Известия Гомельского государственного университета
имени Ф. Скорины, №5(32), 2005

УДК 681.3

Постановка имитационных экспериментов средствами системы моделирования MICIC4

В. Д. Левчук

Введение

Имитационное моделирование разнообразных сложных систем (СС) является эффективным при наличии методо–ориентированного формального способа организации квазипараллелизма [1]. Аналогично, следует стремиться к решению множества возникающих задач на основе ограниченного количества планов имитационных экспериментов (ИЭ). Разработка формальной схемы организации ИЭ облегчается тем фактором, что имитационные модели (ИМ) в первую очередь предназначены для ликвидации дефицита натурной информации, собираемой в процессе функционирования СС. Моделью натурной информации служит полученная по завершению ИЭ с ИМ выборка, математическим описанием которой является матрица – сравнительно тривиальная структура данных. На современном этапе развития информационных технологий дальнейшая обработка этой матрицы, строго говоря, является предметом не имитационного, а статистического моделирования.

В MICIC4 [2] ИЭ представляет собой совокупность опытов (прогонов в широком смысле), каждый из которых состоит из множества реплик. Опыты различаются между собой начальными состояниями (значениями параметров, переменных, составом объектов) ИМ. Каждая реплика в опыте выполняется с одним и тем же начальным состоянием ИМ, но с различными начальными значениями базовых датчиков случайных чисел. За счет реализации реплик с различными последовательностями случайных чисел обеспечивается стохастический характер результатов моделирования.

Как выше отмечалось, выходными данными опыта с ИМ является матрица. Количество строк этой матрицы равно количеству реплик в опыте, а количество столбцов совпадает с количеством откликов ИМ. Инструменты MICIC4 при формировании матрицы обеспечивают решение следующих тактических задач планирования ИЭ:

- задания начального состояния опыта;
- выбора между опытом с повторными репликами или непрерывным прогоном;
- определения условий окончания реплики, опыта и всего эксперимента в целом;
- оценки окончания переходного периода в ИМ;
- постановки ИЭ, позволяющих уменьшить дисперсию (в том числе путем манипуляций с датчиками случайных чисел).

Исследователь СС имеет простой и наглядный программный интерфейс для постановки ИЭ, позволяющих с помощью методов и пакетов статистического анализа данных решать следующие стратегические задачи:

- простейший анализ типа «что-если», когда исследователя интересует количественная оценка, характеризующая функционирование одного из вариантов СС;
- выявление эффектов воздействия некоторых факторов на состояние СС или показатели ее функционирования;
- построение поверхности отклика, которая может дать качественную оценку характеристик системы, например, окрестности экстремумов, крутизну поверхности, наличие седловой точки или гребня и т.п.;

- поиск такой комбинации уровней количественных факторов, которая обеспечивает оптимальное значение критерия качества системы, используя, например, методологию поверхности отклика;
- сравнение по критерию эффективности различных вариантов СС или стратегий их применения с целью определения такого подмножества, которое наиболее предпочтительно по принятому критерию.

В данной статье приводятся ключевые особенности системы моделирования MICIC4, положенные в основу формальной схемы организации ИЭ.

Интерфейс класса для имитационного эксперимента

Все ИЭ в MICIC4 являются наследниками базового класса `Experiment`. Он позволяет организовывать различные виды ИЭ: с наличием и отсутствием переходного периода; с повторными репликами и непрерывным прогоном; с признаками окончания реплики, опыта и всего ИЭ по модельному времени, количеству реализаций или условию. Ниже приводится описание конструктора базового класса и его наиболее важных методов.

Конструктор класса `Experiment`:

```
Experiment(char *outfile=NULL, float run_time=MAXFLOAT, float
warmup_time=0.0,
        int max_repls=1, int max_runs=1, int is_run_replicated=1)
```

Аргументы конструктора:

- `outfile` – файл для отчета ИЭ (для вывода на экран используется значение `NULL`);
- `run_time` – модельное время окончания реплики (если реплика завершается по наступлению условия, то используется значение `MAXFLOAT`);
- `warmup_time` – модельное время окончания переходного периода;
- `max_repls` – максимальное количество реплик в опыте (прогоне);
- `max_runs` – максимальное количество опытов в ИЭ;
- `is_run_replicated` – признак опыта в виде повторных реплик.

Методы класса `Experiment`:

```
virtual void CreateStructure(void)
```

Создает структуру ИМ и ее базовое начальное состояние. Вызывается из программы ИМ не напрямую, а через системный метод `Init`.

```
void Init(void)
```

Подготавливает ИМ к реализации ИЭ, вызывая, в частности, метод `CreateStructure`. Для любого объекта–наследника класса `Experiment` данный метод должен быть вызван единственный раз до первого вызова метода `Execute`.

```
void Execute(void)
```

Реализует ИЭ. С одним и тем же объектом–наследником класса `Experiment` можно поставить любое количество ИЭ. При первом использовании данного метода выделяется память под значения факторов и откликов ИЭ. При повторных вызовах происходит только инициализация переменных класса, отвечающих за реализацию ИЭ, а затем в процессе имитации результаты обновляются. Если состав факторов и откликов изменился, то прежний объект для реализации ИЭ не подходит, т.к. необходимы другие объемы памяти под результаты имитации. Рассмотрим ключевые методы, которые обеспечивают организацию различных видов ИЭ. Они объявлены в базовом классе виртуальными, определяются поэтому разработчиком ИМ в ее информационном модуле, а вызываются из метода `Execute`.

```
virtual void BeginExperiment(void)
```

Обеспечивает выполнение некоторых действий непосредственно перед началом ИЭ. Например, данный метод можно использовать для задания уровней факторов.

```
virtual void FinishExperiment(void)
```

Обеспечивает выполнение некоторых действий по завершению ИЭ.

```
virtual int eoExperiment(void)
```

Задает условие окончания ИЭ, отличное от стандартного (по количеству опытов).

```
virtual void BeginRun(void)
```

Обеспечивает выполнение некоторых действий до начала опыта.

```
virtual void FinishRun(void)
```

Обеспечивает выполнение некоторых действий по завершению опыта.

```
virtual int eoRun(void)
```

Задает условие окончания опыта, отличное от стандартного (по количеству реплик).

```
virtual void BeginReplication(void)
```

Обеспечивает выполнение некоторых действий до начала реплики. В данном методе разработчик ИМ должен перевести генераторы внешних динамических элементов в активное состояние.

```
virtual void FinishReplication(void)
```

Обеспечивает выполнение некоторых действий по завершению реплики.

```
virtual int Report(int depth=0)
```

Этот метод, как и `Execute`, вызывается из функционального модуля и создает отчет по результатам реализации ИЭ в текстовом файле, задаваемом аргументом конструктора `outfile`. Если расширение файла начинается со строки `"xl"`, то отчет будет адаптирован под открытие и дальнейший анализ в табличном процессоре Excel. Аргумент `depth` используется для управления уровнем подробности отчета и по умолчанию принимает значение 0 (отсутствие подробного отчета). При ненулевом значении данного аргумента создается дополнительный файл, содержащий значения откликов во всех репликах. Поскольку данный метод объявлен виртуальным, то разработчик модели может создавать собственный отчет по результатам ИЭ.

Структура функционального модуля

Непосредственная реализация плана ИЭ обеспечивается в функциональном модуле ИМ, написанной с помощью инструментария MICIC4. Для большинства задач код в теле функции `main` выглядит следующим образом.

```
Exp e1(val1, val2, …, valN);      //определение объекта-наследника класса
Experiment
                                  //и инициализация его значениями;
e1.Init();       //создание структуры модели
e1.Execute();     //реализация эксперимента
e1.Report();      //генерация стандартного отчета
```

В программе одного функционального модуля можно реализовать два и более ИЭ с одним и тем же объектом. С другой стороны, для каждого эксперимента можно использовать различные объекты и использовать их внутри линейных, условных и циклических алгоритмов. Написание подобных сценариев под силу «программистам», имеющим квалификацию инженера, системного аналитика или математика. Именно данной категории специалистов чаще всего приходится использовать ИМ в своих исследованиях. При этом одна и та же ИМ позволяет им получать решение одноразовых уникальных задач.

Заключение

Наиболее полно возможности интерфейса MICIC4 по постановке ИЭ проявляются, когда у исследователя внедрена информационная система, а задачи, решаемые методом имитационного моделирования, возникают регулярно. Тогда в информационную систему встраивается функциональный модуль, содержащий совокупность объектов, обеспечивающих реализацию ИЭ и обработку результатов моделирования. Именно в данном случае достигается полноценная предметная ориентация средств автоматизации имитационного моделирования.

При этом множество ИМ оказывается неотъемлемой частью программного обеспечения исследователя, которое по своей сути является автоматизированной системой управления предметной областью. Подобная возможность обеспечивается приведенным выше интерфейсом MICIC4 по постановке ИЭ.

Abstract. Key features of the simulation system MICIC4 proposed as a basis of a formal scheme of a simulation experiment design are considered in the article.

Литература

1. *Технология системного моделирования* / Под общ. ред. С. В. Емельянова, В. В. Калашникова и др., Москва, Машиностроение; Берлин, Техник, 1988.

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

Гомельский государственный
университет им. Ф. Скорины

Поступило 27.06.05