мысления самих принципов построения систем ЧПУ. В числе важнейших из них — объектно-ориентированный подход [6] не только к технологии программирования (для повышения надежности ПМО), но и к проектированию системы. Последнее означает, что основные крупные модули системы PCNC целесообразно рассматривать как "вложенные объекты", между которыми существуют отношения типа "клиент — сервер". Одним из вариантов проектного решения является выделение глобального сервера — программно-аппаратной шины, которая служит основным средством межмодульной коммуникации.

Рассмотрим модели системы PCNC с открытой архитектурой, имеющие различный уровень. Процесс проектирования системы состоит в последовательной трансляции модели менее глубокого уровня в модель более глубокого уровня.

Модель верхнего уровня (рис. 1) представляет собой совокупность базовых модулей и дополнительных модулей — покупных программных пакетов. Каждый модуль — автономный и является "вложенным объектом", т.е. обладает алгоритмической структурой и собственными данными, а также интерфейсной оболочкой, ориентированной на интерактивную работу. Такой модуль представляет собой, как правило, сложную объектно-ориентированную структуру и отражает глобальный объектно-ориентированный подход к проектированию системы PCNC [6 и др.].

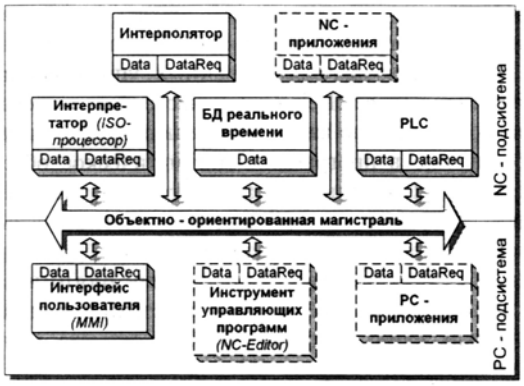

Рис. 1. Модель верхнего уровня (сплошные линии — базовые модули, штриховые — дополнительные)

Модель состоит из двух подсистем — NC и PC. Первая является ведущей и формирует среду функционирования модулей в реальном времени. Среди этих модулей — интерпретатор (ISO-процессор [6]), интерполятор, встроенный программируемый логический контроллер (PLC), база данных (БД) реального времени и (возможно) специальные NC-приложения пользователя. Вторая подсистема образует среду Windows-интерфейса пользователя (MMI — Man-Machine Interface) и может включать в себя также инструментальную систему подготовки и тестирования управляющих программ (NC-Editor) и другие PC-приложения.

Модули взаимодействуют посредством объектно-ориентированной магистрали, которая не только использует программно-аппаратные коммуникационные протоколы, но и (благодаря специальной оболочке) выполняет прикладные серверные функции, т.е. предоставляет всем модулям информационные услуги. Это же предусмотрено и в самих интерфейсах модулей: они могут предоставлять данные (Data), запрашивать их (DataReq, т.е. Data Request) или делать и то, и другое.

Все модули, подключенные к объектно-ориентированной магистрали, запрашивают данные синхронным или асинхронным способом, в том числе "по изменению данных". Выбор способа реализации запроса зависит от конкретной задачи. При синхронном запросе клиент (т.е. модуль, осуществляющий запрос) останавливается в точке запроса и ждет ответа от модуля, выполняющего запрос, до истечения контрольного времени. При асинхронном запросе клиент продолжает работу, а обработка ответа независимо от момента его получения выполняется специальной функцией (callback). Запрос "по изменению данных" (синхронный либо асинхронный) означает, что ответ будет получен только после изменения данных.

Назначение модели, представленной на рис. 1, состоит в выборе модулей системы PCNC и их интерфейсов, а также в установлении типов запросов и составлении технического задания на объектно-ориентированную магистраль.

Архитектурная модель. При разработке этой модели должен быть прежде всего сделан выбор в пользу двухкомпьютерной (рис. 2, *а*) или

ISSN 0869-7566

однокомпьютерной (рис. 2, *б*) системы PCNC. Второй вариант более перспективен, однако особо гибкие и сложные системы PCNC, ориентированные на многокоординатную высокоскоростную и высокоточную обработку, пока еще выполняют по первому варианту (см. рис. 2, *а*), который поэтому принят далее за основу.

При разработке модели следует прежде всего выбрать платформу системы PCNC, т.е. операционную систему (ОС) и аппаратные средства. Для PC-подсистемы наиболее целесообразна ОС Windows NT, а для NC-подсистемы — ОС реального времени (ОС РВ), среди которых наиболее распространенная — UNIX. Обе ОС используют коммуникационные протоколы TCP/IP, что позволяет создать коммуникационную среду, объединяющую две подсистемы. Организация в этой среде некоторого прикладного уровня с функциями доступа к интерфейсам модулей (общее число таких функций может достигать нескольких сот) создает виртуальную шину, обеспечивающую доступ к интерфейсам модулей на низком уровне. Объектно-ориентированная надстройка над этой шиной представляет собой глобальный сервер — единую для обеих подсистем объектно-ориентированную магистраль.

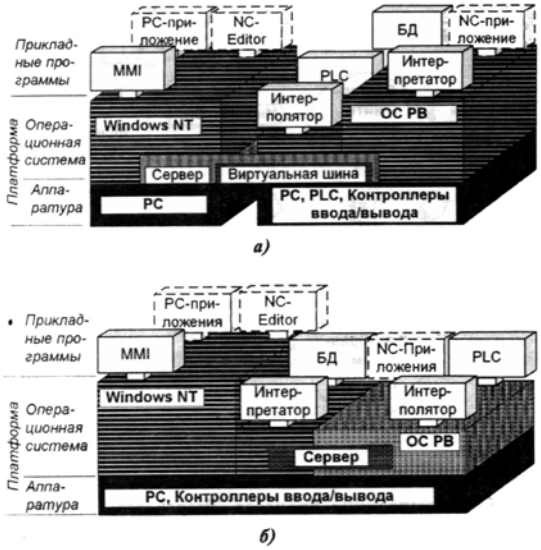

Рис. 2. Архитектурная модель системы PCNC: *а* — двухкомпьютерный вариант; *б* — однокомпьютерный вариант (PC — компьютер; остальные обозначения см. на рис. 1)

Объектно-ориентированная надстройка включает в себя административные, коммуникационные и вспомогательные классы объектов. Объекты административных классов открывают и закрывают доступ к магистрали, устанавливают контрольное время для синхронного запроса, производят диагностирование и тестирование как модулей, так и магистрали. Объекты коммуникационных классов обеспечивают транзакции (сессии) запросов различного типа. Объекты вспомогательных классов переводят данные из одной размерности в другую, осуществляют масштабирование и форматирование данных.

Таким образом, основной задачей архитектурной модели является разработка платформы и сервера (надстройки виртуальной шины) системы PCNC.

DFD-модель (Data Flow Diagram, т.е. диаграмма потоков данных [7]) служит целям углубленной структуризации модулей системы PCNC, предшествующей их объектно-ориентированному проектированию. Эта модель является многоуровневой иерархией, начальные уровни которой показаны на рис. 3—6.

На исходной диаграмме (рис. 3) вся система PCNC представлена в виде глобального процесса "Выполнить задание", а внешние источники и стоки (т.е. адресаты) данных (в прямоугольниках) составляют ее окружение. Потоки данных (обозначены стрелками с именами потоков) моделируют одно- или двунаправленную передачу информации. Назначение процесса состоит в формировании выходных потоков из входных в соответствии с действием, заданным именем процесса.

Далее осуществляется декомпозиция процесса "Выполнить задание" (рис. 4) на два других, соответствующих PC-подсистеме (процесс "Разработать задание") и NC-подсистеме (процесс "Управлять объектом"). В результате декомпозиции появляются новые потоки данных между двумя подсистемами.

Дальнейшая декомпозиция (рис. 5) обращена к процессу "Разработать задание". При этом прежде всего обнаруживается необходимость интерпретации действий оператора (процесс "Интерпретировать состояние"), которые переводят систему PCNC из одного конечно-автоматного состояния в другое. В рамках диалога оператор

выбирает режим (подрежим) и вводит соответствующее задание, инициирующее процесс "Работать в выбранном режиме". Интерфейс пользователя (MMI) является по своей природе клиентом, который обращается к серверу (процесс "Выполнить функции сервера") по любому поводу: для получения информации, для инициирования передачи данных или управляющей программы (УП) и т.д.

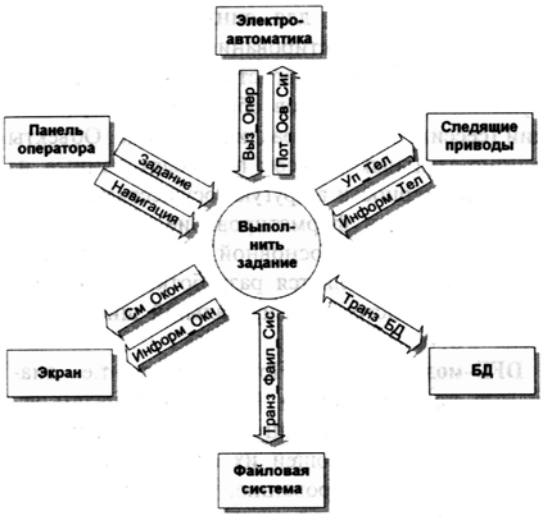

Рис. 3. Исходная диаграмма DFD-модели: См_Окон — смена окон экрана; Информ_Окн — информация для окон; Транз_Файл_Сис — транзакции файловой системы; Транз_БД — транзакции БД; Информ_Тел — информационная телеграмма от следящих приводов подачи; Уп_Тел — управляющая телеграмма приводам подачи; Пот_Осв_Сиг — поток сигналов от приводов электроавтоматики; Выз_Опер — вызов операций электроавтоматики

Аналогично осуществляется декомпозиция процесса "Управлять объектом" (рис. 6), инициированного потоком данных со стороны клиента. Главные функции этого процесса состоят в обработке УП или строки ручного ввода (MDI — Manual Data Input), которая интерпретируется, а затем отрабатывается интерполятором и контроллером PLC. Драйверы ввода-вывода включены в состав как интерполятора, так и PLC, в связи с чем возникает необходимость дальнейшей декомпозиции, которая на рис. 3—6 не отражена.

Следовательно, основное назначение DFD-диаграмм состоит в декомпозиции исходной модели системы PCNC на такую совокупность процессов и потоков данных, которая подлежит непосредственному программированию. Уточнение семантики, структуры и типов потоков данных является другой важной проблемой, рас-

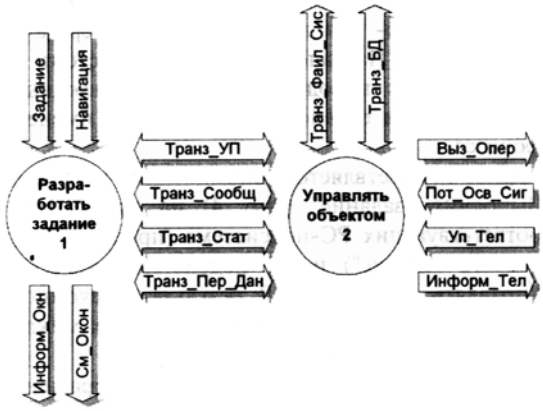

Рис. 4. Декомпозиция процесса "Выполнить задание": Транз_УП — транзакции управляющих программ; Транз_Сообщ — транзакции сообщений для поля сообщений на экране; Транз_Стат — транзакции статусов для поля статусов на экране; Транз_Пер_Дан — транзакции данных для управляющих элементов экрана; остальные обозначения см. на рис. 3

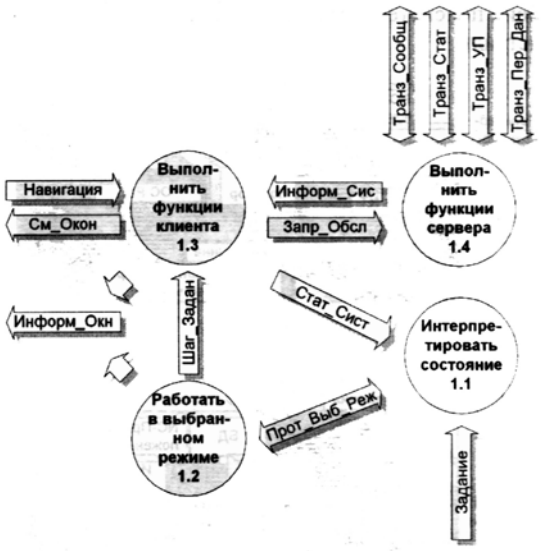

Рис. 5. Декомпозиция процесса "Разработать задание": Шаг_Задан — передача шагов задания; Информ_Сис — информация NC-подсистемы; Запр_Обсл — запросы к NC-подсистеме; Стат_Сист — передача статуса интерпретатору состояний; Прот_Выб_Реж — протокол выбора режима; остальные обозначения см. на рис. 3 и 4

ISSN 0869-7566

Рис. 6. Декомпозиция процесса "Управлять объектом": Транз_Интерпол — транзакции интерполятора; Транз_Интерпрет — транзакции интерпретатора; Транз_PLC — транзакции PLC; остальные обозначения см. на рис. 3—5

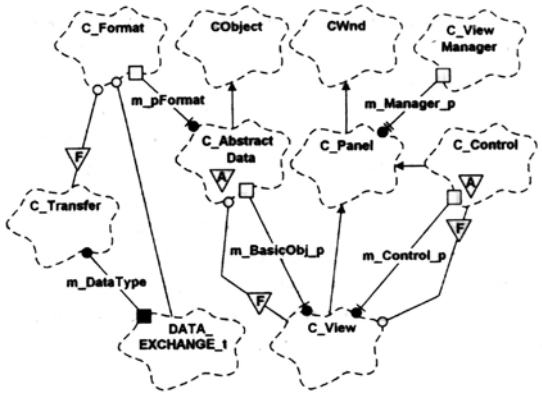

Рис. 7. Объектно-ориентированная модель процессов (фрагмент диаграммы классов)

смотрение которой выходит за рамки данной статьи.

Объектно-ориентированная модель процессов иллюстрируется рис. 7, где представлен в качестве примера небольшой фрагмент такой модели, который демонстрирует общий подход к ее построению. Рассматриваемая модель построена при помощи инструментальной CASE-системы Rational Rose фирмы Rational (США). Система Rational Rose использует формализованное представление классов объектов в виде "облаков Буча" [8], связанных между собой определенными отношениями. Приведенный здесь фрагмент модели (совокупность классов объектов) предназначен для динамического отображения разнообразных данных (текущего положения координатных осей, текущей подачи, текущего кадра программы) в окнах экрана.

Данные выводятся в тот или иной управляющий элемент (control element), соответствующий классу C_Control; этот элемент расположен в окне-панели экрана (класс C_Panel, производный от стандартного класса CWnd библиотеки MFC — Microsoft Foundation Classes). Расположение управляющих элементов в окне-панели настраивается менеджером экрана (класс C_View Manager). Источником данных служит коммуникационный класс, производный от класса C_AbstractData, полученного, в свою очередь, от библиотечного класса CObject.

Гибкость связи между коммуникационным классом и управляющим элементом поддерживается классом отображения C_View. Эта гибкость означает, что одни и те же данные могут быть запрошены в разных форматах (независимо от формата поступающих данных) и направлены в различные управляющие элементы. Формат запроса данных определяется типом (надклассом) DATA_EXCHANGE_t, а конвертацию форматов осуществляет класс C_Format. Передача данных между коммуникационным классом и управляющим элементом обеспечивается классом C_Transfer.

Объекты класса называются его экземплярами. Классы, не имеющие экземпляров, являются абстрактными; они помечены на диаграмме буквой A в треугольнике. Отношения между классами представлены их связями, причем существуют три основных типа отношений: наследование, наполнение и использование.

При *наследовании* производный класс наследует атрибуты и методы родительского класса; это обозначается стрелкой от производного класса к родительскому. При *наполнении* один класс поглощает другой; это обозначается в виде связи с закрашенным кружком на одном конце. На другом конце расположен квадрат — либо закрашенный (если один класс полностью включает в себя другой), либо незакрашенный (если класс содержит только ссылку на другой). При

использовании объекты классов обмениваются сообщениями; это обозначается в виде связи с незакрашенным кружком.

Помимо типа, отношения обладают статусом, который может быть *общедоступным* (public), *защищенным* (protected), т.е. доступным в некотором классе и производных от него классах, и *частным* (private), т.е. доступным в одном определенном классе. Доступность означает "видимость" части интерфейса одного класса из другого класса. Защищенный статус помечают одной чертой, перечеркивающей связь (см. рис. 7), а частный статус — двумя чертами. Ограничения в отношении видимости между двумя и более объектами разных классов снимаются при установлении отношения общности (friend); это обозначается буквой F в треугольнике.

Диаграмма классов, показанная на рис. 7, а также производная от нее диаграмма объектов определяют логическую структуру объектно-ориентированной модели. Задача этой структуры состоит в установлении архитектуры классов и объектов и взаимоотношений между ними. Следующим шагом является разработка физической структуры, т.е. архитектуры процессов и модулей, отображаемых соответственно на диаграмме процессов и диаграмме модулей.

Из всего изложенного следует, что к настоящему времени сложилось определенное представление о принципах построения систем PCNC с открытой архитектурой; однако пока еще не существуют системы, воплощающие эти принципы в полной мере. Причина состоит в исключительной сложности и чрезвычайно большом объеме ПМО, надежность которого может быть обеспечена лишь проверенными на практике методами и инструментальным сопровождением проектирования, а также тщательным документированием. В связи с этим целесообразно использовать модели ПМО, последовательная трансляция которых поддерживает разработку проекта системы PCNC.

Список литературы

1. **Сосонкин В.Л.** Концепция системы ЧПУ на основе персонального компьютера (PCNC) // Станки и инструмент. — 1990. — № 11. — С. 5 — 7.
2. **Сосонкин В.Л.** Новое поколение устройств ЧПУ на базе персонального компьютера // Приборы и системы управления. — 1992. — № 3. — С. 4 — 6.
3. **Сосонкин В.Л.** Персональный компьютер как архитектурный компонент "персональной системы управления" // СТИН. — 1993. — № 5. — С. 2 — 7.
4. **Сосонкин В.Л., Мартинов Г.М.** Принципы построения систем ЧПУ с открытой архитектурой // Приборы и системы управления. — 1996. — № 8. — С. 18 — 21.
5. **Сосонкин В.Л.** Сетевая коммуникационная среда персональной системы управления // СТИН. — 1996. — № 5. — С. 12 — 17.
6. **Сосонкин В.Л., Мартинов Г.М.** Концепция геометрического ISO-процессора для систем ЧПУ // СТИН. — 1994. — № 7. — С. 12 — 20.
7. **Кальянов Г.Н.** CASE структурный системный анализ (автоматизация и применение). — М.: ЛОРИ, 1996. — 240 с.
8. **Буч Г.** Объектно-ориентированное проектирование с примерами применения: Пер. с англ. — М.: Конкорд, 1992. — 519 с.

ISSN 0869-7566