Российская академия наук
Национальный комитет по автоматическому управлению
Научный совет по теории управляемых процессов и автоматизации
ОЭММПУ РАН
Институт проблем управления им. В.А. Трапезникова РАН
Министерство образования и науки Удмуртской Республики
Удмуртский государственный университет
Удмуртский НОЦ ПУ (на базе УдГУ)
Волгоградский НОЦ ПУ (на базе ВолГУ)
Воронежский НОЦ ПУ (на базе ВГАСУ)
Инновационный НОЦ ПУ (на базе МАИ)
Казанский НОЦ ПУ (на базе КГТУ-КАИ)
Курский НОЦ ПУ (на базе КГТУ)
Липецкий НОЦ ПУ (на базе ЛГТУ)
НОЦ «Системный анализ в управлении» (на базе МИФИ)
Пермский НОЦ ПУ (на базе ПГТУ)
Самарский НОЦ ПУ (на базе СГАУ)
Тверской НОЦ ПУ (на базе ТГТУ)
Старооскольский НОЦ ПУ (на базе СТИ)
Удмуртский Совет ИТ-директоров
Институт логики, когнитологии и развития личности РАН

VI Всероссийская школа-семинар молодых ученых

Управление большими системами

посвящается памяти А.А. Маркова

Том 1

31 августа – 5 сентября 2009
г. Ижевск

УДК 007

VI Всероссийская школа-семинар молодых ученых «Управление большими системами»: Сборник трудов. – Т1.- Ижевск: ООО Информационно-издательский центр «Бон Анца», 2009. – 400 с.

В первый том сборника трудов включены научные статьи молодых ученых по фундаментальным основам теории управления, вопросам управления организационными и социально-экономическими системами, управлению качеством.

ISBN 978-5-903140-57-2

Научное издание осуществлено при поддержке РФФИ
грант № 09-07-06039г

© Авторы, постатейно, 2009
©ООО ИИЦ «Бон Анца»
(оформление обложки, верстка)

УДК 004.94

Использование системы событийного программирования SEvents для моделирования сложных систем

Биков Р.М.
Удмуртский государственный университет, г. Ижевск

Аннотация

Описываются возможности системы событийного программирования **SEvents** для моделирования взаимодействий в сложных системах.

Abstract

A system of event programming **SEvents** abilities are described here.

Введение

Определение 1

Событийное программирование — стиль программирования, в котором вычисления активизируются в результате какого-то события, являются реакцией на это событие, изменяя характеристики системы [1].

Данный стиль используется при программировании обработки датчиков, при написании графического интерфейса, при программировании операционных систем, практически везде, где требуется обработка определённых ситуаций. Но в каждой системе круг данных ситуаций, которые могут называться событиями, обычно ограничен потребностями системы.

Однако сама концепция событийного программирования намного шире. На самом деле ничто нам не мешает определить событием ту ситуацию, которая нам интересна. К примеру, мы можем захотеть назвать событием ситуацию, когда значение одной переменной станет больше значения другой, что в универсальных языках программирования является невозможным.

Система событийного программирования ***SEvents***[1] лишена данного недостатка, поэтому может быть использована для моделирования взаимодействия процессов в сложной системе.

1. Описание системы

SEvents – система событийного программирования с фактором случайности. Основное понятие данной системы – это событие. У каждого события есть своё имя и условие выполнения.

---

[1] Систему событийного программирования можно найти по адресу http://sourceforge.net/projects/sevents

Определение 2

Событие – это *информационный объект, говорящий о том, что в системе произошло что-то важное, на что стоит обратить внимание.*

Событие обычно активизирует какое-либо вычисление, поэтому логично связывать событие с его обработчиком, что и делается в **SEvents.** Так как одновременно может произойти несколько событий, то и обработчики этих событий выполняются параллельно. Но как быть, если у нас ограниченное количество процессоров, на которых могут выполняться обработчики наших событий? Какие события должны выполниться в данный момент, прежде всего, а какие могут и подождать? В **SEvents** данная проблема решается при помощи приоритетов событий. Соответственно, чем выше приоритет события, тем больше вероятность того, что обработчик данного события выполнится именно тогда, когда нужно.

Однако какая система бывает без ошибок? Где гарантия, что события (соответственно, и их обработчики, которые фактически являются процессами) будут активизироваться всегда, когда мы захотим, и именно в том порядке, в котором мы хотим? Для моделирования необязательности исполнения процессов в **SEvents** предусмотрена вероятность выполнения события (у каждого события своя). Механизм приоритетов – хорошее средство регулирования порядка выполнения обработчиков событий, однако почему мы обязаны их соблюдать? В **SEvents** обработчики событий с максимально возможным приоритетом выполняются гарантированно раньше остальных, обработчики событий с минимально возможным приоритетом выполняются гарантированно позже других, для обработчиков событий с промежуточными приоритетами есть какая-то вероятность (*DISORDER_PROBABILITY* опционально изменяемая константа системы) несоблюдения одним из событий своего приоритета. Обработчики событий с равным приоритетом выполняются недетерменированно.

Бывает необходимым назначить несколько обработчиков для одного и того же события. Но на самом деле мы можем к каждому событию прикрепить всего один обработчик, создать метасобытие (чей обработчик может быть пустым), а условием выполнения других нужных событий поставить условие активации нашего метасобытия. Также в данной ситуации можно отказаться от условных операторов (любое логическое выражение является условием выполнения события) и от циклов (одно событие может повторяться множество раз, притом его выполнение может зависеть от значений переменных).

Таким образом, в **SEvents** события обладают следующими атрибутами:

1) имя события (чтобы знать, что именно произошло, чтобы другие события могли происходить во время, когда исполняется данное событие, до него либо после);

2) условие выполнения (собственно, какую ситуацию мы хотим обозначить событием, здесь может быть проверка значений переменных, проверка выполнения других событий, вызов функций);

3) флаг возобновимости (не все события могут повторяться: человек рождается только один раз);

4) вероятность выполнения события (молния не всегда ударяет в самое высокое дерево, человек не всегда выполняет свои обещания);

5) приоритет (определяет порядок важности события для вычислительной системы);

6) тело события (тело обработчика, которое не содержит условных операторов и циклов).

Пример 1. (Описание события)

```
1. event
2.   name = "1-й игрок идёт вправо"
3.   cond = isRun("Изменение координат 1-го игрока")
     && x1 < x;
4.   repeatable = true;
5.   probability = 1;
6.   priority = 3;
7. body
8.   x1 = x1 + 1;
9.   min = 60;
10. end.
```

Пояснение.

В строке 1 написано ключевое слово event, *с которого начинается описание события. Далее идёт заголовок события — его атрибуты. Событие называется* «1-й игрок идёт вправо», *условием выполнения события является случай, когда исполняется событие с именем* «Изменение координат 1-го игрока» *и значение переменной* x1 *меньше значения переменной* x. *Также событие является возобновимым (строка 4), вероятность выполнения события -100% (строка 5), приоритет события равен 3. В строке 7 стоит ключевое слово* body, *с которого начинается описание обработчика события – линейной последовательности операторов. Описание обработчика заканчивается ключевым словом* end *в строке 10.*

Событие с данными атрибутами является основой системы **SEvents**, подобно кирпичу или камню, из которых можно построить любое здание, из этих событий можно описать сколь угодно сложное взаимодействие различных процессов большой системы.

Процессы процессами, но как же пользователь узнает о том, что у него происходит в системе? Для этой задачи в **SEvents** реализовано полное журналирование всех изменений. Однако следить за состоянием системы по быстро заполняющемуся журналу не очень удобно... Ещё один из способов оповещения пользователя об изменениях в системе являются *объекты*, способные реагировать на события, изменяя своё состояние либо состояние системы. В принципе, объектом может быть что угодно: курсор мыши на экране монитора, фары автомобиля, загорающиеся в темноте, марш Мендельсона, заполняющий свадьбу, и многое другое...

Для унификации взаимодействия системы с объектами последние в **SEvents** представлены в виде динамической библиотеки, которая может быть написана практически на любом языке программирования и отвечать за что-то своё.

Для связи между объектами и событиями, на которые они могут реагировать, в **SEvents** используется общее пространство переменных. Собственно, все переменные и константы являются глобальными, и к каждой переменной как объекты, так и обработчики событий имеют равный доступ (всё происходит в одном мире, каждый объект волен делать то, что хочет, события могут иметь неожиданные последствия). Единственная переменная, к которой не имеют доступ ни события, ни объекты, – переменная времени.

Переменная времени *time* считается константой для событий и объектов системы, однако её значение постоянно увеличивается системой. В начале исполнения сценария значение этой переменной равно 0. Сценарий исполнения событий системы останавливается тогда, когда переменная времени достигнет своего максимального значения (*MAX_TIME*, значение изменяется опционально в настройках) или когда системной переменной *terminate* обработчиком события либо каким-либо объектом будет присвоено истинное значение.

В каждый момент времени система проверяет на истинность условия выполнения событий. Если условие истинно и событие активно, то это событие с некоторой вероятностью переносится в список кандидатов на исполнение. Из списка кандидатов на исполнение выбираются события с максимальным приоритетом и ставятся в очередь на исполнение. Из очереди на исполнение события выбираются случайным образом, и в том случае, если для текущего события находится свободный процессор в дан-

ный момент времени (количество процессоров системы *PROC_NUMBER* задаётся опционально), выполняется следующая невыполненная строка обработчика данного события. И так происходит до тех пор, пока список кандидатов на исполнение не окажется пустым.

После исполнения событий в данный момент времени последовательно вызываются на исполнение модули объектов системы. В случае если обработчик событий либо объект системы изменяет значение какой-либо переменной, то данное значение с некоторой вероятностью *(UPDATE_PROBABILITY,* значение изменяется опционально) обновляется мгновенно, в остальных случаях значения переменных обновляются в конце тика времени.

Пояснение.

Например, если значение переменной x *в начале тика времени равно 0 и два обработчика событий захотели одновременно изменить её значение (одновременно должны выполниться две строки различных обработчиков:* «x = 5» *и* «x = x + 1»*), то значение этой переменной в конце тика времени может быть равно 5 (если сначала выполнилась вторая строка, а потом выполнилась первая), 1 (если сначала выполнилась первая строка, но значение* x *не обновилось, а потом выполнилась вторая строка), 6 (если выполнилась первая строка, значение* x *обновилось мгновенно, а затем выполнилась вторая строка).*

Таким образом, система событийного программирования **SEvents** позволяет моделировать нежелательные эффекты параллелизма, что может использоваться в обучающих целях.

Заключение

Система событийного программирования **SEvents** с фактором случайности как нельзя кстати подходит для моделирования взаимодействия различных процессов в сложной системе благодаря своей универсальности и широкому простору для расширений. Однако эта система создана для обучения студентов событийному программированию, и её задача состоит не в том, чтобы предохранять программиста от ошибок, а в том, чтобы помочь ему разобраться в своих ошибках. Система написана на языке Java, исходные коды программы открыты.

Список литературы

1. *Непейвода Н.Н.* Стили и методы программирования / Интернет-Университет информационных технологий. – М., 2005.

Предметный указатель

Константа
    DISORDER_PROBABILITY
    MAX_TIME
    PROC_NUMBER
    UPDATE_PROBABILITY
Объект
Обработчик
Определение
    Событие
    Событийное программирование
Переменная
    terminate
    time
Пример
    Описание события (1)
Приоритет
SEvents