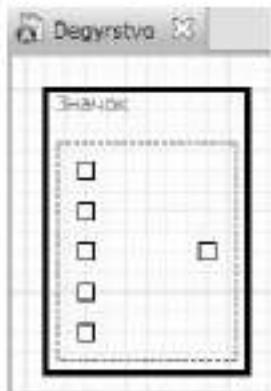


Рис. 5.8. Добавлены на Degyrstvo шесть портов

- На панели Проект выделите *Degyrstvo*, перетащите элемент класса и соедините так, как на [рис. 5.7](#). При этом следует иметь в виду, что положение портов на элементе класса *Degyrstvo* изменить нельзя. Это можно сделать лишь на самом классе.

#### 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:

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

Чтобы перейти к другой области просмотра в режиме выполнения модели:

- Щелкните правой кнопкой мыши в области обрисовки окна презентации, выберите пункт контекстного меню Область и затем выберите из списка, к какой именно области просмотра вы

хотите перейти.

2. Или же щелкните кнопку панели инструментов Показать область... и выберите из выпадающего списка, к какой именно области просмотра вы хотите перейти (эта кнопка принадлежит секции панели инструментов Вид, и возможно, чтобы она стала видна, вам нужно будет вначале показать эту секцию панели инструментов).

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

1. В Палитре выделите Презентация. Перетащите элемент `text`, разместите и введите в поле Текст: Постановка на дежурство, как на [рис. 5.7](#).
2. Перетащите второй элемент `text`, разместите и введите в поле Текст: Имитация дежурства. Текущие результаты.
3. На панели Свойства выделите Динамические и в поле Действие по щелчку: введите следующий Java код:  

```
degyrstvo.degyrt.navigateTo();
```
4. Проделайте то же для Статистика. Введите Java код:  

```
degyrstvo.statistika.navigateTo();
```

## Сегмент Имитация дежурства

### Ввод исходных данных

Организуйте ввод исходных данных для сегмента Имитация дежурства. Для ввода исходных данных используйте также элемент Параметр.

1. Перетащите элемент Прямоугольник. На нём мы разместим элементы для ввода исходных данных.
2. Оставьте имя, предложенное системой, а также установленным

только один флажок На презентации.

- Перейдите на страницу Дополнительные панели Свойства. Введите в поля X: 50, Y: 790, Ширина: 570, Высота: 190.
- Перетащите элемент text и на странице Основные панели Свойства в поле Текст; введите Initial\_data\_D (здесь D -дежурство).
- В Палитре выделите Основная. Перетащите элементы Параметр на элемент с именем Initial\_data\_D и разместите их так, как показано на [рис. 5.9](#).
- На странице Основные панели Свойства каждого элемента Параметр установите свойства согласно [табл. 5.3](#).

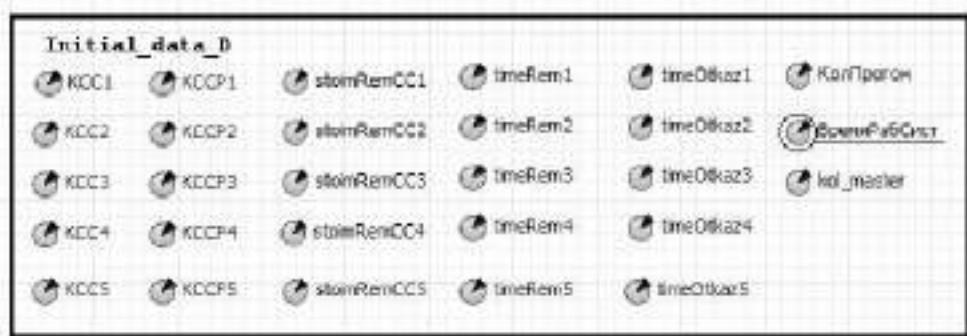


Рис. 5.9. Размещение элементов Параметр для ввода данных

Таблица 5.3. Свойства элементов Параметр на Initial\_data\_D

Имя	Тип	Значение по умолчанию	Отображать имя
KCC1	int	55	
KCC2	int	100	
KCC3	int	60	
KCC4	int	45	
KCC5	int	60	
KCCP1	int	2	
KCCP2	int	4	
KCCP3	int	4	

KCCP4	int	3
KCCP5	int	4
stoimRemCC1	double	17
stoimRemCC2	double	18
stoimRemCC3	double	16
stoimRemCC4	double	20
stoimRemCC5	double	21
timeRem1	double	6,5
timeRem2	double	4,2
timeRem3	double	2,8
timeRem4	double	3
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

Установить флагок во  
всех элементах

## 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).

Затраты	Доходы	ResultsModeling
ZatResCC1	ZatResCC1	ResCC1
ZatResCC2	ZatResCC2	ResCC2
ZatResCC3	ZatResCC3	ResCC3
ZatResCC4	ZatResCC4	ResCC4
ZatResCC5	ZatResCC5	ResCC5
SumZatRes	SumZatRes	SumRes
	DoxMaxCC1	IoeffResCC1
	DoxMaxCC2	IoeffResCC2
	DoxMaxCC3	IoeffResCC3
	DoxMaxCC4	IoeffResCC4
	DoxMaxCC5	IoeffResCC5
	DoxDegCC1	UeffResCC1
	DoxDegCC2	UeffResCC2
	DoxDegCC3	UeffResCC3
	DoxDegCC4	UeffResCC4
	DoxDegCC5	UeffResCC5
	PriCC1	UeffPriCC1
	PriCC2	UeffPriCC2
	PriCC3	UeffPriCC3
	PriCC4	UeffPriCC4
	PriCC5	UeffPriCC5
	PriBil	UeffPriBil
		SumBil

Рис. 5.10. Элементы Простая переменная для выводов результатов моделирования

1. Перетащите элемент Прямоугольник.
2. Перейдите на страницу Дополнительные панели Свойства. Введите в поля X : 54, Y : 436, Ширина : 696, Высота : 278.
3. Результаты разбиты на две группы: затраты и доходы. Перетащите два элемента text и на странице Основные панели Свойства в поле Текст: введите Затраты и Доходы соответственно.
4. В Палитре выделите Основная. Перетащите элементы Простая переменная. Разместите их, как показано на рис. 5.10.
5. У всех переменных установите флагки Отображать имя и тип double.

### 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. Перейдите на страницу Дополнительные панели Свойства. Введите в поля X : 60, Y : 60, Ширина : 600, Высота : 350.
3. Перетащите элемент Прямоугольник. На нём мы разместим элементы, непосредственно имитирующие дежурство СС.

4. Перейдите на страницу Дополнительные панели Свойства. Введите в поля X : 250, Y : 70, Ширина : 170, Высота : 330.
5. Перетащите ещё один элемент Прямоугольник для размещения элементов, имитирующих ремонтное подразделение СС.
6. Перейдите на страницу Дополнительные панели Свойства. Введите в поля X : 440, Y : 70, Ширина : 190, Высота : 330.
7. Перетащите (или введя один элемент, остальные подобные ему скопируйте) на диаграмму класса `Degystv` последовательно: пять объектов `queue`, пять объектов `delay`, один объект `queue`, один объект `delay` и соедините их так, как показано на [рис. 5.11](#).
8. Перетащите три элемента `text` и введите названия в соответствующие поля Текст: согласно [рис. 5.11](#).
9. На странице Основные панели Свойства каждого объекта установите свойства согласно [табл. 5.4](#).

Замечание. Поскольку на входах каждого из объектов `delay` с именами `degCC1` ... `degCC5` стоят объекты `queue` с именами `rez1` ... `rez5` соответственно, то возникает желание поставить такие же объекты на выходах `delay`. Однако это приведёт к неправильной работе модели: некоторые СС не смогут поступать в ремонтное подразделение.

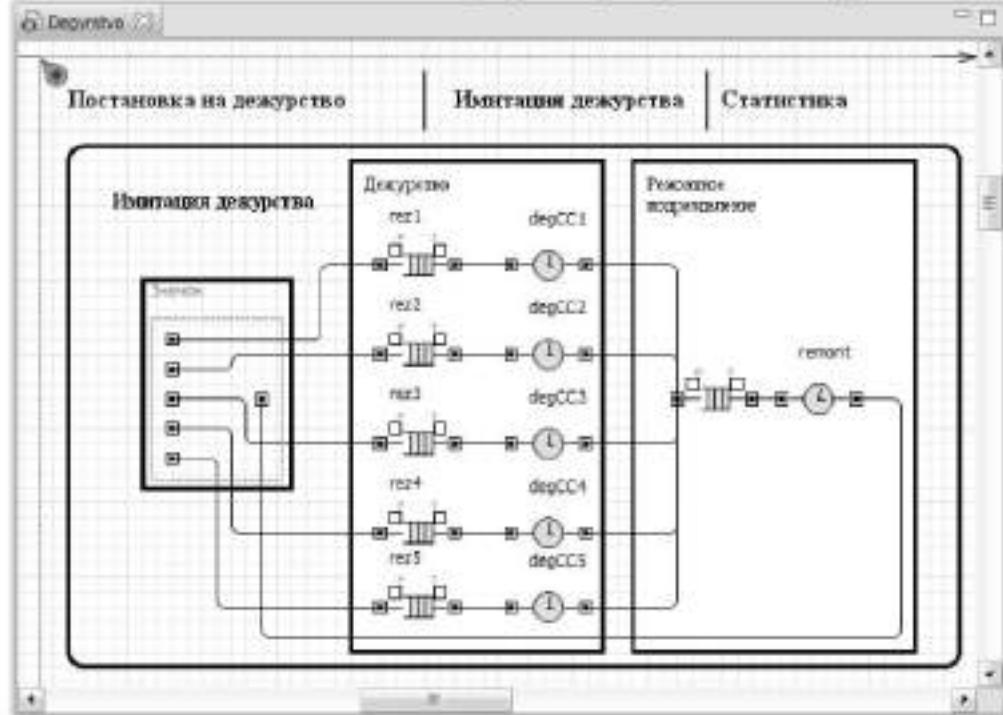


Рис. 5.11. Сегмент Имитация дежурства

Таблица 5.4. Объекты сегмента Имитация дежурства и их описание

Имя	Вместимость	Действие
rez1	KCCP1	entity.timeOut
rez2	KCCP2	entity.timeOut
rez3	KCCP3	entity.timeOut
rez4	KCCP4	entity.timeOut
rez5	KCCP5	entity.timeOut
очРем	Максимальная	

Имя	Время задержки	Выход
degCC1	exponential(entity.timeOtkaz)	KCC1
degCC2	exponential(entity.timeOtkaz)	KCC2
degCC3	exponential(entity.timeOtkaz)	KCC3

	degCC4 exponential(entity.timeOtkaz)	KCC4
	degCC5 exponential(entity.timeOtkaz)	KCC5
	remont exponential(entity.timeMeanRem) 3	
Имя	Действие при выходе	
degCC1	DoxDegCC1+=(time()-entity.nach1)*get_Main(); entity.timeMeanRem=1/timeRem1;	
degCC2	DoxDegCC2+=(time()-entity.nach1)*get_Main(); entity.timeMeanRem=1/timeRem2;	
degCC3	DoxDegCC3+=(time()-entity.nach1)*get_Main(); entity.timeMeanRem=1/timeRem3;	
degCC4	DoxDegCC4+=(time()-entity.nach1)*get_Main(); entity.timeMeanRem=1/timeRem4;	
degCC5	DoxDegCC5+=(time()-entity.nach1)*get_Main(); entity.timeMeanRem=1/timeRem5;	

Кроме свойств, указанных в табл. 5.4, нужно также:

- во всех объектах в поле Класс заявки: Entity заменить ComFacility;
- для всех объектов поставить флаги: 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();
  - Действие 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);}

```

### Переключение между областями просмотра

1. В Палитре выделите Презентация. Перетащите элемент `text`, разместите и введите в поле Текст: Постановка на дежурство, как на [рис. 5.11](#).

2. На панели Свойства выделите Динамические и в поле Действие по щелчку: введите следующий Java код:

```
get_Main().Postanovka.navigateTo();
```

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

4. На панели Свойства выделите Динамические и в поле Действие по щелчку: введите следующий Java код:

```
Degyr.navigateTo();
```

5. Проделайте то же для Статистика. Введите Java код:

```
statistika.navigateTo();
```

## 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](#)).

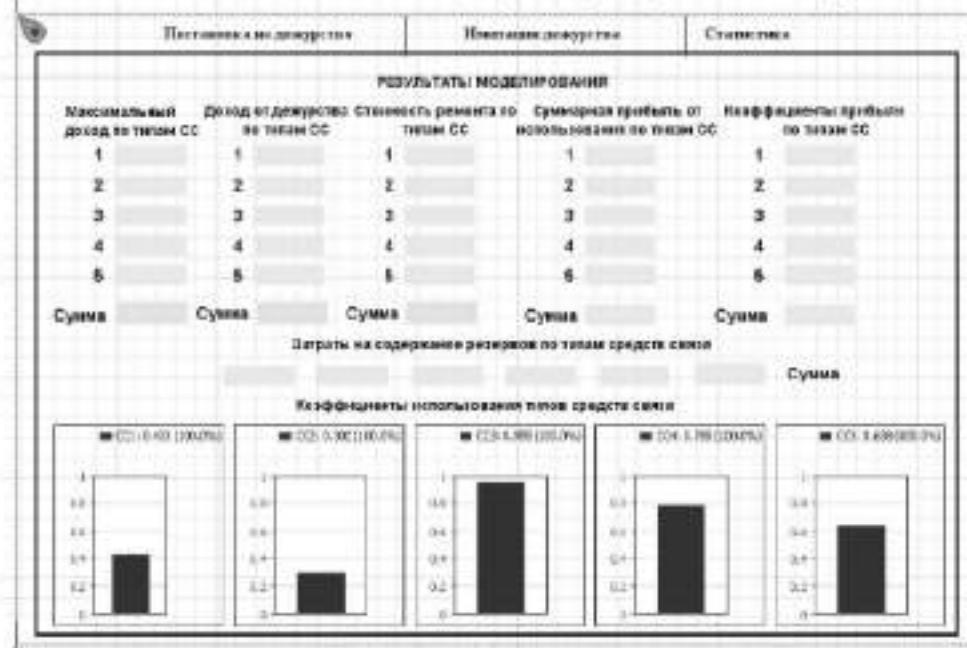


Рис. 5.12. Сегмент Статистика

1. Создайте область просмотра для размещения элементов сегмента Статистика.
2. В Палитре выделите Презентация. Перетащите элемент Область просмотра.
3. Перейдите на страницу Основные панели Свойства.
4. В поле Имя: введите statistika.
5. Перейдите на страницу Дополнительные панели Свойства. Введите в поля X: 0, Y: 1036, Ширина: 960, Высота: 630.
6. Задайте, как будет располагаться область просмотра относительно ее якоря, с помощью элемента управления Выравнивать по: Верхн. левому углу.
7. Выберите режим масштабирования из выпадающего списка Масштабирование: Подогнать под окно.
8. Перетащите элемент Прямоугольник.
9. Перейдите на страницу Дополнительные панели Свойства. Введите в поля X: 20, Y: 1076, Ширина: 920, Высота: 580.
10. Перетащите элемент text и в поле Текст: введите Результаты моделирования. На странице

Дополнительные панели Свойства введите в поля X: 360, Y: 1096.

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: 48, Y: 1126, при размещении надписи Коэффициенты использования типов средств связи укажите в полях X: 468, Y: 1418, а при размещении надписи Затраты на содержание резервов по типам средств связи в полях X: 485, Y: 1385.

### Использование элемента Текстовое поле

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

1. Перетащите элемент Текстовое поле из палитры Элементы управления и разместите согласно [рис. 5.12](#).
2. Выделяя последовательно каждый элемент Текстовое поле, переходите на страницу Основные панели Свойства и в поле Имя: давайте имя элементу согласно табл. 5.5.

Таблица 5.5. Имена элементов Текстовое поле

1	2	3	4	5	6
Максимальный доход по типам СС					
editbox1	editbox2	editbox3	editbox4	editbox5	editbox
Доход от дежурства по типам СС и всего					
editbox11	editbox12	editbox13	editbox14	editbox15	editbox16
Стоимость ремонта по типам СС и всего					
editbox21	editbox22	editbox23	editbox24	editbox25	editbox26
Суммарная прибыль от использования СС и всего					
editbox31	editbox32	editbox33	editbox34	editbox35	editbox36

Коэффициенты прибыли по типам СС и всего

editbox41 editbox42 editbox43 editbox44 editbox45 editbox46

Затраты на содержание резервов по типам СС и всего

editbox6 editbox7 editbox8 editbox9 editbox10 editbox17

## 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 элемент Диаграмма с накоплением из палитры Статистика и разместите согласно [рис. 5.12](#).
2. Выделяя последовательно каждый элемент Диаграмма с накоплением, переходите на страницу Внешний вид панели Свойства и установите:
  - Смещение по оси X: 40
  - Смещение по оси Y: 20
  - Ширина: 92
  - Высота: 138
  - Относительная ширина: 52
  - Направление: вертикальное

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

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

`degCC1.statsUtilization.mean()`

8. Установите Масштаб: Фиксированный и Обновлять автоматически. Цвет столбика, который будет отображать коэффициент использования группы СС одного типа, установите

- по своему усмотрению.
9. Проделайте пп. 3...7 для остальных 2...5 элементов Диаграмма с накоплением. При этом в поле Заголовок: вводите: СС2, СС3, СС4, СС5 соответственно.
10. В поле Значение: вводите Java коды также для объектов 2...5 соответственно:

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

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

#### Переключение между областями просмотра

- В Палитре выделите Презентация. Перетащите элемент `text`, разместите и введите в поле Текст: Постановка на дежурство, как на [рис. 5.12](#).
- На панели Свойства выделите Динамические и в поле Действие по щелчку: введите следующий Java код:

```
get_Main().Postanovka.navigateTo();
```

- Перетащите второй элемент `text`, разместите и введите в поле Текст: Имитация дежурства.
- На панели Свойства выделите Динамические и в поле Действие по щелчку: введите следующий Java код:

```
degyrstvo.degyr.navigateTo();
```

- Проделайте то же для Статистика. Введите Java код:

```
statistika.navigateTo();
```

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

## 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 произошествии.

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.

- Перетащите элемент Событие из палитры Модель на диаграмму класса активного объекта. Измените его имя на ResultsModeling. Нажмите Enter.
- Установите флажок Отображать имя.
- С помощью выпадающего списка Тип события: выберите Потаймауту.
- Установите Режим: Срабатывает один раз.
- Время срабатывания (абсолютное) 1000000.
- 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())*get_Main().ubitok);
UbitokCC1=UbitokCC1/100;
PribCC1=round((DoxDegCC1-(ZatrResCC1+ZatrRemCC1)+UbitokCC));
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())*get_Main().ubitokCC2*;
PribCC2=round((DoxDegCC2-(ZatrResCC2+ZatrRemCC2)+UbitokCC));
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())*
```

```
get_Main().ubitokCC3*ВремяРабСист*KCC3;
PribCC3=round((DoxDegCC3-(ZatrResCC3+ZatrRemCC3+UbitokCC));
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())*
get_Main().ubitokCC4*ВремяРабСист*KCC4;
PribCC4=round((DoxDegCC4-(ZatrResCC4+ZatrRemCC4+UbitokCC));
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())*
get_Main().ubitokCC5*ВремяРабСист*KCC5;
PribCC5=round((DoxDegCC5-(ZatrResCC5+ZatrRemCC5+UbitokCC));
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));
SumPribil=SumPribil/100;
koefPribil=round((SumPribil/SumDoxMax)*1000);
koefPribil=koefPribil/1000;
//вывод максимального дохода, дохода от дежурства по типам СС
```

```
editbox1.setText(DoxMaxCC1);
editbox11.setText(DoxDegCC1);
editbox2.setText(DoxMaxCC2);
editbox12.setText(DoxDegCC2);
editbox3.setText(DoxMaxCC3);
editbox13.setText(DoxDegCC3);
editbox4.setText(DoxMaxCC4);
editbox14.setText(DoxDegCC4);
editbox5.setText(DoxMaxCC5);
editbox15.setText(DoxDegCC5);
editbox.setText(SumDoxMax);
editbox16.setText(SumDoxDeg);
//вывод стоимости ремонта по типам СС и всего
editbox21.setText(ZatrRemCC1);
editbox22.setText(ZatrRemCC2);
editbox23.setText(ZatrRemCC3);
editbox24.setText(ZatrRemCC4);
editbox25.setText(ZatrRemCC5);
editbox26.setText(SumZatrRem);
//вывод чистой прибыли от использования по типам СС и всего
editbox31.setText(PribCC1);
editbox32.setText(PribCC2);
editbox33.setText(PribCC3);
editbox34.setText(PribCC4);
editbox35.setText(PribCC5);
editbox36.setText(SumPribil);
//вывод коэффициентов прибыли по типам СС и всего
editbox41.setText(koefPribCC1);
editbox42.setText(koefPribCC2);
editbox43.setText(koefPribCC3);
editbox44.setText(koefPribCC4);
editbox45.setText(koefPribCC5);
editbox46.setText(koefPribil);
//вывод затрат на содержание резервов по типам СС и всего
editbox6.setText(ZatrResCC1);
editbox7.setText(ZatrResCC2);
editbox8.setText(ZatrResCC3);
editbox9.setText(ZatrResCC4);
editbox10.setText(ZatrResCC5);
```

```
editbox17.setText(SumZatrRes);
```

Из кода следует, что по окончании моделирования, которое длится 1 000 000 единиц модельного времени, сработает метод события, 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. Простая переменная, например, `editbox1.setText(DoxMaxCC1);`

Запустите модель. На [рис. 5.13](#) и [рис. 5.14](#) показаны результаты моделирования.

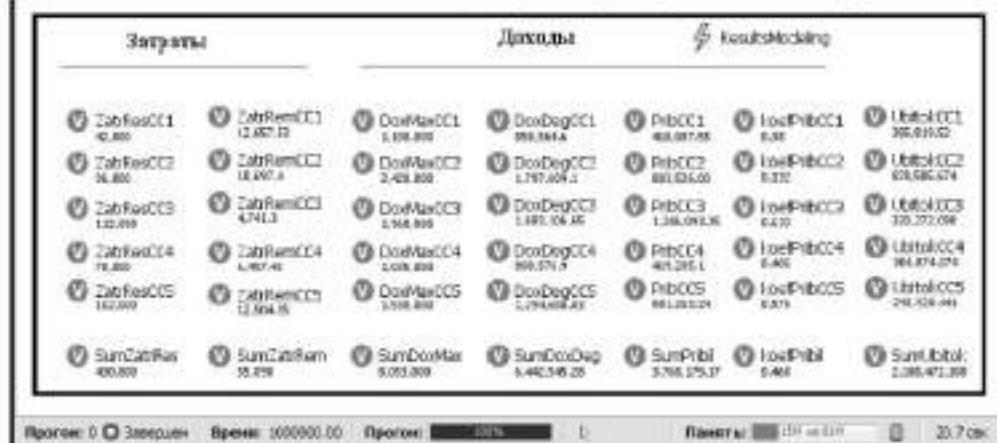


Рис. 5.13. Результаты моделирования

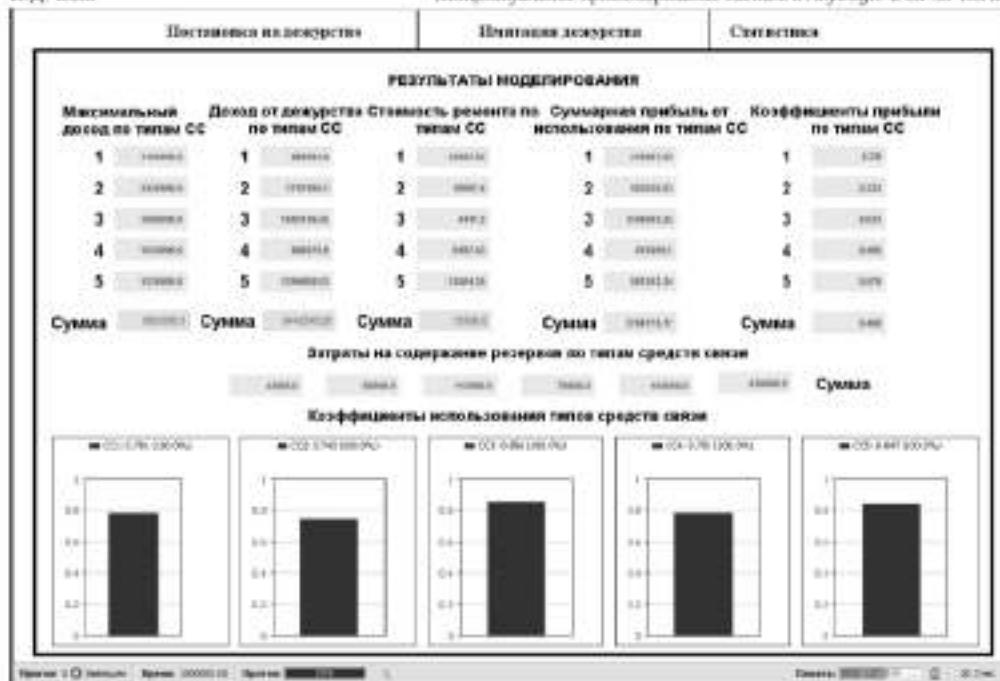


Рис. 5.14. Результаты сегмента Статистика

## 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.

## 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.

### Связывание параметров

Начиная создавать модель в AnyLogic, мы ничего не говорили об экспериментах и особенностях их проведения. Поэтому все исходные данные разместили на *Initial\_data\_PD* ([рис. 5.3](#)) и *Initial\_data\_D* ([рис. 5.9](#)) так, как нам представлялось удобным для построения модели и управления ею в ходе проведения простого эксперимента.

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

В результате связывания значение параметра любого уровня

вложенного объекта будет равно значению параметра объекта самого верхнего уровня.

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

Разместите элементы, как показано на [рис. 5.15](#). Внесите соответствующие изменения в модель. Обратите внимание на различие имён связываемых параметров, например, KCCP\_1.

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

1. Откройте диаграмму класса активного объекта Main.
2. Выберите на диаграмме вложенный объект degyrstvo.
3. Перейдите на страницу Параметры панели Свойства.
4. В таблице Параметры в поле Значение введите имя параметра класса объекта-владельца Main, значение которого нужно передавать этому параметру вложенного объекта. В результате у вас должно быть так, как на [рис. 5.16](#).

#### Initial\_data\_PD

<input checked="" type="radio"/> KolCC1	<input checked="" type="radio"/> zatrResCC1	<input checked="" type="radio"/> doxDegCC1	<input checked="" type="radio"/> ubitokCC1	<input checked="" type="radio"/> KOOP_1
<input checked="" type="radio"/> KolCC2	<input checked="" type="radio"/> zatrResCC2	<input checked="" type="radio"/> doxDegCC2	<input checked="" type="radio"/> ubitokCC2	<input checked="" type="radio"/> KOOP_2
<input checked="" type="radio"/> KolCC3	<input checked="" type="radio"/> zatrResCC3	<input checked="" type="radio"/> doxDegCC3	<input checked="" type="radio"/> ubitokCC3	<input checked="" type="radio"/> KOOP_3
<input checked="" type="radio"/> KolCC4	<input checked="" type="radio"/> zatrResCC4	<input checked="" type="radio"/> doxDegCC4	<input checked="" type="radio"/> ubitokCC4	<input checked="" type="radio"/> KOOP_4
<input checked="" type="radio"/> KolCC5	<input checked="" type="radio"/> zatrResCC5	<input checked="" type="radio"/> doxDegCC5	<input checked="" type="radio"/> ubitokCC5	<input checked="" type="radio"/> KOOP_5
<input checked="" type="radio"/> KolCC	<input checked="" type="radio"/> NumOC			<input checked="" type="radio"/> Kol_master

Рис. 5.15. Размещение элементов на Initial\_data\_PD

degurstvo - Degurstvo		
Основные	KCCS	60
Параметры	KCCP1	KCCP_1
Статистика	KCCP2	KCCP_2
Описание	KCCP3	KCCP_3
	KCCP4	KCCP_4
	KCOPS	KCCP_5
	stoinRemCC1	17
	stoinRemCC3	16
	stoinRemCC2	18
	stoinRemCC4	20
	stoinRemCC5	21
	timeRem1	6.5
	timeRem2	4.2
	timeRem3	2.8
	timeRem4	3
	timeRem5	5.5
	timeOtkaz1	373
	timeOtkaz2	301
	timeOtkaz3	482
	timeOtkaz4	325
	timeOtkaz5	470
	KoProlon	1000
	ВремяРебСист	1000
	kol_master	Kol_master

Рис. 5.16. Фрагмент страницы Параметры после связывания параметров

## Эксперимент Оптимизация стохастических моделей

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

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

1. В панели Проект щелкните правой кнопкой мыши элемент модели ComSystem и из контекстного меню выберите Создать / Эксперимент.
2. В появившемся диалоговом окне из списка Тип эксперимента: выберите Оптимизация (рис. 5.17).

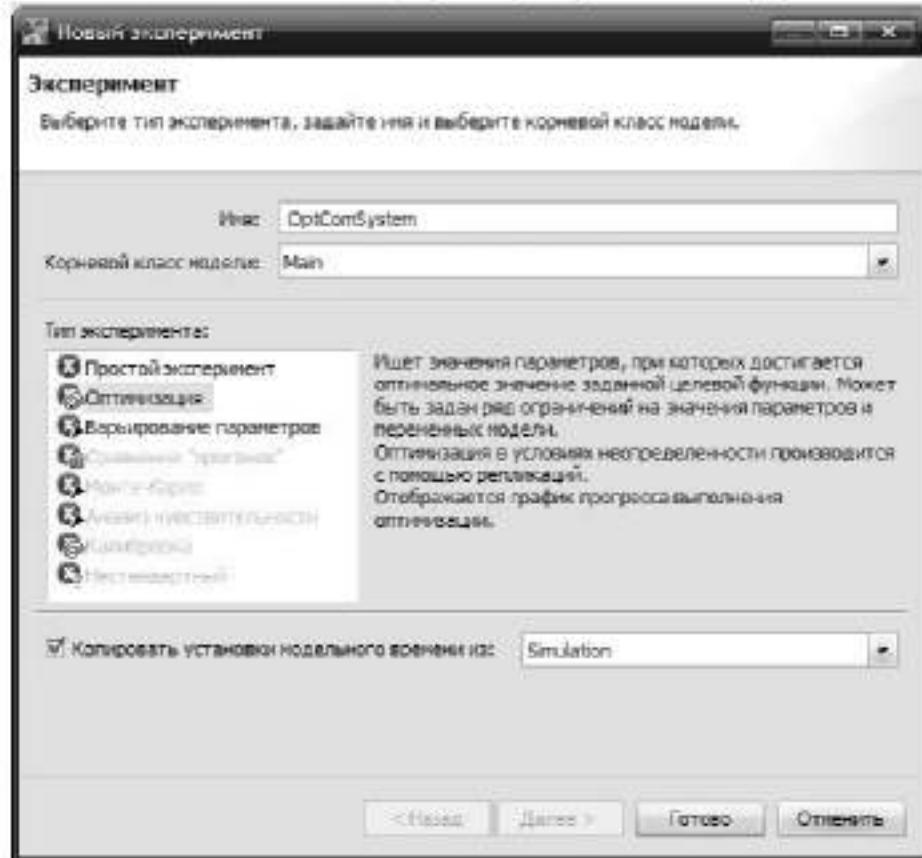


Рис. 5.17. Диалоговое окно Новый эксперимент

3. В поле Имя введите имя эксперимента, например, OptComSystem. Имя эксперимента должно начинаться с заглавной буквы - таково правило названия классов в Java.
4. В поле Корневой класс модели: выберите Main. Этим действием вы задали корневой (главный) класс эксперимента. Объект этого класса будет играть роль корня иерархического дерева объектов модели, запускаемой оптимизационным экспериментом.
5. Если вы хотите применить к создаваемому эксперименту временные установки другого эксперимента, оставьте установленным флажок Копировать установки модельного времени из: и выберите эксперимент из расположенного справа выпадающего списка. В данном случае

оставьте, так как есть: **Simulation**.

6. Щелкните кнопку **Готово**. Появится страница **Основные панели Свойства** ([рис. 5.18](#)).
7. Установите опцию **максимизировать**.
8. Установите **Фиксированное начальное число** (воспроизводимые прогоны).
9. В поле **Начальное число** введите 5672.
10. Целевая функция доступна как **root**. Поэтому в поле **Целевая функция** введите **root.degystvo.koefPribil**.
11. 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.
12. Число итераций модели при полном факторном эксперименте, то есть число всех возможных сочетаний факторов, определяется по формуле:

$$I = k_1 \cdot k_2 \cdot \dots \cdot k_i \cdot \dots \cdot k_m,$$

где  $k_i$  - число уровней  $i$ -го фактора,  $i = \overline{1, m}$ .

13. В нашей модели нужно менять количество резервных средств связи **KCCP\_1..KCCP\_5** и количество мастеров-ремонтников **Kol\_masterov**, 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

$$I = k_1 \cdot k_2 \cdot k_3 \cdot k_4 \cdot k_5 \cdot k_6 = 3 \cdot 6 \cdot 6 \cdot 6 \cdot 5 \cdot 5 = 16200$$

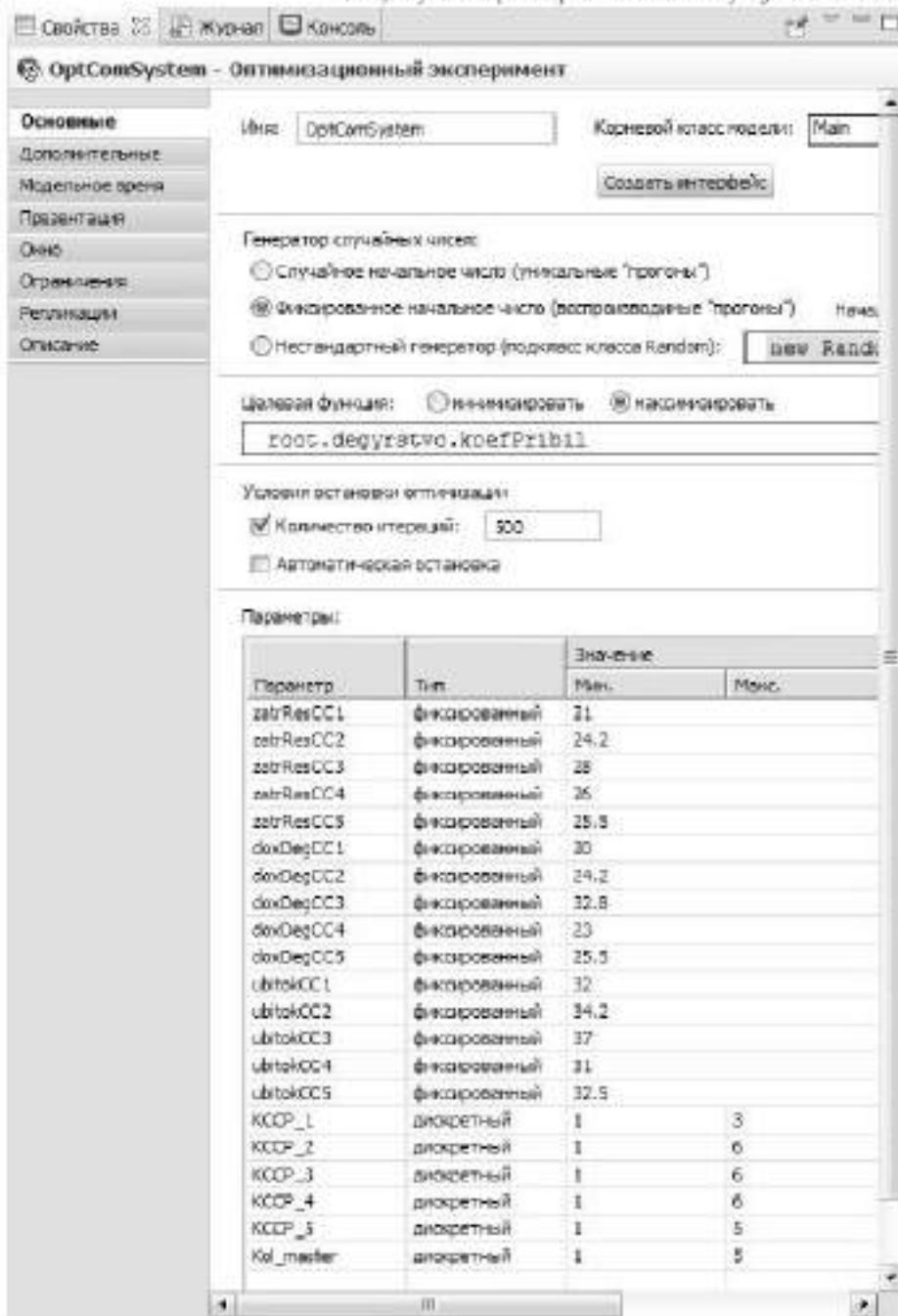


Рис. 5.18. Вкладка Основные оптимизационного эксперимента

14. Оставьте 500 в поле Количество итераций:, так как данная

версия AnyLogic ограничена этим количеством итераций.

15. Задайте параметры, значения которых будут меняться. В таблице на [рис. 5.18](#) перечислены все параметры корневого объекта Main. Чтобы разрешить варьирование параметров оптимизатором, перейдите на строку с параметром КССР\_1. Щелкните 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. 5.18](#).
18. Перейдите на страницу Репликации панели Свойства (рис. 5.19).
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 [1]:

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

где  $t_{\alpha}$  - табулированный аргумент функции Лапласа.

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

22. Выберите опцию Фиксированное количество

репликаций и в соответствующем поле установите 4.

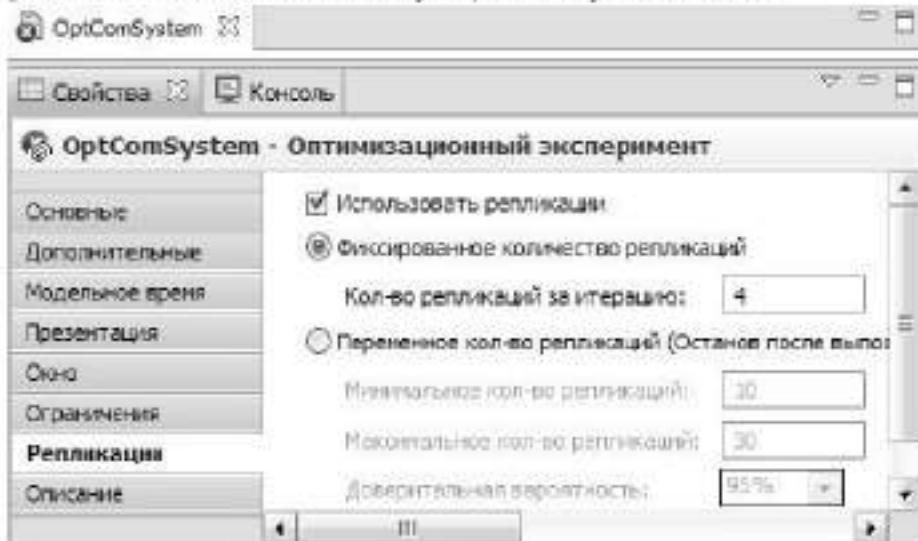


Рис. 5.19. Страница Репликации оптимизационного эксперимента

23. Вернитесь на страницу Основные и щелкните кнопку Создать интерфейс. Кнопка находится в правом верхнем углу страницы Основные. После щелчка удаляется содержимое презентации эксперимента и создается интерфейс эксперимента заново (рис. 5.20) 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.
24. В меню запуск выполните ComSystem/OptComSystem.
25. Щелкните Запустить оптимизацию. Начнет выполняться эксперимент. Во время эксперимента можно видеть на графике изменение значения целевой функции. После  $500*4=2\ 000$  прогонов (см. рис. 5.21) эксперимент остановится.
26. Результаты оптимизационного эксперимента приведены на рис. 5.21. Наилучшее значение целевой функции - коэффициент прибыли - равно 0,959. Получено оно на 154 итерации при следующих оптимальных значениях параметров: KCCP\_1=2, KCCP\_2= KCCP\_3= KCCP\_4=KCCP\_5=1, Kol\_master=5.
27. Вернитесь к простому эксперименту. Измените значения

**KCCP\_1... KCCP\_5 и Kol\_master на Initial\_data\_PD на значения, полученные в оптимизационном эксперименте.**

© OptComSystem

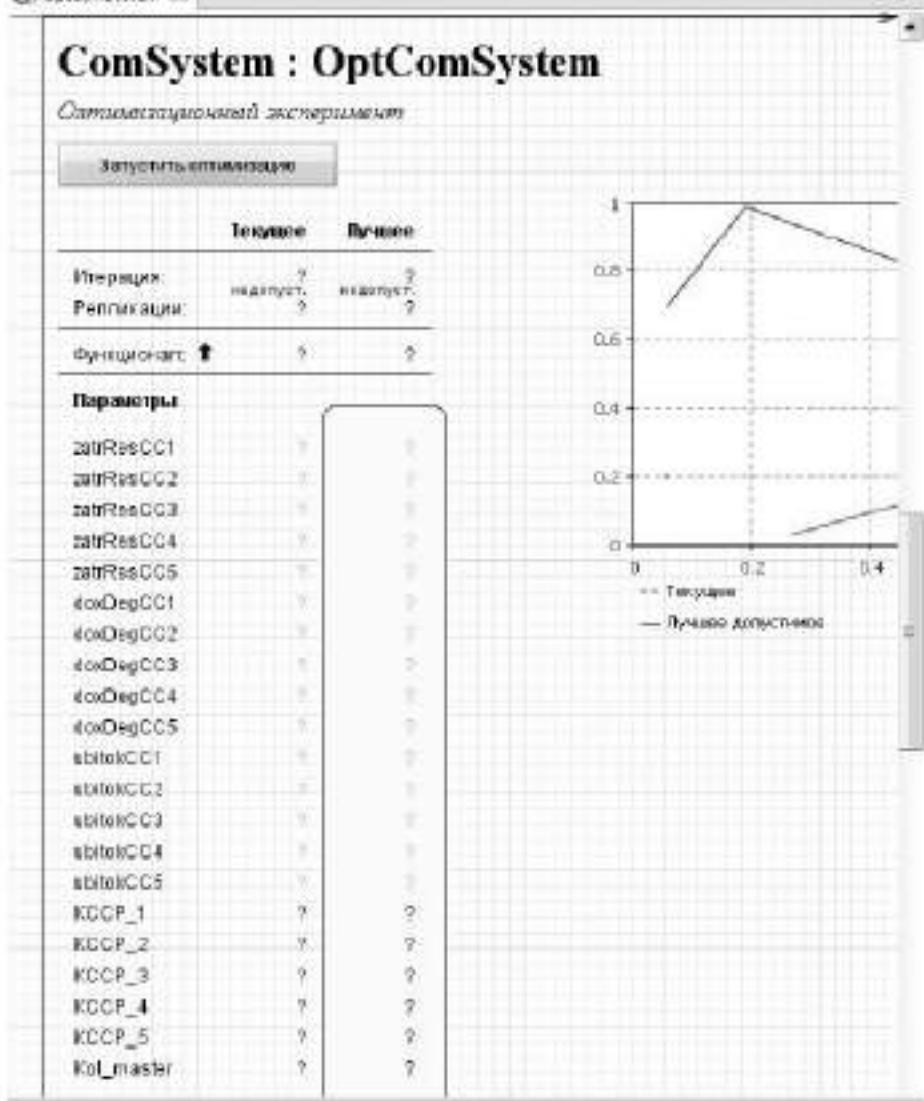


Рис. 5.20. Интерфейс оптимизационного эксперимента

28. Запустите простой эксперимент. Вы получите коэффициент прибыли 0,949, то есть такой же, как и в оптимизационном эксперименте. При этом уменьшатся затраты на содержание резервных СС и возрастут коэффициенты использования СС.

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

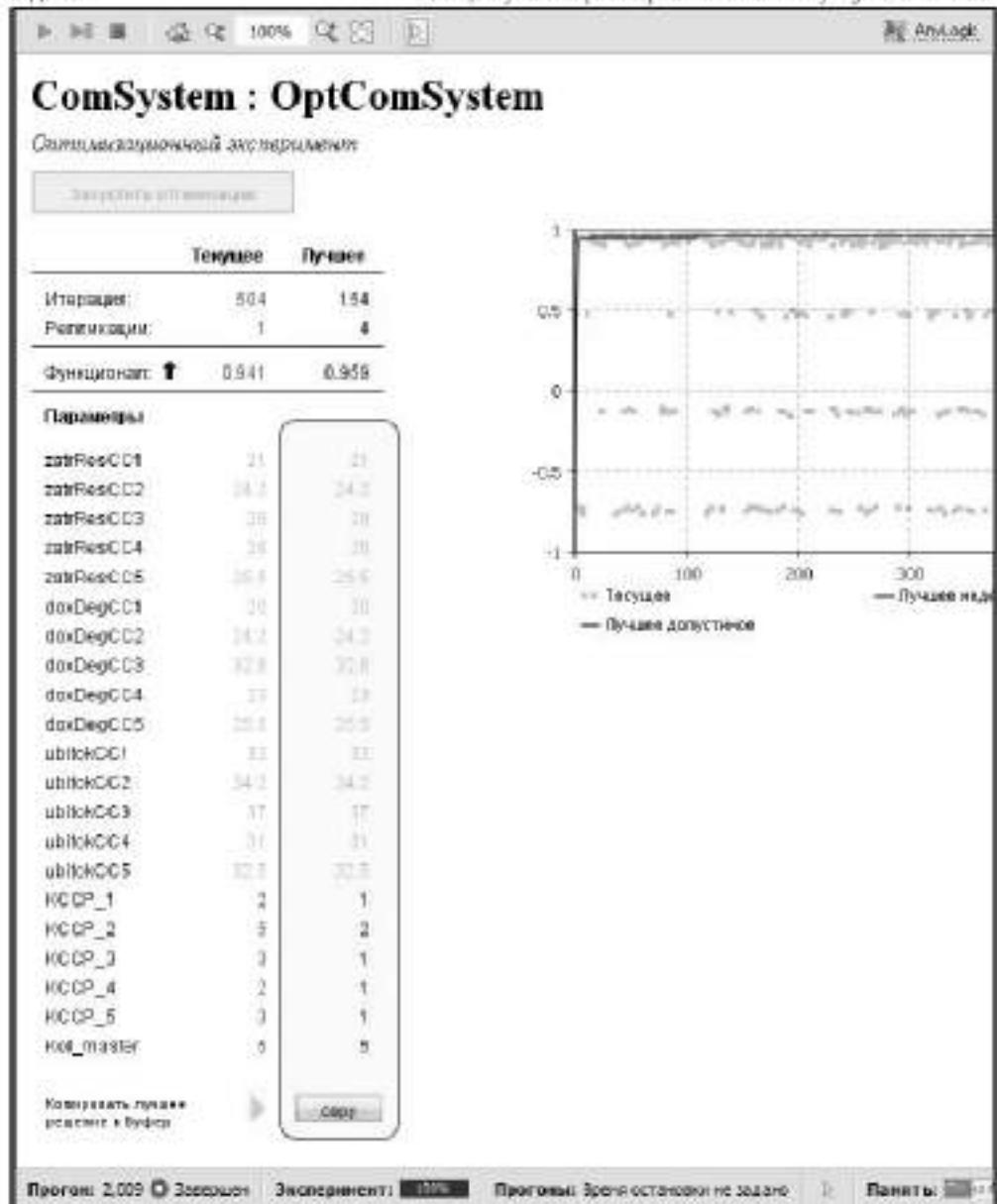


Рис. 5.21. Результаты оптимизационного эксперимента

Обратите внимание на интерфейс оптимизационного эксперимента (рис. 5.20): заготовка для графика относительного коэффициента прибыли расположена в первом квадранте, то есть коэффициент предполагается положительным. Однако при некоторых параметрах

система предоставления услуг связи оказывается убыточной. Поэтому появляется четвёртый квадрант ([рис. 5.21](#)), в котором искомый коэффициент отрицательный.

### Эксперимент Варьирование параметров

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

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

1. В панели Проект щёлкните правой кнопкой мыши элемент модели ComSystem и из контекстного меню выберите Создать эксперимент.
2. В появившемся диалоговом окне из списка Тип эксперимента: выберите Варьирование параметров ([рис. 5.22](#)).
3. В поле Имя введите имя эксперимента, например, ParVarComSystem.
4. Остальные установки оставьте такими, как на [рис. 5.18](#). Назначение их тоже, что и в оптимизационном эксперименте.

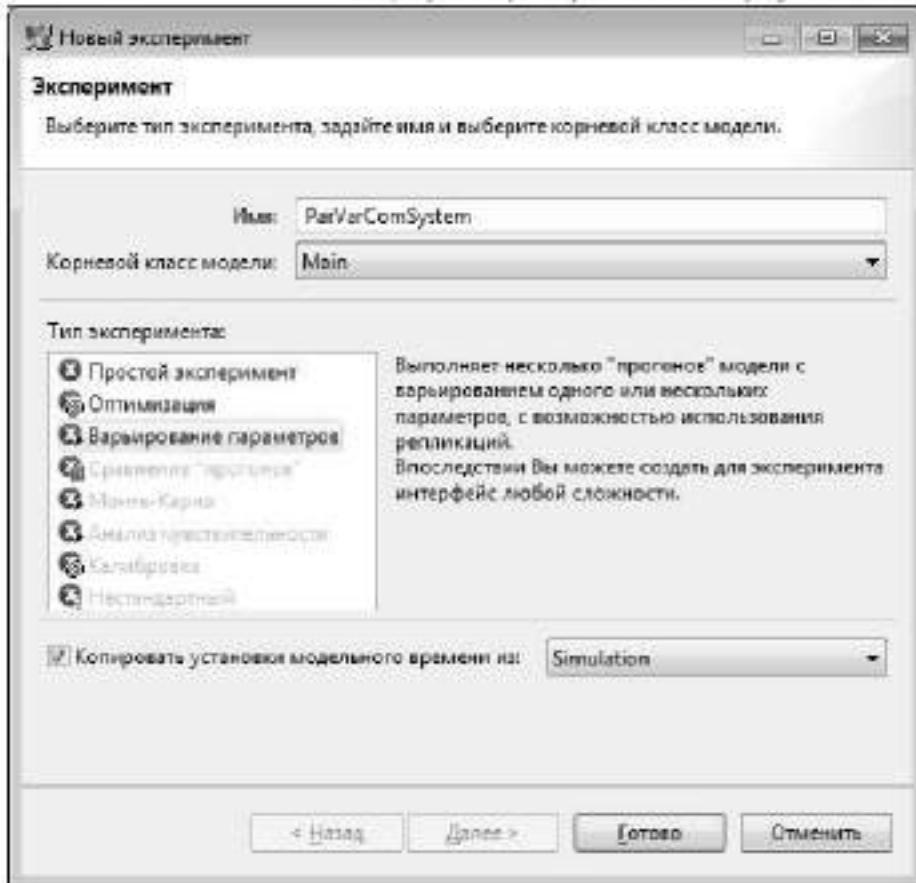


Рис. 5.22. Окно Новый эксперимент с выбранным экспериментом

**ParVarComSystem - Эксперимент варьирования параметров**

<b>Основные</b> <b>Дополнительные</b> <b>Модельное время</b> <b>Примечания</b> <b>Окно</b> <b>Репозиторий</b> <b>Справочник</b>	<input type="text" value="ParVarComSystem"/>	Корневой класс модели: <input type="text" value="Java"/>	<input type="button" value="Создать интерфейс"/>																																																																																																																																				
<small>Генератор случайных чисел:</small> <input type="radio"/> Случайное начальное число (инициализация "программ") <input checked="" type="radio"/> Фиксированное начальное число (встроенные программы) <input type="radio"/> Нестандартный генератор (подкласс класса Random) <span style="float: right;">Начальное число: 5672 Имя: Random()</span>																																																																																																																																							
Параметры: <input checked="" type="radio"/> Варьировать в диапазоне <input type="radio"/> Привязать к значению "программ"																																																																																																																																							
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;">Параметр</th> <th style="text-align: left; padding: 2px;">Тип</th> <th style="text-align: left; padding: 2px;">Значение</th> <th style="text-align: left; padding: 2px;">Нач.</th> <th style="text-align: left; padding: 2px;">Макс.</th> <th style="text-align: left; padding: 2px;">Шаг</th> </tr> </thead> <tbody> <tr> <td style="padding: 2px;">zdrResCC1</td> <td style="padding: 2px;">Фиксированный</td> <td style="padding: 2px;">21</td> <td style="padding: 2px;"></td> <td style="padding: 2px;"></td> <td style="padding: 2px;"></td> </tr> <tr> <td style="padding: 2px;">zdrResCC2</td> <td style="padding: 2px;">Фиксированный</td> <td style="padding: 2px;">24.2</td> <td style="padding: 2px;"></td> <td style="padding: 2px;"></td> <td style="padding: 2px;"></td> </tr> <tr> <td style="padding: 2px;">zdrResCC3</td> <td style="padding: 2px;">Фиксированный</td> <td style="padding: 2px;">28</td> <td style="padding: 2px;"></td> <td style="padding: 2px;"></td> <td style="padding: 2px;"></td> </tr> <tr> <td style="padding: 2px;">zdrResCC4</td> <td style="padding: 2px;">Фиксированный</td> <td style="padding: 2px;">36</td> <td style="padding: 2px;"></td> <td style="padding: 2px;"></td> <td style="padding: 2px;"></td> </tr> <tr> <td style="padding: 2px;">zdrResCC5</td> <td style="padding: 2px;">Фиксированный</td> <td style="padding: 2px;">25.5</td> <td style="padding: 2px;"></td> <td style="padding: 2px;"></td> <td style="padding: 2px;"></td> </tr> <tr> <td style="padding: 2px;">doxDegCC1</td> <td style="padding: 2px;">Диапазон</td> <td style="padding: 2px;">18</td> <td style="padding: 2px;">24</td> <td style="padding: 2px;">0.5</td> <td style="padding: 2px;"></td> </tr> <tr> <td style="padding: 2px;">doxDegCC2</td> <td style="padding: 2px;">Диапазон</td> <td style="padding: 2px;">20</td> <td style="padding: 2px;">28</td> <td style="padding: 2px;">0.5</td> <td style="padding: 2px;"></td> </tr> <tr> <td style="padding: 2px;">doxDegCC3</td> <td style="padding: 2px;">Диапазон</td> <td style="padding: 2px;">18</td> <td style="padding: 2px;">34</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;"></td> </tr> <tr> <td style="padding: 2px;">doxDegCC4</td> <td style="padding: 2px;">Диапазон</td> <td style="padding: 2px;">19</td> <td style="padding: 2px;">27</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;"></td> </tr> <tr> <td style="padding: 2px;"><b>doxDegCC5</b></td> <td style="padding: 2px;">Диапазон</td> <td style="padding: 2px;"><b>22</b></td> <td style="padding: 2px;"><b>28</b></td> <td style="padding: 2px;"><b>1</b></td> <td style="padding: 2px;"></td> </tr> <tr> <td style="padding: 2px;">ubtolCC1</td> <td style="padding: 2px;">Фиксированный</td> <td style="padding: 2px;">22</td> <td style="padding: 2px;"></td> <td style="padding: 2px;"></td> <td style="padding: 2px;"></td> </tr> <tr> <td style="padding: 2px;">ubtolCC2</td> <td style="padding: 2px;">Фиксированный</td> <td style="padding: 2px;">34.2</td> <td style="padding: 2px;"></td> <td style="padding: 2px;"></td> <td style="padding: 2px;"></td> </tr> <tr> <td style="padding: 2px;">ubtolCC3</td> <td style="padding: 2px;">Фиксированный</td> <td style="padding: 2px;">27</td> <td style="padding: 2px;"></td> <td style="padding: 2px;"></td> <td style="padding: 2px;"></td> </tr> <tr> <td style="padding: 2px;">ubtolCC4</td> <td style="padding: 2px;">Фиксированный</td> <td style="padding: 2px;">31</td> <td style="padding: 2px;"></td> <td style="padding: 2px;"></td> <td style="padding: 2px;"></td> </tr> <tr> <td style="padding: 2px;">ubtolCC5</td> <td style="padding: 2px;">Фиксированный</td> <td style="padding: 2px;">32.5</td> <td style="padding: 2px;"></td> <td style="padding: 2px;"></td> <td style="padding: 2px;"></td> </tr> <tr> <td style="padding: 2px;">KCCP_1</td> <td style="padding: 2px;">Фиксированный</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;"></td> <td style="padding: 2px;"></td> <td style="padding: 2px;"></td> </tr> <tr> <td style="padding: 2px;">KCCP_2</td> <td style="padding: 2px;">Фиксированный</td> <td style="padding: 2px;">2</td> <td style="padding: 2px;"></td> <td style="padding: 2px;"></td> <td style="padding: 2px;"></td> </tr> <tr> <td style="padding: 2px;">KCCP_3</td> <td style="padding: 2px;">Фиксированный</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;"></td> <td style="padding: 2px;"></td> <td style="padding: 2px;"></td> </tr> <tr> <td style="padding: 2px;">KCCP_4</td> <td style="padding: 2px;">Фиксированный</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;"></td> <td style="padding: 2px;"></td> <td style="padding: 2px;"></td> </tr> <tr> <td style="padding: 2px;">KCCP_5</td> <td style="padding: 2px;">Фиксированный</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;"></td> <td style="padding: 2px;"></td> <td style="padding: 2px;"></td> </tr> <tr> <td style="padding: 2px;">Kd_nester</td> <td style="padding: 2px;">Фиксированный</td> <td style="padding: 2px;">5</td> <td style="padding: 2px;"></td> <td style="padding: 2px;"></td> <td style="padding: 2px;"></td> </tr> </tbody> </table>				Параметр	Тип	Значение	Нач.	Макс.	Шаг	zdrResCC1	Фиксированный	21				zdrResCC2	Фиксированный	24.2				zdrResCC3	Фиксированный	28				zdrResCC4	Фиксированный	36				zdrResCC5	Фиксированный	25.5				doxDegCC1	Диапазон	18	24	0.5		doxDegCC2	Диапазон	20	28	0.5		doxDegCC3	Диапазон	18	34	1		doxDegCC4	Диапазон	19	27	1		<b>doxDegCC5</b>	Диапазон	<b>22</b>	<b>28</b>	<b>1</b>		ubtolCC1	Фиксированный	22				ubtolCC2	Фиксированный	34.2				ubtolCC3	Фиксированный	27				ubtolCC4	Фиксированный	31				ubtolCC5	Фиксированный	32.5				KCCP_1	Фиксированный	1				KCCP_2	Фиксированный	2				KCCP_3	Фиксированный	1				KCCP_4	Фиксированный	1				KCCP_5	Фиксированный	1				Kd_nester	Фиксированный	5			
Параметр	Тип	Значение	Нач.	Макс.	Шаг																																																																																																																																		
zdrResCC1	Фиксированный	21																																																																																																																																					
zdrResCC2	Фиксированный	24.2																																																																																																																																					
zdrResCC3	Фиксированный	28																																																																																																																																					
zdrResCC4	Фиксированный	36																																																																																																																																					
zdrResCC5	Фиксированный	25.5																																																																																																																																					
doxDegCC1	Диапазон	18	24	0.5																																																																																																																																			
doxDegCC2	Диапазон	20	28	0.5																																																																																																																																			
doxDegCC3	Диапазон	18	34	1																																																																																																																																			
doxDegCC4	Диапазон	19	27	1																																																																																																																																			
<b>doxDegCC5</b>	Диапазон	<b>22</b>	<b>28</b>	<b>1</b>																																																																																																																																			
ubtolCC1	Фиксированный	22																																																																																																																																					
ubtolCC2	Фиксированный	34.2																																																																																																																																					
ubtolCC3	Фиксированный	27																																																																																																																																					
ubtolCC4	Фиксированный	31																																																																																																																																					
ubtolCC5	Фиксированный	32.5																																																																																																																																					
KCCP_1	Фиксированный	1																																																																																																																																					
KCCP_2	Фиксированный	2																																																																																																																																					
KCCP_3	Фиксированный	1																																																																																																																																					
KCCP_4	Фиксированный	1																																																																																																																																					
KCCP_5	Фиксированный	1																																																																																																																																					
Kd_nester	Фиксированный	5																																																																																																																																					

Рис. 5.23. Страница Основные эксперимента варьирования параметров

- Щелкните кнопку Готово. Появится страница Основные панели Свойства (рис. 5.23).
- Обратите внимание, что на странице Основные по сравнению с оптимизационным экспериментом отсутствуют опции минимизировать, максимизировать, Количество итераций. Последнее определяется AnyLogic в зависимости от диапазонов и шагов изменения параметров.
- Задайте диапазон допустимых значений параметра doxDegCC1. Перейдите в таблице на рис. 5.23 на строку с этим параметром. Щелкните мышью в ячейке Тип. Выберите тип параметра, отличный от значения фиксированный. Параметр

- doxDegCC1 типа double, поэтому он может изменяться в диапазоне. Выберите Диапазон. В ячейку Мин введите минимальное значение 18, в ячейку Макс - максимальное значение 24, в ячейке Шаг укажите величину шага 0.5.
8. Задайте также остальные параметры, как на [рис. 5.23](#).
9. Перейдите на страницу Репликации панели Свойства и установите флажок Использовать репликации. В поле Кол-во репликций за итерацию, установите, как и в предыдущем эксперименте, 4.
10. Вернитесь на страницу Основные и щелкните кнопку Создать интерфейс.
11. В эксперименте Варьирование параметров в отличие от эксперимента Оптимизация интерфейс создает пользователь. Связано это с тем, что выходными результатами данного эксперимента могут быть любые показатели моделируемой системы.
12. Создайте интерфейс, показанный на [рис. 5.24](#). Здесь вы видите график, который будет отображать значение коэффициента прибыли для каждой итерации.
13. Перетащите элемент График из палитры Статистика на диаграмму активного класса.
14. Щелкните Добавить набор данных.
15. Установите опцию Набор данных. Заголовок: КоefPribil. Набор данных: dataset. Установите Не обновлять автоматически.
16. Перейдите на страницу Дополнительные и установите: X: 260, Y: 100, Ширина: 510, Высота: 400, Цвет фона: Нет заливки, Цвет границы: Нет линии.
17. Перейдите на страницу Внешний вид. Установите: Смещение по X: 40, Смещение по Y: 20, Ширина: 450, Высота: 330.
18. Также из палитры Статистика перетащите элемент Набор данных. Установите опцию Не обновлять автоматически.
19. Из палитры Основная перетащите элемент Простая переменная. На странице Основные панели Свойства в поле Имя: введите valueOfKoefPribil. Установите Уровень доступа: public. Тип: double.

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

- в поле Действие после прогона модели:

```
valueOfKoefPribil = root.degyrstvo.koefPribil;
```

- в поле Действие после итерации

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

21. Выполните ComSystem2/VarParComSystem.

# ComSystem : ParVarComSystem

Эксперимент варьирования параметров

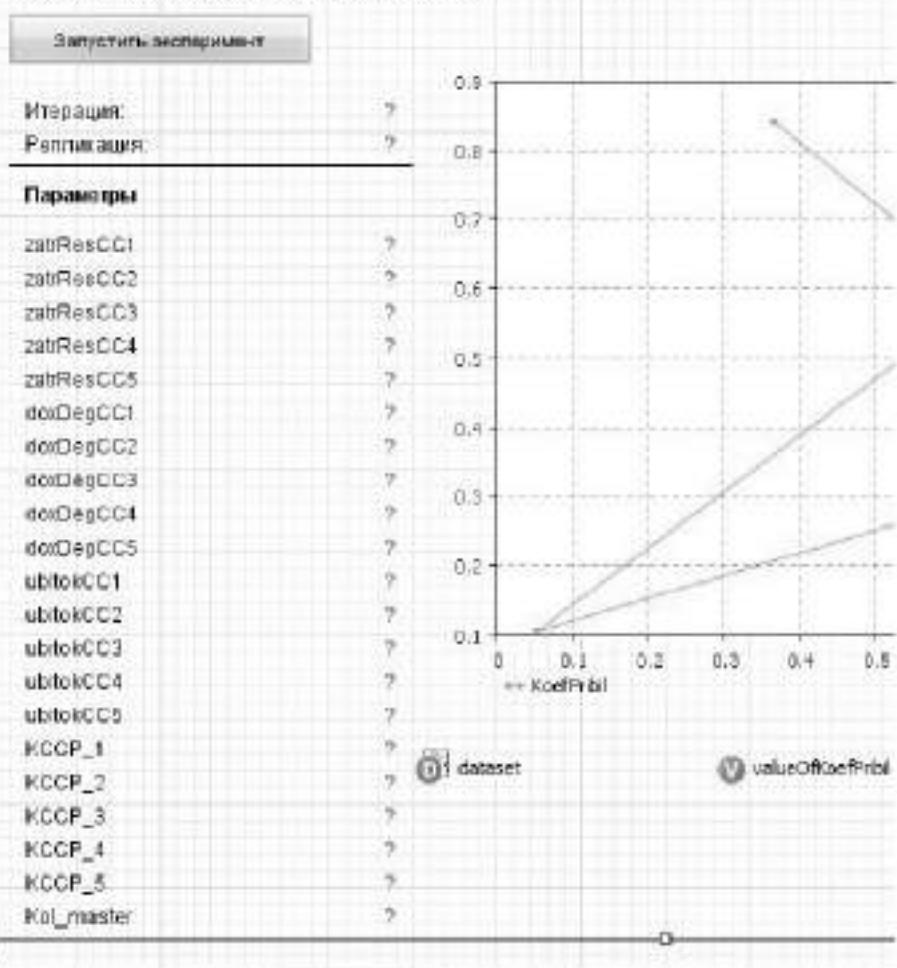


Рис. 5.24. Интерфейс эксперимента варьирования параметров

- Щелкните Запустить эксперимент. Начнет выполняться эксперимент. Во время эксперимента можно видеть на графике изменение значения коэффициента прибыли. Фрагмент результатов выполнения эксперимента Варьирование параметров приведен на рис. 5.25.
- Эксперимент был приостановлен после 74 прогона, то есть на  $74/4 \rightarrow 19$ -й итерации.

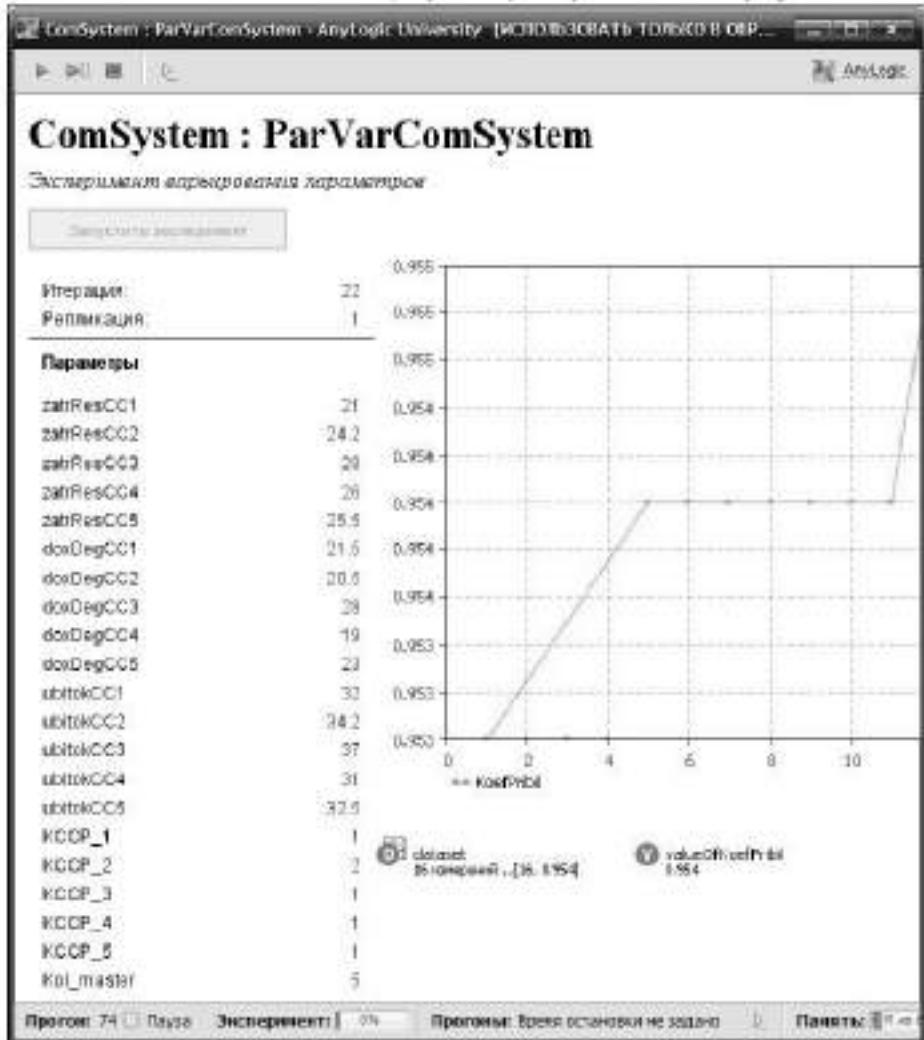


Рис. 5.25. Фрагмент результатов выполнения эксперимента Варьирование параметров

24. Коэффициент прибыли составляет 0,954 при  $\text{dxdDegCC1} = 21,5, \text{dxdDegCC2} = 20,5, \text{dxdDegCC3} = 28, \text{dxdDegCC4} = 19, \text{dxdDegCC5} = 23$ .
25. Поскольку вы оставили значения KCCP\_1... KCCP\_5 и Kol\_master на Initial\_data\_PD, полученные в оптимизационном эксперименте (см. рис. 5.25), то значения коэффициентов прибыли в эксперименте варьирования параметрами близки к 0,949.

В обоих экспериментах мы использовали опцию Фиксированное начальное число (воспроизводимые прогоны), поэтому генератор случайных чисел модели всегда инициализировался одним и тем же начальным числом 5672, заданным нами в поле Начальное число. Все запуски модели были идентичными и воспроизводимыми, что полезно при отладке модели.

Если выбрана опция Случайное начальное число (уникальные прогоны), то при каждом новом запуске модели генератор случайных чисел инициализируется другим числом и результаты оптимизации могут отличаться.

Создайте оптимизационный эксперимент OptComSystem1, который будет отличаться от OptComSystem лишь выбранной опцией Случайное начальное число (уникальные прогоны).

Запустите эксперимент. Вы получите следующие результаты. Наилучшее значение целевой функции - коэффициент прибыли - равно 0,957. Получено оно на 172 итерации при следующих оптимальных значениях параметров: KCCP\_1=1, KCCP\_2=2, KCCP\_3=1, KCCP\_4=1, KCCP\_5=2, Kol\_master=5. Таким образом, получен другой коэффициент прибыли, при этом другое значение оптимального параметра KCCP\_5=2.

Оптимизационный эксперимент OptComSystem1 выполнялся 4839,2 сек (80,66 мин). Это при том, что эксперимент проводился на компьютере с четырёхядерным процессором и одновременно выполнялись четыре прогона модели.

## Экспорт модели как Java апплета

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

Наличие такой возможности позволяет удалённым пользователям запускать интерактивные модели в веб-браузере при отсутствии AnyLogic или какого-либо другого программного обеспечения. В этом

случае на клиентской машине будут запускаться скопированные из сети файлы модели с такой же поддержкой интерактивной работы, что и при запуске из среды AnyLogic.

Экспортируйте модель ComSystem в виде Java апплета.

1. Щелкните в панели Проекты модель ComSystem и выберите



Экспорт/В Java апплет (запускается в веб браузере) из контекстного меню.

2. Откроется диалоговое окно Экспорт модели (рис. 5.26). Щелчком мыши раскройте список Экспортировать эксперимент и выберите в нем Simulation. Настройки этого эксперимента будут применены к экспортируемой модели.
3. В поле Каталог для создаваемых файлов укажите каталог, в который вы хотите поместить файлы экспортируемой модели. Можно также выбрать каталог с помощью диалогового окна навигации, которое становится доступным при нажатии кнопки Выбрать.
4. По умолчанию кнопки панели инструментов и другие элементы пользовательского интерфейса апплета будут на том языке, который выбран в настройках вашего компьютера (Язык: <системный по-умолчанию>). При необходимости можно выбрать для интерфейса апплета другой язык из выпадающего списка Язык: (на данный момент AnyLogic поддерживает русский, английский, китайский, немецкий и итальянский языки).

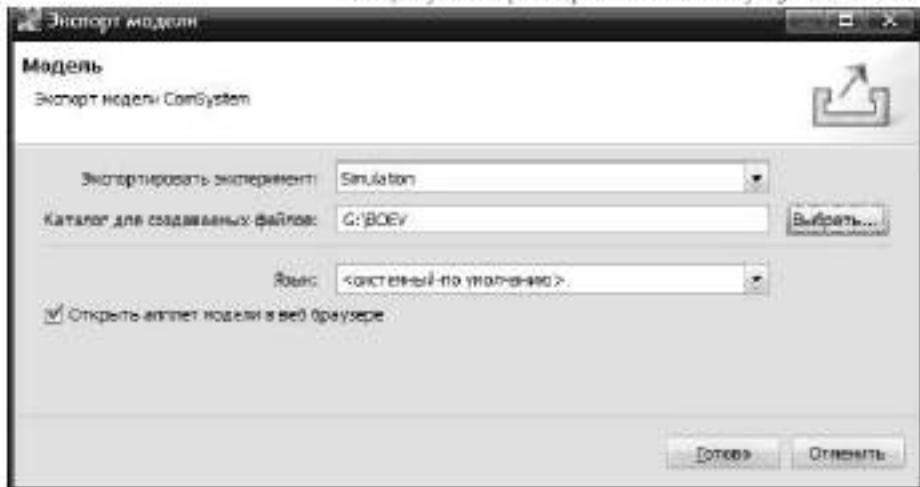


Рис. 5.26. Диалоговое окно Экспорт модели



Рис. 5.27. Набор файлов Java аплет

5. Оставьте установленным флажок Открыть апплет модели

в веб браузере.

6. Щелкните кнопку Готово. Откроется диалоговое окно, в котором будет сообщение об успешном завершении экспортации модели ComSystem: Модель ComSystem была экспортирована в G:\BOEV.
7. Щелкните OK.

Модель, экспортованная как Java апплет, представляет собой набор следующих файлов ([рис. 5.28](#)):

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

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

Запустите апплет модели, дважды щелкнув ComSystem.html. Результаты работы апплета модели показаны на [рис. 5.28](#). Коэффициент прибыли равен 0,949 (см. п. 5.1.8.3).

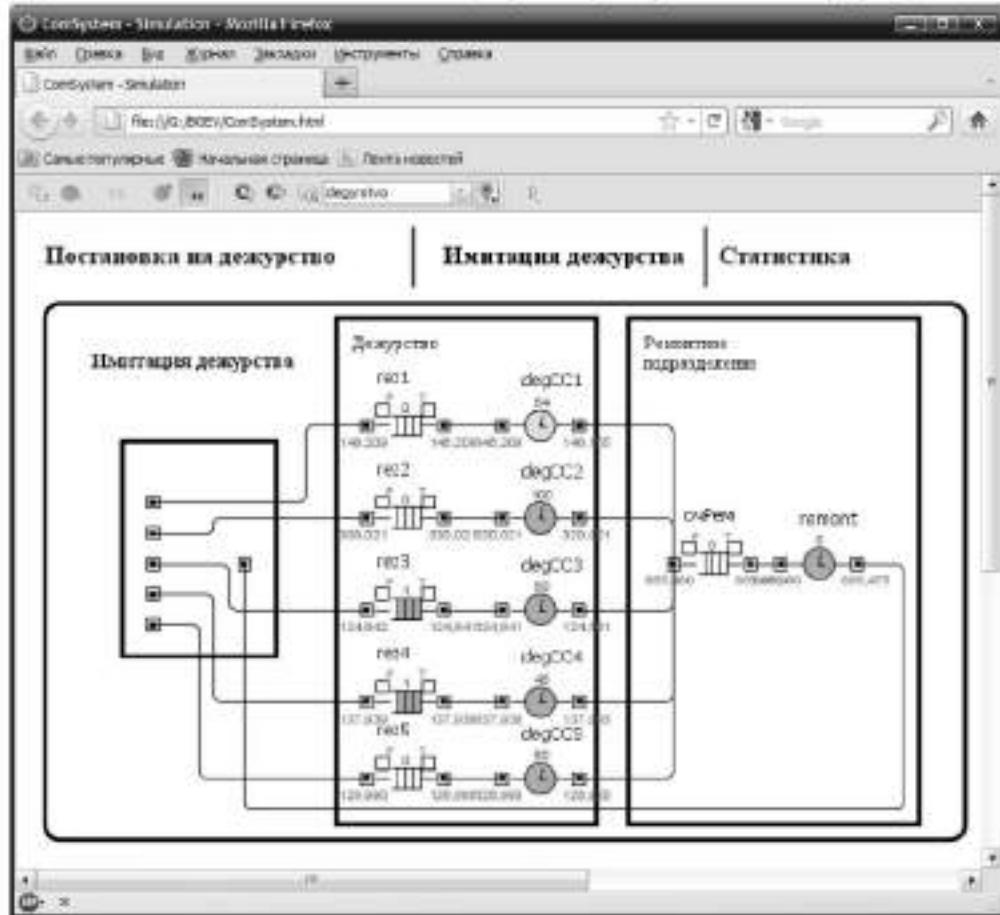


Рис. 5.28. Результаты работы апплета модели ComSystem

## Модель в GPSS World

### Состав модели в GPSS World

Как уже отмечалось, система связи представляет собой многофазную многоканальную систему массового обслуживания замкнутого типа с отказами и ожиданием.

Какие ограничения в системе?

1. Число мастеров-ремонтников в ремонтном подразделении.
2. Максимальное число одновременно находящихся СС на дежурстве.
3. Максимальное число резервных СС.
4. Общее число СС в системе.

Для моделирования двух первых ограничений целесообразно использовать МКУ, а для третьего и четвертого ограничений - транзакты.

Модель системы связи должна состоять из следующих сегментов:

- описание арифметических выражений;
- сегмент имитации постановки на дежурство СС;
- сегмент имитации функционирования системы дежурства СС;
- сегмент имитации функционирования ремонтного подразделения;
- сегмент задания времени моделирования и вычисления результатов моделирования;
- расчет ожидаемой прибыли;
- сегмент переопределения блоков программы - изменения версий модели.

Сегмент переопределения блоков модели необходим для изменения версий модели, т. е. изменения количества резервных СС и мастеров-ремонтников в ремонтном подразделении, а также номеров строк и столбцов матриц, в которые записываются результаты моделирования. Количество переопределений в сегменте соответствует числу версий модели минус один.

## Программа GPSS-модели

Предполагается, что количество типов СС в системе предоставления услуг связи может изменяться от одного до максимального значения  $n2\_$ . Программа модели, как уже омечалось при разработке AnyLogic-модели, построена для  $n2\_ = 5$ .

Для хранения результатов моделирования используются матрицы. В целях придания неизменности программы модели при варьировании

количеством типов СС матрицы должны быть пронумерованы. Однако GPSS World при описании матриц командой MATRIX не позволяет вместо имени указывать число. Матрицы нужно вначале описать, дав им имена, а затем пронумеровать.

Результаты моделирования для одного типа СС хранятся в семи матрицах, например, для СС типа 1 (СС1) с идентификаторами:

- Prib1 - матрица ожидаемой прибыли СС1;
- KPrl - матрица коэффициентов прибыли СС1;
- KZen1 - матрица коэффициентов использования СС1;
- DoxMax1 - матрица максимальных доходов от СС1;
- DoxDeg1 - матрица доходов от дежурства СС1;
- ZatrRem1 - матрица затрат на ремонт СС1;
- ZatrResSS1 - матрица затрат на резервные СС1.

Значит, аналогичных матриц будет тридцать пять. Да плюс еще матрица KRem коэффициентов использования мастеров - ремонтников. Именам этим тридцати шести матрицам даны номера.

Три матрицы для хранения суммарных результатов моделирования для СС всех типов не нумеровались:

- Pribil - матрица суммарной прибыли;
- SrKPrib - матрица средних коэффициентов прибыли СС всех типов;
- SrKIspl - матрица коэффициентов использования СС всех типов.

Именам пяти МКУ `CC1_, CC2_, CC3_, CC4_` и `CC5_`, имитирующими по типам СС, находящиеся на дежурстве, даны номера 1 ... 5. Это позволяет сократить число блоков в модели за счет того, что вместо пяти (в данном варианте модели) сегментов имитации постановки на дежурство СС используется один сегмент.

Для записи исходных данных: количества СС всех типов (в том числе и резервных), среднего времени наработка на отказ и среднего времени восстановления по типам СС, дохода по типам от одного СС, находящегося на дежурстве, убытка по типам при отсутствии одного СС

на дежурстве, стоимости по типам одного резервного СС используются функции KolSS, KolSSRes, NarOtk, SrVrRem, S1\_, S2\_, S3\_ соответственно. Этот способ по сравнению со способом использования матриц для записи этих же данных позволяет сократить программу модели на двадцать пять строк.

В начале работы модели генератор сразу вырабатывает количество транзактов, равное соответствующему количеству типов СС, и перестает быть активным.

Далее блоками SAVEVALUE и ASSIGN в параметр 1 каждого из транзактов последовательно заносятся коды 1..n1\_ - признак типа СС.

Затем каждый из транзактов с помощью блока SPLIT расщепляется (копируется, размножается) по количеству СС (с учетом резервных СС) соответствующего типа.

После расщепления транзакты в соответствии с типом СС сразу занимают все каналы МКУ, имитирующие нахождение СС на дежурстве. Резервные СС остаются в списках задержки соответствующих МКУ.

Вышедшее из строя СС снимается с дежурства, поступает в ремонтное подразделение - транзакт либо занимает свободный канал МКУ Rem, если такой есть, либо при отсутствии свободного канала помещается в список задержки этого МКУ - список транзактов, ожидающих возможность занять освободившиеся каналы МКУ.

После ремонта СС отправляется либо сразу на дежурство, либо в резерв. В обоих случаях транзакт направляется на метку Met1. Здесь также транзакт либо занимает свободный канал, либо помещается в список задержки МКУ, соответствующего типу СС.

Ниже приводится программа только (в целях сокращения) для моделирования случая три мастера-ремонтника, а резервных СС2 - четыре, пять и шесть.

```
; Модель функционирования системы связи  
; Задание номеров матрицам  
Prib1 EQU 1 ; Матрица ожидаемой прибыли СС1
```

KPr1 EQU 2 ; Матрица коэффициентов прибыли CC1  
 KZen1 EQU 3 ; Матрица коэффициентов использования CC1  
 DoxMax1 EQU 4 ; Матрица максимальных доходов CC1  
 DoxDeg1 EQU 5 ; Матрица доходов от дежурства CC1  
 ZatrRem1 EQU 6 ; Матрица затрат на ремонт CC1  
 ZatrResSS1 EQU 7 ; Матрица затрат на резервные CC1  
 Prib2 EQU 8 ; Матрица ожидаемой прибыли CC2  
 KPr2 EQU 9 ; Матрица коэффициентов прибыли CC2  
 KZen2 EQU 10 ; Матрица коэффициентов использования CC2  
 DoxMax2 EQU 11 ; Матрица максимальных доходов CC2  
 DoxDeg2 EQU 12 ; Матрица доходов от дежурства CC2  
 ZatrRem2 EQU 13 ; Матрица затрат на ремонт CC2  
 ZatrResSS2 EQU 14 ; Матрица затрат на резервные CC2  
 Prib3 EQU 15 ; Матрица ожидаемой прибыли CC3  
 KPr3 EQU 16 ; Матрица коэффициентов прибыли CC3  
 KZen3 EQU 17 ; Матрица коэффициентов использования CC3  
 DoxMax3 EQU 18 ; Матрица максимальных доходов CC3  
 DoxDeg3 EQU 19 ; Матрица доходов от дежурства CC3  
 ZatrRem3 EQU 20 ; Матрица затрат на ремонт CC3  
 ZatrResSS3 EQU 21 ; Матрица затрат на резервные CC3  
 Prib4 EQU 22 ; Матрица ожидаемой прибыли CC4  
 KPr4 EQU 23 ; Матрица коэффициентов прибыли CC4  
 KZen4 EQU 24 ; Матрица коэффициентов использования CC4  
 DoxMax4 EQU 25 ; Матрица максимальных доходов CC4  
 DoxDeg4 EQU 26 ; Матрица доходов от дежурства CC4  
 ZatrRem4 EQU 27 ; Матрица затрат на ремонт CC4  
 ZatrResSS4 EQU 28 ; Матрица затрат на резервные CC4  
 Prib5 EQU 29 ; Матрица ожидаемой прибыли CC5  
 KPr5 EQU 30 ; Матрица коэффициентов прибыли CC5  
 KZen5 EQU 31 ; Матрица коэффициентов использования CC5  
 DoxMax5 EQU 32 ; Матрица максимальных доходов CC5  
 DoxDeg5 EQU 33 ; Матрица доходов от дежурства CC5  
 ZatrRem5 EQU 34 ; Матрица затрат на ремонт CC5  
 ZatrResSS5 EQU 35 ; Матрица затрат на резервные CC5  
 KRem EQU 36 ; Матрица коэффициентов использования Rem  
 ; Задание номеров МКУ, имитирующих дежурство СС  
 CC1\_ EQU 1 ; Задание номера МКУ CC1  
 CC2\_ EQU 2 ; Задание номера МКУ CC2  
 CC3\_ EQU 3 ; Задание номера МКУ CC3

CC4\_ EQU 4 ; Задание номера МКУ CC4  
 CC5\_ EQU 5 ; Задание номера МКУ CC5  
 ; Задание по типам количества СС, находящихся на дежурстве  
 n2\_ EQU 5 ; Количество типов СС  
 VrMod EQU 1000 ; Время моделирования, 1 ед. мод. вр. = 1 час  
 KolProg EQU 1000  
 Stroka EQU 1 ; Номер строки матрицы  
 Stolbez EQU 1 ; Номер столбца матрицы  
 Prib1 MATRIX ,3,3 ; Матрица ожидаемой прибыли CC1  
 KPr1 MATRIX ,3,3 ; Матрица коэффициентов прибыли CC1  
 KZen1 MATRIX ,3,3 ; Матрица коэффициентов загрузки CC1  
 DoxMax1 MATRIX ,3,3 ; Матрица максимальных доходов CC1  
 DoxDeg1 MATRIX ,3,3 ; Матрица доходов от дежурства CC1  
 ZatrRem1 MATRIX ,3,3 ; Матрица затрат на ремонт CC1  
 ZatrResSS1 MATRIX ,3,3 ; Матрица затрат на резервные CC1  
 Prib2 MATRIX ,3,3 ; Матрица ожидаемой прибыли CC2  
 KPr2 MATRIX ,3,3 ; Матрица коэффициентов прибыли CC2  
 KZen2 MATRIX ,3,3 ; Матрица коэффициентов загрузки CC2  
 DoxMax2 MATRIX ,3,3 ; Матрица максимальных доходов CC2  
 DoxDeg2 MATRIX ,3,3 ; Матрица доходов от дежурства CC2  
 ZatrRem2 MATRIX ,3,3 ; Матрица затрат на ремонт CC2  
 ZatrResSS2 MATRIX ,3,3 ; Матрица затрат на резервные CC2  
 Prib3 MATRIX ,3,3 ; Матрица ожидаемой прибыли CC3  
 KPr3 MATRIX ,3,3 ; Матрица коэффициентов прибыли CC3  
 KZen3 MATRIX ,3,3 ; Матрица коэффициентов загрузки CC3  
 DoxMax3 MATRIX ,3,3 ; Матрица максимальных доходов CC3  
 DoxDeg3 MATRIX ,3,3 ; Матрица доходов от дежурства CC3  
 ZatrRem3 MATRIX ,3,3 ; Матрица затрат на ремонт CC3  
 ZatrResSS3 MATRIX ,3,3 ; Матрица затрат на резервные CC3  
 Prib4 MATRIX ,3,3 ; Матрица ожидаемой прибыли CC4  
 KPr4 MATRIX ,3,3 ; Матрица коэффициентов прибыли CC4  
 KZen4 MATRIX ,3,3 ; Матрица коэффициентов загрузки CC4  
 DoxMax4 MATRIX ,3,3 ; Матрица максимальных доходов CC4  
 DoxDeg4 MATRIX ,3,3 ; Матрица доходов от дежурства CC4  
 ZatrRem4 MATRIX ,3,3 ; Матрица затрат на ремонт CC4  
 ZatrResSS4 MATRIX ,3,3 ; Матрица затрат на резервные CC4  
 Prib5 MATRIX ,3,3 ; Матрица ожидаемой прибыли CC5  
 KPr5 MATRIX ,3,3 ; Матрица коэффициентов прибыли CC5  
 KZen5 MATRIX ,3,3 ; Матрица коэффициентов загрузки CC5

DoxMax5 MATRIX ,3,3 ; Матрица максимальных доходов CC5  
 DoxDeg5 MATRIX ,3,3 ; Матрица доходов от дежурства CC5  
 ZatrRem5 MATRIX ,3,3 ; Матрица затрат на ремонт CC5  
 ZatrResSS5 MATRIX ,3,3 ; Матрица затрат на резервные CC5  
 KRem MATRIX ,3,3 ; Матрица коэффициентов загрузки Rem  
 Pribil MATRIX ,3,3 ; Матрица суммарной прибыли  
 SrKPrib MATRIX ,3,3 ; Матрица коэффициентов прибыли CC всех типов  
 SrKIspl MATRIX ,3,3 ; Матрица коэффициентов загрузки CC всех типов  
 ;Определение MKU по количеству CC, находящихся на дежурстве  
 CC1\_ STORAGE 55 ; Емкость MKU по количеству CC1  
 CC2\_ STORAGE 100 ; Емкость MKU по количеству CC2  
 CC3\_ STORAGE 60 ; Емкость MKU по количеству CC3  
 CC4\_ STORAGE 45 ; Емкость MKU по количеству CC4  
 CC5\_ STORAGE 60 ; Емкость MKU по количеству CC5  
 Rem STORAGE 3 ; Емкость MKU по числу мастеров-ремонтников  
 ;Описание арифметических выражений  
 DoxMax VARIABLE VrMod#FN\$S1\_#FN\$KoSS  
 ; Максимальный доход от дежурства CC  
 Ubitok VARIABLE VrMod#FN\$KoSS#(1-(SR\*1/1000))#FN\$S2\_ ; Убыток  
 DoxDeg VARIABLE (AC1-P\$Nach1)#FN\$S1\_ ; Полученный доход от дежурства  
 StoRem VARIABLE (AC1-P\$Nach)#FN\$StoMast ; Стоимость ремонта на CC  
 ZatrResSS VARIABLE FN\$S3\_#FN\$KoSSRes#VrMod ; Затраты на содержание CC  
 SumPrib VARIABLE X\$DoxDeg-(X\$ZatrRem+MX\*4(Stroka,Stolbez)+VSU)  
 KoefPr VARIABLE MX\*6(Stroka,Stolbez)/X\$DoxMax ; Коэффициент прибыли  
 StoMast FUNCTION P1,D5 ; Стоимость работы одного мастера  
 1,17/2,18/3,16/4,20/5,21  
 KoSS FUNCTION P1,D5 ; Количество по типам CC, находящихся на дежурстве  
 1,55/2,100/3,60/4,45/5,60  
 KoSSRes FUNCTION P1,D5 ; Количество по типам резервных CC  
 1,2/2,4/3,4/4,3/5,4  
 NarOtk FUNCTION P1,D5 ; Среднее время наработки до отказа по типам CC  
 1,373/2,301/3,482/4,325/5,470  
 SrVrRem FUNCTION P1,D5 ; Среднее время ремонта по типам CC, ч  
 1,6,5/2,4,2/3,2,8/4,3/5,5,5  
 S1\_ FUNCTION P1,D5 ; Доход по типам от одного CC, находящегося на дежурстве  
 1,20/2,24,2/3,32,8/4,23/5,25,5  
 S2\_ FUNCTION P1,D5 ; Убыток по типам при отсутствии одного CC  
 1,32/2,34,2/3,37/4,31/5,32,5  
 S3\_ FUNCTION P1,D5 ; Затраты по типам на содержание одного резервного CC

1,21/2,24,2/3,28/4,26/5,25,5

; Сегмент постановки на дежурство СС

GENERATE „,n2\_

SAVEVALUE TipSS+,1 ; Код 1 ... n2\_ - признак CC1 CCn2 в XSTipSS

ASSIGN 1,XSTipSS ; Код 1 ... n2\_ - признак CC1 CCn2\_ в P1

SPLIT (FN\$KoLSS+FN\$KoLSSRes-1) ; Число СС + резервные СС

; Сегмент имитации дежурства СС

Met1 ENTER P1 ; Встать на дежурство СС типа, номер которого в P1

ASSIGN Nach1,AC1 ; Время начала дежурства

ADVANCE (Exponential(5672,0,FNSNarOtk)) ; Имитация выхода СС из

LEAVE P1 ; Снятие с дежурства из-за выхода из строя СС типа, номер

ASSIGN 3,0

Met3 ASSIGN 3+,1 ; Начало цикла изменения типов СС

TESTE P1,P3,Met4 ; P1=P3?

ASSIGN 4,(P1#7) ; P4=P1#7

ASSIGN 5,(P4-2) ; Номера матриц: 5,12,19,26,33

MSAVEVALUE \*5+,Stroka,Stolbez,V\$DoxDeg

; MSAVEVALUE Pribil+,Stroka,Stolbez,V\$DoxDeg

Met4 TEST GE P3,n2\_,Met3 ; Все ли типы СС?

; Сегмент имитации работы ремонтного подразделения

ENTER Rem ; Занять одного мастера

ASSIGN Nach,AC1 ; Время начала дежурства

ADVANCE (Exponential(5672,0,FNSSrVrRem)) ; Имитация ремонта

LEAVE Rem ; Конец ремонта

ASSIGN 3,0

Met5 ASSIGN 3+,1 ; Начало цикла изменения типов СС

TESTE P1,P3,Met6 ; P1=P3?

ASSIGN 4,(P1#7) ; P4=P1#7

ASSIGN 5,(P4-1) ; Номера матриц: 6,13,20,27,34

MSAVEVALUE \*5+,Stroka,Stolbez,V\$StoRem

; MSAVEVALUE Pribil+,Stroka,Stolbez,V\$StoRem

Met6 TEST GE P3,n2\_,Met5 ; Все ли типы СС?

TRANSFER ,Met1 ; Направить исправное СС на дежурство или в рез

; Сегмент задания времени моделирования и расчета результатов

GENERATE VrMod

TESTE TG1,1,Met2

ASSIGN 1,0

Met7 ASSIGN 1+,1 ; Начало цикла изменения типов СС

ASSIGN 2,(P1#7) ; P2=P1#7

```

ASSIGN 9,(P2-2) ; Номера матриц: 5,12,19,26,33
SAVEVALUEDoxDeg,((MX*9(Stroka,Stolbez))/KoIProg);Доход от дежурст
MSAVEVALUE *9,Stroka,Stolbez,X$DoxDeg
ASSIGN 3,(P2-3) ; Номера матриц: 4,11,18,25,32
MSAVEVALUE *3,Stroka,Stolbez,V$DoxMax; Максимально возможный
SAVEVALUE DoxMax,MX*3(Stroka,Stolbez)
ASSIGN 4,(P2-0) ; Номера матриц: 7,14,21,28,35
MSAVEVALUE *4,Stroka,Stolbez,V$ZatrResSS ; Затраты на резервные (
ASSIGN 5,(P2-1) ; Номера матриц: 6,13,20,27,34
SAVEVALUE ZatrRem,((MX*5(Stroka,Stolbez))/KoIProg)
MSAVEVALUE *5,Stroka,Stolbez,X$ZatrRem
ASSIGN 6,(P2-6) ; Номера матриц: 1,8,15,22,29
MSAVEVALUE *6,Stroka,Stolbez,V$SumPrib
ASSIGN 7,(P2-5) ; Номера матриц: 2,9,16,23,24
MSAVEVALUE *7,Stroka,Stolbez,V$KoeffPr
MSAVEVALUE Pribil+,Stroka,Stolbez,
MX*6(Stroka,Stolbez) ; Суммарная прибыль по СС всех типов
ASSIGN 8,(P2-4)
MSAVEVALUE SrKPrib+,Stroka,Stolbez,(MX*7(Stroka,Stolbez)/n2_) ; Ср
MSAVEVALUE *8,Stroka,Stolbez,(SR*1/1000)
; Коэффициент использования СС
MSAVEVALUE SrKlsp+,Stroka,Stolbez,(SR*1/(1000#n2_)) ; Средний ко
TEST GE P1,n2_,Met7 ; Все ли типы СС?
MSAVEVALUE 31,Stroka,Stolbez,(SR$Rem/1000); Коэффициент использ
SAVEVALUE TipSS,0
Met2 TERMINATE 1
START 1000,NP ; Вариант1: CCP2=4, мастеров=3
KoIStres FUNCTION P1,D5
1,2/2,5/3,4/4,4/5,4
Stolbez EQU 2
CLEAR OFF
START 1000,NP ; Вариант2: CCP2=5, мастеров=3
KoIStres FUNCTION P1,D5
1,2/2,6/3,4/4,4/5,4
Stolbez EQU 3
CLEAR OFF
START 1000,NP ; Вариант3: CCP2=6, мастеров=3
KoIStres FUNCTION P1,D5
1,2/2,4/3,4/4,4/5,4

```

```

Stroka EQU 2
Stolbez EQU 1
Rem STORAGE 4 ; Емкость МКУ по числу мастеров-ремонтников
CLEAR OFF
START 1000,NP ; Вариант4: CCP2=4, мастеров=4
KolSSres FUNCTION P1,D5
1,2/2,5/3,4/4,4/5,4
Stolbez EQU 2
CLEAR OFF
START 1000,NP ; Вариант5: CCP2=5, мастеров=4
KolSSres FUNCTION P1,D5
1,2/2,6/3,4/4,4/5,4
Stolbez EQU 3
CLEAR OFF
START 1000,NP ; Вариант6: CCP2=6, мастеров=4
KolSSres FUNCTION P1,D5
1,2/2,4/3,4/4,4/5,4
Stroka EQU 3
Stolbez EQU 1
Rem STORAGE 5 ; Емкость МКУ по числу мастеров-ремонтников
CLEAR OFF
START 1000,NP ; Вариант7: CCP2=4, мастеров=5
KolSSres FUNCTION P1,D5
1,2/2,5/3,4/4,4/5,4
Stolbez EQU 2
CLEAR OFF
START 1000,NP ; Вариант8: CCP2=5, мастеров=5
KolSSres FUNCTION P1,D5
1,2/2,6/3,4/4,4/5,4
Stolbez EQU 3
CLEAR OFF
START 1000 ; Вариант9: CCP2=6, мастеров=5

```

В программе, кроме методов применения матриц и функций, показывается метод изменения версий модели. Изменение версий модели производится переопределением соответствующих блоков. Переопределяться не могут только блоки GENERATE. Для переопределения блоков, описывающих ОКУ и МКУ, они должны иметь метки. В командах START, кроме последней, указывается операнд

В - NP - не выводить отчет. Однако одного переопределения блоков недостаточно. В GPSS World изменение версий модели достигается также за счет использования команды CLEAR.

В рассматриваемом примере в процессе моделирования необходимо собирать статистику по версиям модели. Нужная собранная статистика должна быть сохранена, а ненужная - сброшена.

Процесс моделирования в исходное состояние возвращает команда CLEAR. Формат записи команды:

CLEAR [A]

Операнд A может быть ON либо OFF. По умолчанию - ON.

Команда CLEAR сбрасывает всю накопленные статистические данные, удаляет все транзакты из процесса моделирования и заполняет все блоки GENERATE первым транзактом. ОКУ и МКУ становятся доступными, устанавливаются в незанятое состояние. Содержимое всех блоков становится нулевым. Генераторы случайных чисел не сбрасываются.

Если в команде CLEAR операнд A равен OFF, то сохраняемые ячейки, матрицы и логические ключи остаются без изменений. Поэтому в модели в команде CLEAR используется операнд A, равный OFF, так как нужно сохранить результаты моделирования предыдущей версии модели.

Однако при этом нужно иметь в виду те ячейки, начальные значения которых должны быть нулевыми в новой версии модели. Необходимо предусмотреть в программе блоки приведения таких ячеек в исходное состояние. В данной модели это показано на примере сохраняемой ячейки TipSS. Если эту ячейку не привести в нулевое состояние, процесс моделирования второй версии будет остановлен по ошибке "Обращение к несуществующей памяти".

Поскольку накопленные и сохраненные в матрицах результаты моделирования нет необходимости выводить после каждого варианта, то в команде START используется операнд B, равный ON. В последней команде START операнд B не используется. Поэтому стандартный

отчет выдаётся после завершения моделирования. В рассматриваемом примере - после девяти наблюдений.

Ниже показан фрагмент журнала с информацией о ходе моделирования. В первом наблюдении (первой версии модели) модельное время изменяется от 0 до 1 000 000 единиц модельного времени (1000 прогонов # 1000 часов работы моделируемой системы).

После переопределения (формирования второй версии модели) выполняется команда CLEAR и абсолютное модельное время вновь изменяется от 0 до 1 000 000 единиц модельного времени.

```
07/11/11 16:56:06 Model Translation Begun.  
07/11/11 16:56:06 Ready.  
07/11/11 16:56:06 Simulation in Progress.  
07/11/11 16:56:26 The Simulation has ended. Clock is 1000000.000000.  
07/11/11 16:56:26 Simulation in Progress.  
07/11/11 16:56:46 The Simulation has ended. Clock is 1000000.000000.  
07/11/11 16:56:46 Simulation in Progress.  
07/11/11 16:57:06 The Simulation has ended. Clock is 1000000.000000.  
07/11/11 16:57:06 Simulation in Progress.  
07/11/11 16:57:31 The Simulation has ended. Clock is 1000000.000000.  
07/11/11 16:57:31 Simulation in Progress.  
07/11/11 16:57:55 The Simulation has ended. Clock is 1000000.000000.  
07/11/11 16:57:55 Simulation in Progress.  
07/11/11 16:58:19 The Simulation has ended. Clock is 1000000.000000.  
07/11/11 16:58:19 Simulation in Progress.  
07/11/11 16:58:44 The Simulation has ended. Clock is 1000000.000000.  
07/11/11 16:58:44 Simulation in Progress.  
07/11/11 16:59:09 The Simulation has ended. Clock is 1000000.000000.  
07/11/11 16:59:09 Simulation in Progress.  
07/11/11 16:59:33 The Simulation has ended. Clock is 1000000.000000.  
07/11/11 16:59:33 Reporting in Модель сеть связи 3.128.1-REPORT.
```

Для включения в формируемый стандартный отчет матриц необходимо при открытом объекте "Модель" выполнить команду:

Edit/Settings/Reports/Matrices/Применить/Ok

Результаты моделирования после 1000 прогонов приведены ниже. Но

опять в целях сокращения приведены для СС1 первые три матрицы с абсолютной и относительной прибылью, а также с коэффициентами использования СС1.

MATRIX	RETRY	INDICES	VALUE
PRIB1	0	1 1	411920.775
		1 2	412047.879
		1 3	395398.298
		2 1	975100.990
		2 2	973737.015
		2 3	972026.043
		3 1	1029846.786
		3 2	1029847.984
		3 3	1029916.195
KPR1	0		
		1 1	.374
		1 2	.374
		1 3	.359
		2 1	.886
		2 2	.885
		2 3	.883
		3 1	.936
		3 2	.936
		3 3	.936
KZEN1	0		
		1 1	.778
		1 2	.778
		1 3	.772
		2 1	.976
		2 2	.976
		2 3	.975
		3 1	.995
		3 2	.995
		3 3	.995

## 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 А этой команды, задающий ёмкость МКУ, должен быть только числом. Факторы же эксперимента обязательно должны быть переменными пользователя и не могут быть на месте операнда А. 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 (CCP2) 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 CCP2 и мастеров-ремонтников для условий данных экспериментов, то можно выбрать вариант 7: CCP2 = 4, мастеров-ремонтников = 5.

Таблица 5.6. Показатели функционирования системы связи

Показатели	GPSS World					AnyLogic				
	Типы средств связи									
	CC1	CC2	CC3	CC4	CC5	CC1	CC2	CC3	CC4	CC5
<b>Вариант 1: CCP2 = 4, мастеров-ремонтников = 3</b>										
Коэффициент прибыли по типам СС	0,374	0,327	0,627	0,4	0,573	0,38	0,332	0,633	0,405	0,57
Суммарный коэффициент прибыли	0,46						0,468			
Коэффициент использования по типам СС	0,779	0,741	0,853	0,78	0,846	0,781	0,743	0,856	0,781	0,84
Суммарный коэффициент использования СС	0,799						0,8			
<b>Вариант 2: CCP2 = 5, мастеров-ремонтников = 3</b>										
Коэффициент прибыли по типам СС	0,377	0,336	0,626	0,397	0,571	0,369	0,332	0,624	0,393	0,57
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	0,84
Суммарный коэффициент использования СС	0,801						0,8			
<b>Вариант 3: CCP2 = 6, мастеров-ремонтников = 3</b>										
Коэффициент прибыли по	0,364	0,33	0,619	0,385	0,564	0,361	0,331	0,62	0,39	0,56

типам СС									
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 0,85	0,775 0,84							
Суммарный коэффициент использования СС	0,797				0,797				
<b>Вариант 4: CCP2 = 4, мастеров-ремонтников = 4</b>									
Коэффициент прибыли по типам СС	0,89 0,895 0,931 0,89 0,911 0,889 0,892 0,931 0,889 0,91								
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 0,99								
Суммарный коэффициент использования СС	0,987				0,987				
<b>Вариант 5: CCP2 = 5, мастеров-ремонтников = 4</b>									
Коэффициент прибыли по типам СС	0,887 0,892 0,93 0,887 0,909 0,885 0,89 0,93 0,886 0,90								
Суммарный коэффициент прибыли	0,901				0,902				
Коэффициент использования	0,977 0,981 0,996 0,988 0,994 0,976 0,98 0,995 0,987 0,99								

по типам СС									
Суммарный коэффициент использования СС	0,987					0,986			
Вариант 6: CCP2 = 6, мастеров-ремонтников = 4									
Коэффициент прибыли по типам СС	0,885	0,89	0,93	0,886	0,909	0,887	0,89	0,93	0,887
Суммарный коэффициент прибыли	0,9					0,903			
Коэффициент использования СС	0,977	0,984	0,995	0,987	0,994	0,977	0,984	0,996	0,988
Суммарный коэффициент использования СС	0,987				0,987				
Вариант 7: CCP2 = 4, мастеров-ремонтников = 5									
Коэффициент прибыли по типам СС	0,935	0,944	0,939	0,914	0,922	0,936	0,945	0,94	0,915
Суммарный коэффициент прибыли	0,931					0,934			
Коэффициент использования СС	0,996	0,998	1	0,999	1	0,996	0,998	1	0,999
Суммарный коэффициент использования СС	0,998				0,998				
Вариант 8: CCP2 = 5, мастеров-ремонтников = 5									

Коэффициент прибыли по типам СС	0,936 0,936 0,939 0,915 0,922 0,936 0,936 0,94	0,915 0,92
Суммарный коэффициент прибыли	0,93	0,932
Коэффициент использования по типам СС	0,996 0,999 1 0,999 1	0,996 0,999 1 0,999 1
Суммарный коэффициент использования СС	0,998	0,998

**Вариант 9: CCP2 = 6, мастеров-ремонтников = 5**

Коэффициент прибыли по типам СС	0,936 0,927 0,939 0,915 0,922 0,936 0,928 0,94	0,915 0,92
Суммарный коэффициент прибыли	0,928	0,929
Коэффициент использования по типам СС	0,996 0,999 1 0,999 1	0,996 0,999 1 0,999 1
Суммарный коэффициент использования СС	0,998	0,998

# Модель функционирования предприятия

## Модель в GPSS World

### Постановка задачи

Предприятие имеет  $n_1$  цехов, производящих  $n_1$  типов блоков, т. е. каждый цех производит блоки одного типа. Себестоимости комплектующих блоков  $C_{k1}, C_{k2}, \dots, C_{kn_1}$ . Стоимости изготовления блоков  $C_{изг1}, C_{изг2}, \dots, C_{изгn_1}$ . Интервалы выпуска блоков  $T_1, T_2, \dots, T_{n_1}$  - случайные. Из  $n_1$  блоков собирается одно изделие.

Перед сборкой каждый тип блоков проверяется на  $n_{11}, n_{12}, \dots, n_{1n}$  соответствующих постах контроля. Длительности контроля одного блока  $T_{11}, T_{12}, \dots, T_{1n}$  случайные. Стоимости проверки блоков  $C_{пр1}, C_{пр2}, \dots, C_{прn_1}$ . На каждом посту бракуется  $q_{11}, q_{12}, \dots, q_{1n}$  % 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  $T_c$  случайное. Стоимость сборки одного изделия  $C_c$ .

После сборки изделие поступает на один из  $n_3$  стендов выходного контроля. На одном стенде одновременно проверяется только одно изделие. Время проверки  $T_h$  случайное. Стоимость проверки одного изделия  $C_h$ . По результатам проверки бракуется  $q_2$  % изделий. В таком изделии с вероятностью  $q_3$  % могут быть забракованы  $m$  блоков. Вероятности порядковых номеров из  $1 \dots n_1$   $P_{6,11} \dots P_{6,n_1}$  соответственно.

Забракованное изделие направляется в цех сборки, где неработоспособные блоки заменяются новыми. Время замены  $T_{зам}$  случайное. Стоимость замены 1-го блока  $C_{зам1}$ . 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  $C_{пр}$ . По результатам приемки бракуется  $q_4$  % изделий, которые направляются вновь на стенд выходного контроля. Принятые приёмкой изделия направляются на склад предприятия.

## Исходные данные

$$\begin{aligned} n_1 &= 4; T_1 = 19; T_2 = 11; T_3 = 15; T_4 = 18; \\ C_{k1} &= 35; C_{k2} = 32; C_{k3} = 43; C_{k4} = 48; \\ C_{изр1} &= 35; C_{изр2} = 27; C_{изр3} = 36; C_{изр4} = 37; \\ n_{11} &= 2; n_{12} = 2; n_{13} = 2; n_{14} = 2; T_{11} = 12; T_{12} = 16; T_{13} = 21; T_{14} = 17; \\ C_{вр1} &= 12; C_{вр2} = 23; C_{вр3} = 32; C_{вр4} = 28; \\ q_{11} &= 0,02; q_{12} = 0,03; q_{13} = 0,04; q_{14} = 0,06; \\ n_2 &= 2; T_c = 22; C_{c6} = 67; n_3 = 2; T_n = 26; C_k = 74; q_2 = 0,05; \\ n_2 &= 2; T_c = 22; C_{c6} = 67; n_3 = 2; T_n = 26; C_k = 74; q_2 = 0,05; \\ m &= 1; p_{k1} = 1,0; P_{бн1} = 0,25; P_{бн2} = 0,25; P_{бн3} = 0,25; P_{бн4} = 0,25; \\ T_{зам1} &= 12; T_{зам2} = 15; T_{зам3} = 12; T_{зам4} = 21; \\ C_{зам1} &= 34; C_{зам2} = 46; C_{зам3} = 38; C_{зам4} = 54; \\ n_4 &= 2; T_{пр} = 18; C_{пр} = 53; q_4 = 0,15. \end{aligned}$$

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

## 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.

## 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 ее 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 забрашенных блоков;
- выходные потоки готовых изделий.

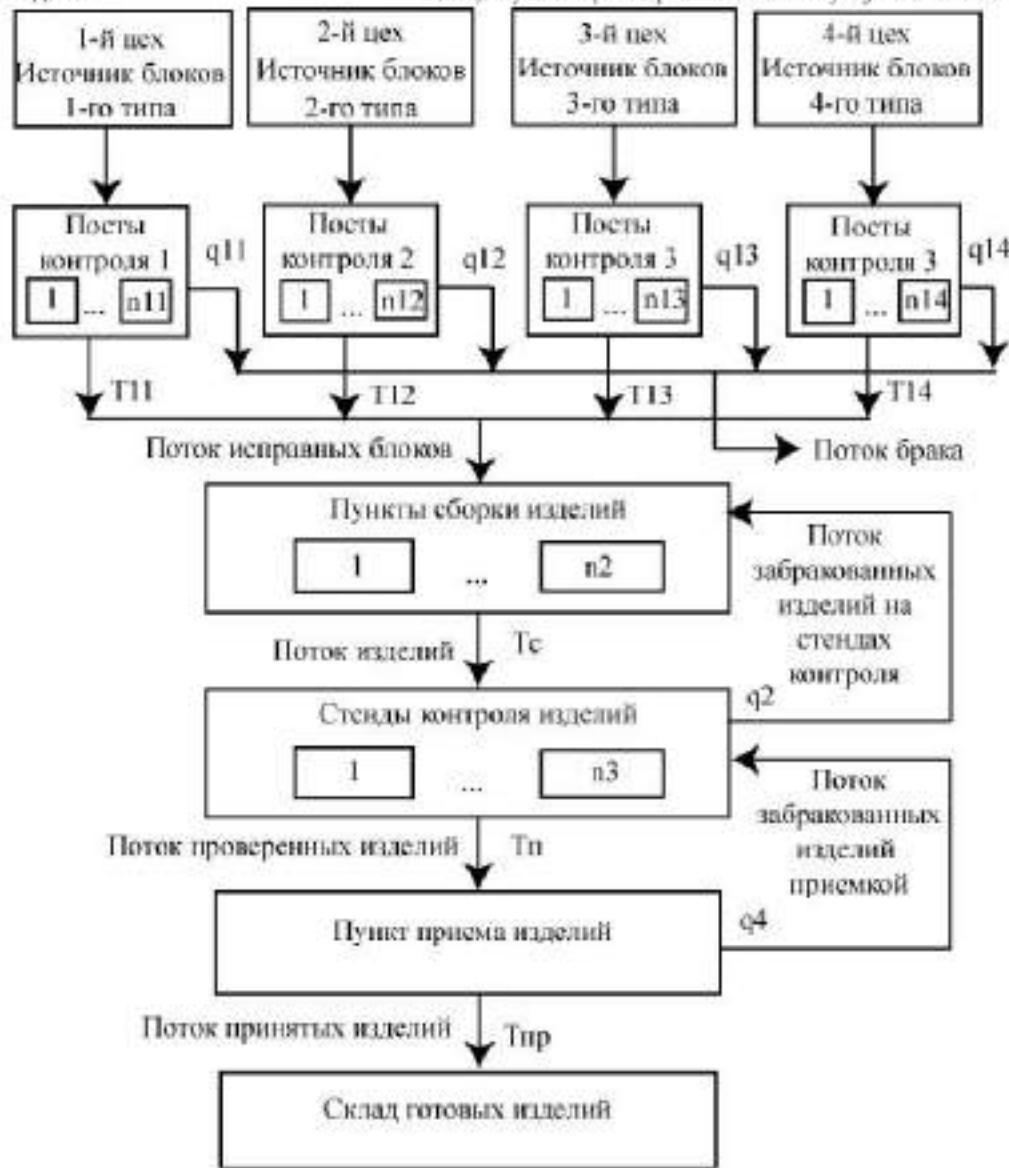


Рис. 6.1. Предприятие как система массового обслуживания

Для имитации МКУ следует использовать блоки ENTER и LEAVE.

Для исходных данных в программе модели возьмем те же идентификаторы, что и в постановке задачи, но для предотвращения случаев совпадения с зарезервированными символами GPSS World добавим символ подчеркивания.

Например, `q11_n1_`. Сделаем это для отличия от зарезервированных символов GPSS World: `q` - системный числовой атрибут, означающий очередь, `n` - используется в качестве ссылки при определении количества транзактов, вошедших в какой-либо блок программы. Добавление символа подчёркивания предотвратит ошибку, которая в противном случае будет выявлена на этапе создания объекта "Процесс моделирования". Другие идентификаторы будем вводить по мере уяснения задачи, а также в ходе разработки программы модели.

Для моделирования необходимо привести в соответствие время протекания реального процесса изготовления блоков и сборки изделий на предприятии и в модели. Это осуществляется введением масштабного коэффициента, например, если для условий рассматриваемой задачи его взять равным 1, а в реальном процессе измерять время в минутах, то 1 мин будет соответствовать 1 ед. мод. вр. Тогда время моделирования  $VtMod = 60 \# 40 = 2400$  ед. мод. вр. Временные параметры изготовления и контроля блоков, сборки, контроля и приёмки изделий даны в минутах, поэтому при выбранном масштабном коэффициенте 1 они не изменятся.

В модели, как процесса, протекающего в СМО (см. рис. 6.1), необходимо иметь:

- задание исходных данных;
- сегмент имитации работы цехов без постов контроля;
- сегмент имитации работы постов контроля блоков;
- сегмент имитации сборки изделий;
- сегмент имитации работы стендов выходного контроля;
- сегмент имитации работы приемки;
- сегмент задания времени моделирования и расчета результатов моделирования.

Для ввода исходных данных целесообразно использовать команды `EQU` и `FUNCTION`. Вторая команда позволит сократить число строк в программе за счёт ввода одномерного массива данных двумя строками.

## Программа модели

При разработке программы модели следует исходить из того, что отдельные элементы модели и модель в целом имеют достаточно различимое подобие со структурой предприятия ([рис. 6.1](#)).

Это подобие может быть также усилено разработчиком за счет продуманного на этапе разработки разделения исследуемого объекта на элементы, на процессы, протекающие в них, а модели - на сегменты.

Однако возможно и другое. В данном примере можно было бы иметь сегменты имитации работы цехов по их количеству, т. е. четыре сегмента. С целью придания универсальности в модели имеется один сегмент имитации работы цехов. При увеличении (уменьшении) количества цехов необходимо только изменить значение переменной пользователя TipBl - число типов блоков (по числу цехов).

Также в примере в каждом цехе имеются свои посты контроля блоков. Поэтому казалось, в модели должны были бы быть сегменты, имитирующие работу цеха и его постов контроля. По предложенному же составу модели видно, что в неё входит сегмент, имитирующий работу каждого из цехов без постов контроля, и сегмент имитации работы всех постов контроля. Т. е. как бы все посты контроля блоков объединены в отдельное подразделение предприятия, но функциональное предназначение соответствующих постов контроля осталось прежним.

Объединение сделано также в интересах универсальности модели. Предположим, количество цехов увеличилось. Нужно было бы добавлять сегменты имитации работы цехов и постов контроля, т. е. количество блоков в модели увеличилось бы. В предлагаемом варианте сегмент имитации функционирования постов контроля блоков остаётся неизменным. Необходимо только командами STORAGE задать ёмкости добавляемых пунктов контроля, заменить их имена номерами и добавить данные: среднее время контроля блоков, доля браков и стоимость контроля блоков в функции TKontr, BrBl и CProv соответственно.

Списки пользователя применяются для имитации работы складов готовых блоков. Предполагается наличие такого склада у каждого цеха.

Для розыгрыша выхода в брак блоков и изделий используется блок

**TRANSFER** в статистическом режиме.

Обратите внимание, что в сегменте имитации сборки изделий блок TEST используется в режиме, который рекомендуется избегать вследствие того, что проверяемое условие может не выполниться. Однако здесь этого не должно быть, так как в противном случае будут отсутствовать готовые блоки для сборки изделий. По мере готовности блоков условие обязательно выполняется и блоки - транзакты направляются на сборку. Первые три транзакта уничтожаются, а четвертый транзакт имитирует собранное из четырех блоков изделие. Он направляется для проверки работоспособности на пункт приема изделий.

**Замечание.** Не путайте блоки изделий с блоками GPSS World в программе модели.

В программе использование МКУ и списков пользователя демонстрируется применением номеров МКУ вместо их имен. Этот метод дает возможность иметь в модели один сегмент имитации работы постов контроля блоков вместо подобного сегмента для каждого цеха, т. е. сократить число блоков в модели.

Обратите внимание, что в программе присвоение номеров именам МКУ указывается в самом начале и только потом, не обязательно следом, определение МКУ командой STORAGE. Если вы построите программу так, что поменяете порядок: вначале определение МКУ командой STORAGE, а потом - присвоение командой EQU номеров именам МКУ, то на этапе выполнения программы модели возникнет ошибка: "Обращение к несуществующей памяти". На этапе создания объекта "Процесс моделирования" ошибка изменения этого порядка не обнаруживается.

Ниже приводится программа модели.

```
; Модель функционирования предприятия
; Замена имен МКУ номерами
Kontr1 EQU 1
Kontr2 EQU 2
Kontr3 EQU 3
Kontr4 EQU 4
```

; Задание исходных данных

TShop FUNCTION P1,D4 ; Средние интервалы времени изготовления  
1,19/2,11/3,15/4,18

CKom FUNCTION P1,D4 ; Стоимости комплектующих блоков  
1,35/2,32/3,43/4,48

Clzg FUNCTION P1,D4 ; Стоимости изготовления блоков  
1,35/2,27/3,36/4,37

TKont FUNCTION P1,D4 ; Среднее время контроля на постах  
1,12/2,16/3,21/4,17

BrBl FUNCTION P1,D4 ; Доли брака блоков на постах контроля  
1,.02/2,.03/3,.04/4,.06

CProv FUNCTION P1,D4 ; Стоимости контроля блоков  
1,12/2,23/3,32/4,28

TZam FUNCTION P1,D4 ; Среднее время замены блоков  
1,12/2,15/3,12/4,21

CZam FUNCTION P1,D4 ; Стоимости замены блоков  
1,34/2,46/3,38/4,54

VerBl FUNCTION RN339,D2 ; Вероятности брака одного или двух блоков  
.0,2/1,1

VerBlNum FUNCTION RN339,D4; Вероятность брака блока номер ...  
.25,1/.5,2/.75,3/1,4

CCb EQU 67 ; Стоимость сборки изделия

q2\_ EQU 0.05 ; Доля забракованных изделий на пункте выходного контроля

q4\_ EQU 0.15 ; Доля забракованных изделий приемкой

TpBl EQU 4 ; Max количество типов блоков

VrMod EQU 2400 ; Время моделирования, 1 ед. мод. вр. = 1 мин

Tc\_ EQU 22 ; Среднее время сборки изделия

Tp EQU 26 ; Среднее время проверки изделия

Tpr EQU 18 ; Среднее время приема изделия

CPr EQU 53 ; Стоимость приемки изделия

CK EQU 74 ; Стоимость контроля изделия

; Задание количества пунктов сборки и постов контроля

Sb STORAGE 2 ; Количество пунктов сборки

Kontr1 STORAGE 2 ; Количество постов n11

Kontr2 STORAGE 2 ; Количество постов n12

Kontr3 STORAGE 2 ; Количество постов n13

Kontr4 STORAGE 2 ; Количество постов n14

KSb STORAGE 3 ; Количество стендов выходного контроля

KPr STORAGE 2 ; Количество пунктов приемки

; Описание арифметических выражений  
 Kollzd VARIABLE (NSTerm7/X\$prog) ; Количество готовых изделий  
 KolGotBl VARIABLE ((CH\*1/X\$Prog)) ; Количество готовых блоков все  
 KolBrBl VARIABLE ((X\*1/X\$Prog)) ; Количество забракованных блоков  
 Tlzd VARIABLE (AC1/X\$Prog)/X\$Kollzd; Среднее время подготовки одн  
 ;Сегмент имитации работы цехов без постов контроля  
 GENERATE „TipBl ; Число транзактов по числу типов блоков  
 SAVEVALUE Tip+,1 ; Пронумеровать типы блоков  
 ASSIGN 1,X\$Tip ; Код в параметре транзакта - тип блока  
 Met20 ADVANCE (Exponential(27,0,FN\$TShop)); Розыгрыш интервала п  
 SPLIT 1,Met20 ; Расщепление на два  
 ASSIGN 2,(Exponential(339,0,FN\$TKom))  
 ; Розыгрыш времени контроля и запись в P2  
 ASSIGN 9,FN\$BrBl ; Запись в P9 доли брака блоков после постов кон  
 ASSIGN Sbor,(Exponential(339,0,Tc\_)); Запись времени сборки изделия  
 ASSIGN Cost,(FN\$CKom+FN\$CI2g); Стоимость комплектующих+стои  
 ; Сегмент имитации работы постов контроля блоков  
 Met1 QUEUE P1 ; Встать в очередь с номером в P1  
 ENTER P1 ; Занять МКУ с номером в P1  
 DEPART P1 ; Покинуть очередь с номером в P1  
 ADVANCE P2;Имитация контроля с временем в P2  
 LEAVE P1 ; Освободить МКУ с номером в P1  
 ASSIGN Cost+,FN\$CProv ; Добавить стоимость проверки  
 TRANSFER P9,,Met14 ; Отправить брак блоков к Met14  
 LINK P1,FIFO ; Готовые блоки на склад с номером в P1  
 ; Сегмент имитации сборки изделий  
 GENERATE „,1  
 Met3 ASSIGN 1,TipBl ; Подготовка к циклу  
 SAVEVALUE Cost,0 ; Обнуление ячейки Cost  
 Met13 TEST NE CH\*1,0 ; Есть ли на складе готовые блоки?  
 TEST NE P1,1,Met4 ; Если последний блок, то на Met4  
 UNLINK P1,Met32,1 ; Нет  
 TRANSFER ,Met31  
 Met4 UNLINK P1,Met22,1 ; Отправить блоки на сборку  
 Met31 LOOP 1,Met13  
 TRANSFER ,Met3 ; Вернуться для проверки наличия всех типов блок  
 Met32 SAVEVALUE Cost+,P\$Cost ; Суммарная стоимость 2-го, 3-го и +  
 SAVEVALUE Cost1,X\$Cost ; Суммарная стоимость 2-го, 3-го и 4-го бл

TRANSFER ,Term5 ; Отправить  
 Met22 ASSIGN Cost+,X\$Cost1 ; Суммарная стоимость 1-го...4-го блок  
 SAVEVALUE Cost2,P\$Cost ; Суммарная стоимость 1-го...4-го блоков  
 ASSIGN Zam,0  
 Met5 QUEUE Sbor ; Занять очередь на пункты сборки  
 ENTER Sb ; Занять пункт сборки  
 DEPART Sbor ; Освободить очередь на пункт сборки  
 ADVANCE P\$Sbor ; Имитация сборки  
 LEAVE Sb ; Освободить пункт сборки  
 TEST E P\$Zam,0,Met9  
 ASSIGN Cost+,CCb ; Добавить стоимость сборки изделия  
 ; Сегмент имитации работы стендов выходного контроля  
 Met9 QUEUE KSbor ; Занять очередь на стенд выходного контроля  
 ENTER KSb ; Занять стенд выходного контроля  
 DEPART KSbor ; Освободить очередь на стенд выходного контроля  
 ADVANCE (Exponential(339,0,Tp)); Имитация работы стендов выходного  
 LEAVE KSb ; Освободить стенд выходного контроля  
 ASSIGN Cost+,CK ; Добавить стоимость контроля изделия  
 TRANSFER q2\_,Met34,Met33 ; Направить в приёмку а брак-на подгот  
 ; Подготовка к замене блоков  
 Met33 ASSIGN KolBl,FNSVerBl ; Розыгрыш количества забракованных  
 ASSIGN Sbor,0 ; Обнуление параметра Sbor  
 Met21 ASSIGN 1,FN\$VerBlNum ; Розыгрыш номера блока  
 TEST NE CH\*1,0 ; Есть на складе готовые блоки?  
 UNLINK P1,Met14,1 ; Да. Тогда блок на замену  
 ASSIGN Sbor,(Exponential(339,0,FN\$TZam)); Розыгрыш времени замены  
 ASSIGN Zam,1  
 ASSIGN Cost+,FN\$CZam ; Добавить стоимость замены блока  
 TRANSFER ,Met5 ; Отправить на пункты сборки  
 ; Сегмент имитации работы приёмки  
 Met34 QUEUE Opr ; Занять очередь в приёмку  
 ENTER KPr ; Занять приемку  
 DEPART Opr ; Освободить очередь в приёмку  
 ADVANCE (Exponential(339,0,Tpr)) ; Имитация работы приемки  
 LEAVE KPr ; Освободить приемку  
 ASSIGN Cost+,CPr ; Добавить стоимость приемки изделия  
 TRANSFER q4\_,Met9 ; Готовые изделия - на склад  
 ; Сегмент счёта блоков и изделий  
 SAVEVALUE CostIzd+,P\$Cost

Term7 TERMINATE ; Количество готовых изделий за все прогоны мод  
 Met14 SAVEVALUE P1+,1 ; Количество забракованных блоков по типу:  
 ASSIGN 5,(TipBl+P1)  
 SAVEVALUE \*5+,PSCost; Стоимости забракованных блоков по типам  
 SAVEVALUE CostBr+,P\$Cost ; Стоимость забракованных блоков за вс.  
 TERMINATE  
 Term5 TERMINATE  
 ; Задание времени моделирования и расчет результатов  
 GENERATE VrMod ; Задание времени моделирования  
 TEST L X\$Prog,TG1,Met10 ; Если X\$Prog < содержимого счетчика завершения  
 SAVEVALUE Prog,TG1 ; записать в X\$Prog содержимое счетчика завершения  
 Met10 TEST E TG1,1,Met12 ; Если содержимое счетчика завершений равно  
 SAVEVALUE Kollzd,V\$Kollzd ; Количество готовых изделий  
 ASSIGN 1,0 ; Подготовка к циклу  
 Met15 ASSIGN 1+,1 ; Начало цикла по числу типов блоков  
 SAVEVALUE (10+P1),V\$KolGotBl ; Количество готовых блоков всех типов  
 SAVEVALUE P1,V\$KolBrBl ; Количество забракованных блоков всех типов  
 ASSIGN 2,(TipBl+P1)  
 SAVEVALUE P2,((X\*2)/X\$Prog); Стоимости забракованных блоков по типам  
 SAVEVALUE StBl+, (FN\$CKom+FN\$CIzg+FN\$CProv); Стоимость блоков  
 TEST GE P1,TipBl,Met15 ; Все ли типы блоков?  
 SAVEVALUE Cmin,(X\$StBl+CCb+CK+CPr) ; Минимальная стоимость блоков  
 SAVEVALUE MinCGotIzd,(X\$Cmin#X\$Kollzd) ; Минимальная стоимость блоков  
 SAVEVALUE CostIzd,(X\$CostIzd/X\$Prog) ; Стоимость готовых изделий  
 SAVEVALUE TIzd,V\$TIzd ; Среднее время подготовки одного изделия  
 SAVEVALUE CostBr,(X\$CostBr/X\$Prog) ; Стоимость забракованных блоков  
 SAVEVALUE Koef,((X\$CostIzd+X\$CostBr)/X\$MinCGotIzd); Коэффициент  
 Met12 TERMINATE 1  
 START 1000

Отладьте модель. Выполните моделирование. В отчете, фрагмент которого приведен ниже,

	SAVEVALUE	RETRY	VALUE
KOLIZD	0	121.628	
TIZD	0	19.732	
COSTBR	0	3091.787	
COSTIZD	0	74410.065	
CMIN	0	582.000	

MINCGOTIZD 0 70787.496

KOEF 0 1.095

найдите, что за 40 часов подготовлено 121,628 изделия, а среднее время подготовки одного изделия  $\approx 20$  мин (19,732 мин). Минимальная стоимость готовых изделий 70787,496, стоимость брака 3091,787, стоимость готовых изделий 74410,065. Минимальная стоимость одного изделия - 582. Коэффициент увеличения стоимости одного изделия составил 1,095.

Другие эксперименты будут проведены в п. 6.3. Данные этого же эксперимента внесены в табл. 6.10.

## Модель функционирования предприятия в AnyLogic

### 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;
- **signal** - 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 timeSbor 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 стенах контроля и пунктах приема изделий;
  - $allBrakBlock1..allBrakBlock4$  - всего забракованных блоков 1...4;
  - $minCostIzd$  - минимальная стоимость одного изделия;
  - $minCostGotIzd$  - минимальная стоимость готовых изделий;
  - $kolGotIzd$  - 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;
- $costBrakBlock$  - суммарные затраты на все забракованные блоки;
- $costGotIzd$  - затраты на выпуск готовых изделий;
- $costBlockIzd$  - стоимость блоков одного изделия;
- $costIzd$  - стоимость одного изделия;

- $timeIzd$  - среднее время изготовления одного изделия;
- $koeffIncrCostIzd$  - коэффициент увеличения себестоимости изделия.

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

$$C_{min} = C_{minIzd} \cdot N_{изд.},$$

где  $N_{изд.}$  - количество готовых изделий (поступивших на склад);

$C_{minIzd}$  - минимальная себестоимость производства одного изделия, вычисляется по формуле:

$$C_{minIzd} = \sum_{i=1}^{nI} (C_{ri} + C_{изгi} + C_{прi}) + C_{сб} + C_{к} + C_{д}$$

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

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

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

$costBlock1 = stKomplBlock1 + stIzqBlock1 + stTestBlock1;$   
 $costBlock2 = stKomplBlock2 + stIzqBlock2 + stTestBlock2;$   
 $costBlock3 = stKomplBlock3 + stIzqBlock3 + stTestBlock3;$   
 $costBlock4 = stKomplBlock4 + stIzqBlock4 + stTestBlock4;$   
 $costBlockIzd = costBlock1 + costBlock2 + costBlock3 + costBlock4;$   
 $C_{minIzd} = costBlockIzd + stSborki + stKontrIzd + stPriebIzd;$   
 $C_{max} = C_{minIzd} \cdot colGotIzd; C_{max} = costGotIzd + costBrakBlock.$

$$K_c = koeffIncrCostIzd = \frac{C_{max}}{C_{min}}$$

Замечание. В приведенных для расчета результатов моделирования

выражениях мы не учитывали те блоки, которые были произведены, но не были использованы для сборки изделий.

## 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 Enterprise. Выберите каталог, в котором будут сохранены файлы модели. Щелкните кнопку Далее.
3. На открывшейся второй странице Мастера создания модели выберите Начать создание модели "с нуля". Щелкните кнопку Далее.
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. Перейдите на страницу Дополнительные панели Свойства. Введите в поля X: 0, Y: 1700, Ширина: 760, Высота: 330.
7. Перетащите элемент Прямоугольник. Оставьте имя, предложенное системой.
8. Перейдите на страницу Дополнительные 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: 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
aveTimeShop1	double	19	stKomplBlock1	double
aveTimeShop2	double	11	stKomplBlock2	double
aveTimeShop3	double	15	stKomplBlock3	double
aveTimeShop4	double	18	stKomplBlock4	double
stIzgBlock1	double	35	timeTestBlock1	double
stIzgBlock2	double	27	timeTestBlock2	double
stIzgBlock3	double	36	timeTestBlock3	double
stIzgBlock4	double	37	timeTestBlock4	double
postKontr1	int	2	procBrakBlock1	double
postKontr2	int	2	procBrakBlock2	double
postKontr3	int	2	procBrakBlock3	double
postKontr4	int	2	procBrakBlock4	double
stTestBlock1	double	12	kolStendKontrIzd	int
stTestBlock2	double	23	timeKontrIzd	double
stTestBlock3	double	32	procBrakIzd	double
stTestBlock4	double	28	stKontrIzd	double
kolPunSborki	int	2	verBlock1	double
timeSborki	double	22	verBlock2	double
stSborki	double	67		
verBlockNum1	double	0,25	timeZamBlock1	double
verBlockNum2	double	0,25	timeZamBlock2	double
verBlockNum3	double	0,25	timeZamBlock3	double
verBlockNum4	double	0,25	timeZamBlock4	double
stZamBlock1	double	34	kolPunPriem	int
stZamBlock2	double	46	timePriemIzd	double
stZamBlock3	double	38	procBrakPriem	double
stZamBlock4	double	54	stPriemIzd	double

## Предприятия

### Исходные данные


Рис. 6.2. Размещение элементов Параметр для ввода исходных данных

## Вывод результатов моделирования

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

1. В Палитре выделите Презентация. Создайте область просмотра Результаты для размещения элементов Простая переменная.
  2. Перетащите элемент Область просмотра. В поле Имя: введите Результаты.
  3. Перейдите на страницу Дополнительные панели Свойства. Введите в поля X: 0, Y: 700, Ширина: 700, Высота: 460.
  4. Перетащите элемент Скруглённый прямоугольник. Оставьте имя, предложенное системой.
  5. Перейдите на страницу Дополнительные панели Свойства. Введите в поля X: 10, Y: 750, Ширина: 680, Высота: 400.
  6. Перетащите элемент text и на странице Основные панели Свойства в поле Текст: введите Данные о количестве

подготовленных и забракованных блоков и изделий.

7. Перетащите второй элемент `text` и на странице Основные панели Свойства в поле Текст: введите Стоимостные показатели функционирования предприятия.
8. В Палитре выделите Основная. Перетащите элементы Простая переменная. 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 - `double`. Значение по умолчанию равно 0.
9. 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.

## Предприятие

### Данные о количестве подготовленных и забракованных блоков и изделий

1) lotGoodBlock	1) rottenBlock	1) transBlock	1) goodBlock	1) outGoodBlock	1) lotGoodBlock	1) transBlock
1) lotGoodBlock2	1) rottenBlock2	1) transBlock2	1) goodBlock2	1) outGoodBlock2	1) lotGoodBlock2	1) transBlock2
1) lotGoodBlock3	1) rottenBlock3	1) transBlock3	1) goodBlock3	1) outGoodBlock3	1) lotGoodBlock3	1) transBlock3
1) lotGoodBlock4	1) rottenBlock4	1) transBlock4	1) goodBlock4	1) outGoodBlock4	1) lotGoodBlock4	1) transBlock4
1) #BlockBlock	1) #BadBlock2	1) #BadBlock3	1) #BadBlock4	1) #BadBlock5	1) #BadBlock6	1) #BadBlock7

### Стоимостные показатели функционирования предприятия

1) costKompBlock	1) costKompBlock2	1) costKompBlock3	1) costKompBlock4	1) costKompBlock5	1) costBlock	1) costBlock2
1) costDigBlock1	1) costDigBlock2	1) costDigBlock3	1) costDigBlock4	1) costDigBlock5	1) costDigBlock6	1) costDigBlock7
1) costTestBlock1	1) costTestBlock2	1) costTestBlock3	1) costTestBlock4	1) costTestBlock5	1) costTestBlock6	1) costTestBlock7
1) sumCostBlock1	1) sumCostBlock2	1) sumCostBlock3	1) sumCostBlock4	1) sumCostBlock5	1) costBlocktot	1) costBlocktot
1) costShorBlock	1) costShorBlock2	1) costPrikBlock	1) costPrikBlock2	1) costPrikBlock3	1) sumCostBlock	1) meCostBlock
1) minCostGoto1	1) costGoto2	1) costGotoBlock	1) costGotoBlock2	1) costGotoBlock3	1) timeGoto	1) ?

Рис. 6.3. Размещение элементов Простая переменная для вывода результатов моделирования

## 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 дальнейшем 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:

- Перейдите на страницу Основные панели Свойства.
- В поле Имя : введите Mainview.
- Перейдите на страницу Дополнительные панели Свойства. Введите в поля X : 0, Y : 0, Ширина : 990, Высота : 390.
- Перетащите элемент Скруглённый прямоугольник. В нём мы разместим все сегменты модели. Оставьте имя, предложенное системой.
- Перейдите на страницу Дополнительные панели Свойства. Введите в поля X : 10, Y : 60, Ширина : 970, Высота : 320.
- 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

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

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

8. Оставьте выбранными флажки Создать конструктор и Создать метод `toString()`.
9. Щелкните кнопку Готово. Появится редактор кода и автоматически созданный код вашего Java класса. Закройте код.

10. Выделите объект *source*. На странице Основные панели Свойства установите свойства согласно табл. 6.3.

Таблица 6.3. Свойства объектов *source*

Имя	Свойства	Значения
	Отображать имя	Установите флажок
	Класс заявки	Product
	Заявки прибывают согласно	
	Время между прибытиями	Времени между прибытиями
	Время между прибытиями	Exponential (1/aveTimeShop1)
	Количество заявок, прибывающих за один раз	1
Цех1	Новая заявка	<pre> new 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;</pre>
	Действие при выходе	

	<pre> entity.timeSbor = exponential(1/timeSborki); costKomplBlock1 += stKomplBlock1; costKomplBlock += stKomplBlock1; costIzgBlock1 += stIzgBlock1; sumCostBlock1 += (stKomplBlock1+stIzgBlock1); costIzgBlock += stIzgBlock1; sumCostBlock += (stKomplBlock1+stIzgBlock1); </pre>
Отображать имя	Установите флагок
Класс заявки	Product
Заявки прибывают согласно	Времени между прибытиями
Время между прибытиями	Exponential(1/aveTimeShop2)
Количество заявок, прибывающих за один раз	1
Новая заявка	new Product()
цех2	<pre> kolIzgBlock2++; entity.numBlock = 2; costKomplBlock2 += stKomplBlock2; costKomplBlock += stKomplBlock2; costIzgBlock2 += stIzgBlock2; sumCostBlock2 += (stKomplBlock2+stIzgBlock2); </pre>
Действие при выходе	

```
costIzgBlock += stIzgBlock2;
```

```
sumCostBlock +=
```

```
(stKomplBlock2+stIzgBlock2);
```

	Отображать имя	Установите флагок
	Класс заявки	Product
	Заявки прибывают согласно	Времени между прибытиями
	Время между прибытиями	Exponential(1/aveTimeShop3)
	Количество заявок, прибывающих за один раз	1
цех3	Новая заявка	<pre>new Product();</pre> <pre>kolIzgBlock3++;</pre> <pre>entity.numBlock = 3;</pre> <pre>costKomplBlock3 += stKomplBlock3;</pre> <pre>costKomplBlock += stKomplBlock3;</pre> <pre>costIzgBlock3 += stIzgBlock3;</pre> <pre>sumCostBlock3 += (stKomplBlock3+stIzgBlock3);</pre> <pre>costIzgBlock += stIzgBlock3;</pre> <pre>sumCostBlock += (stKomplBlock3+stIzgBlock3);</pre>
	Действие при выходе	<pre>costIzgBlock3 += stIzgBlock3;</pre> <pre>sumCostBlock3 += (stKomplBlock3+stIzgBlock3);</pre> <pre>costIzgBlock += stIzgBlock3;</pre> <pre>sumCostBlock += (stKomplBlock3+stIzgBlock3);</pre>
	Отображать имя	Установите флагок
	Класс заявки	Product
	Заявки прибывают согласно	Времени между прибытиями
	Время между	Exponential(1/aveTimeShop4)

	прибытиями	
	Количество заявок, прибывающих за один раз	1
цех4	Новая заявка	<pre>new Product ()</pre>
		<pre>koltzgBlock4++; entity.numBlock = 4; costKomplBlock4 += stKomplBlock4; costKomplBlock += stKomplBlock4; costIzgBlock4 += stIzgBlock4; sumCostBlock4 += (stKomplBlock4+stIzgBlock4); costIzgBlock += stIzgBlock4; sumCostBlock += (stKomplBlock4+stIzgBlock4);</pre>
	Действие при выходе	

- Щелкните выделенный source правой кнопкой мыши и в контекстном меню выберите Копировать.
- 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. Поэтому последовательно выделяя второй, третий и четвертый объекты source, скорректируйте их свойства также согласно табл. 6.3.

### 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 класс активного объекта Test.

1. На панели Проект щелкните Main правой кнопкой мыши и выберите из контекстного меню Создать / Класс активного объекта. Откроется окно Новый класс активного объекта.
2. В поле Имя: задайте имя нового класса Test.
3. Если нужно, в поле Описание: введите описание сущности, моделируемой этим классом. Щелкните кнопку Готово.

Создайте область просмотра на диаграмме класса Test 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. Перейдите на страницу Дополнительные 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 элемент нового класса активного объекта Test.

1. Из Палитры Основная перетащите элемент Порт и разместите сверху в крайнем левом ряду (рис. 6.5).

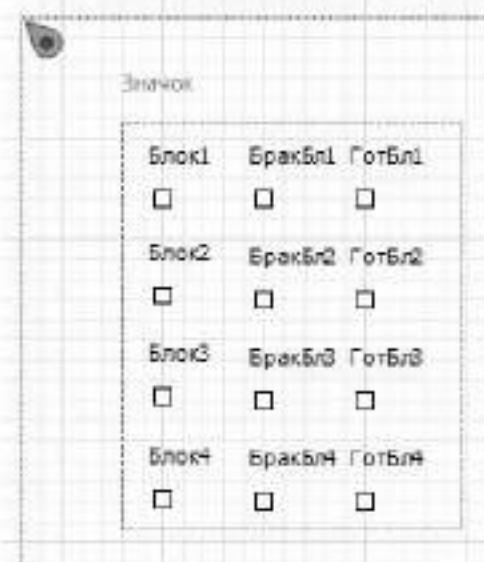


Рис. 6.5. Размещение элементов Порт на объекте Test

2. На странице Основные панели Свойства имя `port` замените именем **Блок1**.
3. Скопируйте элемент Порт с именем **Блок1**.
4. Вставьте три элемента Порт (см. [рис. 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 класса Main.
15. На панели Проект выделите Test, перетащите на диаграмму класса Main элемент класса Test, 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. Элемент диаграммы активного класса Test создан. Возвратитесь на диаграмму класса Test.

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.

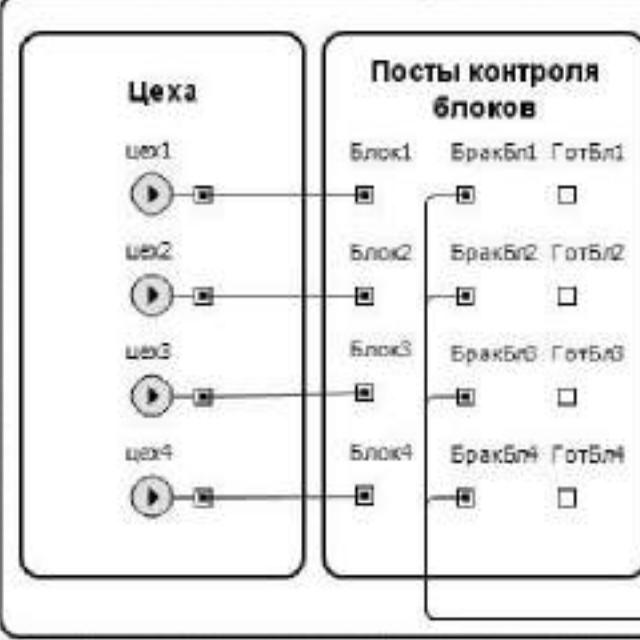


Рис. 6.6. Добавлен элемент диаграммы класса Test

1. Перетащите элемент Скруглённый прямоугольник. В нём мы разместим все объекты сегмента имитации работы постов контроля блоков. Оставьте имя, предложенное системой.
2. Перейдите на страницу Дополнительные панели Свойства. Введите в поля X : 10, Y : 50, Ширина : 570, Высота : 330.

Добавьте на диаграмму класса Test объекты класса Queue.

1. Из библиотеки Enterprise Library 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 Класс 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 склИзгBl1, как на рис. 6.7.

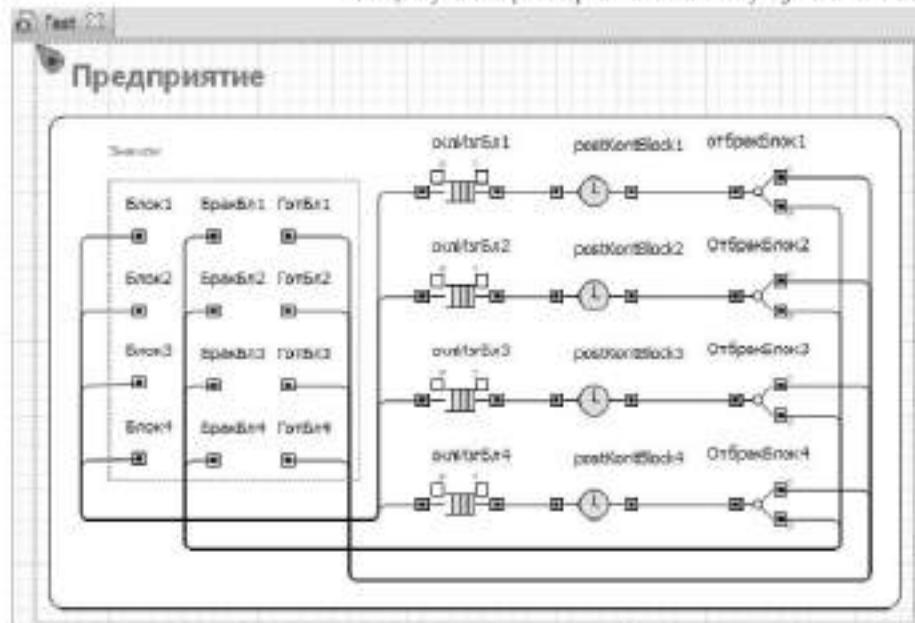


Рис. 6.7. Размещение элементов на диаграмме активного класса Test

2. На странице Основные панели Свойства замените имя `delay` именем `постКонтрБлок1` (пости контроля блоков цеха 1).
3. В поле Класс заявки: Entity замените `Product`.
4. Введите в поле Время задержки: `exponential (1/get_Main().timeTestBlock1)`.
5. Введите `get_Main().postKontr1` в поле Вместимость:.
6. Действие при выходе

```
get_Main().kolTestBlock1++;
get_Main().sumCostBlock1 += get_Main().stTestBlock1;
get_Main().costTestBlock1+=get_Main().stTestBlock1;
get_Main().costTestBlock1 += get_Main().stTestBlock1;
get_Main().sumCostBlock1 += get_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 действие при выходе.

Таблица 6.4.

Имя	Время задержки	Вместимость
постКонтрБлок2	exponential (1/get_Main().timeTestBlock2)	get_Main postKont
постКонтрБлок3	exponential (1/get_Main().timeTestBlock3)	get_Main postKont
постКонтрБлок4	exponential (1/get_Main().timeTestBlock4)	get_Main postKont

Добавьте на диаграмму Test объекты класса SelectOutput.

- Перетащите объект selectOutput и разместите справа рядом с объектом постКонтрБлок1, как на [рис. 6.7](#).
- На странице Основные панели Свойства замените имя selectOutPut именем ОтбракБлок1 (отбраковка блоков цеха 1).
- Установите свойства объекта согласно [табл. 6.5](#).
- Скопируйте объект с именем ОтбракБлок1.
- Вставьте три объекта класса SelectOutput (см. [рис. 6.7](#)).
- Последовательно выделите вставленные объекты и скорректируйте значения их свойств согласно [табл. 6.5](#).
- Соедините входы и выходы объектов диаграммы класса Test согласно [рис. 6.7](#).

Код в свойство Действие при выходе (true) введен для учёта, например, для цеха 1:

- gotBlock1 - количества готовых блоков цеха 1.

Код в свойстве Действие при выходе (false) обеспечивает счёт количества brakBlock1 забракованных блоков цеха 1.

Таблица 6.5.

Свойства	Значение
Имя	ОтбракБлок1
Класс заявки	Product

Выход true выбирается	С заданной вероятностью
Вероятность [0...1]	1-get_Main().procBrakBlock1
Действие при выходе (true)	get_Main().gotBlock1++; get_Main().brakBlock1++;
Действие при выходе (false)	entity.sign1 = 1; entity.numBlBrak1 = 1;
Имя	ОтБракБлок2
Класс заявки	Product
Выход true выбирается	С заданной вероятностью
Вероятность [0...1]	1-get_Main().procBrakBlock2
Действие при выходе (true)	get_Main().gotBlock2++; get_Main().brakBlock2++;
Действие при выходе (false)	entity.sign1 = 1; entity.numBlBrak2 = 1;
Имя	ОтБракБлок3
Класс заявки	Product
Выход true выбирается	С заданной вероятностью
Вероятность [0...1]	1-get_Main().procBrakBlock3
Действие при выходе (true)	get_Main().gotBlock3++; get_Main().brakBlock3++;
Действие при выходе (false)	entity.sign1 = 1; entity.numBlBrak3 = 1;
Имя	ОтБракБлок4
Класс заявки	Product
Выход true выбирается	С заданной вероятностью
Вероятность [0...1]	1-get_Main().procBrakBlock4
Действие при выходе (true)	get_Main().gotBlock4++; get_Main().brakBlock4++;
Действие при выходе (false)	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 класса Main.

### 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 класс активного объекта Sborka.

1. На панели Проект щелкните Main правой кнопкой мыши и выберите из контекстного меню Создать/Класс активного объекта.
2. Откроется окно Новый класс активного объекта.
3. В поле Имя: задайте имя нового класса Sborka.
4. Если нужно, в поле Описание: введите описание сущности, моделируемой этим классом.
5. Щелкните кнопку Готово.

Создайте область просмотра на диаграмме класса Sborka для размещения 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. Перейдите на страницу Дополнительные 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. Перейдите на страницу Дополнительные 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 приема готовых блоков с постов контроля и один порт - для отправки собранных изделий на стенды контроля. Также нужен порт для приема бракованных изделий с целью определения и замены в них забракованных блоков и один порт - для отправки забракованных блоков после их замены на склад забракованных блоков.

1. Из Палитры Основная перетащите элемент Порт и разместите сверху в крайнем левом ряду ([рис. 6.8](#)).
2. На странице Основные панели Свойства имя port замените именем ГотБлок1. Скопируйте элемент Порт с именем ГотБлок1.
3. Вставьте три элемента Порт (см. [рис. 6.8](#)). При вставке последовательно будут изменяться их имена: ГотБлок2, ГотБлок3, ГотБлок4.
4. Из Палитры Основная перетащите еще три элемента Порт, разместите их и дайте имена как на [рис. 6.8](#).
5. Вернитесь на диаграмму класса Main.
6. На панели Проект выделите Sborka, перетащите на диаграмму класса Main элемент sborka, разместите и соедините так, как на [рис. 6.9](#). Порт БракБл соедините с объектом sink сегмента Склад бракованных блоков.
7. Элемент диаграммы активного класса Sborka создан. Возвратитесь на диаграмму класса Sborka.

Для реализации сегмента имитации работы пунктов сборки изделий нам потребуются объекты, но прежде мы рассмотрим вариант построения этого сегмента.

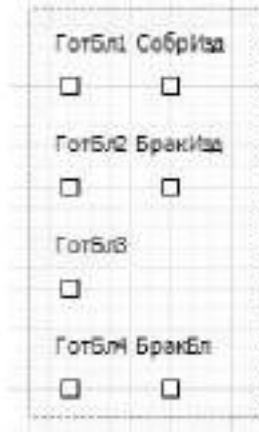


Рис. 6.8. Размещение элементов Порт на объекте Sborka



Рис. 6.9. Добавлен элемент диаграммы класса Sborka

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 дальнейшей 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. Перейдите на страницу **Дополнительные панели Свойства**. Введите в поля X: 320, Y: 80, Ширина: 340, Высота: 230.
3. Из библиотеки **Enterprise Library** 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 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*:
  - Классы заявок: Вход1, Вход2: *Product*, *Product*
  - Условия соответствия *true*
  - Максимальная вместимость 1
  - Максимальная вместимость 2

- `combine..combine2:`
  - Классы заявок: Вход1, Вход2, Выход: Product, Product, Product
  - Объединенная заявка entity1
- `queue:`
  - Класс заявки: Product
  - Максимальная вместимость
- `selectOutput:`
  - Класс заявки: Product
  - Выход true выбирается при выполнении условия
  - Условие `entity.signal == 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 нее ранее было определено значение поля `entity.timeSbor` (см. табл. 6.3).

Изделия после элемента `пунктСборки` разделяются на два потока: собранные изделия первично (`entity.signal=0`) и изделия с замененными блоками (`entity.signal=2`).

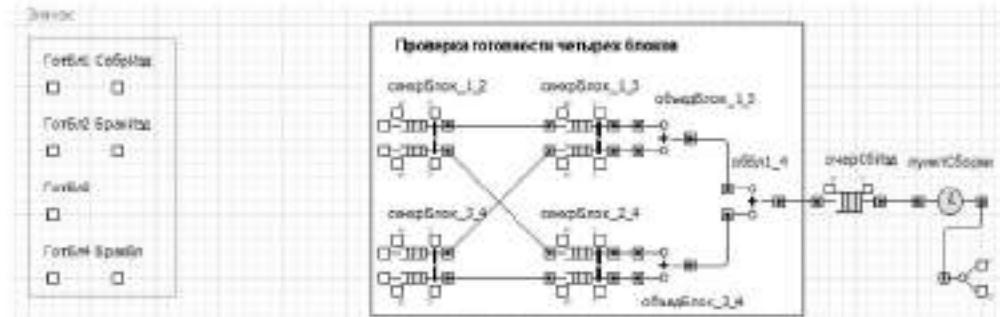


Рис. 6.10. Фрагмент сегмента имитации работы пунктов сборки

Разделение, осуществляющееся объектом `selectOutPut` по условию `entity.sign1==0`, нужно для учёта количества изделий с заменёнными блоками и количества заменённых блоков.

Свойства объекта `delay` установите согласно табл. 6.6.

Таблица 6.6.

Имя	Свойства	Значения
	Отображать имя	Установите флагок
	Класс заявки	Product
	Задержка задаётся	Явно
	Время задержки	<code>entity.timeSbor</code>
пунктборки	Вместимость	<pre>get_Main().kolFunSborki if (entity.sign1 == 0) {get_Main().kolSobrIzd++; get_Main().costSborIzd += get_Main().stSborki; entity.cost += get_Main().stSborki;} if (entity.sign1 == 2) {if (entity.numBlBrak1== 1) {get_Main().zamBlock1++; entity.cost += get_Main().stZamBlock1; get_Main().costSborIzd += get_Main().stZamBlock1;} if (entity.numBlBrak2 == 1) {get_Main().zamBlock2++; entity.cost += get_Main().stZamBlock2; get_Main().costSborIzd += get_Main().stZamBlock2;}}</pre>
	Действие при выходе	

```
if (entity.numBlBrak3 == 1)
    (get_Main().zamBlock3++);
entity.cost += 
get_Main().stZamBlock3;
get_Main().costSborIzd += 
get_Main().stZamBlock3;
if (entity.numBlBrak4 == 1)
    (get_Main().zamBlock4++);
entity.cost += 
get_Main().stZamBlock4;
get_Main().costSborIzd += 
get_Main().stZamBlock4;
get_Main().kolZamIzd++; }
```

Продолжим построение сегмента имитации работы пунктов сборки изделий ([рис. 6.11](#)).



Рис. 6.11. Сегмент имитации пунктов сборки изделий

Бракованные изделия поступают через порт БракИзд. В заявке, имитирующей такое изделие, поле `entity.signal1 = 2`. В одном из полей `entity.numBlBrak1 ... entity.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 пять условий 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 соответствующими 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 пункт`Сборки`.

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
<b>selectOutPut ... selectOutPut3</b>	
Имя	ГотБлок1
Выход true выбирается	При выполнении условия
Условие	БракИздБл1 == 0
Имя	ГотБлок2
Выход true выбирается	При выполнении условия
Условие	БракИздБл2 == 0
Имя	ГотБлок3
Выход true выбирается	При выполнении условия
Условие	БракИздБл3 == 0
Имя	ГотБлок4
Выход true выбирается	При выполнении условия
Условие	БракИздБл4 == 0
<b>selectOutput5</b>	
Имя	БракИзделия
Использовать	Условия
Условие 0	entity.numBlBrak1 == 1
Условие 1	entity.numBlBrak2 == 1
Условие 2	entity.numBlBrak3 == 1
Условие 3	entity.numBlBrak4 == 1
<b>match</b>	
Имя	синхрБлок_1_2
Классы заявок: Вход1, Вход2:	Product, Product
Условие соответствия	true
Максимальная	Установить флагок

вместимость 1	Установить флажок
Максимальная вместимость 2	Установить флажок
Имя	синхрБлок_3_4
Классы заявок: Вход1, Вход2:	Product, Product
Условие соответствия	true
Максимальная вместимость 1	Установить флажок
Максимальная вместимость 2	Установить флажок
Имя	синхрБлок_1_3
Классы заявок: Вход1, Вход2:	Product, Product
Условие соответствия	true
Максимальная вместимость 1	Установить флажок
Максимальная вместимость 2	Установить флажок
Имя	синхрБлок_2_4
Классы заявок: Вход1, Вход2:	Product, Product
Условие соответствия	true
Максимальная вместимость 1	Установить флажок
Максимальная вместимость 2	Установить флажок
<b>combine</b>	
Имя	издБлок1
Классы заявок: Вход1, Вход2, Выход	Product, Product, Product
Действие при входе 1	if (entity.numBlBrak1 == 1)

Объединенная заявка	<pre>entity1 if (entity.numBlBrak1 == 1) БрИздБл1 --; entity.cost += get_Main().stZamBlock1; entity.timeSbor = exponential(1/get_Main() . timeZamBlock1);</pre>
Действие при выходе	<pre>entity1 if (entity.numBlBrak1 == 1) БрИздБл1 --; entity.cost += get_Main().stZamBlock1; entity.timeSbor = exponential(1/get_Main() . timeZamBlock1);</pre>
Имя	издБлок2
Классы заявок: Вход1, Вход2, Выход	Product, Product, Product
Действие при входе 1	<pre>if (entity.numBlBrak2 == 1) БрИздБл2++; entity1 if (entity.numBlBrak2 == 1) БрИздБл2 --; entity.cost += get_Main().stZamBlock2; entity.timeSbor = exponential(1/get_Main() . timeZamBlock2);</pre>
Объединенная заявка	<pre>entity1 if (entity.numBlBrak2 == 1) БрИздБл2 --; entity.cost += get_Main().stZamBlock2; entity.timeSbor = exponential(1/get_Main() . timeZamBlock2);</pre>
Действие при выходе	<pre>entity1 if (entity.numBlBrak2 == 1) БрИздБл2 --; entity.cost += get_Main().stZamBlock2; entity.timeSbor = exponential(1/get_Main() . timeZamBlock2);</pre>
Имя	издБлок3
Классы заявок: Вход1, Вход2, Выход	Product, Product, Product
Действие при входе 1	<pre>if (entity.numBlBrak3 == 1) БрИздБл3++; entity1 if (entity.numBlBrak3 == 1) БрИздБл3 --; entity.cost += get_Main().stZamBlock3; entity.timeSbor = exponential(1/get_Main() .</pre>
Объединенная заявка	<pre>entity1 if (entity.numBlBrak3 == 1) БрИздБл3 --; entity.cost += get_Main().stZamBlock3; entity.timeSbor = exponential(1/get_Main() .</pre>
Действие при выходе	<pre>entity1 if (entity.numBlBrak3 == 1) БрИздБл3 --; entity.cost += get_Main().stZamBlock3; entity.timeSbor = exponential(1/get_Main() .</pre>

<b>Имя</b>	издВлок4
<b>Классы заявок:</b> Вход1, Вход2, Выход	Product, Product, Product
<b>Действие при входе 1</b>	if !entity.numBlBrak4 == 1) БрИздБл1 ++;
<b>Объединенная заявка</b>	entity1  if !entity.numBlBrak4 == 1) БрИздБл4 --;  entity.cost += get_Main().stZamBlock4;  entity.timeSbor = exponential(1/get_Main(). timeZamBlock4);  <b>split</b>
<b>Действие при выходе</b>	
<b>Классы заявок:</b>	Product, Product
<b>Количество копий</b>	1
<b>Новая заявка (копия)</b>	new Product()
<b>Действие при выходе копии</b>	entity.sign1 = 0; entity.cost = original.cost;

### 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 стенах контроля изделий, вследствие недостатка ресурсов, будет создаваться очередь, для имитации которой используйте объект `queue`.

Для имитации непосредственно процесса контроля изделий 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 класса `Main`.
2. Разместите и соедините их согласно [рис. 6.12](#).
3. Выделите объект `queue` и установите на странице Основные панели Свойства:
  - Имя: очерСтендКонтр
  - Класс заявки: `Product`
  - Максимальная вместимость Установите флажок
4. Выделите объект `delay` и установите его свойства:
  - Класс заявки: `Product`
  - Задержка задается явно
  - Время задержки `exponential( 1/timeKontrIzd )`
  - Вместимость `kolStendKontr`
  - Действие при выходе

```
testSobrIzd++;
entity.cost += stKontrIzd;
entity.numBlBrak1 = 0;
entity.numBlBrak2 = 0;
entity.numBlBrak3 = 0;
entity.numBlBrak4 = 0;
```



Рис. 6.12. Диаграмма класса Main с элементами всех сегментов

5. Выделите объект selectOutput и установите его свойства:

- Имя: БрСтендКон
- Класс заявки: Product
- Выход true выбирается с заданной вероятностью
- Вероятность[0..1] 1-procBrakIzd
- Действие при выходе (true) costTestIzd += stKontrIzd;
- Действие при выходе (false)

```
double a = 0;
int numBlock = 0;
entity.sign1 = 2;
a = 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 Действие 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.signal` присваивается значение 2 - признак брака в изделии.

### 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`
  - Класс заявки: `Product`
  - Максимальная вместимость Установите флажок
4. Выделите объект `delay` и установите его свойства:
  - Класс заявки: `Product`
  - Задержка задается Явно
  - Время задержки `exponential( 1/timePremIzd )`

- Вместимость kolPunPriem
- Действие при выходе

```

kolPriemIzd++;
costPriemkilzd += stPriemIzd;
entity.cost += stPriemIzd;

```

5. Выделите объект `selectOutput` и установите его свойства:

- Класс заявки: `Product`
- Выход `true` выбирается с заданной вероятностью
- Вероятность[0..1] 1-procBrakPriem
- Действие при выходе (`false`)

```

entity.sign1 = 2;
brakPriemIzd++;

```

Код свойства Действие при выходе объекта `delay` введен для счета количества `kolPriemIzd` принятых всего изделий.

Код свойства Действие при выходе (`false`) объекта `selectOutput` считает количество `brakPriemIzd` забракованных изделий и полю `entity.sign1` присваивает 2 - признак брака в изделии.

### Имитация склада готовых изделий

Для имитации склада готовых изделий используйте объект `sink` со следующими свойствами:

- Имя: ГотИзделия
- Класс заявки: `Product`
- Действие при выходе

```

kolGotIzd++;
timeIzd = time()/kolGotIzd;
costGotIzd += entity.cost;
minCostGotIzd+=minCostIzd;
koefIncrCostIzd =

```

```
(costGotIzd+costBrakBlock)/minCostGotIzd;
ostGotBlock1 = gotBlock1 - kolGotIzd - zamBlock1;
ostGotBlock2 = gotBlock2 - kolGotIzd - zamBlock2;
ostGotBlock3 = gotBlock3 - kolGotIzd - zamBlock3;
ostGotBlock4 = gotBlock4 - kolGotIzd - zamBlock4;
```

Код свойства *Действие при выходе* введен для счёта количества *kolGotIzd* готовых изделий, их стоимости *costGotIzd* 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 *ostGotBlock1*... *ostGotBlock4* на текущее модельное время.

### 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.signal* присваивались признаки 1 или 2.

1. Перетащите объекты *selectOutput* и *sink* на прямоугольник с именем Склад бракованных блоков диаграммы класса *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 *Действие 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
	<b>selectOutput</b>
Класс заявки:	Product
Выход true выбирается При выполнении условия	
Условие	entity.signal == 1
	<b>sink</b>
Имя	БрПостКонтр
Класс заявки:	Product
	if (entity.numBlBrak1 == 1) {costBrakBlock += costBlock1; allBrakBlock1++;}  if (entity.numBlBrak2 == 1) {costBrakBlock += costBlock2 ; allBrakBlock2++;}  if (entity.numBlBrak3 == 1) {costBrakBlock += costBlock3; allBrakBlock3++;}  if (entity.numBlBrak4 == 1) {costBrakBlock += costBlock4; allBrakBlock4++;}
Действие при выходе	
Имя	БрБлЗам
Класс заявки:	Product
	if (entity.numBlBrak1 == 1) {costBrakBlock += costBlock1; allBrakBlock1++;}  if (entity.numBlBrak2 == 1)

## Действие при выходе

```

costBrakBlock += costBlock2;
allBrakBlock2++;
if (entity.numBlBrak3 == 1)
{costBrakBlock += costBlock3;
allBrakBlock3++;}
if (entity.numBlBrak4 == 1)
{costBrakBlock += costBlock4;
allBrakBlock4++;}

```

Замечание. Можно обойтись и одним объектом `sink` для имитации склада бракованных блоков. Для этого нужно оба кода (см. табл. 6.8) занести в его свойство Действие при выходе.

В начало первого кода добавить:

```

if (entity.sign1 == 1)
{ код для БрПостКонтр }
а во второго
if (entity.sign1 == 2)
{ код для БрБлЗам }

```

Количественный результат будет аналогичным, только во время прогонов модели вы не сможете визуально наблюдать раздельные потоки забракованных блоков.

## Организация переключения 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](#).

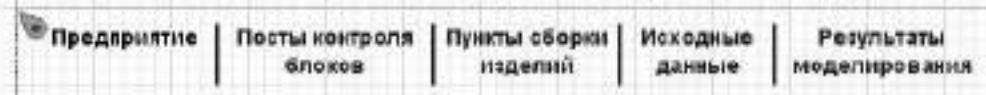


Рис. 6.13. Элементы презентации для переключения

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 странице Основные панели Свойства установите в поле Цвет: black, а также выберите выравнивание текста и шрифт.
2. Перетащите второй элемент `text`, разместите и введите в поле Текст: Посты контроля блоков. На странице Основные панели Свойства установите в поле Цвет: blue, а также выберите выравнивание текста и шрифт. На панели Свойства выделите Динамические и в поле Действие по щелчку: введите код: `test.Kontrl.navigateTo();`  
Замечание. Можете выбрать и другие цвета элементов презентации. При этом нужно исходить из того, чтобы различались по цвету текст открытой области просмотра и тексты закрытых областей просмотра.
3. Перетащите третий элемент `text`, разместите и введите в поле Текст: Пункты сборки изделий. На панели Свойства выделите Динамические и в поле Действие по щелчку: введите: `sborka.Sbor.navigateTo();`
4. Перетащите четвертый элемент `text`, разместите и введите в поле Текст: Исходные данные. На панели Свойства выделите Динамические и в поле Действие по щелчку: введите: `Data.navigateTo();`
5. Перетащите пятый элемент `text`, разместите и введите в поле Текст: Результаты моделирования. На панели Свойства выделите Динамические и в поле Действие по щелчку: введите: `Result.navigateTo();`

Таким образом, с области просмотра Mainview можно будет переходить на любую из четырех областей просмотра. При этом элемент

презентации открытой области высвечивается черным цветом, а остальные элементы презентации - голубым цветом.

- Скопируйте элементы презентации так, как они показаны на [рис. 6.13](#).
- Последовательно переходите на остальные области просмотра, вставьте скопированные элементы презентации и внесите правки в их свойства согласно [табл. 6.9](#).

Построение модели для проведения простого эксперимента завершено. Теперь отладьте модель и проведите эксперимент.

Замечание. При каждом запуске модели результаты будут повторяться, если в модели 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.

Таблица 6.9.

Элемент презентации	Действие по щелчку
<b>Data</b>	
Предприятие	<code>Mainview.navigateTo();</code>
Посты контроля блоков	<code>test.Kontrl.navigateTo();</code>
Пункты сборки изделий	<code>sborka.Sbor.navigateTo();</code>
Исходные данные	
Результаты моделирования	<code>Result.navigateTo();</code>
<b>Result</b>	
Предприятие	<code>Mainview.navigateTo();</code>
Посты контроля блоков	<code>test.Kontrl.navigateTo();</code>
Пункты сборки изделий	<code>sborka.Sbor.navigateTo();</code>

Исходные данные	Data.navigateTo();
-----------------	--------------------

Результаты моделирования	
-----------------------------	--

### Kontrol

Предприятие	get_Main().Mainview.navigateTo();
Посты контроля блоков	
Пункты сборки изделий	get_Main().sborka.Sbor.navigateTo();
Исходные данные	get_Main().Data.navigateTo();
Результаты моделирования	get_Main().Result.navigateTo();

### Sbor

Предприятие	get_Main().Mainview.navigateTo();
Посты контроля блоков	get_Main().test.Kontrl.navigateTo();
Пункты сборки изделий	
Исходные данные	get_Main().Data.navigateTo();
Результаты моделирования	get_Main().Result.navigateTo();

## Интерпретация результатов моделирования

Мы провели эксперименты с моделями, выполнив в GPSS World 1 000 прогонов, а в AnyLogic увеличив модельное время в 1 000 раз. При этом начальное значение генераторов случайных чисел установили 339.

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 табл. 6.10 приняты обозначения: ПК1...ПК4 - пункты контроля блоков 1...4 соответственно; ПС - пункты сборки изделий; СВК - стеллы выходного

контроля изделий; ПП - пункт приёмки изделий.

Сравнительная оценка свидетельствует о том, что результаты моделирования, полученные в различных средах, адекватны.

Различие незначительно, например, для коэффициента увеличения стоимости изделия оно составляет 0,003...0,004.

Коэффициенты использования различных подразделений предприятия также отличаются на тысячные доли.

Таблица 6.10. Показатели функционирования предприятия

Показатели	Системы моделирования	
	GPSS World	AnyLogic
Согласно постановке задачи		
Количество готовых изделий	121,628	122,337
	$\Delta_{11} =  121,628 - 122,337  = 0,700$	
Стоимость готовых изделий	74410	75151
	$\Delta_{12} =  70787 - 71200  = 413$	
Минимальная стоимость изделий	70787	71200
	$\Delta_{13} =  1,095 - 1,099  = 0,004$	
Стоимость забракованных блоков	3099	3122
Коэффициент увеличения стоимости	1,095	1,099
	$\Delta_{14} =  0,315 - 0,316  = 0,001$	
Время изготовления изделия, мин	19,732	19,618
Коэффициент использования ПК1	0,315	0,316
Коэффициент использования ПК2	0,726	0,728
Коэффициент использования ПК3	0,696	0,699
Коэффициент использования		

**ПК4**

Коэффициент использования ПС	0,581	0,586
Коэффициент использования СВК	0,545	0,548
Коэффициент использования ПП	0,539	0,539
aveTimeShop1=15, aveTimeShop2=10, aveTimeShop3=10, aveTimeShop4=15		
Количество готовых изделий	147,909	147,954
$\Delta_{21} =  147,909 - 147,954  = 0,045$		
Стоймость готовых изделий	90487	90902
Минимальная стоймость изделий	86083	86109
$\Delta_{22} =  86083 - 86109  = 26$		
Стоймость забракованных блоков	3848	3824
Коэффициент увеличения стоймости	1,096	1,1
$\Delta_{23} =  1,096 - 1,100  = 0,004$		
Время изготавления изделия, мин	16,226	16,221
Коэффициент использования ПК1	0,398	0,4
Коэффициент использования ПК2	0,8	0,804
Коэффициент использования ПК3	1	1
Коэффициент использования ПК4	0,566	0,563
Коэффициент использования ПС	0,705	0,705
Коэффициент использования СВК	0,66	0,662

**СВК**

Коэффициент использования ПП	0,654	0,654
aveTimeShop4=10		
Количество готовых изделий	153,743	154,327
Стоймость готовых изделий	$\Delta_{31} =   153,743 - 154,327   = 0,584$ 94050	94815
Минимальная стоимость изделий	89478	89818
Стоймость забракованных блоков	$\Delta_{32} =   89478 - 89818   = 340$ 4412	4450
Коэффициент увеличения стоимости	1,1	1,105
Время изготовления изделия, мин	$\Delta_{33} =   1,100 - 1,105   = 0,005$ 15,61	15,551
Коэффициент использования ПК1	0,398	0,399
Коэффициент использования ПК2	0,8	0,797
Коэффициент использования ПК3	1	1
Коэффициент использования ПК4	0,85	0,852
Коэффициент использования ПС	0,733	0,737
Коэффициент использования СВК	0,691	0,691
Коэффициент использования ПП	0,678	0,681
aveTimeShop1=10, postKontr3=3		
Количество готовых изделий	209,234	209,086

	$\Delta_{41} =  209,234 - 209,086  = 0,148$
Стоимость готовых изделий	127880
Минимальная стоимость изделий	121774
	$\Delta_{42} =  121744 - 121688  = 56$
Стоимость забракованных блоков	4952
Коэффициент увеличения стоимости	1,091
	$\Delta_{43} =  1,091 - 1,094  = 0,003$
Время изготовления изделия, мин	11,47
Коэффициент использования ПК1	0,6
Коэффициент использования ПК2	0,799
Коэффициент использования ПК3	0,701
Коэффициент использования ПК4	0,845
Коэффициент использования ПС	1
Коэффициент использования СВК	0,936
Коэффициент использования ПП	0,923

# Модель функционирования терминала

## Модель в GPSS World

### Постановка задачи

Общий вид терминала показан на [рис. 7.1](#). Территория терминала обозначена А, примыкающая городская территория - В.

1. Автомобиль (транспортное средство), груженный или порожний, попадает впорт по дороге общего пользования С. 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 Е. 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.

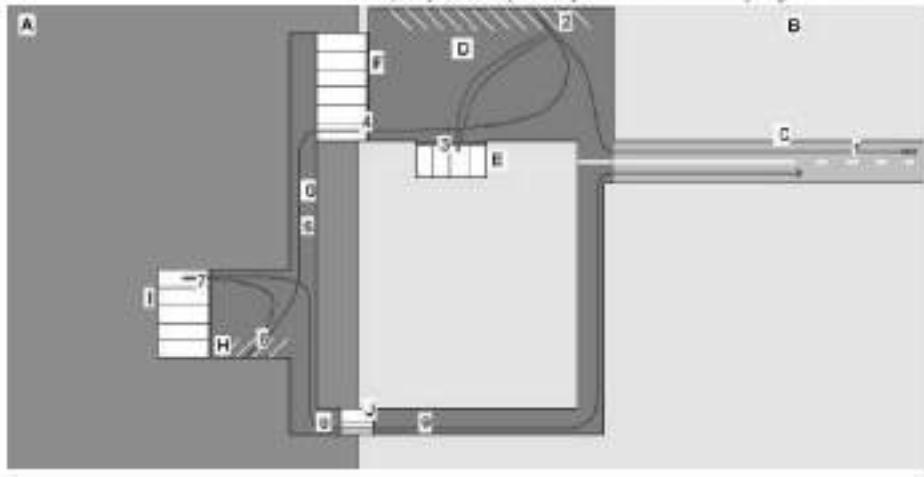


Рис. 7.1. Схема терминала

5. Автомобиль следует на оперативную парковку И, 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 И 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 И, 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.

## Программа модели в GPSS World

В модели автомобили следует представить транзактами. Все остальные элементы терминала (парковку D, офис Е, 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;
- задание времени моделирования и вычисление результатов моделирования.

Как видно, в данном случае можно обойтись без разделения событийной части модели на сегменты.

Ниже приводится программа модели.

```

; Модель функционирования терминала
; Многоканальные устройства
D_ Storage 10 ;Имитирующее парковку D
E_ Storage 5 ;Имитирующее офис E
F_ Storage 5 ;Имитирующее полосы ворот F
I_ Storage 7 ;Имитирующее места у зоны I
ZP_ Storage 2 ;Имитирующее места в зоне Z
J_ Storage 7 ;Имитирующие ворота J
; Исходные данные
timeA EQU 9 ;Среднее время приезда транспорта
timeD EQU 2 ;Среднее время парковки на D
timeE EQU 10 ;Среднее время оформления документов в офисе E
timeF EQU 2 ;Среднее время досмотра на воротах F
timeFH EQU 2 ;Среднее время движения от F к H
timeZ EQU 10 ;Среднее время выполнения заказа на обслуживание авт
timeI EQU 5 ;Среднее время обслуживания автомобиля в зоне I
timeJ EQU 2 ;Среднее время движения от I к J
timeJ EQU 2 ;Среднее время досмотра на воротах J
timeMod EQU 480 ;Время моделирования
; Статистические таблицы
C_ QTABLE C_,1,1,50
E_ QTABLE E_,1,1,30
ZP_ QTABLE ZP_,1,1,40
TMeanP TABLE M1,40,1,80
; Событийная часть модели
GENERATE
(Exponential(371,0,timeA));Имитация прибытия автотранспорта
KoPrib QUEUE C_ ;Занять очередь на парковку D
ENTER D_ ;Занять одно место на парковкeD
DEPART C_ ;Покинуть очередь на парковку
ADVANCE (Exponential(83,0,timeD));Имитация парковки на D
QUEUE E_ ;Встать в очередь в офис E
ENTER E_ ;Занять окошко в офисе E
DEPART E_ ;Покинуть очередь в офис E
ADVANCE (Exponential(113,0,timeE))
;Имитация оформления документов в офисе E
LEAVE E_ ;Покинуть офис E

```

SPLIT 1,1 ;Расщепление для отправки заявки  
TEST E P1,1,Me1 ;Автотранспорт к воротам F, заявка в Z  
QUEUE ZP\_ ;Занять очередь в зону Z  
ENTER ZP\_ ;Занять место обслуживания заявки в зоне Z  
DEPART ZP\_ ;Покинуть очередь в зону Z  
ADVANCE (Exponential(213,0,timeZ));Имитация выполнения заявки в з  
LEAVE ZP\_ ;Освободить место обслуживания заявки в зоне Z  
TRANSFER ,Me2  
Met1 QUEUE F\_ ;Встать в очередь к воротам F  
ENTER F\_ ;Занять полосу у ворот F  
DEPART F\_ ;Покинуть очередь к воротам F  
LEAVE D\_ ;Освободить место на парковке  
ADVANCE (Exponential(183,0,timeF));Имитация досмотра у ворот F  
LEAVE F\_ ;Освободить полосу у ворот F  
ADVANCE (Exponential(163,0,timeFH));Имитация движения от F к H  
Met2 ASSEMBLE \*1 ;Фиксация - выполнение заявки  
ENTER I\_ ;Занять ячейку I  
ADVANCE (Exponential(315,0,timeI))  
;Имитация обслуживания в зоне I  
LEAVE I\_ ;Освободить ячейку I  
ADVANCE (Exponential(511,0,timeII));Имитация движения от I к J  
ENTER J\_ ;Занять место на воротах J  
ADVANCE (Exponential(703,0,timeJ))  
;Имитация осмотра на воротах J  
LEAVE J\_ ;Освободить место на воротах J  
KolObr SAVEVALUE timeSum+,M1  
TABULATE TMeanP  
TERMINATE  
;  
; Задание времени моделирования и расчет результатов  
GENERATE TimeMod  
TEST L X\$Prog,TG1,Me3  
SAVEVALUE Prog,TG1  
Me3 TEST E TG1,1,Me4  
SAVEVALUE TimeObr,(X\$TimeSum/NSKolObr)  
; Среднее время обработки транспорта  
SAVEVALUE KoefIsp,(N\$KolObr/N\$KolPrib)  
; Коеффициент использования терминала  
SAVEVALUE KolObrCar,(N\$KolObr/X\$Prog)

; Количество обработанных транспортов

Met4 TERMINATE 1

START 10000

## Модель функционирования терминала в AnyLogic

### Исходные данные и результаты моделирования

1. Для ввода исходных данных используем Параметр и Бегунок. 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. Щелкните кнопку Далее.
3. На открывшейся второй странице Мастера создания модели выберите Начать создание модели "с нуля". Щелкните кнопку Далее.

Создадим две области просмотра. Первую для ввода исходных данных и вывода результатов моделирования, вторую - для размещения элементов модели.

1. Создайте вторую область просмотра для размещения элементов модели на диаграмме класса Main. Из палитры Презентация перетащите в любое место элемент Область просмотра.
2. Перейдите на страницу Основные панели Свойства. В поле Имя: введите Модель\_Терминал.
3. Задайте, как будет располагаться область просмотра относительно ее якоря, с помощью элемента управления Выравнивать по: Верхн. левому углу.
4. Выберите режим масштабирования из выпадающего списка Масштабирование: Подогнать под окно.
5. Перейдите на страницу Дополнительные панели Свойства. Введите в поля X: 0, Y: 0, Ширина: 680, Высота: 370.
6. Из палитры Презентация перетащите элемент Прямоугольник. Оставьте имя, предложенное системой.

7. Перейдите на страницу Дополнительные панели Свойства. Введите в поля X: 20, Y: 20, Ширина: 640, Высота: 340.
8. Из палитры Презентация перетащите второй элемент Область просмотра.
9. Перейдите на страницу Основные панели Свойства. В поле Имя: введите Данные.
10. Задайте, как будет располагаться область просмотра относительно ее якоря, с помощью элемента управления Выравнивать по: Верхн., левому углу.
11. Выберите режим масштабирования из выпадающего списка Масштабирование: Подогнать под окно.
12. Перейдите на страницу Дополнительные панели Свойства. Введите в поля X: 0, Y: 1000, Ширина: 560, Высота: 400.
13. Из палитры Презентация перетащите элемент Скруглённый прямоугольник. Оставьте имя, предложенное системой.
14. Перейдите на страницу Дополнительные панели Свойства. Введите в поля X: 20, Y: 1030, Ширина: 520, Высота: 350.
15. Перетащите элемент text и на странице Основные панели Свойства в поле Текст: введите Исходные данные и результаты моделирования.
16. В Палитре выделите Основная. Перетащите элементы Параметр на элемент с именем Исходные данные и результаты моделирования. Разместите их и дайте имена так, как показано на [рис. 7.2](#).
17. Значения свойств установите согласно [табл. 7.1](#). Имена параметров (в том числе и знак подчёркивания) приняты те же, что и в GPSS-модели.
18. В Палитре выделите Элементы управления. Перетащите элементы Бегунок. Разместите их так, как показано на [рис. 7.2](#). Значения свойств установите согласно [табл. 7.1](#).

Таблица 7.1.

Параметр			Бегунок		
Имя	Тип	Значение по умолчанию	Связать с	Минимальное значение	Максимальное значение
D_	int	10	D_	1	50

E_	int	5	E_	1	50
F_	int	5	F_	1	50
I_	int	7	I_	1	50
ZP_	int	2	ZP_	1	50
J_	int	7	J_	1	50
timeD	double	2	timeD	1	50
timeE	double	10	timeE	1	50
timeF	double	2	timeF	1	50
timeI	double	5	timeI	1	50
timeZ	double	10	timeZ	1	50
timeJ	double	2	timeJ	1	50
timeA	double	9	timeA	1	50
timeFH	double	2	timeFH	1	50
timeIJ	double	2	timeIJ	1	50

19. Для вывода результатов моделирования используем элемент Простая переменная. В Палитре выделите Основная. Перетащите элементы Простая 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.

Переменные KolObrCar, TimeObr, KoefIsp имеют тот же смысл,





Рис. 7.2. Размещение элементов для ввода исходных данных и вывода результатов моделирования

Переменная `kolJ` введена для счета количества обработанных транспортов. Переменная `TimeSum` введена для счета суммарного времени обработки всех транспортов. По значениям этих переменных рассчитывается среднее время обработки `TimeObr` одного транспорта.

В последующем для определения количественных значений каждой из этих переменных мы напишем соответствующие Java-коды.

## Событийная часть модели

Объекты событийной части модели показаны на рис. 7.3.

Для построения использованы следующие двадцать объектов библиотеки Enterprise Library (в скобках после имени каждого объекта указано их количество в событийной части модели):

- source (один);
- queue (восемь);
- delay (восемь);
- split (один);
- match (один);
- sink (один).

Создадим событийную часть модели. Для неё мы уже создали область присмотра 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 класс заявки Саг с полями:

```
int id;  
double vход;
```

Дополнительное поле `id` нестандартного класса заявки Саг введено для проверки объектом `match` условия объединения двух заявок в одну.

Поле `vход` предназначено для записи времени входа заявки в модель. И в последующем для вычисления времени обработки заявки, хотя можно было бы обойтись и без этого поля, а использовать объект `TimeMeasureStart` вместе с `TimeMeasureEnd` для определения того же времени.

Перетащите остальные объекты, соедините так, как показано на [дис. 7.3](#). Последовательно выделяя эти объекты, установите их свойства согласно [табл. 7.3](#).

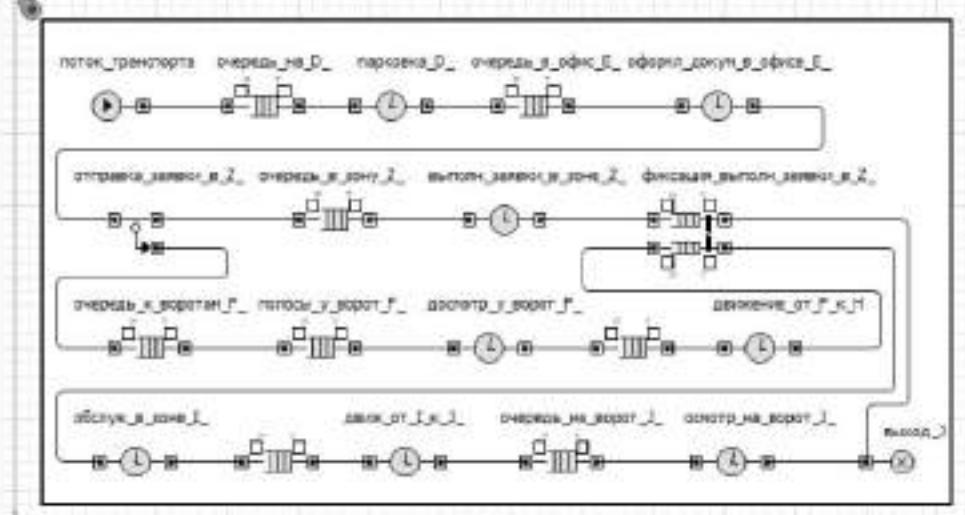


Рис. 7.3. Элементы модели Terminal

Таблица 7.2. Свойства объекта source

Имя	Свойства	Значения
поток_транспорта	Отображать имя	Установите флажок
	Класс заявки	Cat
	Заявки прибывают согласно	Времени между прибытиями
	Время между прибытиями	exponential(1/timeA)
	Количество заявок, прибывающих за один раз	1
	Новая заявка	new Cat()
	Действие при выходе	entity.id = поток_транспорта.count(); entity.vход = time();

Таблица 7.3. Свойства объектов событийной части модели

Свойства	Значение
Имя	очередь_на_D_
Отображать имя	Установить флагок
Класс заявки	Car
Максимальная вместимость	Установить флагок
Включить сбор статистики	Установить флагок
Имя	парковка_D_
Отображать имя	Установить флагок
Класс заявки	Car
Задержка задается	Явно
Время задержки	<code>exponential(1/timeD)</code>
Вместимость	D_
Включить сбор статистики	Установить флагок
Имя	очередь_в_офис_E_
Отображать имя	Установить флагок
Класс заявки	Car
Вместимость	10
Действие при подходе к выходу	<pre>if (очередь_E &lt; очередь_в_офис_E_.size())     очередь_в_офис_E_ = очередь_в_офис_E_.size();</pre>
Включить сбор статистики	Установить флагок
Имя	оформл_докум_в_офисе_E_
Отображать имя	Установить флагок
Класс заявки	Car
Задержка задается	Явно
Время задержки	<code>exponential(1/timeE)</code>
Вместимость	E_

Имя	отправка_заявки_в_Z
Отображать имя	Установить флагок
Класс заявки	Car
Количество копий	1
Новая заявка (копия)	new Car
Действие при выходе копии	entity.id = original.id; entity.vxod = original.vxod;
Имя	очередь_в_зону_Z_
Отображать имя	Установить флагок
Класс заявки	Car
Вместимость	100
Действие при подходе к выходу	if (очередь_Z < очередь_в_зону_Z_.size()) очередь_Z = очередь_в_зону_Z_.size();
Включить сбор статистики	Установить флагок
Имя	выполн_заявки_в_зоне_Z_
Отображать имя	Установить флагок
Класс заявки	Car
Задержка задается	Явно
Время задержки	exponential(1/time2)
Вместимость	2P_
Включить сбор статистики	Установить флагок
Имя	фиксация_выполн_заявки_в_Z_
Отображать имя	Установить флагок
Класс заявки	Car
Условие соответствия	entity1.id == entity2.id
Вместимость	1
Вместимость	100

Вместимость 2	100
Действие при выходе 1	<pre>entity.id = 0;</pre>
Действие при выходе 2	<pre>if (очередь_I &lt; фиксация_выполн_заявки_в_Z_.size2()) очередь_I = фиксация_выполн_заявки_в_Z_.size2();</pre>
Время задержки	<code>exponential(1/timeIJ)</code>
Вместимость	10
Включить сбор статистики	Установить флагок
Имя	<code>очередь_на_ворот_J_</code>
Отображать имя	Установить флагок
Класс заявки	Саг
Вместимость	10
Действие при подходе к выходу	<pre>if (очередь_J &lt; очередь_на_ворот_J_.size() ) очередь_J = очередь_на_ворот_J_.size();</pre>
Включить сбор статистики	Установить флагок
Имя	<code>осмотр_на_ворот_J_</code>
Отображать имя	Установить флагок
Класс заявки	Саг
Задержка задается	Явно
Время задержки	<code>exponential(1/timeJ)</code>
Вместимость	<code>J_</code>

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_Э
Отображать имя	Установить флажок
Класс заявки	<code>Car</code>
	<pre>if (entity.id != 0) {     kolJ++;     KolObrCar = kolJ/10000;     KoefIsp = kolJ/поток_транспорта.count();     TimeSum += (time() - entity.vxod);     TimeObr = TimeSum / kolJ;</pre>
Действие при входе	<pre>KoefIsp_E = оформл_докум_в_офисе_E_.statsUtilization; KoefIsp_F = досмотр_у_ворот_F_.statsUtilization.mean( KoefIsp_Z = выполн_заявки_в_зоне_2_.statsUtilization.</pre>

```

КоefIsp_I =
обслуж_в_зоне_I_.statsUtilization.mean();
КоefIsp_J =
осмотр_на_ворот_J_.statsUtilization.mean()

```

Остановимся на коде свойства Действие 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 приема на выходе 1 объекта `matchentity.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.

## Результаты моделирования

Результаты проведенных нами на построенных моделях 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 AnyLogic-модели установили число 1876. В GPSS World выполнили 10 000 прогонов, а в AnyLogic увеличили время моделирования в 10 000 раз, то есть моделировали в течение 4 800 000 мин.

Таблица 7.5.

Показатели	Системы моделирования	
	GPSS World	AnyLogic
KolObrCar	53,31	53,41
TimeObr	35,485	36,109

KoefIsp	1	1
KoefIsp_E	0,222	0,222
KoefIsp_F	0,044	0,044
KoefIsp_Z	0,556	0,556
KoefIsp_I	0,079	0,08
KoefIsp_J	0,032	0,032
очередь_E	5	5
очередь_F	1	1
очередь_Z	21	20
очередь_I	22	22
очередь_J	1	1
Машинное время 25 мин 34 с	25 с	

Как видно, результаты моделирования идентичны. Кроме машинного времени, которое в GPSS World составляет 25 мин 34 с, то есть примерно в 65 раз больше, чем в AnyLogic. 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.

## 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.

## Первый отсеивающий эксперимент в GPSS World

Сущность этого эксперимента (мы назвали его первым потому, что проведём два отсеивающих эксперимента) состоит в проведении многофакторного дисперсионного анализа с целью выявления степени

влияния различных факторов и их комбинаций (взаимодействий) на значение целевой функции (функции отклика, представленной в виде уравнения регрессии).

В условиях модели функционирования терминала в первом эксперименте требуется исследовать зависимость времени обработки одного автомобиля от шести факторов (это максимальные возможности GPSS World), например, при следующих их минимальных и максимальных значениях (табл. 7.6):

Таблица 7.6.

Уровни факторов	Факторы					
	timeA	timeE	timeF	timeI	timeZ	timeFH
Нижний	10	10	2	5	10	2
Верхний	20	15	4	10	15	4

1. Запустите GPSS World. Откройте модель Terminal. Выберите Edit / Insert Experiment / Screening ... (Правка / Вставить эксперимент / Отсеивающий ...). Откроется диалоговое окно Screening Experiment Generator (Генератор отсеивающего эксперимента). Приступите к заполнению полей диалогового окна.
2. В поля Experiment Name (Имя эксперимента) и Run Procedure Name (Имя процедуры запуска) введите, например, Terminal и RunTerminal соответственно (рис. 7.4). Имена эксперименту и процедуре запуска эксперимента дает пользователь.
3. Введите ранее выбранные согласно стратегическому планированию эксперимента факторы (табл. 7.6), начиная с фактора А. В поле Name (User Variable) (Имя (Переменная пользователя)) введите имя фактора, в поля Value1 и Value2 - его нижний и верхний уровни соответственно. Запомните соответствие между факторами и именами переменных.
4. Ниже идет группа Fraction (Часть полного эксперимента). Установите Full, что соответствует полному

факторному эксперименту. Справа под Run Count появится число 64, так как  $2^6 = 64$ . Это число наблюдений, которое нужно сделать в эксперименте. Количество прогонов в каждом наблюдении определим позже при тактическом планировании эксперимента.

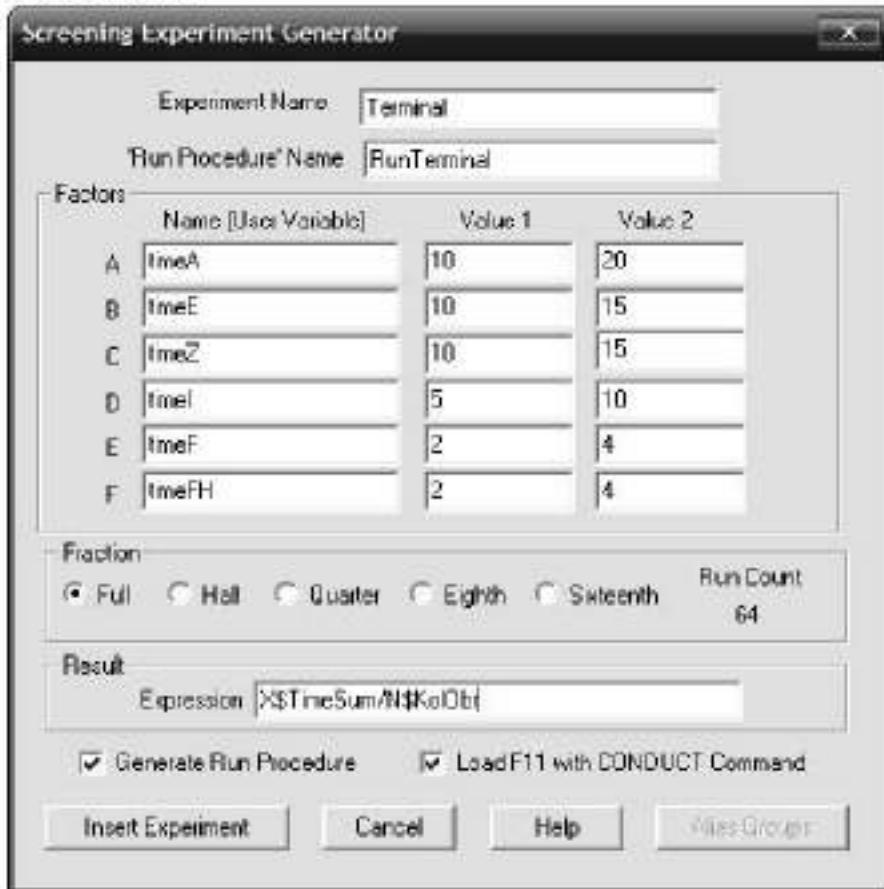


Рис. 7.4. Диалоговое окно (заполненное) Screening ExperimentGenerator (Генератор отсеивающего эксперимента)

5. В поле Expression (Выражение) группы Result (Результат) введите выражение, по которому вычисляется время обработки одного автомобиля: X\$TimeSum/N\$KolObr.
6. После группы Result (Результат) расположены два флажка, позволяющие выбирать опции. При выборе опции Generate Run Procedure вместе с экспериментом

создается стандартная процедура запуска, которую пользователь может корректировать согласно своим требованиям. Выбор второй опции Load F11 with CONDUCT Command закрепляет команду CONDUCT за функциональной клавишей F11. Тогда после создания объекта "Процесс моделирования" для запуска эксперимента нужно только нажать функциональную клавишу F11. Выберите обе опции.

7. Создайте Plus - операторы и вставьте их в нижнюю часть модели Terminal. Для этого нажмите кнопку Insert Experiment (Вставить эксперимент), расположенную в левой нижней части диалогового окна Screening Experiment Generator (Генератор отсеивающего эксперимента) (см. [рис. 7.4](#)). Так как была выбрана опция Generate Run Procedure, то создана стандартная процедура запуска.

Появится её диалоговое окно, дающее возможность пользователю изменить процедуру запуска согласно своим требованиям.

8. Перейдите, пользуясь клавишами вверх-вниз, в конец процедуры запуска. Там в разделе Set up your own run conditions (Задайте свои условия наблюдения) имеются две команды START, между которыми находится команда RESET ([рис. 7.5](#)). Поясним назначение этих команд.
9. Первой командой START

```
DoCommand("START 100,NP"); /*Get past the Startup Period. */
```

определяется количество прогонов в неустоявшемся режиме. Команда RESET сбрасывает в ноль накопленную в этом режиме статистику без удаления транзактов из процесса моделирования.

10. Второй командой START

```
DoCommand("START 1000,NP"); /*Run the Simulation. */
```

определяется количество прогонов в одном наблюдении. В обеих командах используется оператор NP - стандартный отчёт не выводится. На [рис. 7.5](#) показаны условия стандартной процедуры

запуска после корректировки. Обе команды START мы оставили без изменения, а вот после команды RESET вставили строку

```
DoCommand("SAVEVALUE TimeSum,0"); /*Run the Simulation. */
```

которая обнуляет ячейку TimeSum.

В этой ячейке накапливается суммарное время обработки всех автомобилей в одном наблюдении эксперимента. Содержимое сохраняемых ячеек при переходе к следующему наблюдению системой (командой RESET) не обнуляется. Поэтому и введена эта команда, в противном случае результаты моделирования будут неверными.

11. После корректировки процедуры запуска нажмите Ok. Сгенерированный код Plus - эксперимента появится ниже программы GPSS-модели.

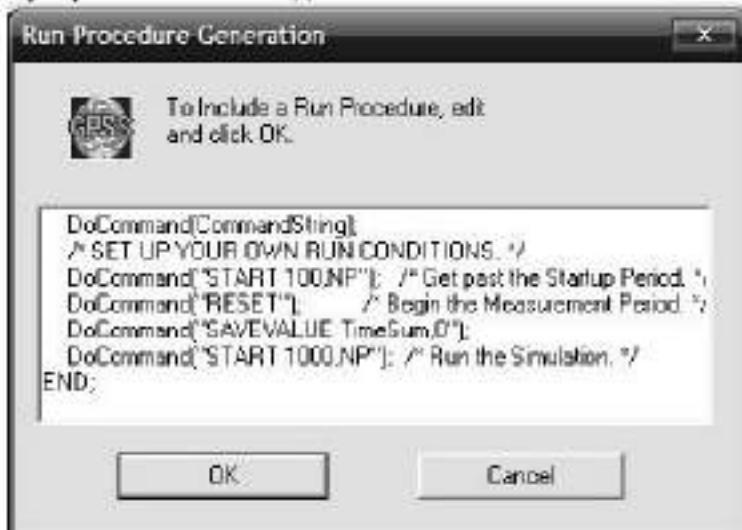


Рис. 7.5. Условия стандартной процедуры запуска после корректировки

12. Проведите трансляцию, т. е. создайте объект "Процесс моделирования", для чего выполните команду Command / Create Simulation (Команда / Создать процесс моделирования).

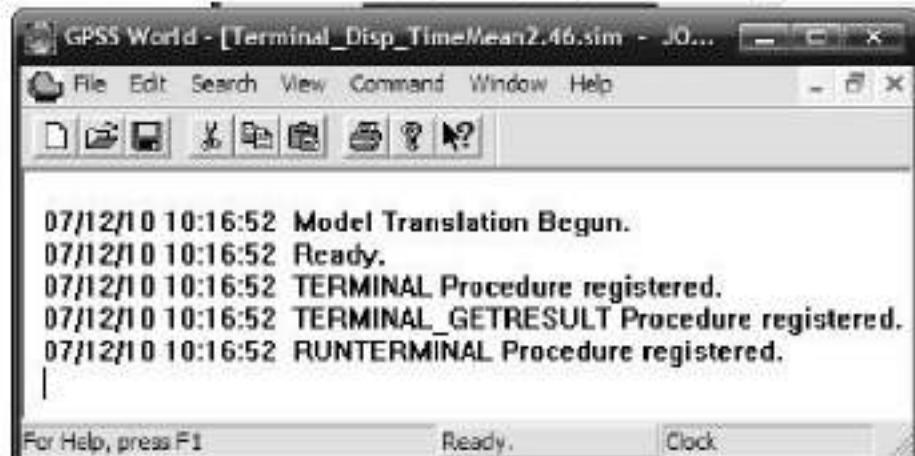


Рис. 7.6. Окно Journal (Журнал) с сообщением об успешном создании объекта "Процесс моделирования"

13. При отсутствии ошибок в сгенерированном эксперименте в окне Journal (Журнал) появится сообщение (см. рис. 7.6).
14. Запустите эксперимент. Для вызова эксперимента предназначена команда CONDUCT. Ранее за функциональной клавишей [F11] была закреплена команда CONDUCT (Edit/Settings/Function Keys (Правка/Настройки /Функциональные клавиши)) выбором соответствующей опции (см. рис. 7.4). Нажмите функциональную клавишу [F11]. Эксперимент начинает работать.
15. В ходе выполнения эксперимента автоматически создается отчет, который по готовности записывается в окно Journal (Журнал) объекта "Процесс моделирования". В отчете содержатся Yield - целевая функция, значения её и значения факторов, при которых получены эти значения целевой функции.
16. Фрагмент отчёта для четырех наблюдений (Run33 ... Run36) показан на рис. 7.7. К этим данным и условиям выбора именно этого фрагмента отчёта мы обратимся позднее.
17. В результате дисперсионного анализа определяется также среднее время обработки одного автомобиля при 64 комбинациях выбранных значений факторов (см. табл. 7.8). В данном случае это время составляет 44,962 мин, то есть около 45 мин.

Terminal\_Dispterminal2:47:dm - JOURNAL

```

02/13/10 11:08:36 "Run 33. Yield=31.84369303784666. timeA=20; timeE=10; timeZ=10; timel=5;
timeF=2; timeFH=2"
02/13/10 11:08:36 Simulation in Progress.
02/13/10 11:08:36 A Simulation in an Experiment has ended. Clock is 48000.000000.
02/13/10 11:08:36 Simulation in Progress.
02/13/10 11:08:36 A Simulation in an Experiment has ended. Clock is 52000.000000.
02/13/10 11:08:39 "Run 34. Yield=31.6477627731682. timeA=20; timeE=10; timeZ=10; timel=5;
timeF=2; timeFH=4"
02/13/10 11:08:39 Simulation in Progress.
02/13/10 11:08:39 A Simulation in an Experiment has ended. Clock is 48000.000000.
02/13/10 11:08:39 Simulation in Progress.
02/13/10 11:08:41 "Run 35. Yield=31.71599631249422. timeA=20; timeE=10; timeZ=10; timel=5;
timeF=4; timeFH=2"
02/13/10 11:08:42 Simulation in Progress.
02/13/10 11:08:42 A Simulation in an Experiment has ended. Clock is 48000.000000.
02/13/10 11:08:42 Simulation in Progress.
02/13/10 11:08:44 A Simulation in an Experiment has ended. Clock is 52000.000000.
02/13/10 11:08:44 "Run 36. Yield=31.8046685858777. timeA=20; timeE=10; timeZ=10; timel=5;
timeF=4; timeFH=4"

```

For Help, press F1      Experiment ended.      Click

Рис. 7.7. Окно Journal (Журнал) с отчётом первого эксперимента

18. Сохраните эксперимент, указав имя DispTerminalTimeObr.

## Второй отсеивающий эксперимент в GPSS World

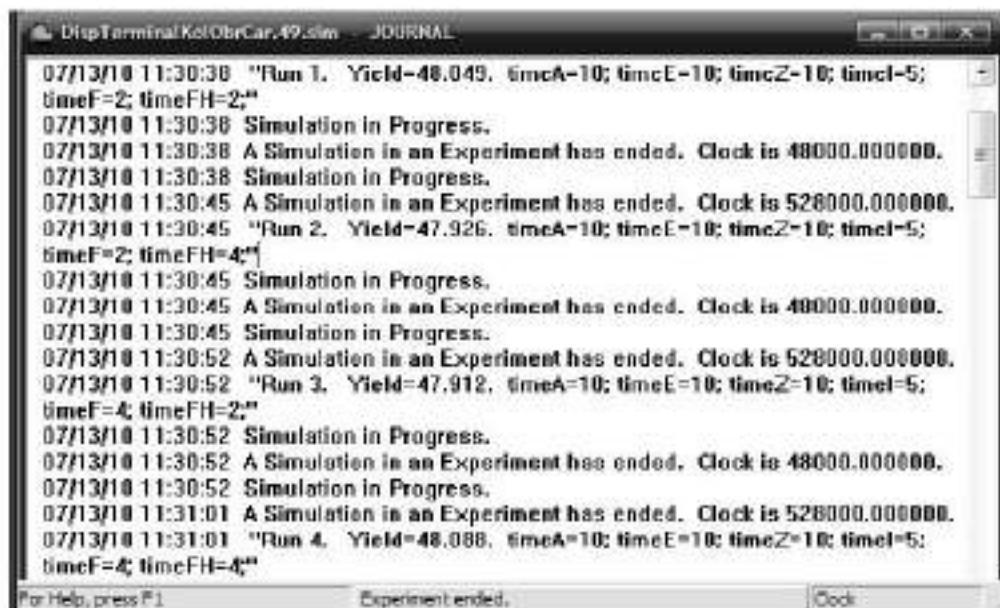
Во втором эксперименте требуется исследовать зависимость количества обработанных автомобилей за 8 часов в зависимости от тех же факторов и их значений, что и в первом эксперименте.

Согласно постановке на второй эксперимент поступите так.

1. Откройте эксперимент DispTerminalTimeObr.
2. Сохраните эксперимент с именем DispTerminalKolObrCar.
3. Удалите генерированный ранее код эксперимента Terminal, который расположен ниже программы модели.
4. Выполните Edit/Insert Experiment/Screening ... (Правка/Вставить эксперимент/Отсеивающий ...). Откроется диалоговое окно Screening Experiment Generator (Генератор отсеивающего эксперимента) с заполненными полями.
5. Оставьте все поля такими же, лишь замените в поле Expression (Выражение) группы Result

(Результат) выражение, по которому вычисляется время обработки одного автомобиля  $X\$TimeSum/N\$KolObr$ , выражением  $N\$KolObr/X\$Prog$  для расчета количества обработанных автомобилей.

6. Выполните пп. 6...16 предыдущего эксперимента.
7. Фрагмент отчёта для четырех наблюдений (Run1...Run4) показан на рис. 7.8. К этим данным мы также обратимся позже.



The screenshot shows the 'DispTerminal.KolObrCar.49.sim - JOURNAL' window. The log output is as follows:

```

07/13/18 11:30:38 "Run 1. Yield=48.049. timeA=10; timeE=10; timeZ=10; timeI=5;
timeF=2; timeFH=2;"  

07/13/18 11:30:38 Simulation in Progress.  

07/13/18 11:30:38 A Simulation in an Experiment has ended. Clock is 48000.000000.  

07/13/18 11:30:38 Simulation in Progress.  

07/13/18 11:30:45 A Simulation in an Experiment has ended. Clock is 528000.000000.  

07/13/18 11:30:45 "Run 2. Yield=47.926. timeA=10; timeE=10; timeZ=10; timeI=5;
timeF=2; timeFH=4;"  

07/13/18 11:30:45 Simulation in Progress.  

07/13/18 11:30:45 A Simulation in an Experiment has ended. Clock is 48000.000000.  

07/13/18 11:30:45 Simulation in Progress.  

07/13/18 11:30:52 A Simulation in an Experiment has ended. Clock is 528000.000000.  

07/13/18 11:30:52 "Run 3. Yield=47.912. timeA=10; timeE=10; timeZ=10; timeI=5;
timeF=4; timeFH=2;"  

07/13/18 11:30:52 Simulation in Progress.  

07/13/18 11:30:52 A Simulation in an Experiment has ended. Clock is 48000.000000.  

07/13/18 11:30:52 Simulation in Progress.  

07/13/18 11:31:01 A Simulation in an Experiment has ended. Clock is 528000.000000.  

07/13/18 11:31:01 "Run 4. Yield=48.088. timeA=10; timeE=10; timeZ=10; timeI=5;
timeF=4; timeFH=4;"  

For Help, press F1           Experiment ended.           Clock

```

Рис. 7.8. Окно Journal (Журнал) с отчетом второго эксперимента

## 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 отсевающем эксперименте.

1. Откройте в AnyLogic модель Terminal.
2. В панели Проект щелкните правой кнопкой мыши элемент модели Terminal 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, OptTerminal. Имя эксперимента должно начинаться с заглавной буквы - таково правило названия классов в Java.
5. В поле Корневой класс модели: выберите Main. Этим выбором вы задали корневой (главный) класс эксперимента. Объект этого класса 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 справа выпадающего списка.
7. 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.
8. Щелкните Готово. Появится страница Основные панели Свойства. Установите опцию минимизировать.
9. В поле Целевая функция введите root.TimeObj, так как корневой активный объект модели доступен здесь как root.
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 все параметры корневого объекта Main.
13. Чтобы разрешить варьирование параметров оптимизатором,

перейдите на строку с параметром `timeA`. Щелкните 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.

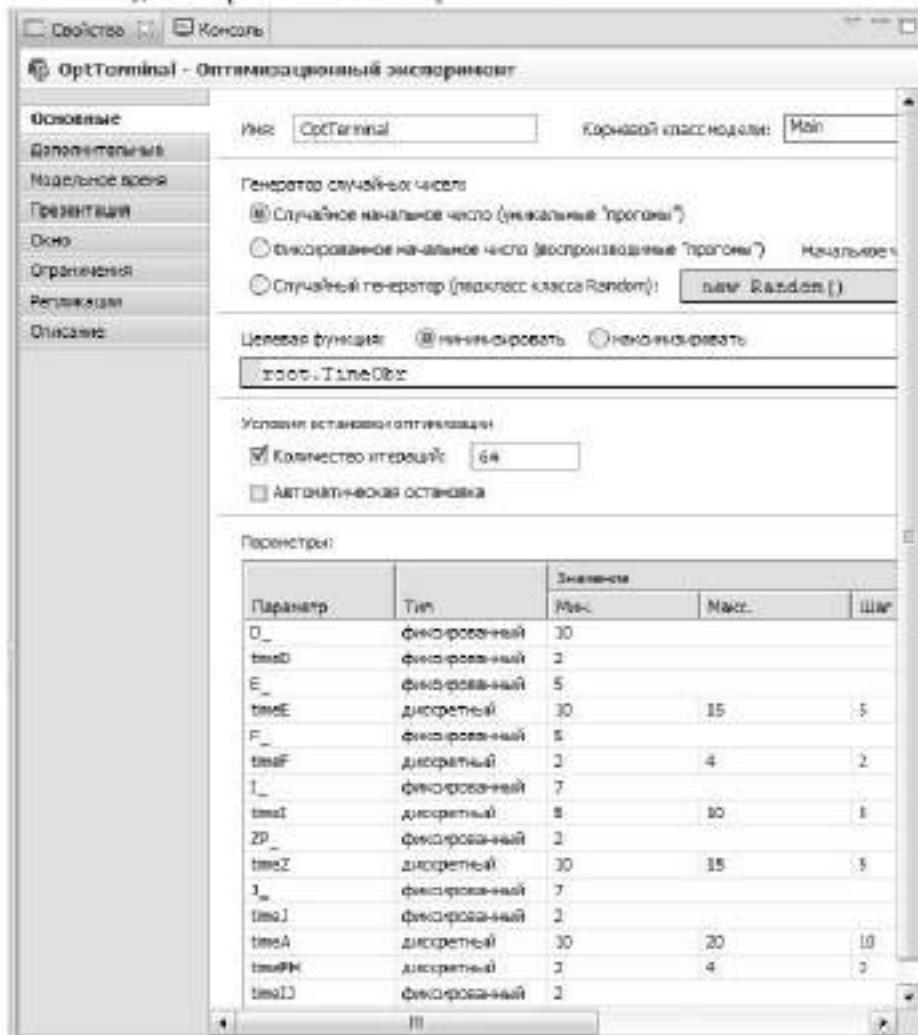


Рис. 7.9. Страница Основные панели Свойства эксперимента

14. Задайте диапазон допустимых значений параметра. Для чего введите в ячейку Мин минимальное значение 10, в ячейку Макс максимальное значение 20. Так как параметр дискретный, в ячейке

Шаг укажите величину шага 10.

15. Задайте так же остальные параметры, как на [рис. 7.9](#).
16. Перейдите на страницу Репликации панели Свойства.
17. Установите флажок Использовать репликации.
18. Число репликаций (прогонов) в одной итерации (наблюдении) может быть фиксированным или переменным. Фиксированное число репликаций, например, при доверительной вероятности  $\alpha = 0,95$ , точности  $\varepsilon = 0,1$  и стандартном отклонении  $\sigma = 0,1$  может быть определено по формуле:

$$N = t_{\alpha}^2 \frac{\sigma^2}{\varepsilon^2} = 1,96^2 \frac{0,1^2}{0,01^2} = 3,8416 \cdot 100 \approx 400$$

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

19. Выберите опцию Фиксированное количество репликаций и в соответствующем поле установите 400.
20. Вернитесь на страницу Основные и щелкните Создать интерфейс.

Кнопка находится в правом верхнем углу страницы Основные. После щелчка удаляется презентация эксперимента и создается 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.

21. В меню запуск выполните Terminal1/OptTerminal.
22. Щелкните Запустить оптимизацию. Начнет выполняться эксперимент. Во время эксперимента можно видеть на графике изменение значения целевой функции (вертикальная ось). После выполнения  $400*64=25\ 600$  прогонов ([рис. 7.10](#)) эксперимент остановится.

## Второй оптимизационный эксперимент в AnyLogic

Создайте второй эксперимент Оптимизация стохастических моделей в AnyLogic. Цель эксперимента: определение максимального

количество обработанных автомобилей в зависимости от тех же факторов и их значений, что и в первом эксперименте.

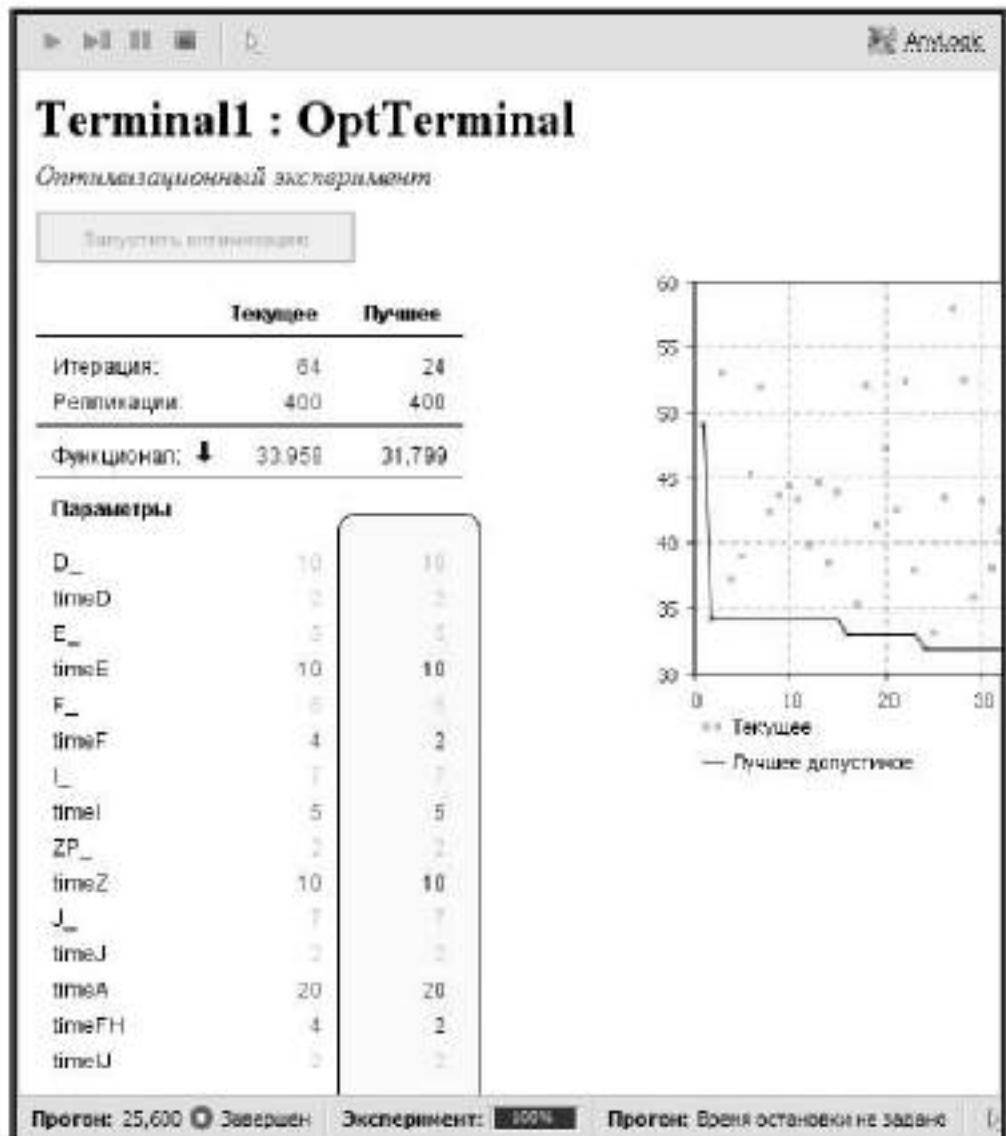


Рис. 7.10. Результаты первого оптимизационного эксперимента

1. Откройте в AnyLogic модель Terminal.
2. В панели Проект щелкните 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, OptTerminal2.
4. Выполните пп. 5...8 первого эксперимента.
5. Установите опцию максимизировать.
6. В поле Целевая функция введите root.Ko1ObrCar.
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. Вернитесь на страницу Основные и щелкните Создать интерфейс.
11. В меню запуск выполните Terminal1/OptTerminal2.
12. Щелкните 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: 48,08.

Перейдем к рассмотрению результатов экспериментов.

## Результаты экспериментов в GPSS World и AnyLogic

Результаты экспериментов с имитационными моделями функционирования терминала сведены в табл. 7.7 и 7.8.

Таблица 7.7.

Показатели, параметры	Системы моделирования	
	GPSS World	AnyLogic

**Первый эксперимент**

TimeObr	31,844	31,799
timeA	20	20
timeE	10	10
timeF	2	2
timeI	5	5
timeZ	10	10
timeFH	2	2
TimeObr	31,605	31,799
timeA	20	20
timeE	10	10
timeF	4	2
timeI	5	5
timeZ	10	10
timeFH	4	2

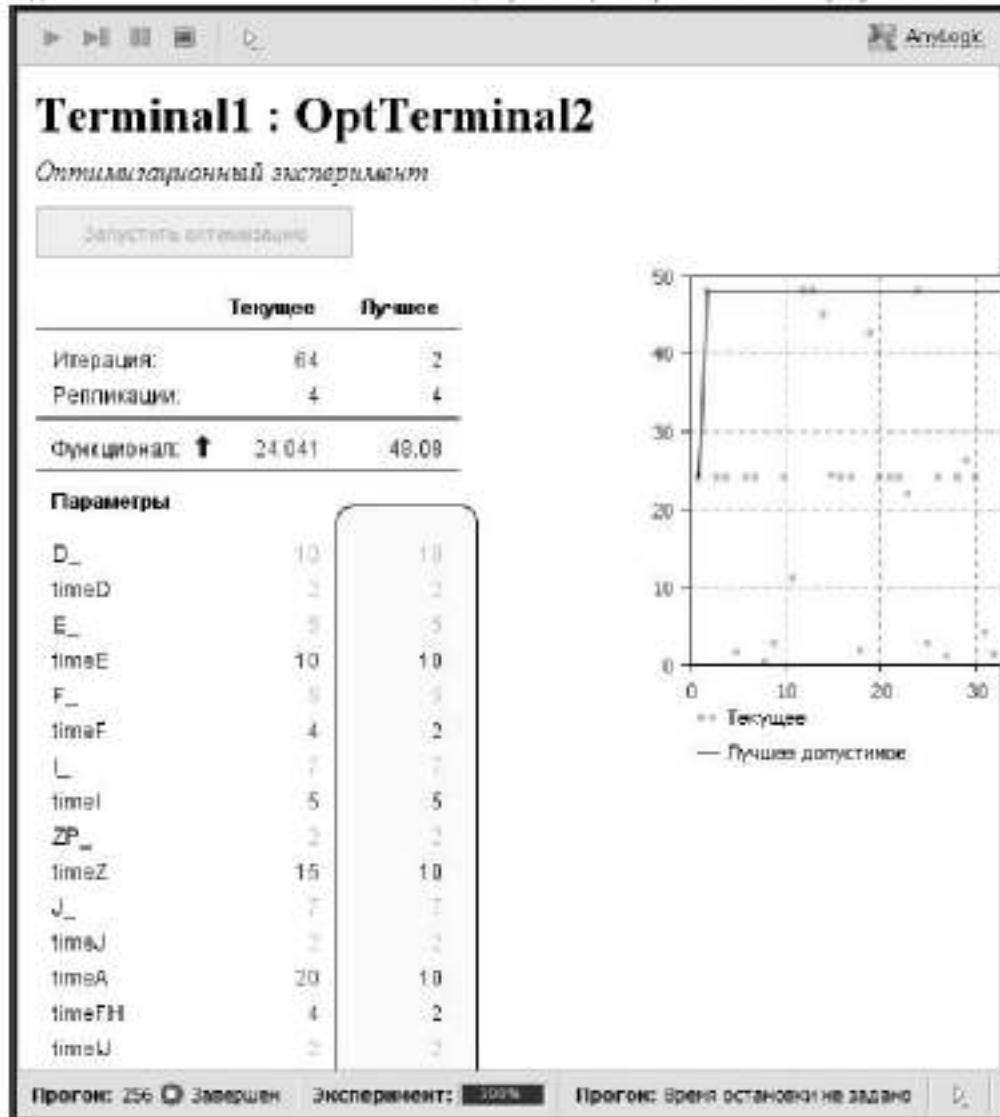


Рис. 7.11. Результаты второго оптимизационного эксперимента

Как уже отмечалось, значения факторов (параметров) в отсеивающих экспериментах GPSS World и оптимизационных экспериментах AnyLogic изменились однаково. Следовательно, в отчете 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.

Таблица 7.8.

<b>Второй эксперимент</b>		
KolObrCar	<b>48,049</b>	48,08
timeA	10	10
timeE	10	10
timeF	2	2
timeI	5	5
timeZ	10	10
timeFH	2	2
KolObrCar	<b>48,3</b>	48,08
timeA	10	10
timeE	15	10
timeF	4	2
timeI	5	5
timeZ	15	10
timeFH	4	2

В первом эксперименте минимальное время обработки одного автомобиля, определенное оптимизатором AnyLogic, составляет TimeObr = 31,799 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 отчете GPSS World (см. рис. 7.7) TimeObr = 31,844 мин. Как видим, отличие составляет 0,045 мин.

Во втором оптимизационном эксперименте максимальное количество обработанных автомобилей, определенное AnyLogic, составляет KolObrCar = 48,08 (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 отчете GPSS World (см. [рис. 7.8](#))  $KolObtCat = 48,049$ . Как видим, результаты практически одинаковы, если учесть измерение показателя целыми значениями (разница 0,03).

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.

# Модель предоставления ремонтных услуг

## Модель в AnyLogic

### Постановка задачи

В фирму предоставления ремонтных услуг поступают заявки п типов с вероятностями  $p_1, p_2, \dots, p_n$ , соответственно. Интервалы времени  $T_{ij}$  между двумя очередными поступлениями одного типа заявок случайные. Каждый любой тип заявки может требовать одного из  $a_1, a_2, \dots, a_k$  видов ремонта с вероятностями  $p_{a1}, p_{a2}, \dots, p_{ak}$  соответственно.

В фирме имеются  $n_1, n_2, \dots, n_p$  мастеров для выполнения заявок каждого типа соответственно. Мастера  $n_1$  выполняют заявки первого типа. Если их нет и мастера  $n_2, \dots, n_p$  групп заняты, они выполняют заявки этих типов. При этом поступающие заявки первого типа ожидают их освобождения. Мастера  $n_2$  выполняют заявки второго типа. Если их нет и мастера  $n_3, n_4, \dots, n_p$  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  $n_p$  выполняют заявки одного  $n$ -го типа.

Время выполнения заявки  $n$ -го типа случайное, не зависит от мастера, а зависит только от вида ремонта:  $T_{11}, T_{12}, T_{13}$  - для СС первого типа,  $T_{21}, T_{22}, T_{23}$  - для СС второго типа, ...,  $T_{n1}, T_{n2}, \dots, T_{np}$  - для СС  $n$ -го типа.

Прием и распределение заявок между группами мастеров осуществляется  $d$  диспетчерами. Время, затрачиваемое одним диспетчером на одну заявку,  $T_1$ , случайное. Диспетчерами не принимаются к ремонту  $q$  заявок всех типов.

### Исходные данные

```
exponential(Tп) = exponential(30); п = 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, Т01) = 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.

## 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.

## 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:

- ввод 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/п. Поэтому вместо п объектов имитации источников заявок будем использовать один.

Код типа заявки определяется в виде чисел 1, 2, 3, 4, так как п=4. Код вида ремонта определяется также числами 1...3 соответственно. Для этого используются следующие исходные данные:

- р1 ... р4 - вероятности поступления заявок 1...4 типов соответственно;
- р11 ... р43 - 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;
- выпРемВид11 ... 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.

## Ввод исходных данных

Элементы для ввода исходных данных разместим на диаграмме класса 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. Щелкните Далее. На второй странице Мастера создания модели выберите Начать создание модели "с нуля". Щелкните Далее.
4. Создайте область просмотра для размещения исходных данных на диаграмме класса Main. Из палитры Презентация перетащите элемент Область просмотра. Перейдите на страницу 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. Перейдите на страницу Дополнительные 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. Перейдите на страницу Дополнительные панели Свойства. Введите в поля X: 10, Y: 670, Ширина: 590, Высота: 560.
10. Перетащите элемент **text** и на странице Основные панели Свойства в поле Текст: введите Исходные данные.



Рис. 8.2. Размещение элементов Параметр и Простая переменная

11. В Палитре выделите Основная. Перетащите элементы Параметр на элемент Скругленный прямоугольник. Разместите их так, как показано на Рис. 8.2.
12. Значения свойств установите согласно табл. 8.1. Во всех элементах установите флагок Отображать имя. Тип элементов с именами колДисп, колМастеров1... колМастеров4 установите int. Тип остальных - 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
Tr	30	T1	15
p	4	Tol	2
T11	30	p11	0,5
T12	40	p12	0,75
T13	50	p13	1
T21	20	p21	0,5
T22	30	p22	0,75
T23	40	p23	1
T31	15	p31	0,5
T32	25	p32	0,75
T33	35	p33	1
T41	25	p41	0,5
T42	35	p42	0,75
T43	45	p43	1
колПрог	1000	колДисп	2
колМастеров12		колМастеров31	
колМастеров21		колМастеров41	

## Вывод результатов моделирования

- 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 Основные панели Свойства в поле Текст: введите Результаты моделирования. Поместите этот текст посередине в нижней части элемента Скругленный прямоугольник.
3. Из палитры Основная перетащите элементы Простая переменная. Разместите их и дайте им имена согласно рис. 8.2. Тип всех переменных `double`.

## 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 диаграмме класса `Main`. Из палитры Презентация перетащите элемент Область просмотра. Переийдите 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. Переийдите на страницу Дополнительные 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. Перейдите на страницу Дополнительные панели Свойства. Введите в поля X: 20, Y: 120, Ширина: 100, Высота: 140.
7. Перетащите элемент `text` на прямоугольник и на странице Основные панели Свойства в поле Текст: 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. Перейдите на страницу Дополнительные панели Свойства. Введите в поля X: 130, Y: 30, Ширина: 590, Высота: 410.
10. Перетащите элемент `text` на прямоугольник и на странице Основные панели Свойства в поле Текст: введите Диспетчеры.

На рис. 8.3 показаны объекты двух сегментов: Источники заявок и Диспетчеры. Приступим к их построению.

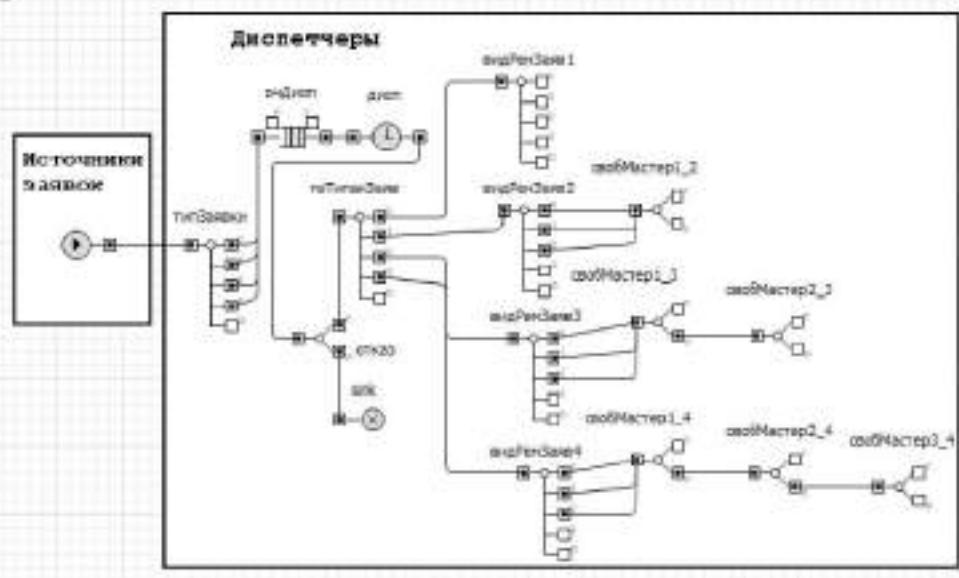


Рис. 8.3. Сегменты Источники заявок и Диспетчеры

### Сегмент Источники заявок

1. Из Основной библиотеки перетащите объект source на прямоугольник с названием Источники заявок (см. рис. 8.3).
  2. В панели Проект щелкните 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 Заявка.
  4. В поле Базовый класс: выберите из выпадающего списка Entity в качестве базового класса. Щелкните Далее.
  5. Появится вторая страница Мастера создания Java класса. Добавьте следующие поля Java класса:
- ```
double тип3;
double видР;
double времяР;
```

6. Оставьте выбранными флажки Создать конструктор и Создать метод `toString ()`.
7. Щелкните кнопку Готово. Вы увидите редактор кода и в автоматически созданный код вашего Java класса. Закройте код.
8. Выделите объект `source`. На странице Основные панели Свойства уберите флажок Отображать имя. В полях Класс заявки: и Новая заявка Entity замените Заявка.

Установите:

- Заявки прибывают согласно Времени между прибытиями
- Время между прибытиями `exponential(1/(Tp/n))`
- Количество заявок, прибывающих за один раз 1
- Действие 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.

## 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                   | типЗаявки                                                              |
| Использовать          | Условия                                                                |
| Условие 0             | entity.типЗ<=p1                                                        |
| Действие при выходе 0 | entity.типЗ=1;<br>постЗаявТип1++;<br>постЗаявТип++;                    |
| Условие 1             | entity.типЗ<=p2                                                        |
| Действие при выходе 1 | entity.типЗ=2;<br>постЗаявТип2++;<br>постЗаявТип++;                    |
| Условие 2             | entity.типЗ<=p3                                                        |
| Действие при выходе 2 | entity.типЗ=3;<br>постЗаявТип3++;                                      |
| Условие 3             | постЗаявТип++;                                                         |
| Действие при выходе 3 | entity.типЗ<=p4<br>entity.типЗ=4;<br>постЗаявТип4++;<br>постЗаявТип++; |
| Имя                   | отказ                                                                  |
| Выход true выбирается | С заданной вероятностью                                                |
| Вероятность [0..1]    | 0,98                                                                   |
| Имя                   | поТипамЗаяв                                                            |
| Использовать          | Условия                                                                |
| Условие 0             | entity.типЗ==1                                                         |
| Условие 1             | entity.типЗ==2                                                         |
| Условие 2             | entity.типЗ==3                                                         |
| Условие 3             | entity.типЗ==4                                                         |

|                              |                                                     |
|------------------------------|-----------------------------------------------------|
| <b>Имя</b>                   | видРемЗаявл                                         |
| <b>Использовать</b>          | Условия                                             |
| <b>Условие 0</b>             | entity.видР<=р11                                    |
| <b>Действие при выходе 0</b> | entity.видР=1;<br>entity.времяР=exponential(1/T11); |
| <b>Условие 1</b>             | entity. видР<=р12                                   |
| <b>Действие при выходе 1</b> | entity.видР=2;<br>entity.времяР=exponential(1/T12); |
| <b>Условие 2</b>             | entity.видР=<р13                                    |
| <b>Действие при выходе 2</b> | entity.видР=3;<br>entity.времяР=exponential(1/T13); |
| <b>Имя</b>                   | видРемЗаяв2                                         |
| <b>Использовать</b>          | Условия                                             |
| <b>Условие 0</b>             | entity.видР<=р21                                    |
| <b>Действие при выходе 0</b> | entity.видР=1;<br>entity.времяР=exponential(1/T21); |
| <b>Условие 1</b>             | entity. видР<=р22                                   |
| <b>Действие при выходе 1</b> | entity.видР=2;<br>entity.времяР=exponential(1/T22); |
| <b>Условие 2</b>             | entity.видР=<р23                                    |
| <b>Действие при выходе 2</b> | entity.видР=3;<br>entity.времяР=exponential(1/T23); |
| <b>Имя</b>                   | видРемЗаяв3                                         |
| <b>Использовать</b>          | Условия                                             |
| <b>Условие 0</b>             | entity.видР<=р31                                    |
| <b>Действие при выходе 0</b> | entity.видР=1;<br>entity.времяР=exponential(1/T31); |
| <b>Условие 1</b>             | entity. видР<=р32                                   |
| <b>Действие при выходе 1</b> | entity.видР=2;<br>entity.времяР=exponential(1/T32); |
| <b>Условие 2</b>             | entity.видР=<р33                                    |
| <b>Действие при выходе</b>   | entity.видР=3;                                      |

|                       |                                                                                                                         |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------|
| В.Д. Бон              | Концептуальное проектирование систем в AnyLogic и GPSS World                                                            |
| 2                     | entity.времяР=exponential(1/T33);                                                                                       |
| Имя                   | видРемЗаяв4                                                                                                             |
| Использовать          | Условия                                                                                                                 |
| Условие 0             | entity.видР<=p41                                                                                                        |
| Действие при выходе 0 | entity.видР=1;<br>entity.времяР=exponential(1/T41);                                                                     |
| Условие 1             | entity.видР<-p42                                                                                                        |
| Действие при выходе 1 | entity.видР=2;<br>entity.времяР=exponential(1/T42);                                                                     |
| Условие 2             | entity.видР=<p43                                                                                                        |
| Действие при выходе 2 | entity.видР=3;<br>entity.времяР=exponential(1/T43);                                                                     |
| Имя                   | свобМастер1_2                                                                                                           |
| Выход true выбирается | При выполнении условия<br><br>(очМастеров1.size()==0) &&<br>(мастеров1.size()<колМастеров1) &&<br>(мастера2.size() !=0) |
| Условие               |                                                                                                                         |
| Имя                   | свобМастер1_2                                                                                                           |
| Выход true выбирается | При выполнении условия<br><br>(очМастеров1.size()==0) &&<br>(мастеров1.size()<колМастеров1) &&<br>(мастера2.size() !=0) |
| Условие               |                                                                                                                         |
| Имя                   | свобМастер1_3                                                                                                           |
| Выход true выбирается | При выполнении условия<br><br>(очМастеров1.size()==0) &&<br>(мастеров1.size()<колМастеров1) &&<br>(мастера3.size() !=0) |
| Условие               |                                                                                                                         |
| Имя                   | свобМастер1_4                                                                                                           |
| Выход true выбирается | При выполнении условия                                                                                                  |

|                          |                                      |
|--------------------------|--------------------------------------|
|                          | (очМастеров1.size() == 0) &&         |
| Условие                  | (мастеров1.size() < колМастеров1) && |
|                          | (мастера4.size() != 0)               |
| Имя                      | свобМастер2_3                        |
| Выход true выбирается    | При выполнении условия               |
|                          | (очМастеров2.size() == 0) &&         |
| Условие                  | (мастера2.size() < колМастеров2) &&  |
|                          | (мастера4.size() != 0)               |
| Имя                      | свобМастер2_4                        |
| Выход true выбирается    | При выполнении условия               |
|                          | (очМастеров2.size() == 0) &&         |
| Условие                  | (мастера2.size() < колМастеров2) &&  |
|                          | (мастера4.size() != 0)               |
| Имя                      | свобМастер3_4                        |
| Выход true выбирается    | При выполнении условия               |
|                          | (очМастеров3.size() == 0) &&         |
| Условие                  | (мастера3.size() < колМастеров3) &&  |
|                          | (мастера4.size() != 0)               |
| Имя                      | очдисп                               |
| Максимальная вместимость | Установить флагок                    |
| Имя                      | дисп                                 |
| Задержка задается        | Явно                                 |
| Время задержки           | normal(T0, T1)                       |
| Вместимость              | колдисп                              |

Объектом типЗаявки разыгрывается код типа заявки. Например, в поступившей заявке entity.тип3=0.723. Проверяется условие 0: entity.тип3=0.723<=p1=0.5. Условие 0 не выполняется. Тогда проверяется условие 1: entity.тип3= 0.723 <=p2=0.75. Условие 1 выполняется. Заявка пропускается на выход 1. При этом

выполняется код, записанный в поле Действие при выходе 1,

```
entity.тип3=2;  
постЗаявТип2++;  
постЗаявТип++;
```

Кроме записи кода 2 в поле `entity.тип3=2`, учитывается количество поступивших заявок 2 типа и количество всех типов поступивших заявок. Последнее в дальнейшем 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 диспетчера рассмотрим в п. 8.1.7.3.

## 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. Перейдите на страницу Дополнительные панели Свойства. Введите в поля X: 740, Y: 30, Ширина: 200, Высота: 410.
3. Перетащите элемент text на прямоугольник и на странице Основные панели Свойства в поле Текст: введите Мастера.
4. Перетащите указанные объекты из Основной библиотеки на диаграмму класса Main. Разместите, дайте имена и соедините их так, как показано на [рис. 8.4](#).
5. У объектов очМастеров1...очМастеров4 укажите максимальную вместимость и класс заявки Заявка.
6. У объектов мастера1...мастера4 установите свойства:
  - Класс заявки: Заявка
  - Задержка задается Явно
  - Время задержки entity.времяР
  - Включить сбор статистики Установите флагок
7. Свойство Вместимость у этих же объектов укажите колМастеров1...колМастеров4 соответственно.

С выходов 0...2 объекта видРемЗаявл 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, то 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.

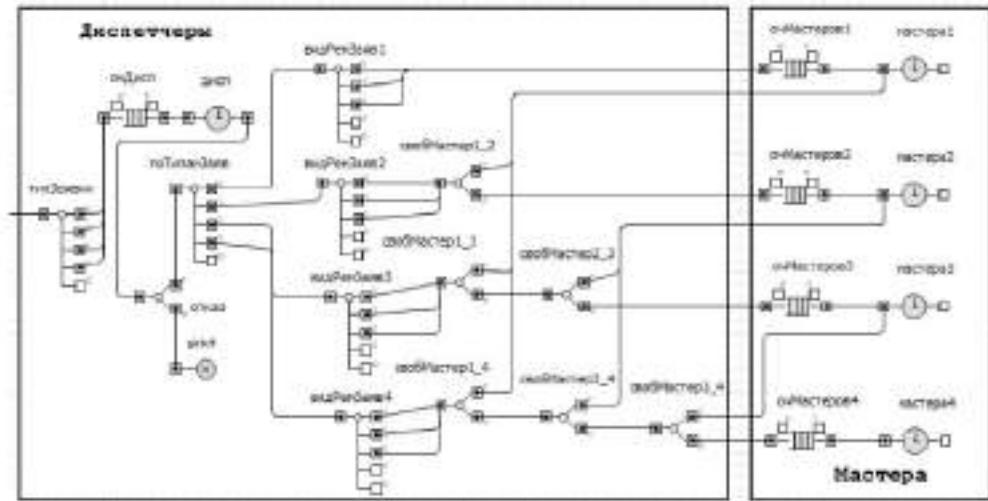


Рис. 8.4. Сегменты Диспетчера и Мастера

### 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. Перейдите на страницу Дополнительные панели Свойства. Введите в поля X: 970, Y: 30, Ширина: 250, Высота: 410.
3. Перетащите элемент `text` на прямоугольник и на странице Основные панели Свойства в поле Текст: введите Учёт выполненных заявок.
4. Перетащите указанные элементы на прямоугольник. Разместите, соедините и дайте имена согласно [рис. 8.5](#).

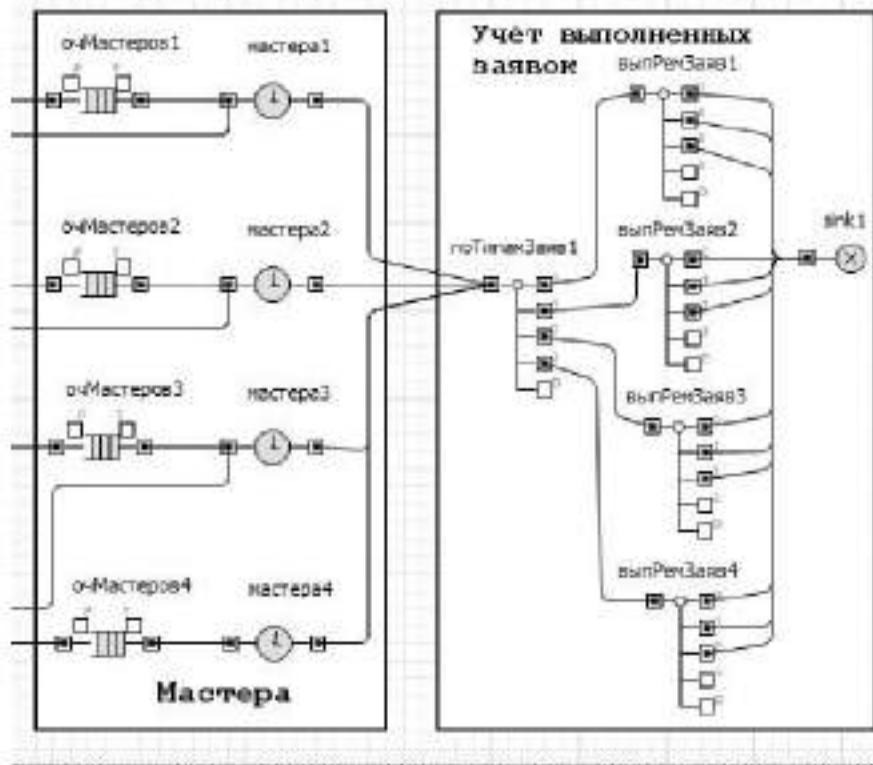


Рис. 8.5. Сегменты Мастера и Учёт выполненных заявок

5. Свойства элементов установите согласно табл. 8.3.

Таблица 8.3.

| Свойства              | Значение                                                          |
|-----------------------|-------------------------------------------------------------------|
| Имя                   | поТипамЗаявл                                                      |
| Использовать          | Условия                                                           |
| Условие 0             | entity.тип3==1<br>выпЗаявТип1++;<br>выпЗаявТип++;<br>верВыпЗаявл= |
| Действие при выходе 0 | выпЗаявТип1/постЗаявТип1;                                         |
| Условие 1             | entity.тип3==2<br>выпЗаявТип2++;                                  |

|                              |                                                                   |
|------------------------------|-------------------------------------------------------------------|
| <b>Действие при выходе 1</b> | выпЗаявТип++;<br>верВыпЗаяв2=                                     |
| <b>Условие 2</b>             | entity.тип3==3<br>выпЗаявТип3++;<br>выпЗаявТип++;<br>верВыпЗаяв3= |
| <b>Действие при выходе 2</b> | выпЗаявТип3 / постЗаявТип3;                                       |
| <b>Условие 3</b>             | entity.тип3==4<br>выпЗаявТип4++;<br>выпЗаявТип++;<br>верВыпЗаяв4= |
| <b>Действие при выходе 3</b> | выпЗаявТип4 / постЗаявТип4;                                       |
| <b>Имя</b>                   | выпРемЗаяв1                                                       |
| <b>Использовать</b>          | <b>Условия</b>                                                    |
| <b>Условие 0</b>             | entity.видР==1                                                    |
| <b>Действие при выходе 0</b> | выпРемВидаС1++;                                                   |
| <b>Условие 1</b>             | entity.видР==2                                                    |
| <b>Действие при выходе 1</b> | выпРемВидаС2++;                                                   |
| <b>Условие 2</b>             | entity.видР==3                                                    |
| <b>Действие при выходе 2</b> | выпРемВидаС3++;                                                   |
| <b>Имя</b>                   | выпРемЗаяв2                                                       |
| <b>Использовать</b>          | <b>Условия</b>                                                    |
| <b>Условие 0</b>             | entity.видР==1                                                    |
| <b>Действие при выходе 0</b> | выпРемВидаС1++;                                                   |
| <b>Условие 1</b>             | entity.видР==2                                                    |
| <b>Действие при выходе 1</b> | выпРемВидаС2++;                                                   |
| <b>Условие 2</b>             | entity.видР==3                                                    |
| <b>Действие при выходе 2</b> | выпРемВидаС3++;                                                   |
| <b>Имя</b>                   | выпРемЗаяв3                                                       |
| <b>Использовать</b>          | <b>Условия</b>                                                    |
| <b>Условие 0</b>             | entity.видР==1                                                    |

|                              |                 |
|------------------------------|-----------------|
| <b>Действие при выходе 0</b> | выпРемВида31++; |
| <b>Условие 1</b>             | entity.видР==2  |
| <b>Действие при выходе 1</b> | выпРемВида32++; |
| <b>Условие 2</b>             | entity.видР==3  |
| <b>Действие при выходе 2</b> | выпРемВида33++; |
| <b>Имя</b>                   | выпРемЗаяв4     |
| <b>Использовать</b>          | <b>Условия</b>  |
| <b>Условие 0</b>             | entity.видР==1  |
| <b>Действие при выходе 0</b> | выпРемВида41++; |
| <b>Условие 1</b>             | entity.видР==2  |
| <b>Действие при выходе 1</b> | выпРемВида42++; |
| <b>Условие 2</b>             | entity.видР==3  |
| <b>Действие при выходе 2</b> | выпРемВида43++; |

6. Установите значения свойств объекта sink1:

- Класс заявки: Заявка
- Действие при входе верВыпЗаяв=выпЗаявТип/  
постЗаявТип

Объект поТипамЗаявл 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.

### 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: 1440000.0 (модельное время  $\uparrow$  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**. 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 37,412. Вероятность выполнения составляет 0, 978. Всего выполнено заявок всех типов 187,756 с вероятностью 0,978.

## Модель в GPSS World

### Состав модели в GPSS World

Как уже отмечалось, фирма предоставления ремонтных услуг представляет собой многофазную многоканальную систему массового обслуживания разомкнутого типа (см. [рис. 8.1](#)).

Модель предоставления ремонтных услуг должна состоять из следующих сегментов:

- ввод исходных данных;
- сегмент имитации поступления заявок;
- сегмент имитации работы диспетчеров;
- сегмент имитации работы мастеров 1 группы;
- сегмент имитации работы мастеров 2 группы;
- сегмент имитации работы мастеров 3 группы;
- сегмент имитации работы мастеров 4 группы;
- сегмент учёта выполненных заявок и ремонтов;
- сегмент задания времени моделирования и расчёта результатов моделирования.

## Программа GPSS-модели

Ниже приводится программа.

; Модель предоставления ремонтных услуг  
; Замена имен МКУ номерами  
Rem1 EQU 1 ; 1 группа мастеров  
Rem2 EQU 2 ; 2 группа мастеров  
Rem3 EQU 3 ; 3 группа мастеров  
Rem4 EQU 4 ; 4 группа мастеров  
; Задание МКУ-групп мастеров  
Dis STORAGE 3 ; Количество диспетчеров  
Rem1 STORAGE 2 ; Количество мастеров 1 группы  
Rem2 STORAGE 2 ; Количество мастеров 2 группы  
Rem3 STORAGE 2 ; Количество мастеров 3 группы  
Rem4 STORAGE 1 ; Количество мастеров 4 группы  
; Задание исходных данных  
VrMod EQU 1440 ; Время моделирования, 1 ед. мод. вр. = 1 мин  
n\_ EQU 4 ; Количество типов заявок  
T1 EQU 15 ; Среднее время работы диспетчера с поступившей заявкой  
To1 EQU 2 ; Среднеквадратичное отклонение времени работы диспетчера  
Tp\_ EQU 20 ; Средний интервал времени поступления одного типа заявки  
q\_ EQU 0.02 ; Доля не принятых заявок  
TipSS FUNCTION RN892,D4 ; Функция распределения поступающих заявок  
.2,1/.5,2/.75,3/1,4  
VidRem FUNCTION RN892,D3 ; Функция распределения видов ремонта  
.5,1/.75,2/1,3  
; Среднее время ремонта по заявкам  
VrRemTip1 FUNCTION P2,D3 ; типа 1  
1,30/2,40/3,50  
VrRemTip2 FUNCTION P2,D3 ; типа 2  
1,20/2,30/3,40  
VrRemTip3 FUNCTION P2,D3 ; типа 3  
1,15/2,25/3,35  
VrRemTip4 FUNCTION P2,D3 ; типа 4  
1,25/2,35/3,40  
; Сегмент имитации поступления заявок  
GENERATE (Exponential(892,0,(Tp\_/\n\_))) ; Источники заявок  
ASSIGN 1,Fn\$TipSS ; Код типа заявки в Р1  
ASSIGN 2,Fn\$VidRem ; Код вида ремонта в Р2  
ASSIGN 5,P1 ; Код типа заявки также в Р5  
; Запись в Р3 среднего времени вида ремонта по заявкам  
Met0 TRANSFER ,(Met0+((P1#2)-1))

```

Met1_ ASSIGN 3,FN$VrRemTip1 ; типа 1
    TRANSFER .Met1
Met2_ ASSIGN 3,FN$VrRemTip2 ; типа 2
    TRANSFER .Met1
Met3_ ASSIGN 3,FN$VrRemTip3 ; типа 3
    TRANSFER .Met1
Met4_ ASSIGN 3,FN$VrRemTip4 ; типа 4
; Сегмент имитации работы диспетчеров
Met1 QUEUE OCH ; Занять очередь к диспетчеру
    ENTER DIS ; Занять свободного диспетчера
    DEPART OCH ; Покинуть очередь к диспетчеру
    ADVANCE (Normal(892,T1,To1)) ; Имитация работы LEAVE DIS ;
    TRANSFER q_.Met20 ; Отказать q заявкам
Met2 TESTE P1,1,Met21 ; Мастерам 1 группы? Да,
Met25 TRANSFER .Met3 ; отправить мастерам 1
Met21 TESTE P1,2,Met22 ; Мастерам 2 группы
Met26 GATE SF P1,Met4 ; Мастера 2 заняты? Если да,
    ASSIGN 4,1 ; запись в Р4 признака мастера 1
    ASSIGN 1,1 ; запись в Р1 признака мастера 1
    GATE SF P4,Met3 ; Свободны ли мастера 1?
    ASSIGN 1,2 ; Заняты, В Р1 признак мастера 2
    TRANSFER .Met4 ; и отправить мастерам 2
Met22 TESTE P1,3,Met23 ; Мастерам 3 группы
Met27 GATE SF P1,Met5 ; Мастера 3 заняты? Если да,
    ASSIGN 4,1 ; запись в Р4 признака мастера 1
    ASSIGN 1,1 ; запись в Р1 признака мастера 1
    GATE SF P4,Met3 ; Свободны ли мастера 1?
    ASSIGN 4,2 ; запись в Р4 признака мастера 2
    ASSIGN 1,2 ; запись в Р1 признака мастера 2
    GATE SF P4,Met4 ; свободны ли мастера 3?
    ASSIGN 1,3 ; Заняты, В Р1 признак мастера 3
    TRANSFER .Met5 ; и отправить мастерам 3
Met23 TESTE P1,4 ; Мастерам 4 группы
    GATE SF P1,Met6 ; Мастера 4 заняты? Если да,
    ASSIGN 4,1 ; запись в Р4 признака мастера 1
    ASSIGN 1,1 ; запись в Р1 признака мастера 1
    GATE SF P4,Met3 ; Свободны ли мастера 1?
    ASSIGN 4,2 ; запись в Р4 признака мастера 2
    ASSIGN 1,2 ; запись в Р1 признака мастера 2

```

GATE SF P4,Met4 ; Свободны ли мастера 2?  
 ASSIGN 4,3 ; запись в P4 признака мастера 3  
 ASSIGN 1,3 ; запись в P1 признака мастера 3  
 GATE SF P4,Met5 ; Свободны ли мастера 3?  
 ASSIGN 1,4 ; запись в P1 признака мастера 4  
 TRANSFER ,Met6 ; и отправить мастерам 4  
 ; Сегмент имитации работы мастеров 1 группы  
 MET3 ENTER P1 ; Занять свободного мастера 1 группы  
 ADVANCE (Exponential(892,0,P3));Имитация ремонта  
 LEAVE P1 ; Освободить свободного мастера 1  
 TRANSFER ,Met7 ; Отправить для учета  
 ; Сегмент имитации работы мастеров 2 группы  
 MET4 ENTER P1 ; Занять свободного мастера 2 группы  
 ADVANCE (Exponential(892,0,P3));Имитация ремонта LEAVE P1 ; Освободить свободного мастера 2  
 TRANSFER ,Met7 ; Отправить для учета  
 ; Сегмент имитации работы мастеров 3 группы  
 MET5 ENTER P1 ; Занять свободного мастера 3  
 ADVANCE (Exponential(892,0,P3));Имитация ремонта  
 LEAVE P1 ; Освободить свободного мастера 3  
 TRANSFER ,Met7 ; Отправить для учета  
 ; Сегмент имитации работы мастеров 4 группы  
 MET6 ENTER P1 ; Занять свободного мастера 4  
 ADVANCE (Exponential(892,0,P3));Имитация ремонта  
 LEAVE P1 ; Освободить свободного мастера 4  
 ; Сегмент учёта выполненных заявок и ремонтов за все прогоны моделей  
 MET7 TEST E P5,1,Met9\_ ; заявок всего  
 MET8 TRANSFER ,(Met8+P2) ; заявок типа 1  
 Met81 TERMINATE ; ремонтов вида 11  
 Met82 TERMINATE ; ремонтов вида 12  
 Met83 TERMINATE ; ремонтов вида 13  
 MET9\_ TEST E P5,2,Met10\_  
 Met9 TRANSFER ,(Met9+P2) ; заявок типа 2  
 Met91 TERMINATE ; ремонтов вида 21  
 Met92 TERMINATE ; ремонтов вида 22  
 Met93 TERMINATE ; ремонтов вида 23  
 MET10\_ TEST E P5,3,Met11  
 Met10 TRANSFER ,(Met10+P2) ; заявок типа 3  
 Met101 TERMINATE ; ремонтов вида 31  
 Met102 TERMINATE ; ремонтов вида 32

```

Met103 TERMINATE      ; ремонтов вида 33
MET11 TRANSFER ,(Met11+P2) ; заявок типа 4
Met111 TERMINATE      ; ремонтов вида 41
Met112 TERMINATE      ; ремонтов вида 42
Met113 TERMINATE      ; ремонтов вида 43
MET20 TERMINATE       ; Количество не принятых заявок
; Сегмент задания времени моделирования и расчета результатов моделей
GENERATE VrMod         ; Время моделирования
TEST L X$Prog,TG1,Met41 ; Если X$Prog<TG1, то
SAVEVALUE Prog,TG1     ; запомнить в X$Prog количество прогонов модели
Met41 TEST E TG1,1,Met42 ; Если TG1=1, то расчет результатов модели
; Количество выполненных заявок
SAVEVALUE KolZajav1,(N$Met8/X$Prog) ; типа 1
SAVEVALUE KolZajav2,(N$Met9/X$Prog) ; типа 2
SAVEVALUE KolZajav3,(N$Met10/X$Prog) ; типа 3
SAVEVALUE KolZajav4,(N$Met11/X$Prog) ; типа 4
; Вероятность выполнения заявок
SAVEVALUE VerZajav1,(N$Met8/N$Met1_) ; типа 1
SAVEVALUE VerZajav2,(N$Met9/N$Met2_) ; типа 2
SAVEVALUE VerZajav3,(N$Met10/N$Met3_) ; типа 3
SAVEVALUE VerZajav4,(N$Met11/N$Met4_) ; типа 4
SAVEVALUE VerZajav,(N$Met7/N$Met0) ; всех типов
; Количество выполненных видов ремонтов
SAVEVALUE 11,(NSMet81/X$Prog) ; вида 11
SAVEVALUE 12,(NSMet82/X$Prog) ; вида 12
SAVEVALUE 13,(NSMet83/X$Prog) ; вида 13
SAVEVALUE 21,(NSMet91/X$Prog) ; вида 21
SAVEVALUE 22,(NSMet92/X$Prog) ; вида 22
SAVEVALUE 23,(NSMet93/X$Prog) ; вида 23
SAVEVALUE 31,(NSMet101/X$Prog) ; вида 31
SAVEVALUE 32,(NSMet102/X$Prog) ; вида 32
SAVEVALUE 33,(NSMet103/X$Prog) ; вида 33
SAVEVALUE 41,(NSMet111/X$Prog) ; вида 41
SAVEVALUE 42,(NSMet112/X$Prog) ; вида 42
SAVEVALUE 43,(NSMet113/X$Prog) ; вида 43
Met42 TERMINATE 1
START 1000 ; Количество прогонов модели

```

Заявки имитируются транзактами, а диспетчеры и группы мастеров -

многоканальными устройствами (МКУ).

Для упрощения построения модели МКУ даны имена Rem1 ... Rem4, которые заменены номерами 1...4. Но в GPSS-программе порядок записи обратный.

Для ввода исходных данных, где это уместно, например, среднее время ремонта, использованы функции, что сокращает количество строк программы и также упрощает её построение.

В сегменте имитации поступления заявок разыгрываются тип заявки и вид ремонта, которые записываются в параметры транзакта 1 и 2 соответственно. Код типа заявки записывается также в параметр 5, так как в последующем в сегменте имитации работы диспетчеров юд в параметре 1 может изменяться, а в параметре 5 нет, что необходимо для сегмента учёта выполненных заявок и ремонтов.

В этом же сегменте в параметр 3 транзакта-заявки заносится среднее время выполнения вида ремонта.

Сегмент имитации работы диспетчеров начинается заданием очереди по аналогии с AnyLogic-моделью, хотя можно было бы обойтись и без этого, так как МКУ имеет свою очередь.

Принято, что, как и в AnyLogic-модели, решение на распределение заявок в текущий момент времени принимает диспетчер.

Предположим, что поступила заявка типа 4. Диспетчер проверяет занятость мастеров 4 группы:

GATE SF P1,Met6 ; Мастера 4 заняты? Если да,

Если мастера 4 группы свободны, то заявка отправляется им, то есть на Met6. Если же они заняты, то проверяется занятость мастеров 1 группы:

```
ASSIGN 4,1 ; запись в Р4 признака мастера 1  
ASSIGN 1,1 ; запись в Р1 признака мастера 1  
GATE SF P4,Met3 ; Свободны ли мастера 1?
```

Но предварительно в Р4 и Р1 заносятся коды мастеров 1 группы. Если мастера 1 группы свободны, заявка отправляется им.

Если же мастера 1 группы заняты, то аналогично проверяется занятость мастеров 2 группы. Если они заняты - мастеров 3 группы. И если мастера 3 группы заняты, заявка отправляется мастерам 4 группы:

```
ASSIGN 1,4 ; запись в Р1 признака мастера 4
TRANSFER ,Met6 ; и отправить мастерам 4
```

Опять же предварительно в параметре 1 транзакта-заявки восстанавливается её первоначальный код.

Сегменты имитации работы мастеров 1...4 групп построены одинаково. Следует заметить, что можно было бы обойтись следующими строками:

```
MET3 ENTER P1 ; Занять свободного мастера 1 группы
ADVANCE (Exponential(892,0,P3));Имитация ремонта
LEAVE P1 ; Освободить свободного мастера 1
TRANSFER ,Met7 ; Отправить для учета
```

Заменив также в сегменте имитации работы диспетчеров Met4, Met5, Met6 на Met3.

В сегменте учёта выполненных заявок и ремонтов в блоки с метками Met8, Met9, Met10, Met11 входят заявки типов 1...4 соответственно за все прогоны модели, поэтому в сегменте задания времени моделирования и расчёта результатов моделирования для определения среднего количества выполненных заявок производится деление на количество выполненных прогонов. Тоже самое производится и при определении среднего количества выполненных видов ремонтов.

Количество выполненных заявок и вероятности их выполнения сохраняются в ячейках KolZajav1... KolZajav4 и VerZajav1 ... VerZajav4 соответственно.

Вероятности выполнения видов ремонтов также сохраняются в ячейках, например, видов 1, 2, 3 по заявке типа 1 в ячейках 11, 12 и 13 соответственно.

## Интерпретация результатов моделирования

Всего выполнено три группы экспериментов. 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  $T_p = 20$  мин.

Для выяснения разницы между полученными результатами поступим так. Поскольку проведено по два эксперимента с разными начальными числами в GPSS World и AnyLogic, будем находить минимальные и максимальные значения показателей, разность между которыми и составит различие.

Например, разница вероятностей выполнения всех заявок составляет  $\Delta_5 = 0,001 \dots 0,002$ , коэффициентов использования мастеров всех групп -  $\Delta_6 = 0,005 \dots 0,025$ . Количество выполненных заявок и ремонтов, если считать с точностью до целого, одно и тоже в обеих системах.

Во второй группе экспериментов (табл. 8.5) все исходные данные соответствуют постановке задачи, то есть средний интервал времени поступления заявок  $T_p = 30$  мин.

Во второй группе разница вероятностей выполнения всех заявок составляет  $\Delta_5 = 0 \dots 0,001$ , коэффициентов использования мастеров всех групп практически такое же -  $\Delta_6 = 0,005 \dots 0,023$ . 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 2 и 3 групп:  $T_p = 20$  мин, колДисп = 3, колМастеров2 = 2, колМастеров3 = 2.

Здесь также вероятность выполнения всех заявок отличается незначительно  $\Delta_5 = 0,001 \dots 0,002$ , но несколько больше разница между коэффициентами занятости групп мастеров-ремонтников:  $\Delta_6 = 0,017 \dots 0,024$ .

Таблица 8.4. Показатели фирмы предоставления ремонтных услуг

| Показатели                              | Системы моделирования                       |        |          |        |
|-----------------------------------------|---------------------------------------------|--------|----------|--------|
|                                         | GPSS World                                  |        | AnyLogic |        |
|                                         | Начальные числа генераторов случайных чисел |        |          |        |
|                                         | 892                                         | 6547   | 892      | 6547   |
| Среднее время поступления заявок 20 мин |                                             |        |          |        |
| Выполнено заявок типа 1                 | 37,356                                      | 37,879 | 37,333   | 37,428 |
| $\Delta_1 = 0,023 \dots 0,451$          |                                             |        |          |        |
| ремонтов: вида 11                       | 18,664                                      | 19,021 | 18,488   | 18,705 |
| вида 12                                 | 9,277                                       | 9,461  | 9,423    | 9,354  |
| вида 13                                 | 9,415                                       | 9,397  | 9,462    | 9,369  |
| Выполнено заявок типа 2                 | 56,633                                      | 56,297 | 56,641   | 56,33  |
| $\Delta_2 = 0,008 \dots 0,033$          |                                             |        |          |        |
| ремонтов: вида 21                       | 28,273                                      | 28,195 | 28,115   | 28,378 |
| вида 22                                 | 14,118                                      | 13,9   | 14,264   | 13,914 |
| вида 23                                 | 13,942                                      | 14,202 | 14,262   | 14,038 |
| Выполнено заявок типа 3                 | 47,305                                      | 47,075 | 47,05    | 46,946 |
| $\Delta_3 = 0,129 \dots 0,255$          |                                             |        |          |        |
| ремонтов: вида 31                       | 23,808                                      | 23,57  | 23,469   | 23,414 |
| вида 32                                 | 11,851                                      | 11,831 | 11,792   | 11,788 |
| вида 33                                 | 11,646                                      | 11,674 | 11,789   | 11,744 |
| Выполнено заявок типа 4                 | 47,16                                       | 46,857 | 47,099   | 47,486 |
| $\Delta_4 = 0,242 \dots 0,326$          |                                             |        |          |        |
| ремонтов: вида 41                       | 23,504                                      | 23,443 | 23,852   | 23,552 |
| вида 42                                 | 11,962                                      | 11,811 | 11,702   | 12,05  |
| вида 43                                 | 11,694                                      | 11,603 | 11,545   | 11,884 |

|                                       |       |       |       |       |
|---------------------------------------|-------|-------|-------|-------|
| Вероятность выполнения всех заявок    | 0,651 | 0,654 | 0,653 | 0,655 |
| $\Delta_5 = 0,001 \dots 0,002$        |       |       |       |       |
| Коэффициенты использования мастеров 1 | 0,829 | 0,832 | 0,834 | 0,838 |
| мастеров 2                            | 0,859 | 0,865 | 0,868 | 0,868 |
| мастеров 3                            | 0,655 | 0,659 | 0,669 | 0,668 |
| мастеров 4                            | 0,633 | 0,637 | 0,655 | 0,662 |
| $\Delta_6 = 0,005 \dots 0,025$        |       |       |       |       |
| Коэффициент использования диспетчеров | 1,000 | 1,000 | 1,000 | 1,000 |

Таблица 8.5. Показатели фирмы предоставления ремонтных услуг

| Показатели                              | Системы моделирования                       |        |          |        |
|-----------------------------------------|---------------------------------------------|--------|----------|--------|
|                                         | GPSS World                                  |        | AnyLogic |        |
|                                         | Начальные числа генераторов случайных чисел |        |          |        |
|                                         | 892                                         | 6547   | 892      | 6547   |
| Среднее время поступления заявок 30 мин |                                             |        |          |        |
| Выполнено заявок типа 1                 | 37,411                                      | 37,49  | 37,412   | 37,466 |
| $\Delta_1 = 0,001 \dots 0,024$          |                                             |        |          |        |
| ремонтов: вида 11                       | 18,723                                      | 18,705 | 18,723   | 18,846 |
| вида 12                                 | 9,274                                       | 9,325  | 9,425    | 9,236  |
| вида 13                                 | 9,414                                       | 9,46   | 9,264    | 9,384  |
| Выполнено заявок типа 2                 | 56,187                                      | 56,597 | 56,41    | 56,36  |
| $\Delta_2 = 0,173 \dots 0,187$          |                                             |        |          |        |
| ремонтов: вида 21                       | 28,19                                       | 28,186 | 28,13    | 27,941 |
| вида 22                                 | 14,004                                      | 14,341 | 14,164   | 14,2   |
| вида 23                                 | 13,993                                      | 14,07  | 14,116   | 14,219 |
| Выполнено заявок типа 3                 | 47,047                                      | 46,811 | 47,082   | 47,151 |
| $\Delta_3 = 0,104 \dots 0,271$          |                                             |        |          |        |
| ремонтов: вида 31                       | 23,382                                      | 23,373 | 23,648   | 23,558 |
| вида 32                                 | 11,907                                      | 11,842 | 11,649   | 11,798 |

|                                           |        |        |        |        |
|-------------------------------------------|--------|--------|--------|--------|
| вида 33                                   | 11,758 | 11,596 | 11,785 | 11,795 |
| Выполнено заявок типа 4                   | 47,259 | 47,223 | 46,852 | 46,909 |
| $\Delta_4 = 0,350 \dots 0,381$            |        |        |        |        |
| ремонтов: вида 41                         | 23,785 | 23,647 | 23,517 | 23,311 |
| вида 42                                   | 11,641 | 11,844 | 11,682 | 11,791 |
| вида 43                                   | 11,833 | 11,732 | 11,653 | 11,807 |
| Вероятность выполнения всех заявок        | 0,977  | 0,978  | 0,978  | 0,978  |
| $\Delta_5 = 0 \dots 0,001$                |        |        |        |        |
| Коэффициенты использования:<br>мастеров 1 | 0,829  | 0,828  | 0,834  | 0,836  |
| мастеров 2                                | 0,859  | 0,858  | 0,867  | 0,868  |
| мастеров 3                                | 0,654  | 0,653  | 0,662  | 0,667  |
| мастеров 4                                | 0,634  | 0,632  | 0,649  | 0,657  |
| $\Delta_6 = 0,005 \dots 0,023$            |        |        |        |        |
| Коэффициент использования<br>диспетчеров  | 0,999  | 1,000  | 0,998  | 0,999  |

Таблица 8.6. Показатели фирмы предоставления ремонтных услуг

| Показатели                                     | Системы моделирования                          |        |          |        |
|------------------------------------------------|------------------------------------------------|--------|----------|--------|
|                                                | GPSS World                                     |        | AnyLogic |        |
|                                                | Начальные числа генераторов<br>случайных чисел |        |          |        |
|                                                | 892                                            | 6547   | 892      | 6547   |
| <b>Среднее время поступления заявок 20 мин</b> |                                                |        |          |        |
| Выполнено заявок типа 1                        | 56,081                                         | 56,592 | 56,351   | 56,221 |
| $\Delta_1 = 0,140 \dots 0,241$                 |                                                |        |          |        |
| ремонтов: вида 11                              | 28,137                                         | 28,398 | 28,098   | 28,069 |
| вида 12                                        | 14,082                                         | 14,158 | 14,056   | 14,152 |
| вида 13                                        | 13,862                                         | 14,036 | 14,197   | 14     |
| Выполнено заявок типа 2                        | 85,05                                          | 84,227 | 84,938   | 84,74  |
| $\Delta_2 = 0,112 \dots 0,513$                 |                                                |        |          |        |

| В.Д. Бон                                  | Концептуальное проектирование систем в AnylLogic и GPSS World |        |        |        |
|-------------------------------------------|---------------------------------------------------------------|--------|--------|--------|
| ремонтов: вида 21                         | 42,538                                                        | 42,087 | 42,221 | 42,513 |
| вида 22                                   | 21,214                                                        | 21,196 | 21,353 | 21,285 |
| вида 23                                   | 21,298                                                        | 20,974 | 21,364 | 20,942 |
| Выполнено заявок типа 3                   | 70,491                                                        | 70,679 | 70,175 | 70,586 |
| $\Delta_3 = 0,103 \dots 0,216$            |                                                               |        |        |        |
| ремонтов: вида 31                         | 35,439                                                        | 35,256 | 35,05  | 35,072 |
| вида 32                                   | 17,549                                                        | 17,671 | 17,584 | 17,659 |
| вида 33                                   | 17,503                                                        | 17,752 | 17,541 | 17,855 |
| Выполнено заявок типа 4                   | 70,355                                                        | 70,244 | 70,21  | 70,486 |
| $\Delta_4 = 0,034 \dots 0,131$            |                                                               |        |        |        |
| ремонтов: вида 41                         | 35,125                                                        | 35,112 | 35,104 | 34,974 |
| вида 42                                   | 17,607                                                        | 17,757 | 17,543 | 17,696 |
| вида 43                                   | 17,623                                                        | 17,375 | 17,563 | 17,816 |
| Вероятность выполнения всех заявок        | 0,978                                                         | 0,979  | 0,977  | 0,98   |
| $\Delta_5 = 0,001 \dots 0,002$            |                                                               |        |        |        |
| Коэффициенты использования:<br>мастеров 1 | 0,928                                                         | 0,932  | 0,949  | 0,949  |
| мастеров 2                                | 0,871                                                         | 0,874  | 0,895  | 0,893  |
| мастеров 3                                | 0,663                                                         | 0,667  | 0,645  | 0,646  |
| мастеров 4                                | 0,771                                                         | 0,772  | 0,792  | 0,796  |
| $\Delta_6 = 0,017 \dots 0,024$            |                                                               |        |        |        |
| Коэффициент использования<br>диспетчеров  | 0,999                                                         | 0,998  | 0,998  | 0,999  |

# Модель функционирования системы воздушных перевозок

## Модель в AnyLogic

### 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 с грузоподъемностями  $q_1$  и  $q_2$  шт. контейнеров соответственно ( $q_1 < q_2$ ). 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.

### 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 |
|                                       | 1         | 1   |

|                                                                                                                      | 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   |
| 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 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 |

## 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.

## 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;
- коэфПолётA12, коэфПолётB12, коэфПолётA21, коэфПолётB21 - 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=достK21/всегоПостK2 - коэффициент доставки грузов из аэропорта 2 в аэропорт 1, достK21 - количество доставленного груза из аэропорта 2 в аэропорт 1, всегоПостK2 - количество всего поступившего груза в аэропорт 2;

- коэфДост12=достK12/всегоПостK1 - коэффициент доставки грузов из аэропорта 1 в аэропорт 2, достK12 - количество доставленного груза из аэропорта 1 в аэропорт 2, всегоПостK1 - количество всего поступившего груза в аэропорт 1;
- коэфДост=
 
$$(\text{достK12} + \text{достK21}) / (\text{всегоПостK1} + \text{всегоПостK2})$$
 - коэффициент доставки грузов системой перевозок в целом;
- коэфИспСам1А=коэфПогр1А+коэфРазгр1А
  - \*коэфПолётA12 - коэффициент использования самолётов типа А в аэропорту 1;
- коэфИспСам1Б=коэфПогр1Б+коэфРазгр1Б
  - +коэфПолётB12 - коэффициент использования самолётов типа Б в аэропорту 1;
- коэфИспСам2А, коэфИспСам2Б - коэффициенты использования самолётов типа А и Б в аэропорту 2.

## 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.

## 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.

- Перетащите элемент Скруглённый прямоугольник на элемент Область просмотра с именем исхДанные.
- Перейдите на страницу Дополнительные панели Свойства. Введите в поля X : 30, Y : 1880, Ширина : 580, Высота : 450.
- Перетащите элемент text и в поле Текст: введите Исходные данные.
- Из библиотеки Основная перетащите элементы Параметр и поместите их так, как на [рис. 9.2](#).
- Значения свойств установите согласно [Табл. 9.3](#).

| Исходные данные                                 |                                                    |                                                 |                                                    |
|-------------------------------------------------|----------------------------------------------------|-------------------------------------------------|----------------------------------------------------|
| Самолёты типа А                                 |                                                    | Самолёты типа Б                                 |                                                    |
| <input checked="" type="radio"/> колSamTypA     | <input checked="" type="radio"/> грузПодSamA       | <input checked="" type="radio"/> колSamTypB     | <input checked="" type="radio"/> грузПодSamB       |
| Аэропорт 1                                      |                                                    | Аэропорт 2                                      |                                                    |
| <input checked="" type="radio"/> минКонтПост1   |                                                    | <input checked="" type="radio"/> минКонтПост2   |                                                    |
| <input checked="" type="radio"/> максКонтПост1  |                                                    | <input checked="" type="radio"/> максКонтПост2  |                                                    |
| <input checked="" type="radio"/> срВрПостКонт1  |                                                    | <input checked="" type="radio"/> срВрПостКонт2  |                                                    |
| <input checked="" type="radio"/> погрКонтSam1A  | <input checked="" type="radio"/> срВрПогрКонтSam1A | <input checked="" type="radio"/> погрКонтSam2A  | <input checked="" type="radio"/> срВрПогрКонтSam2A |
| <input checked="" type="radio"/> погрКонтSam1B  | <input checked="" type="radio"/> срВрПогрКонтSam1B | <input checked="" type="radio"/> погрКонтSam2B  | <input checked="" type="radio"/> срВрПогрКонтSam2B |
| <input checked="" type="radio"/> выгрКонтSam1A  | <input checked="" type="radio"/> срВрВыгрКонтSam1A | <input checked="" type="radio"/> выгрКонтSam2A  | <input checked="" type="radio"/> срВрВыгрКонтSam2A |
| <input checked="" type="radio"/> выгрКонтSam1B  | <input checked="" type="radio"/> срВрВыгрКонтSam1B | <input checked="" type="radio"/> выгрКонтSam2B  | <input checked="" type="radio"/> срВрВыгрКонтSam2B |
| <input checked="" type="radio"/> срВрПолётA12   |                                                    | <input checked="" type="radio"/> срВрПолётA21   |                                                    |
| <input checked="" type="radio"/> отклВрПолётA12 |                                                    | <input checked="" type="radio"/> отклВрПолётA21 |                                                    |
| <input checked="" type="radio"/> срВрПолётB12   |                                                    | <input checked="" type="radio"/> срВрПолётB21   |                                                    |
| <input checked="" type="radio"/> отклВрПолётB12 |                                                    | <input checked="" type="radio"/> отклВрПолётB21 |                                                    |

Рис. 9.2. Элементы Параметр для ввода исходных данных

Таблица 9.3.

| Имя                    | Тип    | Значение по умолчанию |
|------------------------|--------|-----------------------|
| <b>Самолёты типа А</b> |        |                       |
| колSamTypA             | int    | 2                     |
| грузПодSamA            | int    | 50                    |
| <b>Аэропорт 1</b>      |        |                       |
| минКонтПост1           | int    | 1                     |
| максКонтПост1          | int    | 15                    |
| срВрПостКонт1          | double | 0,5                   |
| погрКонтSam1A          | int    | 2                     |
| погрКонтSam1B          | int    | 2                     |
| выгрКонтSam1A          | int    | 3                     |
| выгрКонтSam1B          | int    | 3                     |

|                   |        |     |
|-------------------|--------|-----|
| срВрПогрКонтСам1А | double | 0,1 |
| срВрПогрКонтСам1Б | double | 0,1 |
| срВрВыгрКонтСам1А | double | 0,2 |
| срВрВыгрКонтСам1Б | double | 0,2 |
| срВрПолётаA12     | double | 3,4 |
| отклВрПолётаA12   | double | 0,5 |
| срВрПолётаB12     | double | 3,2 |
| отклВрПолётаB12   | double | 0,5 |

### Самолёты типа Б

|             |     |     |
|-------------|-----|-----|
| колСамТипБ  | int | 1   |
| грузПодСамБ | int | 100 |

### Аэропорт 2

|                   |        |     |
|-------------------|--------|-----|
| минКонтПост2      | int    | 1   |
| максКонтПост2     | int    | 22  |
| срВрПостКонт2     | double | 0,4 |
| погрКонтСам2А     | int    | 2   |
| погрКонтСам2Б     | int    | 2   |
| выгрКонтСам2А     | int    | 2   |
| выгрКонтСам2Б     | int    | 2   |
| срВрПогрКонтСам2А | double | 0,2 |
| срВрПогрКонтСам2Б | double | 0,1 |
| срВрВыгрКонтСам2А | double | 0,2 |
| срВрВыгрКонтСам2Б | double | 0,2 |
| срВрПолётаA12     | double | 3,6 |
| отклВрПолётаA12   | double | 0,6 |
| срВрПолётаB12     | double | 4,1 |
| отклВрПолётаB12   | double | 0,8 |

## 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. Перейдите на страницу Дополнительные панели Свойства. Введите в поля X: 25, Y: 2610, Ширина: 510, Высота: 450.
3. Перетащите элемент text и в поле Текст: введите Результаты моделирования.
4. Из библиотеки Основная перетащите элементы Простая переменная и поместите их так, как на [рис. 9.3](#). Для всех переменных сохраните предлагаемый тип double, значение по умолчанию - 0.

| Результаты моделирования   |                |  |                |                |  |                     |
|----------------------------|----------------|--|----------------|----------------|--|---------------------|
| Аэропорт 1                 |                |  | Аэропорт 2     |                |  | Итоговые показатели |
| ✓ коэфПогр1A               | ✓ коэфДост21   |  | ✓ коэфПогр2A   | ✓ коэфДост12   |  | ✓ коэфДост          |
| ✓ коэфПогр1B               | ✓ коэфИспСан1A |  | ✓ коэфПогр2B   | ✓ коэфИспСан2A |  | ✓ коэфИспСанA       |
| ✓ коэфРазгр1A              | ✓ коэфИспСан1B |  | ✓ коэфРазгр2A  | ✓ коэфИспСан2B |  | ✓ коэфИспСанB       |
| ✓ коэфРазгр1B              |                |  | ✓ коэфРазгр2B  |                |  | ✓ коэфИспСан        |
| ✓ коэфПолётA12             |                |  | ✓ коэфПолётA21 |                |  |                     |
| ✓ коэфПолётB12             |                |  | ✓ коэфПолётB21 |                |  | event               |
| Вспомогательные переменные |                |  |                |                |  |                     |
| ✓ a                        | ✓ погрКонтA1   |  | ✓ d            | ✓ погрКонтA2   |  |                     |
| ✓ всегоПостK1              | ✓ погрКонтB1   |  | ✓ всегоПостK2  | ✓ погрКонтB2   |  |                     |
| ✓ текНалK1                 | ✓ выгрКонтA1   |  | ✓ текНалK2     | ✓ выгрКонтA2   |  |                     |
| ✓ достКА21                 | ✓ выгрКонтB1   |  | ✓ достКА12     | ✓ выгрКонтB2   |  |                     |
| ✓ достKB21                 | ✓ достK21      |  | ✓ достKB12     | ✓ достK12      |  |                     |

Рис. 9.3. Элементы Простая переменная для вывода результатов моделирования

- Перетащите ещё элементы `text` и введите пояснения к результатам моделирования как на [рис. 9.3](#).
- Перетащите элемент `text` и в поле Текст: введите Вспомогательные переменные.
- Перетащите элементы Простая переменная и поместите их согласно [рис. 9.3](#).
- Для всех переменных оставьте предлагаемый тип `double`, значение по умолчанию - 0.
- 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 и аэропорта 2.

## Имитация функционирования аэропорта 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. Перейдите на страницу Дополнительные 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 Основной библиотеки по два объекта source, enter, queue, hold и один объект exit. Поместите и соедините их так, как на [рис. 9.4](#).

При построении модели вам придётся воспользоваться Java-кодом, в котором потребуются дополнительные поля заявок. Для этого вы должны создать нестандартный класс заявки с дополнительными полями для записи и хранения параметров, о которых упоминалось ранее (см. п. 9.1.4).

Создайте класс заявок ТранспСредство.

1. Выделите объект source.
2. В панели Проект щелкните 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. Щелкните кнопку **Далее**.
5. Появится вторая страница Мастера создания Java класса. Добавьте поля Java класса, показанные на [рис. 9.5](#).
6. Оставьте выбранными флажки **Создать конструктор** и **Создать метод `toString ()`**.
7. Щелкните **Готово**. Закройте редактор кода.

Аналогичным образом создайте классы 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.

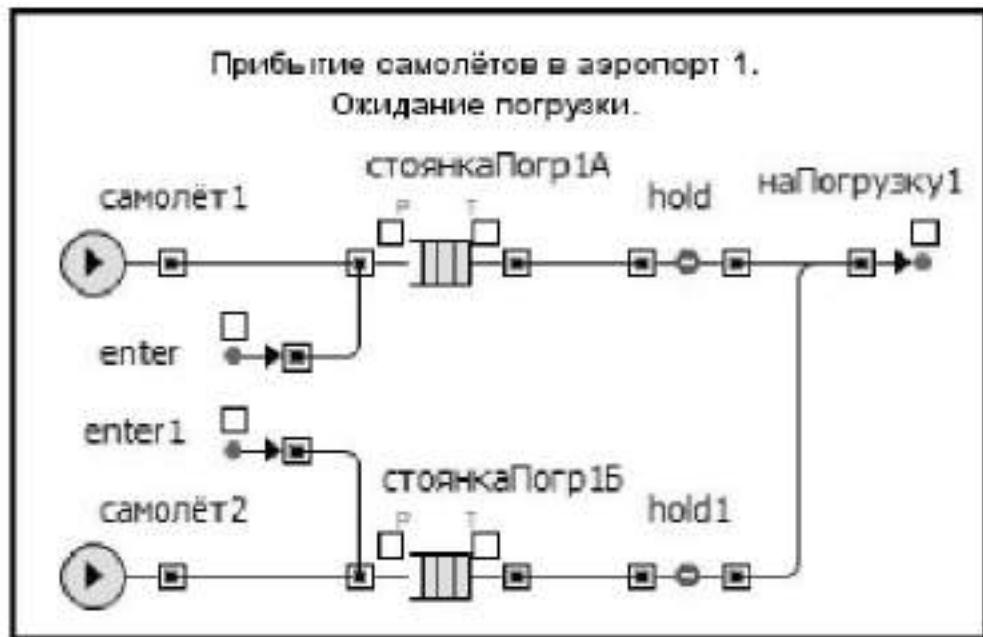


Рис. 9.4. Сегмент Прибытие самолётов в аэропорт 1

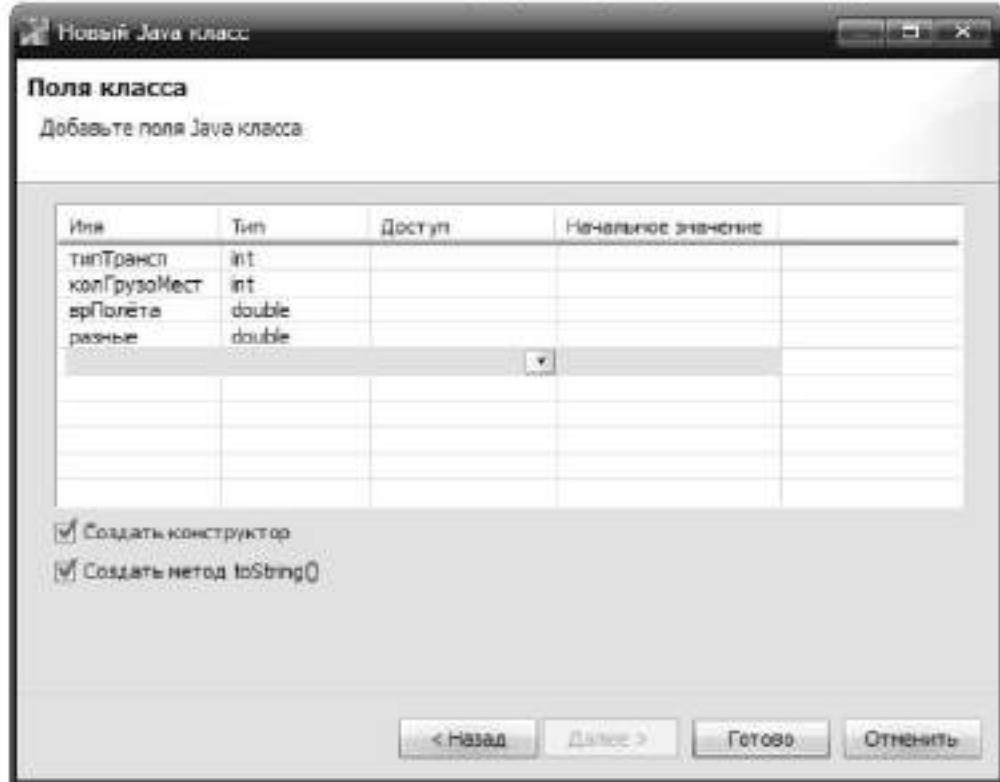


Рис. 9.5. Поля нового Java класса ТранспСредство

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                 |
|---------------------------|--------------------------|
| <b>source</b>             |                          |
| Имя:                      | самолёт1                 |
| Класс заявки:             | ТранспСредство           |
| Заявки прибывают согласно | Времени между прибытиями |
| Время между прибытиями    | 0                        |

|                                            |                                                      |
|--------------------------------------------|------------------------------------------------------|
| Количество заявок, прибывающих за один раз | колСамТипА                                           |
| Ограниченнное количество прибытий          | Установить флагок                                    |
| Максимальное количество прибытий           | 1                                                    |
| Новая заявка                               | <pre>new ТранспСредство() entity.типТрансп=1;</pre>  |
| Действие при выходе                        | <pre>entity.колГрузоМест- грузПодСамА; source1</pre> |
| Имя:                                       | самолёт2                                             |
| Класс заявки:                              | ТранспСредство                                       |
| Заявки прибывают согласно                  | Времени между прибытиями                             |
| Время между прибытиями                     | 0                                                    |
| Количество заявок, прибывающих за один раз | колСамТипА                                           |
| Ограниченнное количество прибытий          | Установить флагок                                    |
| Максимальное количество прибытий           | 1                                                    |
| Новая заявка                               | <pre>new ТранспСредство() entity.типТрансп=2;</pre>  |
| Действие при выходе                        | <pre>entity.колГрузоМест= грузПодСамБ; queue</pre>   |
| Имя:                                       | стоянкаПогр1А                                        |
| Класс заявки:                              | ТранспСредство                                       |
| Вместимость                                | колСамТипА                                           |
| Включить сбор статистики                   | Установить флагок                                    |
| Имя:                                       | <b>queue1</b>                                        |
| Имя:                                       | стоянкаПогр1Б                                        |

|                          |                                                                                   |
|--------------------------|-----------------------------------------------------------------------------------|
| Класс заявки:            | ТранспСредство                                                                    |
| Вместимость              | колСамТипБ                                                                        |
| Включить сбор статистики | Установить флагок<br><b>hold</b>                                                  |
| Класс заявки:            | ТранспСредство                                                                    |
| Изначально заблокирован  | Установить флагок<br><b>hold1</b>                                                 |
| Класс заявки:            | ТранспСредство                                                                    |
| Изначально заблокирован  | Установить флагок<br><b>exit</b>                                                  |
| Имя:                     | наПогрузку1                                                                       |
| Класс заявки:            | <pre>if (entity.типТрансп==1) {hold.setBlocked(true); enter2.take(entity);}</pre> |
| Действие при выходе      | <pre>else {hold1.setBlocked(true); enter3.take(entity);}  enter</pre>             |
| Класс заявки:            | ТранспСредство                                                                    |
|                          | <b>enter1</b>                                                                     |
| Класс 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.

## 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. Перейдите на страницу Дополнительные 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 Основной библиотеки по одному объекту

source, selectOutput5 (имя selectOutput) и sink.  
Поместите и соедините их так, как на рис. 9.6.

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 0 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 0.

### Поступление и учёт контейнеров в аэропорту 1

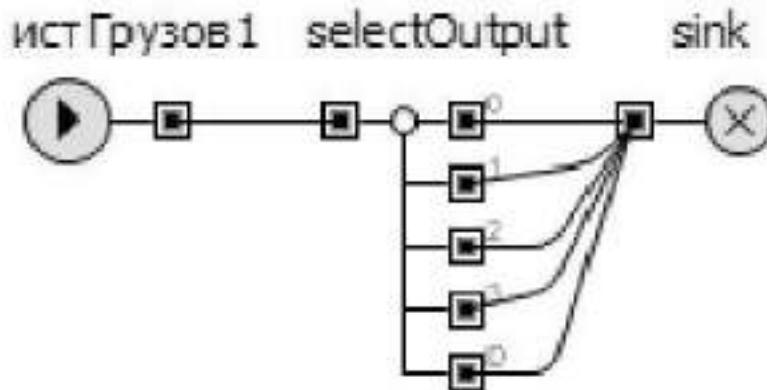


Рис. 9.6. Сегмент Поступление и учёт контейнеров в аэропорту 1

Таблица 9.5.

| Свойство                                      | Значения                                                                                                            |
|-----------------------------------------------|---------------------------------------------------------------------------------------------------------------------|
| <b>source</b>                                 |                                                                                                                     |
| Имя:                                          | истГрузов1                                                                                                          |
| Класс заявки:                                 | ГрузАэропорт1                                                                                                       |
| Заявки прибывают согласно                     | Времени между прибытиями                                                                                            |
| Время между прибытиями                        | exponential(1/<br>срВрПостКонт1)                                                                                    |
| Количество заявок, прибывающих<br>за один раз | 1                                                                                                                   |
| Новая заявка                                  | new ГрузАэропорт1()<br>a=(uniform_<br>discr(минКонтПост1,<br>максКонтПост1));<br>всегоПостК1+=a;<br>текНалКонт1+=a; |
| <b>selectOutput</b>                           |                                                                                                                     |
| Класс заявки:                                 | ГрузАэропорт1                                                                                                       |
| Использовать:                                 | Условия                                                                                                             |
| Условие 0                                     | текНалК1<грузПодСамA                                                                                                |
| Условие 1                                     | стоянкаПогр1A.size()>0 &&<br>погрзка1A.size()==0                                                                    |
| Действие при выходе 1                         | текНалК1-=грузПодСамA;<br>hold.setBlocked(false);                                                                   |
| Условие 2                                     | текНалК1<грузПодСамB                                                                                                |
| Условие 3                                     |                                                                                                                     |
| Действие при выходе 3                         |                                                                                                                     |
| <b>sink</b>                                   |                                                                                                                     |
| Класс заявки:                                 | ГрузАэропорт1                                                                                                       |

После выполнения условия 0 проверяется условие 1: наличие самолётов типа А на стоянке ожидания погрузки и незанятость пунктов погрузки.

При выполнении условия 1 формируется команда на разблокировывание элемента `hold` (см. п. 9.1.8.2) и самолёт типа А отправляется на погрузку.

Если условие 1 не выполняется, например, при отсутствии свободного самолёта типа А, проверяется условие 2.

Если условие 2 не выполняется, то есть недостаточно контейнеров для полной загрузки самолёта типа Б, то заявка-партия уничтожается.

Если условие 2 выполняется (поступило число контейнеров, достаточное для полной загрузки самолёта типа Б), проверяется условие 3: наличие самолётов типа Б на стоянке ожидания погрузки и незанятость пунктов погрузки.

Если условие 3 не выполняется, заявка-партия уничтожается.

При выполнении условия 3 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 класса ГрузАэропорт1 выводятся из модели.

### 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. Перейдите на страницу Дополнительные панели Свойства. Введите в поля X: 350, Y: 50, Ширина: 460, Высота: 200.

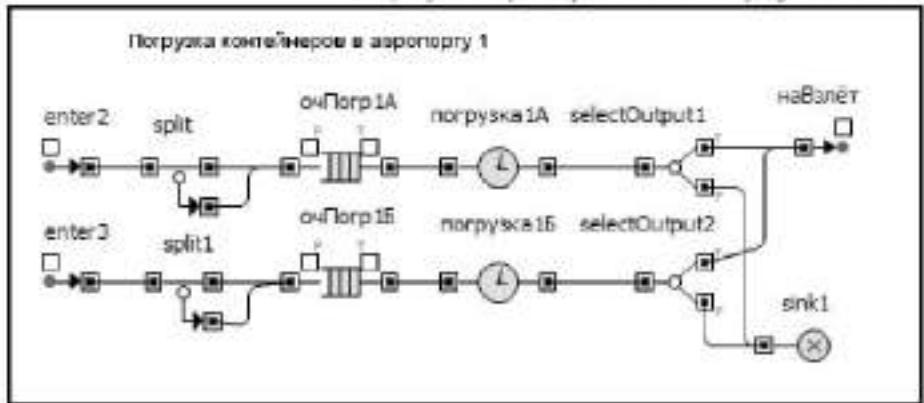


Рис. 9.7. Сегмент Погрузка контейнеров в аэропорту 1

- Перетащите элемент **text** и в поле Текст: введите Погрузка контейнеров в аэропорту 1.
- Перетащите из Основной библиотеки по два объекта **enter**, **split**, **queue**, **delay**, **selectOutput** и по одному объекту **exit** и **sink**. Поместите и соедините их так, как на рис. 9.7.
- Установите свойства объектов согласно табл. 9.6.

Таблица 9.6.

| Свойство             | Значения                                                          |
|----------------------|-------------------------------------------------------------------|
| <b>enter2</b>        |                                                                   |
| Класс заявки:        | ТранспСредство                                                    |
| <b>enter3</b>        |                                                                   |
| Класс заявки:        | ТранспСредство                                                    |
| <b>split</b>         |                                                                   |
| Классы заявок:       |                                                                   |
| Оригинал, Копия      | ТранспСредство,<br>ТранспСредство                                 |
| Количество копий     | entity.колГрузоМест-1                                             |
| Новая заявка (копия) | new ТранспСредство ()<br>entity.типТрансп=<br>original.типТрансп; |

**Действие при выходе копии**

```
entity.колГрузоМест=
original.колГрузоМест;
entity.tPolet=
original.tPolet;
entity.разные =original.разные;
split1
```

|                                  |                                                                                                                                                                             |
|----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Классы заявок:</b>            |                                                                                                                                                                             |
| <b>Оригинал, Копия</b>           | ТранспСредство,<br>ТранспСредство                                                                                                                                           |
| <b>Количество копий</b>          | entity.колГрузоМест-1                                                                                                                                                       |
| <b>Новая заявка (копия)</b>      | new ТранспСредство () entity.типТрансп= original.типТрансп; entity.колГрузоМест= original.колГрузоМест; entity.врПолёта= original.врПолёта; entity.разные =original.разные; |
| <b>Действие при выходе копии</b> | entity.разные =срВрПогрКонтСам1А;                                                                                                                                           |
| <b>Имя:</b>                      | очПогр1А                                                                                                                                                                    |
| <b>Класс заявки:</b>             | ТранспСредство                                                                                                                                                              |
| <b>Максимальная вместимость</b>  | Установить флагок                                                                                                                                                           |
| <b>Действие при выходе</b>       | entity.разные = срВрПогрКонтСам1А;                                                                                                                                          |
| <b>Включить сбор статистики</b>  | Установить флагок                                                                                                                                                           |
| <b>queue1</b>                    |                                                                                                                                                                             |
| <b>Имя:</b>                      | очПогр1Б                                                                                                                                                                    |
| <b>Класс заявки:</b>             | ТранспСредство                                                                                                                                                              |
| <b>Максимальная вместимость</b>  | Установить флагок                                                                                                                                                           |
| <b>Действие при выходе</b>       | entity.разные = срВрПогрКонтСам1Б;                                                                                                                                          |

|                               |                                                                                 |
|-------------------------------|---------------------------------------------------------------------------------|
| Включить сбор статистики      | Установить флажок                                                               |
| <b>delay</b>                  |                                                                                 |
| Имя:                          | погрузкаA                                                                       |
| Класс заявки:                 | ТранспСредство                                                                  |
| Задержка задаётся             | Явно                                                                            |
| Время задержки                | exponential<br>(1/entity.разные)                                                |
| Вместимость                   | погрКонтСам1A                                                                   |
| Действие при выходе           | погрКонтA1++;                                                                   |
| Включить сбор статистики      | Установить флажок                                                               |
| <b>delay1</b>                 |                                                                                 |
| Имя:                          | погрузкаB                                                                       |
| Класс заявки:                 | ТранспСредство                                                                  |
| Задержка задаётся             | Явно                                                                            |
| Время задержки                | exponential<br>(1/entity.разные)                                                |
| Вместимость                   | погрКонтСам1B                                                                   |
| Действие при выходе           | погрКонтB1++;                                                                   |
| Включить сбор статистики      | Установить флажок                                                               |
| <b>selectOutput1</b>          |                                                                                 |
| Класс заявки:                 | ТранспСредство                                                                  |
| Выход true выбирается         | При выполнении условия                                                          |
| Условие                       | entity.колГрузоМест<br>==погрКонтA1                                             |
| Действие при выходе<br>(true) | entity.брПолёта=<br>normal(отклBrПолётаA12,<br>срBrПолётаA12);<br>погрКонтA1=0; |
| <b>selectOutput2</b>          |                                                                                 |

|                               |                                                                                                                                   |
|-------------------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| Класс заявки:                 | ТранспСредство                                                                                                                    |
| Выход true выбирается         | При выполнении условия                                                                                                            |
| Условие                       | <pre>entity.колГрузомест ==погрКонтБ1 entity.срПолёта=</pre>                                                                      |
| Действие при выходе<br>(true) | <pre>normal(отклВрПолётаB12, срВрПолётаB12); погрКонтB1=0;</pre> <p style="text-align: center;"><b>exit</b></p>                   |
| Имя:                          | навзлёт12                                                                                                                         |
| Класс заявки:                 | ТранспСредство                                                                                                                    |
| Действие при выходе           | <pre>if (entity.типТрансп==1) enter4.take(entity); else enter5.take(entity);</pre> <p style="text-align: center;"><b>sink</b></p> |
| Класс 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 `погрузка1A`, имитирующему непосредственно погрузку контейнеров в самолёт типа А в аэропорту 1. При покидании очереди выполняется код `entity.разные=срВрПогрКонтСам1A`, записывающий в поле с именем `разные` заявки-контейнера среднее время погрузки одного контейнера.

После объекта `погрузкаA`, на выходе которого ведётся счёт погруженных контейнеров (`погрКонтA1++`), заявки-контейнеры входят в объект `selectOutput1`.

Этот объект проверяет условие (`entity.колГрузомест == погрКонтB1`): 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`.

### 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. Перейдите на страницу Дополнительные 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 Основной библиотеки по два объекта `enter`, `delay` и один объект `exit`. Поместите и соедините их так, как на [рис. 9.8](#).
5. Установите свойства объектов согласно [табл. 9.7](#).

Предположим, что поступила заявка-самолёт типа А в объект `enter4` и далее в объект с именем `полётA12` (`delay`). Идентификатор

полётA12 означает, что объект имитирует непосредственно полёт из аэропорта 1 в аэропорт 2. В аэропорту 2 заявка-самолёт входит в сегмент ожидания разгрузки.

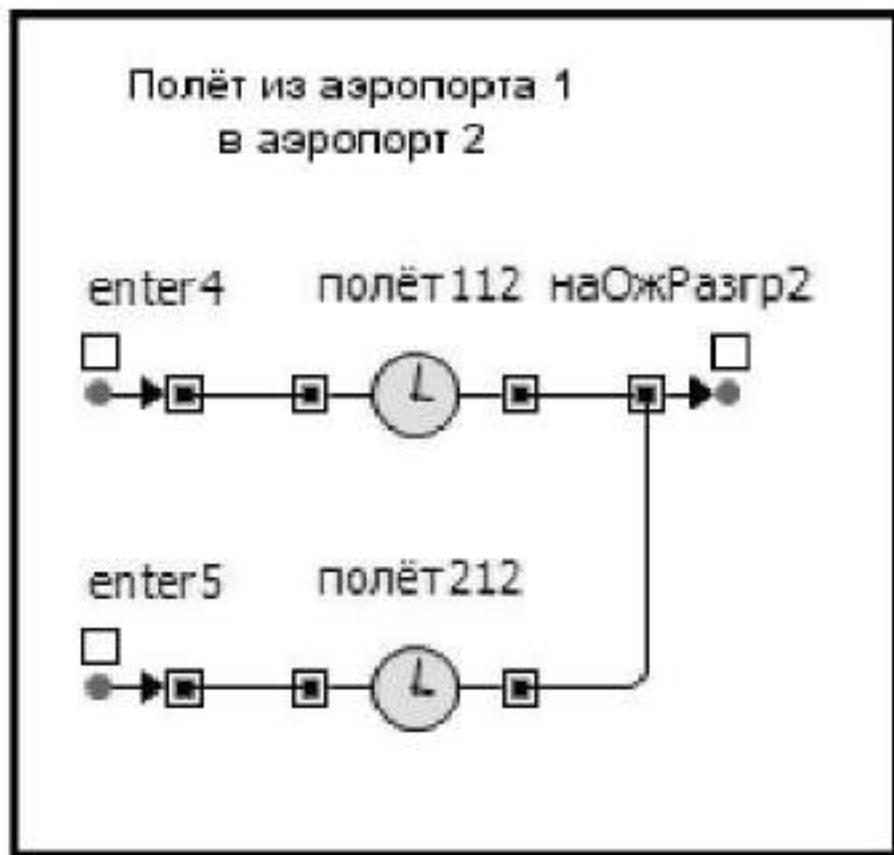


Рис. 9.8. Сегмент Полёт из аэропорта 1 в аэропорт 2

Таблица 9.7.

| Свойство      | Значения       |
|---------------|----------------|
| <b>enter4</b> |                |
| Класс заявки: | ТранспСредство |
| <b>enter5</b> |                |
| Класс заявки: | ТранспСредство |
| <b>delay</b>  |                |
| Имя:          | полётA12       |

|                          |                                                                                      |
|--------------------------|--------------------------------------------------------------------------------------|
| Класс заявки:            | ТранспСредство                                                                       |
| Задержка задаётся        | Явно                                                                                 |
| Время задержки           | entity.врПолёта                                                                      |
| Вместимость              | колСамТипА                                                                           |
| Включить сбор статистики | Установить флагок                                                                    |
| <b>delay1</b>            |                                                                                      |
| Имя:                     | ПолётB12                                                                             |
| Класс заявки:            | ТранспСредство                                                                       |
| Задержка задаётся        | Явно                                                                                 |
| Время задержки           | entity.врПолёта                                                                      |
| Вместимость              | колСамТипB                                                                           |
| Включить сбор статистики | Установить флагок                                                                    |
| <b>exit</b>              |                                                                                      |
| Действие при выходе      | <pre>if (entity.типТрансп==1) enter10.take(entity); else enter11.take(entity);</pre> |

## 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. Перейдите на страницу Дополнительные 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 Основной библиотеки по два объекта

`enter`, `queue`, `hold` и один объект `exit`. Поместите и соедините их так, как на рис. 9.9.

##### 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 (`разгрузка1A.size() == 0`), разблокированывается объект `hold2` и заявка-самолёт входит в объект `наРазгрузку1` и далее в сегмент имитации разгрузки.

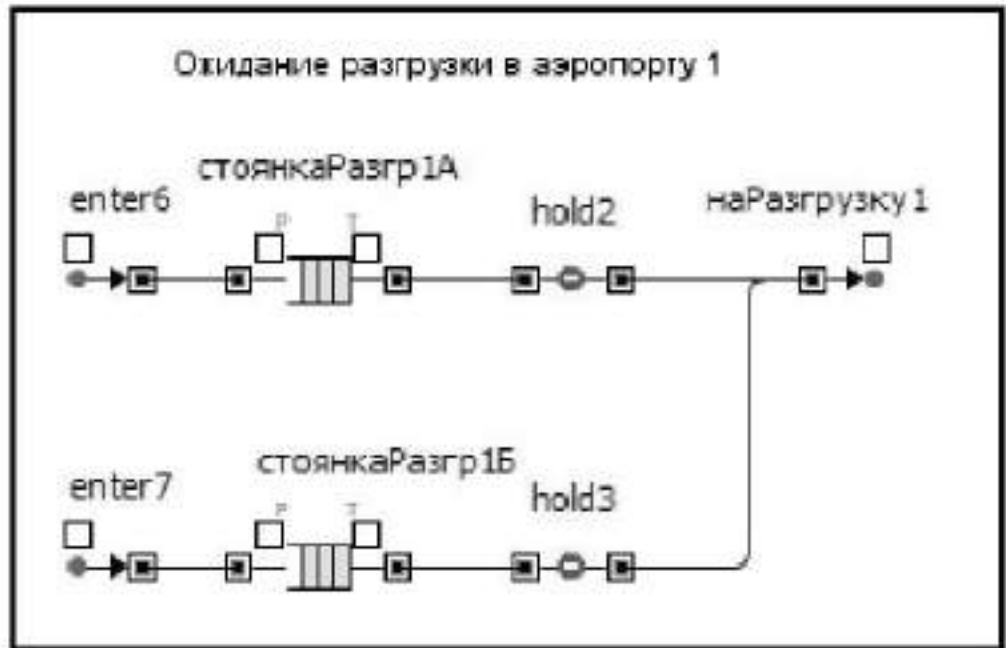


Рис. 9.9. Сегмент Ожидание разгрузки в аэропорту 1

Таблица 9.8.

| Свойство           | Значения                                                               |
|--------------------|------------------------------------------------------------------------|
|                    | <b>enter6</b>                                                          |
| Класс заявки:      | ТранспСредство                                                         |
| Действие при входе | <code>if (разгрузка1A.size() == 0)<br/>hold2.setBlocked(false);</code> |
|                    | <b>enter7</b>                                                          |
| Класс заявки:      | ТранспСредство                                                         |

|                          |                                                                                                                                         |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| Действие при входе       | <pre>if (разгрузка1B.size() == 0) hold3.setBlocked(false);</pre>                                                                        |
| <b>queue</b>             |                                                                                                                                         |
| Имя:                     | стоянкаРазгр1А                                                                                                                          |
| Класс заявки:            | ТранспСредство                                                                                                                          |
| Вместимость              | колСамТипА                                                                                                                              |
| Включить сбор статистики | Установить флагок                                                                                                                       |
| <b>queue1</b>            |                                                                                                                                         |
| Имя:                     | стоянкаРазгр1Б                                                                                                                          |
| Класс заявки:            | ТранспСредство                                                                                                                          |
| Вместимость              | колСамТипБ                                                                                                                              |
| Включить сбор статистики | Установить флагок                                                                                                                       |
| <b>hold2</b>             |                                                                                                                                         |
| Класс заявки:            | ТранспСредство                                                                                                                          |
| Изначально заблокирован  | Установить флагок                                                                                                                       |
| <b>hold3</b>             |                                                                                                                                         |
| Класс заявки:            | ТранспСредство                                                                                                                          |
| Изначально заблокирован  | Установить флагок                                                                                                                       |
| <b>exit</b>              |                                                                                                                                         |
| Имя:                     | наРазгрузку1                                                                                                                            |
| Действие при выходе      | <pre>if (entity.типТрансп==1) (hold2.setBlocked(true); enter8.take(entity);); else (hold3.setBlocked(true); enter9.take(entity);)</pre> |

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.

## 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. Перейдите на страницу Дополнительные 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 из Основной библиотеки по два объекта `enter`, `split`, `queue`, `delay`, `selectOutput` и по одному объекту `exit` и `sink`. Поместите и соедините их так, как на [рис. 9.10](#).
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.

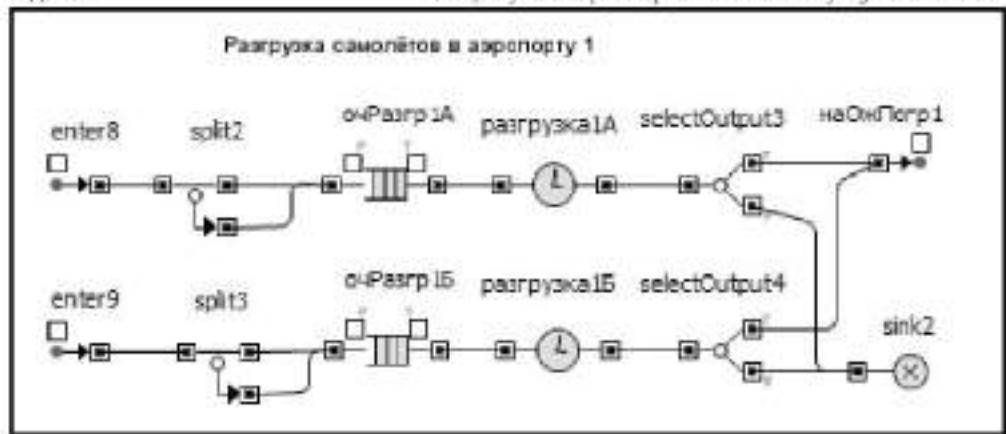


Рис. 9.10. Сегмент Разгрузка самолётов в аэропорту 1

Таблица 9.9.

| Свойство                     | Значения                                                                                                           |
|------------------------------|--------------------------------------------------------------------------------------------------------------------|
| <b>enter8</b>                |                                                                                                                    |
| Класс заявки:                | ТранспСредство                                                                                                     |
| <b>enter9</b>                |                                                                                                                    |
| Класс заявки:                | ТранспСредство                                                                                                     |
| <b>split2</b>                |                                                                                                                    |
| Классы заявок:               |                                                                                                                    |
| Оригинал, Копия              | ТранспСредство,<br>ТранспСредство                                                                                  |
| Количество копий             | entity.колГрузоМест-1                                                                                              |
| Новая заявка (копия)         | new ТранспСредство()<br>entity.типТрансп=<br>original.типТрансп;<br>entity.колГрузоМест=<br>original.колГрузоМест; |
| Действие при выходе<br>копии | entity.врПолёта=<br>original.врПолёта;<br>entity.разные=<br>original.разные;                                       |
|                              | <b>split3</b>                                                                                                      |

| <b>Классы заявок:</b>     |                                                                                                                                                                                                  |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Оригинал, Копия           | ТранспСредство,<br>ТранспСредство                                                                                                                                                                |
| Количество копий          | entity.колГрузоМест-1                                                                                                                                                                            |
| Новая заявка (копия)      | new ТранспСредство ()<br>entity.типТрансп=<br>original.типТрансп;<br>entity.колГрузоМест=<br>original.колГрузоМест;<br>entity.врПолёта=<br>original.врПолёта;<br>entity.разные =original.разные; |
| Действие при выходе копии | <b>queue</b>                                                                                                                                                                                     |
| Имя:                      | очРазгр1А                                                                                                                                                                                        |
| Класс заявки:             | ТранспСредство                                                                                                                                                                                   |
| Максимальная вместимость  | Установить флагок                                                                                                                                                                                |
| Действие при выходе       | entity.разные=<br>срВрВыгрКонтСам1А                                                                                                                                                              |
| Включить сбор статистики  | Установить флагок                                                                                                                                                                                |
| <b>queue1</b>             |                                                                                                                                                                                                  |
| Имя:                      | очРазгр1Б                                                                                                                                                                                        |
| Класс заявки:             | ТранспСредство                                                                                                                                                                                   |
| Максимальная вместимость  | Установить флагок                                                                                                                                                                                |
| Действие при выходе       | entity.разные=<br>срВрВыгрКонтСам1А                                                                                                                                                              |
| Включить сбор статистики  | Установить флагок                                                                                                                                                                                |
| <b>delay</b>              |                                                                                                                                                                                                  |
| Имя:                      | разгрузка1А                                                                                                                                                                                      |
| Класс заявки:             | ТранспСредство                                                                                                                                                                                   |

|                                              |                                                                               |
|----------------------------------------------|-------------------------------------------------------------------------------|
| Задержка задаётся                            | <b>Явно</b>                                                                   |
| Время задержки                               | <code>exponential<br/>(1/entity.разные)</code>                                |
| Вместимость                                  | <code>выгрКонтСам1А</code>                                                    |
| Действие при выходе                          | <code>выгрКонтА1++;</code>                                                    |
| Включить сбор статистики                     | Установить флагок                                                             |
| <b>delay1</b>                                |                                                                               |
| Имя:                                         | <code>разгрузка1Б</code>                                                      |
| Класс заявки:                                | ТранспСредство                                                                |
| Задержка задаётся                            | <b>Явно</b>                                                                   |
| Время задержки                               | <code>exponential<br/>(1/entity.разные)</code>                                |
| Вместимость                                  | <code>выгрКонтСам1Б</code>                                                    |
| Действие при выходе                          | <code>выгрКонтБ1++;</code>                                                    |
| Включить сбор статистики                     | Установить флагок                                                             |
| <b>selectOutput3</b>                         |                                                                               |
| Класс заявки:                                | ТранспСредство                                                                |
| Выход <code>true</code> выбирается           | При выполнении условия                                                        |
| Условие                                      | <code>entity.колГрузоМест==<br/>выгрКонтА1</code>                             |
|                                              | <code>выгрКонтА1=0;</code>                                                    |
| Действие при выходе<br>( <code>true</code> ) | <code>достКА21+=<br/>entity.колГрузоМест;<br/>hold2.setBlocked(false);</code> |
| <b>selectOutput4</b>                         |                                                                               |
| Класс заявки:                                | ТранспСредство                                                                |
| Выход <code>true</code> выбирается           | При выполнении условия                                                        |
| Условие                                      | <code>entity.колГрузоМест==<br/>выгрКонтБ1</code>                             |

|                               |                                                                                              |
|-------------------------------|----------------------------------------------------------------------------------------------|
|                               | выгрКонтБ1=0;                                                                                |
| Действие при выходе<br>(true) | достК21+=<br>entity.колГрузоМест;<br>hold3.setBlocked(false);<br><b>exit</b>                 |
| Имя:                          | наОжПогр1                                                                                    |
| Класс заявки:                 | ТранспСредство                                                                               |
|                               | достК21+=<br>entity.колГрузоМест;                                                            |
| Действие при выходе           | if (entity.типТрансп==1)<br>enter.take(entity);<br>else enter1.take(entity);<br><b>sink2</b> |
| Класс заявки:                 | ТранспСредство                                                                               |

Заявки-контейнеры занимают очередь к объекту разгрузка1A, имитирующему непосредственно выгрузку контейнеров из самолёта типа А в аэропорту 1. При покидании очереди выполняется код entity.разные=срВрВыгрКонтСам1A, записывающий в поле с именем разные заявки-контейнера среднее время выгрузки одного контейнера.

После объекта разгрузка1A, 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, разблокированывается 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`.

## Имитация функционирования аэропорта 2

Сегменты, имитирующие функционирование аэропорта 2, построены в основном также, как и сегменты аэропорта 1, поэтому будут отмечены лишь некоторые особенности.

### 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. Перейдите на страницу Дополнительные 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. 9.11.

Установите свойства объектов согласно табл. 9.10.

## Поступление и учёт контейнеров в аэропорту 2

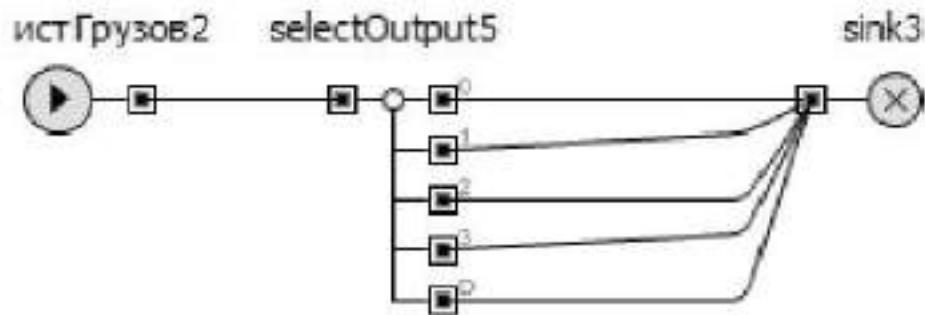


Рис. 9.11. Сегмент Поступление и учёт контейнеров в аэропорту 2

Таблица 9.10.

| Свойство                                      | Значения                                                                           |
|-----------------------------------------------|------------------------------------------------------------------------------------|
|                                               | <b>source</b>                                                                      |
| Имя:                                          | истГрузов2                                                                         |
| Класс заявки:                                 | ГрузАэропорт2                                                                      |
| Заявки прибывают согласно                     | Времени между прибытиями                                                           |
| Время между прибытиями                        | Exponential<br>(1/срВрПостКонт2)                                                   |
| Количество заявок,<br>прибывающих за один раз |                                                                                    |
| Новая заявка                                  | 1                                                                                  |
|                                               | <code>new ГрузАэропорт2()<br/>d=</code>                                            |
| Действие при выходе                           | <code>(uniform_discr(минКонтПост2,<br/>максКонтПост2));<br/>всегоПостК2+=d;</code> |

**selectOutput5**

|                       |                                                                     |
|-----------------------|---------------------------------------------------------------------|
| Класс заявки:         | ГрузАэропорт2                                                       |
| Использовать:         | Условия                                                             |
| Условие 0             | техНалК2 < грузПодСамА                                              |
| Условие 1             | стоянкаПогр2А.size() > 0 &&<br>погрузка2А.size() == 0               |
| Действие при выходе 1 | техНалК2 -= грузПодСамА;<br>hold6.setBlocked(false);                |
| Условие 2             | техНалК2 < грузПодСамБ                                              |
| Условие 3             | стоянкаПогр2Б.size() > 0 &&<br>погрузка2Б.size() == 0               |
| Действие при выходе 3 | техНалК2 -= грузПодСамБ;<br>hold7.setBlocked(false);<br><b>sink</b> |
| Класс 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.

### 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. Перейдите на страницу Дополнительные 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 Основной библиотеки по два объекта `enter`, `queue`, `hold` и один объект `exit`. Поместите и соедините их так, как на рис. 9.12.
5. Установите свойства объектов согласно табл. 9.11

### Ожидание разгрузки в аэропорту 2

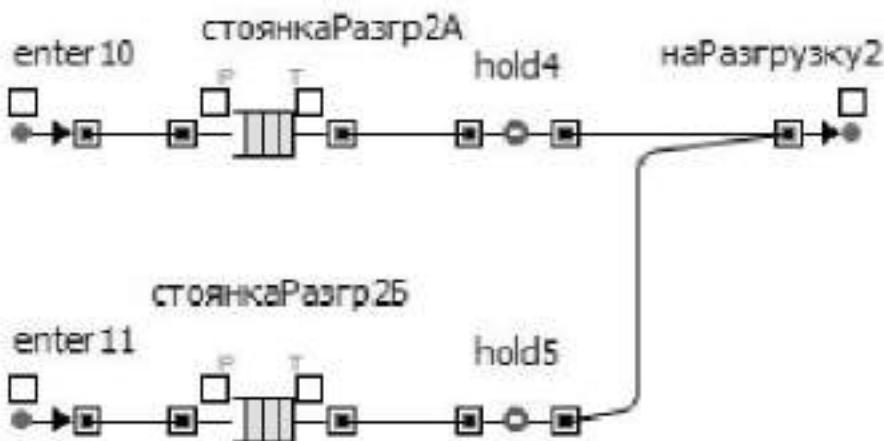


Рис. 9.12. Сегмент Ожидание разгрузки в аэропорту 2

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 разблокируется объект `hold5` и заявка-

самолёт типа Б входит в объект на Разгрузку2 и далее в сегмент имитации разгрузки аэропорта 2.

Таблица 9.11.

| Свойство                 | Значения                                                                                              |
|--------------------------|-------------------------------------------------------------------------------------------------------|
| <b>enter10</b>           |                                                                                                       |
| Класс заявки:            | ТранспСредство                                                                                        |
| Действие при входе       | <pre>if (стоянкаРазгр2A.size() == 0 &amp;&amp;разгрузка2A.size() == 0) hold4.setBlocked(false);</pre> |
| <b>enter11</b>           |                                                                                                       |
| Класс заявки:            | ТранспСредство                                                                                        |
| Действие при входе       | <pre>if (стоянкаРазгр2B.size() == 0 &amp;&amp;разгрузка2B.size() == 0) hold5.setBlocked(false);</pre> |
| <b>queue</b>             |                                                                                                       |
| Имя:                     | стоянкаРазгр2A                                                                                        |
| Класс заявки:            | ТранспСредство                                                                                        |
| Вместимость              | колСамТипА                                                                                            |
| Включить сбор статистики | Установить флагок                                                                                     |
| <b>queue1</b>            |                                                                                                       |
| Имя:                     | стоянкаРазгр2B                                                                                        |
| Класс заявки:            | ТранспСредство                                                                                        |
| Вместимость              | колСамТипБ                                                                                            |
| Включить сбор статистики | Установить флагок                                                                                     |
| <b>hold4</b>             |                                                                                                       |
| Класс заявки:            | ТранспСредство                                                                                        |
| Изначально заблокирован  | Установить флагок                                                                                     |
| <b>hold5</b>             |                                                                                                       |

|                         |                                                                                                                                                  |
|-------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| Класс заявки:           | ТранспСредство                                                                                                                                   |
| Изначально заблокирован | Установить флагок                                                                                                                                |
| <b>exit</b>             |                                                                                                                                                  |
| Имя:                    | наРазгрузку2                                                                                                                                     |
| Действие при выходе     | <pre> if (entity.типТрансп==1)     (hold4.setBlocked(true); enter12.take(entity);); else (hold5.setBlocked(true); enter13.take(entity);); </pre> |

### Разгрузка самолётов в аэропорту 2

Сегмент Разгрузка самолётов в аэропорту 2 предназначен для имитации разгрузки самолётов, прибывающих из аэропорта 1.

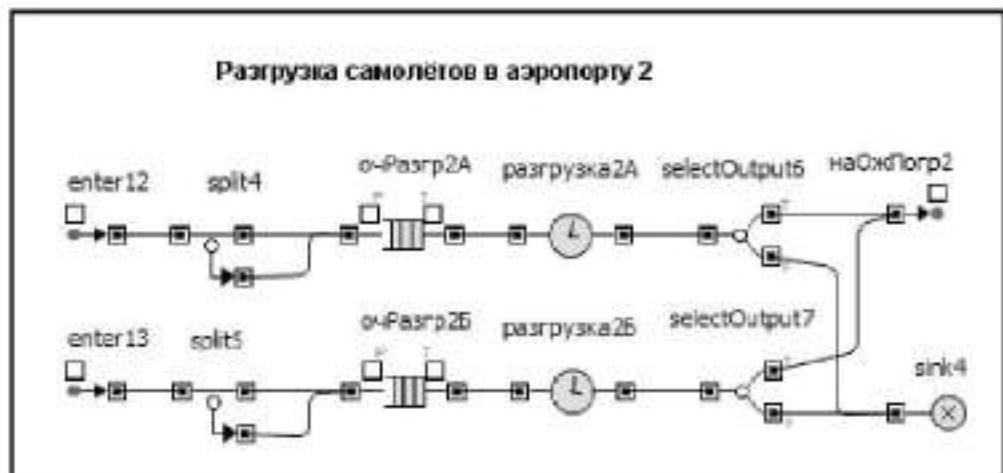


Рис. 9.13. Сегмент Разгрузка самолётов в аэропорту 2

Создайте сегмент.

1. Из палитры Презентация перетащите элемент Прямоугольник.
2. Перейдите на страницу Дополнительные 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 из Основной библиотеки по два объекта `enter`, `split`, `queue`, `delay`, `selectOutput` и по одному объекту `exit` и `sink`. Поместите и соедините их так, как на [рис. 9.13](#).
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 2, 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                           |
|------------------|------------------------------------|
|                  | <b>enter12</b>                     |
| Класс заявки:    | ТранспСредство                     |
|                  | <b>enter13</b>                     |
| Класс заявки:    | ТранспСредство                     |
|                  | <b>split4</b>                      |
| Классы заявок:   |                                    |
| Оригинал, Копия  | ТранспСредство,<br>ТранспСредство  |
| Количество копий | <code>entity.колГрузоМест-1</code> |

|                                  |                                                                                                        |
|----------------------------------|--------------------------------------------------------------------------------------------------------|
| <b>Новая заявка (копия)</b>      | <pre>new ТранспСредство() entity.типТрансп= original.типТрансп; entity.колГрузоМест= </pre>            |
| <b>Действие при выходе копии</b> | <pre>original.колГрузоМест; entity.врПолёта= original.врПолёта; entity.разные = original.разные;</pre> |
|                                  | <b>split5</b>                                                                                          |
| <b>Классы заявок:</b>            |                                                                                                        |
| <b>Оригинал, Копия</b>           | ТранспСредство,<br>ТранспСредство                                                                      |
| <b>Количество копий</b>          | entity.колГрузоМест-1                                                                                  |
| <b>Новая заявка (копия)</b>      | <pre>new ТранспСредство() entity.типТрансп= original.типТрансп; entity.колГрузоМест= </pre>            |
| <b>Действие при выходе копии</b> | <pre>original.колГрузоМест; entity.врПолёта= original.врПолёта; entity.разные= original.разные;</pre>  |
|                                  | <b>queue</b>                                                                                           |
| <b>Имя:</b>                      | очРазгр2A                                                                                              |
| <b>Класс заявки:</b>             | ТранспСредство                                                                                         |
| <b>Максимальная вместимость</b>  | Установить флажок                                                                                      |
| <b>Действие при выходе</b>       | <pre>entity.разные= срВрВыгрКонтСам2A;</pre>                                                           |
| <b>Включить сбор статистики</b>  | Установить флажок                                                                                      |
|                                  | <b>queue1</b>                                                                                          |
| <b>Имя:</b>                      | очРазгр2B                                                                                              |

|                          |                                     |
|--------------------------|-------------------------------------|
| Класс заявки:            | ТранспСредство                      |
| Максимальная вместимость | Установить флагок                   |
| Действие при выходе      | entity.разные=срВрВыгрКонтСам2Б;    |
| Включить сбор статистики | Установить флагок                   |
| <b>delay</b>             |                                     |
| Имя:                     | разгрузка2А                         |
| Класс заявки:            | ТранспСредство                      |
| Задержка задаётся        | Явно                                |
| Время задержки           | exponential<br>(1/entity.разные)    |
| Вместимость              | выгрКонтСам2А                       |
| Действие при выходе      | выгрКонтА2++;                       |
| Включить сбор статистики | Установить флагок                   |
| <b>delay1</b>            |                                     |
| Имя:                     | разгрузка2Б                         |
| Класс заявки:            | ТранспСредство                      |
| Задержка задаётся        | Явно                                |
| Время задержки           | exponential<br>(1/entity.разные)    |
| Вместимость              | выгрКонтСам2Б                       |
| Действие при выходе      | выгрКонтБ2++;                       |
| Включить сбор статистики | Установить флагок                   |
| <b>selectOutput6</b>     |                                     |
| Класс заявки:            | ТранспСредство                      |
| Выход true выбирается    | При выполнении условия              |
| Условие                  | entity.колГрузомест==<br>выгрКонтА2 |

|                                      |                                                                                                 |
|--------------------------------------|-------------------------------------------------------------------------------------------------|
|                                      | выгрКонтA2=0;                                                                                   |
| <b>Действие при выходе</b><br>(true) | достКА12+=<br>entity.колГрузоМест;<br>hold4.setBlocked(false);<br><b>selectOutput7</b>          |
| <b>Класс заявки:</b>                 | ТранспСредство                                                                                  |
| <b>Выход true</b><br>выбирается      | При выполнении условия                                                                          |
| <b>Условие</b>                       | entity.колГрузоМест==<br>выгрКонтB2<br>выгрКонтB2=0;                                            |
| <b>Действие при выходе</b><br>(true) | достКБ12+=<br>entity.колГрузоМест;<br>hold5.setBlocked(false);<br><b>exit</b>                   |
| <b>Имя:</b>                          | наОжПогр2                                                                                       |
| <b>Класс заявки:</b>                 | ТранспСредство                                                                                  |
|                                      | достК12+=<br>entity.колГрузоМест;                                                               |
| <b>Действие при выходе</b>           | if (entity.типТрансп==1)<br>enter14.take(entity);<br>else enter15.take(entity);<br><b>sink4</b> |
| <b>Класс заявки:</b>                 | ТранспСредство                                                                                  |

## 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.

Прямоугольник.

- Перейдите на страницу Дополнительные панели Свойства. Введите в поля X: 340, Y: 830, Ширина: 290, Высота: 190.
- Перетащите элемент **text** из вкладки Основной библиотеки в поле Текст: введите Ожидание погрузки в аэропорту 2.
- Перетащите из Основной библиотеки по два объекта **enter**, **queue**, **hold** и один объект **exit**. Поместите и соедините их так, как на [рис. 9.14](#).
- 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.

## Ожидание погрузки в аэропорту 2

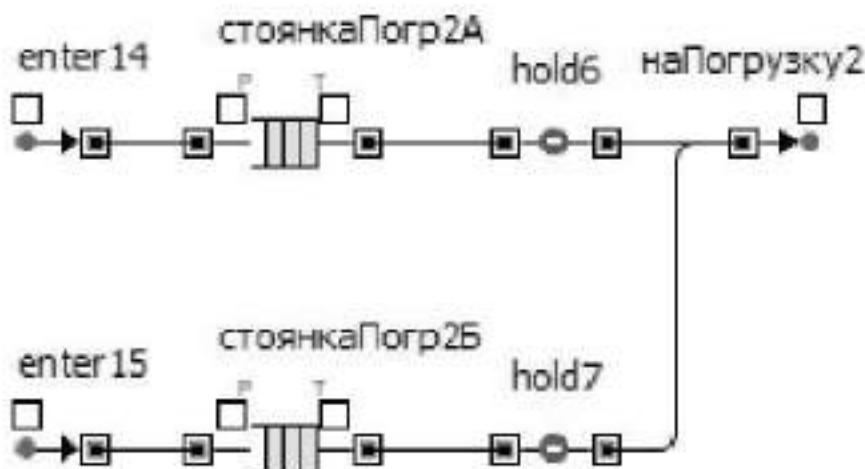


Рис. 9.14. Сегмент Ожидание погрузки в аэропорту 2

Таблица 9.13.

| Свойство                 | Значения                                                                       |
|--------------------------|--------------------------------------------------------------------------------|
|                          | <b>enter14</b>                                                                 |
| Класс заявки:            | ТранспСредство                                                                 |
|                          | <b>enter15</b>                                                                 |
| Класс заявки:            | ТранспСредство                                                                 |
|                          | <b>queue</b>                                                                   |
| Имя:                     | стоянкаПогр2А                                                                  |
| Класс заявки:            | ТранспСредство                                                                 |
| Вместимость              | колСамТипА                                                                     |
| Включить сбор статистики | Установить флагок                                                              |
|                          | <b>queue1</b>                                                                  |
| Имя:                     | стоянкаПогр2Б                                                                  |
| Класс заявки:            | ТранспСредство                                                                 |
| Вместимость              | колСамТипБ                                                                     |
| Включить сбор статистики | Установить флагок                                                              |
|                          | <b>hold6</b>                                                                   |
| Класс заявки:            | ТранспСредство                                                                 |
| Изначально заблокирован  | Установить флагок                                                              |
|                          | <b>hold7</b>                                                                   |
| Класс заявки:            | ТранспСредство                                                                 |
| Изначально заблокирован  | Установить флагок                                                              |
|                          | <b>exit</b>                                                                    |
| Имя:                     | наПогрузку2                                                                    |
| Действие при             | if (entity.типТрансп==1)<br>(hold6.setBlocked(true));<br>enter16.take(entity); |

```

else (hold7.setBlocked(true);
enter17.take(entity););

```

## 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. Перейдите на страницу Дополнительные 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 из Основной библиотеки 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. 9.14.

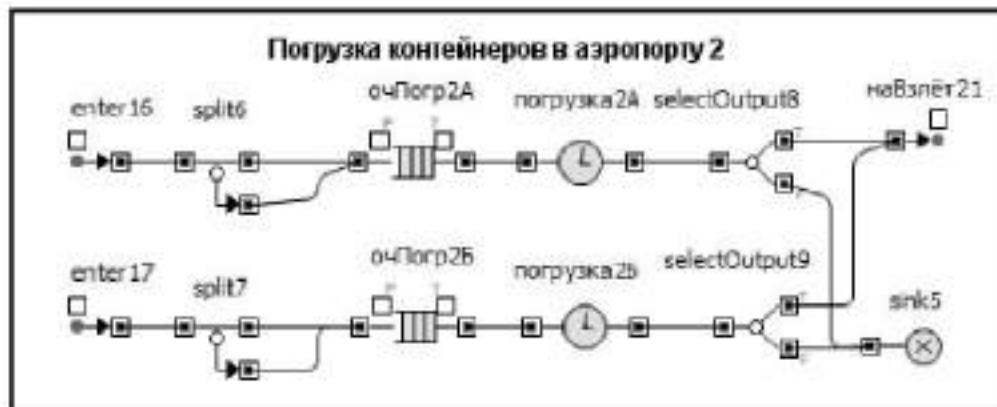


Рис. 9.15. Сегмент Погрузка контейнеров в аэропорту 2

Таблица 9.14.

Таблица 9.14.

| Свойство                  | Значения                                                                                                                                                                              |
|---------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                           | <b>enter2</b>                                                                                                                                                                         |
| Класс заявки:             | ТранспСредство                                                                                                                                                                        |
|                           | <b>enter3</b>                                                                                                                                                                         |
| Класс заявки:             | ТранспСредство                                                                                                                                                                        |
|                           | <b>split6</b>                                                                                                                                                                         |
| Классы заявок:            |                                                                                                                                                                                       |
| Оригинал, Копия           | ТранспСредство,<br>ТранспСредство                                                                                                                                                     |
| Количество копий          | entity.колГрузоМест-1                                                                                                                                                                 |
| Новая заявка (копия)      | <pre>new ТранспСредство() entity.типТрансп= original.типТрансп; entity.колГрузоМест= original.колГрузоМест; entity.tPolet= original.tPolet; entity.разные= original.разные;</pre>     |
| Действие при выходе копии | <b>split7</b>                                                                                                                                                                         |
| Классы заявок:            |                                                                                                                                                                                       |
| Оригинал, Копия           | ТранспСредство,<br>ТранспСредство                                                                                                                                                     |
| Количество копий          | entity.колГрузоМест-1                                                                                                                                                                 |
| Новая заявка (копия)      | <pre>new ТранспСредство() entity.типТрансп= original.типТрансп; entity.колГрузоМест= original.колГрузоМест; entity.врПолёта= original.врПолёта; entity.разные= original.разные;</pre> |
| Действие при выходе копии | <b>queue</b>                                                                                                                                                                          |

|                          |                                  |
|--------------------------|----------------------------------|
| Имя:                     | очПогр2А                         |
| Класс заявки:            | ТранспСредство                   |
| Максимальная вместимость | Установить флагок                |
| Действие при выходе      | entity.разные=срВрПогрКонтСам2А; |
| Включить сбор статистики | Установить флагок                |

**queue1**

|                          |                                  |
|--------------------------|----------------------------------|
| Имя:                     | очПогр2Б                         |
| Класс заявки:            | ТранспСредство                   |
| Максимальная вместимость | Установить флагок                |
| Действие при выходе      | entity.разные=срВрПогрКонтСам2Б; |
| Включить сбор статистики | Установить флагок                |

**delay**

|                          |                                  |
|--------------------------|----------------------------------|
| Имя:                     | Погрузка2А                       |
| Класс заявки:            | ТранспСредство                   |
| Задержка задаётся        | Явно                             |
| Время задержки           | exponential<br>(1/entity.разные) |
| Вместимость              | погрКонтСам2А                    |
| Действие при выходе      | погрКонтА2++;                    |
| Включить сбор статистики | Установить флагок                |

**delay1**

|                   |                                  |
|-------------------|----------------------------------|
| Имя:              | Погрузка2Б                       |
| Класс заявки:     | ТранспСредство                   |
| Задержка задаётся | Явно                             |
| Время задержки    | exponential<br>(1/entity.разные) |

**Действие при выходе** погрКонтB2++;

**Включить сбор статистики** Установить флагок

**selectOutput8**

**Класс заявки:** ТранспСредство

**Выход true выбирается** При выполнении условия

**Условие** entity.колГрузоМест ==погрКонтA2

entity.врПолёта=

**Действие при выходе** normal(отклВрПолётаA21,  
(true) срВрПолётаA21);  
погрКонтA2=0;

**selectOutput9**

**Класс заявки:** ТранспСредство

**Выход true выбирается** При выполнении условия

**Условие** entity.колГрузоМест ==погрКонтB2

entity.врПолёта=

**Действие при выходе** normal(отклВрПолётаB21,  
(true) срВрПолётаB21);  
погрКонтB2=0;

**exit**

**Имя:** наВзлёт21

**Класс заявки:** ТранспСредство

if (entity.типТрансп==1)

**Действие при выходе** enter18.take(entity);  
else enter19.take(entity);

**sink**

**Класс заявки:** ТранспСредство

## 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. Перейдите на страницу Дополнительные 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. 9.16. Сегмент Полёт из аэропорта 2 в аэропорт 1

4. Перетащите из Основной библиотеки по два объекта enter, delay и один объект exit. Поместите и соедините их так, как на [рис. 9.16](#).

## 5. Установите свойства объектов согласно табл. 9.15.

Таблица 9.15.

| Свойство                 | Значения                                                                      |
|--------------------------|-------------------------------------------------------------------------------|
|                          | <b>enter18</b>                                                                |
| Класс заявки:            | ТранспСредство                                                                |
|                          | <b>enter19</b>                                                                |
| Класс заявки:            | ТранспСредство                                                                |
|                          | <b>delay</b>                                                                  |
| Имя:                     | полётA21                                                                      |
| Класс заявки:            | ТранспСредство                                                                |
| Задержка задаётся        | Явно                                                                          |
| Время задержки           | entity.врПолёта                                                               |
| Вместимость              | колСамТипА                                                                    |
| Включить сбор статистики | Установить флагок                                                             |
|                          | <b>delay1</b>                                                                 |
| Имя:                     | ПолётB21                                                                      |
| Класс заявки:            | ТранспСредство                                                                |
| Задержка задаётся        | Явно                                                                          |
| Время задержки           | entity.врПолёта                                                               |
| Вместимость              | колСамТипБ                                                                    |
| Включить сбор статистики | Установить флагок                                                             |
|                          | <b>exit</b>                                                                   |
| Имя:                     | наОжРазгр1                                                                    |
| Действие при выходе      | if (entity.типТрансп==1)<br>enter6.take(entity);<br>else enter7.take(entity); |

Вывод результатов моделирования с использованием способа Событие

- Перетащите элемент Событие  из палитры Модель на область просмотра Результаты как на [рис. 9.3](#). Измените его имя на ОбрабРезультатМодел. Нажмите Enter.
- Установите флажок Отображать имя.
- С помощью выпадающего списка Тип события: выберите Потаймауту.
- Установите Режим: Срабатывает один раз.
- Время срабатывания (абсолютное) 720000.
- В поле Действие введите Java код, который будет выполняться при появлении этого события.

```

коэфДост21=достК21/всегоПостК2;
коэфДост12=достК12/всегоПостК1;
коэфПогр1А=погрузка1A.statsUtilization.mean();
коэфПогр1Б=погрузка1Б.statsUtilization.mean();
коэфПогр2А=погрузка2A.statsUtilization.mean();
коэфПогр2Б=погрузка2Б.statsUtilization.mean();
коэфРазгр1А=разгрузка1A.statsUtilization.mean();
коэфРазгр1Б=разгрузка1Б.statsUtilization.mean();
коэфРазгр2А=разгрузка2A.statsUtilization.mean();
коэфРазгр2Б=разгрузка2Б.statsUtilization.mean();
коэфПолётА12=полётA12.statsUtilization.mean();
коэфПолётБ12=полётB12.statsUtilization.mean();
коэфИспСам1А=коэфПогр1А+коэфРазгр1А+коэфПолётА12;
коэфИспСам1Б=коэфПогр1Б+коэфРазгр1Б+коэфПолётБ12;
коэфПолётА21=полётA21.statsUtilization.mean();
коэфПолётБ21=полётB21.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;

```

## Запуск и отладка модели

Прежде чем запустить модель:

Прежде чем запустить модель:

1. В окне Проекты выделите Воздушные Перевозки.
2. На странице Основные в поле Единицы модельного времени: установите часы.
3. В окне Проекты выделите Simulation: Main.
4. На странице Основные установите Фиксированное начальное число (воспроизводимые "прогоны").
5. В поле Начальное число: введите 15657 (можно ввести и другое начальное число, тогда результаты моделирования будут отличаться от тех, что получены при рекомендуемом начальном числе и показаны на рис. 9.17).
6. Перейдите на страницу Модельное время. В поле Остановить: выберите В заданное время.
7. В поле Конечное время: введите 720000.0. 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).
8. Запустите модель. Если появятся ошибки, исправьте их.

При правильном построении модели вы получите результаты, показанные на рис. 9.17.

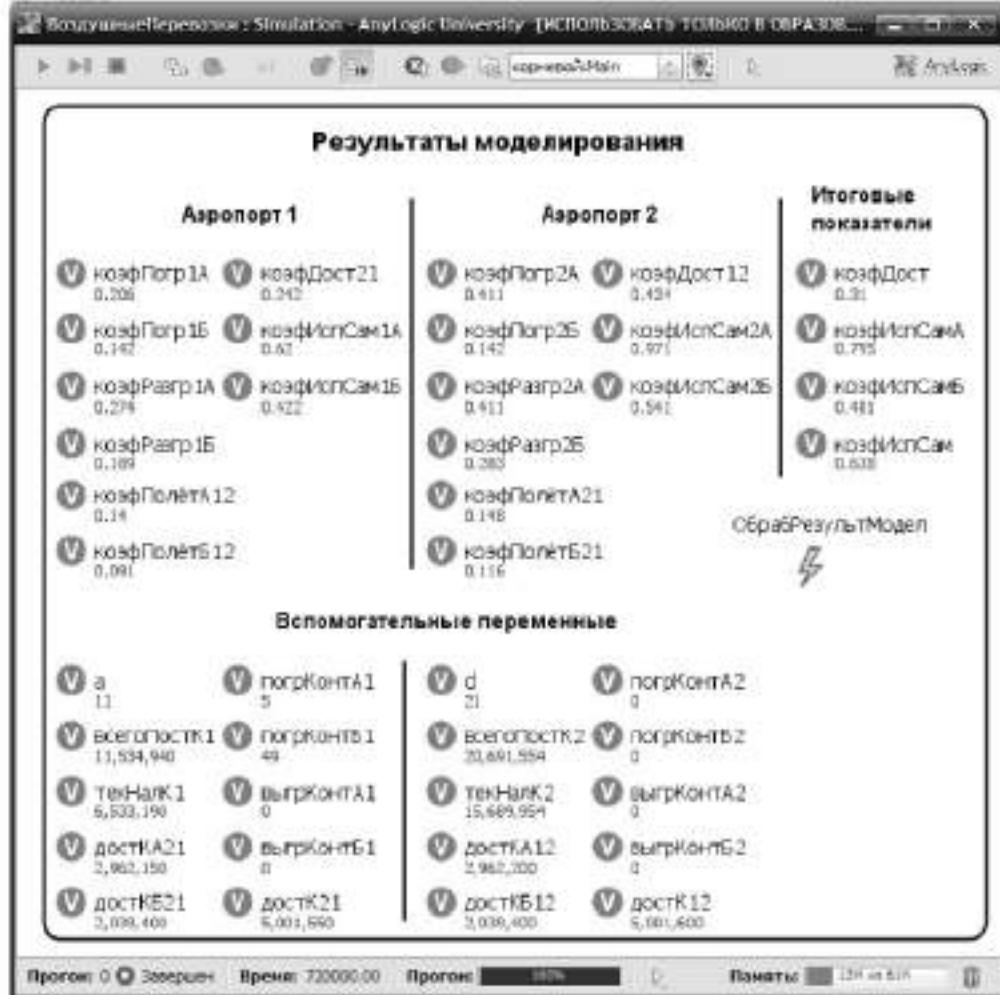


Рис. 9.17. Результаты моделирования

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 коэфИспСамA = 0,795.

# Модель обработки документов в организации

## 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 -  $t_{обе} = 12$  мин. Каждый сотрудник может принимать документы из любой организации. Освободившийся сотрудник обрабатывает последний из поступивших документов. Поступающие документы должны обрабатываться с вероятностью не менее 95 %.

Определить, достаточно ли назначенной группы из трёх сотрудников для выполнения поставленной задачи.

## Аналитическое решение задачи

Группа сотрудников работает как СМО с отказами, без очереди, состоящая из трёх каналов. Поток документов с интенсивностью  $\lambda = 15 \frac{1}{час}$  можно считать простейшим, так как он суммарный от нескольких организаций. Интенсивность обслуживания  $\mu = \frac{1}{t_{обе}} = \frac{60}{12} = 5 \frac{1}{час}$ . Закон распределения неизвестен, но это несущественно, так как показано, что для систем с отказами он может быть произвольным.

Граф состояний СМО - это схема "тибели и размножения". Для неё имеются готовые выражения для предельных вероятностей состояний:

$$P_1 = \frac{\rho}{1!} P_0, P_2 = \frac{\rho^2}{2!} P_0, \dots, P_n = \frac{\rho^n}{n!} P_0, P_{n+1} = \frac{\rho^{n+1}}{(n+1)!} P_0, \dots, P_{n+m} = \frac{\rho^{n+m}}{(n+m)!} P_0$$

$$P_0 = \left( 1 + \frac{\rho}{1!} + \dots + \frac{\rho^n}{n!} + \frac{\rho^{n+1}}{(n+1)!} + \dots + \frac{\rho^{n+m}}{(n+m)!} \right)^{-1}$$

Отношение  $\rho = \frac{\lambda}{\mu}$  называют приведенной интенсивностью потока

документов (заявок). Физический смысл её следующий: величина  $\rho$  представляет собой среднее число заявок, приходящих в СМО за среднее время обслуживания одной заявки.

$$\text{В задаче } \rho = \frac{\lambda}{\mu} = \frac{15}{5} = 3$$

В рассматриваемой СМО отказ наступает при занятости всех трёх каналов, то есть  $P_{\text{отк}} = P_3$ . 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 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  $P_{\text{обе}} = 1 - P_{\text{отк}} = 1 - 0,052 \approx 0,95$

Таким образом, только группа из шести сотрудников сможет обрабатывать поступающие документы с вероятностью 95 %.

## Решение задачи в AnyLogic

Создайте модель ОбрДокументов.

1. Выполните команду Файл/Создать/Модель на панели инструментов. Откроется диалоговое окно Новая модель.
2. В поле Имя модели диалогового окна Новая модель введите ОбрДокументов. Выберите каталог, в котором будут сохранены файлы модели.

3. Щелкните Далее. На второй странице Мастера создания модели выберите Начать создание модели "с нуля". Щелкните Далее.
4. Объекты и элементы модели ОбрДокументов показаны на рис. 10.1. Перетащите их на диаграмму класса Main, разместите, соедините и установите значения свойств согласно табл. 10.1.

Для ввода исходных данных используйте элементы Параметр, тип первых двух `double`, а третьего - `int`:

- `срИнтПост` - средний интервал поступления документов, по умолчанию - 4;

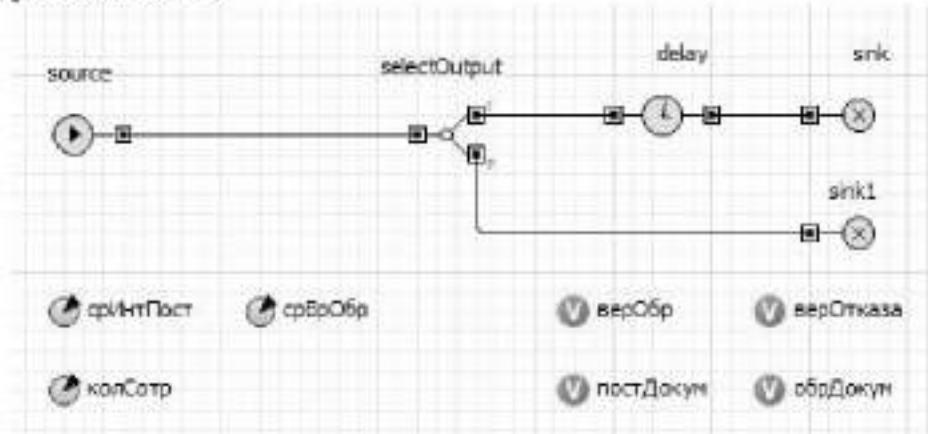


Рис. 10.1. Объекты и элементы модели ОбрДокументов

- `срБрОбр` - среднее время обработки документа, по умолчанию - 12;
- `колСотр` - количество сотрудников, по умолчанию - 3.

Таблица 10.1.

| Свойство                  | Значения                                   |
|---------------------------|--------------------------------------------|
| Имя                       | source                                     |
| Класс заявки              | Entity                                     |
| Заявки прибывают согласно | Времени между прибытиями                   |
| Время между прибытиями    | <code>exponential(1/<br/>срИнтПост)</code> |

|                                            |                                                                      |
|--------------------------------------------|----------------------------------------------------------------------|
| Количество заявок, прибывающих за один раз | 1                                                                    |
| Действие при выходе                        | постДокум++;                                                         |
| Имя                                        | selectOutput                                                         |
| Выход true выбирается                      | При выполнении условия                                               |
| Условие                                    | delay.size()<колСотр                                                 |
| Имя                                        | delay                                                                |
| Задержка задается                          | Явно                                                                 |
| Время задержки                             | exponential(1/срВрОбр)                                               |
| Вместимость                                | колСотр                                                              |
| Включить сбор статистики                   | Установить флагок                                                    |
| Имя                                        | sink                                                                 |
| Действие при входе                         | обрДокум++;<br>верОбр=обрДокум/<br>постДокум;<br>верОтказа=1-верОбр; |

Для вывода результатов моделирования используются элементы Простая 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 Основные установите Фиксированное начальное число (воспроизводимые прогоны) и Начальное число: 1055. Перейдите на страницу Модельное время, выберите из списка Остановить: В заданное время. Введите Конечное время: 600000.0 (модельное время увеличено в 10 000).

Запустите модель. Вы должны получить результаты, приведенные на рис. 10.2.

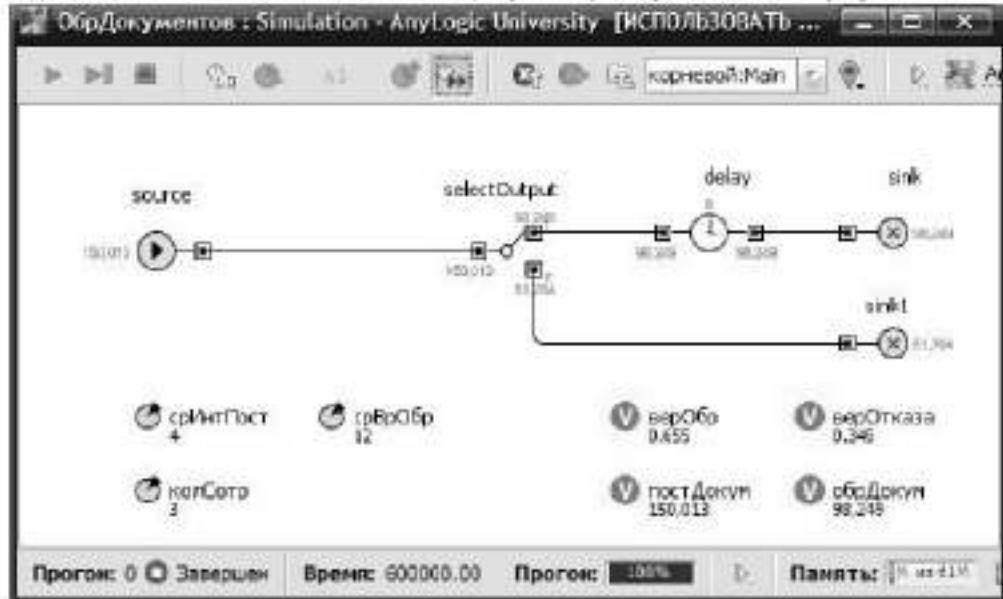


Рис. 10.2. Результаты решения задачи в AnyLogic

Вероятность не обработки документов  $\text{верОтказа}=0,345$ , то есть отличается от полученного аналитическим путём решения на 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 `колСотр` и установите по умолчанию 6. Всё остальные данные оставьте без изменения. Запустите модель. Вероятность не обработки документов  $\text{верОтказа}=0,051$ , то есть также отличается от полученного аналитическим путём решения на 0,001.

Сравнительную оценку можно было бы провести и при проведении расчётов с большим числом знаков после запятой, то есть с большей точностью.

## 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 95 %, вносят свои коррективы, в которых мы убедились, решая эту задачу и аналитически, и в двух системах моделирования.

## Решение обратных задач в AnyLogic

Решение обратных задач средствами GPSS World было рассмотрено в главах 1 и 2. Теперь приступим к решению этих же задач в AnyLogic.

Определение среднего времени обработки группы запросов сервером

1. В п. 1.2.6 была сохранена модель простой эксперимент с именем Сервер Обратная задача. Откройте её.
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 Простая 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. Введите в поле Значение по умолчанию: 1.
6. Тип элемента времяВыполнения double.
7. Выделите блок sink и в поле Действие при входе: введите Java код:

```
if( sink.in.count() == количествоЗапросов ){
    времяВыполнения = time();
    stopSimulation();
}
```

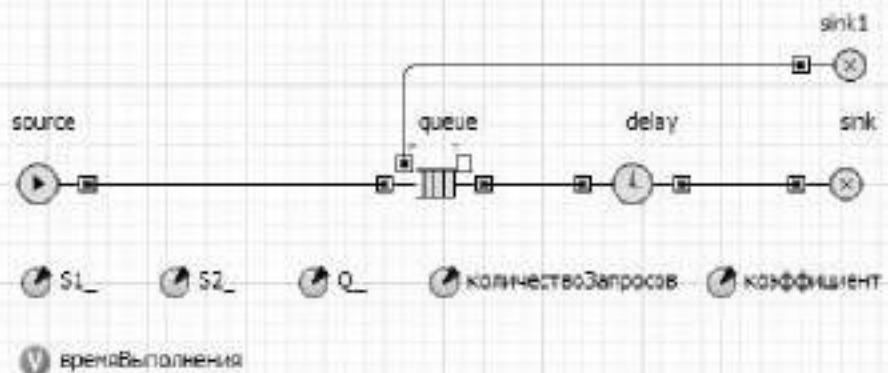


Рис. 11.1. Простой эксперимент Сервер Обратная задача

Этот код сравнивает заданное количество запросов в группе (в данном случае 29) с обработанным количеством запросов. При равенстве заданных и обработанных запросов переменной время выполнения 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 щелкните 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 оставьте имя эксперимента, рекомендованное системой.
4. Щелкните кнопку Готово. Появится страница Основные панели Свойства (рис. 11.2). Установите свойства согласно рис. 11.2.
5. Перейдите на страницу Дополнительные. В поле Код инициализации эксперимента: введите `data.reset();`. А в поле Действие после прогона модели:

```
data.add( time() / коэффициент );
```

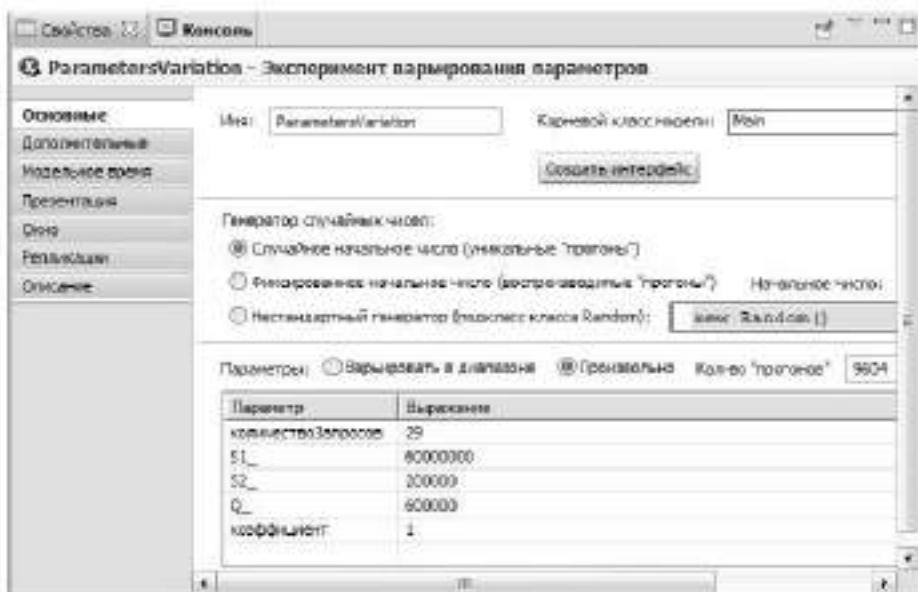
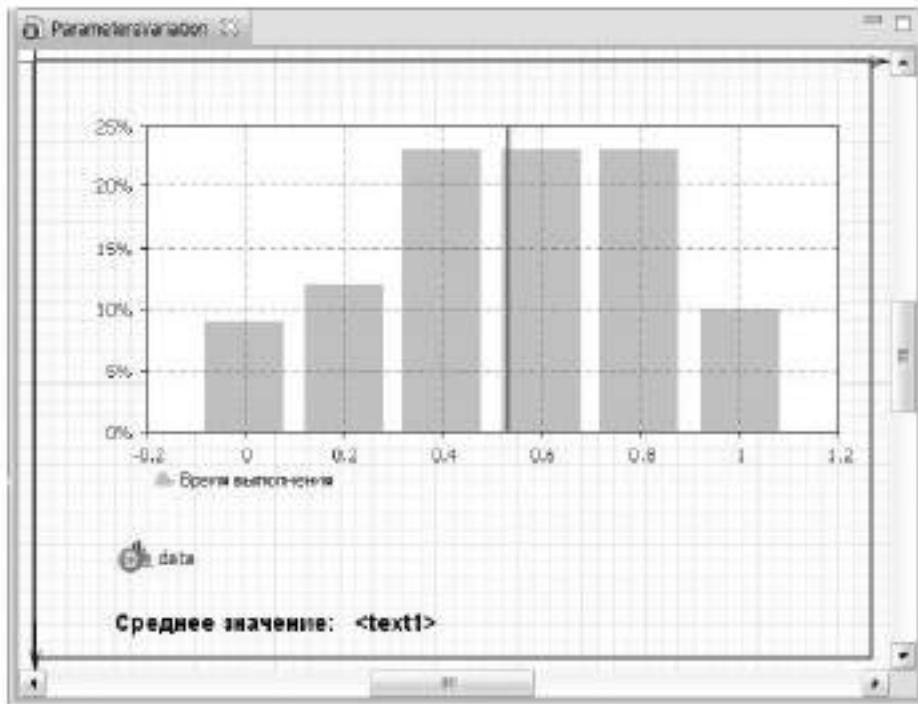


Рис. 11.2. Страница Основные Эксперимента варьирования параметров



### Рис. 11.3. Интерфейс Эксперимента варьирования параметров

6. В эксперименте Варьирование параметров, как вы помните, в отличие от эксперимента Оптимизация интерфейс создает пользователь.
7. Создайте интерфейс, показанный на [рис. 11.3](#). Здесь вы видите гистограмму, которая будет отображать время выполнения группы запросов. Начните с нажатия Создать интерфейс.
8. Перетащите элемент Гистограмма из палитры Статистика на диаграмму активного класса. На странице Основные установите флажок Отображать среднее. Оставьте флажок Отображать плотность вероятности.
9. Щёлкните Добавить данные.
10. Установите Заголовок: Время выполнения. Набор данных: data. Цвет плотности верти: silver. Цвет линии среднего: crimson. Установите Не обновлять автоматически.
11. Перетащите элемент Данные гистограммы. В поле Имя: введите data. Остальные свойства оставьте установленными по умолчанию.
12. Добавьте элемент text и введите Среднее значение:.
13. Добавьте второй элемент text. Оставьте имя, предложенное системой. Перейдите на страницу Динамические и в поле Текст: введите data.mean().
14. Построение Эксперимента варьирование параметров завершено. Запустите его. При наличии ошибок, устранийте их.

Далее проведём по четыре эксперимента в AnyLogic и GPSS World. Результаты одного из экспериментов при количестве прогонов 29,16 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 количествоЗапросов заменяется на 291, а значение по умолчанию параметра коэффициент учёта дробной части - на 10 на диаграмме простого эксперимента.

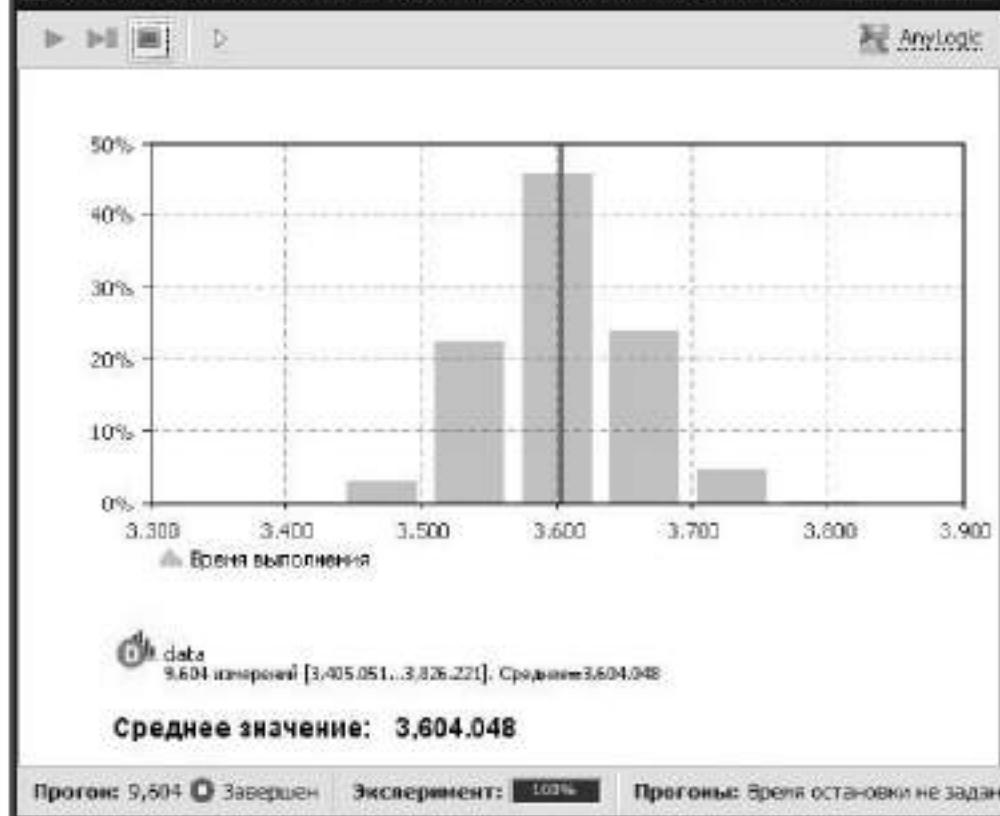


Рис. 11.4. Вариант результатов решения обратной задачи

Таблица 11.1.

| Параметры и показатели                       | Системы моделирования |          |
|----------------------------------------------|-----------------------|----------|
|                                              | GPSS World            | AnyLogic |
| <b>Эксперимент 1</b>                         |                       |          |
| Количество запросов                          | 29                    | 29       |
| Коэффициент учёта дробной части              | 1                     | 1        |
| Машинное время выполнения модели 2 сек       | 2 сек                 | 1,3 сек  |
| Время выполнения группы запросов:            | 3589,688              | 3781,755 |
| $\Delta_1 =  3589,688 - 3781,755  = 192,067$ |                       |          |
| <b>Эксперимент 2</b>                         |                       |          |
| Количество запросов                          | 291                   | 291      |

|                                  |          |          |
|----------------------------------|----------|----------|
| Коэффициент учёта дробной части  | 10       | 10       |
| Машинное время выполнения модели | 24 сек   | 7 сек    |
| Время выполнения группы запросов | 3596,113 | 3614,877 |

$$\Delta_2 = |3596,113 - 3614,877| = 18,764$$

### Эксперимент 3

|                                  |              |          |
|----------------------------------|--------------|----------|
| Количество запросов              | 2916         | 2916     |
| Коэффициент учёта дробной части  | 100          | 100      |
| Машинное время выполнения модели | 4 мин 48 сек | 44 сек   |
| Время выполнения группы запросов | 3603,039     | 3604,676 |

$$\Delta_3 = |3603,089 - 3604,676| = 1,637$$

### Эксперимент 4

|                                  |               |               |
|----------------------------------|---------------|---------------|
| Количество запросов              | 29161         | 29161         |
| Коэффициент учёта дробной части  | 1000          | 1000          |
| Машинное время выполнения модели | 40 мин 48 сек | 11 мин 17 сек |
| Время выполнения группы запросов | 3604,526      | 3603,182      |

$$\Delta_4 = |3604,526 - 3603,182| = 1,344$$

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:  $\Delta_1 = 192,067 \dots \Delta_4 = 1,344$ . При этом среднее время в обеих моделях приближается к времени моделирования (3600) в прямой задаче. 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](#)).

## 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

1. Откройте модель Изготовление\_в\_цехе\_деталей. Сохраните с именем Изготовление\_в\_цехе\_деталей Обратная задача.
2. Добавьте на диаграмму класса Main два элемента Параметр и один элемент Простая переменная.
3. Тип параметра Количество деталей 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 диаграмму Kontrol. Выделите блок sink сегмента Склад готовых изделий и в поле Действие при входе: введите вместо имеющегося там Java код:

```

if( склГотДет.in.count() == get_Main().количествоДеталей ){
    get_Main().времяИзготовления=time();
    stopSimulation();
}

```

Для выполнения заданного количества прогонов также как и в предыдущем случае создайте эксперимент варьирования параметров.

1. В панели Проект щелкните 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 оставьте имя эксперимента, рекомендованное системой.
4. Щелкните 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 Дополнительные.
6. В поле Код инициализации эксперимента: введите  

```
data.reset();
```

А в поле Действие после прогона модели:

```
data.add( time() /коэффициент );
```

7. Перейдите на страницу Модельное время. В поле Остановить выберите из списка Нет.
8. Постройте интерфейс эксперимента также как и предыдущей модели (п. 11.1), выполнив пп. 7...13.
9. Построение Эксперимента варьирование параметров завершено. Запустите его. При наличии ошибок, устраните их.

При построении мы ввели данные для проведения эксперимента с учётом всех 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

Таблица 11.2.

| Параметры и показатели           | Системы моделирования |               |
|----------------------------------|-----------------------|---------------|
|                                  | GPSS World            | AnyLogic      |
| Количество деталей               | 9017                  | 9017          |
| Коэффициент учёта дробной части  | 1000                  | 1000          |
| Машинное время выполнения модели | 2 час 23 мин 48 сек   | 30 мин 54 сек |
| Время изготовления деталей       | 7,275 час             | 7,286 час     |

$$\Delta_1 = |7,275 - 7,286| = 0,011$$

В табл. 11.2 сведены результаты экспериментов решения обратной

задачи в GPSS World и AnyLogic. Они свидетельствуют об их адекватности, так как  $\Delta_1 = 0,011$ . Если, конечно, исследователя устраивает такое  $\Delta_1$ .

Но машинное время выполнения модели в GPSS World примерно в пять раз больше, чем в AnyLogic, и равно 2 час 23 мин 43 сек.

Обратим внимание ещё раз на то, что время изготовления при повторных запусках эксперимента может отличаться, так как выбраны были уникальные прогони 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.

## 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 объеме 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 среди 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.

# Приложения

## Приложение 1

### Объекты Основной библиотеки

#### 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 решать, что делать с ними                                     |
|  | Hold          | Блокирует/разблокирует поток заявок на определенном участке блок-схемы                                                                                                 |
|  | Split         | Для каждой поступающей заявки объект создает заданное число новых заявок и пересыпает 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 |

|                                                                                    |                             |                                                                                                                                |
|------------------------------------------------------------------------------------|-----------------------------|--------------------------------------------------------------------------------------------------------------------------------|
|     | <b>Queue</b>                | Моделирует очередь заявок, ожидающих приема объектами, следующими за данным в диаграмме                                        |
|    | <b>Match</b>                | Синхронизирует два потока заявок путем нахождения пар заявок, удовлетворяющих заданному критерию соответствия                  |
|    | <b>Restricted AreaStart</b> | Обозначает вход в область процесса, в которой одновременно может находиться ограниченное количество заявок                     |
|    | <b>Restricted AreaEnd</b>   | Обозначает выход из области процесса, в которой может находиться только ограниченное количество заявок                         |
| <b>Работа с содержимым заявки</b>                                                  |                             |                                                                                                                                |
|    | <b>Batch</b>                | Преобразует заданное количество поступающих в объект заявок в одну заявку-партию                                               |
|    | <b>Unbatch</b>              | Извлекает все заявки, содержащиеся в поступающей заявке-партии и пересыпает их далее. Сама заявка-партия при этом уничтожается |
|    | <b>Pickup</b>               | Удаляет заявки из заданного объекта Queue и добавляет их к содержимому поступающей заявки-контейнера                           |
|  | <b>Dropoff</b>              | Удаляет выбранные заявки из поступающей заявки-контейнера и пересыпает их далее                                                |
|  | <b>Assembler</b>            | Осуществляет сборку одной новой заявки из определенного числа заявок, пришедших из различных источников (до 5)                 |
| <b>Обработка</b>                                                                   |                             |                                                                                                                                |
|  | <b>Delay</b>                | Задерживает заявки на заданный период времени                                                                                  |
| <b>Работа с ресурсами</b>                                                          |                             |                                                                                                                                |
|  | <b>ResourcePool</b>         | 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                                                            |
|                                                                                    |                     | 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 |
|                                                                                    |                     | Моделирование транспортных сетей                                                                                                                                               |
|    | Network             | Задает топологию сети и управляет сетевыми ресурсами                                                                                                                           |
|    | NetworkEnter        | Регистрирует заявку в сети и помещает ее в заданный узел сети                                                                                                                  |
|    | NetworkExit         | Удаляет заявку из сети                                                                                                                                                         |
|  | NetworkMoveTo       | Перемещает заявку в новое место сети                                                                                                                                           |
|  | NetworkResourcePool | Задает набор сетевых ресурсов, которые могут захватываться и освобождаться заявками                                                                                            |
|  | NetworkRelease      | Освобождает ранее захваченные заявкой сетевые ресурсы                                                                                                                          |
|  | NetworkSeize        | Захватывает для заявки заданное количество сетевых ресурсов                                                                                                                    |
|  | NetworkSendTo       | Посыпает (перемещает) указанные движущиеся/переносные сетевые ресурсы из их текущего местоположения в заданный узел сети                                                       |

|                                   |                    |                                                                                                                                                                                                                                                                                                                                                                                                             |
|-----------------------------------|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                   | NetworkAttach      | Присоединяет к заявке указанные движущиеся/переносные сетевые ресурсы                                                                                                                                                                                                                                                                                                                                       |
|                                   | NetworkDetach      | Отсоединяет от заявки ранее присоединенные ресурсы                                                                                                                                                                                                                                                                                                                                                          |
| Моделирование зон хранения в сети |                    |                                                                                                                                                                                                                                                                                                                                                                                                             |
|                                   | NetworkStorage     | Моделирует два стоящих друг напротив друга стеллажа и проход между ними                                                                                                                                                                                                                                                                                                                                     |
|                                   | NetworkStorageZone | Моделирует зону хранения, состоящую из набора стеллажей и проходов между ними (моделируемыми объектами NetworkStorage), предоставляющий централизованный доступ и управление этими стеллажами                                                                                                                                                                                                               |
|                                   | NetworkStoragePick | Извлекает заявку из ячейки стеллажа или зоны хранения и перемещает ее в заданный узел сети                                                                                                                                                                                                                                                                                                                  |
|                                   | NetworkStoragePut  | Помещает заявку в ячейку заданного стеллажа NetworkStorage или зоны хранения NetworkStorageZone                                                                                                                                                                                                                                                                                                             |
| Измерение времени                 |                    |                                                                                                                                                                                                                                                                                                                                                                                                             |
|                                   | TimeMeasureStart   | TimeMeasureStart вместе с TimeMeasureEnd составляет пару объектов, позволяющую измерять время, проведенное заявками между двумя точками диаграммы процесса. TimeMeasureStart задает начальную точку, он запоминает момент времени, в который заявка проходит через этот объект. Обычно с их помощью измеряется время нахождения заявки в системе или длительность пребывания заявки в каком-то подпроцессе. |
|                                   | TimeMeasureEnd     | TimeMeasureEnd вычисляет для каждой поступившей в него заявки разность между текущим моментом времени и моментом, запомненным объектом TimeMeasureStart, на который ссылается этот объект                                                                                                                                                                                                                   |



## Приложение 2

### Палитры

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

- **Основная** - Палитра Основная содержит основные элементы, с помощью которых Вы можете задать динамику модели, ее структуру и данные.
- **Системная динамика** - Палитра Системная динамика содержит элементы, часто используемые в системной динамике: элементы диаграммы потоков и накопителей, а также параметр, соединитель и табличную функцию.
- **Диаграмма состояний** - Палитра Диаграмма состояний содержит блоки диаграмм состояний - диаграмм, позволяющих графически задавать поведение объекта.
- **Диаграмма действий** - Палитра Диаграмма действий содержит блоки диаграмм действий - структурированных блок-схем, позволяющих задавать алгоритмы визуально.
- **Статистика** - Палитра Статистика содержит элементы, используемые для сбора, анализа и отображения результатов моделирования.
- **Презентация** - Палитра Презентация содержит элементы, используемые для рисования презентаций моделей: примитивные фигуры, с помощью которых Вы можете рисовать сложные презентации.
- **3D** - Палитра 3D 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 Элементы управления содержит элементы управления, с помощью которых Вы можете сделать

анимации Ваших моделей интерактивными.

- **Внешние данные** - Палитра Внешние данные содержит инструменты для работы с внешними данными - базами данных и текстовыми файлами.
- **Картинки** - Палитра Картинки содержит набор картинок наиболее часто моделируемых объектов: человек, медсестра, врач, грузовик, фура, погрузчик, склад, завод и т. д.
- **3D Объекты** - Палитра 3D Объекты содержит набор трехмерных изображений наиболее часто моделируемых объектов: человек, медсестра, врач, грузовик, фура, погрузчик, поддон и т.д.

Элементы, блоки, изображения палитр приведены ниже.

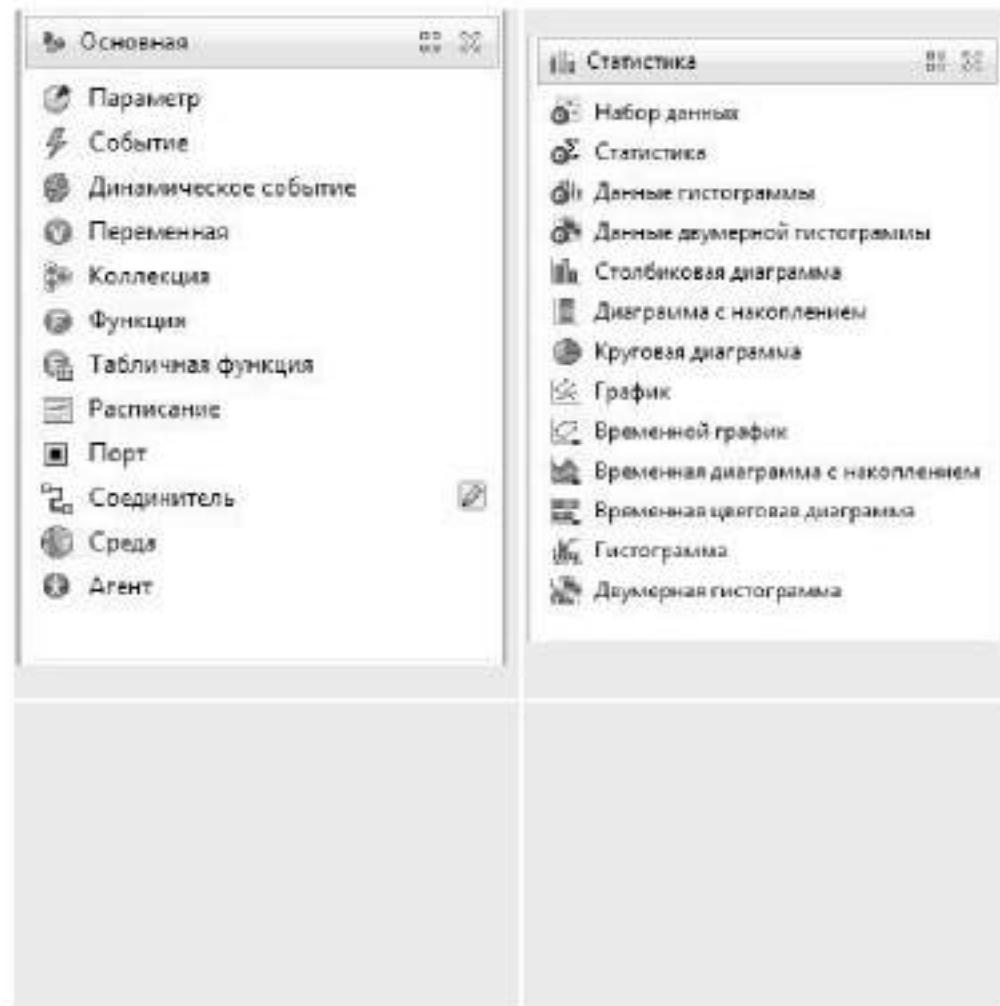


 Диаграмма действий

33 33

-  Диаграмма действий
-  Код
-  Решение (If ... Else)
-  Локальная переменная
-  Цикл While
-  Цикл Do While
-  Цикл For
-  Вернуть значение (Return)
-  Выход из цикла (Break)

 Системная динамика

33 33

-  Накопитель
-  Поток
-  Вспомогательная переменная
-  Связь
-  Параметр
-  Табличная функция
-  Цикл
-  Копия

 Элементы управления

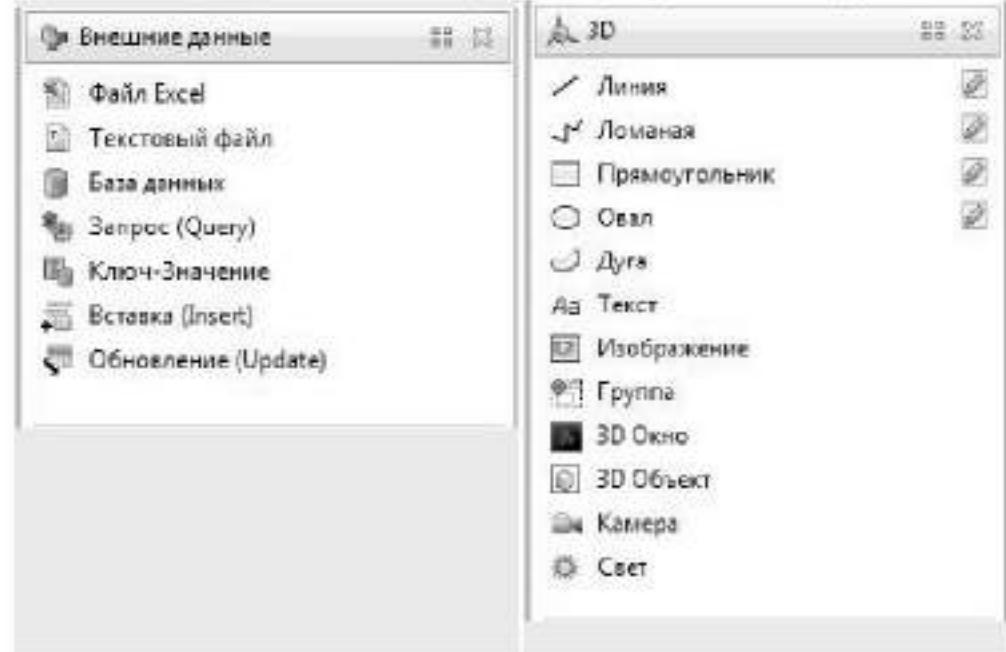
33 33

-  Кнопка
-  Флажок
-  Текстовое поле
-  Переключатель
-  Бегунок
-  Выпадающий список
-  Список
-  Элемент выбора файла
-  Индикатор прогресса

 Презентация

33 33

-  Линия
-  Ломаная
-  Кривая
-  Прямоугольник
-  Скругленный прямоугольник
-  Овал
-  Дуга
-  Точка
-  Текст
-  Изображение
-  Группа
-  Область просмотра
-  Чертеж САПР
-  Карта ГИС



## Приложение 3

Методика концептуального проектирования 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.

**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*, URL: <http://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., Кирик Д.И., Сыпченко Р.П., *Компьютерное моделирование: Пособие по курсовому и дипломному проектированию*, URL: <http://www.xjtek.ru>, 2011
14. 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
15. 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
16. Варжапетян А.Г., *Имитационное моделирование на GPSS/H: Монография / А. Г. Варжапетян*, Вузовская книга, 2007
17. Девятков В.В., *Разработка приложений в среде GPSS World: Статья — В сб.*

- докладов конференции ИММОД-2005 «Имитационное моделирование. Теория и практика», СПб.: ФГУП ЦНИИТС, 2005
18. Девятков В.В, *Мир имитационного моделирования: взгляд из России*, Прикладная информатика, № 4 (34), 2011. С. 9-29
19. Карпов Ю.Г, *Имитационное моделирование систем. Введение в моделирование с AnyLogic*, СПб.: БХВ-Петербург, 2005
20. Колесов Ю.Б., Сениченков Ю.Б, *Моделирование систем. Объектно-ориентированный подход*; Учеб. пособие, СПб.: БХВ-Петербург, 2006
21. Колесов Ю.Б., Сениченков Ю.Б, *Моделирование систем. Практикум по компьютерному моделированию*; Учеб. пособие, СПб.: БХВ-Петербург, 2007
22. Лоу А., Кельтон Д, *Имитационное моделирование*, СПб.: Питер, БХВ-Петербург, 2004
23. 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).
24. Шеннон Р, *Имитационное моделирование — искусство и наука*, М.: Мир, 1978
25. Шрайбер Т.Дж, *Моделирование на GPSS*, М.: Машиностроение, 1980
26. "Экс-Джей Технолоджис", URL: <http://www.xjtek.ru>

# Содержание

|                                                                  |     |
|------------------------------------------------------------------|-----|
| Титульная страница                                               | 2   |
| Выходные данные                                                  | 3   |
| Лекция 0. Введение                                               | 4   |
| Лекция 1. Модель обработки запросов сервером                     | 7   |
| Лекция 2. Модель процесса изготовления в цехе деталей            | 92  |
| Лекция 3. Модель функционирования направления<br>связи           | 138 |
| Лекция 4. Модель функционирования сети связи                     | 161 |
| Лекция 5. Модель функционирования системы связи                  | 248 |
| Лекция 6. Модель функционирования предприятия                    | 328 |
| Лекция 7. Модель функционирования терминала                      | 390 |
| Лекция 8. Модель предоставления ремонтных услуг                  | 424 |
| Лекция 9. Модель функционирования системы<br>воздушных перевозок | 459 |
| Лекция 10. Модель обработки документов в<br>организации          | 515 |
| Лекция 11. Решение обратных задач в AnyLogic                     | 522 |
| Лекция 12. Приложения                                            | 533 |
| Список литературы                                                | 541 |