УДК 681.3.06

**В.Л. СОСОНКИН, д-р техн. наук, проф.,**
**Г.М. МАРТИНОВ, канд. техн. наук,**
**А.Б. ЛЮБИМОВ, инж.**

# Интерпретация диалога в Windows-интерфейсе систем управления

Рассмотрены функции диалога системы управления с оператором и технические средства поддержания диалога. Приведена формальная модель диалога в виде иерархического графа состояний. Введено представление об интерпретаторе диалога как механизме последовательного вызова услуг соответственно действиям оператора, выражающимся в нажатии на клавиши. Разработана объектная структура интерпретатора (в смысле объектно-ориентированного подхода). Продемонстрирована возможность инструментальной поддержки процесса проектирования интерпретатора диалога.

The functions of dialog between control system and human operator as well as dialog support hardware is considered. A formal dialog model in the form of hierarchical state graph is presented. An idea of dialog interpreter as of a mechanism, which consequently calls for services according to operator's actions expressed by key pressing, is introduced. The interpreter's object structure (in the sense of object-oriented approach) is developed. An opportunity of instrumental support of dialog interpreter design process is demonstrated.

## Постановка задачи

Современные системы управления используют архитектуру персонального компьютера (ПК) и в этой связи располагают широкими возможностями организации человеко-машинного интерфейса MMI (Man-Machine Interface) в операционных средах Windows NT и Windows'95. Проблему построения MMI называют терминальной задачей управления [1]. Эта задача выполняет функции клиента в клиент-серверной структуре математического обеспечения системы управления; роль же сервера принадлежит программно-аппаратной виртуальной шине [2], являющейся единым средством коммуникации для всех модулей системы управления.

Если унифицировать доступ к виртуальной шине, то появляется возможность независимого проектирования конкретного MMI-приложения, которое включает четыре этапа: 1) создание скелета MMI-приложения; 2) реализацию экранов; 3) разработку интерпретатора диалога с оператором; 4) организацию информационных сессий с остальными модулями системы управления. Три последних этапа особенно тесно связаны между собой, причем этап разработки интерпретатора наиболее сложен. По этой причине предпринята попытка формализации третьего этапа, что и определяет содержание последующего изложения.

## Функции диалога с оператором, средства поддержания диалога

В числе функций диалога можно выделить получение разнообразной текущей информации о процессе управления; тестирование системы и объекта и получение диагностической информации; редактирование и моделирование управляющей программы; ручной ввод и управление отработкой его данных; ввод программы и управление ее отработкой в автоматическом режиме; управление наладочными операциями. Содержанием диалога служат допустимые переходы между состояниями MMI-приложения, в рамках которых и обеспечивается воспроизведение необходимых функций. Оператор системы управления осуществляет переходы между состояниями посредством аппаратной и функциональной клавиатуры, причем роль последней доминирует. Следует заметить, что по условиям эксплуатации и в соответствии с требованиями заказчика использование мыши во многих случаях недопустимо.

На рис. 1 показан пример экрана MMI-приложения, на котором размещается базовое окно (frame window) MMI-приложения, состоящее из трех компонентов (сверху вниз): окна статуса, статусбара (StatusBar); рабочей области; области функциональной клавиатуры, тулбара (ToolBar). Размещение компонентов базового окна определяется дизайном экрана системы управления. Вспомогательные элементы окна (заголовок, меню, строка состояния) используются по усмотрению заказчика системы управления и разработчика MMI.

Рабочая область покрыта окнами-панелями с рамками и заголовками; каждая панель предоставляет функционально-однородную информацию. Панель, в свою очередь, содержит управляющие элементы (Control elements), которые предназначены для динамического отображения данных, поступающих от сервера или вводимых оператором с помощью клавиатуры (ПК или панели оператора). Управляющие элементы могут быть выбраны при проектировании экрана из стандартной библиотеки (галереи управляющих элементов) или подлежат проектированию заново в соответствии с требованиями заказчика (с последующим включением в галерею).

Статусбар включает в себя панели с управляющими элементами: специфическими (пиктограммами режимов и подрежимов, индикаторами готовности, индикаторами используемых системных ресурсов, индикаторами даты и времени) или общего характера (такими же, как и в рабочей области: для слежения за информацией, закрытой при смене картинки рабочей области).

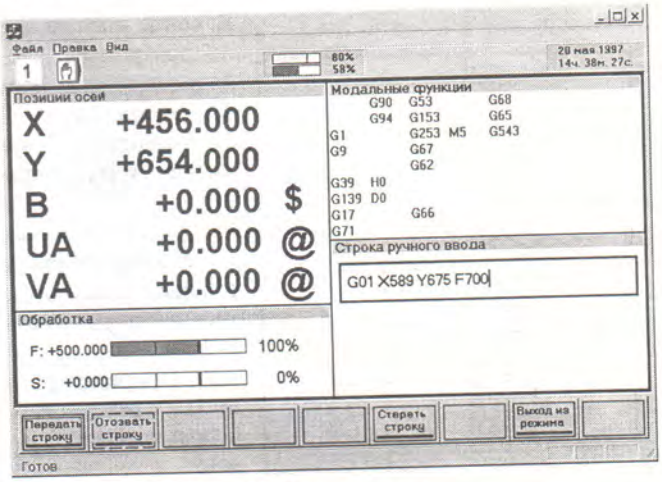


*Рис. 1. Пример экрана MMI-приложения*

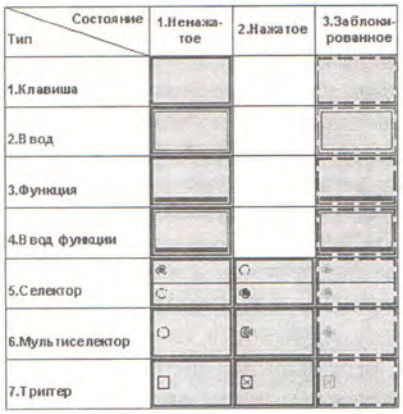


*Рис. 2. Типы и состояния кнопок тулбара*

Тулбар состоит из кнопок функциональной клавиатуры, каждая из которых является управляющим элементом, имеющим динамическое имя. Типы и состояния кнопок тулбара представлены на рис. 2. Традиционный тип кнопки – "Клавиша", нажатие на которую переводит систему в новый режим (подрежим) или открывает диалоговое окно. Кнопка типа "Ввод" работает так же, как и соответствующая клавиша на панели оператора. Кнопка типа "Функция" инициирует выполнение некоторой функции без предварительных запросов и подтверждений. Кнопка "Ввод функции" объединяет возможности двух предыдущих. Кнопка "Селектор" выбирает при нажатии на нее одну из двух функций (например, функцию метрической или дюймовой размерности). "Мультиселектор" представляет собой группу, в которой активизация одной кнопки деактивирует все остальные. Кнопка "Триггер" последовательно активирует и деактивирует выбранную функцию. Все кнопки могут находиться в состоянии: готовности, когда они представлены в тулбаре и доступны оператору (ненажатое состояние); работы под воздействием оператора (нажатое состояние); блокировки, когда они отсутствуют в тулбаре или когда воздействие на них оператора игнорируется (заблокированное состояние).

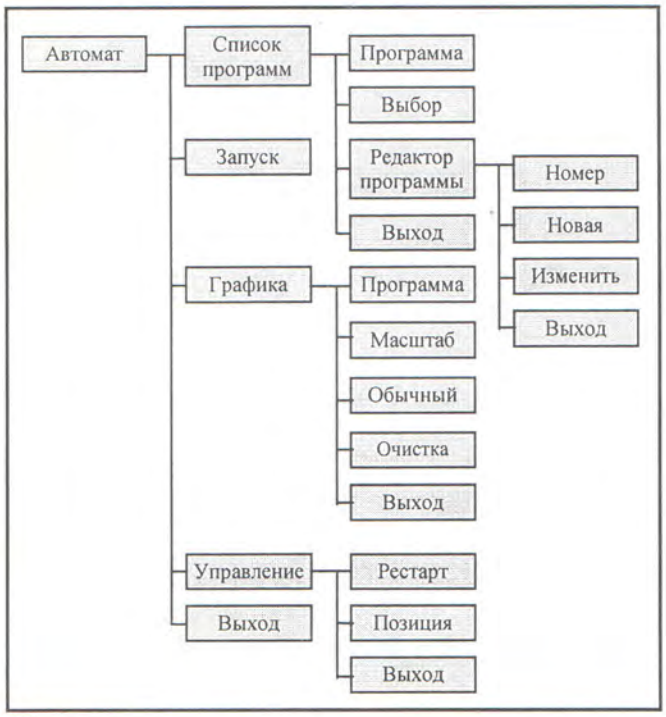


*Рис. 3. Фрагмент дерева для режима автоматического управления*

**Формальное описание диалога**

Структура диалога в гибкой системе управления с открытой архитектурой [3] определяется заказчиком системы управления, для которого привычным языком внешнего описания диалога служит дерево режимов-подрежимов. Ветвям дерева приписаны имена функциональной клавиатуры тулбара. В качестве примера на рис. 3 приведен фрагмент дерева для режима автоматического управления в системе числового программного управления: "Sentrol" (Южная Корея). Недостатки подобного представления дерева состоят в следующем: оно не показывает возвратов к началу диалога или его ранним стадиям; не обеспечивает хорошую обозримость дерева многорежимных и многоуровневых диалогов; не поясняет структуры тулбара.

В этой связи предложено описывать диалог так, как это сделано на рис. 4, – для того же фрагмента диалога системы "Sentrol". Формальной моделью диалога служит граф состояний, вершины которого отражают устойчивые состояния MMI после нажатия оператором на ту или иную клавишу панели оператора, а дуги нагружены именами клавиатуры тулбара или других клавиш.

Дополнительно введено представление об иерархическом графе состояний, образом которого является сложное состояние (на рис. 4 обозначено двойным кружком), интерфейсом – входные и выходные порты. Сложное состояние само по себе – граф, раскрывающийся так, как показано в прямоугольном контуре на рис. 4. Иерархические графы – это удобное средство описания многорежимных многоуровневых диалогов, позволяющее проектировать диалог "шаг за шагом" – от укрупненного выбора режимов к детальному определению подрежимных функций.

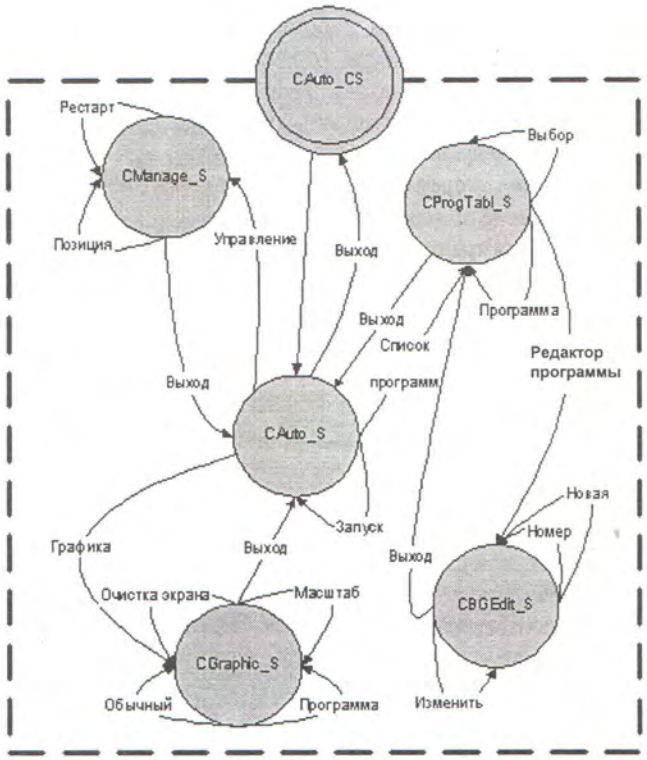


*Рис. 4. Фрагмент графа режима автоматического управления*

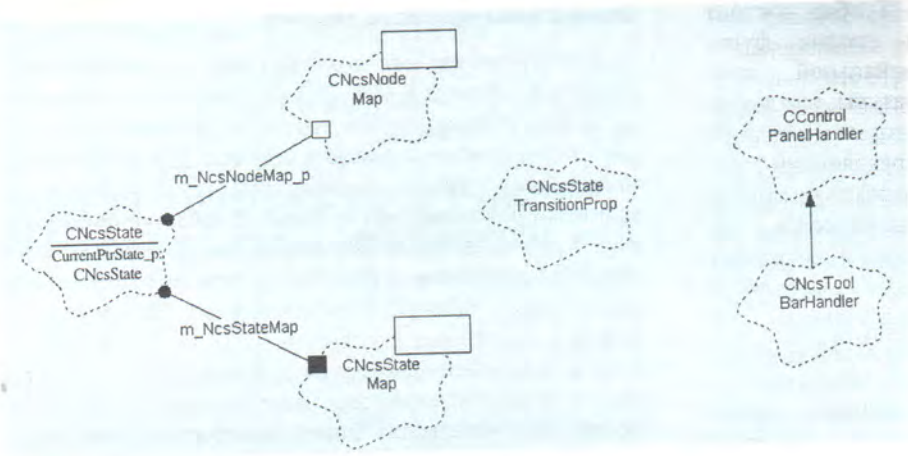


*Рис. 5. Диаграмма классов интерпретатора диалога*

Иерархический граф описывает процесс управления не только с использованием тулбара, но и с помощью меню, дерева навигации и т.д. В связи с этим введено представление "Панель управления", с помощью которой может быть реализован диалог системы с оператором.

### Представление об интерпретаторе диалога

В графе состояний могут быть определены две группы переходов: транзитивные, связанные со сменой состояний, и нетранзитивные, отображаемые дугой, возвращающейся в прежнее состояние. Переход любого типа инициируется нажатием на клавишу, имя которой приписано переходу (дуге). Поскольку каждая клавиша генерирует свой собственный код (скан-код), этот код также сопоставляется переходу (дуге).

По своей сути переход есть запрос на услугу, содержательно представляющую собой смену экранов, обновление имен клавиатуры экрана, обращения к серверу и др. Услуги реализуются объектами, работающими в состояниях ММІ (имена классов объектов означают состояния графа, см. рис. 4). Таким образом, ход диалога представляет собой очередность следующих событий: нажатие на клавишу оператором и генерация соответствующего скан-кода; обращение к объекту (для транзитивных переходов – в новом состоянии; для нетранзитивных – в старом) с запросом на услугу (или услуги); реализация услуг объектом; ожидание очередных действий оператора. Эту цепочку называют последовательностью актов [4], в числе которых есть входные (нажатия на клавиши) и выходные (все остальные). Выходные акты порождаются в результате интерпретации входных. Следовательно, интерпретатор диалога есть механизм последовательного вызова услуг соответственно действиям оператора, выражающимся в нажатии на клавиши. Помимо вызова услуг интерпретатор осуществляет синтаксический контроль действий оператора, препятствуя неправильным действиям.

### Объектная структура интерпретатора

Диаграмма классов интерпретатора диалога в нотации Booch [5] приведена на рис. 5. Она отражает состав и взаимоотношения классов. В соответствии с соглашением нотации каждый класс изображают облаком. Имя класса начинается с буквы "С" (например, CNcsState). Отношения между классами (а также структурами, перечисляемыми типами и объединениями) показывают стрелками или другими линиями. Различные виды отношений классов даны на рис. 6.

Прототипом класса состояния, как и сложного состояния (см. рис. 5), служит класс CNcsState. Этот класс хранит в m-поле m_NcsStateMap таблицы возможных переходов из текущего состояния. Структуру и тип переходов описывает объект класса CNcsStateTransitionProp. В числе свойств перехода – скан-код (или номер порта), сопоставленный переходу; указатель на следующее или сложное состояние; описание элемента панели управления (сопоставленные переходу текст, тип и т.д.).

Таким образом, совокупность описаний элементов панели управления, сопоставленных возможным переходам из состояния, определяет вид конкретной панели управления для данного состояния. Тип панели управления (тулбар, меню, пользовательская панель управления) зависит от выбора класса управления, унаследованного от класса CControlPanelHandler.

Класс CNcsState содержит также указатель на таблицу вложенных состояний в m-поле m_NodeObject_p. Эта

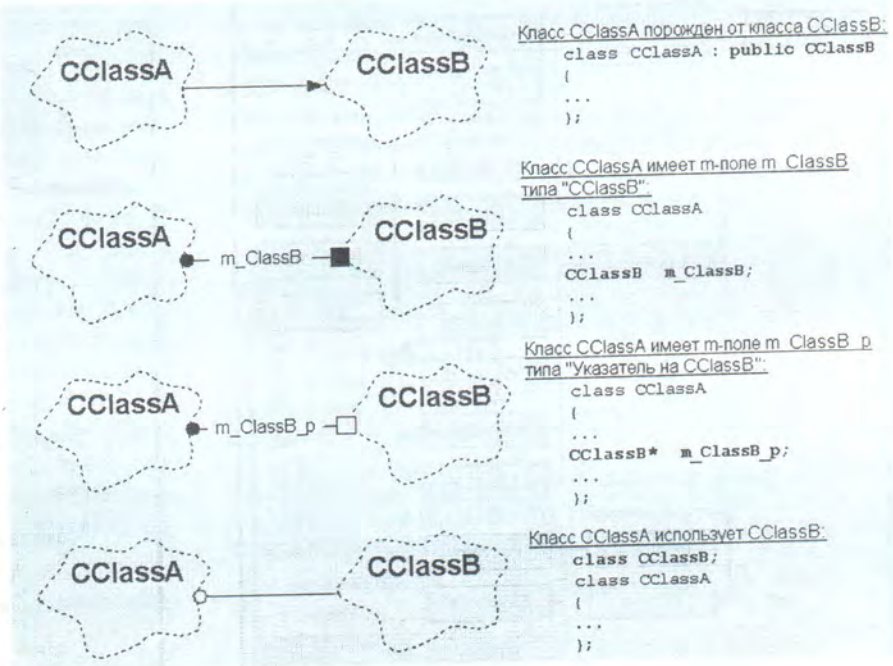


*Рис. 6. Отображение взаимоотношений классов на диаграммах*


*Рис. 7. Блок-схема перехода из одного состояния в другое*

таблица инициализируется и заполняется только для сложных состояний.

Ссылка на текущее состояние диалога оператора хранится в статическом поле CurrentPtrState_p класса CNcsState. Будучи статическим, поле является глобальным для всех объектов класса CNcsState.

Укрупненная блок-схема перехода из одного состояния в другое представлена на рис. 7. Пункт 5 блок-схемы игнорируется при переходе в сложное состояние. При переходе из одного сложного состояния в другое роль скан-кода играет номер порта. При этом переходы в сложном состоянии носят динамический характер и автоматически следуют один за другим до тех пор, пока интерпретатор диалога не осуществит переход в новое сложное состояние.

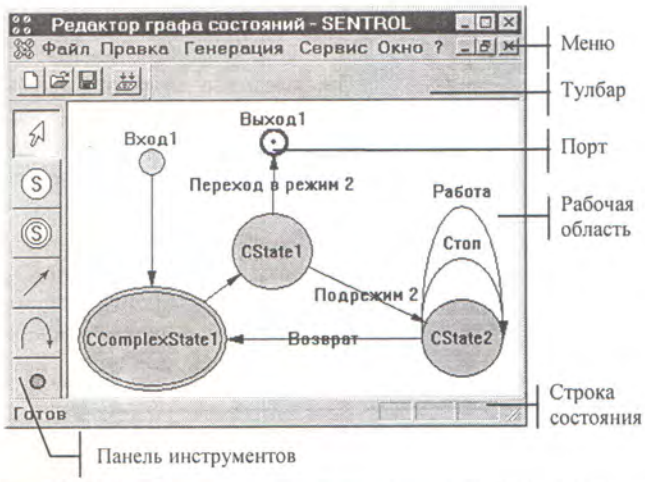


*Рис. 8. Рабочее окно инструмента генерации C++ кода интерпретатора диалога*

## Инструментальные средства разработки интерпретатора

Процесс разработки интерпретатора диалога итеративен, причем число итераций может достигать нескольких десятков. Формализация процесса разработки диалога интерпретатора, обнаружение однотипности скелета исходного кода – все это позволило разработать инструмент визуального проектирования (рис. 8), генерирующий исходные файлы на языке C++.

В рамках инструментальной системы задание вводят непосредственно в виде иерархического графа. В диалоговом режиме устанавливают имена состояний и свойства переходов. Глубина вложения назначается разработчиком по его усмотрению, и это позволяет последовательно концентрировать внимание на текущих фрагментах диалога. Разработчик должен также обозначить имена файлов, в которые будут сгенерированы классы состояния. Ему предоставляются дополнительные возможности изменять шрифт, масштабировать изображения, добавлять пользовательские классы в качестве базовых, вводить комментарии, определять формат распечатки при документировании проекта.

Применение инструмента визуального проектирования многократно повышает производительность разработчика, позволяет создавать очень сложные интерпретаторы, реализация которых без инструментальной поддержки весьма проблематична.

## Выводы

Коды, реализующие диалог с оператором, образуют в MMI-приложении вполне обособленный модуль, выполняющий специфичные функции и имеющий свой интерфейс. Специфика состоит, например, в проверке доступности предполагаемого перехода. Механизм обработки сообщений служит всего лишь основой для построения интерпретатора диалога. Чрезвычайно сложные графы, описывающие интерпретатор диалога, становятся обозримыми при использовании концепции иерархических графов. Универсальная объектно-ориентированная модель для реализации интерпретатора диалога может обеспечить работу с различными панелями управления и сценариями диалогов. Построение и отладка сложных интерпретаторов диалога радикально упрощается с помощью инструментальных средств разработки.

*Контактный телефон (095) 972-94-40 (МГТУ "Станкин").*

## Список литературы

1. *Сосонкин В.Л.* Программное управление технологическим оборудованием: Учебн. для вузов. М.: Машиностроение, 1991.
2. *Сосонкин В.Л., Мартинов Г.М.* Принципы построения систем ЧПУ с открытой архитектурой // Приборы и системы управления. 1996. № 8.
3. *Сосонкин В.Л., Мартинов Г.М.* Концепция открытой архитектуры персональных систем числового программного управления // Тр. 3-го междунар. конгресса "Конструкторско-технологическая информатика" – КТИ – 96 (22 – 24 мая, 1996, Москва). М.: МГТУ "Станкин", 1996.
4. *Грис Д.* Конструирование компиляторов для цифровых вычислительных машин / Пер. с англ. М.: Мир, 1975.
5. *Буч Г.* Объектно-ориентированное проектирование с примерами применения / Пер. с англ. М.: Конкорд, 1992.