ИНСТРУМЕНТ ДЛЯ ОБЪЕКТНОГО МОДЕЛИРОВАНИЯ СИСТЕМ С ДИСКРЕТНЫМИ СОБЫТИЯМИ OBJECTSIM

А. В. Приступа (Томск)

В нашей стране за последние годы наметилась положительная тенденция в области имитационного моделирования. Она проявляется в увеличении числа публикаций отечественных специалистов, появлении новых экспериментальных инструментов имитационного моделирования. Однако, как справедливо замечено В. В. Девятковым, наши разработки часто уступают в технологиях реализации зарубежным аналогам [1]. Некоторые из них не поддерживают возможности графического создания моделей, другие не предоставляют средств для написания кода, который мог бы описывать дополнительное поведение модели или, к примеру, сбор нестандартной статистики [2]. В данной работе речь пойдет об инструменте для объектного моделирования систем с дискретными событиями ObjectSim, разработка которого производилась автором с учетом современных требований, предъявляемых к программному обеспечению вообще и к системам имитационного моделирования в частности.

Разработка моделей в ObjectSim состоит из нескольких этапов. Наиболее привычным является создание стандартных компонентов, таких как источники заявок, очереди, обслуживающие устройства. Пользователь может изменять параметры работы стандартных компонентов с помощью окна настроек. Пример компонента и его настроек приведен на рис. 1.

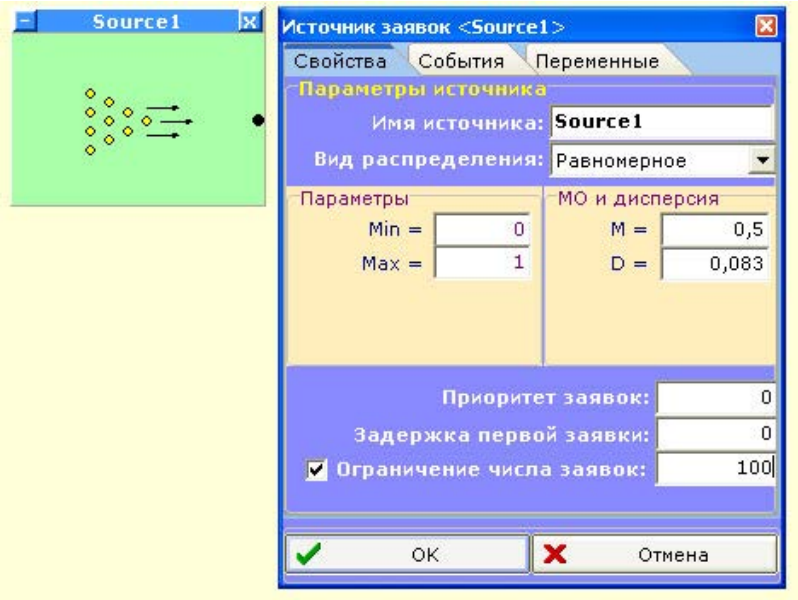

Рис. 1. Источник заявок

Вид распределения и параметры задают закон распределения времени между поступлениями соседних заявок. По введенным параметрам рассчитываются математическое ожидание и дисперсия. Ограничение числа заявок показывает максимальное число заявок, которое источник может сгенерировать. Другие стандартные компоненты реализованы подобным образом – у каждого есть собственные свойства, а также переменные и события, о которых речь пойдет ниже.

Однако с помощью стандартных компонентов можно описать лишь узкий круг моделей. Для обеспечения большей гибкости при моделировании используются так называемые компоненты пользователя, которые разработчик модели создает в специальном редакторе, а затем также помещает на форму.

Компонент пользователя включает в себя порты (для связи с другими компонентами), переменные (которые могут быть зависимыми, случайными, внешними) и константы. Более подробно такое представление изложено в [5]. Зависимые переменные изменяются по законам, которые задает разработчик, случайные – в соответствии с выбранным законом распределения, внешние переменные предоставляют доступ к переменным других компонентов. Для переменных и констант предусмотрено четыре типа: целый (vctInteger), вещественный (vctExtended), логический (vctBoolean) и специальный тип времени (vctTime). Внешний вид компонента пользователя приведен на рис. 2.

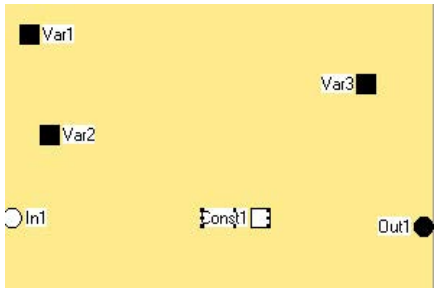

Рис. 2. Компонент пользователя

Входные порты (обозначаются кружками) имеют белый цвет, выходные – черный, переменные и константы имеют квадратную форму. Все объекты отображаются в дереве элементов, а их свойства – в инспекторе элементов, подобно тому, как это сделано в средах разработки программного обеспечения Borland Delphi, Borland C++ Builder и т.п. (рис. 3).

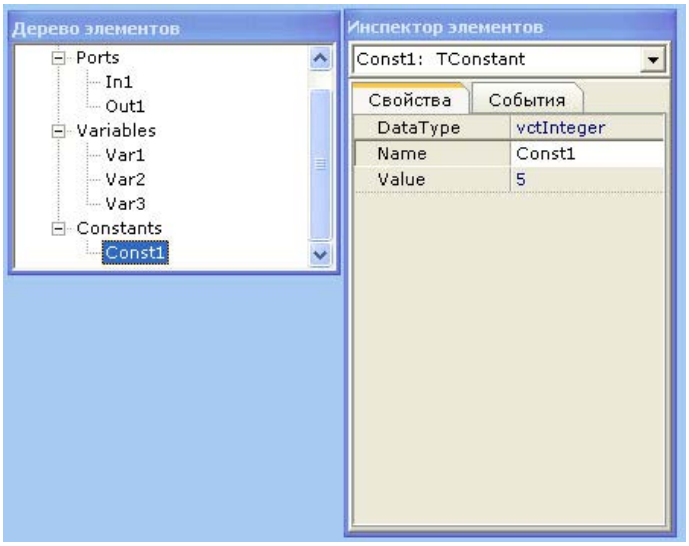

Рис. 3. Дерево и инспектор элементов

Кроме управления свойствами элементов, ObjectSim поддерживает механизм событий. Так, например, пользователь может написать собственные обработчики для событий OnEnter и OnExit, которые возникают соответственно при входе заявки в компонент и выходе из него. В качестве инструмента для написания обработчиков была выбрана библиотека FastScript, разработанная в Fast Reports Inc, позволяющая обращаться к переменным, константам, свойствам и методам классов, вызывать процедуры и функции и многое другое. В качестве языка написания кода используется PascalScript. Простейшими примерами использования обработчиков являются ситуации, когда необходимо изменить значение какой-либо переменной, например, при входе заявки в компонент (при этом могут использоваться любые математические выражения и опера-

ции, допустимые в Delphi) или изменить мАршрут движения заявок в зависимости от истинности определенного условия.

```
        if (Var1 < 20)
        then InPort1.OutPort:= OutPort1                    (1)
    else InPort1.OutPort:= OutPort2;
```

Например, допустима конструкция вида (1), направляющая заявки от первого входного порта InPort1 к первому выходному порту OutPort1 в случае, если значение Var1 меньше 20, и ко второму выходному порту OutPort2 в противном случае.

Теперь вновь вернемся к стандартным компонентам. Для них, как и для компонентов пользователя, реализован механизм событий, поскольку их параметры могут меняться во время выполнения модели. Примером может служить закон распределения времени поступления или обслуживания заявок, когда интенсивность является функцией (это может быть функция времени или функция других переменных). В этом случае в окне настроек компонента в списке доступных распределений необходимо выбрать "Собственное", задать необходимые переменные и написать обработчик события OnExit в виде, подобным (2).

```
        x:= GetValueFrom ('UserComponent1', 'Var1');
        InterArrivalTime:= F (x, …),                       (2)
```

при этом переменные, входящие в состав функции F, должны быть предварительно описаны в разделе "Переменные". Они могут иметь один из четырех допустимых типов (в том числе и vctTime), также, как и в компонентах пользователя, могут быть внешними, т.е. принимать значения переменных других компонентов.

Запуск модели может осуществляться по времени, по количеству заявок и в совокупности этих вариантов (т.е. модель остановится, когда наступит первое из событий "Превышение числа заявок" и "Истечение времени моделирования"). Более сложные условия останова модели пользователь может записать сам в обработчике событий какого-нибудь из компонентов с помощью конструкции (3).

```
        TModel.GetInstance.StopFlag:= True;                (3)
```

После создания всех компонентов пользователь должен соединить их связями, показывающими маршруты движения заявок (рис. 4).

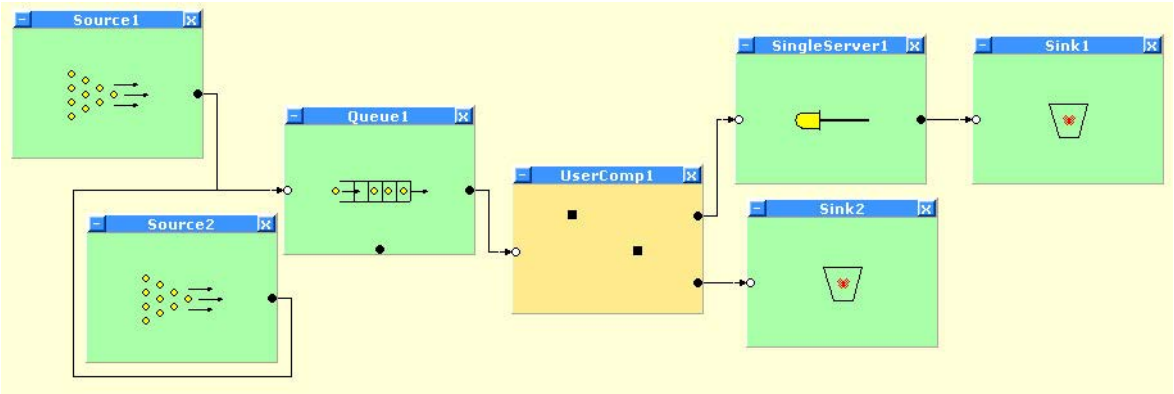

Рис. 4. Графическое представление модели

После того, как выполнение модели завершится, пользователь может получить статистику двух типов по каждому из компонентов. Первый тип включает в себя стандартные характеристики работы имитационных моделей (среднее время ожидания, средняя длина очереди, среднее время обслуживания и т. д.). Статистика второго типа собирается пользователем самостоятельно в ходе выполнения модели. Для этого перед

запуском модели он должен "известить" систему о его намерении собирать соответствующую информацию, задав количество и названия статистик, который впоследствии будут отображаться на графиках. Затем в коде в нужном месте добавляет пару (X, Y) к требуемой статистике в виде (4).

```
AddPoint ('StatisticName', X, Y).          (4)
```

Как правило, X – это текущее время или номер заявки, а Y – наблюдаемая характерисика. По окончании работы модели можно посмотреть всю накопленную статистику в текстовом виде и в виде графиков (рис. 5, 6).

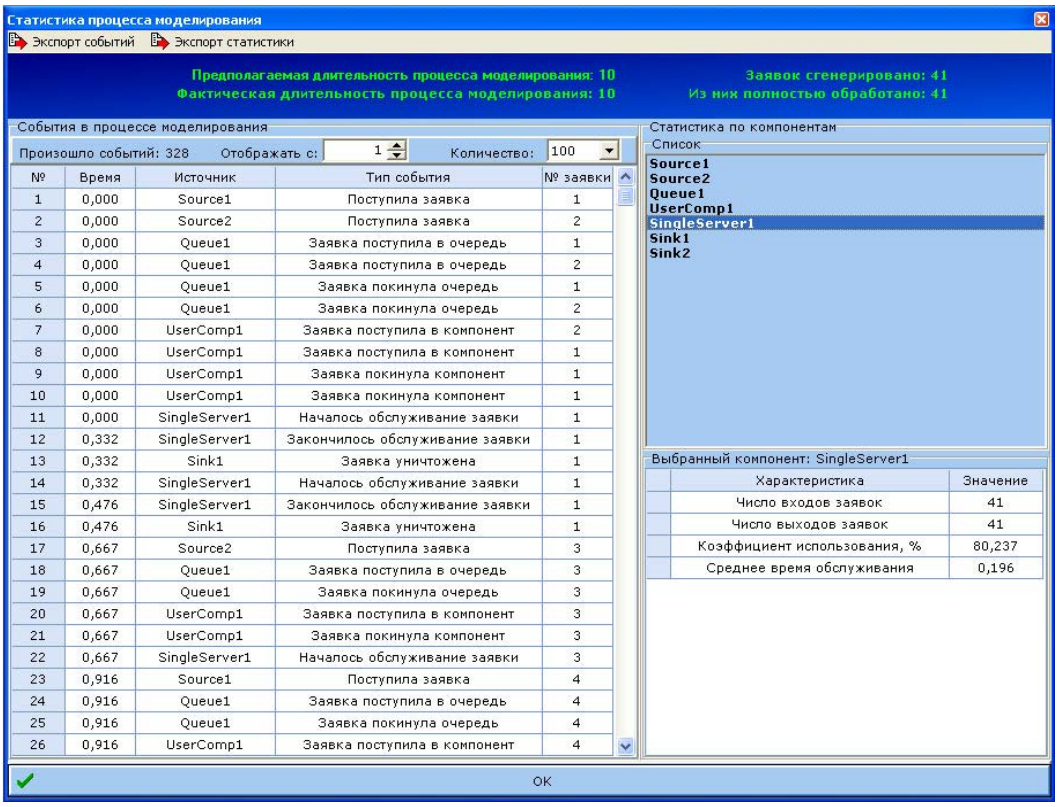

Рис. 5. Статистика в текстовом виде

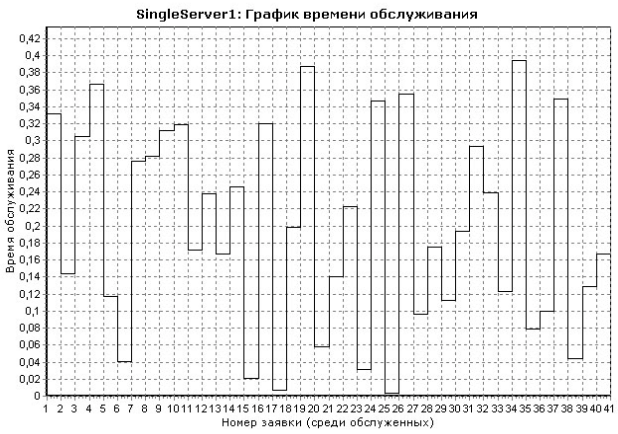

Рис. 6. Статистика в графическом виде

Возможен экспорт произошедших событий, статистики по компонентам в текстовом виде в документы Microsoft Word, Microsoft Excel, текстовый файл, а также экспорт графиков в форматы BMP, JPEG, WMF.

Планирование и проведение экспериментов с моделями, заключающееся в многократном запуске модели с одинаковыми параметрами и получении усредненной статистики, в настоящее время не поддерживается, но будет реализовано в ближайшее время.

Литература

1. **Девятков В. В.** Практическое применение имитационного моделирования в России и странах СНГ: обзор, анализ перспектив//Материалы I Всероссийской конференции ИММОД- 2003 (Санкт-Петербург, 23–24 октября 2003 г.).
2. **Змеев О. А., Приступа А. В.** Разработка объектно-ориентированного программного комплекса имитационного моделирования систем массового обслуживания//Вестник Томского государственного университета. – 2004. – № 284. – С. 174–176.
3. **Лоу А. М., Кельтон В. Д.** Имитационное моделирование. Классика CS. 3-е изд. – СПб.: Питер, 2004. – 848 с.
4. **Рыжиков Ю. И.** Имитационное моделирование. Теория и технологии. – СПб.: Корона принт, 2004. – 384 с.
5. **Шмидт Б.** Искусство моделирования и имитации/Пер. с нем. Ю.А. Ивашкин, В.Л. Конюх – М.: Франтэра, 2003. – 550 с.