УДК 004.738.5 : 519.682 : 519.872

Имитационное моделирование с применением библиотеки классов языка Java, разработанной для «облачных» сервисов

А.Ю. Быков [1], Ф.А. Панфилов [1], О.О. Сумарокова [1]

[1] МГТУ им. Н.Э. Баумана, Москва, 105005, Россия

Исследована тенденция перехода к «облачным» вычислениям при разработке инструментальных средств имитационного моделирования. Для разработки «облачных» сервисов имитационного моделирования предложено использовать язык Java. Представлено краткое описание библиотеки классов языка Java для разработки реализаций имитационных моделей в «облачных» сервисах. Рассмотрен пример реализации имитационной модели системы массового обслуживания типа M/M/1 в виде Java-апплета. Для проверки результатов моделирования представлен аналитический расчет модели и результаты, полученные на языке GPSS.

E-mail: abykov@bmstu.ru

Ключевые слова: *имитационное моделирование, языки моделирования, «облачные» вычисления.*

Инструментальные средства имитационного моделирования и «облачные» вычисления. В исследованиях сложных систем широко используется имитационное моделирование (ИМ) [1]. Имитационные модели применяются в различных областях экономики, авиации, на железнодорожном транспорте, в металлургии, нефтедобыче, судостроении и т. д. [2].

Для разработки реализаций имитационных моделей применяют инструментальные программные средства, основанные на использовании специализированных языков моделирования и графического интерфейса пользователя. В настоящее время на рынке имеются различные специализированные средства ИМ, например GPSS World, Arena, Extend и др. [1, 3].

Кроме создания специализированных средств ИМ существует второй подход к созданию средств ИМ, основанный на использовании универсальных языков программирования типа C, C++, Паскаль, Бейсик и т. п. Преимущество данного способа заключается в том, что дополнительно к специальным средствам моделирования можно использовать возможности языка программирования общего назначения. Также язык программирования может быть интегрирован в средство моделирования, например в AnyLogic интегрирован язык программирования Java. С помощью данных языков разрабатываются специализированные библиотеки функций и/или процедур (классов) для целей имитационного моделирования.

В работе [4] для разработки реализаций имитационных моделей предложено использовать библиотеку процедур и функций языка Паскаль. В статьях [5, 6] рассмотрена межплатформенная библиотека функций языка С, реализованная для операционных систем типа Windows и UNIX.

В последние годы получила развитие технология так называемых облачных вычислений [2], которая является развитием давно известной технологии распределенных вычислений, она ориентирована на использование сети Интернет. Учитывая то обстоятельство, что эксперименты на реализациях имитационных моделей, как правило, требуют много вычислительных ресурсов, технологии распределенных вычислений в ИМ используются достаточно давно. В докладе [2] исследованы тенденции развития средств имитационного моделирования, а также существующие реализации средств ИМ, в которых применяются технологии «облачных» вычислений. В частности, в [2] рассмотрена реализация, так называемого сервера GPSS, который может запускать одну или несколько копий GPSS World на удаленном сервере. Также представлена технология системной шины моделирования, основанная на сервис-ориентированной архитектуре. Перспективный подход основан на полноценной архитектуре «облачного» моделирования, получивший названия GPSS Cloud. По сути, рассмотренные «облачные» имитационные средства ориентированы на язык GPSS.

Кроме использования специализированных языков моделирования в «облачных» имитационных средствах перспективным является использование языка Java [7], характеризуемого свойствами универсальных языков программирования. Применение языка Java для «облачных» вычислений имеет ряд преимуществ:

– язык Java, по сути, является специализированным языком для сети Интернет и имеет специальные модели безопасности;

– приложения языка Java могут выполняться как на стороне сервера — приложения-сервлеты, так и на стороне клиента через Web-браузер — приложения-апплеты (использование апплетов как реализаций имитационных моделей применяется в средстве AnyLogic); также можно создавать обычные настольные приложения и приложения для мобильных устройств;

– язык Java является межплатформенным, приложения могут работать как на Windows-платформе, так и на различных реализациях операционных систем Linux;

– язык Java поддерживает механизмы многопоточности и другие современные возможности объектно-ориентированных языков (обработка исключений, графические библиотеки, компонентная модель и др.).

Обзор библиотеки классов языка Java для имитационного моделирования. Библиотека классов языка Java ориентирована на дискретно-событийное моделирование, реализует, по сути, те же ал-

горитмы, что и библиотека процедур и функций языка Паскаль [4] и библиотека функций языка С [5, 6], при этом используется объектно-ориентированный подход. Основные методы классов библиотеки по своим возможностям во многом повторяют операторы языка GPSS. Классы библиотеки включены в один пакет с именем SimJava. Для использования библиотеки классов достаточно иметь установленный на компьютере пакет разработчика JDK SE (Java Development Kit Standard Edition) версии 6 или выше. Для удобства можно дополнительно использовать одну из интегрированных сред разработки: Eclipse, NetBeans IDE и др.

Основные объекты модели создаются как объекты следующих классов:

Queue — класс для задания очередей в модели;

Facility — класс для задания одноканальных устройств;

Storage — класс для задания многоканальных устройств;

Histogram — класс для представления гистограмм;

Transact — класс для задания динамических объектов-транзактов в модели;

Rand — класс для задания генераторов псевдослучайных чисел;

Syst — класс для создания объекта «модель», любая имитационная модель должна создавать объект этого класса или объект производного класса; класс содержит основные системные методы, а также поддерживает отдельный поток для запуска модели; класс содержит методы для вывода результатов моделирования как в файл, так и в контейнер типа окна или апплета (в объект класс Container или производного класса).

Библиотека включает и вспомогательные классы для организации списков различных объектов: *ListF* — список одноканальных устройств; *ListS* — список многоканальных устройств; *ListQ* — список очередей; *ListH* — список гистограмм; *ListT* — список транзактов и некоторые другие.

Результаты моделирования одноканальной системы массового обслуживания (СМО) с неограниченной очередью. В качестве примера рассмотрим одноканальную СМО типа *M*/*M*/1 [8], которая обслуживает пуассоновский поток заявок. Время между заявками является случайной величиной, распределенной по показательному закону, среднее время между заявками 10 с ($\lambda = 0{,}1$ заявки в секунду). Время обслуживания одной заявки также распределено по показательному закону, среднее время обслуживания 8 с ($\mu = 1/8$ заявки в секунду). Если заявка приходит в момент времени, когда канал занят, то она становится в очередь, длина очереди неограниченна. Необходимо провести имитационное моделирование СМО при обслуживании 200 000 заявок, определить параметры СМО: параметры очереди задач (среднюю длину очереди и среднее время ожидания задачи в очереди), загрузку канала обслуживания.

Для контроля полученных результатов рассмотрим аналитическое решение этой задачи [8]. Основные соотношения имеют следующий вид:

$p_0 = 1 - \rho$ — вероятность того, что в системе нет заявок;

$\bar{Q}_{\text{очер}} = \dfrac{\rho^2}{1-\rho}$ — средняя длина очереди;

$\bar{T}_{\text{очер}} = \dfrac{\rho^2}{\lambda(1-\rho)}$ — среднее время ожидания в очереди;

$K_{\text{загр}} = 1 - p_0 = \rho$ — коэффициент загрузки канала обслуживания.

Подставив заданные значения, получаем результаты:

$$\bar{Q}_{\text{очер}} = 3{,}2;\ \bar{T}_{\text{очер}} = 32 \text{ с};\ K_{\text{загр}} = 0{,}8.$$

Реализация имитационной модели этой системы выполнена в виде Java-апплета, запускаемого через Web-браузер на стороне клиента. Исходный код модели — класс, производный от класса *Syst*, и исходный код апплета — класс, производный от класса *JApplet*, — представлены в распечатках 1 и 2. Результаты моделирования выводятся в рабочую область апплета — объект класса *Container* (контейнер). Для создания апплета был использован пакет javax.swing. Результаты работы апплета в браузере представлены на рисунке. Полученные результаты с определенной погрешностью соответствуют расчетам на аналитической модели. В распечатке 3 представлены результаты, полученные в среде GPSS World.

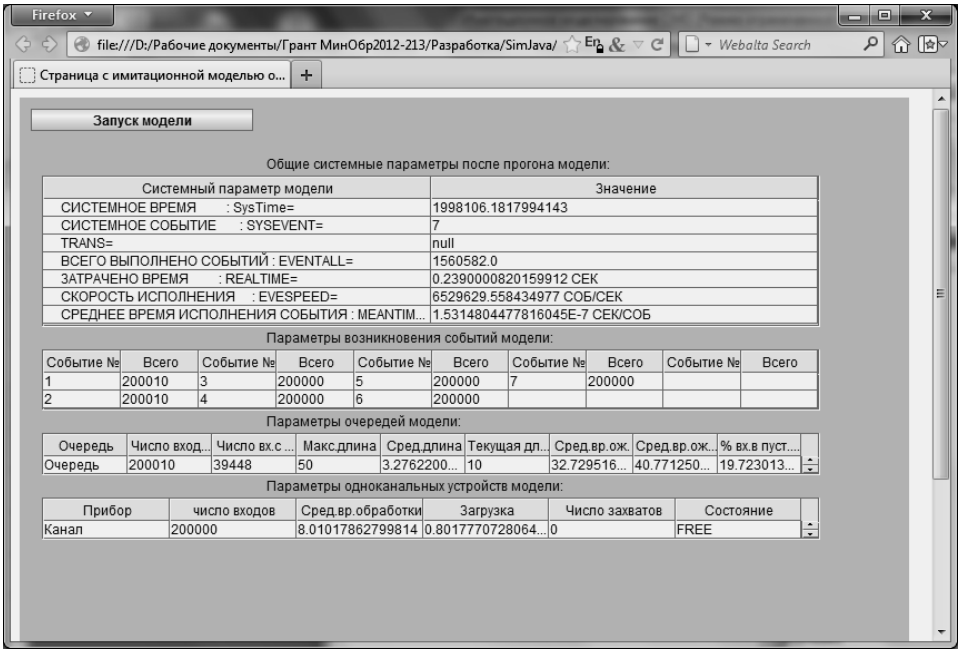
Распечатка 1. Исходный код реализации модели

```
import java.awt.Container;
import SimJava.*;
public class Model extends Syst implements Runnable {
     Facility pF;                          // Ссылка на канал обслуживания
     Queue pQ;                                          // Ссылка на очередь
     public Model(int tg1, Container con)          // Конструктор класса

     {
          super(tg1,                       // Значение счетчика завершения
                null,                     // Ссылка на файл с результатами
                con);           // Ссылка на контейнер для вывода данных
          th=new Thread(this);          // Создание потока запуска модели
                                              // Создание модельной среды
          pQ=new Queue(this, "Очередь");               // Создаем очередь
          pF=new Facility(this, "Канал");   // Создаем канал обслуживания
     }

     public void run()    // Метод, определяющий функциональность потока
                                                    // (алгоритм модели)
     {
          initGenerate(1, 0); // Транзакт заявка направляется к первому
                                                                 // событию
          while(tg1!=0)                 // Конец моделирования, когда TG1
                                          // (счетчик завершения равен 0)
          {
```

```
            plan();                          // Протяжка модельного времени
            switch(sysEvent)
            {
            case 1:
                generate(v1.randExp(0.1, false)); break;    // Генерация
                                                              // заявок
            case 2:
                pQ.queue(1); break;                  // Занимаем очередь
            case 3:
                pF.seize(); break;         // Занимаем канал обслуживания
            case 4:
                pQ.depart(1); break;               // Освобождаем очередь
            case 5:
                advance(v12.randExp(1./8, false)); break;  // Задержка –
                                                     // время обслуживания
            case 6:
                pF.release(); break;   // Освобождаем канал обслуживания
            case 7:
                terminate(1);                  // Транзакт уничтожается,
                                  // счетчик завершения уменьшается на 1
            }
        }
        printAllScreen();  // Вывод результатов моделирования в контейнер
                                                    // (окно или апплет)
    }

}
```

Распечатка 2. Исходный код апплета

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class MyApplet extends JApplet {
    Model model;
    JButton but=new JButton("Запуск модели");
    public void init()
    {
     setSize(800, 600);
     getContentPane().setLayout(null);
     but.setBounds(10, 10, 200, 20);
     getContentPane().add(but);
        but.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e)
            {
                model=new Model(200000,
                                MyApplet.this.getContentPane());
                model.hy=50; // Координаты y - начало вывода данных
                                                  // в апплете
                model.th.start();           // Запуск потока модели
                try { model.th.join(); } catch(Exception e2) {}
            }

        }
        );
    }

}
```



Внешний вид апплета в браузере с результатами моделирования

Распечатка 3. Результаты, полученные в среде GPSS World

```
GPSS World Simulation Report – Mod2.13.1

Wednesday, September 05, 2012 22:43:43

START TIME END TIME BLOCKS FACILITIES STORAGES
0.000 20016642.235 7 1 0
LABEL LOC BLOCK TYPE ENTRY COUNT CURRENT COUNT RETRY
1 GENERATE 2000003 0 0
2 QUEUE 2000003 2 0
3 SEIZE 2000001 1 0
4 DEPART 2000000 0 0
5 ADVANCE 2000000 0 0
6 RELEASE 2000000 0 0
7 TERMINATE 2000000 0 0

FACILITY ENTRIES UTIL. AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
1 2000001 0.800 8.005 1 2000001 0 0 0 2

QUEUE MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME AVE.(-0) RETRY
1 51 3 2000003 403713 3.047 30.500 38.214 0
```

В заключение отметим, что разработанная библиотека классов языка Java по своим возможностям близка к возможностям языка GPSS, она может быть использована для имитационного моделирования СМО. Эта библиотека классов является начальным этапом разработки «облачных» сервисов для имитационного моделирования на основе языка Java. Библиотеку можно применять для реализации имитационных моделей, выполняемых как на стороне сервера, так и на стороне клиента через Web-браузер.

СПИСОК ЛИТЕРАТУРЫ

1. Кельтон В., Лоу А. Имитационное моделирование. Классика CS. 3-е изд. СПб.: Питер; Киев: Изд. группа BHV, 2004. 847 с.
2. Власов С.А., Девятков В.В., Кобелев Н.Б. Имитационные исследования: от классических технологий до облачных вычислений // Пятая (юбилейная) Всероссийская науч.-практич. конф. по имитационному моделированию и его применению в науке и промышленности «Имитационное моделирование. Теория и практика» ИММОД-2011 (Санкт- Петербург, 2011 г.): сб. докл. Т. 1. С. 42–50.
3. Кудрявцев Е.М. GPSS World. Основы имитационного моделирования различных систем. М.: ДМК Пресс, 2004. 320 с.
4. Марков А.А. Моделирование информационно-вычислительных процессов: учеб. пособие для вузов. М.: Изд-во МГТУ им. Н.Э. Баумана, 1999. 360 с.
5. Медведев Н.В., Быков А.Ю., Гришин Г.А. Имитационное моделирование систем массового обслуживания с использованием межплатформенной библиотеки функций языка Си++ // Вестник МГТУ им. Н.Э. Баумана. Сер. «Приборостроение». 2005. № 4. С. 85–93.
6. Журавлев А.М., Медведев Н.В., Быков А.Ю. Использование межплатформенной библиотеки функций языка Си++ для реализации имитационных моделей типовых систем массового обслуживания // Вестник МГТУ им. Н.Э. Баумана. Сер. «Приборостроение». 2008. № 4. С. 3–15.
7. Ноутон П., Шилдт Г. Java 2. Наиболее полное руководство. СПб.: BHV-Санкт-Петербург, 2007. 1072 с.
8. Клейнрок Л. Теория массового обслуживания. М.: Машиностроение, 1979. 432 с.

Статья поступила в редакцию 25.10.2012