У статті розглянуто найбільш прогресивну та постійно розвиваючуюся галузь науки і техники - телекомунікаційний зв'язок. Описана сучасна цифрова телекомунікаційна система SI2000. Наведено доцільність використання цифрової автоматичної телефонної станції SI2000 для побудови комунікаційних мереж зв'язку.

Метою дослідження є застосування сучасної цифрової автоматичної телефонної станції для побудови міських та сільських мереж зв'язку. SI 2000 – це цифрова телекомунікаційна системи з функціями ОКС №7 (система сигналізації), ЦМІС, xDSL, IPOP, СОРМ, V5.2, яка забезпечує надання телекомунікаційних послуг для аналогових абонентів та цифрових абонентів, а також реалізацію функцій керування та технічного обслуговування. Цифрова мережа з інтегрованими службами (ЦМІС) - це загальнодоступна телефонна мережа, що використовує цифрову технологію передавання сигналу і містить великий набір цифрових послуг, які стають доступними для кінцевих користувачів. Функція xDSL – це сімейство технологій, що дозволяють значно розширити пропускну здатність абонентської лінії місцевої телефонної мережі шляхом використання ефективних лінійних кодів і адаптивних методів корекції викривлень лінії на базі сучасних досягнень мікроелектроніки і методів цифрової обробки сигналу. Функція СОРМ – це комплекс технічних засобів, призначених для проведення оперативно-пошукових мероприємств у мережах телефонного, пересувного бездротового зв'язку та радіозв'язку. Функція V5.2 забезпечує надання телекомунікаційних послуг для аналогових абонентів та цифрових абонентів, а також реалізацію функцій керування та технічного обслуговування.

Функції керування і технічного обслуговування дозволяють контролювати роботу системи, абонувати та анулювати телекомунікаційні послуги, додавати та змінювати характеристики маршрутизації, виконувати заміри та збір статистичних даних по окремим частинам системи.

цифрова автоматична телефонна станція, коммунікаційні мережі, міська та сільська коммунікація

Одержано 15.04.14

УДК 681.513.2

Н.В. Смирнова, канд.техн. наук, В.В. Смирнов, доц., канд. техн. наук
Кировоградский национальный технический университет

Применение теории конечных автоматов в разработке программных систем

Приведено решение задачи устранения неоднозначностей в работе конечного автомата при создании управляющих программных систем. Создана структурная схема модели конечного автомата с учетом времени ожидания выполнения функций и результата их выполнения, подсчета количества событий и стека состояний.

конечный автомат, класс, SWITCH-технология, объект управления

Н.В. Смірнова, канд.техн. наук, В.В. Смірнов, доц., канд. техн. наук
Кіровоградський національний технічний університет

Застосування теорії кінцевих автоматів у розробці програмних систем

Наведено рішення задачі усунення неоднозначностей в роботі кінцевого автомата при створенні керуючих програмних систем. Створена структурна схема моделі кінцевого автомата з урахуванням часу очікування виконання функцій і результату їх виконання, підрахунку кількості подій і стека станів.

кінцевий автомат, клас, SWITCH-технологія,об'єкт управління

В настоящее время расширяется область применения SWITCH-технологии в разработке программного обеспечения систем управления различными объектами.

В основе SWITCH-технологии лежит теория конечных автоматов (Finite State Machine). Системы на основе конечных автоматов обладают детерминированным поведением, однозначными результатами тестирования, простой диагностикой, документированием и сопровождением на протяжении всего жизненного цикла системы.

Разработка программных систем на базе SWITCH-технологии в чистом виде пока не получила широкого распространения, хотя основные положения теории конечных автоматов реализованы в явном или неявном виде практически во всех компиляторах, управляющих программах в системах управления и графических интерфейсах операционных систем и прикладных программ.

SWITCH-технология обладает большим потенциалом в области разработки программных систем общего и специального назначения. К сожалению, существующие методики применения теории конечных автоматов в области разработки программного обеспечения не являются исчерпывающими и достаточными, что сдерживает ее широкое распространение.

Анализ исследований и публикаций. Наибольшее распространение в области создания управляющих программных систем получили системы на базе конечного автомата Мура [1], работа которого описывается выражением:

$$\begin{aligned} a(t+1) &= f[a(t), x(t)], \\ y(t) &= f(a(t)), \end{aligned} \qquad (1)$$

где $a(t+1)$ – состояние автомата в момент времени $t+1$;

$a(t)$ – состояние автомата в момент времени t;

$x(t)$ – входное воздействие (событие) на автомат в момент времени t;

$y(t)$ – управляющее воздействие (control action), соответствующее выходной функции автомата в состоянии a в момент времени t.

В работе [2] представлена концепция программирования на базе SWITCH-технологии, которая формулируется следующим образом: программа представляет собой совокупность конечных автоматов, выполняющихся параллельно и обменивающихся между собой сообщениями.

Другим ключевым свойством предлагаемой концепции является широкое использование таймеров, которые предназначены для привязки работы программы к реальному времени.

В ряде других публикаций [3], описаны примеры применения, преимущества и недостатки SWITCH-технологии, в той или иной мере повторяющие положения, изложенные в [1,2].

Постановка задачи. В соответствии с выражением (1) построим программную модель (шаблон) автомата Мура, на основании которой предполагается разработка базового класса, реализующего автомат программной системы (рис.1).

В модель автомата А входят следующие переменные: x – для фиксации входного воздействия X, a – для фиксации текущего состояния автомата, y - для указателя на функцию f управляющего воздействия Y, вызываемую в текущем состоянии a автомата:

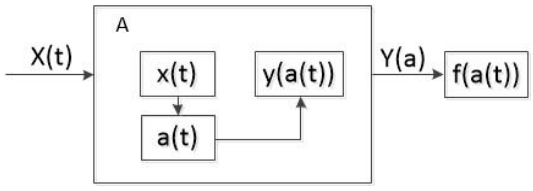

Рисунок 1 – Программная модель автомата Мура

Анализ модели показывает, что программная система, имеющая в своей основе данную модель, является нежизнеспособной, поскольку не учитывает целый ряд факторов, присущих реальным объектам управления и требований, предъявляемых к управляющим программам.

Например, отсутствие в выражении (1) оценки результата выполнения функции *f(a)* может привести к непредсказуемым шагам в работе автомата.

Результат выполнения функции *f(a)* может быть:

-положительным, когда функция завершена с ожидаемым результатом;

-отрицательным, когда функция завершена с результатом, отличным от ожидаемого;

-неопределенным, когда функция находится в состоянии выполнения.

Ожидание завершения выполнения функции *f(a)* приводит к «зависанию» программы системы управления и неспособности системы реагировать на другие входные воздействия.

Поэтому, для использования положений SWITCH-технологии в разработке программного обеспечения необходимо устранить ряд ограничений, сдерживающих ее широкое применение, таких, как время ожидания выполнения функции, возможность возврата в предыдущее состояние автомата, дифференциация перехода в следующее состояние по счетчику событий и др.

Основная часть. Современные языки программирования являются объектно-ориентированными и реализуют событийную модель создания программного обеспечения (ПО), которая хорошо согласуется с основными положениями SWITCH-технологии.

Однако, ни один язык программирования общего применения не содержит в себе никаких специальных средств для поддержки SWITCH-технологии. Поэтому создание автоматных шаблонов проектирования на основе доступных программных средств, позволит в полной мере реализовать преимущества данной технологии.

Обозначенная выше проблема потери управляемости системы во время ожидания завершения функции *f(a)* может иметь два варианта решения:

- выход из состояния ожидания по истечению интервала тайм-аута;
- запуск функции *f(a)* в параллельном потоке с использованием тайм-аута.

В этом случае, часть выражения (1) примет следующую форму:

$$a(t+1) = f[a(t), x(t), To(f(a))], \tag{2}$$

где *To(f(a))* – время тайм-аута, отведенное для выполнения функции *f(a)*.

Необходимым условием правильной работы автомата и является оценка результата выполнения функции *f(a)*, то есть, функция по своему завершению должна возбуждать внешнее событие *R*, включающее в себя результат завершения. При этом выражения (2) примет следующую форму:

$$a(t+1) = f[a(t), x(t), To(f(a)), R(f(a))], \tag{3}$$

где *R(f(a))* – результат выполнения функции *f(a)*.

Следующей проблемой является неоднозначность при возврате автомата из состояния *a(t)* в состояние *a(t-1, t-2 .. t-n)* при одинаковых событиях *X0*, как показано на рис. 2:

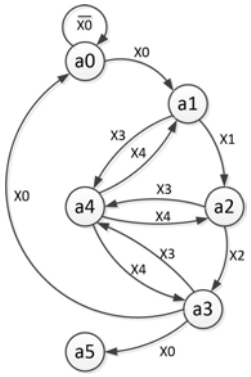

Рисунок 2 – Неоднозначность при возврате автомата в предыдущее состояние

На рис.2 представлен граф автомата, в котором по событию *X3* автомат из состояний *a1, a2* и *a3* переходит в состояние *a4*. После выполнения функции *f(a4)* автомат по событию *X4* должен вернуться в состояние *a(t-1)*, то есть в то состояние из которого был осуществлен переход для реализации программного или аппаратного прерывания. При этом возникает проблема неоднозначности перехода.

Решением задачи устранения этой неоднозначности является введение в модель автомата стека состояний *S*. При этом выражения (3) примет форму:

$$a(t+1) = f[a(t), x(t), To(f(a)), R(f(a)), S(a(i))], \tag{4}$$

где *S(a(i))* – стек состояния автомата.

Еще одна неоднозначность возникает в состоянии *a3* при поступлении события *X0,* когда необходимо осуществить переход автомата в процессе выполнении цикла (рис. 2). Если цикл не завершен, то должен произойти переход автомата из состояния *a3* в состояние *a0*, в противном случае – переход в состояние *a5* для завершения работы.

Поэтому модель автомата должна содержать счетчик циклов или входных событий *C*. Учитывая все дополнения, выражение (1) примет форму:

$$\begin{aligned} a(t+1) &= f[a(t), x(t), To(f(a)), R(f(a)), S(a(i)), C(x(j)] \\ y(t) &= f(a(t)). \end{aligned} \tag{5}$$

где *S(a(i))* – счетчик входных событий (циклов).

Таким образом, модель конечного автомата для создания шаблона базового класса будет иметь следующую структуру:

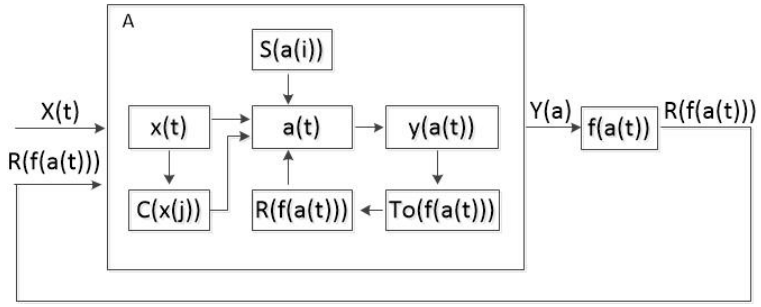

Рисунок 3 – Структура модели конечного автомата для реализации базового класса

Полученная структура модели конечного автомата легко реализуема в терминах языков программирования C++, Java и других языков программирования.

Выводы. Применение конечных автоматов в области создания программных систем позволяет детерминировать поведение программы, исключить ошибки в логике программы и формализовать процесс разработки.

Применение механизма наследования позволяет программисту использовать созданные шаблоны автоматного программирования не вникая во внутреннюю структуру базового класса. Реализация базового класса конечного автомата на основе созданной модели и результаты работы тестовой программы позволяет сделать выводы о жизнеспособности и достаточной эффективности управляющих программ на базе SWITCH-технологии.

К некоторым неудобствам применения конкретной реализации технологии относится использование оператора switch(){} для осуществления проверки условий и переходов автомата. Поэтому, более удобным является использование индексно-матричного подхода для решения задачи изменения состояний автомата.

С целью создания «интеллектуальных» автоматов с динамически изменяемой структурой и поведением, представляется целесообразным проведение исследований на предмет использования нейронных сетей и генетических алгоритмов для построения модели конечного автомата.

Список литературы

1. Шалыто А. А. Switch-технология. Алгоритмизация и программирование задач логического управления / А. А. Шалыто/ - СПб.: Наука, 1998. – 628 с.
2. Татарчевский В. Применение Switch-технологии при разработке прикладного программного обеспечения для микроконтроллеров / В. Татарчевский. - Компоненты и технологии №11, №12 - 2006, №1, №2- 2007.
3. Шалыто А. А. Автоматное программирование / А. А. Шалыто, Н. И. Поликарпова. - СПб.: Питер, 2011. - 176 с.

Nataliya Smirnova, Vladimir Smirnov
Kirovograd National Technical University

The finite state machine theory in developing software systems application

The purpose of this article is the solution to eliminate the finite state machine ambiguities when creating control software systems. Created a block diagram of a finite automaton model with the time to wait for the functions results and their implementation, counting the number of events and status stack.

The model combines the advantages of an event object-oriented technology and the SWITCH-programming technology. The resulting model is the basis of the base class machine that is used in the process of creating software systems for management of the facility.

finite state machine, class, SWITCH-technology, controlled object

Получено 02.4.14