УДК 681.3

В.Д. ЛЕВЧУК

ПРОГРАММНЫЕ ИНТЕРФЕЙСЫ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ СЛОЖНЫХ ДИСКРЕТНЫХ СИСТЕМ

Abstract: *Program interfaces needed for the creation of a simulation model are discussed in this report. The composition of the simulation toolkit MICIC4 oriented to a team way of a simulation project implementation is considered.*
Key words: *simulation toolkit MICIC4, simulation model, simulation experiment, program interface, complex discrete system.*

Анотація: *Розглядаються питання розробки програмних інтерфейсів для створення імітаційної моделі. Приводиться склад програмно-технологічного інструментарію MICIC4, орієнтованого на колективну реалізацію проекту на імітаційне моделювання.*
Ключові слова: *програмно-технологічний інструментарій MICIC4, імітаційна модель, імітаційний експеримент, програмний інтерфейс, складна дискретна система.*

Аннотация: *Рассматриваются вопросы разработки программных интерфейсов для создания имитационной модели. Приводится состав программно-технологического инструментария MICIC4, ориентированного на коллективную реализацию проекта на имитационное моделирование.*
Ключевые слова: *программно-технологический инструментарий MICIC4, имитационная модель, имитационный эксперимент, программный интерфейс, сложная дискретная система.*

1. Введение

Среди факторов, препятствовавших применению метода имитационного моделирования вплоть до 90-х годов прошлого века, первым назывался ресурсоемкость создания программы имитационной модели (ИМ) [1, 2]. Программированием в то время занимались исключительно профессионалы. Однако при этом была поставлена задача последовательно, шаг за шагом приобщить к нему другие категории специалистов. Дополнительным стимулом стала идея поставлять пользовательские приложения вместе с макроязыком, предназначенным для автоматизации наиболее часто выполняемых операций.

В процессе решения данной задачи интенсивно развивались как сами информационные технологии, так и методики их изучения. В настоящее время технологии программирования, включая объектно-ориентированное программирование, входят в программу обучения всех инженерных специальностей. Поэтому имеются существенные предпосылки для уменьшения веса фактора ресурсоемкости при изучении сложных систем методом имитационного моделирования.

Изначально такие популярные языки моделирования, как GPSS, SOL, GASP IV, Simscript, НЕДИС, АСИМ [1–5], были ориентированы, в первую очередь, на пользователя, имеющего отдаленное представление о программировании. В силу данного обстоятельства разработчики моделей зачастую не имели возможности адекватно отразить функционирование изучаемого объекта в достаточной степени. Упрощение исходных предпосылок (концептуальных моделей) ставило под сомнение у заказчиков проектов ценность получаемых результатов.

С другой стороны, пользователь при таком подходе в едином лице представлял и разработчика, и исследователя, и заказчика ИМ. Однако в реальности ИМ создаются не одиночками, а, как правило, научными коллективами, в которых существуют жесткая иерархия и распределение труда. Новые информационные технологии как раз и направлены на согласование коллективного характера работы. Настоящая статья посвящена разработке такого моделирующего

инструментария, который позволяет эффективно объединять усилия различных специалистов для решения задач исследования и проектирования сложных дискретных систем (СДС).

2. Взаимодействие специалистов при имитационном моделировании

Естественно предполагать, что каждый специалист имеет свое представление о сложной системе. 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 (представитель коллектива заказчика проекта на имитационное моделирование) и системный аналитик (член исполнителя проекта) 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 СДС. 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 программно-технологического инструментария (ПТИ) MICIC4, как и предыдущих его версий, положена концепция многоуровневого представления СДС. Это даёт возможность совместно работать в одном проекте на моделирование различным специалистам. Уровни специалистов определяются целями работы с 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 СДС, соответствующее представлениям аналитика и служащее исходным заданием для программиста;

– алгоритмизация, кодирование, отладка и верификация механизмов информационного взаимодействия отдельных элементов ИМ программистом;

– постановка, реализация и обработка результатов имитационных экспериментов (ИЭ) с ИМ, которые выполняются системным аналитиком;

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

3. 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 [6]. Центральное место в данной иерархии целей занимает этап программирования ИМ. Программист должен воспользоваться *интерфейсом системного модуля* MICIC4, чтобы обеспечить реализацию ИЭ. Системный модуль представляет собой библиотеку к широко распространенному объектно-ориентированному языку программирования C++ [7], который далее по сложившейся традиции будем называть язык моделирования MICIC4. В результате программисту не требуется дополнительно изучать новый язык моделирования, детально вникать в синтаксические конструкции, приобретать опыт отладки и верификации программ ИМ. Он должен использовать

привычный ему полнофункциональный инструментарий *интегрированной среды разработки приложений на C++* и технологическое обеспечение MICIC4.

Таким образом, MICIC4 предоставляет программисту интерфейс для определения структуры ИМ, описания информационного взаимодействия между ее элементами, сбора и накопления статистической информации в процессе реализации ИЭ. После завершения ИЭ MICIC4 обеспечивает передачу во внешнюю программную среду результатов моделирования, позволяющую эффективно их обработать и наилучшим для заказчика образом презентовать достигнутый эффект.

4. Структура программы имитационной модели

Программа ИМ, реализованная средствами MICIC4, представляет собой совокупность трех модулей: системного, информационного и функционального. Системный модуль является уникальным и неизменным для всех ИМ, написанных на языке моделирования MICIC4. Его программный интерфейс является постоянным и функционально полным в рамках базовой схемы формализации ПТИ MICIC4, что позволяет создавать ИМ различных по своей природе СДС.

Информационный модуль, создаваемый программистом, предназначен для определения структуры ИМ и отображения механизмов взаимодействия элементов ИМ в соответствии с определенной концептуальной моделью СДС. Программист использует интерфейс, предоставляемый ему разработчиком системного модуля MICIC4. Именно программист является единственным представителем коллектива исполнителя проекта на имитационное моделирование, который должен владеть технологией объектно–ориентированного программирования.

Наконец, с одной и той же ИМ можно поставить произвольное количество ИЭ. Поэтому аналитик выбирает способы постановки планов ИЭ и обработки результатов моделирования, определяя функциональные модули ИМ на основе одного и того же информационного модуля. Написание функционального модуля – это тривиальный описательный процесс, в котором аналитику не требуется даже знания алгоритмов.

5. Создание информационного модуля

После того, как в соответствии с базовой схемой формализации построена концептуальная и формальная модели, необходимо создать информационный модуль программы ИМ. Для его программирования необходимо написать два C++ файла. Назовем их условно model.h и model.cpp. Далее все описания должны соответствовать системному интерфейсу языка MICIC4.

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 и статистики имитационной модели;

– определить наследуемые в данной ИМ классы компонентов и различных видов ИЭ;

– описать интерфейсы функций и методов, которые будут использоваться в файле model.cpp.

Файл model.cpp служит для определения описанных в файле model.h методов классов. Это, в первую очередь, активности, а также определения виртуальных методов. Активность представляет собой реализованный в виде функции некоторый алгоритм, аппроксимирующий определённые функциональные действия при взаимодействии динамического и статического элементов ИМ и заканчивающийся вызовом функции преобразования массива событий. То есть все функциональные действия совершаются в один и тот же момент модельного времени, а перед выходом из активности программист должен запланировать очередное событие в активном процессе. Таким образом, выполнение активности приводит к свершению очередного события в ИМ и, следовательно, к изменению ее состояния. Порядок смены активностей в модельном времени обеспечивается с помощью массива событий, доступ к которому напрямую из программы ИМ не рекомендуется.

В MICIC4 активность, то есть метод одного из базовых классов, 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.

Множество стандартных функций MICIC4 можно разделить на следующие группы:

– преобразование массива событий;

– создание элементов модели: транзактов, устройств, генераторов;

– локализация элементов модели и получение значений их свойств;

– изменение состояния элементов модели;

– организация дисциплины выбора транзакта на обслуживание;

– управление случайными потоками;

– автоматизация сбора и обработки откликов и трассировочной информации.

Коды активностей занимают подавляющую часть файла model.cpp. Дополнительно необходимо переопределить виртуальные методы наследуемых классов, предназначенных для постановки ИЭ.

6. 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.

Функция main() в простейшем ИЭ содержит вызовы следующих методов созданного объекта:

– инициализировать эксперимент;

– выполнить эксперимент;

– создать отчет о результатах эксперимента.

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

```
#define MAXTRANSACTIONS 2000
#define MAXEVENTS 2000
#define MAXRNGS 20              //размеры глобальных массивов
#define MAXDEVICES 50
#define MAXGENERATORS 10
#define MAXRESPONSES 30
#include "model.h"              //включение интерфейса информационного модуля
void main(void) {
                //создание объекта, соответствующего некоторому эксперименту
        model m1("data.txt", "results.txt");
        m1.Init();              //подготовительные операции
        m1.Execute();   //выполнить эксперимент
        m1.Report(0);   //создать отчет
}
```

7. Выводы

В статье обоснован подход, примененный при разработке ПТИ автоматизации имитационного моделирования MICIC4, где язык моделирования реализован как библиотека к широко распространенному объектно-ориентированному языку программирования C++. В соответствии со сделанными предпосылками в языке моделирования MICIC4 предложена трехмодульная структура программы ИМ. Разработчик языка моделирования MICIC4 предоставляет прикладному программисту системный модуль, содержащий интерфейс для определения структуры ИМ, описания информационного взаимодействия между ее элементами, обработки результатов моделирования в процессе реализации ИЭ. В результате выполнения данных этапов создается информационный модуль, который служит исходным шаблоном для написания множества тривиальных по своему синтаксису и семантике функциональных модулей. С их помощью исследователь реализует решение задач моделирования предметной области. Таким образом, информационный модуль обеспечивает проблемную ориентацию базового инструментария имитационного моделирования. Множество функциональных модулей программы ИМ на языке моделирования MICIC4 определяет форму предметной ориентации.

СПИСОК ЛИТЕРАТУРЫ

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. Томашевский В.Н., Жданова Е.Г. Имитационное моделирование в среде GPSS. – М.: Бестселлер, 2003. – 416 с.
4. Рыжиков Ю.И. Имитационное моделирование. Теория и технологии. – СПб.: КОРОНА принт; М.: Альтекс-А, 2004. – 384 с.
5. Лоу А., Кельтон В. Имитационное моделирование. Классика CS. 3-е изд. – СПб.: Питер; Киев: BHV, 2004. – 847 с.
6. Левчук В.Д. Базовая схема формализации системы моделирования MICIC4 // Проблеми програмування. – 2005. – №1. – С. 85–96.
7. Страуструп Б. Язык программирования C++. – 3-е изд. – СПб.: Невский диалект; М.: Издательство БИНОМ, 1999. – 991 с.