Министерство образования и науки Российской Федерации
НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

О.К. АЛЬСОВА

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ СИСТЕМ В СРЕДЕ ExtendSim

Утверждено Редакционно-издательским советом университета
в качестве учебного пособия

НОВОСИБИРСК
2016

УДК 004.94(075.8)
А 579

Рецензенты:

М.Г. Гриф, д-р техн. наук, профессор
В.М. Зыбарев, канд. техн. наук, доцент

Работа подготовлена на кафедре вычислительной техники
для студентов II курса АВТФ по дисциплинам
«Математическое моделирование» и «Имитационное моделирование»

Альсова О.К.

А 579 Имитационное моделирование систем в среде *ExtendSim* : учебное пособие / О.К. Альсова. – Новосибирск: Изд-во НГТУ, 2016. – 104 с.

ISBN 978-5-7782-2840-5

Рассмотрены вопросы разработки и исследования имитационных моделей систем средствами визуальной среды моделирования *ExtendSim*.

Приведено описание базового инструментария *ExtendSim* для разработки моделей систем в рамках дискретно-событийного подхода.

В пособии содержится большое количество примеров, позволяющих изучить основные способы и приемы разработки моделей систем в *ExtendSim*, оценить эффективность системы на основе модели, провести анализ чувствительности модели, статистически обработать результаты моделирования. Также рассмотрены основные типовые ситуации, которые возникают при моделировании потоков поступления и обслуживания заявок в системе и которые необходимо учесть в модели. Приведены контрольные вопросы и задачи для использования в рамках лабораторного практикума при изучении соответствующих разделов дисциплин «Имитационное моделирование», «Математическое моделирование».

Предназначено для бакалавров II курса АВТФ, обучающихся по направлениям 09.03.01 «Информатика и вычислительная техника», 09.03.04 «Программная инженерия».

УДК 004.94(075.8)

ISBN 978-5-7782-2840-5

1. ВВЕДЕНИЕ В СРЕДУ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ *ExtendSim*

1.1. Общая характеристика среды *ExtendSim*

ExtendSim – универсальная среда имитационного моделирования, разработанная компанией Imagine *That* (Сан-Хосе, Калифорния). Среда *ExtendSim* поддерживает разные подходы к проектированию имитационных моделей систем и процессов:

– методы моделирования непрерывных динамических систем;

– методы дискретно-событийного моделирования;

– методы агентного моделирования;

– гибридный подход (комбинация различных методов моделирования в одной модели).

ExtendSim – среда визуального моделирования, в которой модель представляет собой совокупность взаимосвязанных блоков. Каждый блок имеет условное графическое обозначение, блоки связаны между собой линиями-соединителями. Блоки включены в библиотеки. Всего в *ExtendSim* семь специализированных библиотек для моделирования различных типов систем.

Процесс разработки модели заключается в выборе блоков из библиотеки, размещении блоков в модельном окне, настройке блоков и связывании их с помощью линий-соединителей. С каждым блоком связано диалоговое окно, в котором задаются параметры работы блока (например, закон распределения интервалов между поступлением заявок, время обслуживания заявки, дисциплина обслуживания и т. п.). В *ExtendSim* встроен также внутренний язык программирования *ModL*, позволяющий создавать новые блоки, которые затем можно применять при конструировании моделей наряду со стандартными блоками.

ExtendSim позволяет создавать наглядные, интуитивно-понятные имитационные модели процессов и систем разных типов: непрерывных, дискретно-событийных, основанных на агентах, линейных, нелинейных и смешанной природы.

1.2. Описание объектов и основных приемов работы в среде *ExtendSim*

К основным объектам среды относятся блоки, входные и выходные коннекторы, соединители и диалоговые окна для настройки работы блоков. Рассмотрим на простом примере приемы работы и последовательность разработки в среде *ExtendSim* модели простейшей системы.

Пример 1. Моделируется процесс заполнения бассейна в течение 36 месяцев. Бассейн заполняется ежемесячно двумя входными потоками: случайный поток, описывающий дождевую воду, и постоянный поток, объем воды в котором задан константой и зависит от месяца года. Модель бассейна описывает изменение водного уровня бассейна за наблюдаемый период. В начале моделирования в бассейне нет воды. Изменение уровня воды происходит один раз в месяц.

Конечный вид модели показан на рис. 1.

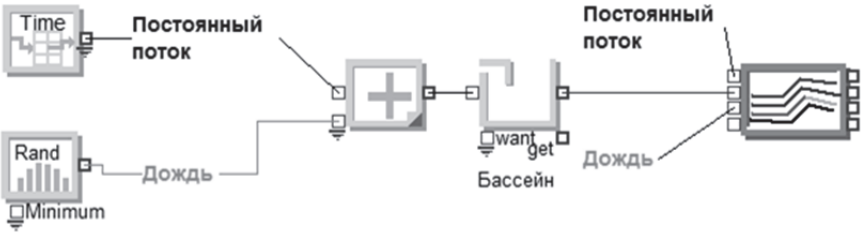


Рис. 1. Интерфейс модели «Бассейн»

В модели бассейна используется пять блоков. Информация входит в блок, обрабатывается и/или изменяется, и потом пересылается к следующему блоку через линию-соединитель (рис. 2).

Рис. 2. Части модели «Бассейн»

Блоки. Каждый блок в *ExtendSim* представляет часть моделируемого процесса или системы. Блоки хранятся в архивах, называемых библиотеками. В библиотеке для каждого входящего в нее блока определены функционал, значок, диалоговое окно блока. У блоков есть названия, например, «*Math*» – «Математика» или «*Queue*» – «Очередь», отражающие выполняемую блоком функцию. Пользователь может также задать собственное имя блоку. При включении блока в модель сам блок не копируется. Вместо этого включается и сохраняется в модели справочная информация о блоке. Любые данные, которые вводятся в диалоговом окне блока, также хранятся в пределах модели. Есть много преимуществ в использовании в модели справочной информации о блоках вместо фактических блоков. Если изменяется определение блока в библиотеке, все модели, которые используют этот блок, автоматически обновляются. Кроме того, определения блока много «весят», поэтому хранение только справочной информации экономит вычислительные ресурсы, память, уменьшает время моделирования.

Коннекторы. У большинства блоков в *ExtendSim* есть коннекторы ввода и вывода (маленькие квадраты, приложенные к блоку, рис. 2). Потоки информации поступают в блок через входные коннекторы и выходят из блока через выходные. Блок может иметь много входов и/или выходов.

Диалоговые окна. Настройка работы блока выполняется с помощью диалогового окна, связанного с каждым блоком. Диалоговые окна используются для ввода значений и параметров настройки модели перед моделированием и вывода результатов моделирования. Чтобы открыть диалог блока, дважды щелкните значок блока или щелкните правой кнопкой мыши по значку и выберите из выпадающего меню пункт *Open Dialog*. На рис. 3 изображено диалоговое окно, соответствующее блоку *Holding Tank* (Бассейн). Вверху диалогового окна выводятся глобальный номер блока, его название и библиотека, в которой находится блок.

Глобальные номера блока – уникальные идентификаторы, назначаемые последовательно в соответствии с порядком добавления пользователем блока в модель. Внизу каждого диалогового окна кнопка *Help* (Помощь). В разделе «Помощь» представлена вся информация о блоке: назначение и использование блока, коннекторов, описание каждого элемента диалогового окна и т. п. Около кнопки *Help* расположено текстовое поле, в которое можно ввести метку (имя) блока, до 31 сим-

вола. В диалоговом окне также отображаются результаты (выходные характеристики) моделирования, причем можно наблюдать изменение характеристик в процессе моделирования, если оставить диалоговое окно открытым.

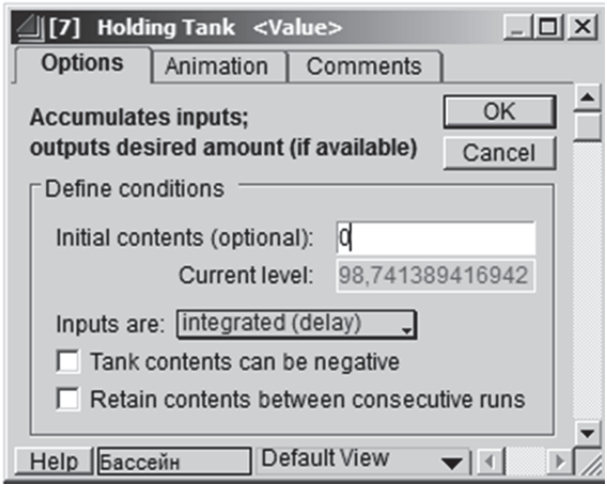


Рис. 3. Диалоговое окно блока Holding Tank

Соединители блоков. Соединители – линии, связывающие коннекторы ввода и вывода информации. В *ExtendSim* функции большинства соединителей предопределены для каждого блока. Например, блок *Math* в зависимости от заданных пользователем настроек складывает (либо вычитает, делит, умножает, логарифмирует, …) значения, поданные на вход. Всего в блоке *Math* доступно 38 математических и логических операций.

У блока также могут быть переменные входные или выходные соединители, позволяющие увеличить число входов (выходов) блока, обозначающиеся черной стрелкой. У блока *Math*, например, есть переменный входной соединитель, перетащив вниз черную стрелку, можно увеличить число входов блока.

Типы соединений. Есть два типа соединений в *ExtendSim*: подключения линии и именные подключения. Подключения линии соединяют выход одного блока со входом другого; именные подключения используют текстовые метки как выходы и входы, заставляя данные «спрыгнуть» с выхода ко входу, не используя линии подключения.

Линии соединения могут быть нарисованы с использованием трех различных стилей: прямо, правый угол и мультисегмент (рис. 4). Стиль по умолчанию – правый угол.

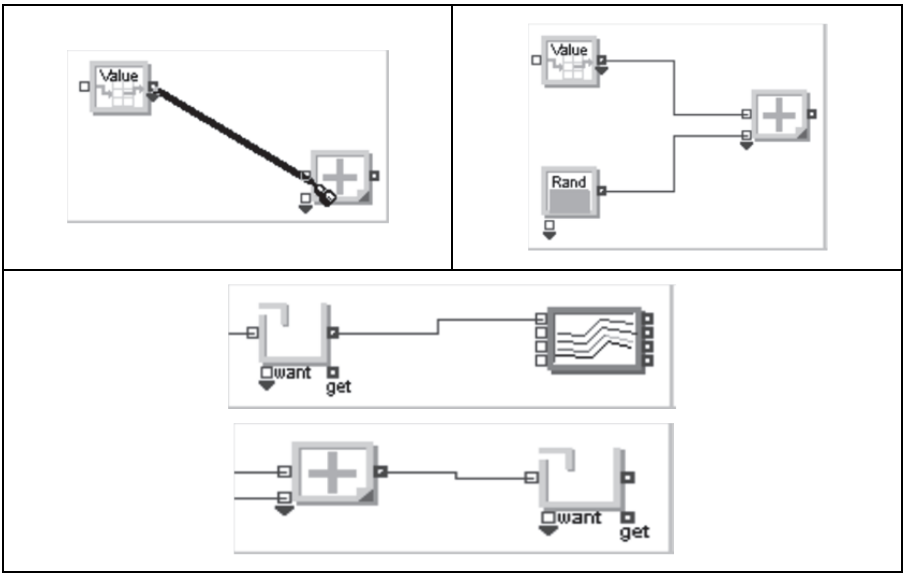


Рис. 4. Типы соединений блоков

Для создания модели бассейна необходимо выполнить следующую последовательность дейтвий:

- открыть новый модельный рабочий лист;
- установить параметры моделирования;
- построить модель, используя блоки из библиотек;
- выбрать и задать параметры настройки блоков.

Открытие нового модельного рабочего листа. Для открытия нового модельного рабочего листа выберите пункт меню *File>New Model*. В результате будет создан пустой модельный рабочий лист под названием *Model-1*.

Установка параметров моделирования. Для задания параметров моделирования выберите пункт меню *Run>Simulation Setup*. В результате откроется диалоговое окно (рис. 5), в котором устанавливаются параметры моделирования и анимации, а именно: время моделирования, начальные значения для генераторов случайных чисел, режим взаимо-

действия между моделированием и анимацией и т. д. В диалоговом окне есть вкладки *Setup*, *Continuous*, *Random Numbers*, *3D Animation*, *Comments*. Самые общие параметры моделирования, которые необходимо задать (и часто единственные): *End time* (время окончания моделирования) и *Global time units* (глобальные единицы времени), расположенные на вкладке *Setup*. В большинстве случаев требуется, чтобы моделирование началось в нулевой момент времени (по умолчанию).

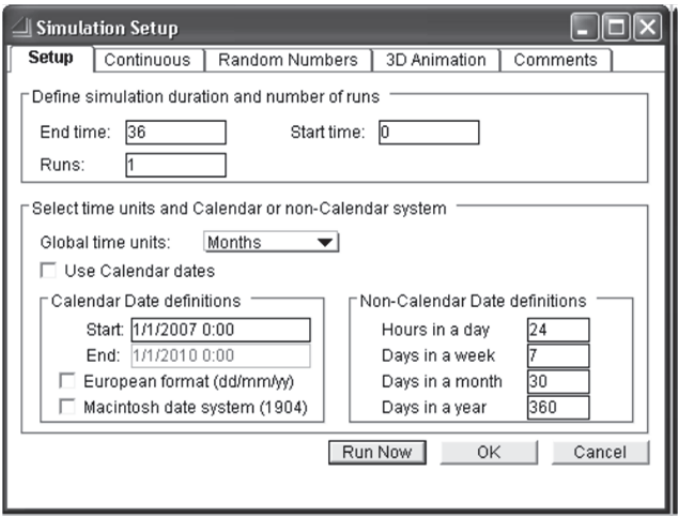


Рис. 5. Задание параметров моделирования

Для модели бассейна устанавливаются следующие параметры:

- время окончания (*End time*): 36;
- начальное время (*Start time*): 0 (значение по умолчанию);
- количество прогонов модели (*Runs*): 1 (значение по умолчанию);
- глобальные единицы времени (*Global time units*): месяц.

Модель бассейна выполняется в течение 36 месяцев модельного времени, производя вычисления уровня воды каждый месяц. Настройки параметров моделирования сохраняются при повторном запуске модели, т. е. параметры настройки задаются только один раз.

Выбор блоков модели. Блоки, используемые в модели бассейна, хранятся в библиотеках *Value* и *Plotter*. Для открытия библиотеки выберите пункт меню *Library>Open Library* и далее файл необходимой библиотеки (*Value.lix* или *Plotter.lix*). Открытые библиотеки перечис-

лены в алфавитном порядке внизу в меню *Library*. Теперь для получения доступа к блокам библиотеки необходимо открыть окна библиотек *Library>Value.lix>Open Library Window* и *Library>Plotter.lix>Open Library Window*.

Есть два метода добавления нового блока к модели:

– выбор блока из его библиотеки в пределах меню *Library*;

– перетаскивание блока в модель с помощью мыши из окна библиотеки.

В модели бассейна используются следующие блоки: *Lookup Table*, *Random Number block*, *Math block*, *Holding Tank block*, *Plotter I/O block*. Для их добавления первым методом необходимо выбрать пункты меню:

– *Library>Value.lix>Math>Lookup Table*;
– *Library>Value.lix>Inputs> Random Number block*;
– *Library>Value.lix>Math> Math block*;
– *Library>Value.lix>Holding>Holding Tank block*;
– *Library>Plotter.lix>Plotter I/O block*.

В результате значки блоков появятся в верхнем углу в модельном окне. Для перемещения блоков в пределах модельного окна выберите блок и перетащите его мышкой или с помощью клавиш перемещения курсора в желаемую позицию. Далее свяжите блоки (см. рис. 1) с помощью линий соединителей.

Для сохранения модели выберите *File>Save Model As* и назовите файл. В имени файла желательно использовать только буквы латинского алфавита без пробелов.

Теперь, когда все блоки помещены и связаны в модели, необходимо задать настройки каждого блока.

Настройка блока *Lookup Table* 

. Блок описывает постоянный поток воды, поступающий в бассейн ежемесячно, и выводит объем воды, соответствующий текущему времени моделирования. Во вкладке *Table* диалогового окна блока задаются следующие настройки: *Lookup the*: *time*; *Output is*: *stepped* (значение по умолчанию); *Time units*: *months* (значение по умолчанию). Также необходимо ввести данные о ежемесячном притоке в таблицу (рис. 6). На вкладке *Options* задаются названия столбцов таблицы.

Настройка *time* означает, что блок сравнит текущее время моделирования со временем в таблице и выведет соответствующее значение.

Настройка *stepped* означает, что будут использованы точные значения, которые введены в таблицу, а не интерполированные.

	Month	Rainfall (inches)
0	0	2.6
1	1	4.4
2	2	6.7
3	3	3.4
4	4	1.9
5	5	1.1
6	6	0.7
7	7	0.5
8	8	0.4
9	9	0.7
10	10	2.6
11	11	3.4

Link

Рис. 6. Данные по месяцам

Левый столбец таблицы теперь определяет месяц, правый столбец определяет объем воды в дюймах. Данные таблицы должны повторяться каждые 12 месяцев, для этого в поле *Repeat table every* введите значение *12 months*.

Настройка блока *Random Number block* 

. В модели бассейна блок используется для описания случайного потока дождевой воды. Случайный поток задан равномерным законом распределения в диапазоне от 0 до 1: в бассейн каждый месяц добавляется равновероятно от 0 до 1 дюйма воды. В диалоговом окне блока введите следующие настройки: *Distribution*: *Uniform Real* (значение по умолчанию); *Minimum*: *0* (значение по умолчанию); *Maximum*: *1* (значение по умолчанию).

Настройка блока *Math*  . Блок *Math* складывает значения из двух входных источников (обозначено знаком «плюс» на его значке). Используются настройки блока по умолчанию.

Настройка блока *Holding Tank* 

. Блок представляет уровень воды в бассейне. Используются следующие настройки блока: *Initial contents*: *0* (значение по умолчанию); *Inputs are*: *integrated* (*delay*).

Настройка блока *Plotter*  . Блок отображает суммарный объем воды в бассейне, объем дождевой воды и объем постоянного потока в течение времени моделирования. В диалоговом окне блока щелкните текстовую метку *Value* в верхнем левом углу графика и введите «Дюймы» в текстовое поле. Измените другие метки в области окна графика следующим образом: *Plotter I/O* (расположенный наверху графика) измените на «Модель Бассейна»; *Time* (расположенный под графиком) измените на «Месяц».

Использование разных типов соединений. В модели бассейна для соединения блоков можно использовать либо правоугловой тип подключения (задан по умолчанию), либо прямой, либо мультисегментный. Тип подключения задается в пункте меню *Model>Connection Lines*. Недостаток первых двух типов подключения – загромождение и трудная читабельность модели (линии могут пересекать блоки).

Использование стиля мультисегмента имеет преимущество, разрешая расположить линии вокруг блоков. Рассмотрим использование стиля мультисегмента на примере соединения блоков *Lookup Table* и *Plotter*.

Выберите пункт меню *Model>Connection Lines* и опцию *straight line* (второй элемент). Щелкните на соединителе выхода блока *Lookup Table* и перетащите курсор до участка над *Holding Tank*, отпустите мышь. Это создаст первый сегмент (рис. 7). Курсор остается техническим пером и указывает на начало выделения.

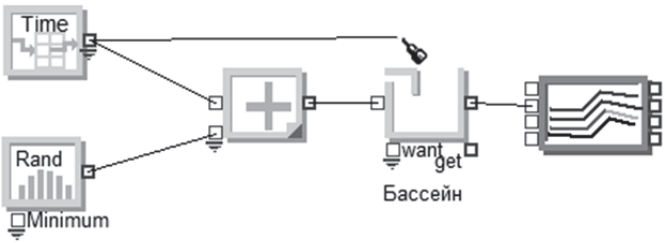


Рис. 7. Создание первого сегмента

Немедленно щелкните еще раз и перетащите курсор на входной соединитель на *Plotter*, отпустите кнопку мыши (рис. 8).

В результате создано подключение мультисегмента. Для удаления подключения необходимо дважды щелкнуть мышью на сегменте пока

вся линия подключения не станет толще, затем нажать *Delete* или *Backspace*. Чтобы удалить только один сегмент линии, необходимо щелкнуть на сегменте один раз и нажать клавишу *Delete*.

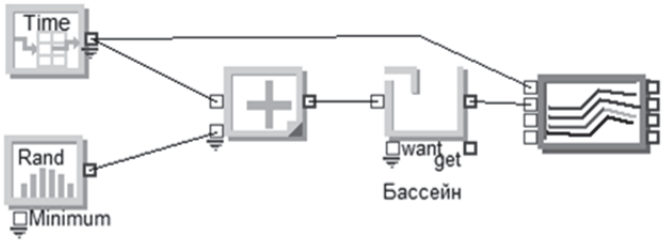


Рис. 8. Создание мультисегмента

Кроме вышеперечисленных способов соединений блоков также используют именные соединения. Именные соединения – текстовые метки, которые используются для того, чтобы представить выход (вход) блока в разных местах в модели. Именные соединения часто используют, чтобы не загромождать модель.

Рассмотрим последовательность создания именного соединения между блоками *Lookup Table* и *Plotter*:

– выберите пункт меню *Model>Connection Lines* и опцию правоугловой или прямой тип линии;

– дважды щелкните в модельном окне, немного выше и правее от соединителя выхода блока *Lookup Table*; в результате откроется текстовое поле;

– напечатайте «Постоянный поток» в текстовом поле и нажмите кнопку мыши где-нибудь в модельном окне;

– соедините соединитель выхода *Lookup Table* с меткой «Постоянный поток», перетаскивая линию от соединителя до текста, и, когда линия станет жирной, отпустите мышь;

– щелкните по метке «Постоянный поток», чтобы его выделить, затем выберите пункт меню *Edit>Duplicate*;

– перетащите дублированный текст к блоку *Plotter*;

– начертите линию между этим текстом и входным соединителем на *Plotter*.

Повторите последовательность действий для создания именного подключения между блоком *Random Number* и *Plotter*, как показано на рис. 1.

Выполнение моделирования. Если вы следовали всем инструкциям, то модель бассейна готова и соответствует рис.1. Для запуска процесса моделирования выберите пункт меню *Run>Run Simulation* или щелкните кнопкой *Run Simulation* на панели инструментов. Справа появится график процесса моделирования, на котором отображено три линии: красная – общий объем воды в бассейне; зеленая – объем дождевой воды; синяя – объем воды от постоянного потока. В конце моделирования *Plotter* автоматически масштабирует свою ось, чтобы отобразить все значения для всех столбцов данных. Однако диапазон изменения у данных разный, и график выглядит так, как показано на рис. 9.

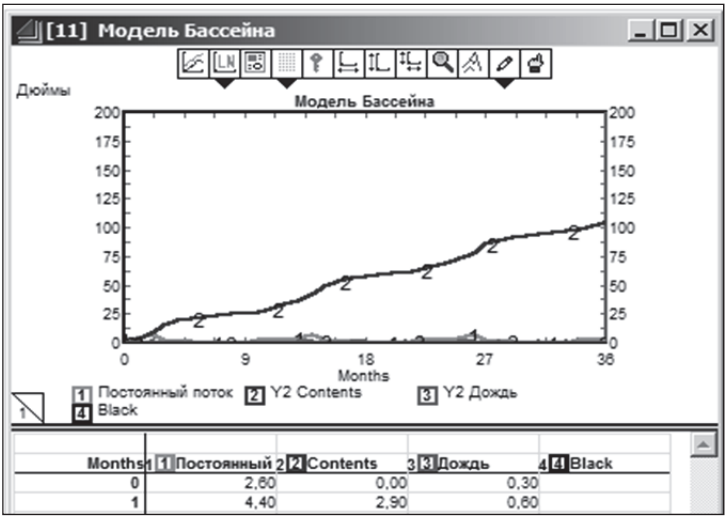


Рис. 9. Исходный вид графика

Построение графиков на нескольких осях. Для решения проблемы с разным масштабом графиков можно добавить отдельную ось (*Y2*) на правой стороне графика и отображать объем дождевой воды и объем постоянного потока на этой оси. Выполните следующую последовательность действий:

– дважды щелкните по блоку *Plotter*, если он не открыт;

– щелкните кнопкой *Trace properties* – крайняя левая на панели инструментов в окне графика, в результате откроется диалог с инструментами;

– щелкните кнопкой *Y1/Y2* (вторая справа в первой строке) с именем «Постоянный поток», в результате рисунок на кнопке зеркально отразится;

– щелкните кнопкой *Y1/Y2* в третьей строке, названной «Дождь», в результате рисунок на кнопке зеркально отразится;

– закройте диалоговое окно *Tools*;

– щелкните кнопкой *Run Simulation* на панели инструментов.

На графике теперь отображаются объем дождевой воды и объем постоянного потока на левой оси, а общий объем бассейна – на правой оси (рис. 10).

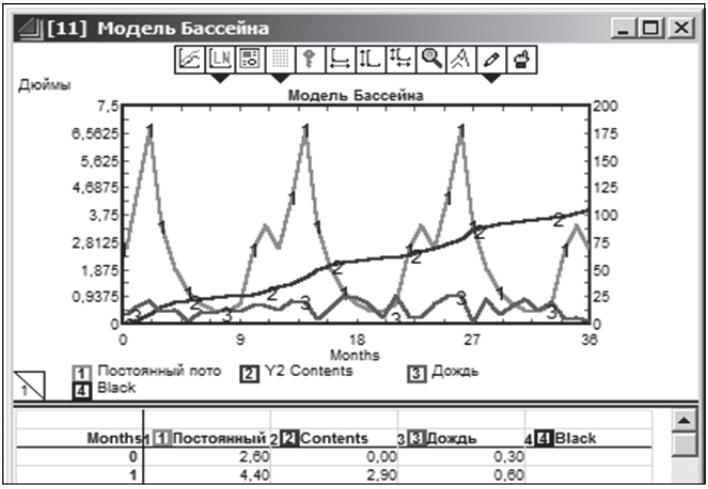


Рис. 10. Многоосный график

Дополнительные возможности. Введение в иерархию. В модели бассейна используется принцип «один блок – одна функция». Этот принцип применим для простых моделей, включающих относительно небольшое число блоков. Для больших моделей, состоящих из тысяч блоков, используется принцип иерархии. *ExtendSim* позволяет создавать иерархические блоки (*H-blocks*), которые собирают несколько блоков в один, разрешая пользователю получать доступ к блокам более низких уровней. Например, в модели бассейна можно сгруппировать блоки, представляющие источник воды вместе в один иерархический блок. Для этого выполните следующую последовательность действий:

– зажмите *Shift* и щелкните мышкой на блоках *Lookup Table*, *Random Number* и *Math*, чтобы выбрать их;

– выберите пункт меню *Model>Make Selection Hierarchical* и введите название для иерархического блока: «Источник воды», нажмите кнопку *Make H-Block*.

В результате три отдельных блока заменены единственным иерархическим блоком:  с тремя выходными соединителями. По умолчанию, у иерархического блока есть тень, чтобы отличить его от других блоков. Внешний вид блока можно изменить, выбрав пункт меню *Edit>Options*, вкладка *Model*.

Дважды щелкните на значке иерархического блока, чтобы видеть его подмодель или структуру (рис. 11).

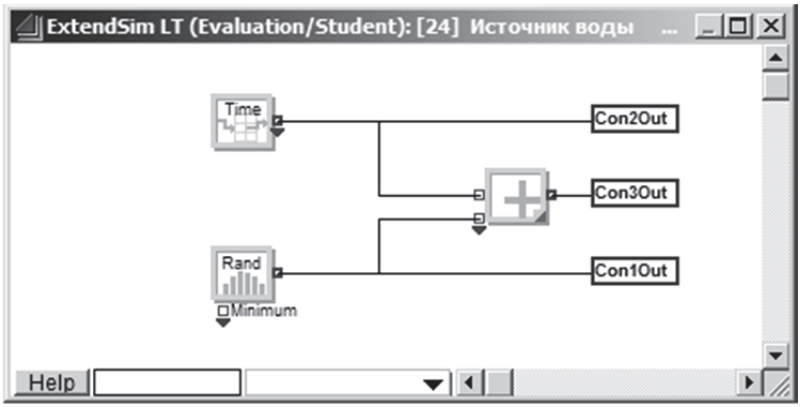


Рис. 11. Структура иерархического блока

В области заголовка окна выводится название иерархического блока. Заметьте, что подключения, передающие данные из иерархического блока к внешней модели, представлены в подмодели как именные соединения с красными границами вокруг текста. Эти подключения соответствуют трем соединителям на значке блока.

Также в *ExtendSim* можно построить иерархический блок «с нуля», выбрав пункт меню *Model>New Hierarchical Block*, и использовать индивидуальный значок блока.

Навигатор ExtendSim. Навигатор – подобное *Internet Explorer* окно, которое может использоваться:

– для передвижения по иерархической структуре модели;
– обращения к базам данных, используемых в модели;
– добавления блока к модельному рабочему листу, как альтернатива использования меню *Library*.

Выберите пункт меню *Window>Navigator* или щелкните кнопку *Open Navigator* на панели инструментов. По умолчанию Навигатор открывается в режиме *Model Navigator* со словом «*Model*», выбранным в крайнем левом всплывающем меню (рис. 12). Название активной модели приведено вверху окна, отображены значок каждого блока и информация о нем: название, метка и глобальный номер блока.

Щелкните знаком «плюс» около иерархического блока «Источник воды». Иерархический блок раскроется (рис. 12), чтобы показать блоки внутри него. Выберите блок *Lookup Table* в Навигаторе. Соответствующий блок выберется в модельном окне. Дважды щелкните блок *Lookup Table* в Навигаторе, в результате откроется диалоговое окно блока.

Навигатор необходим для того, чтобы работать со сложными многоуровневыми моделями и экземплярами класса иерархии.

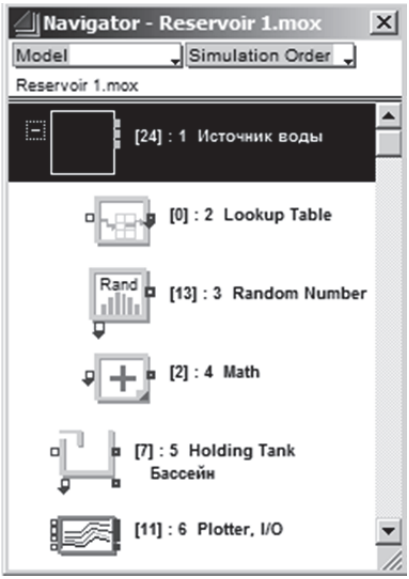


Рис. 12. Навигация по модели

Дублирование. *ExtendSim* позволяет добавлять элементы диалогового окна и/или графика в отчет, используя методику под названием «Дублирование». В результате создаются точные копии оригинальных элементов. При изменении дублируемого элемента оригинал также изменяется.

Рассмотрим дублирование графика блока *Plotter* в Отчет:

– выберите пункт меню *Window>Notebook* или выберите кнопку  *Open Notebook* на панели инструментов. В результате откроется пустое окно отчета *Notebook*;

– дважды щелкните мышью на блоке *Plotter* в модели бассейна;

– используя инструмент *Clone layer*  на панели инструментов, щелкните график *Plotter* и перетащите его в окно *Notebook*;

– выполните моделирование.

В результате график в отчете *Notebook* тот же самый, что и на *Plotter*.

На этом завершим краткий обзор возможностей и приемов создания простейших моделей в *ExtendSim.*

1.3. Контрольные вопросы и задания

1. Опишите последовательность разработки модели системы в *ExtendSim*.

2. Дайте общую характеристику среды *ExtendSim*, как средства разработки имитационной модели системы. Сравните *ExtendSim* с другими известными вам средствами разработки. В чем ее достоинства и недостатки?

3. Каково функциональное назначение блоков, входных и выходных коннекторов, соединителей в *ExtendSim*? Как выполнить настройку блоков, задать параметры моделирования?

4. Какие основные типы соединений реализованы в среде? В чем преимущества и недостатки каждого типа соединений?

5. Для чего используются иерархические блоки в среде? Как создать иерархический блок? Приведите примеры задач.

6. Модифицируйте построенную модель бассейна так, чтобы вода, превышающая объем бассейна, удалялась из модели. Постройте графики процесса моделирования.

2. ИНСТРУМЕНТАРИЙ *ExtendSim* ДЛЯ РАЗРАБОТКИ И ИССЛЕДОВАНИЯ ДИСКРЕТНО-СОБЫТИЙНЫХ МОДЕЛЕЙ

2.1. Разработка простейшей модели

Один из основных подходов к созданию имитационных моделей систем, который реализован в среде *ExtendSim*, – дискретно-событийное моделирование.

Система называется дискретно-событийной, если изменение состояний в ней происходит под влиянием явно определенных (дискретных) событий. Находясь в некотором состоянии, система сохраняет его (не изменяет своих характеристик) до наступления очередного события, под воздействием которого переменные системы (и, следовательно, ее состояние) изменяются «скачком». Термин «дискретно-событийное моделирование» исторически закрепился за моделированием прежде всего систем массового обслуживания (СМО) [6].

Понятие СМО охватывает широкий круг систем. СМО – любая динамическая система по обслуживанию заявок в условиях ограничения на ресурсы системы [3, 7]. Примерами СМО являются: библиотека, вычислительный центр, система передачи данных, информационная система, автоматизированная система управления, аэропорт, производственный участок, АЗС и т. д. Любая информационно-вычислительная система может быть представлена, формализованно описана и исследована как СМО.

При построении модели, например, вычислительной системы состояние системы может быть представлено количеством заявок в системе, числом занятых компьютеров. Состояние системы изменяется, если новая заявка поступает в систему или когда освобождается компьютер. Приход заявки или окончание обслуживания заявки – событие, которое изменяет состояние системы «скачком».

Реализованный в *ExtendSim* инструментарий для создания дискретно-событийной модели рассмотрим на примере простой задачи.

Пример 2. Известно, что вычислительная система состоит из одного компьютера. Интервал времени между двумя последовательными поступлениями заданий к компьютеру подчиняется равномерному закону распределения в интервале 6 ± 5 мин. Перед компьютером допу-

стима очередь заданий, длина которой не ограничена. Время выполнения задания также равномерно распределено в интервале 10 ± 9 мин. Промоделировать процесс обработки 100 заданий.

Для создания модели используются блоки двух библиотек: *Item.lix* (см. приложение) и *Plotter.lix*. Библиотека *Item.lix* является основной при моделировании дискретно-событийных систем, библиотека *Plotter.lix* содержит разные типы графиков. Конечный вид модели приведен на рис. 13.

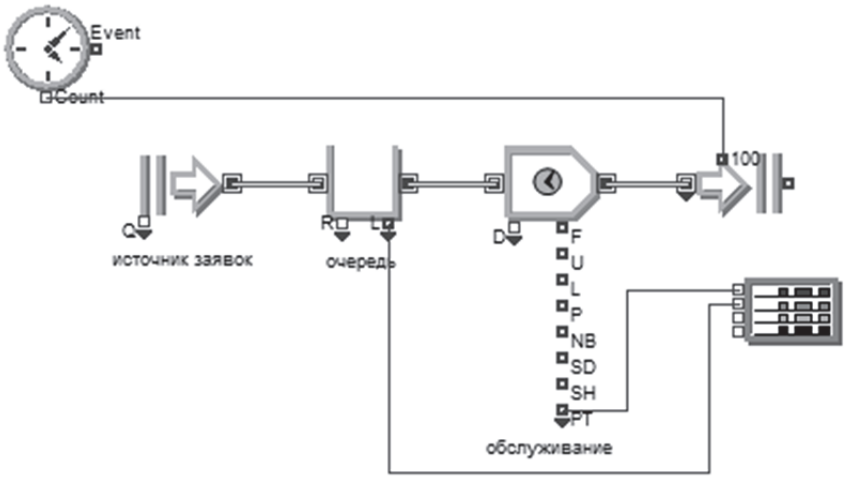


Рис. 13. Конечный вид модели

Рассмотрим построение модели по шагам. В модели используются следующие блоки: *Create*, *Exit*, *Queue*, *Executive*, *Activity*, *Plotter Discrete Event*. Остановимся подробно на настройках каждого из блоков.

В блоке *Executive* устанавливается общее время моделирования, моделирование завершается, когда через модель пройдут 100 заявок. В диалоговом окне блока выберите вкладку *Control*, выберите опцию: *Stop simulation*: *when count connector value*>= и введите значение 100.

В блоке *Create* устанавливается интервал между поступлением двух заявок и закон распределения интервала. Настройки блока *Create* приведены на рис. 14.

В блоке *Activity* устанавливается закон распределения интервала между поступлением двух заявок и параметры закона. На рис. 15 приведены настройки блока *Activity*.

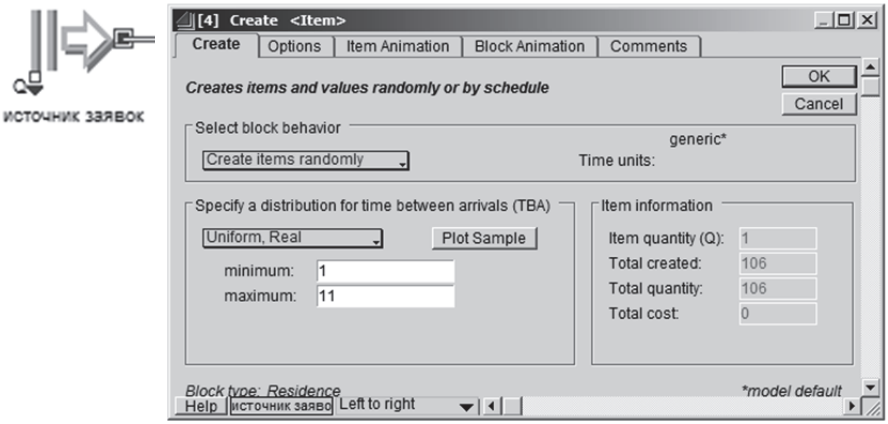


Рис. 14. Настройка блока Create

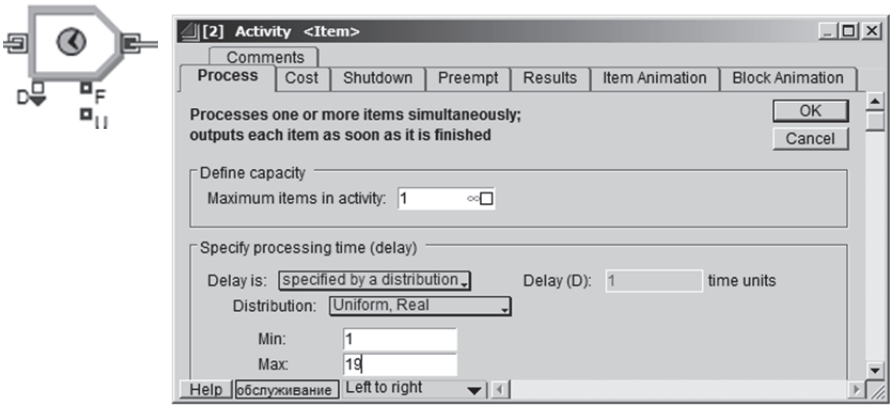


Рис. 15. Настройка блока Activity

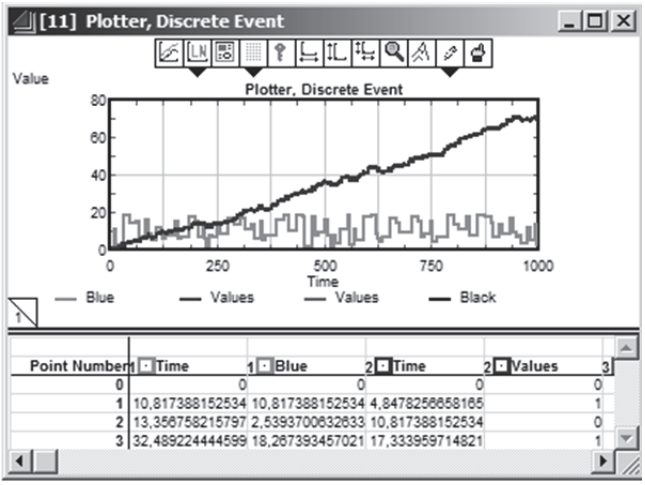
В настройках блоков *Exit*, *Queue*, *Plotter Discrete Event* используются параметры по умолчанию, и ничего изменять не нужно.

Обратите внимание на то, что блоки *Queue* и *Activity* содержат информационные выходные коннекторы, с которых можно снять информацию о характеристиках текущих (проходящих через модель) заявок или текущем состоянии блока. Например, у блока *Queue* следующие информационные выходные коннекторы: L – текущая длина очереди;

W – время ожидания в очереди текущей заявки; *F* – состояние очереди: возвращает 1, если очередь полна, 0 – в противном случае; *P* – приоритет текущей заявки.

У блока *Activity* следующие основные информационные выходные коннекторы: *L* – текущее количество обрабатываемых заявок; *U* – коэффициент загрузки; *PT* – время обработки текущей заявки и др. Подробную информацию о коннекторах блока можно прочитать, нажав кнопку *help* в диалоговом окне блока.

Выходные коннекторы блоков (рис. 13) соединяются с графиком *Plotter Discrete Event*. После запуска модели на выполнение (пункт меню *Run>Run Simulation*) выводятся график изменения выбранных характеристик системы в процессе моделирования (красная линия – средняя длина очереди, синяя линия – время обработки одной заявки) и временной ряд значений характеристик в табличном виде (рис. 16).



Point Number	Time 1	Blue 2	Time 2	Values 3
0	0	0	0	0
1	10,817388152534	10,817388152534	4,8478256658165	1
2	13,356758215797	2,5393700632633	10,817388152534	0
3	32,489224444599	18,267393457021	17,333959714821	1

Рис. 16. График изменения выходных характеристик

Выходные характеристики эффективности функционирования системы по результатам моделирования (например, коэффициент загрузки системы, среднее время обработки заявок, средняя длина очереди и т. п.) выводятся во вкладке *Results* в диалоговом окне блока. Например, для блока *Activity* выводятся следующие результаты (рис. 17): текущее, среднее и максимальное число заявок в устройстве (*Current*, *Average*, *Maximum Length*); текущее, среднее и максимальное время обработки

заявки (*Current*, *Average*, *Maximum Wait*); коэффициент использования устройства (*Utilization*); число заявок, зашедших в устройство (*Arrivals*); число заявок, вышедших из устройства (*Departures*).

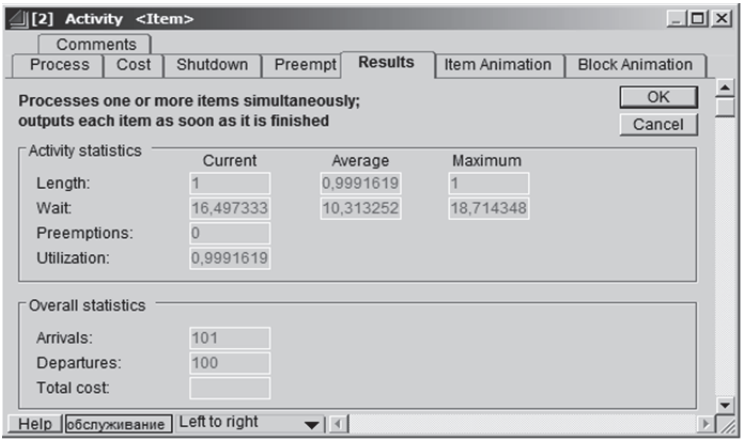


Рис. 17. Выходные характеристики блока Activity

Для блока *Queue* выводятся следующие результаты (рис. 18): текущее, среднее и максимальное число заявок в очереди (*Current*, *Average*, *Maximum Length*); текущее, среднее и максимальное время нахождения

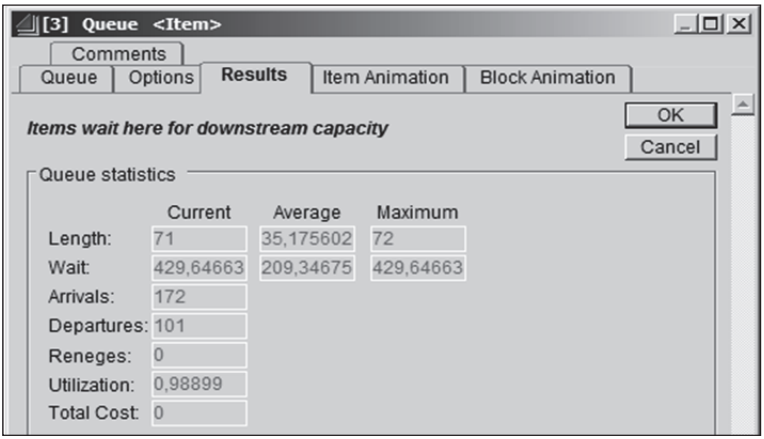


Рис. 18. Выходные характеристики блока Queue

заявки в очереди (*Current*, *Average*, *Maximum Wait*); коэффициент использования очереди (*Utilization*); число заявок, зашедших в очередь (*Arrivals*); число заявок, вышедших из очереди (*Departures*).

Очевидно, что моделируемая система не справляется с потоком заявок, так как средний интервал поступления (6 единиц времени) заявок меньше среднего интервала обработки заявки (10 единиц времени). Компьютер загружен на 99 %, очередь заявок линейно растет и на конец моделирования включает 71 заявку.

По результатам моделирования также можно сформировать итоговый текстовый отчет. Для этого в модель необходимо добавить блок 

из библиотеки *Value.lix*. Блок статистики не требует подключения и обеспечивает сбор итоговой статистики по блокам модели. Для формирования отчета выберите пункты меню *Run>Generate Report*, *Run>Report type*(*Statistics*)*/Statistics*, *Run>Add all to report*. В результате моделирования будет выведен текстовый отчет.

2.2. Инструменты 2D- и 3D-анимации процесса моделирования

Наряду с инструментами моделирования и статистического анализа результатов среда *ExtendSim* также предоставляет инструменты наглядной визуализации процесса моделирования. Реализована *2D*- и *3D*-анимация.

Настройки анимации задаются в диалоговом окне блока, вкладки *Item Animation* и *Block Animation*. Сначала настраивается анимация блока *Create*. Во вкладке диалогового окна *Item Animation* (рис. 19) задается: графический объект (картинка), который будет соответствовать и изображать заявку при *2D*- и *3D*-анимации; во вкладке *Block Animation* задается трехмерный графический объект, который будет изображать блок *Create* в окне *3D*, и задаются его настройки (размер, положение, цвет и т. п.). Таким образом, блоки представляют собой *3D*-объекты в окне трехмерной анимации *E3D*. По умолчанию они расположены там же, где расположены блоки самой модели в модельном окне. Однако если убрать в настройках анимации блока галочку с пункта *Link 2D/3D positions*, то трехмерный объект можно будет перемещать независимо от блоков модели.

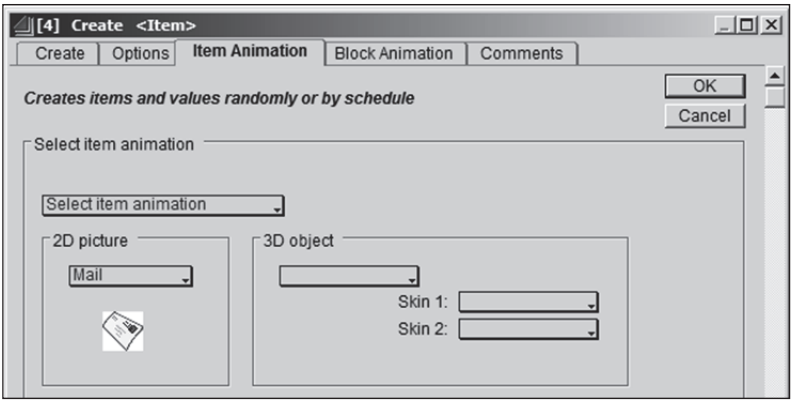


Рис. 19. Настройка анимации блока Create

Для того чтобы показать *2D*- и *3D*-анимацию, необходимо выбрать пункты меню *Run>Show2D animation* и *Run>Show3D animation*. Для открытия окна *3D* текущей модели нужно выбрать пункт меню *Window>E3D Window*. В появившемся окне будет отображаться вся анимация. Здесь же можно отредактировать окружение (нажать иконку в левом верхнем углу окна и выбрать *Editor*): изменить рельеф, добавить текстуры, отредактировать положение блоков и т. д.

Перемещаться по виртуальной *3D*-среде можно при помощи клавиш *W* (вверх), *A* (влево), *S* (вниз), *D* (вправо) или стрелок клавиатуры.

Если объект-заявка должен измениться (например, сырье -> готовый продукт) в процессе моделирования, то в блоке, который осуществляет преобразование, на вкладке *Item Animation* необходимо выбрать опцию *Change all items to*.

В отличие от блоков заявки имеют возможность перемещаться. Они могут это делать по конвейеру, если это заявки на производстве, или вдоль обозначенного пути, если это, например, клиенты банка. По умолчанию заявки движутся методом наикратчайшего пути от блока к блоку. Однако есть возможность задать собственную траекторию. Для этого необходимо разместить блок *Transport* из библиотеки *Item.lix*. Необходимую траекторию сначала нужно создать в редакторе *E3D*, а затем выбрать в настройках блока *Transport*, вкладка *Transport Animation*: *3D animation shows simultaneous item movement* опцию *along connections*.

2.3. Инструменты и способы генерации потоков поступления и обслуживания заявок

При имитационном моделировании необходим учет случайных факторов и воздействий на систему, которые представляются случайными последовательностями чисел. Математически случайные последовательности описываются в виде случайных событий, значений дискретных и непрерывных случайных величин, значений случайных векторов и процессов [2, 3, 6, 7–9]. Для описания случайных последовательностей используется аппарат теории вероятностей и математической статистики.

Программная имитация случайной последовательности любого типа основана на генерации базовой последовательности случайных чисел (СЧ) $\{x_i\} = x_0, x_1, ..., x_n$, представляющих собой выборку из равномерно распределенной на интервале [0, 1] генеральной совокупности значений величины ξ.

При разработке дискретно-событийных моделей один из ключевых моментов заключается в математическом описании случайных последовательностей (потоков) поступления и обслуживания заявок. Например: поток заданий на обработку в вычислительной системе; поток вызовов на телефонной станции; поток отказов (сбоев) компьютера в ходе его работы и т. д. Математически поток представляет собой в общем случае просто последовательность случайных точек $\theta_1, \theta_2, \ldots, \theta_n, \ldots$ на оси времени $0t$ с разделяющими их случайными интервалами τ_1, $\tau_2, \ldots, \tau_{n-1}, \tau_n, \ldots$, так что $\tau_1 = \theta_2 - \theta_1$, $\tau_2 = \theta_3 - \theta_2, \ldots, \tau_n = \theta_{n+1} - -\theta_n$, ….

Для математического задания потока необходимо описать закон распределения интервалов между поступлением заявок и их обслуживанием.

В среде реализованы разные способы задания потоков с помощью блоков *Create* и *Activity*.

В блоке *Create* может быть описано поступление заявок:

– через случайный интервал на основе задания его закона распределения (опция *Create items randomly* в диалоговом окне настройки блока) всего реализовано 37 базовых законов распределения;

– через интервал, определенный с помощью расписания (опция *Create items by schedule*);

– «плавно», заявки поступают по требованию (опция *Create items infinitely*); в этом случае блок *Create* связывается с блоком *Gate*, который открывает либо закрывает поток заявок.

В блоке *Activity*, описывающем длительность обработки заявок, доступны следующие опции задания интервала обслуживания:

– случайный интервал на основе задания его закона распределения (опция s*pecified by a distribution*);

– постоянный интервал (опция *constant*);

– интервал задается (поступает) через входной коннектор *D* (опция *from the* «*D*» *connector*);

– интервал задается из таблицы (опция *from a lookup table*);

– интервал определяется атрибутом (параметром) заявки (опция *an items attribute values*).

Рассмотрим несколько примеров, в которых иллюстрируются разные способы задания интервалов поступления и обслуживания заявок.

Пример 3. В систему поступает поток заявок по экспоненциальному закону распределения. Интенсивность поступления заявок зависит от времени суток: с 0 до 6 часов заявки поступают в среднем через 8 минут; с 6 до 12 часов – через 6 минут; с 12 до 18 часов – через 7 минут; с 18 до 24 часов – через 6 минут. Заявки обрабатываются одним из двух одинаковых устройств. Время обработки зависит от типа заявки: 20 % заявок обрабатывается за 0,1 часа, 80 % заявок обрабатывается за 0,12 часа. Необходимо промоделировать работу системы в течение 1000 дней.

Конечный вид модели изображен на рис. 20. Модель включает в себя следующие блоки из библиотеки *Item.lix*: *Create*, *Exit*, *Queue*, *Executive*, *Activity*, *Set Attributes*; из библиотеки *Value.lix* блоки: *Lookup table* и *Random Number*; из библиотеки *Plotter.lix* блок *Plotter Discrete Event*. Остановимся подробно на настройках каждого из блоков.

Время моделирования (*End time=1000*) и глобальные единицы времени (*Global time units=hours*) задаются в диалоговом окне *Simulation Setup* (пункт меню *Run>Simulation Setup*). В диалоговом окне блока *Executive*, вкладка *Control*, устанавливается окончание времени моделирования по времени: *Stop simulation*: *at end time*.

Входной поток заявок и его параметры описываются блоками *Create* и *Lookup table*. В блоке *Lookup table* задается изменение интенсивности входного потока заявок с течением времени. Во вкладке *Table* диалого-

вого окна блока устанавливаются опции: *Lookup the*: *time*; *Output is*: *stepped*; *Time units*: *hours* (глобальные единицы времени – часы). Изменение среднего интервала между поступлением заявок в течение времени описывается таблицей (рис. 21).

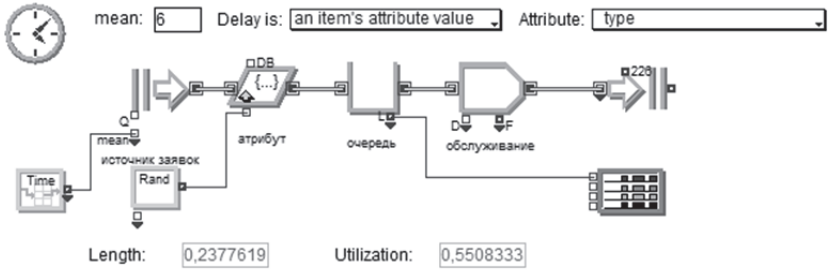


Рис. 20. Модель системы (пример 3)

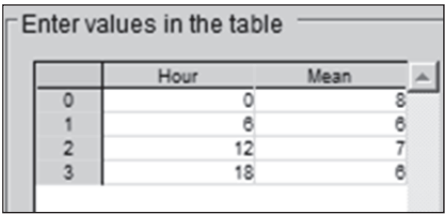


Рис. 21. Задание параметров блока Lookup table

В первом столбце задается момент времени в часах, в который меняется интенсивность входного потока, во втором столбце – средний интервал между заявками в минутах. Единицы измерения времени в первом столбце определяются заданными глобальными единицами времени, а во втором – определяются настройками блока, связанного с *Lookup table*, в этом случае блоком *Create*. Данные таблицы повторяются каждые 24 часа, режим: *Repeat table every 24 hours*.

В блоке *Create* устанавливаются настройки: *Create items randomly*, *Disribution exponential* с параметрами *mean* = 1, *location* = 0 и *Time unit*: *minutes*. В поле *mean* можно задать любое значение, так как это значение будет переопределно в процессе моделирования в соответствии с таблицей блока *Lookup table*.

Далее заявка направляется в блок *Set Attributes*, где определяется ее тип. В настройках блока в поле *Property Name* выбирается из выпада-

ющего списка *New Value Attributes* и вводится имя атрибута (*type*). В блоке *Random Number*, связанном с блоком *Set Attributes*, задается эмпирическая таблица вероятностей, определяющая время обработки заявки в зависимости от ее типа (рис. 22).

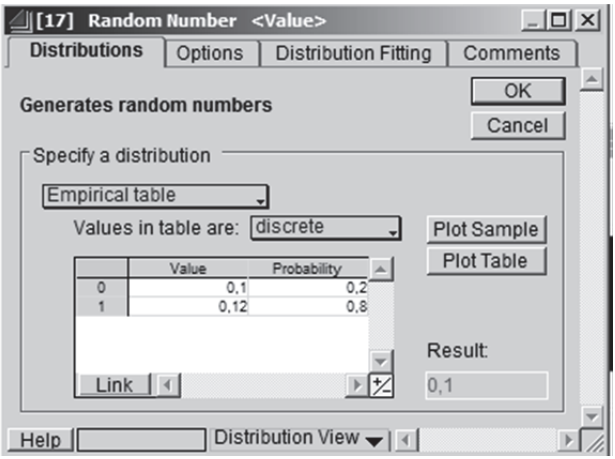


Рис. 22. Настройки блока Random Number

В блоке *Queue* используются настройки по умолчанию.

В блоке *Activity*, описывающем процесс обработки, устанавливаются настройки: *Maximum items in activity* = 2 (число устройств), *Delay is*: *an item's attribute value* (время обработки определяется типом заявки), *Attribute*: *type* (имя атрибута заявки); *Delay D*: 0,1 *hours* (начальное время обработки).

В процессе моделирования бывает необходимо отслеживать изменение значений входных и выходных переменных модели, например, изменение среднего интервала поступления заявок (*mean*) или изменение длины очереди во времени. В *ExtendSim* реализована возможность вынести настройки модели, входные переменные, параметры и результаты моделирования в область модельного окна.

Для модели примера 3 (см. рис. 20) в модель вынесены входные настройки модели: *mean*; *Delay is*: *an item's attribute value*; *Attribute*: *type* и результаты моделирования: *Length* (средняя длина очереди), *Utilization* (коэффициент загрузки устройства).

Для копирования элементов в область окна модели в диалоговом окне блока надо щелкнуть правой кнопкой мыши на любом элементе

окна, и в выпадающем списке выбрать *Clone tool*. В результате все элементы диалогового окна выделяются и их можно перетащить в область модели с помощью мыши.

После запуска модели на выполнение отображается график изменения средней длины очереди во времени (рис. 23) и выводятся в соответствующих полях в модели (см. рис. 20) средняя длина очереди (*Length* = 0,24) и коэффициент загрузки устройства (*Utilization* = 0,55).

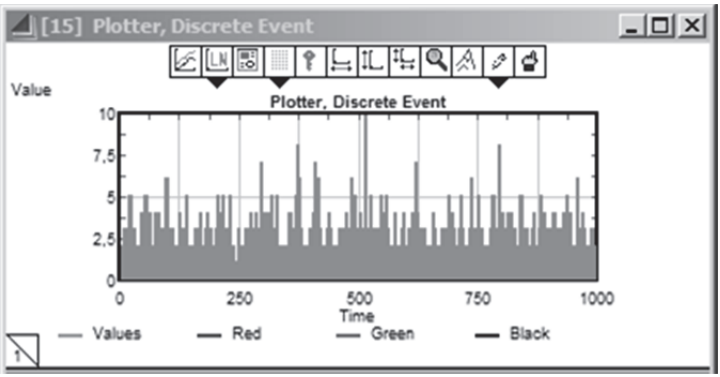


Рис. 23. График изменения длины очереди (пример 3)

В целом исследуемая система справляется с потоком заявок, загружена примерно на 50 %, время ожидания заявок в очереди в сравнении с интервалом обслуживания – незначительно.

В приведенном выше примере поток заявок описывался стандартным экспоненциальным законом распределения, моделирование последовательности значений которого реализовано в *ExtendSim*. Может также возникнуть ситуация, когда поток поступления или обслуживания заявок описывается законом, не реализованным в среде пакета. В этом случае реализуют моделирование на базе использования одного из методов имитации последовательности значений случайных величин с заданным законом распределения (метод обратной функции, метод Неймана, метод кусочной аппроксимации функции плотности распределения вероятностей и другие) [6, 7]. Приведем пример моделирования потока заявок с использованием метода обратной функции.

Пример 4. В систему поступает поток заявок, интервалы между которыми описываются СВ Y, распределенной по закону, заданному функцией плотности распределения вероятности: $f(y) = 2y,\ y \in [0,1]$. Далее

заявки обслуживаются одним каналом в течение одной единицы времени. Смоделировать обработку заявок в течение 100 единиц времени.

Согласно методу обратной функции найдем выражение для функции распределения вероятностей СВ Y и приравняем к случайному числу, распределенному равномерно на [0,1]:

$$F(y) = \int_0^y f(y)dy = y^2 = x_i \,. \qquad (1)$$

Из (1) найдем выражение для y_i:

$$y_i = \sqrt{x_i} \,. \qquad (2)$$

Выбран положительный корень с учетом диапазона изменения СВ Y. Конечный вид модели изображен на рис. 24.

Рис. 24. Модель системы (пример 4)

В блоке *Random Number* генерируется случайное число, равномерно распределенное (*Uniform*, *Real*) в диапазоне от 0 до 1. Затем в блоке *Equation* (*Value.lix*) происходит вычисление значения СВ Y в соответствии с формулой (2). В диалоговом окне блока в поле формул задается выражение: *outCon0=sqrt*(*inCon0*). В блоке *Create* устанавливаются настройки: *Create items randomly*, *Disribution Constant*. В поле *constant* можно задать любое значение, так как это значение будет переопределено в процессе моделирования в соответствии со значением, считанным с выходного коннектора блока *Equation*.

В результате моделирования обслужено 100 заявок, средняя длина очереди заявок равна 25,7; коэффициент использования устройства: 1.

Система с потоком заявок не справляется, так как интервал поступления заявок меньше или равен интервалу обслуживания заявки, и в системе накапливаются заявки в очереди на обслуживание в процессе моделирования.

2.4. Инструменты для моделирования событий

Целый класс задач моделирования связан с имитационным моделированием событий, например, оценка отказоустойчивости сетевого узла, оценка надежности кластерной вычислительной системы и т. п. В основе моделирования лежит имитация элементарного события.

Имитация элементарного события. Пусть необходимо реализовать случайное событие *A*, наступающее с заданной вероятностью *p*. Это могут быть отказ элемента или узла устройства, поступление сообщения, поражение цели и т. п. Определим *A* как событие, состоящее в том, что выбранное значение x_i равномерно распределенной на интервале [0,1] случайной величины удовлетворяет неравенству

$$x_i \le p\ . \qquad (3)$$

Тогда вероятность события *A* будет $P(A) = \int_0^p dx = p.$

Противоположное событие состоит в том, что $x_i > p$, его вероятность равна 1 – *p*.

Пример 5. Пусть нам известна вероятность отказа устройства $P_0 = 0{,}1$. Смоделировать выпадение этого события можно, разыграв равномерно распределенное случайное число из диапазона от 0 до 1 и установив, в какой из двух интервалов (от 0 до 0,1 или от 0,1 до 1) оно попало (рис. 25). Если число попадает в диапазон (0; 0, 1], то устройство отказало, т. е. событие произошло, если иначе – событие не произошло (устройство работает). При значительном числе экспериментов частота попадания чисел в интервал от 0 до 0,1 будет приближаться к теоретической вероятности $P_0 = 0{,}1$, а частота попадания чисел в интервал (0,1] будет приближаться к $P_\text{р} = 0{,}9$.

Имитация полной группы событий. Несовместные события составляют полную группу, если суммарная вероятность их появления равна единице.

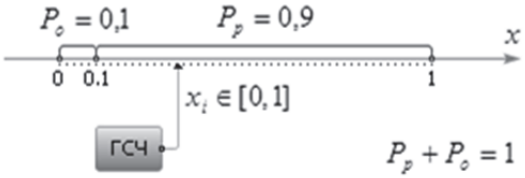


Рис. 25. Имитационное моделирование простого события

Рассмотрим группу событий. Пусть $A_1, A_2, \ldots, A_n$ – полная группа событий, наступающих с вероятностями $p_1, p_2, \ldots, p_n$ соответственно. Определим событие A_{m+1} как событие, состоящее в том, что выбранное значение x_i случайной величины, равномерно распределенной на интервале [0,1], удовлетворяет неравенству

$$l_m < x_i \le l_{m+1},\ 0 \le m \le n-1,\ l_0 = 0, \tag{4}$$

где $l_{m+1} = \sum_{i=1}^{m+1} p_i$.

Процедура моделирования испытаний в этом случае состоит в последовательном сравнении случайных чисел x_i со значениями l_m, l_{m+1}. Если условие выполняется, исходом испытания оказывается событие A_{m+1}.

Описанный алгоритм называют алгоритмом «розыгрыша по жребию» (рис. 26).

Имитация сложного события, состоящего, например, из двух независимых элементарных событий *A* и *B*, заключается в проверке неравенств

$$\begin{cases} x_{i1} \le p_A, \\ x_{i2} \le p_B. \end{cases} \tag{5}$$

Здесь x_{i1} и x_{i2} – случайные числа, равномерно распределенные на интервале [0, 1]; p_A, p_B – вероятности наступления соответственно событий *A* и *B*.

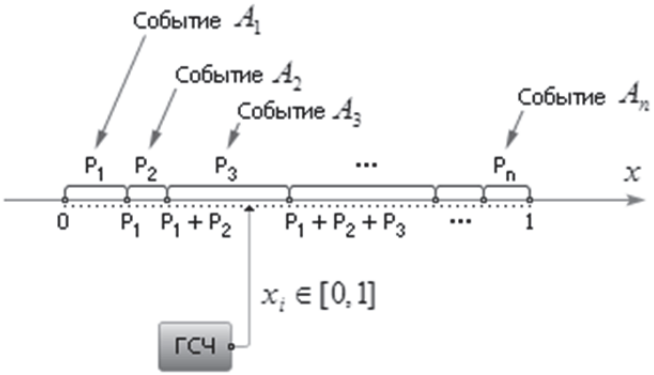


Рис. 26. Имитационное моделирование полной группы событий

В зависимости от исхода проверки неравенств делается вывод, какой из вариантов сложного события: AB, $\overline{AB}$, $A\overline{B}$, $\overline{A}B$ имеет место.

Имитация зависимых событий. В случае, когда сложное событие состоит из элементарных зависимых событий A и B, имитация сложного события производится с помощью проверки следующих неравенств:

$$\begin{cases} x_{i1} \le p_A, \\ x_{i2} \le p_{B/A}; \end{cases} \quad \begin{cases} x_{i1} \le p_A, \\ x_{i2} > p_{B/A}; \end{cases} \quad \begin{cases} x_{i1} > p_A, \\ x_{i2} \le p_{B/\overline{A}}; \end{cases} \quad \begin{cases} x_{i1} > p_A, \\ x_{i2} > p_{B/\overline{A}}. \end{cases} \qquad (6)$$

В зависимости от того, какая из этих четырех систем неравенств выполняется, делается вывод о том, какой из четырех возможных исходов (AB, $A\overline{B}$, $\overline{A}B$, $\overline{AB}$) имеет место.

Рассмотрим примеры имитации событий средствами системы *ExtendSim*.

Пример 6. Оценить надежность устройства, состоящего из двух узлов и элементов A, B, C (рис. 27). Узел выходит из строя, когда выходят из строя все элементы, входящие в узел. Устройство выходит из строя, когда отказывает хотя бы один из его узлов. Вероятности безотказной работы элементов равны соответственно: $P(A) = 0{,}8$; $P(B) = 0{,}7$; $P(C) = 0{,}6$. Рассчитать вероятность безотказной работы устройства.

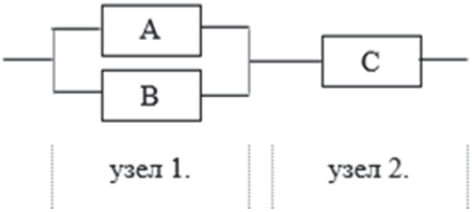


Рис. 27. Схема устройства

Ниже приведены два варианта реализации модели работы устройства в среде *ExtendSim* (рис. 28 и 29).

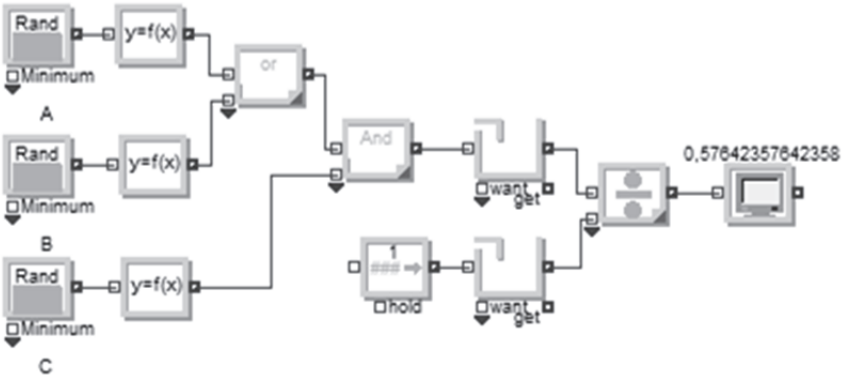


Рис. 28. Схема устройства (вариант модели 1)

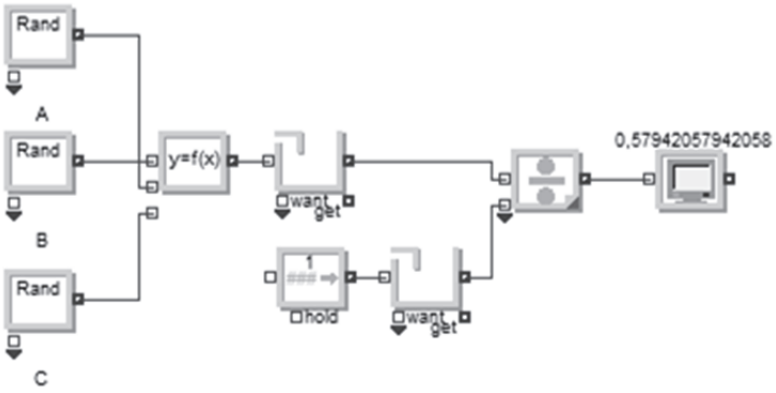


Рис. 29. Схема устройства (вариант модели 2)

При построении модели используются блоки, находящиеся в библиотеке *Value.lix*:

Random Number генерирует случайное число по заданному закону, например, генерирует либо 0, либо 1 с заданными вероятностями.


Math выполняет заданную математическую или логическую операцию над входными операндами и выдает результат.

Equation вычисляет значение заданной функции.

Holding Tank накапливает (суммирует) входные значения.

Constant генерирует постоянное заданное значение на каждом шаге. Значение задается в диалоговом окне настройки блока. Если вход (слева) подключен, то входное значение добавляется к постоянному (заданному в блоке), и на выходе появляется сумма этих значений.


Display Value отображает входное значение.

В первом варианте модели в блоках *Random Number*, соответствующих элементам устройства *A*, *B*, *C*, генерируется случайное число, равномерно распределенное (*Uniform*, *Real*) в диапазоне от 0 до 1. Далее в блоках *Equation* сравнивается сгенерированное число с заданной вероятностью работы элемента устройства. Например, для элемента *A* в диалоговом окне блока *Equation* в поле формул задается выражение *outCon0* = *inCon0* < 0.8, переменные *outCon0* и *inCon0* соответствуют значениям на выходном и входном коннекторах блока. На выходе блоков *Equation* значение либо 0 – элемент не работает, либо 1 – элемент работает. В блоках *Math* реализуется логика работы устройства:

And – И; *or* – ИЛИ. В верхнем блоке *Holding Tank* накапливается суммарное количество раз, когда устройства работало (*n*), в нижнем – общее количество тактов симуляции (*m*). Вероятность работы устройства рассчитывается как отношение *n/m*, для чего используется блок *Math* в режиме *Divide*. Рассчитанная вероятность отображается с помощью блока *Display Value*. Количество тактов симуляции работы устройства задается в диалоговом окне *Simulation Setup* (пункт меню *Run/Simulation Setup*) в поле *End time*.

Во втором варианте модели используется меньше блоков. В блоках *Random Number* задается режим генерации случайных чисел в соответствии с эмпирической таблицей: *Empirical Table*. В таблице задаются значения (*Value*) и вероятность их генерации (*Probability*). Например, для элемента *A*: 0 – 0.2; 1 – 0.8. Таким образом, на выходе блока либо 0 – элемент не работает, либо 1 – устройство работает. Далее в блоке *Equation* задается логика работы устройства: *outCon0* = (*A* or *B*) and *C*.

В результате моделирования работы устройства 1000 раз обе модели выдают близкие значения оценки вероятности его безотказной работы: 0,5764 (вариант 1) и 0,5794 (вариант 2). Рассчитаем теоретическую вероятность работы устройства:

$$p = (1 - (1 - 0,8) \cdot (1 - 0,7)) \cdot 0,6 = 0,576.$$

Таким образом, абсолютная погрешность оценки вероятности имитационными моделями невелика и составляет 0,0004 (вариант 1) и 0,0034 (вариант 2) по результатам одного прогона имитационной модели.

2.5. Инструменты для статистической обработки результатов моделирования

В ходе имитационного моделирования формируются выборочные данные, описывающие процесс и результаты функционирования системы. Выборочные данные являются исходным статистическим материалом для нахождения приближенных значений (оценок) показателей эффективности функционирования изучаемой системы. Например, в качестве такого показателя могут выступать: время пребывания заявки в системе; время нахождения заявки в очереди на обслуживание; время обработки заявки и ряд других характеристик.

По результатам имитационного моделирования рассчитывают оценки функциональных характеристик и оценки выборочных числовых

характеристик показателей эффективности системы: оценка математического ожидания (выборочное среднее); выборочная медиана; выборочная мода; оценка дисперсии; оценка среднеквадратического отклонения; выборочные минимальное и максимальное значения; выборочный размах; коэффициент вариации и т. д.

По выборочным данным может быть построена эмпирическая функция распределения вероятностей и эмпирическая функция плотности распределения вероятностей (гистограмма), которые являются выборочными аналогами соответствующих теоретических характеристик. При обработке результатов машинного эксперимента с моделью системы часто возникает задача определения эмпирического закона распределения выходной характеристики. Для этого строят гистограмму и на основе ее анализа выдвигают гипотезу о типе закона распределения характеристики. Проверка гипотезы реализуется с помощью критериев согласия (Колмогорова–Смирнова, χ^2-Пирсона, Крамера–Мизеса–Смирнова и других).

Кроме точечных оценок, используют также интервальные оценки характеристик. Интервальной называют оценку, которая определяется двумя числами – концами интервала, покрывающего оцениваемый параметр. Доверительным называют интервал, который с заданной вероятностью покрывает заданную характеристику. Например, строят доверительный интервал для среднего значения или дисперсии характеристики.

Система *ExtendSim* предоставляет несколько блоков, обеспечивающих сбор и вывод статистических данных. Основные блоки статистики следующие.

Clear Statistics (*Value.lix*) – очищает статистику других блоков, устраняя статистические искажения переходного периода.

Display Value (*Value.lix*) – отображает и выводит значение входящей в блок величины.

History (*Item.lix*) – записывает информацию о заявках и их свойствах, таких как значение атрибута, время прибытия заявки в систему, ее приоритет и т. д.

Information (*Item.lix*) – отображает статистику по заявкам, такую как количество заявок, время пребывания их в системе, время между поступлениями заявок.

Mean&Variance (*Value.lix*) – рассчитывает среднее, дисперсию, среднеквадратическое отклонение и доверительный интервал по входящим в блок данным.

Statistics (*Value.lix*) – подводит статистику для определенного типа блока, таких как *Activity* или *Queue*; результаты организуются в таблицу; информация собирается с использованием определенного статистического метода, который задается пользователем.


Histogram (*Plotter.lix*) – строит гистограмму по входным данным.

В начале моделирования очереди часто пусты и в системе нет данных для статистической обработки. После того как модель выполнялась в течение какого-то времени после старта, она начинает функционировать наиболее сходно с реальной системой на нормальном эксплуатационном уровне. Интервал от момента запуска системы до момента, когда система начинает функционировать в устойчивом или нормальном состоянии, называется переходным периодом. Блок *Clear Statistics* используется для сброса накопителей статистической информации определенных блоков, устраняя тем самым статистические ошибки переходного периода.

Построение модели рекомендуется выполнять поэтапно, проверяя на каждом этапе, что построенная часть модели работает корректно, и затем уже дополнять и усложнять модель. Блок *History* особенно полезен для проверки модельных данных, поскольку он предоставляет информацию о каждой заявке в течение моделирования.

Блок *History* в модель подключается последовательно, размещается между другими блоками так, чтобы заявки проходили через него.

Каждой заявке, проходящей через блок, выделяется строка в таблице истории. Первая колонка таблицы отображает время поступления заявки в систему. Также можно вывести значение атрибута, свойства заявок и отобразить даты календаря.

Блок *Mean & Variance* может рассчитывать как взвешенную по времени, так и наблюдаемую статистику:

– если включен флажок *Use time weighted statistics* в диалоговом окне блока *Mean&Variance*, тогда рассчитываются среднее, дисперсия и среднеквадратическое отклонение при помощи взвешивания входных данных по времени моделирования. Это достигается делением входного значения на длительность наблюдения этого входного значения и затем суммированием полученных результатов по ходу моделирования;

– если флажок *Use time weighted statistics* не включен, сумма входных значений (например, при расчете среднего) будет поделена на общее количество полученных входных значений, что будет являться наблюдаемой статистикой.

Используя блок *Mean&Variance*, важно определить, какой тип статистики необходим, и следовать следующим рекомендациям:

– если характеристика, по которой собирается статистика, имеет значение в каждой точке моделирования, используется взвешенная статистика. Типичный пример такой характеристики – количество заявок в очереди (снимается с информационного коннектора *L*). В любой момент моделирования в блоке находится определенное количество заявок. Для определения средней величины очереди следует применить взвешивание по времени;

– если характеристика, по которой собирается статистика, имеет значение только при определенных событиях, не следует использовать взвешенную статистику. Например, время ожидания в очереди (снимается с коннектора *W*). Для расчета среднего времени ожидания в очереди взвешивание по времени не нужно.

Блок *Statistics* накапливает данные и рассчитывает статистику для определенного типа блока, используя определенный статистический метод. Вдобавок к номеру блока, его имени и времени сбора информации блок отображает метрики, специализированные для данного типа блока, такие как использование или среднее время обработки для блоков типа *Activity*, или средние, дисперсия и среднеквадратическое отклонение для всех *Mean&Variance* блоков модели. Модели с очередями могут использовать блок *Statistics* для сбора информации об очередях.

Пример 7. В систему поступают задания в среднем через 10 мс, которые обрабатываются одним компьютером в среднем 11 мс. Закон распределения времени поступления и обслуживания заданий – экспоненциальный. Время моделирования составляет 500 мс. Определить статистические характеристики и построить гистограмму времени пребывания задания в системе.

Конечный вид модели изображен на рис. 30.

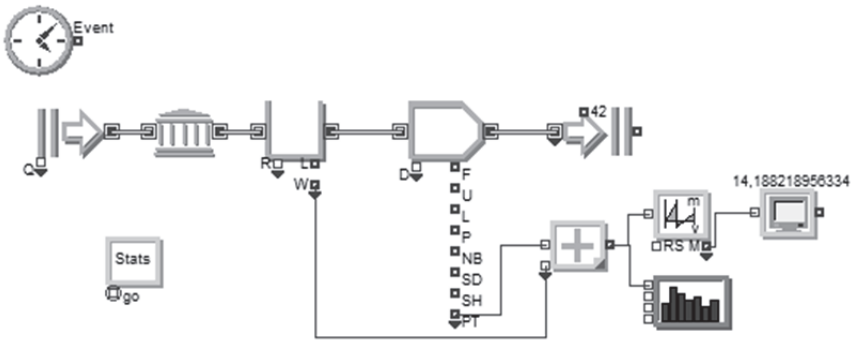


Рис. 30. Модель системы (пример 7)

В модели с информационных коннекторов блоков *Queue* (*L*) и *Activity* (*PT*) снимается информация соответственно о времени нахождения задания в очереди и на обработке. Время пребывания задания в систему рассчитывается как сумма этих времен, для расчета используется блок *Math* (*Value.lix*). В блоке *Mean&Variance* используются настройки по умолчанию и рассчитывается невзвешенная статистика.

После запуска модели на выполнение и окончания моделирования в диалоговом окне блока *History* выводится время поступления каждого задания в систему. В блоке *Mean&Variance*, вкладка *Results* выводятся статистические характеристики времени пребывания задания в системе (рис. 31): среднее (*mean*), дисперсия (*variance*), среднее квадратическое отклонение (*standard deviation*), количество наблюдений (*number of observations*), доверительный интервал 95 % (*Confidence interval +/–*), доля доверительного интервала от среднего (*Relative CI error*). Во вкладке *History* выводится таблица с выходными данными, в которой первый столбец (*Time*) – время, прошедшее с начала моделирования, второй столбец (*Value*) – время пребывания задания в системе.

По результатам моделирования строится гистограмма времени пребывания задания в системе (рис. 32).

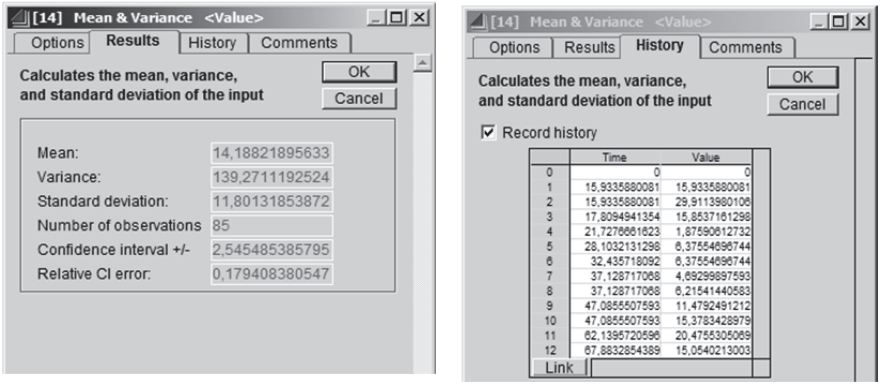


Рис. 31. Результаты моделирования (пример 7)

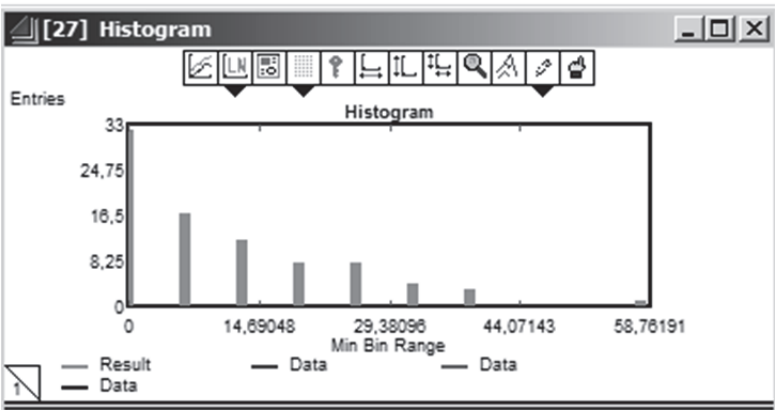


Рис. 32. Гистограмма времени пребывания задания в системе (пример 7)

Для проведения более углубленного статистического анализа выходные результаты моделирования могут быть экспортированы в любой статистический пакет (например, *Statistica*) для дальнейшей обработки.

Также в среду *ExtendSim* встроен модуль *StatFit* (*Run>Launch StatFit*), в котором реализованы расчет основных статистических характери-

стик, критерии согласия для проверки гипотезы о типе закона распределения, методы оценки параметров распределения и многое другое. К сожалению, в студенческой версии *ExtendSim* введено ограничение на использование модуля: объем выборочных данных не должен превышать 50 наблюдений. Как правило, объемы выходных выборок в несколько раз больше, поэтому возможностей модуля недостаточно для решения практических задач.

2.6. Инструменты анализа чувствительности модели

Анализ чувствительности модели определяет оценку влияния колебаний значений входных переменных на отклики (выходные) характеристики модели [7]. Необходимо установить, при каком разбросе входных данных сохраняется справедливость основных выводов, сделанных по результатам моделирования.

Под анализом чувствительности понимается определение чувствительности наших окончательных результатов моделирования к изменению используемых значение входных переменных и параметров модели. Анализ определяет, как меняется выходная переменная Y при небольших изменениях различных параметров модели или ее входов X.

Простота проведения анализа чувствительности в имитационном моделировании – одно из преимуществ этого метода. Оценка чувствительности является исключительно важной процедурой и подготовительным этапом перед планированием имитационного эксперимента.

Дело в том, что величины параметров систематически варьируются в некоторых представляющих интерес пределах ($X_{\min}; X_{\max}$) и наблюдается влияние этих вариаций на характеристики системы ($Y_{\min}; \ Y_{\max}$). Если при незначительных изменениях величин некоторых параметров результаты меняются очень сильно, – это основание для затраты большого количества времени и средств с целью получения более точных оценок. И, наоборот, если конечные результаты при изменении величин параметров в широких пределах не изменяются, то дальнейшее экспериментирование в этом направлении бесполезно и неоправданно. Поэтому очень важно определить степень чувствительности результатов относительно выбранных для исследования величин – параметров.

Исследование чувствительности является предварительной процедурой перед планированием эксперимента и позволяет определить стратегию планирования экспериментов на имитационной модели.

Этой информации бывает достаточно для ранжирования компонент вектора параметров модели X по значению чувствительности вектора отклика модели. Если модель отказывается малочувствительной по какой-либо q-й компоненте вектора параметров модели X_q, то, как правило, не включают в план имитационного эксперимента переменную X_q, чем достигается экономия ресурса времени моделирования.

Анализ чувствительности поможет также внести коррективы в разрабатываемую модель, т.е. упростить, например, перейти от использования закона распределения к использованию среднего значения переменной, а некоторые подсистемы вообще отбросить (или процессы не детализировать). И наоборот, анализ чувствительности может показать, какие части модели было бы полезно разобрать более детально.

Чувствительность имитационной модели представляется величиной минимального приращения выбранного критерия качества, вычисляемого по статистикам моделирования, при последовательном варьирования параметров моделирования на всем диапазоне их изменения.

Методика оценки чувствительности [7]

По каждой входной переменной X_q определяется интервал изменения ($\min X_q, \max X_q$). Далее изменяют по очереди каждую q-ю переменную, а остальные переменные при этом не изменяются и соответствуют центральной точке. Проводят модельные эксперименты и получают отклики модели ($\min Y_q, \max Y_q$). Для оценки чувствительности используют абсолютные значения или относительные. В последнем случае вычисляют приращение вектора входных параметров:

$$\delta X_q^0 = \frac{(\max X_q - \min X_q)^2}{(\max X_q + \min X_q)} 100\,\% \qquad (7)$$

и вычисляют приращение вектора отклика:

$$\delta Y_q = \frac{(\max Y_q - \min Y_q)^2}{(\max Y_q + \min Y_q)} 100\,\%. \qquad (8)$$

Выбирают $\delta Y_q^0 = \max\{\delta Y_q\}$.

Итак, чувствительность модели по q-й компоненте вектора параметров определяют парой значений $\left\{\delta X_q^0, \delta Y_q^0\right\}$.

Анализ чувствительности модели реализован в среде *ExtendSim*. Техника проведения анализа чувствительности состоит в изменении выбранных параметров (входных переменных) в определенных пределах при условии, что остальные параметры остаются неизменными. Рассмотрим последовательность проведения анализа чувствительности на примере простейшей модели.

Пример 8. В систему поступают задания в среднем через 4 мс, которые обрабатываются одним компьютером в среднем 8 мс. Закон распределения времени поступления и обслуживания заданий – экспоненциальный. Оценить, как влияет изменение интенсивности входного потока заданий на среднее время пребывания задания в очереди на обработку. Время моделирования составляет 400 мс.

Конечный вид модели приведен на рис. 33.

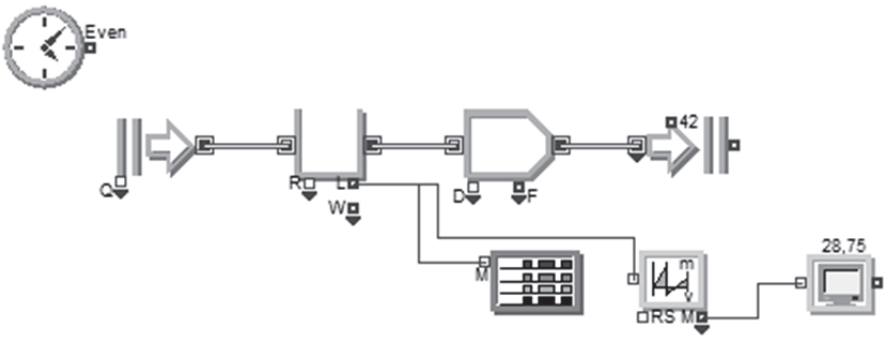


Рис. 33. Модель системы (пример 8)

Для проведения анализа чувствительности в диалоговом окне блока *Create* надо задать режим *Create items randomly*, распределение *Exponential* с параметрами *mean* = 4 и *location* = 0 (рис. 34). Затем щелкнуть правой кнопкой мыши в области значения параметра *mean* и в выпадающем списке выбрать пункт *Sensitize Parameter*. В открывшемся диалговом окне *Sensitivity Setup* установить количество запусков моделирования *Set Simulation Setup to*: 4 *runs*, начальное значение параметра *Starting at*, равным 4, и шаг изменения значения параметра *change by*, равным 1. Помимо явного указания можно также задать условия изме-

нения параметра, используя какой-либо закон распределения, либо считать данные из файла. После закрытия диалогового окна *Sensitivity Setup* выбранный параметр выделяется зеленой рамкой.

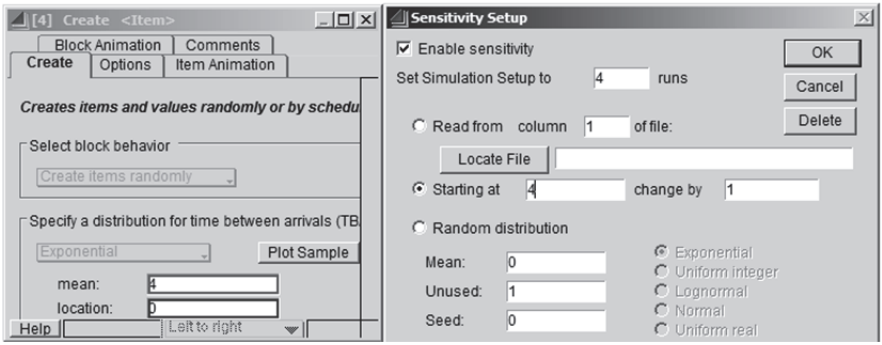


Рис. 34. Настройка анализа чувствительности

Блок *Plotter*, *DE MultiSim* (*Plotter.lix*) подсоединяется к информационному коннектору очереди *L* и отображает до четырех итераций на одной плоскости, тем самым показывая, какое влияние оказывает изменение интенсивности входного потока заданий на длину очереди перед компьютером.

Блок *Mean&Variance* (*Value.lix*) также соединяется с информационным коннектором *L* для сбора статистики о выходных характеристиках очереди. В диалоговом окне блока выбирается режим *Calculate for multiply simulations*, задающий вычисление статистических характеристик по результатам четырех прогонов модели.

Перед запуском моделирования необходимо проверить, чтобы пункт меню *Run>Use Sensitivity Analysis* был выделен галочкой.

После запуска модели на выполнение *ExtendSim* выполнит моделирование четыре раза с разными значениями интенсивности входного потока. На графике *Plotter* отображается изменение длины очереди от времени моделирования для четырех прогонов. Для масштабирования графика используется инструмент *AutoScale Y* в верхней области *Plotter*. Различия между графиками очевидны, как и ожидалось, увеличение среднего интервала между поступлением заданий или уменьшение интенсивности выходного потока приводит к уменьшению длины очереди на обработку (рис. 35). В более сложных моделях эффект внесения изменений не был бы так очевиден.

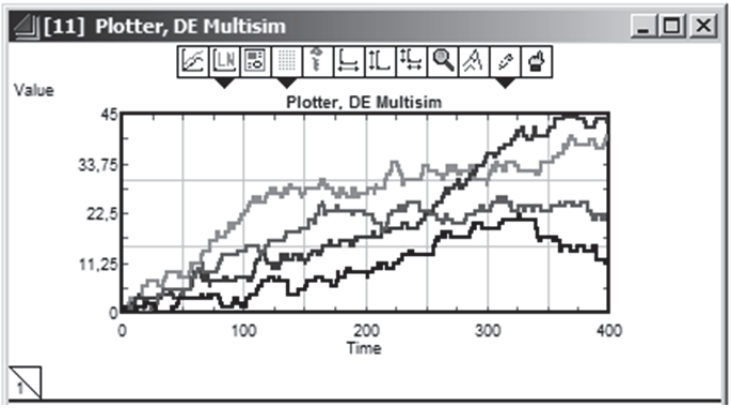


Рис. 35. График изменения длины очереди (пример 8)

После моделирования в диалоговом окне блока *Mean&Variance* (вкладка *History* выводится статистика (средняя длина очереди) по каждому прогону длительностью 400 мс (рис. 36). В блоке *Display Value* отображается средняя длина очереди по результатам четырех прогонов (см. рис. 33).

	Time	Value
0	400	40
1	400	43
2	400	21
3	400	11

Рис. 36. Выходная статистика

Согласно формулам (7) и (8) соответственно приращение входного параметра (среднего интервала между заданиями) равно $\delta X = 81{,}8$ %, а приращение выходной характеристики (средней длины очереди) – $\delta Y = 1896$ %. Таким образом, интенсивность входного потока заявок статистически значимо влияет на среднюю длину очереди. Приращение выходной характеристики в 23,2 раза больше приращения входной. Следовательно, относительно небольшие изменения в интенсивности входного потока вызывают значительное увеличение средней длины очереди.

Анализ чувствительности может быть проведен одновременно для нескольких переменных. Однако рекомендуется проводить анализ не более двух переменных одновременно.

2.7. Контрольные вопросы и задания

1. Какие способы генерации потоков поступления и обслуживания заявок реализованы в *ExtendSim*? Какие блоки среды используются? Приведите примеры задач.

2. Как смоделировать элементарное событие, полную группу событий, зависимые и независимые события в *ExtendSim*? Какие основные блоки среды используются? Приведите примеры задач и способы их решения.

3. Какие блоки среды *ExtendSim* используются для сбора и статистической обработки результатов моделирования?

4. Для чего проводится анализ чувствительности модели? Какие средства реализованы в *ExtendSim* для проведения анализа чувствительности? Приведите примеры задач.

5. Оценить надежность устройства (рис. 37–39), состоящего из трех узлов и элементов A, B, C, D, E, F. Узел выходит из строя, когда выходят из строя все элементы, входящие в узел. Устройство выходит из строя, когда отказывает хотя бы один из его узлов. Вероятности безотказной работы элементов равны соответственно: $P(A) = 0{,}8$; $P(B) = 0{,}7$; $P(C) = 0{,}95$; $P(D) = 0{,}85$; $P(E) = 0{,}9$; $P(F) = 0{,}7$. Рассчитать вероятность безотказной работы устройства аналитическим и имитационным методами. Оценить погрешность имитационного метода.

6. Вероятности того, что во время работы технического устройства произойдет сбой в первом, втором и в третьем блоках, относятся как 3 : 2 : 5. Вероятности обнаружения сбоя в первом, втором и в третьем блоках соответственно равны 0,8; 0,9; 0,95. Рассчитать вероятность обнаружения сбоя аналитическим и имитационным методами. Оценить погрешность имитационного метода.

7. Техническая система состоит из пяти блоков, надежность каждого определяется вероятностью: $p_1 = 0{,}8$; $p_2 = 0{,}75$; $p_3 = 0{,}9$; $p_4 = 0{,}85$; $p_5 = 0{,}9$. Выход из строя хотя бы одного блока влечет за собой выход из строя всей системы. С целью повышения надежности системы производится дублирование, для чего выделено еще пять таких же блоков. Надежность устройств переключения (на рис. 40, 41 они обозначены УП) полная. Определить, какой способ дублирования дает большую надежность системы: дублирование каждого блока (рис. 40); дублирование всей системы (рис. 41). Решить задачу аналитическим и имитационным методами. Оценить погрешность имитационного метода.

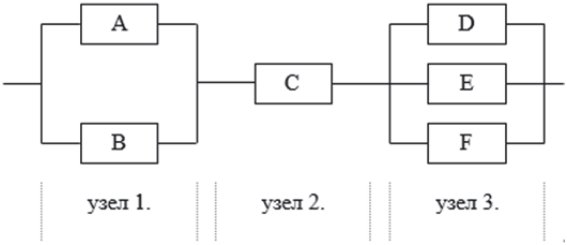


Рис. 37. Схема устройства 1

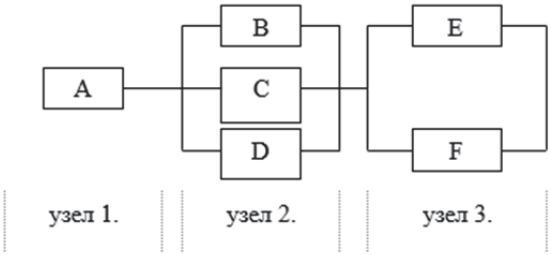


Рис. 38. Схема устройства 2

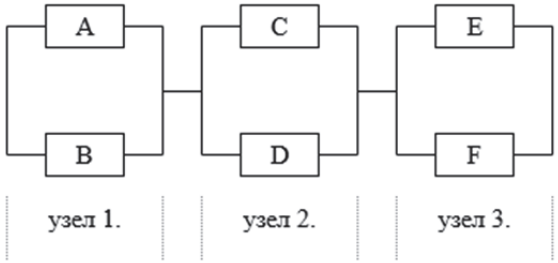


Рис. 39. Схема устройства 3

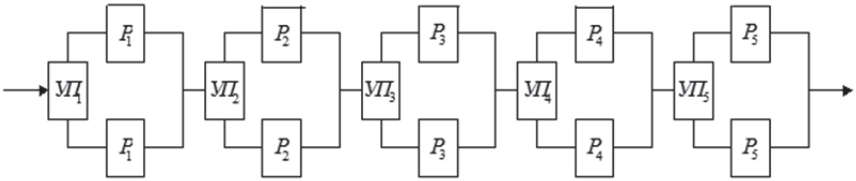


Рис. 40. Схема технической системы (дублирование каждого блока)

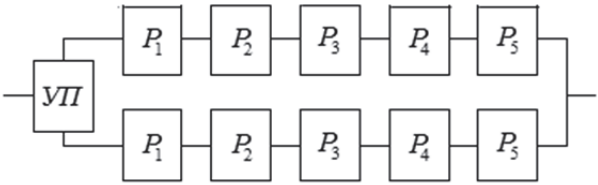


Рис. 41. Схема технической системы (дублирование всей системы)

8. Сравнить две системы, изображенные на рис. 40 и 42, если $p_1 = 0{,}7$; $p_2 = 0{,}9$; $p_3 = 0{,}6$; $p_4 = 0{,}95$; $p_5 = 0{,}4$. В качестве критерия сравнения взять надежность (вероятность безотказной работы). Надежность устройств переключения полная. Решить задачу аналитическим и имитационным методами. Оценить погрешность имитационного метода.

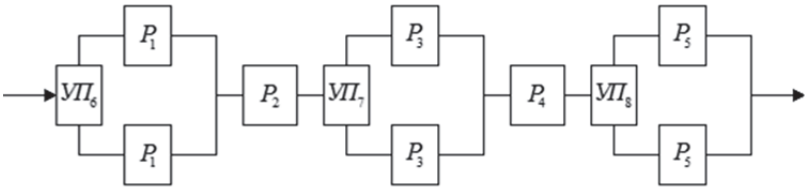


Рис. 42. Схема технической системы (частичное дублирование блоков)

9. Сравнить две системы, изображенные на рис. 40 и 42, учитывая, что устройства переключения (УП) характеризуются надежностью q_k, $k = \overline{1,8}$. Значения вероятностей, определяющих надежность, равны: $p_1 = 0{,}7$; $p_2 = 0{,}8$; $p_3 = 0{,}6$; $p_4 = 0{,}9$; $p_5 = 0{,}7$; $q_1 = 0{,}9$; $q_2 = 0{,}85$; $q_3 = 0{,}8$; $q_4 = q_5 = 0{,}85$; $q_6 = q_7 = q_8 = 0{,}9$. Решить задачу аналитическим и имитационным методами. Оценить погрешность имитационного метода.

10. Два устройства обрабатывают одинаковые задания**.** Производительность первого устройства втрое больше производительности второго. Первое устройство выполняет 95 % заданий без сбоев; второе устройство выполняет 98 % заданий без сбоев. Во время выполнения одного из наудачу выбранных заданий сбоя не произошло. Найти вероятность того, что это задание выполнено вторым устройством. Решить задачу аналитическим и имитационным методами. Оценить погрешность имитационного метода.

11. Два из трех независимо работающих элементов вычислительного устройства отказали. Найти вероятность того, что отказали первый и третий элементы, если вероятности отказа первого, второго и третьего элементов соответственно равны 0,1; 0,2; 0,3. Решить задачу аналитическим и имитационным методами. Оценить погрешность имитационного метода.

12. Техническое устройство, состоящее из трех узлов, работало в течение времени *T*. За это время первый узел оказывается неисправным с вероятностью $p_1 = 0{,}15$; второй – с вероятностью $p_2 = 0{,}1$ и третий – с вероятностью $p_3 = 0{,}2$. Оценить вероятность того, что оказался неисправен только один узел; что оказался неисправным только второй узел. Решить задачу аналитическим и имитационным методами. Оценить погрешность имитационного метода.

13. В систему поступает поток заявок, интервалы между которыми описываются СВ *Z*, распределенной по закону, заданному функцией плотности распределения вероятности. Далее заявки обслуживаются одним каналом в течение заданного интервала времени (см. табл. 1). Смоделировать обработку заявок в течение 100 единиц времени и оценить эффективность системы. Для моделирования последовательности значений СВ *Z* использовать метод обратной функции.

Таблица 1

Исходные данные

Вариант	Функция плотности распределения СВ *Z*	Интервал времени
1	$f(z) = \lambda(1 - \lambda z/2),\ \lambda = 0{,}5,\ 0 \le z \le 2/\lambda$	2
2	$f(z) = c/(1+bz)^2,\ c = 6, b = 5,\ 0 \le z \le 1/(c-b)$	0,5
3	$f(z) = c(1+z)^3,\ c = 2,\ 0 \le z \le 0.316$	0,1
4.	$f(z) = \cos(z),\ 0 \le z \le \pi/2$	1,2
5	$f(z) = z - 1/2,\ 1 \le z \le 2$	2
6	$f(z) = \sin(z),\ 0 \le z \le \pi/2$	1,5
7	$f(z) = c/(1+bz)^2,\ \ c = 4, b = 2,\ 0 \le z \le 1/(c-b)$	0,4
8	$f(z) = \lambda(1 - \lambda z/2),\ \lambda = 1,\ 0 \le z \le 2/\lambda$	1,8

Окончание табл. 1

Вариант	Функция плотности распределения СВ Z	Интервал времени
9	$f(z) = c(1-z)^3,\ c = 4,\ 0 \le z \le 1$	0,9
10	$f(z) = 2\cos(z),\ 0 \le z \le \pi/6$	0,4
11	$f(z) = 2z - 1,\ 0 \le z \le 1{,}618$	1,5
12	$f(z) = 2\sin(z),\ 0 \le z \le \pi/3$	1,1

14. В систему поступает поток заявок через заданный интервал времени (табл. 1). Далее заявки обслуживаются одним каналом в течение случайного интервала времени Z, описываемого заданной функцией плотности распределения вероятностей (табл. 1). Смоделировать обработку заявок в течение 100 единиц времени и оценить эффективность системы. Для моделирования последовательности значений СВ Z использовать метод Неймана.

15. Исследовать работу локального сетевого концентратора при низкой и высокой загруженности. На вход концентратора поступает поток транзактов с интенсивностью 0,2 и 0,5. Далее транзакты обрабатываются с интенсивностью 0,45. Длина очереди транзактов на обработку не ограничена. Потоки поступления и обслуживания транзактов описываются: 1) экспоненциальным законом; 2) законом Эрланга второго порядка. Построить графики зависимости длины очереди транзактов на обслуживание от времени при разной загруженности и разных типах потоков, протекающих в системе, оценить характеристики очереди и загрузку концентратора. Построить гистограмму времени пребывания транзакта в системе. Сделать выводы о влиянии входных параметров системы на длину очереди и загруженность концентратора.

16. Исследовать работу локального коммутатора, который может параллельно обрабатывать несколько сессий, при низкой и высокой загруженности. На вход коммутатора поступает поток транзактов с интенсивностью 0,4 и 2,5. Далее транзакты обрабатываются с интенсивностью 0,5. Длина очереди транзактов на обработку не ограничена. Потоки поступления и обслуживания транзактов описываются: 1) экспоненциальным законом; 2) законом Эрланга второго порядка. Параллельно может обрабатываться 1) два транзакта; 2) три транзакта. Построить графики зависимости длины очереди транзактов на обслуживание от времени для разных вариантов системы, оценить характери-

стики очереди на обслуживание и характеристики коммутатора. Построить гистограмму длины очереди заявок. Сделать выводы об эффективности работы коммутатора для разных вариантов реализации, выбрать оптимальный вариант.

17. В систему поступает поток заявок по экспоненциальному закону распределения со средним интервалом 12 с. Интенсивность обработки заявок зависит от времени суток (закон распределения экспоненциальный): с 0 до 8 ч заявки обрабатываются в среднем 30 с; с 8 до 12 ч – в среднем 42 с; с 12 до 17 ч – в среднем 48 с; с 17 до 24 ч – в среднем 27 с. Заявки обрабатываются одним из трех одинаковых устройств. Смоделировать работу системы по обслуживанию 1000 заявок. Оценить характеристики очереди заявок, загрузку устройств. Построить графики зависимости пребывания заявки в системе от времени и очереди заявок от времени. Построить гистограмму длины очереди заявок. Оценить эффективность системы, предложить варианты оптимизации системы.

18. В систему поступает два потока заданий на обработку с интенсивностями 4 и 6 мс, которые обрабатываются одним из двух компьютеров с интенсивностью 5 мс. Закон распределения времени поступления и обслуживания заданий – экспоненциальный. Выбор компьютера, обрабатывающего задание, равновероятен. Оценить, как влияет изменение интенсивности входных потоков заданий (1-й поток – от 2 до 5 мс; 2-й поток – от 5 до 8 мс) на среднее время пребывания задания в системе. Время моделирования составляет 1000 мс. Привести графики, сделать выводы о чувствительности модели к изменению интенсивности входных потоков.

19. В задаче п. 18 провести анализ чувствительности модели к изменению интенсивности обслуживания задания в диапазоне от 3 до 6 мс. Как изменение интенсивности обслуживания влияет на среднее время пребывания задания в очереди на обслуживание?

20. На обработку поступают задания трех типов в диапазоне соответственно: 2 ± 1 мс; 4 ± 2 мс; 4 ± 1 мс (закон распределения равномерный). Обработка состоит из двух этапов: расчет и тестирование. Время расчета и тестирования зависит от типа задания и составляет соответственно: расчет: 5 ± 3 мс; 5 ± 1 мс; 7 ± 1 мс; тестирование: 4 ± 1 мс; $6{,}5 \pm 0{,}5$ мс; $9{,}5 \pm 0{,}5$ мс (закон распределения равномерный). Расчет выполняют шесть процессоров; тестирование – семь процессоров. Оценить статистические характеристики очереди заданий на двух эта-

пах обработки, статистические характеристики загрузки процессоров системы. Построить гистограмму времени пребывания задания в очереди на обслуживание.

2.8. Инструменты для управления потоком заявок

2.8.1. Маршрутизация заявок

В *ExtendSim* реализован инструментарий для управления потоком заявок, позволяющий разделять или объединять потоки в зависимости от заданных условий (критериев). Маршрутизация потоков реализуется блоками  *Select Item In* и  *Select Item Out*.

Блок *Select Item In* выполняет слияние нескольких потоков заявок в один поток, выбирает заявку с одного из входных коннекторов в зависимости от условий и посылает заявку на выходной коннектор. Доступны разные режимы работы блока:

– по приоритету (*Item priority*). Выбирается входной коннектор (вход), на котором доступна заявка с наивысшим приоритетом (соответствует наименьшему числовому значению приоритета);

– по вероятности (*Random*). Вход выбирается по вероятности, заданной в диалоговом окне блока. Вероятности вводятся в десятичном формате (например, можно ввести 0,25 или 25 %), сумма вероятностей должна быть равна единице. Если выбрана опция *Select from*: *all input*, то вход будет выбран случайным образом независимо от того, доступна или нет заявка на данном входе. Подобная ситуация может потенциально привести к зависанию. Если выбран пункт *Select from*: *only inputs with available items*, блок будет выбирать только входы с доступными заявками;

– по значению на коннекторе блока (*Select connector*). Значение, полученное на коннекторе *select*, определяет, какой выбирается вход. В диалоговом окне блока есть поле ввода для значения, соответствующего верхнему входу; по умолчанию установлен 0. Нижние коннекторы будут пронумерованы последовательно от верхнего коннектора, т. е. если верхний вход определяется значением 1, то второй вход будет пронумерован как 2, следующий нижний вход как 3 и так далее. В этом случае значение 3 на коннекторе *select* приведет к выбору

заявки с третьего входа. Даже если доступны заявки на других входах, блок ожидает заявки на входе 3;

– последовательно (*Sequential*). Выбираются входы в строго последовательном порядке, начиная сверху. Это также известно, как выбор по методу «*round robin*» – циклический перебор;

– объединение (*Merge*). Заявки выбираются по мере доступности. В общем случае данный режим используется для объединения потоков заявок, где отсутствует блокировка заявок, поступающих на блок *Select Item In*. Входы выбираются в порядке циклического перебора, начиная сверху; после получения заявки с входа выбор будет продолжен со следующего входа.

Блок *Select Item Out* разделяет поток заявок и направляет их по какому-либо конкретному пути в зависимости от условий. Доступны следующие режимы работы блока:

– по свойствам (*Property*). Выход определяется с использованием параметров заявки – атрибута или приоритета;

– по приоритету коннекторов (*Connector priority*). Делается попытка отправить заявку на каждый коннектор в порядке их приоритета, пока заявка не будет принята присоединенным блоком. Приоритет для коннекторов вводится в таблице в диалоговом окне блока. Наивысший приоритет соответствует наименьшему числу;

– по вероятности (*Random*). Выходы выбираются по вероятности исходя из настроек, заданных в таблице вероятности блока. При выборе режима *If output is blocked*: *item will try unblocked outputs* блок будет случайным образом выбирать выход для отправления заявки. При выборе режима *If output is blocked*: *item will wait for blocked output* блок будет выбирать выход, и заявка будет в состоянии ожидания, пока выход не сможет принять заявку;

– по значению на коннекторе блока (*Select connector*). Значение, полученное на коннекторе *select*, определяет, какой выбирается выход. В диалоговом окне блока есть поле ввода для значения, соответствующего верхнему выходу; по умолчанию установлен 0. Нижние коннекторы будут пронумерованы последовательно после верхнего коннектора. Блок удерживает заявку, пока выходной канал занят;

– последовательно (*Sequential*). Выходы выбираются в последовательном порядке, начиная с верхнего (режим циклического выбора «*round robin*»). При выборе пункта *If output is blocked*: *item will try unblocked outputs* блок будет проверять последовательно все коннекто-

ры, пока не найдет свободный выход. Если выбран пункт *If output is blocked*: *item will wait for blocked output*, блок выберет выход и заявка будет ждать до тех пор, пока выход не сможет принять заявку.

Блоки *Select Item In* и *Select Item Out* работают только с целыми числами. Поэтому, если выбран режим *select connector*, то числа 0,001 и 0,999, полученные на входе *select*, будут приведены к 0.

Пример 9. Специализированная вычислительная система получает на вход задания. Задания поступают в систему в соответствии с нормальным законом распределения со средним значением 20 мс и средним квадратическим отклонением (СКО) – 2 мс. Задания распределяются на два компьютера с вероятностями 0,8 и 0,2. Время обработки задания на каждом из компьютеров распределено по равномерному закону в диапазоне 10 ± 2 мс; 8 ± 2 мс соответственно. Промоделировать работу системы по обработке 2000 заявок.

Конечный вид модели показан на рис. 43.

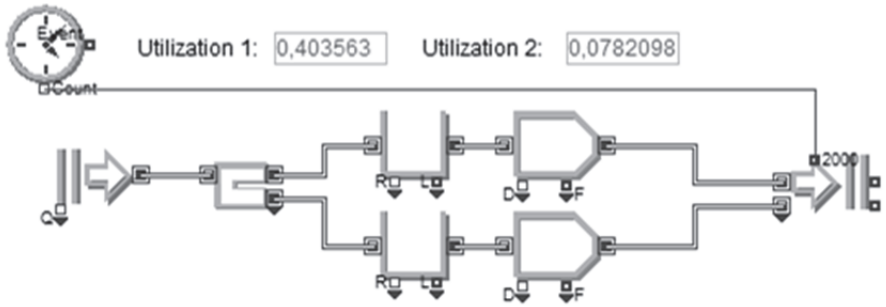


Рис. 43. Модель системы (пример 9)

Для построения модели необходимо выполнить следующие шаги.

1. Создать, разместить и соединить блоки, как показано на схеме (см. рис. 43).

2. Задать в блоке *Executive* режим окончания моделирования: *Stop simulation*: *when count connector value* = 2000.

3. Задать в диалоговом окне блока *Create* режим *Create items randomly*, распределение *Normal* с параметрами *mean* = 20 и *Std.dev* = 2.

4. Задать в диалоговом окне блока *Select Item Out* режим *Select output based on*: *random* опцию *if output is bloked*: *item will wait for blocked output* и таблицу вероятностей *Probability*: 0,8 – первый выход, 0,2 – второй выход.

5. Задать в диалоговом окне блоков *Activity* режим *Delay is*: *specified by a distribution*, распределение *Uniform*, *Real* с параметрами *Min* = 8 (6) и *Max* = 12 (10) и вынести в область модельного окна выходную характеристику *Utilization* (коэффициент использования устройства).

Для остальных блоков модели используются настройки «по умолчанию».

В результате моделирования выдается информация о загрузке устройств: 0,4 (для первого компьютера) и 0,08 (для второго), которая позволяет сделать вывод о неравномерной загрузке компьютеров системы и неполном использовании вычислительных ресурсов системы в целом. Система имеет резервы вычислительной мощности, и входной поток заявок может быть значительно увеличен.

Пример 10. На обработку в вычислительную систему поступают задания. Средний интервал между поступлением двух заданий подчиняется равномерному закону распределения в диапазоне 12 ± 2 мс. Задания обрабатываются одним из трех компьютеров. Время обработки – 36 мс. Если в очереди на обработку стоит три задания, то вновь поступившее задание выводится из системы без обработки. Промоделировать работу системы в течение 200 мс.

Конечный вид модели с ограничением очереди представлен на рис. 44.

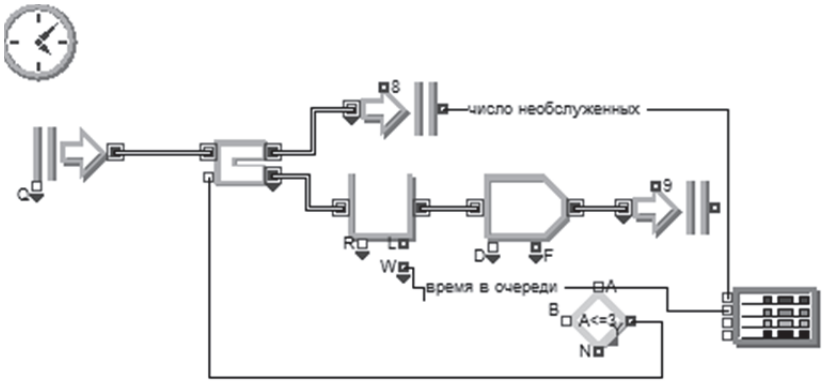


Рис. 44. Модель системы (пример 10)

Последовательность разработки модели следующая.

1. Создать, разместить и соединить блоки, как показано на схеме (рис. 44).

2. Задать в блоке *Executive* режим окончания моделирования: *Stop simulation*: *at end time*. В пункте меню *Run>Simulation Setup* задать *End time* = 200.

3. Задать в диалоговом окне блока *Create* режим *Create items randomly*, распределение *Uniform*, *Real* с параметрами *Min* = 10 и *Max* = 14.

4. Задать в диалоговом окне блока *Select Item Out* режим *Select output based on*: *select connect*.

5. Задать в диалоговом окне блоков *Activity* режим *Delay is*: *a constant*, *Delay* (*D*) = *36 time units*, *Maximum items in activity* = 3.

6. Задать в диалоговом окне блока *Decision* (библиотека *Value.lix*) опцию *Outputs 1 at Y* (*TRUE*) *and 0 at N* (*FALSE*) *if*: *A<= B and B is*: 3.

Для остальных блоков модели используются настройки «по умолчанию».

В блоке *Decision* анализируется длина очереди. Если длина очереди меньше или равна пороговой величине, определенной в диалоговом окне блока *Decision* (3 заявки), на выходе *Y* будет выдано значение 1. Это укажет блоку *Select Item Out*, что нужно направлять заявки через нижний выходной коннектор. Если длина очереди больше порогового значения, выход *N* блока *Decision* выдаст 0, и заявки будут направлены на выход, игнорируя очередь.

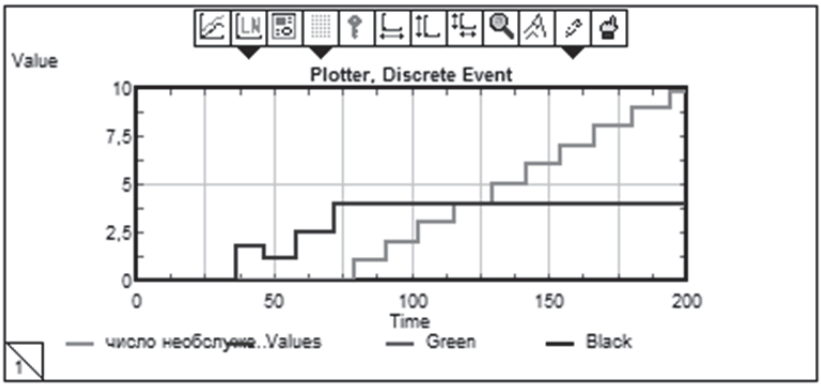


Рис. 45. График изменения выходных характеристик системы (пример 10)

В результате моделирования выдается график изменения времени ожидания заявки в очереди и количества необслуженных заявок во времени (рис. 45). Анализ выходных характеристик позволяет сделать

вывод о низкой эффективности моделируемой системы: обрабатывается примерно только половина поступающих заявок (9 из 17 заявок по результатам одного прогона модели), остальные заявки выводятся из системы без обработки.

2.8.2. Условная маршрутизация

В режиме условной маршрутизации путь прохождения заявки определяется в зависимости от заданных условий. Например, заявка может быть направлена по определенному пути в случае достижения очередью порогового значения или превышения заданного времени ожидания. Для реализации режима условной маршрутизации в *ExtendSim* используются блок *Decision* (задает условие, включен в библиотеку *Value.lix*) и блок *Gate* (ограничивает прохождение заявок, открывая или закрывая путь в зависимости от заданных условий, включен в библиотеку *Item.lix*).

Пример 11. На обрабатывающий участок цеха предприятия через интервалы времени, распределенные экспоненциально со средним значением 10 мин, поступают партии, каждая из которых состоит из трех деталей. Обработка деталей производится основным станком в течение 3 мин. Если в очереди к станку скапливается более четырех деталей, подключается резервный станок, который обрабатывает деталь за 5 мин. Смоделировать обработку на участке 500 деталей. Определить загрузку основного и резервного станков.

Конечный вид модели изображен на рис. 46.

Последовательность разработки модели следующая.

1. Создать, разместить и соединить блоки, как показано на схеме (рис. 46).

2. Задать в блоке *Executive* режим окончания моделирования: *Stop simulation*: *when count connector value* = 500.

3. Задать в диалоговом окне блока *Create* режим *Create items randomly*, распределение *Exponential* с параметрами *mean* = 10 и *location* = 0 и во вкладке *Options* задать число деталей в партии: *item quntities* = 3.

4. Задать в диалоговом окне блока *Decision* опцию *Outputs 1 at Y* (*TRUE*) *and 0 at N* (*FALSE*) *if*: *A>=B and B is*: *4*.

5. Задать в диалоговом окне блока *Gate* режим *Type*: *conditional gating with values* и *Recheck demand connector after each item passes*.

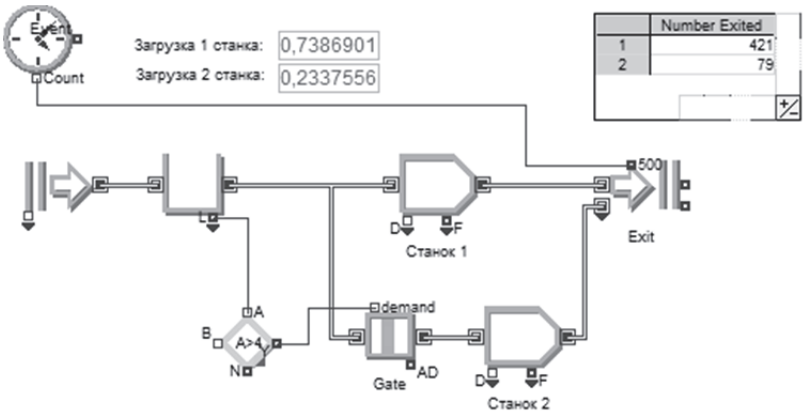


Рис. 46. Модель обрабатывающего участка (пример 11)

6. Задать в диалоговом окне блоков *Activity* режим *Delay is*: *a constant*, *Delay* (*D*) = *3* (*5*) *time units*, *Maximum items in activity* = 1.

В модели выход *L* блока очереди, подсоединенной к станкам, выдает количество заявок, ожидающих обработки. Если это значение выше некоторого порога, к обработке заявок подключается резервный станок. В настройках блока *Decision* задано, что коннектор *Y* выдает значение истины – 1, когда значение на входе *A* больше четырех. Этот сигнал активирует коннектор *demand* блока *Gate* и разрешает поступление заявок на резервный станок, который до этого был заблокирован. Когда в очереди содержится четыре или менее заявок, коннектор блока *Gate* закрывается.

Также можно промоделировать обратную ситуацию, когда все блоки обрабатывают заявки, и затем один или более закрываются при определенном условии. При осуществлении этого условия заявки могут остаться в закрытых блоках до тех пор, пока не продолжится обработка.

Результаты работы модели после обработки 500 деталей по результатам одного прогона представлены в модельном окне (рис. 46): коэффициент использования основного станка – 0,74; резервного – 0,23. Основной станок обработал 421 деталь, резервный – 79.

Пример 12. На регулировочный участок цеха поступают устройства первого типа в среднем через 12 мин и устройства второго типа в среднем через 14 мин. Каждый тип устройства проходит первичную

регулировку, причем в зависимости от типа регулировку осуществляет специалист соответствующей квалификационной категории. Всего задействовано три специалиста первой квалификационной категории, которые выполняют настройку устройства в среднем за 34 мин и пять специалистов второй квалификационной категории, которые выполняют регулировку в среднем за 80 мин. Далее все устройства направляются на вторичную регулировку, которая вне зависимости от типа выполняется в среднем за 35 мин, вторичную регулировку выполняют пять специалистов. Все величины, заданные средними значениями, распределены экспоненциально.

Смоделировать работу участка в течение 1000 часов. Определить характеристики очереди на втором этапе регулировки.

Конечный вид модели показан на рис. 47.

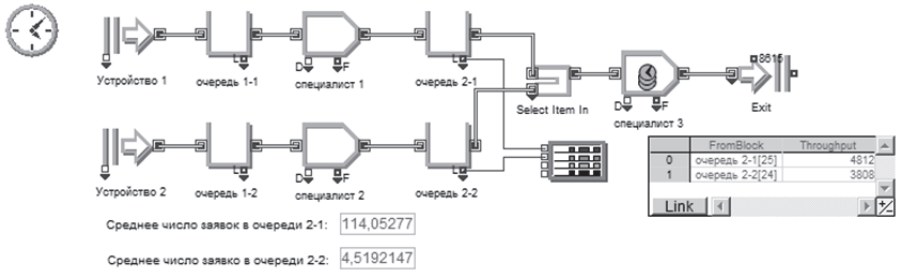


Рис. 47. Модель регулировочного участка (пример 12)

Последовательность разработки модели следующая.

1. Создать, разместить и соединить блоки, как показано на схеме (рис. 47).

2. Задать в блоке *Executive* режим окончания моделирования: *Stop simulation*: *at end time*. В пункте меню *Run>Simulation Setup* задать *End time* = 60000 мин.

3. Задать в диалоговом окне блоков *Create* режим *Create items randomly*, распределение *Exponential* с заданным средним *mean* = 12 (14) и *location* = 0.

4. Задать в диалоговом окне блока *Select Item In* режим *select input based on*: *merge*.

5. Задать в диалоговом окне блоков *Activity* режим *Delay is*: *specified by a distribution*, распределение *Exponential* с параметрами *mean* = 34 (80, 35) и *location* = 0.

Для остальных блоков модели используются настройки «по умолчанию».

В модели регулировочного участка блок *Select Item In* будет принимать заявки от любого из двух источников. Диалог установлен в режим *Select input based on*: *merge*. Если выход блока *Select Item In* заблокирован, блок приведет к ожиданию заявок в очередях (помеченных как очередь 2-1, 2-2). Когда *Select Item In* выходит из блокировки, он проверяет каждый вход по очереди, чтобы получить заявку для дальнейшей обработки на блоке *Activity* (специалист 3), причем блок получает любую доступную заявку. Это может привести к состоянию, когда некоторые очереди становятся длиннее остальных. Так, в результате моделирования работы участка средняя длина очереди на вторичную регулировку для устройств первого типа составила 114, а для устройств второго типа – 4,5 (рис. 47). На рис. 48 изображен график изменения средней длины очереди в течение времени моделирования для каждого типа устройств. Очевидно, что длина первой очереди в несколько раз больше длины второй очереди.

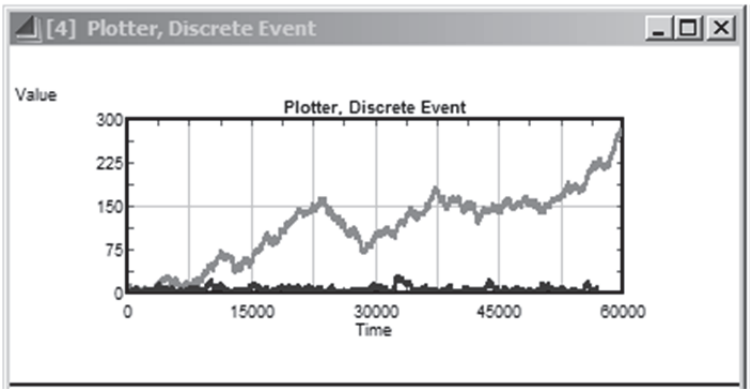


Рис. 48. График изменения длины очереди (пример 12)

Вариантом модификации модели может быть переключение блока *Select Item In* в режим *Select inputs based on*: *random* и выставление вероятности выбора каждого входа в соответствии со временем между поступлениями заявок. Таким образом, можно уменьшить несбалансированность очередей на выходе устройств.

Однако хотя этот способ способствует балансировке выходных очередей, он не является достаточно эффективным в связи с тем, что

вероятности выбора входных блоков должны быть связаны со временами поступления заявок. К тому же работа в данном режиме может привести к зависанию системы, так как блок *Select Item In* после выбора входа будет ожидать поступления заявок на этот вход, в то время как на других входах заявки будут ожидать обработки.

Другой способ балансировки очередей заключается в использовании блока *Select Item In* в связке с блоком *Max&Min* (*Value.lix*), который проверяет длину каждой очереди.

Пример 13. Модернизируем модель регулировочного участка цеха и реализуем режим балансировки очередей.

Конечный вид модели изображен на рис. 49.

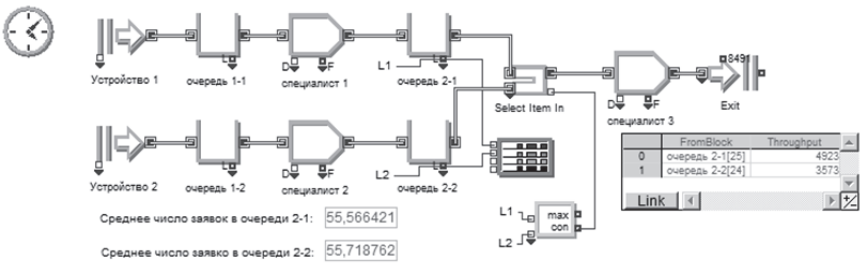


Рис. 49. Модель регулировочного участка (пример 13)

Последовательность построения модели следующая.

1. Создать, разместить и соединить блоки, как показано на схеме (рис. 49).

2. Задать настройки блоков *Create* и *Activity*, такие же как в модели примера 12.

3. Задать в диалоговом окне блока *Select Item In* режим *Select input based on*: *select connector*.

4. Задать в диалоговом окне блока *Max&Min* параметры *Output the*: *maximum value* и *top input connector starts at*:*0*.

5. Создать именованные соединения *L1*, *L2* от выходных коннекторов, передающих текущую длину очереди к блоку *Max & Min*.

В модели выходной коннектор *con* блока *Max&Min* сообщает, какой из входов имеет наибольшее значение, которое в данном примере означает очередь на выходе обработчика с наибольшим количеством заявок. Блок *Max&Min* сообщает блоку *Select Item In*, из какой очереди получить следующую заявку.

Заявки выбираются равномерно с каждой линии, и длины очередей примерно равны, по сравнению с тем, что было приведено в примере 12. В результате моделирования средняя длина очереди на вторичную регулировку для устройств первого типа составила: 55,6, а для устройств второго типа – 55,7 (рис. 49). На рис. 50 приведен график изменения средней длины очереди в течение времени моделирования для каждого типа устройств, который наглядно иллюстрирует сбалансированность очередей.

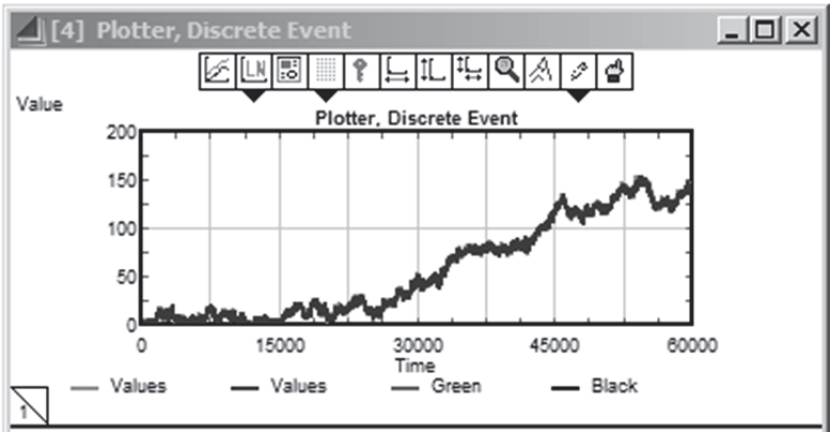


Рис. 50. График изменения длины очереди (пример 13)

2.8.3. Моделирование процессов прерывания обслуживания

В задачах дискретно-событийного моделирования часто возникает ситуация прерывания процесса обработки заявки (транзакта). Прерывание процесса может быть вызвано разными причинами. Например, выход из строя (поломка) канала обслуживания; приход более приоритетной заявки, которая вытесняет заявку с более низким приоритетом; плановое отключение оборудования и т. п.

В *ExtendSim* настройки процесса прерывания задаются в блоке *Activity*. В диалоговом окне блока есть две вкладки *Preemption* (приоритетность) и *Shutdown* (отключение) для описания процесса прерывания.

Во вкладке *Preemption* задаются настройки для моделирования ситуации прихода более приоритетной заявки. В случае ее прихода

прерванная заявка выводится из блока *Activity* через дополнительный выходной коннектор (*PE_ItemOut*). В зависимости от настроек может быть прервано выполнение заявки с более низким приоритетом (*the item with the lowest priority*); заявки, на завершение которой требуется наибольшее/наименьшее время (*the item furthest/closest to finishing*); все текущие заявки (*all items currently in processing*); только заявки с определенным значением атрибута (*only items with a particular attribute value*). Также можно задать режим прерывания только в случае занятости всех каналов в блоке *Activity* (*Preempt only if block is full*) и задать атрибут, в котором будет храниться оставшееся время обработки заявки (*Store remaining process time in attribute*).

Во вкладе *Shutdown* задаются настройки для моделирования ситуации временной блокировки (отказа) блока *Activity*. Блокировка происходит, когда на входной коннектор *SD* блока *Activity* приходит сигнал (*value*) или элемент (*item*).

Для задания расписания работы блока *Activity* можно использовать блок 

Shutdown, в диалоговом окне которого задаются законы распределения интервалов между блокировками (*TBF – time between failures*) и восстановлением (*TTR – time to repair*) блока *Activity*. На выходном коннекторе блока *Shutdown* (*SD_ValueOut*) генерируется значение 1 (блокировка) или 0 (работа). Этот коннектор связывается с коннектором *SD* блока *Activity*, и таким образом происходит управление процессом обслуживания заявок.

В диалоговом окне блока *Activity* во вкладке *Shutdown* также можно задать, что происходит с заявкой в случае блокировки блока: заявка сохраняется и ее обслуживание возобновляется после восстановления (*keep items*, *resume process after shutdown*); заявка сохраняется и ее обслуживание начинается сначала после восстановления (*keep items*, *restart process after shutdown*), завершается обработка текущей заявки, а потом только блок блокируется (*finish processing items before shutting down*); заявка уничтожается (*discard items that are being processed*).

Приведем пример, в котором иллюстрируются разные варианты прерывания процесса моделирования.

Пример 14. В вычислительную систему на обработку поступают два потока пакетов с низким и высоким уровнями приоритета. Интенсивность поступления пакетов первого потока – 0,2; второго потока – 0,25. Пакеты проходят два этапа обработки. На первом этапе обработ-

ки задействовано одно устройство, средний интервал обслуживания равен 2 с. В случае поступления пакета с высоким приоритетом во время обслуживания пакета с низким приоритетом происходит прерывание обработки, пакет с низким приоритетом выводится из системы и снова становится в очередь на обработку. На втором этапе происходит обработка пакета равновероятно одним из двух устройств, средний интервал обслуживания равен 3 с. На втором этапе возможна блокировка устройства. Средний интервал между блокировками – 100 с, средний интервал восстановления – 10 с. Все потоки, протекающие в системе, – пуассоновские.

Смоделировать систему в течение 1000 с. Определить коэффициенты загрузки всех устройств, среднюю длину очереди пакетов на обслуживание на всех этапах, общее количество прерываний на первом этапе обработки, количество и общее время блокировки устройств на втором этапе обработки.

Конечный вид модели представлен на рис. 51.

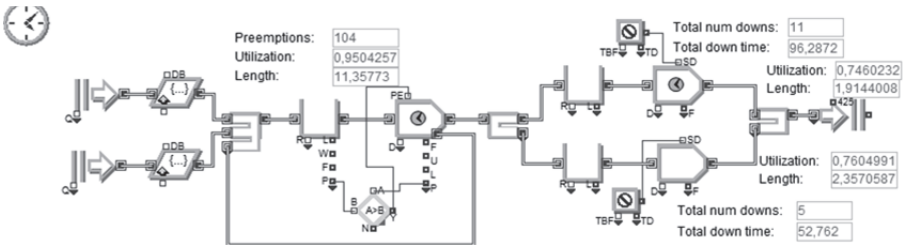


Рис. 51. Модель вычислительной системы (пример 14)

Настройки основных блоков следующие.

1. Задать в диалоговом окне блоков *Create* режим *Create items randomly*, распределение *Exponential* с заданным средним *mean* = 5 (4) и *location* = 0.

2. Задать в диалоговом окне блоков *Set* в поле *Property Name* атрибут приоритет: *Item Priority*, равный 1 и 2 соответственно для первого и второго потоков.

3. Задать в диалоговом окне блоков *Activity* во вкладке *Process* режим *Delay is*: *specified by a distribution*, распределение *Exponential* с *mean* = 2 (3) и *location* = 0. В первом блоке *Activity* во вкладке *Preempt* включить режим прерывания: *Enable preemption*, задать режим прерывания пакета с низким приоритетом: *the item with the lowest priority*, прерывание только в случае полностью занятого устройства: *Preempt*

only if block is full, задать имя атрибута, в котором хранится оставшееся время обработки прерванного пакета: *Store remaining process time in attribute – Remainingtime* и его использование: *Use this attribute as delay*. Во втором и третьем блоках *Activity* во вкладке *Shutdown* включить режим блокировки: *Enable shutdown*, опцию сохранения прерванного пакета и возобновления его обслуживания после снятия блокировки: *When activity shuts down Keep items*, *resume process after shutdown*.

4. Задать в диалоговом окне первого блока *Select Item In* режим *select input based on*: *merge*, во втором блоке *Select Item In* режим *select input based on*: *random*.

5. Задать в диалоговом окне первого блока *Queue* метод сортировки очереди по приоритету: *Select sort method – Sort by Priority*.

6. Задать в диалоговом окне блока *Decision* режим сравнения приоритетов пакета, находящегося на обслуживании, и пакета, стоящего в очереди на обслуживание: *A>B*.

7. Задать в диалоговом окне блоков *Shutdown* распределение времени между блокировками (поломками): *Set time between failures* и времени восстановления: *Set time to repair* по экспоненциальному закону *Exponential* с параметрами: *mean* = 100 (10) и *location* = 0.

Для остальных блоков модели используются настройки «по умолчанию».

В результате моделирования коэффициенты использования устройств *Utilization* на первом и втором этапах обработки равны соответственно 0,95 (0,74; 0,76); средняя длина очереди на обработку *Length* по этапам обслуживания: 11,3 (1,9; 2,4). Количество прерываний *Preemptions* обработки пакетов с низким уровнем приоритета на первом этапе обслуживания: 104. Количество блокировок *Total num downs* и суммарное время блокировки устройств *Total down time* на втором этапе обслуживания: 11 (5) и 96,2 (52,7). В целом система справляется с обработкой потока пакетов, наибольший коэффициент загрузки соответствует первому устройству.

2.9. Контрольные вопросы и задания

1. Какие блоки реализованы в среде *ExtendSim* для управления потоком заявок? Перечислите основные режимы работы блоков.

2. Исследовать работу магистрального маршрутизатора. На вход маршрутизатора поступает три потока пакетов с интенсивностями

соответственно 0,3; 0,4; 0,8. Параллельно маршрутизатор может обрабатывать три пакета. Интенсивность обработки – 0,7. Допустимый размер очереди пакетов – 30. При превышении размера очереди происходит потеря пакета. Все потоки, протекающие в системе, пуассоновские. Смоделировать работу маршрутизатора в течение 1000 тактов времени. Построить графики изменения длины очереди и времени пребывания пакета в системе в процессе моделирования. Оценить характеристики очереди пакетов и загрузки маршрутизатора, рассчитать количество обслуженных и потерянных пакетов. Сделать выводы об эффективности системы, предложить варианты оптимизации.

3. В задачу п. 2 добавить условие: входные потоки пакетов имеют разный приоритет с уровнями соответственно 1, 2, 3. В случае прихода пакета с более высоким приоритетом происходит прерывание обслуживания пакета с более низким приоритетом и прерванный пакет снова встает в очередь на обслуживание. Сравните характеристики эффективности работы маршрутизатора в случае использования абсолютного приоритета и без приоритета. Сделайте выводы.

4. В трехканальную вычислительную систему поступает два потока задач с интенсивностями 0,3 и 0,4 соответственно. Заявки обрабатываются с интенсивностью 0,12. Потоки поступления и обслуживания задач – пуассоновские. На обработку задачи выделяется квант времени, равный 9. Если за это время задача не обрабатывается, то она выводится из системы и снова встает в очередь на обработку. Смоделировать работу системы в течение 100 единиц времени. Оценить эффективность системы.

5. На вход вычислительной системы поступают три типа заданий: *A*, *B* и *C*. Задания типа *A* поступают по нормальному закону со средним, равным 8 с, и СКО, равным 1 с; задания типа *B* – по закону Эрланга 2-го порядка со средним, равным 6 с; задания типа *C* – с постоянным интервалом, равным 9 с. Далее задания предварительно записываются, процесс записи занимает 0,5 с. Затем задания поступают на обработку, причем каждый тип задания на выделенный компьютер – это обработка первого уровня (интенсивность обработки 2 с, закон экспоненциальный). После первичной обработки задачи распределяются случайным образом с вероятностями 0,3 и 0,7 на компьютеры, обеспечивающие обработку второго уровня (интенсивность обработки 3 с, закон экспоненциальный). Каждый компьютер одновременно может обрабатывать только одно задание. Смоделировать обработку 2000 заданий.

Оценить характеристики очереди заданий каждого типа на первом этапе обработки, загрузку компьютеров системы. Оценить эффективность системы.

6. Измените условие задачи п. 5: задания на первом уровне обработки поступают к компьютеру, который может одновременно обрабатывать три задания (интенсивность обработки 1 с, закон распределения экспоненциальный). Исследуйте два варианта реализации системы: 1) с балансировкой очередей заданий *A*, *B* и *C* перед первичной обработкой; 2) без балансировки. Сделайте выводы.

7. Вычислительная система состоит из трех процессоров и общей оперативной памяти. Задания поступают на обработку по равномерному закону в интервале 6 ± 1 с. После трансляции на первом процессоре (5 ± 1 с, закон распределения равномерный) 40 % заданий идут на второй процессор на редактирование (2 ± 1 с, закон распределения равномерный), а 60 % – сразу на решение на третий процессор. Отредактированные задания поступают на третий процессор на решение, требующее 2 ± 1 с, закон распределения равномерный. Смоделировать работу вычислительной системы в течение 100 часов. Оценить характеристики загрузки процессора, характеристики очереди заданий на каждом этапе обработки. Определить «узкие» места в системе, предложить и исследовать варианты ее оптимизации.

8. В распределенный банк данных поступают запросы с интенсивностью 4 с (закон распределения экспоненциальный) с уровнями приоритета от 1 до 5 (задания с разным приоритетом приходят равновероятно). Далее запросы проходят первичную обработку, которая занимает $1 \pm 0,2$ с по равномерному закону распределения в соответствии с уровнями приоритета. Затем запросы поступают на вторичную обработку, которая длится $1,5 \pm 0,5$ с по равномерному закону распределения и происходит в порядке очереди. Одновременно на всех этапах обработки может обслуживаться только один запрос. Смоделируйте обработку 1000 запросов. Исследуйте систему, оцените ее эффективность.

9. Исследуется работа локальной вычислительной сети с отказами. На рис. 52 и 53 изображены схемы сети маршрутизаторов. Все маршрутизаторы, входящие в сеть, идентичные и представляют собой одноканальную систему массового обслуживания, на вход которой поступает пуассоновский поток транзактов с интенсивностью 0,1. Поток обслуживания также пуассоновский с интенсивностью 0,9; допустимая длина очереди – 20. Выбор маршрутизатора транзактом в местах раз-

ветвления – равновероятный. Смоделировать работу сети в течение 1000 тактов времени. Оценить загрузку маршрутизаторов, характеристики очереди транзактов, построить графики загруженности маршрутизаторов в процессе моделирования, рассчитать процент потерь транзактов на каждом маршрутизаторе. Оценить эффективность работы локальной сети. При построении модели использовать иерархические блоки.

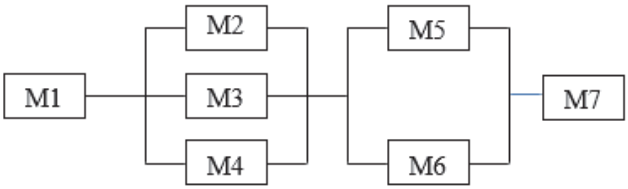


Рис. *52*. Схема сети маршрутизаторов (вариант 1)

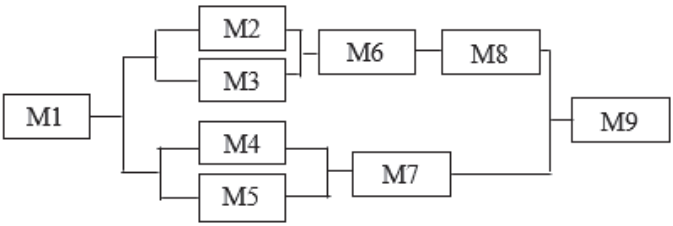


Рис. *53*. Схема сети маршрутизаторов (вариант 2)

10. В задачу п. 9 добавить условие: один из маршрутизаторов (M2) в процессе работы может выйти из строя. Поток поломок – пуассоновский с интенсивностью 0,01, время восстановления распределено по экспоненциальному закону с интенсивностью – 0,1. Если поломка произошла во время передачи транзакта, то транзакт ожидает восстановления маршрутизатора. Сравните два вариант работы сети. Какие характеристики изменились и как?

11. Магистраль передачи данных состоит из двух каналов (основного и резервного) и общего накопителя. Сообщения поступают от двух источников с интенсивностями 0,1 и 0,5 (закон распределения экспоненциальный). Приоритет сообщений от первого источника выше, чем от второго. При нормальной работе сообщения передаются по основному каналу за 4,5 ± 0,5 единиц времени (закон распределения равномерный). В основном канале происходят сбои через 250 ± 50 единиц

времени (закон распределения равномерный). Если сбой происходит во время передачи, то сообщение передается по резервному каналу с самого начала. Восстановление основного канала занимает в среднем 20 единиц времени, СКО равно 2 (закон распределения нормальный). Сообщение передается по резервному каналу только в случае сбоя. Одновременно по основному и резервному каналам может быть передано только одно сообщение. Смоделировать работу магистрали передачи данных в течение 10 000 единиц времени. Оценить загрузку каналов, количество сообщений, переданных по основному и резервному каналам, характеристики очереди сообщений. Сделать выводы об эффективности системы, сформулировать и обосновать рекомендации по оптимизации системы.

12. Моделируется работа планировщика заданий в облачной системе, имеющей три типа вычислителей: традиционные *CPU*, многоядерные *GPU* и специализированные *FPGA*. На вход планировщика с бесконечной очередью поступает четыре потока заданий. Интенсивности потоков заданы в табл. 2. Время обработки зависит от типа потока и типа обслуживающего прибора (табл. 2). Все потоки, протекающие в системе, пуассоновские. Задания хранятся в общей очереди планировщика и попадают на первый освободившийся прибор обслуживания. Смоделировать работу системы в течение 1000 единиц времени. Оценить характеристики загрузки вычислителей и очереди заданий. Сделать выводы.

Таблица 2

Исходные данные

Номер потока	Интенсивность поступления	Интенсивность обслуживания		
		CPU	*GPU*	*FPGA*
1	0,8	0,7	1,4	1,6
2	0,7	0,5	1,2	1,2
3	0,6	0,4	0,8	0,7
4	0,3	0,3	0,4	0,5

13. Ввести модификации в постановку задачи п. 12:

– потоки заданий имеют разный приоритет: 1–4 в соответствии с номером потока;

– очередь заданий ограничена: в очереди планировщика не более 30 заданий; в случае превышения очереди задание теряется;

– традиционные и многоядерные вычислители обслуживают задания первого, второго и третьего потоков; четвертый поток заданий обрабатывается специализированными вычислителями.

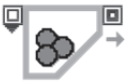
Исследуйте разные варианты работы системы. Сравните эффективность вариантов работы планировщика по критерию максимизации количества обслуженных заявок за время моделирования.

2.10. Инструменты для объединения и разделения потоков заявок (элементов)

2.10.1. Объединение и разделение элементов

При моделировании дискретно-событийных систем часто возникает задача объединения и/или разделения используемых элементов (заявок) и ресурсов. Примером могут служить сборка изделия из узлов на производстве, объединение корабля с буксирами на время швартовки с последующим разделением в порту судна и буксиров, упаковка партии товара в тару для транспортировки и т. п.

Для построения моделей с объединением и/или разделением элементов используются два блока, включенных в библиотеку *Item.lix*:



Batch (объединение или группировка) позволяет объединять несколько элементов в единый для последующего использования, т. е. поступающие на вход элементы заменяются одним на выходе. В дальнейшем, если требуется, объединенный элемент может быть разъединен;

Unbatch (разделение или разгруппировка) позволяет разъединить один входящий элемент на несколько. В зависимости от настроек блок может разделять элементы, объединенные ранее, или дублировать входящий элемент в несколько выходящих.

Объединение позволяет сгруппировать несколько элементов из различных источников в один. При этом исходные элементы удаляются, а их параметры могут быть как удалены, так и перенесены на создаваемый элемент в зависимости от настроек.

Число элементов, необходимых для объединения, называется размером группы (*batch size*). Он может быть фиксированным или динамически меняться в зависимости от внешних факторов (например, времени, отведенном на группировку). Также может меняться и назначение объединения элементов: постоянное, когда объединенный элемент является готовым изделием или продуктом, и временное, когда несколько элементов объединены только для совершения какой-нибудь групповой операции. Например, завод изготавливает изделие и упаковывает в коробку с сопроводительной документацией. Затем партия коробок помещается в более крупную тару. В первом случае будет постоянное объединение, так как в таком виде товар поступит в продажу, во втором – тару транспортируют до магазина и расформируют.

В диалоговом окне блока *Batch* три основные вкладки для настройки параметров работы блока.

1. Вкладка *Batch*, в которой задается поведение блока (*Select block behavior*):

– опция *Batch items into a single item*. При установке этой опции создается группа из элементов, поступающих на вход блока. Количество необходимых для формирования группы элементов задается на этой же вкладке в таблице *Specify the quantity needed for each batch*;

– опция *Match items into a single item*. При установке этой опции входящие элементы группируются в один выходящий с едиными значениями параметров. Опция может применяться, например, при объединении элементов по серийному номеру или номеру заказа. При этом не имеет значения, на какой вход поступил элемент.

Во вкладке *Batch* также доступна таблица *Specify the quantity needed for each batch*, в которой задается: количество элементов *Quantity needed*, необходимое по каждой ветке для формирования группы; текущее количество элементов по каждой ветке *Quantity in Block*; *Delay Kit* – опция, ограничивающая поступление указанных элементов на входной коннектор блока, пока на всех коннекторах не будет достаточного количества элементов для объединения.

2. Вкладка *Options*, в которой задаются параметры блока:

– опция *Preserve uniqueness* используется для сохранения значений свойств элементов для последующего разделения;

– опция *Use quantity input connectors* служит для задания размера группы на входах блока. При включении опции рядом с каждым входным коннектором появляется дополнительный входной коннектор

(маленький квадратик), в котором отображается текущее количество элементов, поступивших на коннектор. Если дополнительный входной коннектор подключен к какому-либо блоку, например *Lookup Table*, то значения для объединения будут считываться с него. В противном случае будут использоваться значения из таблицы вкладки *Batch*. После включения данной опции становятся доступны настройки поведения блока при изменении количества элементов для объединения:

– *Dynamically as batch is created* – если изменение количества будет выполнено до того как произошло непосредственное объединение, то количество необходимых элементов будет изменено сразу;

– *By first item at each connector* – значение количества элементов для объединения не будет изменяться, после того как первый элемент поступил в блок, новые настройки применятся для следующей группы;

– опция *Show Demand connector* позволяет создать группу по требованию или расписанию.

3. Вкладка *Properties*, в которой задаются свойства объединенного элемента.

Разделение может применяться для элементов, объединенных ранее, или для копирования элементов, которые не были сгруппированы. Может быть использовано в следующих ситуациях:

– возвращение ресурсного элемента в блок *Resource Item*;

– разделение элементов, которые были временно сгруппированы, для того чтобы они могли быть обработаны одновременно;

– создание элементов на основе одного эталонного элемента;

– создание логического дублирующего элемента для вызова какого-то действия в модели, в то время как основной физический элемент будет продолжать обрабатываться.

Если до этого элементы были объединены с опцией *Preserve uniqueness*, доступной в диалоговом окне блока *Batch*, то их первоначальные значения параметров буду восстановлены (также для этого надо выбрать одноименную опцию в блоке *Unbatch*). Однако следует быть осторожными при применении блоков, меняющих значения параметров элементов, на пути из блока *Batch* в блок *Unbatch*.

Во вкладке *Unbatch* диалогового окна блока можно задать поведение блока *Select block behavior*:

– опция *Create multiple items* задана по умолчанию. В случае использования параметра стоимости в модели стоимость распределяется по выходным элементам, как указано в таблице *Select how to define property values of outgoing items* на вкладке *Properties*;

– опция *Release cost resources*. Если элемент был объединен с ресурсом, имеющим заданную стоимость, то информация о стоимости сохраняется. При разъединении ресурс будет выдаваться на выходной коннектор, соответствующий входному коннектору блока *Batch* при его объединении.

Во вкладке *Unbatch* также доступна таблица *Specify the quantity to unbatch*, в которой задается количество элементов, передаваемых через каждый выходной коннектор блока при разъединении.

Пример 15. Моделируется процесс сборки изделия из нескольких комплектующих. Запрос на сборку изделия поступает по нормальному закону распределения с параметрами: средний интервал между запросами один час, СКО – 0,5 часа. Изделие может быть равновероятно одного из трех типов. В зависимости от типа изделия определяется необходимое количество комплектующих типа *A* и типа *B*, а также время сборки (табл. 3). Комплектующие типа *A* и *B* поступают каждые 10 часов по 20 единиц. Одновременно может собираться по пять изделий.

Таблица 3

Исходные данные

Тип изделия	Количество комплектующих типа *A*	Количество комплектующих типа *B*	Время сборки
1	0	2	4
2	3	1	5
3	1	3	6

Смоделировать процесс сборки в течение 1000 часов. Определить характеристики очереди запросов на сборку и очередей комплектующих типа *A* и *B* и оценить эффективность системы.

Конечный вид модели изображен на рис. 54.

Поясним основные моменты, связанные с разработкой модели.

1. Входной поток запросов описывается блоком *Create*. Во вкладке *Create* диалогового окна блока задается опция *Create items randomly*, закон распределения Normal с параметрами *Mean* = 1 и *Std.Dev* = 0,5.

2. Входные потоки комплектующих *A* и *B* описываются блоками *Create*. Во вкладке Create диалогового окна блока задается опция *Create items by Schedule*, указывается в таблице необходимое количество элементов *Item Quantity* = 20 и включается опция *Repeat the schedule every 10 hours*.

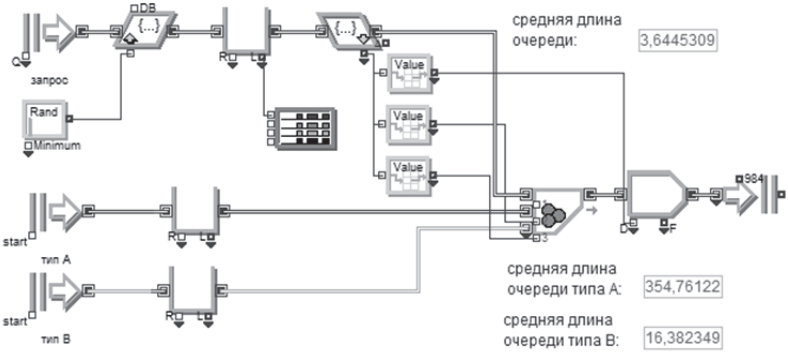


Рис. 54. Модель процесса сборки (пример 15)

3. В блоке *Random Number* во вкладке диалогового окна *Distributions* задается тип заявки: заявка одного из трех типов (1, 2, 3), тип заявки определяется по равномерному закону *Uniform*,*Integer* с параметрами *Minimum* = 1, *Maximum* = 3.

4. В блоке *Set* во вкладке *Set Properties* диалогового окна задается имя атрибута *Request Type*, определяющего тип заявки. Соответственно в блоке *Get* во вкладке *Get Properties* диалогового окна выбирается заданное в блоке *Set* имя атрибута.

5. В блоках *Lookup Table* задается количество комплектующих *A* и *B* для сборки изделия определенного типа и время сборки. Тип изделия считывается с информационного выходного коннектора блока *Get*. В диалоговом окне блока *Lookup Table* во вкладке *Table* задается таблица, в которой каждому типу изделия соответствует количество комплектующих или время сборки согласно данным табл. 3.

6. В блоке *Batch* во вкладке *Batch* диалогового окна в таблице задается необходимое количество *Quantity Needed* = 1 элементов для сборки по верхней ветке, соответствующее запросу на сборку. Количество комплектующих *A* и *B* для сборки считывается с дополнительных входных коннекторов блока (маленькие квадратики), соединенных с выходами соответствующих блоков *Lookup Table*.

7. В блоке *Activity* во вкладке *Process* диалогового окна задается опция *from the* «*D*» *connector*, в результате чего время сборки изделия считывается с информационного коннектора *D* блока, связанного с выходом блока *Lookup Table*. Также задается максимальное количество одновременно собираемых изделий: *Maximum items in activity* = 5.

На рис. 55 представлен график зависимости длины очереди запросов на сборку от времени.

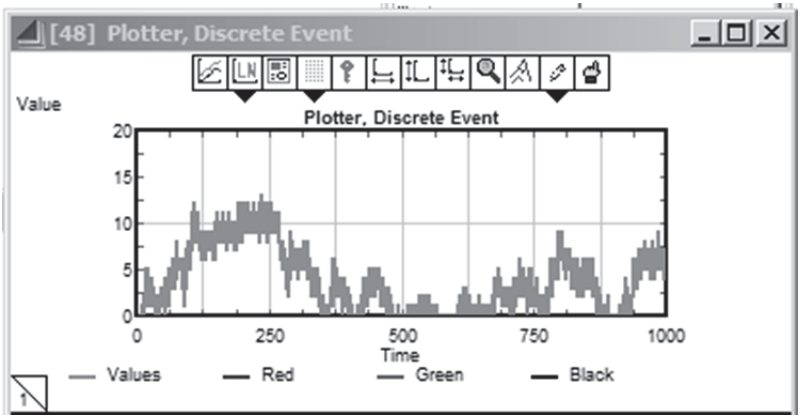


Рис. 55. График изменения длины очереди запросов во времени (пример 15)

Очевидно, что очередь запросов в течение моделирования не растет и колеблется около среднего значения, равного 3,6 единицы, что можно считать приемлемым результатом. Однако количество комплектующих избыточно. Средняя длина очереди комплектующих типа *A* составляет 354,7 единицы, *B* – 16,3 единицы в конце моделирования. Поэтому необходимо сократить поступление в систему комплектующих, особенно типа *A*. Оптимальные параметры работы системы могут быть подобраны путем проведения имитационного эксперимента с моделью системы.

2.10.2. Свойства элементов при объединении и разделении

Для каждого элемента могут быть заданы атрибуты (свойства элемента). Например, приоритет, элемент анимации, время обработки, тип заявки и т. п. При объединении элементов в группу необходимо определить, какими свойствами будет обладать объединенный элемент.

В диалоговом окне блока *Batch* во вкладке *Properties* можно задать свойства объединенного элемента. В таблице *Select how to define property values of outgoing items* в первой колонке перечислены все свойства элементов в модели, во второй колонке в выпадающем списке

показаны возможные варианты, которые могут быть установлены для значений свойств объединенного элемента. На рис. 56 изображен вид вкладки *Properties* для модели примера 15.

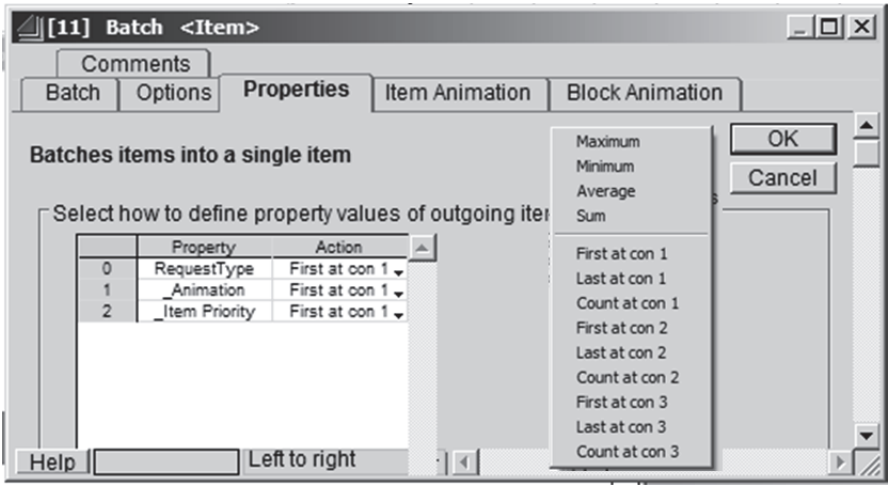


Рис. 56. Вкладка Properties блока Batch (пример 15)

Варианты значений, которые могут быть установлены для свойств объединенного элемента:

– *Maximum* – наибольшее значение свойства, найденное в любом из элементов объединения;

– *Minimum* – наименьшее значение свойства, найденное в любом из элементов объединения;

– *Average* – среднее значение свойства всех элементов, входящих в объединение;

– *Sum* – суммарное значение свойства всех элементов, входящих в объединение;

– *First at con X* (по умолчанию) – значение свойства первого элемента, который поступил на коннектор *X*; коннектор 1 является самым верхним;

– *Last at con X* – значение свойства последнего элемента, который поступил на коннектор *X*;

– *Count at con X* – значение, равное общему количеству элементов, поступивших на коннектор *X*.

Например, атрибут анимации (*_Animation*) хранит индексы анимации *2D* и *3D* картинок для элементов, движущихся в модели. Обратите внимание, что атрибут имеет значения только *First at con X* или *Last at con X*.

При разделении элементов в диалоговом окне блока *Unbatch* во вкладке *Properties* указывается, как определяются свойства элементов после разгруппировки. Доступны следующие варианты:

– *Preserved value* – каждому элементу назначается первоначальное значение свойства (до объединения), если была также выбрана опция *Preserve uniqueness* в диалоговом окне блока *Batch*;

– *Batched value* – каждому элементу будет присвоено свойство объединенного элемента;

– *Distribution* – значение, полученное при объединении, будет разделено поровну между всеми элементами.

2.10.3. Объединение элементов с помощью сопоставления

Выбор опции *Match items into a single item* в диалоговом окне блока *Batch* позволяет указать атрибут, по которому элементы должны сопоставляться, и атрибут, определяющий размер группы. При использовании этой опции каждая группа будет состоять из элементов, чьи значения атрибута имеют одинаковое значение; атрибут размера первого элемента в группе указывает на то, сколько всего элементов в ней.

Когда эта опция активна, не имеет значения, на какой вход поступил элемент. По мере поступления элементов в блок они сопоставляются друг с другом на основе значения атрибута до тех пор, пока общее количество элементов не будет равно размеру группы. После этого они объединяются и покидают блок.

Пример 16. В систему поступают заявки по равномерному закону распределения в интервале (3 ± 2) мин. Для каждой заявки создается одна копия. Заявка и копия проходят параллельную обработку в двух каналах обслуживания с одинаковой интенсивностью обслуживания (6 ± 2) мин по равномерному закону распределения. После обработки заявка и копия собираются в один пакет, который обслуживается третьим каналом с интенсивностью (6 ± 1) минут по равномерному закону распределения. Смоделировать работу системы в течение 200 мин и оценить ее эффективность.

Конечный вид модели изображен на рис. 57.

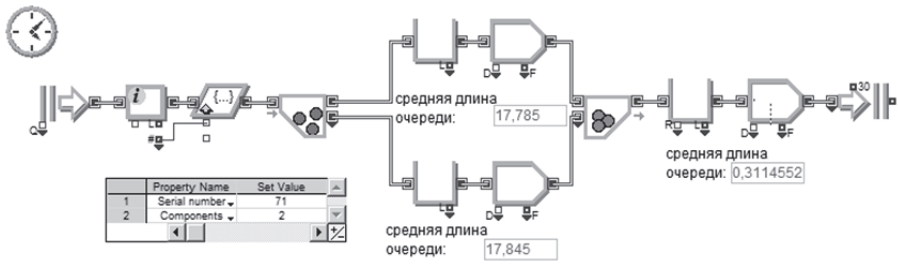


Рис. 57. Модель системы (пример 16)

Поясним основные моменты построения модели.

1. Входной поток заявок описывается блоком *Create*. Во вкладке *Create* диалогового окна блока задается опция *Create items randomly*, закон распределения *Uniform*,*Real* с параметрами *Minimum* = 1, *Maximum* = 5. Аналогично задаются параметры процесса обработки заявок в блоках *Activity*.

2. В блоке *Set* во вкладке *Set Properties* диалогового окна задаются имена атрибутов *Serial number* и *Components*, определяющих соответственно порядковый номер заявки и количество элементов в группе, равное двум (заявка + копия). Порядковый номер заявки считывается с информационного коннектора блока *Information*.

3. В блоке *UnBatch* во вкладке *UnBatch* диалогового окна в таблице задается количество элементов каждого типа после разделения (одна заявка и одна копия заявки).

4. В блоке *Batch* во вкладке *Batch* диалогового окна выбирается опция *Match items into a single item*, выбирается атрибут для объединения элементов *Match on attribute*: *Serial number* и атрибут *Get batch size from attribute*: *Components*, определяющий размер группы.

В результате моделирования обработано 30 заявок, хотя на вход поступила 71 заявка. «Узким» местом системы является параллельная обработка заявки и ее копии на первом этапе. Средняя длина очереди на обработку: 17,8. На заключительном этапе задержки в обслуживании не наблюдается, средняя длина очереди 0,3. Таким образом, система недостаточно эффективна, необходимо повысить интенсивность обработки заявок на первом этапе обслуживания.

2.10.4. Объединение и разделение с динамическим размером группы

В предыдущих моделях группа объединенных элементов имела постоянный размер. Также в *ExtendSim* реализована возможность динамически менять размер группы в процессе моделирования в зависимости от условий.

Пример 17. Поток посетителей, пришедших на мероприятие, пуассоновский, со средним значением, равным пяти минутам. Посетители объединяются в группы от 5 до 10 человек. Мероприятие длится в среднем 40 мин (закон распределения экспоненциальный), после чего все расходятся. Смоделировать работу системы в течение рабочего дня (8 ч).

Конечный вид модели показан на рис. 58.

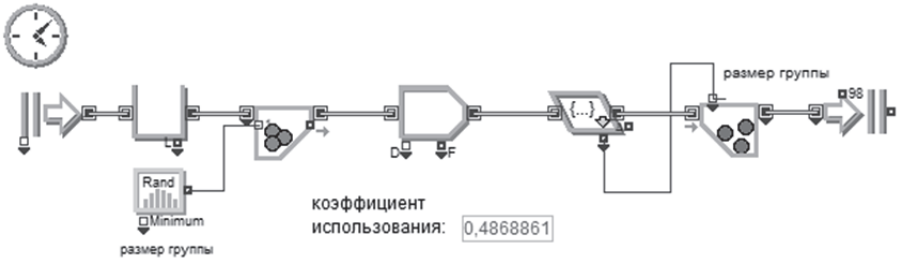


Рис. 58. Модель системы (пример 17)

Поясним основные моменты построения модели.

1. Входной поток посетителей описывается блоком *Create*. Во вкладке *Create* диалогового окна блока задается опция *Create items randomly*, закон распределения *Exponential* с параметрами *mean* = 5, *location* = 0. Аналогично задаются параметры процесса обслуживания в блоке *Activity*.

2. В блоке *Random Number* во вкладке диалогового окна *Distributions* задается размер группы, распределенный по равномерному закону: *Uniform,Integer* с параметрами *Minimum* = 5, *Maximum* = 10.

3. В блоке *Batch* во вкладке *Options* диалогового окна выбирается опция *Use quantity input connectors* и задается атрибут, в котором хранится размер группы: *Store number of items in batch in attribute*: *batchsize*.

4. В блоке *Get* во вкладке *Get Properties* диалогового окна задается имя атрибута *batchsize*, хранящего размер группы.

5. В блоке *UnBatch* во вкладке *UnBatch* диалогового окна выбирается опция *Use unbatch quantity connectors*. Верхний входной коннектор блока связывается с информационным коннектором блока *Get* для считывания размера группы.

По результатам моделирования мероприятие посетили в течение 8-часового рабочего дня 98 человек. Обслуживание групп заняло около четырех часов (коэффициент использования: 0,48).

2.11. Инструменты для моделирования ресурсов

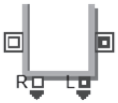
2.11.1. Использование ресурсов при моделировании

Часто для моделирования процессов, протекающих в системе, требуется описать не только потоки заявок, но и так называемые ресурсы. Например, может потребоваться выделение оперативной памяти или места на жестком диске для обработки задания в вычислительной системе или выделение причалов для входа корабля в порт и т. п. Ресурсы обеспечивают сервис для заявок в модели, их наличие или отсутствие может наложить ограничения на обработку заявок.

Одна из главных причин моделирования процесса – анализ наличия и использования ресурсов для определения влияния ресурсных ограничений на пропускную способность системы. Результатом моделирования являются сведения о том, насколько эффективно используются имеющиеся ресурсы и что произойдет в условиях их недостатка. Основная цель моделирования заключается в попытке повысить эффективность использования ресурсов, не вызывая слишком длинных промежутков ожидания обслуживания, или определить, как уменьшить задержку в обработке заявки без привлечения дополнительных ресурсов.

В систему *ExtendSim* включено несколько блоков, обеспечивающих работу с ресурсами. Приведем их описание.

Resource Pool (*Item.lix*) – представляет пул ресурсов; хранит заданное количество ресурсов и предоставляет запрашиваемое количество ресурсов блоку *Queue* (режим *Resource pool queue*).



Queue (*Item.lix*) режим *Resource pool queue* – предназначен для ожидания заявками необходимых ресурсов из пула; когда все ресурсы выделены, блок проверяет доступность принимающего блока и только потом выпускает заявку.


Resource Pool Release (*Item.lix*) – освобождает заданное количество единиц заданного ресурса, делая его доступным для повторного использования и увеличивая значение счетчика ресурсов в блоке *Resource pool*.

Resource Item (*Item.lix*) – хранит заданное количество ресурсов, но в отличие от *Resource Pool* ресурсы представляются как отдельные элементы для использования в модели; экземпляры ресурса могут быть объединены с заявками, ожидающими их, и разъединены в другой точке модели.


Shift (*Item.lix*) – создает график переключений, который используется для изменения емкости или остановки работы других блоков модели.

Различные блоки модели могут совместно использовать один и тот же ресурс, но не одновременно. В то время как ресурс используется в одном месте в модели, он недоступен для любых других блоков модели. Таким образом, наличие или отсутствие ресурсов накладывает ограничения на модель.

Существует два основных способа моделирования ресурсов при построении модели. Ресурсы могут быть смоделированы с использованием специализированных блоков для управления ресурсами. Этот метод имеет преимущество прямого доступа к функциям, таким как автоматический подсчет затрат и стоимости утилизации ресурсов. Другой способ заключается в использовании ограничений на емкость блоков, например, ограничить длину очереди в блоке *Queue* или ограничить число каналов в блоке *Activity*.

При использовании специализированных блоков доступны два метода представления ресурсов в модели:

– метод ***пула ресурсов***, когда определенное количество ресурсов доступно в пуле (т. е. ресурс рассматривается как поток);

– метод ***экземпляров ресурсов***, когда доступны один или несколько экземпляров ресурса (т. е. ресурс рассматривается, как отдельная единица).

2.11.2. Метод пула ресурсов

Блок *Resource Pool* ведет подсчет количества ресурсов, которые в настоящее время доступны для использования. Когда заявка входит в блок *Queue*, тот запрашивает у пула, доступно ли необходимое количество указанных ресурсов. Если доступно, то соответствующее количество ресурсов будет вычтено из пула, а элемент, ожидающий ресурсы, покинет очередь. В противном случае ждущий элемент останется в очереди до тех пор, пока ресурсы не станут доступными. В закрытых системах ресурс после использования возвращается в блок *Resource Pool*, в открытых – удаляется из системы.

Преимущества метода заключаются в следующем:

– блок *Resource Pool* не требует связей с другими блоками в модели, из-за чего один и тот же ресурс может использоваться в нескольких разных местах модели или для одной заявки может потребоваться несколько разных ресурсов;

– не требуется сложная маршрутизация, так как ресурсы являются потоком экземпляров, проходящих через модель, на которые накладываются определенные ограничения;

– заявки, ожидающие ресурсы из пула, могут быть проранжированы по приоритету либо в порядке очереди. В таком случае первой будет обслужена заявка, имеющая самый высокий приоритет.

У метода пула ресурсов есть следующие недостатки:

– метод пула ресурсов не позволяет использовать атрибуты для отслеживания информации об отдельных ресурсах;

– при использовании метода пула ресурсов сложнее контролировать использование конкурирующих ресурсов с несколькими разными очередями.

Пример 18. В систему обработки данных поступают задания в среднем через 10 мс, которые последовательно обрабатываются двумя

устройствами. Устройства используют для обработки соответственно 1 и 2 единицы оперативной памяти и 1 и 3 единицы записывающего устройства. Всего в системе пять единиц оперативной памяти и пять единиц записывающего устройства. Обработка на первом устройстве занимает 4 мс, на втором – 5 мс. Интервалы поступления и обработки заданий распределены по экспоненциальному закону.

Смоделировать работу системы в течение 600 мс. Определить коэффициент использования ресурсов системы, построить график изменения коэффициента использования ресурсов во времени.

Конечный вид модели представлен на рис. 59.

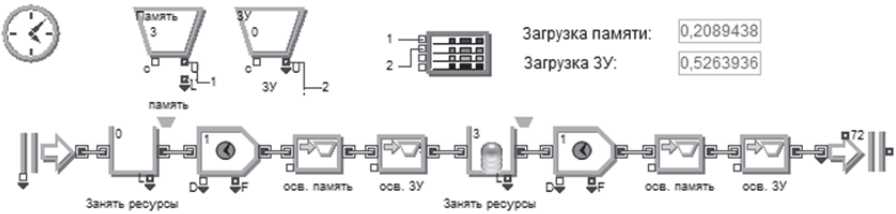


Рис. 59. Модель системы (пример 18)

Основные настройки блоков модели следующие.

1. Задать в диалоговом окне блоков *Resource Pool* имя ресурса *Pool name*: память и ЗУ – записывающее устройство; установить количество выделенного ресурса *Initial number*: 5.

2. Задать в диалоговом окне блоков *Create* и *Activity* режим *Create items randomly*, распределение *Exponential* с заданными параметрами *mean* и *location* = 0.

3. Задать в диалоговом окне блоков *Queue* тип очереди *Select queue behavior*: *resource pool queue* (очередь, управляемая ресурсами); выбрать режим *All resourcece pools are required* (одновременно требуется выделить оперативную память и записывающее устройство); установить в таблице *Select resource pools and set quantity* имя выделяемого ресурса *Resource Pool* (память или ЗУ) и его количество *Quantity*.

4. Задать в диалоговом окне блоков *Resource Pool Release* имя освобождаемого ресурса *Release by*: *name* (память или ЗУ) и количество освобождаемого ресурса *Release quantity*: *fixed number*. Количество освобождаемого ресурса должно соответствовать количеству выделенного ресурса в блоке *Queue*.

5. Создать именованные соединения 1 и 2 от выходного коннектора пулов ресурсов *U*, передающего текущий коэффициент использования ресурсов на график *Plotter*.

В результате запуска модели будет построен график использования ресурсов системы в процессе моделирования (рис. 60). Очевидно, что ресурсы в системе используются не полностью, есть резервы. Оперативная память загружена в среднем на 21 %, записывающее устройство – на 53 %. По графику можно приближенно определить момент, после которого система начинает функционировать в стационарном режиме: 220…230 мс после начала моделирования. До этого момента система функционировала в переходном режиме, и загрузка ресурсов системы менялась во времени.

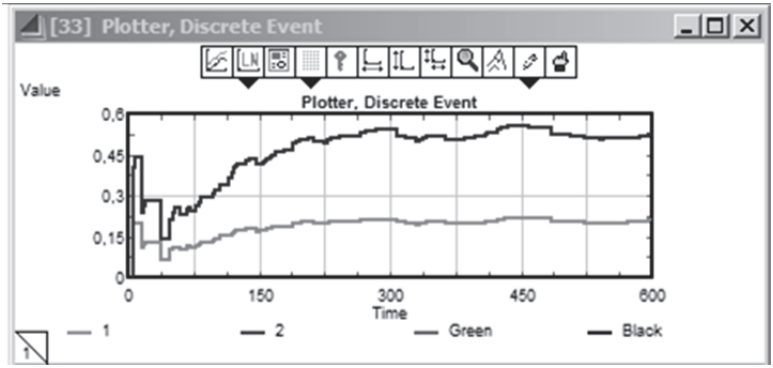


Рис. 60. График загрузки ресурсов системы (пример 18)

2.11.3. Метод экземпляров ресурсов

Еще один метод моделирования работы с ресурсами основан на использовании блока *Resource Item*. Ресурс представляется отдельным экземпляром, который служит для предоставления услуг для других элементов в модели. Количество первоначально доступных ресурсов задается в диалоговом окне блока *Resource Item*. Для использования данного ресурса необходимо его объединение с обслуживаемой заявкой (осуществляется с помощью блоков *Batch* и *Unbatch*). Если ресурс объединен с какой-либо заявкой, то он не может использоваться больше нигде в модели. Если ресурс недоступен, заявка будет ждать освобождения ресурса. Как и в методе пула ресурсов, движение заявок в модели регулируется в зависимости от наличия или отсутствия ресурсов.

При моделировании закрытой системы ресурс после использования должен быть разъединен с обслуживаемой заявкой и отправлен обрат-

но в блок *Resource Item*, после чего он может быть использован повторно. В открытой системе, например, где ресурс является расходным материалом, он может остаться объединенным с заявкой.

После возвращения ресурса в блок *Resource Item* в его атрибутах могут содержаться сведения, не относящиеся к нему, полученные при объединении и разделении. Блок *Resource Item* позволяет очищать или сохранять эти атрибуты (по умолчанию осуществляется очистка).

Экземпляр ресурса может иметь свойства (атрибуты), такие как приоритет, количество, индекс анимации, как и любой другой элемент. По этой причине данный метод является предпочтительным по сравнению с методом пула ресурсов, если необходимо отслеживать информацию о ресурсах.

Недостатки метода следующие:

– блок *Resource Item* должен быть подключен в модели и соединение должно быть таким, чтобы экземпляры ресурсов могли быть объединены с заявками, ожидающими их;

– экземпляры ресурса не могут отслеживать заявки, ожидающие их; для того чтобы направить ресурс в нужный блок *Batch* или *Unbatch*, необходимо использовать блоки маршрутизации.

Пример 19. Моделируется процесс обслуживания клиентов операторами. Клиенты поступают в систему в среднем через 5 мин (закон распределения экспоненциальный) и обслуживаются оператором в течение $11{,}5 \pm 8{,}5$ мин по равномерному закону. Одновременно оператор работает только с одним клиентом. Всего доступны три оператора.

Смоделировать работу системы в течение суток. Определить характеристики очереди клиентов и оценить эффективность системы.

Конечный вид модели изображен на рис. 61.

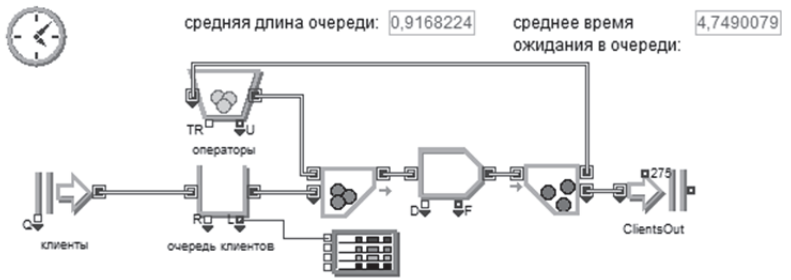


Рис. 61. Модель системы (пример 19)

Настройки основных блоков модели следующие.

1. Задать в диалоговом окне блока *Resource Item* имя ресурса: операторы; установить количество выделенного ресурса *Initial number of items*: 3.

2. Задать в диалоговом окне блоков *Create* и *Activity* режим *Create items randomly*, распределение соответственно *Exponential* и *Uniform*, *Real* с заданными параметрами. В блоке *Activity* установить максимальное количество одновременно обрабатываемых заявок *Maximum items in activity*: 3.

3. Задать в диалоговом окне блока *Batch* опцию *Batch items into a single item* и количество объединяемых элементов в таблице ниже (один оператор и один клиент).

4. Задать в диалоговом окне блока *Unbatch* опцию *Create multiple items* и количество разъединяемых элементов в таблице ниже (один оператор и один клиент).

При поступлении клиента в очередь проверяется, есть ли свободный оператор. Если оператор доступен, он изымается из пула ресурсов и собирается единая заявка, соответствующая работе с клиентом. После обработки заявка вновь разделяется, и оператор возвращается в пул ресурсов, а клиент покидает модель. Таким образом, рассмотренная система относится к комбинированному типу. Система замкнута с точки зрения использования операторов (в системе постоянно присутствуют три оператора) и разомкнута для клиентов (поток клиентов на входе теоретически бесконечен).

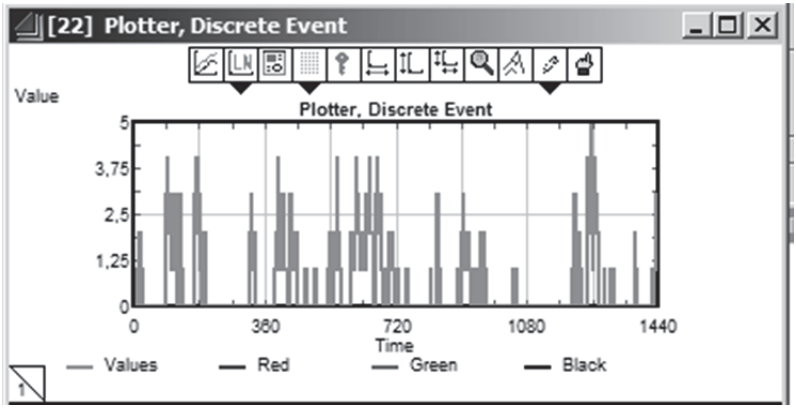


Рис. 62. График изменения длины очереди клиентов (пример 19)

По результатам моделирования можно сделать вывод об эффективности системы обслуживания клиентов: средняя длина очереди клиентов – 0,92; среднее время ожидания в очереди – 4,8 мин относительно невелико, очередь клиентов не накапливается. На рис. 62 показан график изменения длины очереди клиентов во времени.

2.11.4. Планирование использования ресурсов

Чтобы описать использование ресурсов, в имитационной модели введено понятие «логика планирования ресурсов». Задается, где, когда и как каждый ресурс должен быть использован. Например, если один экземпляр ресурса требуется в нескольких местах для объединения с другими ресурсами и элементами, то логика его планирования добавляется в модель.

Есть несколько способов планирования использования ресурсов. Некоторые способы могут применяться как с методом пула ресурсов, так и с методом экземпляров ресурсов, другие только с последним.

Существует два основных способа планирования.

1. Использование коннектора *TR* (*total resources*, общее количество ресурсов) на блоках ресурсов, который используется для изменения общего количества доступных ресурсов. Эти изменения могут быть запланированными, например во время перерыва у рабочих, или незапланированными в случае неисправности оборудования. Значение на коннекторе TR определяет, сколько ресурсов блок имеет в текущий момент.

2. Использование блока *Shift*. Блок используется для планирования размера и доступной емкости других блоков в модели. Это необходимо для моделирования ситуации, когда ресурсы включаются и выключаются по расписанию в течение длительного времени. Например, моделирование рабочих смен с регламентированными перерывами. В блоке задается таблица расписания работы. Доступны два режима работы блока:

– режим *On/Off* действует как бинарный переключатель, который включает или выключает связанные блоки в определенные моменты времени. Например, это может использоваться для выключения блоков *Activity* на время перерыва или после окончания рабочего дня;

– режим *Number* явно определяет размер емкости блока в течение времени. Например, можно установить разный размер пула ресурсов для утренней смены, во время обеда, для второй смены и т. п.

Кроме рассмотренных выше, существует еще, по крайней мере, четыре дополнительных способа планирования использования ресурсов в *ExtendSim*:

– использование блока *Resource Item* и последующего блока *Gate* для контроля выпуска экземпляров ресурсов;

– использование блока *Resource Item* и последующего *SelectItemOut* для управления направлениями движения ресурсов;

– использование комбинации блоков *Gate* и *SelectItemOut* для планирования как места, так и времени использования ресурсов;

– использование блока *QueueEquation* для задания сложной логики использования ресурсов.

Ниже приведен пример, в котором иллюстрируется применение разных блоков, моделирующих ресурсы.

Пример 20. Моделируется процесс сборки аппаратуры. Заказы на сборку поступают по экспоненицальному закону распределения со средним значением 0,1 ч. Для сборки необходима одна единица оборудования, сборка выполняется одним рабочим. Время сборки распределено по нормальному закону со средним значением 0,25 и СКО = 0,05 ч. Всего сборкой занимаются 7 рабочих, рабочая смена длится 10 ч с (8.00 до 18.00), после трех часов работы по окончании сборки очередного изделия рабочий имеет регламентированный перерыв в течение одного часа (за смену всего два перерыва), после чего продолжает работать. Первые четыре часа рабочей смены с 8.00 до 12.00 часов доступно пять единиц оборудования, далее с 12.00 часов – три единицы оборудования.

Смоделировать процесс сборки аппаратуры в течение 100 дней. Оценить эффективность процесса сборки.

Конечный вид модели показан на рис. 63.

Приведем настройки основных блоков модели.

1. В диалоговом окне верхнего блока *Create* задать режим генерации ресурсов по расписанию *Create values by schedule*, заполнить таблицу расписания, содеражащую два столбца: *Create time* и *Value* и три строки: соответсвенно 0 (время) и 5 (значение); 4 и 3; 10 и 0 (после окончания трудовой смены, через 10 часов после начала работы ресурсы оборудования недоступны). Выбрать опцию повторения расписания каждые 24 часа: *Repeart the schedule every 24 hours*.

2. В диалоговом окне блока *Resource Pool* задать имя ресурса *Pool name*: *Machinery* и исходное количество ресурса *Initial number*: 5.

Рис. 63. Модель системы (пример 20)

3. В диалоговом окне блока *Schift* устанавливается время трудовой смены. Выбираются опции: *Shift time*: *Off/On*; *Time unit*: *hours*; *Repeart schedule every 24 hours* и заполняется таблица расписания: 0 – *on*; 10 – *off*.

4. В диалоговом окне блоков *Create* и *Activity* выбрать режим *Create items randomly*, распределение соответственно *Exponential* и *Normal* с заданными параметрами. В блоке *Activity* установить максимальное количество одновременно обрабатываемых заявок по количеству рабочих: *Maximum items in activity*: 7.

5. В диалоговом окне блока *Resource Item* установить количество выделенного ресурса *Initial number of items*: 7 (количество рабочих), и в таблице ниже установить свойство ресурса *Property Name*: *hours*.

6. В диалоговом окне блока *Batch* задать опцию *Batch items into a single item* и количество объединяемых элементов в таблице ниже (одна заявка и один рабочий).

7. В диалоговом окне блока *Queue* выбрать тип очереди *resource pool queue* и установить в таблице ниже необходимое количество оборудования для обслуживания заявки: *Machinery*: 1.

8. В диалоговом окне блока *Set* выбрать имя атрибута *Wait time*, определяющего время ожидания заявки в очереди. Время ожидания считывается с информационного коннектора блока *Queue*.

9. В диалоговом окне блока *Equation* выполнить установки согласно рис. 64.

В блоке рассчитывается текущее время, которое отработал рабочий с начала трудовой смены. При расчете учитываются время выполнения заявки (считывается с информационного коннектора блока *Activity*) и время ожидания оборудования (*Wait time*).

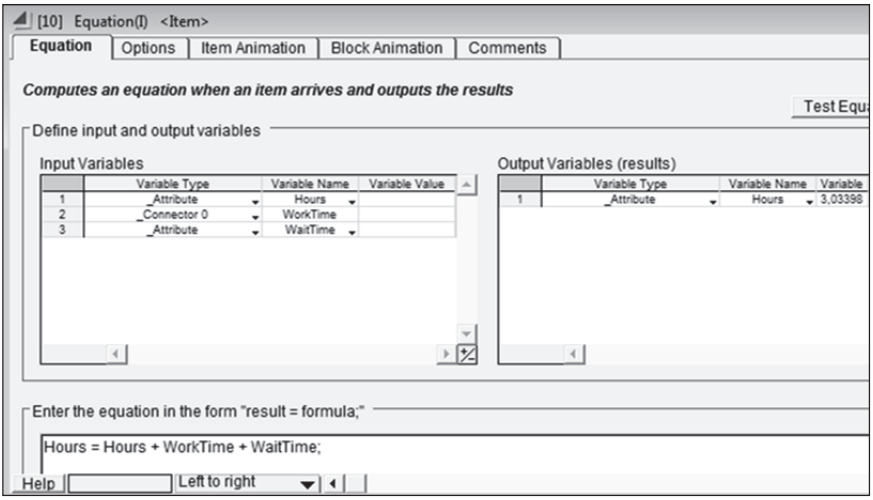


Рис. 64. Настройка блока *Equation*

10. В диалоговом окне блока *Resource Pool Release* задать имя освобождаемого ресурса *Release by*: *name Machinery* и его количество *Release quantity*: *fixed number 1*.

11. В диалоговом окне блока *Get* выбрать атрибут *Hours*, хранящий текущее время работы рабочего.

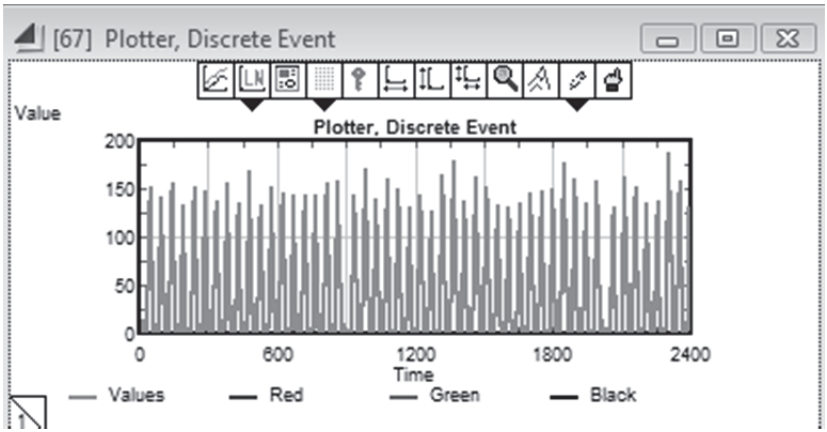


Рис. 65. График изменения длины очереди заявок во времени (пример 20)

12. В диалоговом окне блока *UnBatch* задается количество элементов каждого типа после разделения (одна заявка и один рабочий). Обработанная заявка направляется на выход, а для рабочего проверяется текущее время его работы. Если это время превышает 3 часа (проверка задается в блоке *Decision*), то рабочий отдыхает 1 час (задержка в блоке *Activity*) и возвращается к работе (в пул ресурсов), иначе рабочий сразу возвращается в пул ресурсов.

По результатам моделирования можно сделать вывод о невысокой эффективности моделируемой системы. Суммарное время обработки заявок рабочими составляет 35 % (коэффициент использования) от времени трудовой смены. Остальное время тратится на ожидание оборудования *Machinery* (длина очереди на ресурс равна 2,1). Заявки на сборку аппаратуры поступают с одинаковой интенсивностью в течение суток, а время работы – 10 часов в сутки (с 8.00 до 18.00). Поэтому к началу рабочего дня скапливаются заявки (рис. 65), которые постепенно выполняются в течение трудовой смены. Таким образом, время ожидания обработки зависит от того, в какое время суток заявка поступила.

2.12. Контрольные вопросы и задания

1. При моделировании каких ситуаций используется объединение и разделение элементов (заявок)? Приведите примеры задач.

2. Какие основные блоки и способы их использования реализованы в *ExtendSim* для объединения и разделения элементов?

3. Опишите инструментарий *ExtendSim* для работы с ресурсами. Какие способы планирования ресурсов реализованы? Приведите примеры задач.

4. Компьютер задействован в управлении технологическим оборудованием. Для контроля состояния оборудования каждые 20 мин запускается одна из трех типов задач. Через каждый 5 мин работы процессора каждая задача выводит результаты работы в базу данных. При обращении двух и более задач к базе данных (БД) образуется очередь, которая обслуживается по правилу FIFO. Общий объем памяти, выделенный для решения задач, 800 Кбайт. Компьютер работает в мультипрограммном режиме, и во время выполнения операций вывода в БД процессор может выполнять другую задачу, если она загружена в память. После последнего вывода в БД задача выгружается из памяти и завершает свою работу. Количественные характеристики системы приведены в табл. 4.

Таблица 4

Исходные данные

Тип задачи	1	2	3
Вероятность возникновения	0,5	0,35	0,15
Объем памяти, Кбайт	200	300	400
Время обработки ЦП, мин	15	20	25
Время выводы в БД, мин	3	5	7

Смоделировать работу системы в течение 10 000 минут. Оценить характеристики загрузки БД, ЦП, памяти и характеристик очереди задач к БД. Построить графики загрузки элементов системы в процессе моделирования. Сделать выводы об эффективности системы.

5. К трем компьютерам вычислительной системы с интенсивностью 4 с (закон распределения экспоненциальный) поступают пакеты данных, которые необходимо обработать и передать на запись в общее внешнее хранилище. Обработка данных длится 10 ± 1 с (закон распределения равномерный) и требует три условные единицы оперативной памяти, которая освобождается после обработки. Запись во внешнее хранилище происходит в режиме эксклюзивного доступа и длится $2,5 \pm 0,5$ с. В каждом компьютере выделено пять условных единиц оперативной памяти для обработки данных. Смоделировать обработку 1000 пакетов данных. Оценить характеристики эффективности работы вычислительной системы. Сделать выводы.

6. К компьютеру, работающему в системе управления технологическим процессом, с интенсивностью 10 мс поступает блок данных от датчиков. Обработка данных включает три этапа: запись, сравнение, выдача управляющего воздействия. Запись блока данных требует 2 ± 1 мс (закон распределения равномерный) и две единицы ресурса компьютера; сравнение длится $2,5 \pm 0,5$ мс (закон распределения равномерный) и две единицы ресурса компьютера; управляющее воздействие по результатам сравнения занимает $1,5 \pm 0,5$ мс (закон распределения равномерный) и одну единицу ресурса компьютера. Всего доступно пять единиц ресурса. Смоделировать работу системы в течение 2000 мс. Оценить характеристики очереди на каждом этапе обработки, построить графики зависимости длины очереди от времени. Выявить «узкие» места в системе, подобрать оптимальные параметры функционирования системы.

7. На сервер в интервале $11,5 \pm 1,5$ с (закон равномерный) поступают сообщения (равновероятно) одного из трех видов: пользовательские, технические и управляющие. Сервер обрабатывает поступившую информацию по блочному принципу, а именно происходит сбор пакета из отдельных сообщений с последующей обработкой. Процесс сборки пакета имеет следующую последовательность:

– компоновка пользовательского и технического сообщений в течение $1,5 \pm 0,5$ с (закон распределения равномерный) и формирование в результате неразъемной пары;

– присоединение к неразъемной паре управляющего сообщения в течение $3,5 \pm 0,5$ с (закон распределения равномерный).

Далее собранный пакет обрабатывается равномерно в течение $17,5 \pm 2,5$ с. Одновременно может обрабатываться пять пакетов. Смоделировать работу системы в течение 1000 с. Оценить эффективность системы, сделать выводы.

8. Сообщения поступают на сервер с интенсивностью 2 мс (закон распределения экспоненциальный). Для дальнейшей обработки и передачи по каналу связи сообщения группируются по 7. Пакет сообщений обрабатывается с интенсивностью 10 мс (закон распределения экспоненциальный) и передается по каналу связи 13 ± 1 мс (закон распределения равномерный). После передачи пакет разгруппировывается на исходные сообщения. В процессе разгруппировки 3 % сообщений считаются утерянными. Смоделировать работу системы в течение 2000 мс. Определить по результатам моделирования загрузку системы на всех этапах, количество потерянных и переданных сообщений.

9. На обработку от двух источников поступают заявки с интенсивностью 0,2 и 0,3 (закон распределения экспоненциальный). Заявки от первого источника имеют приоритет в обслуживании. Далее заявки группируются, размер группы распределен равномерно в интервале 4 ± 1. Далее заявки обрабатываются двухканальным устройством с интенсивностью 0,1 (закон распределения экспоненциальный). Смоделировать работу системы в течение 1000 единиц времени. Оценить характеристики эффективности системы. Сделать выводы.

10. Моделируется работа грид-системы. Поступают заявки от пользователей в среднем через 3 мс по экспоненциальному закону распределения. Заявки предварительно обрабатываются в течение $0,6 \pm 0,1$ мс по равномерному закону распределения первым компьютером. Далее

заявки направляются на параллельную обработку по двум направлениям. По первому направлению заявки последовательно обрабатывают второй и третий компьютеры в среднем соответственно 1 и 1,5 мс по экспоненциальному закону распределения. По второму направлению заявки обрабатывает четвертый компьютер в среднем 2,8 мс по экспоненциальному закону распределения. После обработки происходит объединение результатов и их отправка пользователю пятым компьютером за 1 мс. Исследовать работу системы в течение 1000 мс. Сделать выводы о загруженности системы.

11. В вычислительную систему от датчиков поступают сообщения типа *A* и *B*. Сообщения типа *A* поступают равномерно в интервале $12,5 \pm 2,5$ мс, сообщения типа *B* – равномерно в интервале 13 ± 1 мс. Перед обработкой сообщения группируются в задания, в каждом задании два сообщения типа *A* и два сообщения типа *B*. Далее задание проходит обработку на одном из двух процессоров, выбор процессора равновероятен. Обработка занимает в среднем 4 мс (закон распределения экспоненциальный). После обработки 2 % заданий бракуется и поступает на повторную обработку. Успешно обработанные задания поступают в распределитель, где задания разбиваются на исходные сообщения и записываются в хранилище по одному в течение $1,5 \pm 0,5$ мс (закон распределения равномерный). Смоделировать работу системы в течение 1000 мс. Оценить загрузку системы, характеристики очереди сообщений на всех этапах обработки, количество сообщений, отправленных на повторную обработку. Сделать выводы об эффективности работы системы.

12. Некоторая фирма производит центробежные насосы, сборка которых осуществляется по заказу покупателей. Заказы поступают в случайные моменты времени равномерно с интервалом 20 ± 2 мин. Когда поступает заказ, делается две его копии. Оригинал заказа используется для получения двигателя со склада и подготовки его для сборки (время выполнения 9 ± 3 мин). Первый экземпляр копии используется для заказа и адаптации насоса (12 ± 2 мин), а второй экземпляр для начала изготовления плиты основания ($15,5 \pm 0,5$ мин). Когда насос и плита основания готовы, производится пробная подгонка (5 ± 1 мин). Далее все три компонента собираются вместе (6 ± 1 мин). Обработка заказа на всех этапах обслуживания подчиняется равномерному закону распределения. Промоделировать сборку 100 центробежных насосов.

Единица модельного времени одна секунда. Сделать выводы об эффективности процесса сборки насосов.

13. В систему поступают заявки по равномерному закону в интервале 5 ± 2 мин. Для каждой заявки создается одна копия. Заявка и копия проходят параллельную обработку в двух каналах обслуживания равномерно в интервале 6 ± 2 мин. После обработки заявка и копия собираются в один пакет, который обслуживается третьим каналом равномерно в интервале 6 ± 1 мин. Смоделировать работу системы по обработке 100 пакетов. Оценить загрузку каналов системы, характеристики очереди заявок на этапах обработки. Оценить эффективность системы, сделать выводы.

14. Специализированная вычислительная система состоит из трех процессоров и общей оперативной памяти. Задания, поступающие на обработку в среднем через интервалы времени 5 мин (закон распределения экспоненциальный), занимают объем оперативной памяти размером в одну страницу. После трансляции первым процессором в течение 5 ± 1 мин (закон распределения равномерный) их объем увеличивается до двух страниц, и они поступают в оперативную память. Затем после редактирования на втором процессоре, которое занимает $2{,}5 \pm 0{,}5$ мин (закон распределения равномерный) на страницу, их объем возрастает до трех страниц. Отредактированные задания через оперативную память поступают в третий процессор на решение, требующее в среднем 1,5 мин, СКО = 0,4 мин (закон распределения нормальный) на страницу, и покидают систему, минуя оперативную память. Выделенный объем оперативной памяти позволяет хранить 100 страниц. Смоделировать работу вычислительной системы в течение 500 ч. Определить характеристики занятия оперативной памяти по всем трем этапам обработки задания, вероятности загрузки процессоров, среднее время прохождения задания через систему. Сделать выводы об эффективности системы.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Акопов. А.С.* Имитационное моделирование: учебник и практикум для академического бакалавриата. – М.: Изд-во Юрайт, 2014. – 389 с.

2. *Алгазинов Э.К.*, *Сирота А.А*. Анализ и компьютерное моделирование информационных процессов и систем / под ред. А.А. Сирота. – М.: Диалог-МИФИ, 2009. – 416 с.

3. *Альсова О.К.* Моделирование систем: учеб. пособие / О.К. Альсова. – Новосибирск: Изд-во НГТУ, 2007. – 72 с.

4. *Заборовский В.С.* Имитационное моделирование телематических систем: учеб. пособие / В.С. Заборовский, А.С. Ильяшенко, В.А. Мулюха. – СПб: Изд-во СПбГПУ, 2013. – 72 с.

5. *Замятина О.М.* Моделирование сетей: учеб. пособие / О.М. Замятина: Томский политехнический университет. – Томск: Изд-во Томского политехнического университета, 2011. – 168 с.

6. *Кельтон В*., *Лоу А.* Имитационное моделирование. Классика CS. – 3-е изд. – СПб.: Питер; Киев: 2004. – 847 с.

7. *Максимей И.В*. Имитационное моделирование сложных систем: учебное пособие. Часть 1. Математические основы / В.И. Максимей. – Минск: Изд. Центр БГУ, 2009. – 263 с.

8. Моделирование систем: учебник для вузов / С.И. Дворецкий, Ю.Л. Муромцев, В.А. Погонин, А.Г. Схиртладзе. – М.: Издательский центр «Академия», 2009. – 320 с.

9. *Советов Б.Я*., *Яковлев С.А.* Моделирование систем. – 7-е изд. – М.: Изд-во Юрайт, 2014. – 344 с.

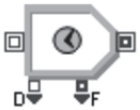
10. *Советов Б.Я*., *Яковлев С.А.* Моделирование систем. Практикум: учеб. пособие для вузов – М.: Изд-во Юрайт, 2014. – 304 с.

11. *ExtendSim/UserGide* – документация к пакету *ExtendSim*. – 1211 с.

Приложение

БЛОКИ БИБЛИОТЕКИ *Item.lix*

Описание блоков библиотеки Item.lix

Пиктограмма и название блока	Назначение блока
 *Executive*	Наличие этого блока **обязательно**! для всех дискретно-событийных систем, так как именно этот блок отвечает за отсчет модельного времени и, как следствие, за синхронизацию всех событий, происходящих в системе. По сути, блок представляет собой таймер и позволяет управлять процессом моделирования. Блок должен быть расположен левее всех блоков, расположенных в модельном окне
 *Create*	Генерирует заявки через определенные интервалы времени. Интервалы могут быть заданы как константой, так и являться случайной величиной, распределенной по какому-либо закону
 *Queue*	Накапливает заявки и позволяет имитировать различные дисциплины обслуживания очереди: по порядку, по приоритету, по атрибуту, по количеству выделенного ресурса. Имеет информационные соединители (*L*), благодаря которым можно выводить информацию о характеристиках очереди (время ожидания в очереди, средняя длина очереди и т. д.) в процессе моделирования на графики
 *Queue Equation*	Задерживает заявки на интервал времени, вычисленный на основе введенного пользователем уравнения
 *Activity*	Задерживает одну или несколько заявок на заданное время, тем самым имитируя какой-либо процесс обработки. Реализована возможность задания определенного числа одновременно обрабатываемых заявок. Можно также задать закон распределения времени обработки заявки, взять значения из таблицы *Lookup Table*, задать постоянное время (константу), использовать параметры заявки или получить время обслуживания с информационного входа *D*, что бывает полезно, например, при изменении интенсивности обслуживания в течение суток

Продолжение таблицы

Пиктограмма и название блока	Назначение блока
 *Select Item Out*	Позволяет разделять потоки событий (заявок). Обладает набором критериев разделения. Например, можно задать процент заявок, приходящийся на каждый выход, или разделять заявки по уникальному свойству. Количество выходов может быть увеличено, для этого надо навести курсор мыши на стрелку, зажать кнопку мыши и потянуть вниз
 *Select Item In*	Позволяет соединять потоки событий (заявок). Обладает набором критериев соединения. Количество входов может быть изменено аналогично блоку *Select Item Out*
 *Exit*	Является точкой выхода заявки из системы. Содержит в себе информацию о количестве прошедших сквозь него элементов. Количество входов можно изменить
 *Trow Item*	Используется для передачи заявки в блок *Catch Item*, без использования соединительных линий
 *Catch Item*	Используется для приема заявок, отправленных с помощью блока *Throw Item*, без соединительных линий
 *Convey Item*	Имитирует конвейер, перенося заявки из одного места в другое
 *Set*	Устанавливает параметры (атрибуты) для проходящих через блок заявок, например, приоритет или время обработки
 *Get*	Отображает и выводит параметры (атрибуты) заявок, которые проходят через блок. Информация может быть выведена в виде диалога или быть послана на выходы блока. Можно работать с несколькими параметрами и несколькими выходами

Продолжение таблицы

Пиктограмма и название блока	Назначение блока
 *Cost By Item*	Просматривает и отображает стоимость обработки заявок, проходящих через этот блок
 *Cost Stats*	Выводит статистику обо всех блоках системы, которые подсчитывают стоимость
 *Equation*	Производит расчет уравнения при прохождении элемента через этот блок. Можно задать количество входов и выходов, использовать разные атрибуты заявок в качестве переменных
 *Resource Item*	Содержит и предоставляет ресурсы для модели. Может использоваться как в открытых, так и в закрытых системах
 *Resource Pool*	Содержит пул ресурсов, которые будут использоваться в моделировании
 *Resource Pool Release*	Возвращает ресурс в пул ресурсов при прохождении заявки
 *Gate*	Ограничивает прохождение заявок на основе введенных ограничений
 *History*	Просматривает и отображает информацию о проходящих заявках

Продолжение таблицы

Пиктограмма и название блока	Назначение блока
 *Information*	Выводит статистику о проходящих заявках, например, время цикла обработки или время между заявками
 *Queue Matching*	Используется для сопоставления типа различных заявок. Содержит две отдельные очереди для сравниваемых заявок
 *Read*	Считывает данные из базы данных при прохождении заявки. Можно установить количество чтений, которое должен совершить блок при поступлении заявки
 *Write*	Записывает данные в базу данных при поступлении заявки на вход. Можно установить количество записей, которое должен обработать блок при поступлении заявки
 *Unbatch*	Разбивает одну заявку на несколько (создает копию)
 *Batch*	Позволяет объединять заявки из различных источников в единый элемент-заявку
 *Shift*	Создает расписание, которое может изменять возможности других блоков с течением времени. Несколько таких блоков могут быть соединены вместе для создания сложных конструкций изменения поведения системы
 *Shutdown*	Генерирует информацию о выключении блоков в зависимости от входов или распределения времени между поломкой и ремонтом

Окончание таблицы

Пиктограмма и название блока	Назначение блока
 *Transport*	Перемещает элемент из одной точки в другую, основываясь на расстоянии и скорости. Используется для моделирования с использованием *3D* анимации
 *Workstation*	Имитирует поведение рабочей станции. Может выполнять функции блоков *Activity* и *Queue*

ОГЛАВЛЕНИЕ

Альсова Ольга Константиновна

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ СИСТЕМ В СРЕДЕ *ExtendSim*

Учебное пособие

Редактор *Л.Н. Ветчакова*
Выпускающий редактор *И.П. Бровановa*
Корректор *И.Е. Семенова*
Дизайн обложки *А.В. Ладыжская*
Компьютерная верстка *Л.А. Веселовская*

Налоговая льгота – Общероссийский классификатор продукции
Издание соответствует коду 95 3000 ОК 005-93 (ОКП)

Подписано в печать 28.01.2016. Формат 60 × 84 1/16. Бумага офсетная. Тираж 100 экз.
Уч.-изд. л. 6,04. Печ. л. 6,5. Изд. № 221/15. Заказ № 227. Цена договорная

Отпечатано в типографии
Новосибирского государственного технического университета
630073, г. Новосибирск, пр. К. Маркса, 20