Известия Гомельского государственного университета
имени Ф. Скорины, №4(37), 2006

УДК 681.3

Методика создания информационного модуля при построении имитационных моделей технологических процессов производства услуг

В. В. СТАРЧЕНКО

Введение

Программы имитационных моделей (ИМ) технологических процессов производства услуг (ТППУ), создаваемые при помощи программно-технологического инструментария (ПТИ) автоматизации имитационного моделирования (АИМ) ТППУ состоят из трех независимо компилируемых модулей: системного, информационного и функционального [1]. Системный модуль является единым для всех ИМ и реализован в виде заголовочного и программного файла. Информационный модуль также разделяется на заголовочный и программный файлы. В узком смысле программа ИМ ТППУ представляется информационным модулем. Для реализации функционального модуля достаточно одного программного файла. При создании ИМ ТППУ перед разработчиком ИМ неизбежно встаёт задача правильного задания информационного модуля программы модели. Это может быть как модификация стандартного информационного модуля ПТИ АИМ ТППУ, так и его расширение за счёт создания новых дополнительных секций.

Структура информационного модуля

Информационный модуль создается отдельно для каждой ИМ. Правила определения информационного модуля вытекают из способа организации программы модели. Они диктуются системным модулем и принципами ООП языка программирования C++. Подразумевается, что разработчик ИМ ТППУ обязан придерживаться этих правил.

Информационный модуль ИМ ТППУ состоит из двух секций. Содержимое каждой секции приведено в табл. 1.

Таблица 1 – Секционная структура информационного модуля

Название файла	*Содержимое файла*
Заголовочный файл информационного модуля (model.h)	#include "micic4.h" //заголовочный файл системного модуля <перечисление компонентов модели> <определение массива компонентов> <перечисление статистик> <определение массива статистик> <описание прототипов активностей и вспомогательных функций> <описание классов источников и ресурсов сервисов> <описание классов потребителей и операндов сервисов> <описание классов–наследников для откликов, определение их виртуальных методов и датчиков псевдослучайных чисел> <описание классов экспериментов>
Программный файл информационного модуля (model.cpp)	#define MODEL #include "model.h" //заголовочный файл информационного модуля <определение вспомогательных переменных> <доопределение при необходимости конструкторов классов-наследников> <определение методов источников и ресурсов сервисов> <определение виртуальных методов классов экспериментов> <определение активностей> <реализация вспомогательных функций>

Правила определения секций заголовочного файла

Рассмотрим состав секций заголовочного файла информационного модуля и приведём основные правила определения таких секций.

- <перечисление компонентов модели>

Перечисление компонентов осуществляется в той же последовательности, что и при определении массива компонентов. Традиционно для перечисления компонентов модели используются заглавные буквы латинского алфавита. Например, `enum{VISGEN, QUEUE, …}`.

- <определение массива компонентов>

Последовательность создания объектов массива компонентов должна совпадать с последовательностью перечисления компонентов модели. Каждому компоненту даётся уникальное имя, указывается его тип, уровень вложенности в иерархической структуре модели и другая вспомогательная информация. Например,

```
Component Components[]= {
   Component("Источник сервиса",GEN),
   Component("Очередь",DVC,TOP,1),
   …
};
```

- <перечисление статистик>

Перечисление статистик осуществляется в той же последовательности, что и при определении массива статистик. Традиционно для перечисления статистик модели используются заглавные буквы латинского алфавита.

- <определение массива статистик>

Последовательность создания объектов массива статистик должна совпадать с последовательностью перечисления статистик модели. Каждой статистике даётся собственное имя. Для условных откликов в имени статистики после точки с запятой можно указать строковое название принципа деления откликов. Это название будет использовано в методах документирования результатов ИЭ для автоматического отображения данных по откликам в разрезе условия. Например,

```
Stat Stats[]={
   ...,
   Stat("Количество обслуженных заявок;по типу"),
   ...
};
```

- <описание прототипов активностей>, <описание классов источников и ресурсов сервисов>, <описание классов потребителей и операндов сервисов>

Происходит полностью в соответствии с правилами наследования ООП и нуждами разработчиков ИМ. Часто это добавление в базовые классы вспомогательных информационных полей и методов работы с ними.

- <описание классов–наследников для откликов, определение их виртуальных методов и датчиков псевдослучайных чисел>

Обычно это описание условных и результирующих откликов. В случае условных откликов важное место занимает описание функции `Condition`. Она должна возвращать ненулевое значение, если выполняется условие расчета отклика. Например,

```
int Condition(void){
   Request *p=(Request *)ActiveTns;
   if(Type==-1) return 1;
   return p->type==Type; //совпадение по заданным признакам из требуемых классов
};
```

- <описание классов экспериментов>

Наследование от базового класса `Experiment` позволяет организовывать различные виды имитационных экспериментов (ИЭ). Ключевым моментом при этом является создание откликов в конструкторе определяемого класса, описание возможных факторов эксперимента, а также возможное определение виртуальных функций-членов класса.

В общем случае строка создания отклика в конструкторе класса имеет вид:

```
Responses[AddResponse(имя_из_перечисления_статистик)]=new имя_класса_отклика();
```

Например, `Responses[AddResponse(iwQue1)]=new AdditiveResponse()`.

Для иллюстрации возможности создания группы откликов в пределах одной статистики приведём следующую конструкцию:

```
Responses[AddResponse(iNumOut)]=new NumInResponse(-1);
for(int i=0;i<groups;i++)     //groups – количество типов заявок
  Responses[AddResponse(iNumOut)]=new NumInResponse(i);
```

Правила определения секций программного файла

Чтобы обеспечить правильную область видимости системных внешних идентификаторов, используемых при написании программ ИМ, программный файл информационного модуля обязательно должен начинаться определением макро `MODEL`.

- <определение виртуальных методов классов экспериментов>

В этой секции подлежат определению виртуальные методы класса `Experiment`:

– `CreateStructure` – задание всей структуры модели; здесь создаются объекты датчиков случайных чисел, источников, ресурсов, потребителей и операндов сервисов для всего эксперимента;

– `FinishExperiment` – обратные для `CreateStructure` операции; также здесь может производиться расчёт некоторых откликов модели;

– `BeginReplication` – функция срабатывает перед началом каждого опыта с моделью и предназначена для изменения конфигурации модели перед опытом, корректировки откликов и выполнения других промежуточных действий; номер текущего опыта доступен через переменную `curRepl`;

– `FinishReplication` – в этой функции могут производиться все необходимые действия по окончанию каждого опыта с ИМ по усмотрению разработчика;

– `BeginRun` – выполняется перед началом каждого прогона с ИМ; в случае опыта с повторными прогонами в этой функции необходимо инициализировать цепочки событий для начала прогона при помощи функции `Active` с параметрами «номер источника сервиса» и «модельное время срабатывания» и/или отработать с откликами ИМ; номер текущего прогона доступен через переменную `curRun`;

– `FinishRun` – в этой функции могут по усмотрению производиться все необходимые действия при окончании каждого прогона с ИМ (например, расчёт откликов после окончания прогона);

– `Report(int x)` – полное переопределение этой функции позволяет разработчику заменить стандартные средства документирования результатов ИЭ своими собственными. Расширение позволяет более гибко использовать стандартные средства. Параметр `x=1` означает необходимость вывода статистической отчётности в том числе и по всем прогонам ИЭ (а не только по всем опытам эксперимента); параметр `x=0` означает необходимость вывода статистической отчётности только по опытам ИЭ;

– `ParametersReport` - определение этой виртуальной функции позволяет организовать вывод в файл дополнительных параметров эксперимента перед основной статистической отчётностью.

- <определение активностей>

В соответствии с БСФ [2] ТППУ активность – это функция, включающая в себя: а) алгоритм обработки события, изменяющий состояние элементов ИМ; б) определение следующего события, которое произойдет в связи с изменением состояния элемента; в) планирование времени наступления следующего события, что приводит к преобразованию массива событий; г) возврат в управляющую программу моделирования (УПМ) признака продолжения или окончания моделирования. При этом самыми важными являются следующие основные сочетания системных функций для завершения активности и корректного

продолжения моделирования:

1) `SetDirection(int numdvc), StopService()`

Функция `SetDirection` устанавливает новый указатель направления движения (`numdvc`) потребителя сервиса, после чего вызывается функция `StopService` для завершения обслуживания потребителя на данном ресурсе сервиса.

2) `SetNextActivity(имя_активности)`

Функция `SetNextActivity` устанавливает новую активность для текущего процесса, а соответствующий вызов происходит без изменения модельного времени.

3) `SetNextActivity(имя_активности), WaitTime(float step)`

Функция `SetNextActivity` устанавливает новую активность для текущего процесса, а соответствующий вызов происходит через `step` единиц модельного времени. Можно также указывать функцию `StopService` в качестве аргумента функции `SetNextActivity`. В этом случае окончание обслуживания будет реализовано прямым вызовом из УПМ.

4) `SetDirection(int numdvc), WaitStopService(float maxwaittime)`

Функция `WaitStopService` используется вместо функции `StopService`, когда время ожидания потребителя сервиса на текущем ресурсе сервиса (в случае невозможности дальнейшего его продвижения по модели) ограничивается величиной `maxwaittime`. Новый ресурс сервиса для дальнейшего движения или удаление потребителя сервиса из ИМ обеспечивается соответствующей активностью ресурса сервиса.

Заключение

Правила определения информационных модулей программ ИМ ТППУ позволяют выдержать единый стиль описания для всех моделей библиотеки ПТИ АИМ ТППУ. Это обстоятельство существенно облегчает разработчикам поиск нужных секций в программах моделей, повышает прозрачность и читабельность кода, что положительно влияет на обучение новых специалистов.

Abstract. The information module is the essential part of every simulation program written with the help of the program-technological toolkit of simulation modeling automation of technological processes of services production. The most important rules of the definition of such kind of modules are given in this article.

Литература

1. В. Д. Левчук, Базовая схема формализации системы моделирования MICIC4, Проблеми програмування, № 1 (2005), 85–96.

2. И. В. Максимей, В. Д. Левчук и др., Задачи и модели ИСО, Ч. 3. Технология имитации на ЭВМ и принятие решений, Уч. пособие, Гомель, БелГУТ, 1999.

Гомельский государственный
университет имени Ф. Скорины

Поступило 15.05.06