СИСТЕМА OBJECT GPSS. ВЛОЖЕНИЕ ЯЗЫКА GPSS В DELPHI.

А. Г. Королев (Северодонецк, Украина)

Введение

Имитационное моделирование, как и моделирование вообще – мощное средство для изучения сложных систем. Для имитационного моделирования используется целый спектр языков, в частности язык GPSS. Лучше всего он описан в известной книге Шрайбера Т. Дж. «Моделирование на GPSS» («красная книга»). Эта книга является фундаментальным самоучителем по языку GPSS и прекрасно описывает как проблематику задач моделирования систем массового обслуживания, так и методы их решения с помощью данного языка.

Язык GPSS существует более 40 лет, и за время своего существования многократно подвергался нападкам как недостаточно гибкий и устаревший. Тем не менее, сам дискретно-событийный подход, заложенный в его основу, был достаточно прочен, чтобы обеспечить его жизнеспособность в течение всех этих десятилетий. Язык GPSS был и остается весьма эффективным при решении множества простых задач, посвященных исследованию систем массового обслуживания. Очень негативно на языке GPSS отразилось то, что в восьмидесятых годах фирма IBM прекратила его поддержку. Поэтому, его дальнейшая история связана с работой таких его энтузиастов, и даже можно сказать подвижников, как Thomas J. Schriber, Springer Cox, Julian Reitman, James O. Henriksen, Peter Lorenz, Ingolf Stahl и целый ряд других.

В целом, ценность такого языка как GPSS для исследования систем массового обслуживания не подлежит сомнению. Однако многолетний опыт преподавания этого языка в университете, показал, что за годы своего развития, язык вобрал в себя и немало негативного, такого, что не выдержало проверку временем, и на сегодня мешает освоению GPSS студентами. Кроме того, этот язык, даже в новой версии, GPSS World, плохо сочетается с теми новыми возможностями, которые предоставляет программисту и пользователю операционная система Windows. Короче говоря, назрела задача попытаться модернизировать язык GPSS, и расширить его возможности по работе с Windows.

Конкретные предложения

Попытка осмыслить эту проблему в целом, привела к пониманию, что начинать все нужно с погружения возможностей языка GPSS в один из хороших языков программирования, обеспечивающих эффективную работу с Windows. Естественно, что в первую очередь рассматривались возможности таких языков, как Delphi, C++ Builder и C#. Очевидно, что реализация основных блоков GPSS не так уж и сложна. Она включает в себя работу со списками заявок и работу с такими объектами, как очереди, устройства, таблицы, и так далее.

Вопрос о том, как реализовать порядок выполнения блоков, управляемый заявками, имеет свое очевидное решение. Нужно, чтобы все блоки модели были оформлены как вызовы процедур и находились в операторе Switch для языка C++, или в операторе Case для языка Delphi. Тогда выбор нужного блока (вызова процедуры) можно выполнять на основе номера следующего блока активной заявки, которая продвигается в данный момент. После выполнения очередного блока, управление вновь передается активной заявке. Естественно, что в качестве блоков возможно использование не любых процедур языка высокого уровня, а только тех, которые обеспечивают корректную ра-

боту со списками заявок, в частности, с номерами текущего и следующего блока активной заявки.

Таким образом, основные блоки, команды и стандартные числовые атрибуты языка GPSS можно представить как процедуры и функции языка C++ или Delphi, причем код модели на GPSS может быть адекватно представлен как последовательность соответствующих вызовов процедур языка программирования.

Рассмотрим, например, простейшую модель обслуживания заявок в одноканальном устройстве, реализованную на языке GPSS World.

```
Generate 100,90
Queue que1
Seize fac1
Depart que1
Advance 95,90
Release fac1
Terminate 1
```

Базовая процедура собственно моделирования на языке Delphi может выглядеть, например следующим образом:

```
{~mtb}procedure ModelTxt;begin with SYS do case NextBlock of
{/Gen} ::Gen_ 1:Gen.Generate(Exponential(120));
  2:Que.Queue;
  3:Fac.Seize;
  4:Que.Depart;
  5:Advance  (Exponential(100));
  6:Fac.Release;
  7:Terminate (1);
{~mte} else modelerror;end;end;
```

В процессе моделирования каждая заявка будет выбирать для выполнения процедуры с номерами 1..7, в зависимости от того, куда она должна идти.

Пока мы не будем обращать внимания на остальные строки этого текста, смысл которых будет разъяснен позднее.

Развитие такой идеи построения моделей привело к тому, что была создана система Object GPSS, в которой текст модели оказывается написанным на языке Delphi (Object Pascal). При этом в предлагаемой системе доступны все основные возможности базового языка. В частности, в такой модели можно легко использовать файлы для ввода и вывода данных, можно просто и естественно отображать ход моделирования в виде графиков или перемещения компонентов. Можно описывать массивы и матрицы любых объектов модели, в частности, многоканальных устройств, списков пользователей, таблиц, числовых функций и многого другого.

В предлагаемой системе можно достаточно просто построить произвольную процедуру управления ходом моделирования, и сбора статистики, в том числе, и управляемую пользователем.

Здесь просто и естественно создаются исполнимые файлы (программы), являющиеся моделями конкретных систем массового обслуживания. Нет никаких проблем в создании новых блоков для построения модели, в том числе и блоков, которые нужны только для конкретной модели. Фактически, возможности разработчика модели более не ограничены набором имеющихся блоков GPSS, а определяются только его потребностями.

Именно такая система для построения моделей в стиле GPSS и названа автором Object GPSS. Некоторое увеличение времени набора кода модели и несколько большее процессорное времени моделирования, в сравнении, например, с GPSS World, с избытком компенсируется расширенными возможностями системы.

В данной системе предусмотрена возможность прекращения моделирования с сохранением текущего состояния модели. Это позволяет позднее продолжить моделирование именно с этого состояния.

В рассматриваемой системе GPSS-модель представляет собой Include-файл, который вставляется в головную программу с помощью указания компилятору вида:

{$INCLUDE model}

Файл Model.pas содержит описание конкретной модели на Delphi в стиле GPSS.

Весь проект ObjectGPSSProject вместе с моделью, компилируется системой Delphi с получением конкретной модели, с которой уже можно проводить эксперименты.

Все файлы проекта, кроме файла Model.pas, не зависят от конкретной модели и вместо их кодов можно использовать DCU–файлы, то есть они могут быть представлены в виде уже откомпилированных частей, которые нуждаются только в сборке в единую программу.

Как следствие вышесказанного, при распространении системы нет нужды поставлять исходные коды наиболее важных частей модели.

Пример полного файла Model.pas для приведенной выше модели, выглядит следующим образом.

```
    {~cb} Const
 // model description auto.
 modelname='E:\work\Object Client\ObjectGPSS\models\tmp.rtf' ;
Gen_=1;
num_bl=7;
  {~ce}
  {~vb} Var
{/Fac} Fac:TFacility;
{/Que} Que:TQueue;
{/Tab} Tab:TTable;
{/Gen} Gen:TGenerate;
  {~ve}
  {~pfe}
  {~ib} procedure Initial;begin
  setstart(1000);
{/Fac} TFacility.Init(Fac);
{/Que} TQueue.Init(Que);
{/Tab} TTable.Init(Tab,0,100,100);
{/Gen} TGenerate.Init(Gen,Gen_,Exponential(120));
  {~ie}end;
  {~mtb}procedure ModelTxt;begin with SYS do case NextBlock of
{/Gen} {::Gen_} 1:Gen.Generate(Exponential(120));
    2:Que.Queue;
    3:Fac.Seize;
    4:Que.Depart;
    5:Advance  (Exponential(100));
    6:Fac.Release;
    7:Terminate (1);
  {~mte} else modelerror;end;end;
  {~mb} procedure Modeling;  begin
  start(GetTg1);
  Tab.Show(1);
  {~me}end;
  {~rb} procedure Report;begin
{/Fac} Fac.Report('Fac');
{/Que} Que.Report('Que');
```

```
{/Tab} Tab.Report('Tab');
{/Gen} Gen.Report('Gen');
  {~re} end;
  {~rab} procedure ResetAllObj;begin
{/Fac} Fac.Reset;
{/Que} Que.Reset;
{/Tab} Tab.Reset;
  {~rae} end;
  {~cab} procedure CloseAllObj;begin
{/Gen} Gen.Freeobj;
{/Tab} Tab.Freeobj;
{/Que} Que.Freeobj;
{/Fac} Fac.Freeobj;
  {~cae} end;
```

Этот файл содержит следующие части:

1. описание констант, переменных и объектов модели;
2. процедуру Initial (инициализация всех переменных и объектов модели);
3. процедуру CloseAllObj (уничтожение всех переменных и объектов модели);
4. процедуру ResetAll (сброс статистики по объектам модели);
5. процедуру ModelTxt (описывается собственно модель системы);
6. процедуру Report (формирование выходной статистики по объектам модели);
7. процедуру Modeling (конкретные команды манипуляции с моделью).

В процедуре Modeling можно управлять моделью и готовить ее к новому моделированию. В частности, можно менять значения X-параметров, функций, многоканальных устройств, таблиц, переменных и так далее.

Несмотря на кажущуюся громоздкость описания модели, практически все ее части, кроме «начинки» процедуры ModelTxt, создаются программой–конвертером, по сути дела, автоматически в процессе вставки объектов модели.

В данном случае были описаны следующие объекты: генератор заявок, одноканальное устройство и очередь.

Созданная при компиляции модель обеспечивает следующие возможности проведения экспериментов:

– запуск модели, то есть вызов процедуры Modeling с указанным значением счетчика завершений TG1;

– временную, точнее досрочную остановку процесса моделирования, с возможностью продолжения текущего моделирования;

– полную очистку модели с возвратом ее в начальное состояние;

– сброс собранной статистики по модели;

– формирование стандартного отчета по моделированию;

– настройку перечня выводимой в стандартный отчет информации;

– просмотр выводимой в ходе моделирования или выведенной по окончании моделирования графической информации;

– завершение моделирования, как с сохранением текущего состояния модели, так и без его сохранения.

Созданная программа (EXE-файл) может быть переименована и перенесена в другую папку, правда вместе с файлом Model.pas. Этот файл нужен для формирования отчета по модели.

Модель, в той форме, в какой она представлена в файле Model.pas, не удобна для разработки и модификации, поэтому ее лучше сохранять и создавать в некоторой промежуточной форме. В этой форме вместо номеров блоков используется комбинация двух символов *: , а метки блоков, которые используются для построения модели,

должны начинаться с пары символов :: . Конвертер выполняет подстановку номеров блоков, определяет численные значения для меток блоков, а также определяет общее число блоков и имя файла с моделью. Эти сведения автоматически попадают в файл Model.pas при конвертировании модели.

Система содержит 12 типов объектов, 94 блока, 109 команд для процедуры Modeling и 200 функции. Естественно, что большая часть этих команд будет использоваться крайне редко, так что реально нужно знать и использовать не более 50–100 команд, блоков и функций. Система полностью перекрывает возможности GPSS World и дает массу дополнительных возможностей, тщательно отобранных и систематизированных.

Выводы

Система Object GPSS легко может быть развита или расширена в любом направлении, которое необходимо автору модели. Так как Delphi относится к языкам класса 4GL, то развитие системы не требует высокой квалификации. Систему можно также использовать для обучения. Тогда ее прозрачность и легкость развития окажется особенно полезной. Хотя фактически, при работе с системой, вы пишете код программы на языке Object Pascal, однако для работы с ней достаточно знать только основы этого, или какого либо иного языка высокого уровня, и знать набор процедур и функций, используемых для моделирования. Наборы личных процедур и функций для моделирования можно записывать как непосредственно в модели, так и в дополнительном файле, который можно включать в систему.

Оценивая опыт эксплуатации Object GPSS, следует отметить, что сами модели для этой системы выглядят более естественно, чем на традиционных версиях GPSS. Логика построения моделей более прозрачна и более соответствует логики обычных программ. При этом, основные усилия разработчика моделей тратятся на саму модель, а не на борьбу с «особенностями» языка GPSS.

Развитые средства визуализации, дополнительные блоки вывода, и возможность приостановить исполнение модели в любой момент и посмотреть полные результаты моделирования, положительно сказываются на отладке моделей.

Разработанная система может быть полезна как для тех, кто начинает осваивать дискретно-событийное моделирование, и хочет, чтобы его модели выглядели профессионально, так и для тех, кто разрабатывает сложные модели «под заказ».

Литература

1. Minuteman Software. 2000. *GPSS World Reference Manual*. Holly Springs NC: Minuteman Software.
2. **Ståhl I**. 2001. GPSS – 40 years of development. In *Proceedings of the 2001 Winter Simulation Conference*, ed B. A. Peters et al. Piscataway, New Jersey: IEEE.
3. Wolverine Software Corporation. 1996. *SLX: An introduction for GPSS/H users.* Alexandria, Virginia: Wolverine Software Corporation.
4. **Lorenz P., Dorwarth P., Ritter K.-C. and Schriber T.J.** 1997. Towards a web based simulation environment. In *Proceedings of the 1997 Winter Simulation Conference*, ed. S. Andradottir, K. J. Healy, D. H. Withers, and B. L. Nelson. Piscataway, NJ: IEEE.
5. **Королев А. Г.** Сравнение команд и блоков GPSS World и Object GPSS. Сайт GPSS.RU, раздел статьи.