КОНКРЕТИЗИРУЕМЫЕ СЕТЕВЫЕ МОДЕЛИ ЗАДАЧ ДЛЯ ДИСТАНЦИОННОГО ИЗУЧЕНИЯ ОБЪЕКТНЫХ ТЕХНОЛОГИЙ

М.П. Ревотюк, Т.В. Тиханович, В.В. Зобов

Белорусский государственный университет информатики и радиоэлектроники, Минск, Беларусь, rmp@bsuir.by

Abstract. Template of abstract successor's class for service control systems design, based on network model, was presented.

Характерное для дистанционной формы образования малое количество регламентируемых точек контроля и контакта с преподавателем при неудачном выборе учебных задач может стать причиной дискредитации потенциальных возможностей современных технологий объектно-ориентированного программирования и проектирования. Формулировка таких задач должна отражать ориентацию объектных технологий на поддержку инкрементальной итерационной разработки открытых для расширения прикладных систем.

Предмет рассмотрения – опыт формулировки учебных задач для решения в рамках индивидуальных и практических занятий и курсового проектирования на примере задач реализации управляющей части систем с дискретным характером поведения и императивным управлением. Цель работы – построение шаблона проектирования, реализующего принципы GRASP(General Responsibility Assignment Software Patterns) объектно-ориентированного моделирования и анализа [1] применительно к задачам координации взаимодействия в иерархических системах.

Анализ принципов построения шаблонов проектирования GRASP показывает, что такие шаблоны, ориентированные на этап программной реализации системы, можно использовать для конкретизируемой спецификации задач управления с целью анализа и обучения. Для этого применим известный в объектных технологиях прием построения шаблона интерпретации [1] моделей спецификации решаемых задач.

Технологии разработки программного обеспечения опираются на спецификацию задачи в терминах конечных автоматов или сетей Петри. Известно, что формального различия между этими формами спецификации нет.

Конечный автомат – это пятерка $FSM = (Q, \Sigma, \Delta, \delta, \Gamma)$, где Q – конечное множество состояний; Σ – конечный входной алфавит; Δ – конечный выходной алфавит; $\delta : Q \times \Sigma \to Q$ – функция отображения текущих состояния и входа в следующее состояние; $\Gamma : Q \times \Sigma \to \Delta$ – функция отображения текущих состояния и входа в выходной символ.

Сеть Петри – четверка $PN = (P, T, I, O)$, где P – конечное множество позиций, $|P| \geq 0$; T – конечное множество переходов, $|T| \geq 0$, $P \cap T = \varnothing$; $I : T \to P^{\infty}$, $O : T \to P^{\infty}$ – функции отображения переходов в комплекты позиций.

Соответствие любого автомата сети Петри задается следующим образом [2]:

$$P = Q \cup \Sigma \cup \Delta, \quad T = \{t_{q,\sigma} \mid q \in Q, \ \sigma \in \Sigma\}, I(t_{q,\sigma}) = \{q, \sigma\}, \ O(t_{q,\sigma}) = \{\delta(q,\sigma), \Gamma(q,\sigma)\}.$$

Однако структурное определение как FSM, так и PN являются лишь статической моделью разрабатываемой системы. В качестве атомарных моделей дискретных процессов в технологическом отношении более практичны переходные системы вида "условие – действие", определяемые тройками множеств

$S_c = \langle V_c, P_c, A_c \rangle$, где V_c – переменные состояния; P_c – условия на V_c, задаваемые вычислимыми предикатами; A_c – действия, биективно соответствующих условиям P_c.

Реализация принципа детализации и конкретизации описания в технологии объектно-ориентированного программирования (ООП), как известно, основана на модулях-функциях языка программирования. Определяя базовые классы в корне иерархии, удобно использовать расширения двух известных вариантов аппарата описания – временных сети Петри с задержками в переходах *TPN* (Timed Petri Nets) и систем переходов, определяемых средствами языка C++ [2].

Рассмотрим далее основные принципы конструирования сетевых моделей в шаблонах проектирования систем обслуживания. Шаблоном представления как однородных, так и иерархических и вложенных систем переходов может быть пятерка функций $S_c = (E_i, C_i, D_i, F_i, I_i)$, где E_i – условия активизации перехода; C_i – действия, вызываемые в системе при входе перехода в активное состояние; D_i – длительность во времени активной фазы; F_i – действие перехода при выходе из активного состояния; I_i – действия над переходом при внешнем прерывании. Индекс i соответствует экземпляру перехода.

Процесс активизации отдельного перехода, подобно переходам сетей Петри, естественно связать с его восприимчивостью к изменению локальных переменных состояния $V_i = dom(E_i) \cup dom(C_i) \cup dom(F_i) \cup dom(I_i)$. Здесь $dom(f)$ – множество переменных состояния, связываемых функцией f. Такое множество образуют глобальные переменные модуля программы, реализующего функцию f. В случае, когда f – функция-элемент класса, $dom(f)$ может включать элементы данных класса.

Фаза использования модулей-функций языков процедурного типа является элементарным автоматным переходом, что соответствует концепции GRASP. На переходах системы S_c можно формально построить однодольную сеть $IPN = (A, B, V)$, связывающую переходы A с потенциальной возможностью активизации. Здесь B – переменные состояния – аналог позиций расширенных сетей Петри, а $V = \{A \times A\} \to \{0,1\}$. Обозначим $'x$ и x' – множество входных и выходных элементов любой вершины x ориентированного графа, соответственно. Структура смежности графа сети IPN определяется в виде *FSF*(Forward Star Form) как $FSF = \{a : a'\}$, где $a' = \{x \mid dom(E_a) \cap (dom(F_x) \cup dom(I_x)), x \in A\}$, $a \in A$.

Определение графа сети IPN может быть использовано как для оптимизации управления, так и моделирования активности переходов – интерпретации процессов на сети. Рекуррентная схема интерпретации процессов на сети IPN построена на основе понятий виртуального стартового перехода s и виртуального финишного перехода f, для которых справедливо

$$\begin{cases} 's = \varnothing, \ s' = \{a \mid dom(E_a) \cap dom(F_s) \neq \varnothing\}, \ D_s = 0; \\ f' = \varnothing, \ D_f = 0; \ A \leftarrow A \cup \{s\} \cup \{f\}. \end{cases}$$

Изменение состояния IPN при этом оказывается однозначно привязанным к моментам выхода переходов из активного состояния. Это влечет не только эффективную реализацию схемы отражения последствий изменения состояния сети, но и позволяет рассматривать внешние события в реальном времени как переходы

специального вида [3]. Состояние сети на любом этапе k представляется списком событий I_k, элементы которого – момент времени t_k и номер перехода $a \in A$:

$$\begin{cases} I_0 = \{(0, s)\}; \\ I_k = \{(t_i, a),\ i > k-1,\ a \in A\},\ k > 0. \end{cases}$$

Показано, что построенный подобным образом полиморфный класс интерпретации системы переходов обладает не только возможностями полиморфизма операторов изменения разметки сети, но позволяет задать закон управления последовательностью событий на описании графа *IPN* [3,4]. Способ задания такого закона и для *IPN* базируется на детализации графа связей переходов, составляющих множество $dom(E_i) \cup dom(C_i) \cup dom(F_i)$.

Рассматривая проекцию набора продукционных правил на сети *IPN*, легко построить траектории эволюции состояния системы после изменения переменных, связанных с внешней средой. Однако любая используемая для спецификации системы формальная модель не всегда учитывает реальные пространственно-временные соотношения между внешними событиями. Например, конъюнкция некоторых правил, представленная предикатами в нормальной форме, в реальных условиях не требует параллельной или одновременной проверки отдельных условий. Сказанное касается и других элементов логического вывода – множества противоречий и стандартных стратегий разрешения противоречий (новизны или конкретности).

Предлагается использовать свойства ассоциативности и коммутативности правил определения систем продукций для представления таких правил на дополнительных промежуточных состояниях. Это позволит снизить степень связности графов прямого и инвертированного отображения связи переменных состояния. Как следствие, время обработки последствий изменения состояния (время реакции) сокращается пропорционально уменьшению степени связности.

Интерпретация процессов на сетях интересна для решения задач, связанных с контролем формальных свойств сети, например, живости, ограниченности и достижимости. В случае неоднородных сетей переходов этот метод единственный. В контуре управления сеть используется как для поиска решений, так и обнаружения возмущений запланированных траекторий посредством расширения набора продукционных правил. Задачи координации процессов на сети, возникающие из-за свободы выбора альтернатив, могут быть решены посредством упорядочения выходных связей элементов сети.

На примерах задач управления обслуживанием в серверных системах и оптимизации маршрутов обсуждаются особенности их представления шаблонами классов и функций, допускающие специализацию на условия применения [3]. Рассматриваются варианты интерпретации сетевых моделей в реальном времени, а также для организации распределенных вычислений.

Литература

1. Ларман, К. Применение UML 2.0 и шаблонов проектирования. Введение в объектно-ориентированный анализ, проектирование и итеративную разработку. – М.: Вильямс, 2009.–736

2. Питерсон, Дж. Теория сетей Петри и моделирование систем. – М.: Мир, 1984. – 264 с.

3. Чан, З.А. Полиморфные модели процессов на сетях переходов/З.А.Чан, М.П. Ревотюк//Известия Белорусской инженерной академии, № 1(15)/2. – 2003. – С. 185-188.

4. Ревотюк, М.П. Полиморфные сетевые модели дискретных процессов/М.П. Ревотюк, Н.В. Хаджинова//Труды V Междунар. конф. “Идентификация систем и задачи управления” SICPRO’06, Москва, 30 января – 2 февраля 2006 г.//М: ИПУ им. В.А. Трапезникова РАН, 2006. – С. 2042-2158.