Известия Гомельского государственного университета
имени Ф. Скорины, №3(18), 2003

УДК 681.3

Структура программы имитационной модели на языке моделирования MICIC4

В.Д. Левчук, В.В. Старченко, А.С. Помаз, А.М. Евтухов, В.Н. Леванцов

Введение

Решение многих актуальных задач управления, проектирования и исследования технических, экономических, телекоммуникационных и других систем требует их предварительного изучения на моделях, что приводит к привлечению специалистов разных профилей [1–3]. Современные информационные технологии создают предпосылки для их эффективного сотрудничества. При этом само исследование должно проводиться в рамках общей методологии, позволяющей разрабатывать проект на моделирование от постановки задачи до передачи конечных результатов в промышленную эксплуатацию.

Рассмотрим взаимодействие специалистов, вовлеченных в процесс моделирования. Идея воспользоваться инструментальными средствами моделирования может прийти к администратору, который имеет внутри своего предприятия научное подразделение, где по определению должны быть системные аналитики (математики) и прикладные программисты (или просто программисты). При этом новое поколение системных аналитиков владеет технологиями программирования и основами алгоритмизации. В противном случае заказчик оплачивает разработку проектов внешней научной организации, укомплектованной, как правило, еще более профессиональными кадрами.

С другой стороны, реализованные модели (приложения или некоторые подсистемы приложения) являются составной частью инструментария инженерного подразделения заказчика, которое решает повседневные задачи функционирования и управления предприятием. От сотрудников данного подразделения ни в коем случае нельзя требовать знания программирования. С системным аналитиком инженера объединяет знание методики постановки натурных и модельных экспериментов. Итак, встает задача разделения труда в рамках программно-технологического инструментария имитационного моделирования.

1. Цели имитационного моделирования сложной системы

Естественно предполагать, что каждый специалист имеет свое представление о сложной системе (СС). Оно соответствует его профессиональным интересам, знаниям, опыту. Администратор склонен рассматривать управляемую им систему как «черный ящик», который в ответ на входные воздействия получает определенные результаты. Инженер и системный аналитик рассматривают отклики системы как следствие заданной структуры объекта моделирования и значений параметров отдельных его подсистем и элементов. При этом они четко представляют механизм взаимодействия всех составных частей СС. Передача информации идет от инженера к аналитику. Последний в результате абстрагирования должен формализовать задачу и сузить ее до отдельных конструкций и понятий, с которыми работают программисты.

Таким образом, в основу системы моделирования (СМ) MICIC4, как и предыдущих ее версий [4], положена концепция многоуровневого представления СС. Это даёт возможность совместно работать в одном проекте на моделирование различным специалистам. Уровни специалистов определяются целями работы с имитационной моделью (ИМ). Можно выделить следующие цели:

• создание концептуальной модели, отражающей в первую очередь взгляд на объект моделирования со стороны коллектива заказчика;

• преобразование концептуальной модели в формальное описание СС, соответствующее представлениям аналитика и служащее исходным заданием для программиста;

• алгоритмизация, кодирование и отладка механизмов информационного взаимодействия отдельных элементов ИМ;

• описание вариантов использования ИМ, т.е. схемы постановки, реализации, и обработки результатов имитационных экспериментов (ИЭ), настроенных на квалификацию инженера или исследователя–аналитика, минимально владеющего программированием;

• демонстрация результатов моделирования и их следствий администратору и потенциальным клиентам предприятия, заказавшего проект на моделирование.

2. Инструменты достижения целей

При построении концептуальной модели следует придерживаться блочно–сетевой концепции структуризации, принятой в MICIC4. Это позволит системному аналитику естественным образом построить описание объекта в соответствии с базовой схемой формализации MICIC4. Центральное место в данной иерархии целей занимает этап программирования ИМ. Программист должен воспользоваться языком моделирования MICIC4, чтобы обеспечить реализацию ИЭ. Поскольку в настоящее время существуют очень развитые технологии программирования, то данный язык моделирования реализован как библиотека к широко распространенному объектно–ориентированному языку программирования C++. В таком случае от программиста не потребуется дополнительно изучать новый язык моделирования, детально вникать в синтаксические конструкции, приобретать опыт отладки и верификации программ ИМ. Он будет использовать привычный ему полнофункциональный инструментарий интегрированной среды разработки приложений на C++. Именно отсутствие необходимости специального обучения может привести к решению реализации проекта силами собственного научно–исследовательского подразделения заказчика.

Таким образом, разработчик языка моделирования предоставляет программисту интерфейс для определения структуры ИМ, описания информационного взаимодействия между ее элементами, обработки результатов моделирования в процессе реализации ИЭ. После завершения ИЭ важно обеспечить передачу результатов моделирования во внешнюю программную среду, позволяющую наилучшим для заказчика образом презентовать достигнутый эффект, или для дальнейшего использования.

3. Взаимодействие интерфейсов при программировании имитационной модели

В соответствии с поставленными целями в языке моделирования MICIC4 предлагается трехмодульная структура программы ИМ. Назначение каждого модуля представлено в табл. 1. Из таблицы следует, что в процессе написания программы ИМ участвуют три класса специалистов. Аналитик занимается постановкой экспериментов с некоторым семейством моделей, которое в обобщённом виде создает программист, и обработкой результатов моделирования. Все множество ИМ соответствует концептуальному описанию исходной СС. Программист, в свою очередь, использует тот интерфейс, который предоставляет ему разработчик СМ MICIC4. То есть именно программист, активно использующий язык моделирования для определения структуры ИМ и описания информационного взаимодействия элементов в общем виде, должен владеть технологией объектно–ориентированного программирования. Программный интерфейс является постоянным и функционально полным в рамках базовой схемы формализации СМ MICIC4, что позволяет создавать ИМ различных по своей природе СС.

В силу того, что разработчик СМ MICIC4 всегда один и тот же, системный модуль является уникальным и неизменным для всех ИМ, написанных на языке моделирования

MICIC4. Аналитик и программист модели представляют различные научные или производственные коллективы. Так как в любом эксперименте независимо от конкретной структуры ИМ ее элементы взаимодействуют по фиксированным алгоритмам, то информационный модуль, создаваемый программистом, соответствует определенной концептуальной модели СС. Наконец, с одной и той же ИМ можно поставить произвольное количество ИЭ. Поэтому аналитик наиболее подходящим образом формирует способы постановки планов ИЭ и обработки результатов моделирования, определяя функциональные модули ИМ на основе одного и того же информационного модуля. При этом, если ИЭ будут встроены в некоторый стандартизированный пользовательский диалог, то непосредственно решением проблем исследования и проектирования СС будет заниматься инженер, который принимал активное участие в постановке задачи на моделирование. Таким образом, предложенная схема реализации проекта органично сочетает знания и опыт различных коллективов специалистов.

Таблица 1. Назначение модулей в программе на языке MICIC4.

Модуль	Назначение	Содержание
Функциональный	Определение и реализация эксперимента с моделью	• Определение глобальных данных; • выбор типа эксперимента; • инициализация параметров модели; • определение откликов; • запуск модели на имитацию; • обработка и вывод результатов.
Информационный	Описание информационного взаимодействия элементов модели	• Определение иерархической структуры имитационной модели; • описание интерфейсов компонентов модели; • определение типов экспериментов; • программирование активностей; • переопределение необходимого для данной модели множества функций и методов; • предоставление требуемых аналитиком возможностей по постановке экспериментов.
Системный	Предоставление интерфейса для программиста имитационной модели	• Реализация базовых классов для компонентов модели, статистики, откликов, постановки эксперимента и стохастических потоков внешних событий; • разработка множества системных функций; • обеспечение корректной работы алгоритмов организации квазипараллелизма и управления имитацией; • предоставление механизмов работы с датчиками псевдослучайных чисел; • настройка типовых и часто используемых внутренних переменных для объектов модели; • программирование функций и свойств элементов в соответствии с базовой схемой формализации.

4. Создание информационного модуля

После того, как в соответствии с базовой схемой формализации построена концептуальная и формальная модели, необходимо на основе имеющихся данных приступить к наполнению информационного модуля программы, т.е. к описанию информационного взаимодействия элементов модели. Для программирования этого модуля необходимо создать два C++ файла. Назовем их условно `model.h` и `model.cpp`. Далее все описания должны опираться на системный интерфейс языка MICIC4 и его документацию.

Файл `model.h` предоставляет интерфейс для функционального модуля и целиком включается в него. В данном файле нужно:

- включить интерфейс системного модуля;
- перечислить и определить компоненты имитационной модели;
- перечислить и определить статистики имитационной модели;
- определить наследуемые в данной модели классы;
- описать интерфейсы функций и методов, которые будут использоваться в файле `model.cpp`.

Файл `model.cpp` служит для определения описанных в файле `model.h` функций. Это в первую очередь активности, а также функции–условия, уникальные дисциплины выбора транзактов на обслуживание, собственные и виртуальные функции–члены классов и т.п.

Активность представляет собой реализованный в виде функции некоторый алгоритм, аппроксимирующий определённые функциональные действия при обслуживании транзакта на устройстве либо функционирования генератора и заканчивающийся вызовом функции преобразования списка событий. То есть все функциональные действия совершаются в один и тот же момент модельного времени, а перед выходом из активности разработчик модели должен запланировать очередное событие в активном процессе. Таким образом, выполнение активности приводит к свершению очередного события в имитационной модели и, следовательно, к изменению состояния имитационной модели. Порядком смены активностей в модельном времени управляет подсистема имитации с помощью массива событий на основании состояния модели и определённой разработчиком последовательности смены активностей.

В MICIC4 активность, то есть функция языка программирования C++, не имеет аргументов и возвращает признак завершения прогона ИМ: 1 – продолжить, 0 – остановить. В теле активности могут быть описаны локальные переменные, использоваться операторы и вызовы стандартных функций языка C++, функций языка моделирования MICIC4, а так же собственных функций разработчика ИМ.

Множество стандартных функций MICIC4 можно разделить на следующие группы:

- преобразования массива событий;
- создания элементов модели: транзактов, устройств, генераторов;
- локализации элементов модели и получения значений их свойств;
- изменения состояния элементов модели;
- организации дисциплины выбора транзакта на обслуживание;
- управления случайными потоками;
- автоматизации сбора и обработки откликов.

Практически коды активностей занимают подавляющую часть файла `model.cpp`. Обязательно также необходимо включить переопределение виртуальных функций, предназначенных для подготовки прогона, опыта и эксперимента.

5. Создание функционального модуля

Функциональный модуль представляет собой отдельный файл, начинающийся с определения размеров глобальных системных массивов в виде макросов языка C++. Далее исследователю ИМ необходимо включить заголовочный файл информационного модуля `model.h` и написать функцию `main()`. В простейшем и наиболее распространенном случае в главной функции программы ИМ необходимо создать объект одного из наследников класса семейства `Эксперимент`. Этот объект должен реализовать эксперимент, все параметры и правила проведения которого были определены в информационном модуле разработчиком модели. Исследователю не обязательно рассчитывать все отклики в эксперименте. Именно те, которые ему нужны для решения конкретной задачи, он создает с помощью функции `AddResponce()`.

Завершается функция `main()` вызовом следующих методов созданного объекта:

• выполнить эксперимент;

• создать отчет о результатах эксперимента;

• освободить память, где находятся результаты эксперимента, если необходимо реализовать еще один эксперимент.

В заключение, чтобы проиллюстрировать простоту программирования функционального модуля и отказ от требования фундаментальных знаний языка программирования C++ у исследователя модели, ниже приведен типичный пример функционального модуля.

```
      #define MAXTRANSACTIONS 200
#define MAXEVENTS 200
#define MAXRNGS 20              //размеры глобальных массивов
#define MAXDEVICES 20
#define MAXGENERATORS 5
#define MAXRESPONSES 30

#include "model.h"              //включение интерфейса информационного модуля
void main(void) {
                        //параметры эксперимента (как правило, и всей модели)
      int gnum=2, applnum=2, channels=2, queuelen=4, int runnum=3;
      float maxwaittime=4.0,downserv=6.0,upserv=8.0;
      float transition_time=300, sim_time=900, freq1[]={5.0,6.35,9.5,12.0,20.3};
            //создание объекта, соответствующего однофакторному эксперименту
      smo2 s1(gnum,applnum,freq1,queuelen,maxwaittime,channels,downserv,upserv,
            runnum,transition_time,sim_time);
                        //создание множества необходимых откликов
      Responses[AddResponse(iNumIn)]=new AdditiveResponse();
      Responses[AddResponse(iNumBroke)]=new AdditiveResponse();
      Responses[AddResponse(ipBroke)]=new PBrokeResponse();
      Responses[AddResponse(ilQue)]=new ContinuousResponse();
      s1.Execute();             //выполнить эксперимент
      s1.Report();              //создать отчет
      s1.FreeResults();         //освободить память
}
```

Abstract

The structure of the simulation model program written in accordance with the simulation toolkit MICIC4 is discussed in this article. The offered methodology solves the problem of the division of labor within the framework of the same simulation toolkit.

Литература

1. Максимей И.В. Имитационное моделирование на ЭВМ. – М.: Радио и связь, 1988. – 232 с.

2. Технология системного моделирования/ Е.Ф. Аврамчук, А.А. Вавилов, С.В. Емельянов и др. Под общ. ред. С.В. Емельянова, В.В. Калашникова и др. – М.: Машиностроение, Берлин: Техник, 1988. – 520 с.

3. Шеннон Р. Имитационное моделирование систем – искусство и наука. – М.: Мир, 1978. – 418 с.

4. Задачи и модели ИСО. Ч.3. Технология имитации на ЭВМ и принятие решений: Уч. пособие / И.В.Максимей, В.Д.Левчук, С.П.Жогаль, В.Н.Подобедов. – Гомель: БелГУТ, 1999. – 150 с.

Гомельский государственный
университет им. Ф.Скорины

Поступило 7.04.03