УДК 62. 529

Проект OBJECT GPSS и его новые продукты GPSS – Future и Segmental GPSS

Королёв Анатолий Георгиевич, Северодонецк, Украина

В статье приводится сравнение команд и блоков для GPSS –World и систем ряда Object GPSS. Object GPSS – это средство для написания моделей в стиле GPSS непосредственно на языке Delphi (Object Pascal). Каждая модель на Object GPSS представляет собой модуль (Model.pas), содержащий описание всех объектов модели и набор из ряда процедур: Initial, ModelTxt, Report, Simulation, CloseAllObj. Практически все части модели, кроме «начинки» процедуры ModelTxt, создаются программой – конвертером. То есть модель, строится на основе шаблона, примерно как, например, программы в Delphi или в C#. Для создания исполняемой модели следует скомпилировать модуль вместе с остальными стандартными частями проекта. Полученный .EXE – файл является конкретной моделью и с ней можно проводить эксперименты. В этой системе моделирования легко расширять набор команд и блоков.

Система GPSS – Future развивает проект Object GPSS в плане более гибкой работы со списком будущих событий, а система Segmental GPSS позволяет создавать модели из нескольких сегментов (аналога процедур). Сегменты могут быть скомпилированы отдельно и использоваться во многих моделях без раскрытия их реализации.

Введение

Имитационное моделирование, как и моделирование вообще – мощное средство для изучения сложных объектов. Для имитационного моделирования используется целый спектр языков, в частности язык GPSS. Лучше всего он описан в известной книге Шрайбера Т.Дж. «Моделирование на GPSS», которая также известна, как «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 50 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 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, показал, что за годы своего развития, в языке накопилось и немало негативного, такого, что не выдержало проверку временем, и мешает освоению GPSS студентами. Кроме того, этот язык, даже в современной версии, GPSS World, плохо сочетается с теми новыми возможностями, которые предоставляет программисту операционная система Windows. Короче говоря, назрела задача попытаться модернизировать язык GPSS, и расширить его возможности по работе с Windows.

Это, впрочем, уже делалось в таких системах, как Arena Simulation, SLX, Any Logic, eM-Plant, PilGrim и ряде других.

Проект Object GPSS начинался в 2002 году, во многом, случайно. И с тех пор по сей день с разной степенью интенсивностью он продолжает развиваться.

Данный проект, как и все проекты, которые делаются на энтузиазме, имеет как свои достоинства, так и недостатки. Вам решать, чего в нем больше, первого или второго.

Если после чтения статьи вы заинтересуетесь данным проектом, то прошу на сайт objectgpss.ucoz.ru . Там вы найдете различные материалы по проекту, включая учебные пособия и бесплатную пробную версию.

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 качестве кандидатов рассматривались такие языки, как Delphi, Java, C++ 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, тоже имеет свое очевидное решение. Нужно чтобы все блоки модели были оформлены как вызовы процедур и находились в операторе Switch для языков группы 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, а только тех, которые обеспечивают корректную работу 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 выбранного языка, причем код модели на GPSS может быть адекватно представлен как последовательность соответствующих вызовов процедур этого языка программирования.

Модели в системах ряда Object GPSS состоят из следующих частей.

1. Описания констант, переменных и объектов модели, а также специальных процедур и функций, используемых в модели.
2. Процедуры Initial, в которой выполняется инициализация всех переменных и объектов модели.
3. Процедуры ModelTxt, в которой и описывается собственно модель системы.
4. Процедуры Simulation, в которой указаны конкретные команды манипуляции с моделью.
5. Процедуры Report, в которой выполняется вывод дополнительных элементов выходной статистики. Обычно эта процедура пуста.
6. Процедуры CloseAllObj, в которой выполняется уничтожение всех дополнительных динамических объектов модели. Обычно эта процедура пуста.

Для создания моделей в проекте используется программа - конвертер. Вся работа по созданию модели начинается с описания объектов, использующихся для формирования модели. Затем формируется код, определяющий собственно работу модели. Так, например, рассмотрим простейшую модель такой системы массового обслуживания.

Пусть заявки поступают в систему со средним интервалом 100, распределенным по экспоненциальному закону. Они должны обслуживаться одним из 3 устройств. 4-канальным, за время 800+-400, или 6- канальным, за время 1100+-500, или 1- канальным, за время 600+-300. промоделировать обслуживание 10000 заявок. Выполнить табулирование времени обслуживания заявок.

Для такой системы необходимо вставить генератор заявок, 3 устройства соответствующего типа, очередь, и таблицу. А затем написать следующий текст модели.

```
{\Gen} ::Gen_ *:Gen.Generate(Exponential(100));
```

```
  *:Que.Queue;
  *:Transfer ([mm1,mm2,mm3]);
::mm1  *:Stor.Enter;
    *:Que.depart;
    *:Advance (800,400);
    *:Stor.Leave;
    *:Tab.Tabulate (TranAge);
    *:Terminate (1);
::mm2  *:Stor0.Enter;
    *:Que.depart;
    *:Advance (1100,500);
    *:Stor0.Leave;
    *:Tab.Tabulate (TranAge);
    *:Terminate (1);
::mm3  *:Fac.Seize;
    *:Que.depart;
    *:Advance (600,300);
    *:Fac.Release;
    *:Tab.Tabulate (TranAge);
    *:Terminate (1);
```

Аналогичный текст модели на GPSS World выглядит следующим образом:

```
Stor Storage 4
Stor0 Storage 6
tab table m1,0,50,100
   Generate (Exponential(1,0,100))
   Queue Que
   Transfer all,mm1,mm3,6
mm1  Enter Stor
     Depart Que
     Advance 800,400
     Leave stor
     Tabulate Tab
     Terminate 1
mm2  Enter Stor0
     Depart Que
     Advance 1100,500
     Leave Stor0
     Tabulate Tab
     Terminate 1
mm3 Seize Fas;
    Depart Que
    Advance 600,300
    Release Fas
    Tabulate Tab
    Terminate 1
 start 10000
```

Полная модель этой системы в Object GPSS выглядит следующим образом:

```
 {~vb} Var
{\Gen} Gen:TGenerate;
{\Stor} Stor:TStorage;
{\Stor0} Stor0:TStorage;
{\Fac} Fac:TFacility;
{\Que} Que:TQueue;
{\Tab} Tab:TTable;
 {~ve}
 {~pfe}
 {~ib} procedure Initial;begin
 setstart(10000);
{\Gen} Init(Gen,'Gen',Gen_,Exponential(100));
{\Stor} Init(Stor,'Stor',4);
{\Stor0} Init(Stor0,'Stor0',6);
{\Fac} Init(Fac,'Fac');
```

```
{\Que} Init(Que,'Que');
{\Tab} Init(Tab,'Tab',0,50,100);
 {~ie} end;
 {~mtb}procedure ModelTxt;begin  case ActiveBlock of
{\Gen} ::Gen_ *:Gen.Generate(Exponential(100));
  *:Que.Queue;
  *:Transfer ([mm1,mm2,mm3]);
::mm1  *:Stor.Enter;
    *:Que.depart;
    *:Advance (800,400);
    *:Stor.Leave;
    *:Tab.Tabulate (TranAge);
    *:Terminate (1);
::mm2  *:Stor0.Enter;
    *:Que.depart;
    *:Advance (1100,500);
    *:Stor0.Leave;
    *:Tab.Tabulate (TranAge);
    *:Terminate (1);
::mm3  *:Fac.Seize;
    *:Que.depart;
    *:Advance (600,300);
    *:Fac.Release;
    *:Tab.Tabulate (TranAge);
    *:Terminate (1);
 {~mte} end;end;
{~mb} procedure Simulation;  begin
 start(GetTg1);
 Show(Tab,0);
 {~me}end;
{~rb} procedure Report;begin
 {~re} end;
 {~cab} procedure CloseAllObj;begin
 {~cae} end;
```

Несмотря на кажущуюся громоздкость такой модели, на самом деле она довольно проста, так как весь текст этой модели, кроме содержимого процедуры ModelTxt, генерируется автоматически при вставке объектов.

Для создания модели ее текст должен пройти конвертирование в текст на языке Object Pascal. Конвертирование состоит в том, что в паре символов “*:” символы “*” заменяются на номера блоков, а метки блоков преобразуются в константы целого типа, значения которых равны номеру блока с меткой. В итоге получается файл Model.pas, который компилируется совместно с другими модулями. В результате компиляции получается исполняемая модель в виде .Exe - файла, с которой уже можно проводить эксперименты.

Все файлы проекта, кроме файла Model.pas, в принципе, 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 этого, при распространении системы нет нужды поставлять исходные коды наиболее важных частей модели.

Созданное конвертером приложение, обеспечивает следующие возможности проведения опытов с моделью.

Запуск модели, то есть вызов процедуры Simulation с указанным значением счетчика завершений Term.

Временную, точнее досрочную остановку процесса моделирования, с возможностью продолжения текущего моделирования.

Полную очистку модели с возвратом ее в начальное состояние.

Сброс собранной по модели статистики.

Формирование полного или частичного отчета по моделированию.

Настройку перечня выводимой в стандартный отчет информации.

Просмотр выводимой в ходе моделирования, или выведенной по окончании моделирования графической информации.

Завершение моделирования.

Созданная программа может быть переименована и/или перенесена в другую папку, правда, вместе с некоторыми дополнительными файлами, которые понадобятся при исполнения модели.

Для представления списков заявок и работы с ними удобно использовать таблицы баз данных. Таблицы хорошо подходят и для представления параметров заявок и параметров системы, а также для представления групп заявок и списков прерывания.

Развитие такой идеи построения моделей систем, привело к тому, что для языка Delphi были созданы процедуры, обеспечивающие возможность включения моделей, написанных в стиле GPSS в любую программу на языке Delphi. Поэтому в модели, написанной в стиле GPSS, доступны все возможности самого языка реализации. 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 величин прямоугольников. Можно описывать массивы и матрицы любых объектов модели, в частности, многоканальных устройств, списков пользователей, таблиц, числовых функций, и многое другое.

Достаточно просто здесь можно 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 такой ряд систем для построения моделей в стиле GPSS и назван автором объектным GPSS. Некоторое увеличение времени набора кода модели, например, в сравнении с GPSS World, с избытком компенсируется расширенными возможностями системы.

То, что для реализации системы первоначально был выбран язык Delphi, а не C++, или, например, C# - не является принципиальным. Просто на этом языке код модели выглядит несколько более привычным, по сравнению с аналогичным кодом на языках группы C.

В отличие от других версий GPSS, здесь предусмотрена типизация P и X параметров. Они могут быть: Integer, Double, Boolean, String.

Так как Object GPSS создает модели на основе классов объектов, то при работе с объектами вначале должно указываться имя объекта, затем точка, а затем, имя процедуры или функции из класса. Например, fac1.Seize – означает «занять устройство с именем fac1», Tab.Tabulate(TranAge) выполнить табуляцию времени жизни заявки в таблице Tab и так далее.

Система Object GPSS содержит следующие классы объектов и набор блоков общего назначения.

Набор блоков общего назначения содержит основные функции и блоки системы GPSS. Некоторые функции и блоки имеют другие имена, по нашему мнению, более подходящие, чем, например, в GPSS – World. Так вместо СЧА M1 – текущее время жизни заявки, TG1 – текущее значение счетчика завершений используются функции TranAge и Term. В принципе, процедуры блоки и функции этого набора повторяют то, что есть в GPSS World. К существенным изменениям можно отнести замену параметров заявки и системы (СЧА P и X) на функцииRP(Num), IP(Num), BP(Num), SP(Num), которые выдают

значения вещественных, целых, логических и строковых параметров заявок, по умолчанию – активных заявок. Для работы с параметрами системы, (СЧА X) имейте в виду, что любой объект имеет свой набор параметров системы. Так как параметры системы и заявки аналогичны, то для работы с параметрами системы можно использовать те же блоки, процедуры и функции, что и для заявок. В этом случае последний аргумент должен быть NameObject.N. Хотя параметры заявок и системы задаются номерами, вместо номеров всегда можно использовать целые именованные константы, тогда текст модели становится более наглядным.

В системе в качестве блоков (содержимого процедуры ModelTxt) можно использовать любые операторы базового языка, которые не отправляют заявки по новому пути, в частности, это могут быть операторы присваивания или ввода-вывода.

Существенные и важные отличия от GPSS состоят в следующем.

Параметры неактивной заявки доступны для чтения и записи, если известен номер заявки.

Моделирование можно прервать программным путем с помощью блока Stop.

Возможна установка нового значения счетчика завершений в ходе моделирования, блок SetTerm.

Роль блоков GATE, TEST и TRANSFER взял на себя расширенный блок Test, который обеспечивает разветвление на произвольное число направлений, в зависимости от выполнения условий, а также задержку заявки, если все условия неверны.

Блок Transfer обеспечивает переход на подпрограмму, вероятностное или безусловное перенаправление заявок, а также выбор перехода на ту метку из списка, которая сопоставлена блоку, готовому принять заявку.

Появился блок розыска указанной заявки FindTran, который определяет для нее текущий и следующий блок, а также список, в котором она находится.

Модель может использовать и блоки, которые предназначены для конкретной головной программы. Они, строятся на основе имеющихся блоков. К ним относятся блоки ToPoint, ToString, ToValue и CurrentBlocks, которые выводят очередную точку, очередную строку, очередное значение и текущую информацию о блоках.

Набор функций очень велик, и включает в себя, в частности, аналоги функций типов C и D в GPSS World. Функции можно также выводить в виде графиков.

Класс TGenerate описывает генераторы заявок, то есть фактически блоки Generate. Каждый блок Generate должен быть описан и проинициализирован. Дополнительно, у блока Generate может указываться условие, которое определяет, будет ли выпущена из него заявка. Приход новой заявки планируется, только если предыдущая заявка вышла из блока.

Класс TTable описывает таблицы, предназначенные для сбора статистики. Он пополнен процедурой, которая выводят таблицы в виде гистограмм.

Класс TQueue описывает очереди. Его процедуры и функции, по сути, не отличаются от аналогичных блоков и СЧА GPSS– World.

Класс TStorage описывает многоканальные устройства. Он пополнен блоком SetStorage, который устанавливает новую предельную емкость устройства, и блоком SetGoto, который задает перенаправление заявок с входа устройства на другой блок, или обеспечивает запрет на вход заявок в устройство.

Класс TFacility описывает одноканальные устройства. Для таких устройств также имеется блок SetGoto, и несколько иначе построена система блоков для реализации прерываний, и для реализации доступности и недоступности устройства. А именно: блоки PREEMPT, RETURN, FAVAIL и FUNAVAIL заменены на блоки Extract, ExtractPreempt, Preempt, Preempt0, Return, Return0. Они удаляют обслуживаемые и прерванные на устройстве заявки, прерывает обслуживание заявки с занятием и без занятия устройства, а также возвращает из прерывания заявку после обслуживания текущей заявки или возвращает ее принудительно, после периода недоступности устройства.

Классы TRGroup, TIGroup, TSGroup описывают группы для вещественных чисел, целых чисел и для строк. Их процедуры и функции, по сути, мало отличаются от аналогичных блоков и СЧА GPSS World.

Класс TTGroup описывает группы заявок. Его набор функций и процедур заметно расширен, за счет типизации данных, то есть в данном случае параметров заявки, а также за счет обеспечения доступа к параметрам заявок из списка группы при отборе нужных заявок. Отбор нужных заявок и изменение значений параметров заявки, ведется с помощью отдельно описанных функцию выбора и функций вычисления значений. Эта особенность повышает возможности блоков работы с группами заявок. В этом классе блоки SCAN и ALTER заменены блоками RScan, IScan, BScan, SScan и RAlter, IAlter, BAlter, SAlter, которые отыскивают и изменяют значения вещественных, целых, логических и строковых параметров заявок соответственно.

Класс TUser описывает списки пользователя. Его набор функций заметно расширен за счет доступа к параметрам заявок из списка пользователя при выборе удаляемых заявок. Отбор нужных заявок ведется с помощью отдельно описанных функций выбора. Эта особенность повышает возможности блоков работы со списками пользователя. Заявки в списке упорядочены по 2 параметрам: целому и вещественному. Просмотр заявок при отборе возможен сверху вниз или снизу вверх. Просмотр может ограничиваться по количеству заявок. Просмотр возможен полный, потенциально по всему списку заявок, или в пределах одного значения целого параметра.

Имеются также функции анализа значений заявок из списка пользователя – это функции Sum Avg Min Max Count. В функциях можно указывать все те же элементы управления выбором, что и при выборе удаляемых заявок, а кроме того, в них используются функции значений для формирования результата, по которому собственно и проводится суммирование, усреднение, и так далее.

Важно, что в системе не предусмотрено использование списков задержки для очередей к устройствам, а вместо этого, если нужно, следует использовать списки пользователя. Для этого, в частности, существенно расширены возможности блока Unlink в смысле отбора извлекаемых заявок, и в смысле перенаправления их на желаемые блоки. Это, на мой взгляд, заставит более ответственно относиться к оптимизации времени выполнения модели, а не полагаться на встроенные возможности, которые в иных случаях могут стать проблемой на пути к созданию полноценной модели.

Имеется обширный набор функций, которые, в основном, совпадает с теми, которые есть в GPSS World. Однако к ним добавлены функции, которые выбирают значения по логическим условиям, или по номеру. Это функции RByBool, IByBool, S BByBool, RByNum, IByNum BByNum для вещественных, целых и строковых значений соответственно.

Значения параметров для модели можно получать из диалогового окна или из компонента AddMemo главного окна. Их получение обеспечивают функции RInput, IInput, BInput, SInput, InputItem и RGet, IGet, BGet, SGet для вещественных, целых и строковых значений соответственно.

Имеется также ряд других блоков, процедур и функций для организации моделей и процесса моделирования.

Если возникает необходимость создания новых блоков, то это возможно сделать следующим образом.

Можно просто вместо блока в процедуре ModelTxt, поместить составной оператор, обрамленный ключевыми словами Begin - End. Составной оператор может содержать блоки Object GPSS, и должен быть устроен таким образом, чтобы при любых ситуациях, в нем был исполнен, ровно один блок, из числа описанных в системе, или ни одного блока.

Можно также оформить этот составной оператор как процедуру, и тогда ее можно использовать как новый блок.

Программа - конвертер подготовки моделей позволяет легко вставлять и удалять описания объектов модели и требуемые вызовы процедур. Она позволяет выполнить открытие, редактирование и сохранение модели в промежуточной форме, а также выполнить ее конвертирование для включения кода модели в программу и вызова модели на исполнение. Имена объектов можно оставить такими, какие они предлагаются по умолчанию, а можно ввести их вручную. Типы объектов выбираются из списка.

Конвертер также управляет созданием модели с помощью псевдокомментариев.

Псевдокомментарий сборки модели из фрагментов.

{#Include `FileName` `Start`->`End` `OldName`~`NewName` }

Этот псевдокомментарий позволяет вставить на его место фрагмент файла FileName, начиная от фрагмента с текстом Start до фрагмента с текстом End включительно. При этом будет проведена замена подстрок OldName на подстроки NewName. Последняя часть псевдокомментария может повторяться. Вторая его часть - не обязательна.

Псевдокомментарий выполняется исключительно по подпункту Include из пункта Edit.

Примечание: замена подстрок не выполняется в самих псевдокомментариях {#Include }. Так что вставки фрагментов файла могут быть вложенными.

Псевдокомментарий замены имен в модели.

{#Replace FileName}

Этот псевдокомментарий позволяет выполнить в модели замену имен (не подстрок), взяв данные из файла FileName.

В файле должны чередоваться старые и новые имена. Каждое имя - отдельной строке. Поэтому в файле должно быть чётное число строк. Если вместо имени файла стоит знак ? , то тогда файл надо будет выбрать из диалога открытия файла.

Псевдокомментарий выполняется исключительно по подпункту Replace Names из пукта Edit.

Псевдокомментарий задания имени файла с моделью.

{#Model FileName.pas}

FileName.pas - имя этого файла. Если псевдокомментарий такого вида отсутствует в модели, то считается, что он имеет вид: {#Model Model.pas}

Псевдокомментарий выполняется каждый раз по пункту Run.

Псевдокомментарии для моделей из нескольких сегментов.

Если вы строите модель из нескольких сегментов, то вам придётся использовать следующие псевдокомментарии.

{#Add FileName.txt}

{#NotDelete }

Первый из них должен быть в главном сегменте, и FileName должен быть именем файла с одним из дополнительных сегментов. Всего таких псевдокомментариев должно быть столько же, сколько дополнительных сегментов.

В каждом из дополнительных сегментов должен быть псевдокомментарий {#NotDelete } чтобы обеспечить сохранность текста для каждого из дополнительных сегментов.

Работа по созданию конкретной модели требует выполнения следующих шагов.

1. Запустить приложение converter.exe

2. Открыть в нем имеющуюся модель, или создать новую. Если создана новая модель, то ее следует сохранить на диске.

3. Выполнить пункт «Run» в приложении converter.exe. Если в файле Model.pas имеются ошибки, то нужно их исправить и вновь выполнить шаг 3. Если ошибок нет, то запускается приложение с конкретной моделью и с ним можно проводить опыты..

Для создания очередной модели, (её EXE – файла) достаточно повторить пункты 2,3.

Возможно сохранение в файле, отчета о результатах моделирования или графиков и гистограмм, сформированных моделью.

В заключение, следует отметить, что эксперименты с системой Object GPSS подтвердили реалистичность всех возложенных на нее ожиданий. Она устойчиво работает и создает программы для конкретных моделей систем массового обслуживания на языке Delphi, в том числе и довольно сложных. В системе легко обеспечивается развитие за счет создания новых блоков и функций. Она может взаимодействовать с компонентами головной программы, как в ходе моделирования, так и на стадии подготовки модели к моделированию. Система поддерживает работу с текстовыми файлами.

Выводы

Указанный подход к построению моделей можно использовать и в других языках программирования, например, в Borland C++ или в C#. Преимущества рассматриваемой системы в том, что вся работа модели происходит совершенно прозрачно, и модель может использовать все возможности базового языка программирования, в данном случае – Delphi (Object Pascal). При моделировании выполняется только то, что явно указал разработчик модели. Система легко может быть развита или расширена в любом направлении, которое необходимо автору модели. Так как Delphi относится к языкам класса 4GL, то использование и развитие системы не требует высокой квалификации. Систему можно использовать и для обучения. Тогда ее прозрачность и легкость развития окажется особенно полезной. Хотя при работе с системой, вы пишете код программы на языке высокого уровня (Object Pascal), однако знать этот язык вам не обязательно. И уж тем более вам не нужно знать, как следует работать с оболочкой Delphi. У вас её просто нет. Для работы с Object GPSS нужно знать только основы любого языка программирования, 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. Все ваши процедуры и функции будут «невидимы » для модуля GPSS_SYS, который реализует базовые возможности GPSS.

Оценивая опыт эксплуатации Object GPSS, следует отметить, что сами модели на нем, выглядят более естественно, чем у традиционных версий 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, благотворно сказываются на отладке моделей, в то время как многие модели на GPSS, даже попадая в разряд «образцовых», на самом деле содержат ошибки, иногда довольно грубые.

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».

Приложение: GPSS - Future:

Всем, кто активно использует GPSS, известно, что концепция будущего в этой системе весьма ущербна. То есть, если вы запланировали задержку заявки на некоторое время в блоке Advance, то с этой заявкой уже практически ничего сделать нельзя. Нужно ждать, пока она вновь появится в списке текущих событий. В определённом смысле, это

концепция плановой экономики, когда всё, что произойдет в будущем – планируется заранее и не подлежит никакой ревизии. В новой системе – GPSS – Future список будущих событий рассматривается как наследник списка пользователя. А поэтому всегда можно извлечь из него заявки, удовлетворяющие нужным вам условиям, а также, если необходимо, продвинув текущее время в системе к новому значению. Здесь вы полностью контролируете заявки из списка будущих событий, пока они находятся в этом списке.

В новой системе можно намного легче моделировать задачи с управляемым будущим, которое будет в полной вашей власти, до самого момента наступления этого будущего, то есть в этой системе будущее более соответствует тому, которое реально есть в нашем мире.

Например, вы просто и естественно сможете промоделировать ситуацию в банке, когда вы захотите предложить избранным кредиторам новые условия депозитного договора до истечения его срока.

Когда больше не удалось продвинуть ни одну заявку из списка текущих событий, то в любой версии GPSS должна вызываться процедура продвижения к новому моменту модельного времени. В GPSS – Future, как правило, вначале из списка будущих событий извлекаются заявки, чьё время ожидания уже истекло (так как, возможно, вы решили увеличить текущее время в процессе извлечения заявок из списка будущих событий), а, если их нет, то извлекаются заявки с минимальным временем завершения ожидания. В последнем случае, продвигается вперед текущее время моделирования. Эта процедура используется по умолчанию. Однако, если вы этого желаете, вы можете сами переписать процедуру продвижения к новому моменту модельного времени по своему вкусу. Набор практически всех возможных процедур продвижения к новому моменту модельного времени заранее подготовлен и имеется в системе.

В GPSS – Future для генерации заявок используются процедура Future.NewWaitProc (инициализация) и блок Future.NewWait (генерация). Задержку выполняет блок Future.Wait. Здесь Future – это список будущих событий, который предлагается системой по умолчанию прямо в шаблоне.

Приложение: Segmental GPSS.

Известно, что в GPSS довольно сложно использовать технику построения моделей по частям, или технику работы с процедурами, которая доказала свою эффективность в других сферах создания программного обеспечения.

Отсутствие в GPSS полноценных средств отображения модели по частям ярко подтверждается тем фактом, что большие модели на GPSS не пишутся руками, а генерируются специальными программами – оболочками.

В GPSS нельзя разрабатывать части модели «впрок», с тем, чтобы применять их в будущих моделях по мере надобности без адаптации.

В системе Segmental GPSS конечная модель может строиться из отдельно описанных и даже скомпилированных сегментов модели. Один сегмент является главным и инициирует, готовность к работе других сегментов. Все сегменты модели имеют потенциально одинаковую структуру и возможности. Единственным исключением является план моделирования, который должен располагаться только в главном сегменте.

Для перехода заявки из сегмента в сегмент используется блок

Jump(NModelTxt, NumBl, NumIP, NumModelIP)

Здесь NModelTxt – номер сегмента модели, NumBl - точка входа, NumIP - номер P параметра для запоминания места возврата, NumModelIP - номер P параметра для запоминания номера сегмента модели для возврата. Последние два параметра не нужны, если команда просто выполняет возврат из сегмента в точку вызова (аналог Return).

Как правило, значения, необходимые для корректной работы сегмента с перешедшей в него заявкой передаются как P – параметры этой заявки. Тут могут быть

заданы как числовые значения, так и ссылки на нужные объекты: устройства, очереди, таблицы, процедуры, функции, и так далее.

При построении модели часть сегментов можно скомпилировать заранее, тогда автор модели может не знать, как именно реализован тот или иной вид специфического обслуживания заявок. Это позволяет разрабатывать и распространять сегменты модели как библиотеки, расширяющие и облегчающие разработку новых моделей. Пользователь системы может и сам создавать новые сегменты, которые позволят быстрее и эффективнее разрабатывать и модернизировать модели. Это важно, когда приходится часто дорабатывать модели в связи с изменениями в предметной области.

Данная система обеспечивает сбор статистики по блокам для всех сегментов.

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., H. Dorwarth, K.-C. Ritter, and T. J. Schriber. 1997. Towards a web based simulation environment. In Proceedings of the 1997 Winter Simulation Conference, ed. S.
5. Andradottir, K. J. Healy, D. H. Withers, and B. L. Nelson. Piscataway, NJ: IEEE.

Приложение 1. Конвертер.

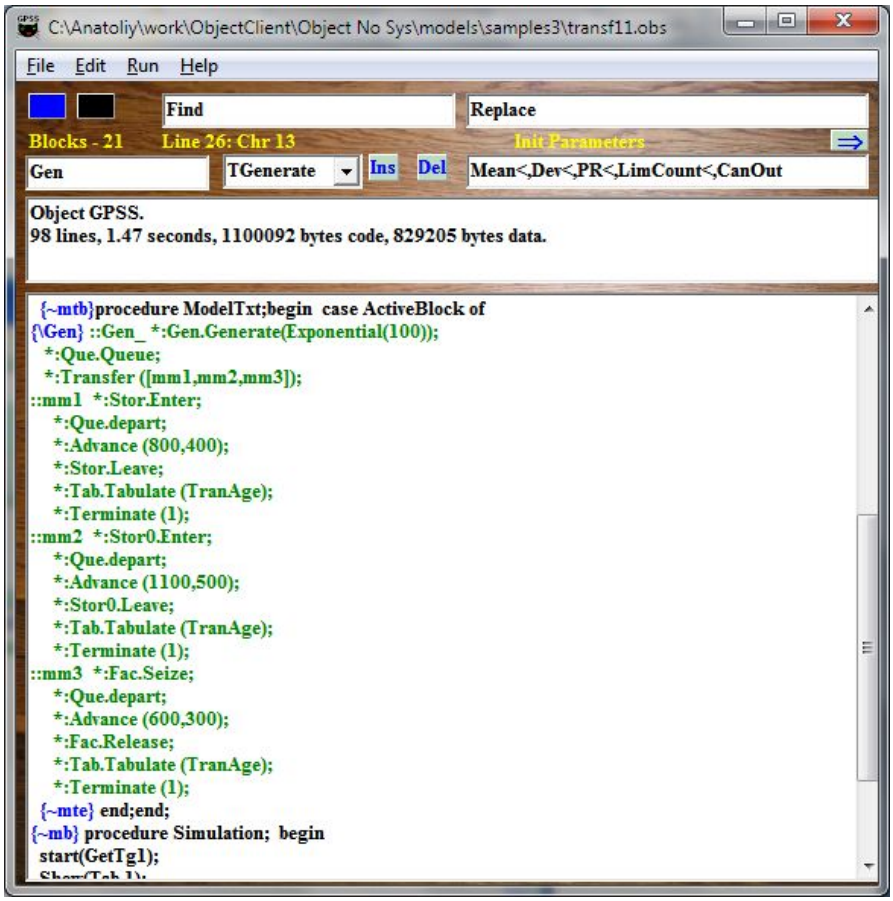

Рисунок 1. Внешний вид программы - конвертера.

Приложение 2. Пример.

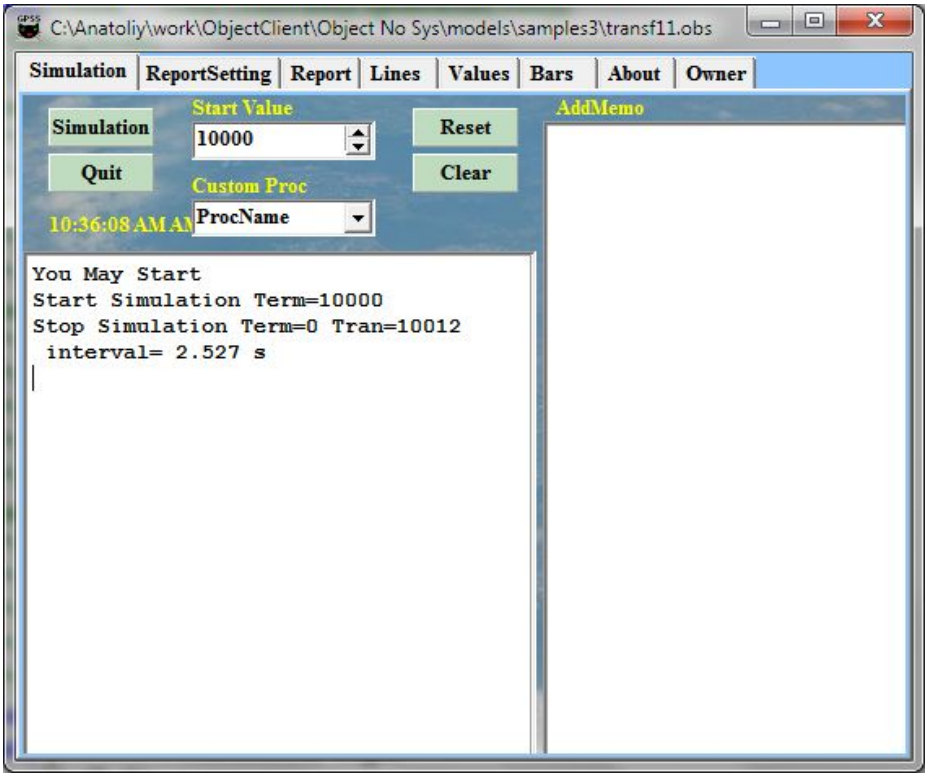

Рисунок 2. Головная форма модели

Отчет по модели.

```
Персонифицированная версия системы 2007.
По вопросам приобретения - обращайтесь по адресу:objectgpss@yandex.ru

Date Time Stamp= 6/28/2013 10:46:06 AM   Ctrl Click |Name - HyperLink `R
             REPORT
C:\Anatoliy\work\ObjectClient\Object No Sys\models\samples3\transf11.obs
StartTime          0.00000 EndTime    1002194.49889
 |Model |Blocks |Values |CurrentList |FutureList
 |UserList |SynchronizeList |InterruptList |PList
   Num      NameObject
     2           |Gen
     3          |Stor
     4         |Stor0
     5           |Fac
     6           |Que
     7           |Tab

 |R `Model
 {\Gen} ::Gen_ 1:Gen.Generate(Exponential(100));
   2:Que.Queue;
   3:Transfer ([mm1,mm2,mm3]);
::mm1  4:Stor.Enter;
     5:Que.depart;
     6:Advance (800,400);
     7:Stor.Leave;
     8:Tab.Tabulate (TranAge);
     9:Terminate (1);
::mm2  10:Stor0.Enter;
     11:Que.depart;
     12:Advance (1100,500);
     13:Stor0.Leave;
     14:Tab.Tabulate (TranAge);
     15:Terminate (1);
::mm3  16:Fac.Seize;
     17:Que.depart;
     18:Advance (600,300);
     19:Fac.Release;
     20:Tab.Tabulate (TranAge);
     21:Terminate (1);

 |R `Blocks    BLOCK Report
   Location     Total   Current
        1     10021         0
        2     10021        11
        3     10010         0
        4      4548         0
        5      4548         0
        6      4548         4
        7      4544         0
        8      4544         0
```

```
         9      4544         0
        10      4473         0
        11      4473         0
        12      4473         6
        13      4467         0
        14      4467         0
        15      4467         0
        16       989         0
        17       989         0
        18       989         0
        19       989         0
        20       989         0
        21       989         0
|R `Values
|R `CurrentList Report Current Transactions List  Count= 11
     Tran     Assem     Prior   Current      Next       TimeStamp       TimeEnter Inter          GenName
    10011     10011         0         2         3   1001725.87492   1001725.87492     0              Gen
    10012     10012         0         2         3   1001738.37779   1001738.37779     0              Gen
    10013     10013         0         2         3   1001747.68761   1001747.68761     0              Gen
    10014     10014         0         2         3   1001780.63682   1001780.63682     0              Gen
    10015     10015         0         2         3   1001838.72880   1001838.72880     0              Gen
    10016     10016         0         2         3   1001889.34477   1001889.34477     0              Gen
    10017     10017         0         2         3   1001945.17053   1001945.17053     0              Gen
    10018     10018         0         2         3   1001979.67139   1001979.67139     0              Gen
    10019     10019         0         2         3   1002011.10167   1002011.10167     0              Gen
    10020     10020         0         2         3   1002025.88733   1002025.88733     0              Gen
    10021     10021         0         2         3   1002188.24042   1002188.24042     0              Gen
|R `FutureList Report Future Transactions List Count= 11
     Tran     Assem     Prior   Current      Next       TimeStamp         EndTime Inter          GenName
    10005     10005         0        12        13   1000770.01604   1002258.55383     0              Gen
    10022     10022         0         0         1   1002318.50724   1002318.50724     0              Gen
    10006     10006         0         6         7   1000816.29118   1002328.75710     0              Gen
     9999      9999         0        12        13   1000575.78943   1002364.95874     0              Gen
    10009     10009         0         6         7   1001356.15973   1002373.02176     0              Gen
    10002     10002         0        12        13   1000672.63797   1002520.59409     0              Gen
    10008     10008         0         6         7   1001327.27760   1002580.82714     0              Gen
    10003     10003         0        12        13   1000690.52161   1002671.81363     0              Gen
    10010     10010         0         6         7   1001477.22956   1003023.68923     0              Gen
    10007     10007         0        12        13   1000816.47717   1003067.52051     0              Gen
    10004     10004         0        12        13   1000749.09383   1003115.99657     0              Gen
|R     Storage   Entries   Current         Average        AverTime         Utility       Max       Min
        `Stor      4548         4         3.61778       797.21094         0.90444         4         0
     Tran    ACapac  Capacity     Active Time    NextGo
    10010     4.000         4   1002194.49889         0
|R     Storage   Entries   Current         Average        AverTime         Utility       Max       Min
       `Stor0      4473         6         4.88472      1094.44267         0.81412         6         0
     Tran    ACapac  Capacity     Active Time    NextGo
    10007     6.000         6   1002194.49889         0
|R    Facility   Entries   Current         Utility        AverTime      Tran   IntTran     Active Time    NextGo
         `Fac       989         0         0.58902       596.87417         0         0   1002194.49889         0
|R       Queue   Entries   Current       AverQueue        AverTime   AverTime(-Ze)       Max       Min      Zero      Active Time
NextGo
         `Que     10021        11         1.14653       114.66428       266.04556        13         0      5702   1002194.49889
0
|R  Table `Tab  Entries     10000.00000 Mean      1024.45545 StdDev       360.19602
          Range       Frequency        Integral
      300.00000         0.00000         0.00000
      350.00000        35.00000         0.00350
      400.00000        36.00000         0.00710
      450.00000       251.00000         0.03220
      500.00000       253.00000         0.05750
      550.00000       269.00000         0.08440
      600.00000       298.00000         0.11420
      650.00000       431.00000         0.15730
      700.00000       450.00000         0.20230
      750.00000       461.00000         0.24840
      800.00000       514.00000         0.29980
      850.00000       509.00000         0.35070
      900.00000       532.00000         0.40390
      950.00000       531.00000         0.45700
     1000.00000       521.00000         0.50910
     1050.00000       466.00000         0.55570
     1100.00000       498.00000         0.60550
     1150.00000       560.00000         0.66150
     1200.00000       492.00000         0.71070
     1250.00000       338.00000         0.74450
     1300.00000       318.00000         0.77630
     1350.00000       293.00000         0.80560
     1400.00000       290.00000         0.83460
     1450.00000       299.00000         0.86450
     1500.00000       241.00000         0.88860
     1550.00000       263.00000         0.91490
     1600.00000       226.00000         0.93750
     1650.00000       125.00000         0.95000
     1700.00000       109.00000         0.96090
     1750.00000        82.00000         0.96910
     1800.00000        68.00000         0.97590
     1850.00000        52.00000         0.98110
     1900.00000        47.00000         0.98580
     1950.00000        39.00000         0.98970
     2000.00000        26.00000         0.99230
     2050.00000        21.00000         0.99440
     2100.00000        10.00000         0.99540
     2150.00000        17.00000         0.99710
     2200.00000         3.00000         0.99740
     2250.00000        10.00000         0.99840
     2300.00000         5.00000         0.99890
     2350.00000         8.00000         0.99970
     2400.00000         3.00000         1.00000
     2450.00000         0.00000         1.00000
```

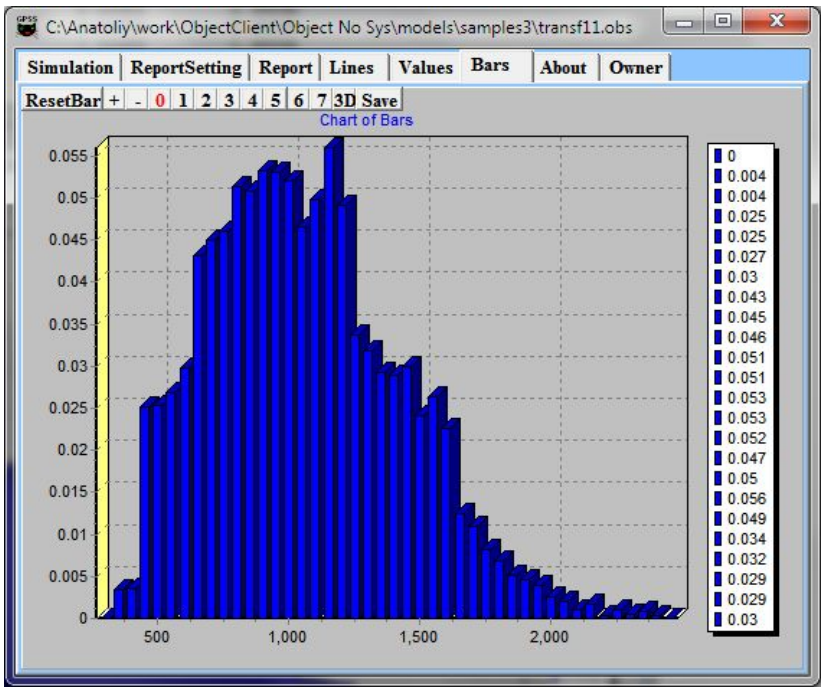

Рисунок 3. Гистограмма таблицы