УДК 004.942/.75 : 519.876.5

Ахмад Али (Абдель Карим) Альмхерат, Г.Н. Жолткевич

Харьковский национальный университет им. В.Н. Каразина

ОБ ОДНОЙ МОДЕЛИ МЕХАНИЗМА СПОНТАННЫХ ИЗМЕНЕНИЙ СОСТОЯНИЙ АРХИТЕКТУРНЫХ КОМПОНЕНТОВ ИНФОРМАЦИОННЫХ СИСТЕМ

В статье рассматривается модель механизма, который обеспечивает спонтанное изменение состояния архитектурного компонента информационной системы. В качестве параметра модели выбрана функция, определяющая зависимость вероятности возникновения события перехода от времени нахождения капсулы в текущем состоянии. Для простейшего случая, допускающего точное исследование, проведено сравнение с имитационным экспериментом, которое показало, что квадратичное отклонение экспериментальных данных от теоретических составляет 4,82%.

информационная система, архитектурное проектирование, капсула, спонтанный переход, имитационная модель

Введение

Растущая сложность задач, для решения которых применяются информационно-вычислительные системы, приводит к необходимости интеграции программных комплексов в единое целое. Возникающие в результате такой интеграции системы характеризуются, как правило, распределением данных и их обработки. Существует ряд архитектурных шаблонов, рекомендуемых при проектировании таких систем [1], однако обоснование их применения носит умозрительный, а не научно обоснованный характер. В связи с этим возникает проблема оценки эффективности использования того или иного архитектурного шаблона еще на фазе архитектурного проектирования системы.

Анализ публикаций, постановка задачи. Одной из основных концепций программной инженерии является концепция повторного использования решений на разных уровнях: код, структура, проектное решение, архитектура [2]. Использование шаблонов (паттернов) проектирования при разработке программного обеспечения с момента появления работы [3] стало одним из основных инструментов повышения эффективности разработки за счет повторного использования на уровне архитектурных и проектных решений. Роль шаблонов проектирования как современного средства разработки программного обеспечения отмечена также в [4]. Детально шаблоны проектирования архитектурного уровня для корпоративных программных систем описаны в [5]. В работе [6] предложено архитектурное решение, ориентированное на интеграцию информационных систем на уровне корпорации с сохранением суверенитета каждого из интегрируемых компонентов над своим информационным ресурсом. Общей чертой всех указанных работ является подробное описание различных архитектурных решений при отсутствии рекомендаций относительно методов оценки этих решений с целью обоснованного выбора какого-либо из них.

Учитывая вышеизложенное, можно сделать вывод об актуальности задачи разработки методологии, позволяющей объективно оценивать возможности и последствия использования существующих шаблонных решений для конкретных проектных ситуаций. В работе [7] предложено решать эту задачу путем имитационного моделирования поведения распределенной информационной системы. В указанной работе авторы предлагают описание подхода к имитационному моделированию информационных систем на структурно-логическом уровне. В работе также вводятся основные характеристики, анализ которых по результатам моделирования позволяет

сравнить возможные последствия использования различных архитектурных шаблонов. Однако динамические аспекты каркаса имитационной модели в работе [7] не рассмотрены.

Таким образом, необходимым элементом каркаса имитационной модели является построение и исследование динамики процессов, происходящих в информационных системах.

Модель спонтанного изменения состояний капсул информационной системы

Как и в работе [7] мы следуем архитектурной модели предложенной И. Грэхемом (см. [4]). Напомним, что основным понятием подхода И. Грэхема является капсула информационной системы (КИС), которая представляет собой сложный, возможно распределенный, объект, который взаимодействует со своим окружением. Для построения динамической модели информационной системы на архитектурном уровне предположим, что капсула информационной системы может находиться в одном из нескольких состояний, множество которых будет обозначаться через Q. В процессе функционирования КИС осуществляется смена ее состояний, которая может быть вызвана

– реакцией капсулы на внешнее воздействие;

– спонтанно – в результате внутренних недетерминированных процессов, проходящих в капсуле.

В настоящей работе нас будет интересовать динамика смен состояний КИС второго вида.

Простейшей моделью таких смен состояний является цепь Маркова, однако наблюдения за капсулами реальных информационных систем показывают, что вероятность смены состояния, вообще говоря, зависит от времени пребывания КИС в этом состоянии, т.е.

$$P\left(\xi_t = s' \mid \xi_{t-1} = \xi_{t-2} = \ldots = \xi_{t-k} = s, \xi_{t-k-1} \neq s\right) \neq$$

$$\neq P\left(\xi_t = s' \mid \xi_{t-1} = \xi_{t-2} = \ldots = \xi_{t-l} = s, \xi_{t-l-1} \neq s\right),$$

при $k \neq 1$.

Таким образом, цепь Маркова не является адекватной моделью механизма спонтанного изменения состояния КИС. Для того, чтобы учесть зависимость вероятности спонтанного изменения состояния капсулы от времени пребывания в этом состоянии, с одной стороны, и остаться в рамках предположений, соответствующих наблюдаемым данным, рассмотрим вероятностную модель, базирующуюся на следующих предположениях:

– время пребывания капсулы в состоянии $s \in \mathbf{Q}$ зависит только от этого состояния;

– состояние $s' \in \mathbf{Q}$, в которое переходит капсула из состояния $s \in \mathbf{Q}$, зависит только от этого состояния.

Учитывая сформулированные предположения, предлагается следующая модель спонтанной динамики капсулы.

Рассмотрим $\left\{\tau(s', s'') \mid s', s'' \in \mathbf{Q}, s' \neq s''\right\}$ – семейство положительных случайных величин, которые соответствуют времени разрешения перехода капсулы из состояния $s' \in \mathbf{Q}$ в состояние $s'' \in \mathbf{Q}$;

$\left\{p_X : X \to \square \mid \varnothing \neq X \subset \mathbf{Q},\right.$

$\left.(\forall s \in X)\, p_X(s) \geq 0, \sum_{s \in X} p_X(s) = 1\right\}$ – семейство распределений вероятностей на подмножествах множества Q, индексированное соответствующими подмножествами.

Изменение состояния определяется следующим образом: если капсула в момент времени t_0 находится в состоянии $s \in \mathbf{Q}$, $t(s, s')$ – реализация случайной величины $\tau(s, s')$, $\Delta\tau = \min_{s' \in \mathbf{Q}, s \neq s'} \left\{t(s, s')\right\}$, $X = \left\{s' \in \mathbf{Q} \mid t(s, s') = \Delta\tau\right\}$, то в моменты времени t, удовлетворяющие неравенству $t_0 \leq t < t_0 + \Delta\tau$, капсула остается в состоянии $s \in \mathbf{Q}$, а в момент времени $t = t_0 + \Delta\tau$ происходит переход в состояние $s'' \in X$, которое выбирается случайно в соответствии с распределением p_X.

Таким образом, ключевым компонентом модели является время, необходимое для разрешения перехода из состояния s' в отличное от него состояние s''.

Имитационная модель спонтанного изменения состояний капсул информационной системы

Имитация описанной выше модели строится путем связывания с каждым возможным переходом капсулы из состояния $s' \in \mathbf{Q}$ в состояние $s'' \in \mathbf{Q}$ семафора – случайной системы с двумя состояниями, модель функционирования которой в терминах диаграмм состояний языка UML [8, 9] показана на рис. 1.

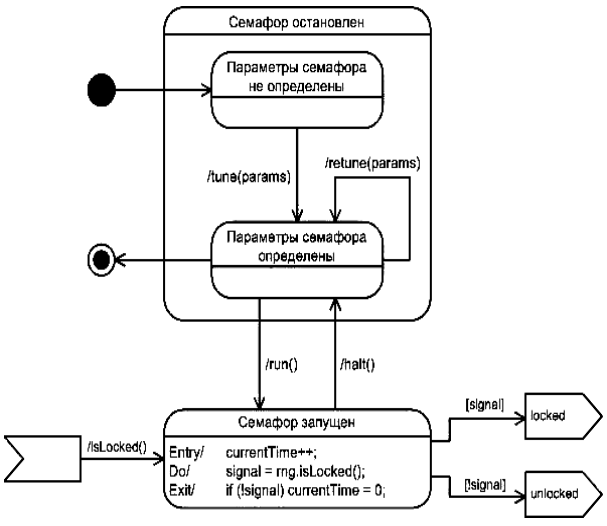

Рис. 1. Модель динамики семафора

Объект `rng`, метод `isLocked( )` которого выполняется во время пребывания семафора в состоянии «семафор запущен», является генератором случайных событий, позволяющих имитировать случайную величину времени необходимого для разрешения перехода семафора из состояния $s' \in \mathbf{Q}$ в состояние $s'' \in \mathbf{Q}$: реализация случайной величины $\tau(s', s'')$ совпадает с количеством обращений к семафору до тех пор, пока выполняется условие `isLocked() == true`.

Легко видеть, что поведение семафора полностью определяется свойствами генератора случайных событий `rng`, точнее, его метода `isLocked( )`, обеспечивающего генерацию случайных событий. Мы предполагаем, что это поведение определяется функцией $g : N \to [0,1]$, значение g(t) которой задает вероятность события **unlocked** при условии, что после предыдущего события **unlocked** было подряд получено t событий **locked**. Тем самым, если последний момент времени, когда было сгенерировано событие **unlocked**, равен t_0 и на протяжении интервала времени $t \geq 0$ после t_0 генерировались только события **locked**, то вероятность генерации события **unlocked** в момент времени $t_0 + t + 1$ равна g(t).

С целью исследования статистических свойств генератора случайных событий для различных функций g(t) авторами была реализована имитационная модель семафора в среде Maple 10. Исходный текст симулятора семафора на встроенном языке системы Maple 10 приведен на рис. 2.

```
Semaphore:= module()
      export
        init,          # инициировать
                       # семафор
        tune,          # настроить семафор
        retune,    # перенастроить
                       # семафор
        run,           # запустить семафор
        halt,          # остановить семафор
        isLocked;  # определить
    # заблокирован ли
                       # семафор
      local
        currentTime,   # показание
    # таймера
    # семафора
        state,         # состояние
    # семафора
        band,              # разрешающая
    # функция
    # семафора
        parms,         # параметры
    # разрешающей
    # функции
        rng;               # генератор
    # случайных
    # событий
      init:= proc()
        state:= halted:
        parms:= []:
        currentTime:= -1:
        rng:= Statistics[
          RandomVariable](
            Uniform(0, 1)):
    # разгон генератора
        Statistics[Sample](rng,
          trunc(1000*time())):
      end proc:
      tune:= proc(g::procedure[
    numeric]())
        local n;
        if evalb(band <> 'band') then
          return "Undefined parameter"
        elif evalb(state = started) then
            return "Semaphore is run"
        end if;
        band:= g:
        n:= nops([op(op(1, g))]):
        if n < 1 then
            return "Invalid function"
        end if;
        if n = 1 then parms:= []:
        else parms:= [op(op(1, g))][2..(n - 1)]:
        end if;
        return "Tune is complete"
      end proc:
      retune:= proc(g::procedure[
      numeric]())
        local n;
        if evalb(band = 'band') then
            return "Undefined parameter"
        elif evalb(state = started) then
            return "Semaphore is run"
        end if;
        band:= g;
        n:= nops([op(op(1, g))]);
        if n < 1 then
            return "Invalid function";
        end if;
        if n = 1 then parms:= []:
        else parms:= [op(op(1, g))][
    2..(n - 1)]:
        end if;
        return "Retune is complete"
      end proc:
      run:= proc()
        if evalb(band = 'band') then
            return "Parameter undefined"
        elif evalb(state = started) then
            return "Semaphore is running"
        end if;
        state:= started;
        return "Semaphore is run"
      end proc:
      halt:= proc()
        if evalb(state = halted) then
            return "Semaphore is halted"
        end if;
        state:= halted;
        currentTime:= 0;
        return "Semaphore is halted"
      end proc:
      isLocked:= proc()
        if evalb(state = halted) then
            return " Semaphore is halted"
        elif nargs <> nops(parms) then
            return "Invalid parameters"
        end if;
        currentTime:= currentTime + 1:
        if evalb(Statistics[Sample](
      rng, 1)[1] <
              evalf(band(currentTime,
    op(args)))) then
            currentTime:= 0:
            return false
        end if;
        return true
      end proc:
    end module:
```

Рис. 2. Исходный текст симулятора семафора на встроенном языке системы Maple 10

Результаты имитационного эксперимента

С целью проверки правильности реализации симулятора семафора был поставлен имитационный эксперимент для случая $g(t) = p$, где p – константа из диапазона $(0,1]$.

В этом случае вероятность открытия семафора точно в момент времени $t \in \square_0$, очевидно, равна $(1-p)^t p$ и, следовательно, среднее время пребывания семафора в закрытом состоянии задается формулой:

$$\mathbf{M}\tau = \sum_{t=0}^{+\infty} t(1-p)^t p . \qquad (1)$$

Если обозначить $q = 1 - p$, то формула (1) переписывается в виде:

$$\mathbf{M}\tau = p\left(\frac{d}{dq}\left(\sum_{t=0}^{+\infty} q^t\right) - \sum_{t=0}^{+\infty} q^t\right) = p\left(\frac{1}{(1-q)^2} - \frac{1}{1-q}\right). \qquad (2)$$

Возвращаясь к исходным обозначениям в формуле (2), получим

$$\mathbf{M}\tau = p\left(1/p^2 - 1/p\right) = (1-p)/p . \qquad (3)$$

Анализ метода `isLocked()` показывает: при $p = 0$ симулятор семафора в состояние «открыт» не перейдет, что соответствует формуле (3).

Имитационный эксперимент для $p = 0,1; 0,2; \ldots; 0,9; 1,0$ дает при 500-ах прогонах для каждого указанного выше p, эмпирические значения среднего времени нахождения семафора в закрытом состоянии (€), приведенные в табл. 1.

Таблица 1

Сравнение результатов имитационного эксперимента с точными значениями

№ эксп.	Значения p	Значения €	Значения **M**τ	Квадрат. отклон.
1	0,0	+∞	+∞	0,0
2	0,1	8,550	9,000	0,203
3	0,2	4,036	4,000	0,001
4	0,3	2,226	2,333	0,012
5	0,4	1,378	1,500	0,015
6	0,5	0,988	1,000	0,000
7	0,6	0,674	0,667	0,000
8	0,7	0,390	0,428	0,001
9	0,8	0,240	0,250	0,000
10	0,9	0,090	0,111	0,000
11	1,0	0,000	0,000	0,000
Среднеквадратичное отклонение:				**4,82%**

Данные из табл. 1 отображены на рис. 2.

Полученные результаты позволяют сделать вывод об адекватности построенной имитационной модели предложенной выше математической модели.

Значительный интерес представляет изучение характеристик, связанных с интенсивностью срабатываний семафора в зависимости от функционального параметра модели.

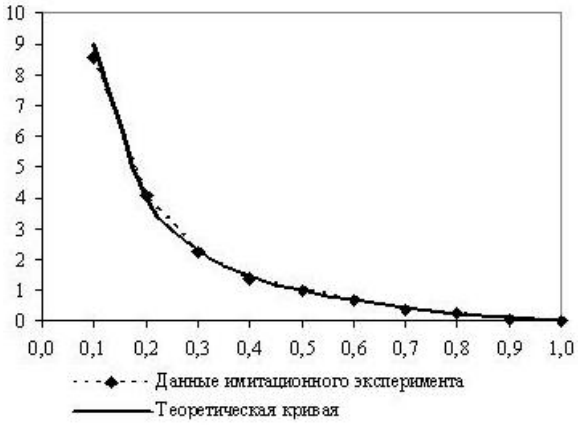

Рис. 3. Сравнение результатов имитационного эксперимента и теоретических данных

Выводы

В настоящей работе предложена модель механизма, обеспечивающего спонтанное изменение состояния архитектурного компонента (капсулы) информационной системы. Модель специфицирована в терминах иерархических диаграмм состояний/переходов языка UML.

Параметром предложенной модели является функция, определяющая зависимость вероятности возникновения события перехода от времени нахождения капсулы в текущем состоянии. Поскольку аналитическое исследование характеристик модели для нетривиальной параметризации вряд ли возможно, разработана имитационная модель семафора, реализующего предложенный механизм спонтанных переходов.

С целью верификации модели для простейшей параметризации, допускающей точное исследование, проведено сравнение теоретических данных с результатами имитационного эксперимента, которое показало, что квадратичное отклонение экспериментальных данных от теоретических составляет 4,82%. Последнее подтверждает достоверность имитационной модели.

Дальнейшие исследования предполагается посвятить внедрению описанного механизма имитации спонтанного перехода состояния капсулы в имитационную модель архитектурного решения информационной системы.

Список литературы

1. Таненбаум Э., ван Стен М. Распределенные системы. Принципы и парадигмы. – С.-Пб.: Питер, 2003. – 877 с.

2. Соммервилл И. Инженерия программного обеспечения. – М.: Вильямс, 2002. – 624 с.

3. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гемма, Р. Хелм и др. – С.-Пб.: Питер, 2003. – 366 с.

4. Грэхем И. Объектно-ориентированные методы. Принципы и практика. – М.: Вильямс, 2004. – 879 с.

5. Фаулер М. Архитектура корпоративных программных приложений. – М.: Вильямс, 2004. – 544 с.

6. Альмхерат Ахмад Али (Абдель Карим), Жолткевич Г.Н., Игнатов С.Ю. Об одном подходе к интеграции неоднородных информационных ресурсов // Вісник Харк. нац. ун-та. – 2004: – № 629. Математичне моделювання. Інформаційні технології. Автоматизовані системи управління. – Вип. 3. – С. 48-55.

7. Альмхерат Ахмад Али (Абдель Карим), Жолткевич Г.Н., Жолткевич А.Г. Моделирование обмена информацией в информационных системах: концептуальный уровень // Вестник НТУ «ХПИ». – 2006: – № 19. Системный анализ, управление и информационные технологи. – С. 57-61.

8. Unified Modeling Language Specification: An Adopted Formal Specification / Object Management Group, Inc. – Version 1.5. – Needham, MA, 2003. – 736 p.

9. Unified Modeling Language: Superstructure / Object Management Group, Inc. – Version 2.0. – Needham, MA, 2005. – 709 p.

Поступила в редколлегию 15.12.2006

Рецензент: д-р техн. наук, проф. Л.Г. Раскин, Национальный технический университет «ХПИ», Харьков.