Решение на задачата за разпределено взаимно изключване чрез проектен шаблон

Милен Луканчевски, Николай Костадинов, Хованес Авакян

Solution of Mutual Exclusion Problem with Software Design Pattern: *The contact center solution eMOSys.6K, developed by the authors, is typical distributed system. Such a system consists from multiple concurrent asynchronous processes communicating only by passing messages. That processes share a collection of resources, then mutual exclusion is required to ensure consistency when accessing shared resources. The processes are formalized as Finite State Machines where each state accumulates the process history. State transitions are reaction of events which determines the reactive nature of distributed systems. Events are processed by appropriate handlers. The distributed algorithm is described by pseudo-code which corresponds to reactive nature of the system. Hence a solution of distributed mutual exclusion problem based on design pattern of FSM could be given.*

Key words: *Contact Centers, Design Patterns, Distributed Mutual Exclusion, Distributed Systems, FSM, State Transition Diagram.*

ВЪВЕДЕНИЕ

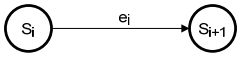
През 1999 г, авторите получиха запитване от БТК ЕАД за възможността да се проектира контактен център, с който да се подменят както шнуровите номератори, така и електронния номератор на Siemens от типа ADMOSS. В резултат на направеното в периода от 1999 до 2000 г. проучване, бе направено предложение за проектирането и изграждането на контактен център с разпределена архитектура, базиран на VoIP комутация. След положителния отговор от страна на БТК ЕАД, авторите пристъпиха към проектирането на първата версия eMOSys.4K на контактния център. В рамките на 2001 г. тази версия се изпитва на територията на РУД Бургас, а през лятото на 2002 г. бе официално приета от БТК ЕАД, с което стана първия български електронен операторски център. От тогава до момента центърът непрекъснато се усъвършенства главно с добавянето на нова функционалност. Текущата версия eMOSys.6K е цялостно решение за средноголеми и големи контактни центрове с до няколкостотин работни станции и десетки хиляди повиквания на ден.

В основата на успеха на контактния център от типа eMOSys е заложения още в самото начало разпределен модел, използването на теорията на разпределените системи, както и методологията на гъвкавото (Agile) програмиране на всички етапи от развитието на центъра.

Докладът е посветен на една от многото задачи, решавани при проектирането на центъра – осигуряването на съгласуван достъп до общите поделяни ресурси в разпределена среда. За повишаване надеждността на решението на тази подзадача то трябва да позволява формализираното описание на алгоритъма да се вгражда в кода.

РАЗПРЕДЕЛЕН ИЗЧИСЛИТЕЛЕН МОДЕЛ

Разпределените системи се състоят от множество асинхронни конкурентни процеси, които взаимодействат само чрез обмен на съобщения по канали. Процесите поделят общи ресурси, откъдето следва необходимостта от взаимното им изключване при достъпа до тези ресурси.

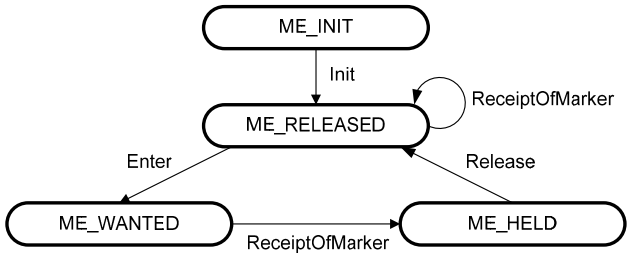


Фиг. 1. Преход между състоянията под въздействието на събитие

В хода на изпълнението си даден процес преминава през множество предварително дефинирани състояния. Преходите между две състояния се извършва под въздействието на определено събитие (фиг. 1), откъдето следва и реактивния характер на процесите в разпределената система. Самото събитие представлява или изпращане на съобщение (*send*), или приемане на съобщение (*receive*), или вътрешно за процеса действие [1].

КРЪГОВ АЛГОРИТЪМ ЗА РАЗПРЕДЕЛЕНО ВЗАИМНО ИЗКЛЮЧВАНЕ

Взаимното изключване е известно като метод за съгласуване достъпа до поделени ресурси от многозадачните операционни системи. Там методът се основава на обектите на ядрото: семафори, пощенски кутии, мютекси и др. При разпределените системи не са достъпни подобни централизирани обекти, поради което разпределеното взаимно изключване по правило се основава на критични секции.



Фиг. 2. Диаграма на преходите и състоянията

Основните алгоритми за разпределено взаимно изключване са алгоритъма с централен сървър, кръговия алгоритъм и алгоритъма на Ricart-Agrawala [1, 4]. Последните два са с разпределено управление и представляват особен интерес.

Като пример в доклада се разглежда кръговия алгоритъм, чиято диаграма на преходите и състоянията е показана на фиг. 2.

Подобно е разглеждането и при останалите алгоритми.

ПРОЕКТЕН ШАБЛОН

Използването на проектни шаблони е част от съвременните програмни технологии като например *екстремалното* (XP) програмиране, част от методологията на *гъвкавото* (Agile) програмиране [5]. Проектните шаблони въвеждат следващо ниво на абстракция и осигуряват общо и гъвкаво решение за даден клас задачи [2, 3]. Различават се три основни групи проектни шаблони – *за създаване* на обекти, *структурни* и *поведенчески*.

Поведенческите проектни шаблони от типа *състояния* отговарят на реактивния характер на процесите в разпределените системи и служат за представяне на машина на състоянията [2, 3].

В настоящата работа се предлага методика за използване на проектни шаблони на машина на състоянията. Предлаганата методика включва три етапа.

На първия етап се разглежда проектния шаблон на машината на състоянията (фиг. 3). Шаблонът включва декларациите на множеството на състоянията, променливата на състоянието (която по правило е проста, но може да бъде и съставна), функцията на преходите, множеството на събитията и манипулаторите на събитията.

```
[множество състояния]
<System States> := {<System State 0>, <System State 1>, …, <System State k>, …}

[променлива на състоянието]
<State Variable>

[функция на преходите]
<result> StateTransitionTo <new state> where <new state> is part of <System States>

[множество на събитията]
<System Events>

[манипулатори на събитията]
On <event> where <event> is part of <System Events>
  If StateTransition <new state>
        {обработка на събитието <event>}
```

Фиг. 3. Проектен шаблон

На фиг. 4 е представена и другата част от проектния шаблон – дефиницията на функцията на преходите.

```
[псевдокод на функцията на преходите]
<result> StateTransitionTo <new state> where <new state> is part of <System States>
{
        result := false
        switch <State Variable>
        {
                [дефиниция на преходите за всяко едно системно състояние]
                case <System State 0>:
                        switch <new state>
                                [обработка на разрешените преходи от <System State 0>]
                case <System State 1>:
                        switch <new state>
                                [обработка на разрешените преходи от <System State 1>]
                …
                case <System State k>:
                        switch <new state>
                        {
                                [обработка на разрешените преходи от <System State k>]
                                case <System State i>:
                                        {set <System State i> context}
                                        result := true
                                case <System State j>:
                                        {set <System State j> context}
                                        result := true
                                …
                        }
                …
        }
        If <result> = true
                <State Variable> := <new state>
        Return result
}
```

Фиг. 4. Псевдокод на функцията на преходите от проектния шаблон

На втория етап проектният шаблон се параметризира с формализираното описание на алгоритъма (фиг. 5). Множеството на състоянията и множеството на преходите отговарят на съответната машина на състоянията. В случая се използва

машината, отговаряща на кръговия алгоритъм за разпределено взаимно изключване от фиг. 2.

Достатъчна е проста променлива на състоянието, чието съдържание определя текущото местоположение на процеса в диаграмата на преходите и състоянията.

Предвидени са четири манипулатора на събитията – по един за всяко събитие, съответно преход. Самият преход се изършва от функцията на преходите, която проверява допустимостта му.

```
[Множество състояния]
<System States> := {ME_INIT, ME_RELEASED, ME_WANTED, ME_HELD}

[Променлива на състоянието]
<State Variable> := meStatus

[Функция на преходите]
<result> StateTransitionTo <new state> where <new state> is part of <System States>

[Множество на събитията]
<System Events> := {<Init>, <Enter>, <ReceiptOfMarker>, <Release>}

[Манипулатори на събитията]
On <Init>
  If StateTransition <new state>
        {обработка на събитието <Init>}

On <Enter>
  If StateTransition <new state>
        {обработка на събитието <Enter>}

On <ReceiptOfMarker>
  If StateTransition <new state>
        {обработка на събитието <ReceiptOfMarker>}

On <Release>
  If StateTransition <new state>
        {обработка на събитието <Release>}
```

Фиг. 5. Параметризиран шаблон

Поради ограниченото място в това изложение, параметризираната функция на преходите се представя само в презентацията към доклада.

На третия етап параметризираният шаблон се кодира и така формализираната схема на алгоритъма се вгражда в програмната му реализация. Важна особеност на предложеното решение е съобразяването на структурата и съдържанието на шаблона със събитийния характер на съвременните програмни системи. Така, програмният код е непосредствена реализация на схемата, съдържаща се в параметризирания шаблон, получен в края на втория етап.

ЗАКЛЮЧЕНИЕ

Докладът е посветен на една от многото задачи, възникнали в хода на проектирането на контактния център eMOSys – осигуряването на съгласуван достъп на процесите до общите поделяни ресурси в разпределена среда. Описаното решение намира приложение в основните подсистеми на контактния център: разпределения логически комутатор на разговорните канали, модула за разпределяне на входящите повиквания, модула за гласова поща, телемаркетинг модула и т.н.

Използването на машина на състоянията при формализираното описание на алгоритъма за разпределено взаимно изключване отговаря в максимална степен на реактивния характер на процесите в разпределените системи.

Предложеното решение е достатъчно общо и едновременно с това позволява почти директното интегриране в програмния код. Възможността формализираната схема да се вгради практически непосредствено в кода гарантира надеждността на решението.

ЛИТЕРАТУРА

[1] Coulouris, G., J. Dollimore, Tim Kindberg. Distributed Systems: Concepts and Design. New York: Addison-Wesley, 2000.

[2] Douglass, B. Design Patterns for Embedded Systems in C. London: Elsevier Inc., 2011.

[3] Gamma, E., et al. Design Patterns: Elements of Reusable Object-Oriented Software. New York: Addison-Wesley, 2009.

[4] Kshemkalyani, A., M. Singhal. Distributed Computing: Principles, Algorithms, and Systems. New York: Cambridge University Press, 2008.

[5] Martin, R. Agile Software Development, Principles, Patterns, and Practices. New Jersey: Prentice Hall, 2002.

За контакти:

гл.ас. д-р Милен Луканчевски, Катедра “Компютърни системи и технологии”, Русенски университет “Ангел Кънчев”, тел.: 0887 303 850, e-mail: mil@ieee.org

гл.ас. Николай Костадинов, Катедра “Компютърни системи и технологии”, Русенски университет “Ангел Кънчев”, тел.: 082 888 674, e-mail: nkostadinov@ecs.uni-ruse.bg

гл.ас. Хованес Авакян, Катедра “Компютърни системи и технологии”, Русенски университет “Ангел Кънчев”, тел.: 082 888 674, e-mail: havakian@ecs.uni-ruse.bg

Докладът е рецензиран.

РУСЕНСКИ УНИВЕРСИТЕТ „АНГЕЛ КЪНЧЕВ”
UNIVERSITY OF RUSE „ANGEL KANCHEV“

Д И П Л О М А

Програмният комитет на
Научната конференция РУ&СУ’11
награждава с КРИСТАЛЕН ПРИЗ
“THE BEST PAPER”
д-р МИЛЕН ЛУКАНЧЕВСКИ,
НИКОЛАЙ КОСТАДИНОВ, ХОВАНЕС АВАКЯН
автори на доклада
“Решение на задачата за разпределено взаимно изключване чрез проектен шаблон ”

D I P L O M A

The Programme Committee of
the Scientific Conference RU&SU'11
Awards the Crystal Prize
"THE BEST PAPER"
to MILEN LUKANCHEVSKI, PhD
NIKOLAY KOSTADINOV, HOVANES AVAKYAN
authors of the paper
“Solution of Mutual Exclusion Problem with Software Design Pattern”

РЕКТОР **доц. дтн Христо Белоев**
RECTOR **Prof. DSc Hristo Beloev**

29.10.2011