На правах рукописи

Мартинов Георги Мартинов

Теория и техника систем числового программного управления с открытой модульной архитектурой для автоматизации машиностроительного оборудования

Специальность: 05.13.06 – Автоматизация и управление технологическими процессами и производствами (промышленность)

Автореферат
диссертации на соискание ученой степени
доктора технических наук

Москва - 2001 г.

Работа выполнена в Московском Государственном технологическом университете “СТАНКИН”

Научный консультант:	доктор технических наук, профессор Сосонкин В. Л.
Официальные оппоненты:	доктор технических наук, профессор Аршанский М. М. доктор технических наук, профессор Горнев В. Ф. доктор технических наук, профессор Фролов Е. Б.
Ведущая организация	Институт проблем управления РАН

Защита состоится “____” _____ 2001 г. в ___ часов на заседании диссертационного Совета Д 212.142.03 в Московском Государственном технологическом университете “СТАНКИН” по адресу: 101472, ГСП, Москва, Вадковский переулок, д. 1.

С диссертацией можно ознакомиться в библиотеке Московского Государственного технологического университета “СТАНКИН”.

Автореферат разослан “____” _____ 2001 г.

Ученый секретарь
диссертационного Совета
к.т.н. Е.Г. Семячкова

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность работы. Работа направлена на решение актуальной научной проблемы, имеющей важное народно-хозяйственное значение: создание теоретического базиса нового поколения систем ЧПУ широкого назначения, обладающих гибкостью на всех этапах жизненного цикла; разработка методических основ проектирования и организации производства таких систем при многообразии требований потребителей и заказчиков.

Существо научной проблемы состоит в том, что до настоящего времени теория создания сложных открытых модульных систем, а также методы определения их архитектуры и формализации процессов построения систем управления, адаптируемых к производственным условиям, с высокой конкурентоспособностью и низкими эксплуатационными затратами, фактически отсутствуют.

В результате комплекса исследований и разработок, выполненных при подготовке диссертации, созданы научные основы построения открытых модульных систем ЧПУ широкого назначения.

Цели работы, вытекающие из научной проблемы. В соответствии с указанной научной проблемой в работе поставлены следующие цели:

- создание научных основ и методологии проектирования открытых модульных систем ЧПУ нового поколения, обладающих гибкостью на всех этапах жизненного цикла, обеспечивающих, в конечном счете, повышение общей эффективности производства, в котором такие системы применяются;
- снижение себестоимости, повышение надежности систем ЧПУ и обеспечение конкурентоспособности за счет возможности изменения компоновки и встраивания готовых коммерческих компонентов;
- сокращение времени разработки, настройки и адаптации систем ЧПУ за счет формализации процесса проектирования и инструментальной поддержки этого процесса;

Постановка задач исследования. Для достижения поставленных целей необходимо решить следующие фундаментальные задачи:

- разработать теоретические аспекты построения модульной архитектуры и модели систем ЧПУ типа PCNC;
- разработать методологические аспекты проектирования открытых систем ЧПУ и сформировать инструментальное окружение процесса их разработки;

- разработать теоретические аспекты построения диспетчера реального времени и коммуникационной среды для решения проблемы интеграции модулей системы PCNC;
- на базе теоретических и методологических знаний о построении открытой модульной системы ЧПУ разработать практические аспекты реализации модулей открытой системы ЧПУ.

На защиту выносятся:

1. Результаты комплексного анализа вопросов создания открытых модульных систем ЧПУ.
2. Результаты теоретических исследований модульной архитектуры и моделей систем ЧПУ типа PCNC, результаты исследования задач диспетчирования в реальном времени и организации коммуникационной среды.
3. Результаты методологических разработок и экспериментальных исследований открытых систем ЧПУ.
4. Результаты практической разработки геометрической, логической, терминальной и диагностической задач систем ЧПУ.

Методы исследования. Теоретические исследования базировались на теории интерфейсов, теории формальных грамматик, теории графов, теории конечных автоматов, теории алгоритмов; на методах объектно-ориентированного проектирования (декомпозиции, абстракции, иерархии) и методах компонентного моделирования и OLE (Object Linking and Embedding) автоматизации.

Научная новизна работы заключается в следующем:

1. Разработана новая концепция открытых, модульных, масштабируемых, переносимых, экономичных систем ЧПУ широкого назначения с выделенными в их архитектуре компонентами и оригинальным механизмом межкомпонентного взаимодействия. Концепция показывает пути решения разнообразных задач управления технологическими машинами как за счет различных компоновок и конфигураций системы из предложенных модулей, так и за счет дополнительного включения модулей конечных пользователей.
2. Созданы модели системы ЧПУ; разработана теория построения модульной архитектуры систем ЧПУ (на основе новой концепции), предполагающая последовательную трансформацию моделей для получения целостного описания системы.
3. Разработаны теоретические основы формального построения компонентной COM-модели системы ЧПУ, которые позволяют с единых методологических позиций

описывать COM-интерфейсы компонентов; систематизирован базовый набор интерфейсов.

4. Предложен методологический базис построения открытых систем ЧПУ и формирования окружения процесса разработки, предполагающий использование средств языковых процессоров, стандартных средств операционной системы, стандартных инструментальных средств и оригинальных инструментальных средств.

5. Разработан теоретический базис для создания системы ЧПУ по типу открытого языкового процессора и для формирования системы команд этого процессора.

6. Разработаны теоретические аспекты построения диспетчера реального времени и коммуникационной среды PCNC-системы на базе объектно-ориентированного и компонентного подходов.

7. Разработаны методические аспекты и создано «ноу-хау» в области реализации геометрической, логической, терминальной и диагностической задач открытой системы ЧПУ, в том числе:

a) формализован процесс построения основных модулей геометрического канала системы ЧПУ, модуля интерпретации и модуля интерполяции;

b) предложен подход к формализации языка ISO-7bit, позволяющий проектировать такие модули систем ЧПУ, которые способны воспринимать любую версию языка программирования;

c) предложен новый подход к проектированию электроавтоматики и к разработке диалога с оператором, который использует метод иерархических графов, позволяющий создавать визуальные среды разработки соответствующих приложений;

d) систематизированы и формализованы принципы построения интерфейса оператора;

e) предложен новый подход для создания диагностического и проблемно-ориентированного программного обеспечения, использующий концепцию виртуальных приборов.

Практическая ценность работы заключается:

- в разработке базовых понятий открытой модульной архитектуры систем ЧПУ широкого назначения;

- в разработке такой методики проектирования системы ЧПУ типа PCNC, которая обеспечивает:

a) возможность адаптации и настройки системы ЧПУ типа PCNC как к управ-

ляемому объекту, так и к технологическому процессу;

b) порождение версий систем ЧПУ без их перепрограммирования и перекомпиляции; динамическое изменение и реконфигурацию системы на основе стандартного механизма компонентного подхода;

c) возможность использования стандартных средств операционной системы и расширения реального времени, а также коммерческих инструментальных средств разработки программного обеспечения системы ЧПУ типа PCNC;

- в разработке основных модулей, а также полного прототипа открытой модульной системы ЧПУ типа PCNC на базе стандартной компьютерной аппаратуры, стандартной операционной системы с расширением реального времени, стандартного и оригинального инструментария.

Реализация работы. Научные и практические результаты внедрены:

- в Федеральном государственном унитарном предприятии «НИИ Автоэлектроника», в опытном производстве автотракторного электрооборудования.
- в учебной лаборатории кафедры “Компьютерные системы управления” МГТУ “СТАНКИН” при организации учебного процесса.

Кроме того, научные и практические результаты диссертации использованы при чтении лекций и руководстве курсовыми и магистерскими работами на кафедре компьютерных систем управления МГТУ “СТАНКИН”; а также в работе с аспирантами кафедры.

Апробация работы.

Теоретические и практические результаты, полученные автором, неоднократно докладывались на заседаниях кафедры компьютерных систем управления, а также на конференциях, в том числе международных.

Работа выполнена в Университете «СТАНКИН» на кафедре компьютерных систем управления при научной консультации профессора д.т.н. Сосонкина В.Л.

Публикации. По материалам диссертации опубликованы 34 печатных работы.

Структура и объем диссертации. Диссертационная работа состоит из введения и пяти глав, изложенных на 211 страницах машинописного текста; содержит 98 рисунка, 17 таблиц и список литературы из 181 наименовании. Общий объем работы 230 страниц.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

ВВЕДЕНИЕ

Мелкие и средние фирмы, составляющие большинство в развитых странах, нужда-

ются в производствах, которые обладают структурной, функциональной и технологической гибкостью. Такие производства создают на базе оборудования с ЧПУ, так как только системы ЧПУ располагают потенциалом обеспечения гибкости на всех стадиях жизненного цикла: у производителя, станкостроителя и конечного пользователя.

В настоящее время сокращается выпуск специализированных систем ЧПУ для станков разных технологических типов (для фрезерных, шлифовальных, электроэрозионных, лазерных станков и т.д.). Возрастание сложности систем ЧПУ, повышение технологических и потребительских требований к ним не позволяют производителям систем ЧПУ выпускать принципиально разные системы ЧПУ для каждой новой технологической задачи. Существует потребность в системах ЧПУ нового поколения, обладающих отрытой модульной архитектурой, конфигурируемых под разнообразные конкретные задачи.

Большой вклад в разработку теоретических и практических аспектов управления технологическим оборудованием внесли российские и зарубежные ученые: Горнев В.Ф., Кутин А.А., Митрофанов В.Г., Павлов В.В., Соломенцев Ю.М., Сосонкин В.Л., Фролов Е.Б., Притчев Г. (Pritschow G.), Спур Г. (Spur G.), Сторр А. (Storr A.) и многие другие. Проблема создания систем ЧПУ нового поколения вышла за пределы внутрипроизводственных и отраслевых задач и превратилась в национальную и межгосударственную. Важность и актуальность этой проблемы подтверждается тем, что за последние несколько лет разработан ряд национальных и международных проектов, с целью создания систем ЧПУ нового типа. В американском проекте OMAC, общеевропейских проектах OSACA и OPC; в немецком проекте HUMNOS, в японских проектах IROFA и OSEC приняли участие десятки национальных и международных компаний.

В рамках указанных проектов тенденции развития систем ЧПУ обозначены как коренное переосмысление потребительских свойств, структуры, архитектуры и математического обеспечения систем ЧПУ. В идеале любая современная система ЧПУ должна обладать открытой модульной архитектурой, в которой стандартизованы как отдельные модули, так и их интерфейсы. Подобная проблема до сих пор полностью решена не была. Именно персональные системы ЧПУ типа PCNC (Personal Computer Numerical Control) относятся к принципиально новому поколению систем управления. Они допускают конфигурирование на всех этапах жизненного цикла, позволяют интегрировать покупные программные продукты, обеспечивают непрерывную эволюцию в условиях максимальной независимости от базовой платформы, предос-

тавляют возможность доступа к информации о состояниях любого программного модуля системы, а также к диагностической информации аппаратуры, приводов и объекта.

Несмотря на наличие определенного опыта и немалого количества научных работ, можно констатировать следующее: отсутствует единая глобальная концепция построения реальных систем ЧПУ типа PCNC с открытой модульной архитектурой; не выработаны принципы и методы формализации проектирования и разработки подобных систем, которые бы в полной мере отвечали широкому спектру требований производителей, станкостроителей и конечных пользователей систем ЧПУ.

1. Глава 1. Современные системы ЧПУ. Тенденции и проблемы развития

Причины радикальных изменений потребительских свойств, структуры, архитектуры и математического обеспечения систем ЧПУ состоят: в резком повышении доли специального технологического оборудования; в расширении активности оператора непосредственно в цеху; в общем росте привлекательности персональных систем ЧПУ типа PCNC, поддерживающих такой стиль управления, который соответствует работе на компьютере. Однако есть и более глубокие причины: внедрение новых технологий, как, например, технологии объектно-ориентированного программирования и COM-технологии (Component Object Model), без которых создание программного обеспечения систем ЧПУ в объеме многих мегабайт попросту невозможно.

1.1 Современные достижения в области ЧПУ, обусловленные потребностями рынка

Оптимизация потребительских свойств машин, устройств и отдельных предметов, диктуемая рынком, неизбежно приводит к росту масштабов выпуска “нетехнологичных” деталей. Анализ обработки нетехнологичных деталей на станках с ЧПУ выделил такие особенности операций: сложные многомерные траектории инструмента, не укладывающиеся в привычные комбинации прямых и окружностей; нетрадиционный набор переходов, составление которого недоступно неискушенному конечному пользователю; экзотические рабочие процессы с привлечением новейших технологий; обилие в управляющих программах ЧПУ условных переходов, которые концептуально определяют сам стиль программирования в системах ЧПУ как условный.

Эволюция систем ЧПУ привела к таким их потребительским свойствам, о которых несколько лет назад можно было только мечтать. Современные системы ЧПУ обеспечивают скорости рабочей подачи до 60 м/мин и более, управляют обработкой поверхностей практически любой формы, реализуют нетрадиционные рабочие процес-

сы с привлечением новых технологий, используют новые типы интерполяции (сплайн-интерполяцию и ее разновидности), и сложные алгоритмы управления разгоном и торможением и алгоритмы сглаживания траекторий (получившие наименование Look-a-head – опережающий просмотр), используют языки высокого уровня. Системы ЧПУ предоставляют богатый сетевой сервис, начиная с простого подключения к внешней среде и вплоть до дистанционного диагностирования управления электроавтоматикой и приводами.

Подобные потребительские свойства накладывают жесткие временные ограничения на алгоритмы подготовки и отработки (интерполяции) кадров управляющих программ, а также требуют использования цифровых приводов SERCOS. Обработка поверхностями свободной формы нуждается в сложных алгоритмах и математическом обеспечении, а также в серьезных вычислительных ресурсах.

Современные тенденции построения систем ЧПУ нового поколения опираются на следующие особенности: аппаратная база предполагает технологические и стоимостные преимущества персональных компьютеров; программная реализация использует все достоинства объектно-ориентированного подхода; оператору предлагается привычный Windows-интерфейс; архитектурная организация предполагает эволюцию в сторону однокомпьютерной платформы; структура управления построена на основе принципов многоканальности; технологу-программисту предлагаются новые способы задания управляющих программ.

Основным достоинством современной системы ЧПУ является гибкость на всех стадиях ее жизненного цикла: у производителя, у станкостроителя, у конечного пользователя. Производители систем ЧПУ заинтересованы в некоторой *структурно гибкой* базовой системе управления, на основе которой можно строить многочисленные модификации соответственно запросам чрезвычайно подвижного рынка систем управления. Станкостроители отдают предпочтение *функционально гибкой* системе управления, которая наиболее просто адаптируется к фазовым пространствам станка и рабочего процесса. Конечные пользователи нуждаются в *технологически гибкой* системе ЧПУ, способной учитывать специфику конкретных рабочих процессов собственного производства.

Свойства гибкости необходимы, но не достаточны для построения концепции системы ЧПУ с открытой архитектурой. К другим важным признакам такой системы относятся: стандартизация модульной архитектуры и базовых внутренних структур данных, определение модулей в виде гибких командных процессоров. Они определяют наиболее значимые направления, на которые ориентированы основные междуна-

родные проекты по системам управления.

1.2 Пути эволюции, предлагаемые международными программами

Известные международные программы были направлены на создание систем ЧПУ с открытой архитектурой для обеспечивающих интеграцию программных модулей от разных поставщиков, интеграцию систем ЧПУ в информационном производственном пространстве, реализацию распределенных производственных систем. Конкретными целями были: привычное восприятие пользовательского интерфейса; уменьшение времени разработки и интеграции систем управления; обеспечение открытости для технических изменений и развития систем; снижение себестоимости и повышение надежности за счет использования стандартных PC-решений; создание единых спецификаций для всех проектов. При этом каждая программа была ориентирована на свои собственные задачи, которые как бы дополняли друг друга, но при отсутствии единой концепции.

Ключевым словом международных программ стала открытая архитектура, которая наиболее полно раскрывает потенциал новых функциональных возможностей систем ЧПУ. Важнейшие из них: конфигурирование; интегрирование покупных программных пакетов; эволюция систем управления; доступ к информации; стандартизация пользовательских интерфейсов; включение системы в сетевую коммуникационную среду.

Система ЧПУ может развиваться в двух направлениях. Быстрое решение состоит в использовании объектно-ориентированного подхода с разработкой обширных библиотек классов, виртуальных структур и абстрактных уровней в иерархической модели системы. Перспективное решение состоит в привлечении COM-подхода, подразумевающего крупномодульную архитектуру, совместимость на основе стандартизованных COM-интерфейсов и реализацию систем на базе стандартных механизмов распределенного функционирования.

2. Глава 2. Теоретические аспекты построения модульной архитектуры и модели систем ЧПУ типа PCNC

Основу предлагаемой далее концепции следует рассматривать не как альтернативу существующим представлениям о PCNC-архитектуре, но как естественное развитие наиболее плодотворных идей, сформулированных ранее. Ключевые архитектурные представления переосмыслены с учетом новейших достижений и тенденций развития индустрии ЧПУ и смежных областей, а также с учетом современных требований конечных пользователей.

2.1 Представление о модульной архитектуре системы ЧПУ

На прикладном уровне проблема модульности архитектуры системы PCNC остается открытой.

Обычно под модульностью на прикладном уровне понимают разбиение системы ЧПУ на функциональные части, разрабатываемые разными группами разработчиков. Такие системы практически не поддаются модернизации.

Новый взгляд на модульность, который предложен в работе, предполагает: формальную структуру модуля, наличие четко обозначенного интерфейса для каждого модуля и общего механизма межмодульного взаимодействия. Это приводит к новой архитектуре, в которой коммуникационная среда выполнена особым образом в виде канала - магистрали для обмена данными и командами.

Магистраль служит единым механизмом не только для внутреннего обмена между модулями, но и для обмена информацией с внешней средой.

Обеспечение межмодульной коммуникации подразумевает создание среды, включающей программные и аппаратные средства, средства связи и интерфейсы; использующей специфицированные форматы данных и протоколы. Эта среда в своей основе имеет развивающиеся, доступные и общепризнанные стандарты; обеспечивает высокую степень переносимости, коммуникабельности и масштабирования приложений и данных

Единый подход к реализации модулей предполагает использование клиент-серверных отношений при организации транзакций, привлечение объектно-ориентированного подхода к определению макро-структуры и на уровне технологии программирования.

Изложенное выше предопределяет принципиально иную (в сравнении с известными решениями) организацию системы ЧПУ, в которой даже модули с традиционными наименованиями имеют новое функциональное и алгоритмическое наполнение, а также и новую программную реализацию.

2.2 Прогноз эволюции архитектуры систем ЧПУ

В результате анализа эволюции архитектуры системы ЧПУ, которая имеет спиралеобразный характер развития, современные требования и тенденции в области ЧПУ, было спрогнозировано, что следующим этапом в развитии станет предложенный нами переход на конфигурируемую распределенную архитектуру. Это будет архитектура с выделенными компонентами и динамическими связями, возникающими непосредственно в процессе функционирования. В соответствии с выдвинутыми по-

зициями, компоненты системы устанавливают прямую связь между собой на необходимый период времени посредством коммуникационной среды.

Тем самым обеспечивается оптимальное использование вычислительных ресурсов, наряду с исключительно высокой гибкостью системы. Систему ЧПУ можно конфигурировать и реконфигурировать во время ее работы (run time), обеспечивать локальное или распределенное (удаленное) функционирование ее модулей, подключать систему к сетям управления на уровне предприятия.

Таким образом, далее будем говорить о системе следующего поколения, имеющей объектно-ориентированную реализацию, и о системе, через поколение построенной на базе компонентного подхода. При этом систему ЧПУ с компонентной реализацией построим на базе системы с объектно-ориентированной реализацией, обеспечив при этом преемственность базовых решений.

Рис. 1. Последовательная трансформация моделей систем ЧПУ

2.3 Принцип выделения модулей в архитектуре систем ЧПУ

Логическая схема последовательного создания и трансформации моделей систем ЧПУ типа PCNC для их полного описания и формализации представлена на Рис. 1.

Декомпозиция системы ЧПУ осуществляется с целью выделения задач и модулей, закрепленных за каждой из задач, а также составляющих модули блоков. Построение *архитектурной модели* определяет платформа системы. Создается виртуальная модель, а также определяются абстрактные модели для выявления иерархических зависимостей между платформой и прикладной компонентой. Построение потоковой модели систематизирует основные потоки данных в прикладной компоненте. Построение *объектно-ориентированной* модели увязывает модули системы ЧПУ и потоки данных между модулями в целостную систему. На базе объектно-ориентированной модели осуществляется реализация. Построение *компонентной модели* специфицирует интерфейсы компонентов и определяет уровень

независимого программного обеспечения от особенностей аппаратной реализации. Создание *клиент-серверной модели* определяет способ интеграции компонентов. Построение *распределенной модели* определяет сетевую схему функционирования компонентов.

Суть **принципа выделения модулей** заключается в выделении модулей как функционально законченных элементов системы ЧПУ, обладающих специфицированным интерфейсом взаимодействия между ними.

Рис. 2. Декомпозиция ядра системы ЧПУ типа PCNC

В системах ЧПУ нового поколения принято выделять системную платформу PC и ядро системы ЧПУ). Системная платформа строится, в основном, путем использования унифицированной PC-аппаратуры и стандартной операционной системы. Для ядра системы ЧПУ можно обозначить три уровня декомпозиции (см.*Рис.* 2).

Первый уровень состоит в выделении «задач управления». В числе подобных задач: геометрическая; логическая; технологическая; терминальная.

Второй уровень декомпозиции состоит в выделении модулей в составе задач управления; причем каждый отдельный модуль соответствует фазе решения задачи управления. Часть таких модулей имеет собственный прикладной интерфейс API; другие же сильно зависимые между собой, объединяются в группы с целью построения общего интерфейса API. Примером подобной группы может послужить геометрический “канал ЧПУ”, объединяющий интерпретатор и интерполятор. Объединение модулей в группы снижает нагрузку на коммуникационную среду; однако при этом уменьшается гибкость системы ЧПУ и ее способность к реконфигурации.

Третий уровень декомпозиции наиболее глубок, и в принципе, не обязателен. Он означает выделение блоков в составе модулей по их функциональному назначению.

2.4 Варианты архитектурных моделей систем ЧПУ

Сегодня особо гибкие и сложные системы ЧПУ типа PCNC с открытой архитектурой, ориентированные на многокоординатную, многостаночную, высокоскоростную, высо-

коточную обработку, - выполняют в виде двухкомпьютерной архитектурной модели (системы Typ3 osa (Bosch), Andronic 2000 (Andron) и Sinumeric 840 D (Siemens)). Выбор двухкомпьютерного решения обусловлен большей вычислительной мощностью по сравнению с однокомпьютерным вариантом и возможностью отделения терминальной задачи от задач реального времени на разных компьютерах. В перспективе преимущества на стороне однокомпьютерного архитектурного варианта, который предполагает меньшую себестоимость системы за счет исключения второго компьютера и его операционной системы и увеличивает надежность за счет исключения межкомпьютерной коммуникации, являющийся узким местом.

Вариант однокомпьютерной модели стал возможен сегодня, благодаря увеличению вычислительных мощностей современных процессоров и за счет использования в промышленных разработках новой шины CompactPCI.

Однокомпьютерная модель (см. Рис. 3) предполагает использование традиционного PC-компьютера, оснащенного специальными устройствами в виде плат-контроллеров.

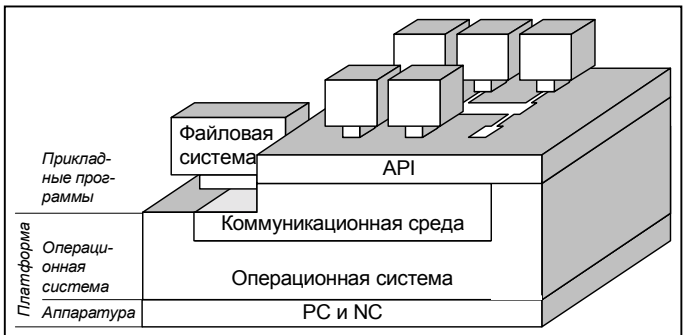

Рис. 3. Однокомпьютерный архитектурный вариант PCNC

Операционной системой сегодня должна быть Windows NT, в силу ее многозадачности, больших сетевых возможностей и мощного графического интерфейса.

Стремление к систематизации функциональных возможностей коммуникационной среды и к сокращению времени разработки прикладных модулей за счет использования сервисных механизмов, выдвигает задачу создания дополнительного слоя виртуальной шины для оказания высокоуровневых услуг. Дополнительный слой образует по сути глобальный сервер, единый для обеих подсистем (PC и NC).

Применение виртуальной шины в обеих архитектурных моделях создает некий промежуточный слой, отделяющий базовую платформу от прикладной компоненты; при этом различия архитектурных моделей остаются на различиях базовой платформы.

Основная задача построения архитектурной модели заключается в определении базовой платформы системы PCNC; в выделении коммуникационной среды в качестве платформо-независимого уровня абстракции; в формировании коммуникационной среды по типу виртуальной шины; в установлении необходимости надстройки виртуальной шины с целью выполнения функции глобального сервера системы.

2.5 Виртуальная модель системы ЧПУ

Формально структуру PCNC-системы в целом, а также структуру ее компонентов можно представить как некоторую совокупность трех элементов абстрактной модели: данные, команды и процессы. Каждый из этих элементов характеризуется набором свойств и набором функциональных возможностей, позволяющих осуществлять операции над элементом. Указанные элементы позволяют определить единообразные механизмы как для системы в целом, так и для ее компонентов. На базе элементов абстрактной модели построены: обмен между модулями, режимы системы и система команд PCNC-системы.

Первый элемент абстрактной модели – данные - характеризируется типом, размерностью, текущим значением и т.д. Функциональные возможности определяют способ их запросом.

Второй элемент абстрактной модели системы ЧПУ - процессы характеризируется текущим состоянием, списком предшествующих операций, а также набором предусмотренных акций. Управление акциями осуществляется синхронным или асинхронным способом. Понятие «процесс» отражает режимы системы ЧПУ (автоматический, ручной, режим запуска строки ручного ввода, толчковый режим и т.д.). Практически все модули системы ЧПУ так или иначе работают в этих режимах и, следовательно, связаны с процессом.

Третий элемент абстрактной модели - команды. Каждая команда имеет свой идентификатор, список параметров и приоритет. Предусмотренные акции команд состоят в проверке на возможность выполнения и в самом выполнении. Команды входят в сложные отношения между собой для образования системы команд модулей системы PCNC, построенной по типу языкового процессора.

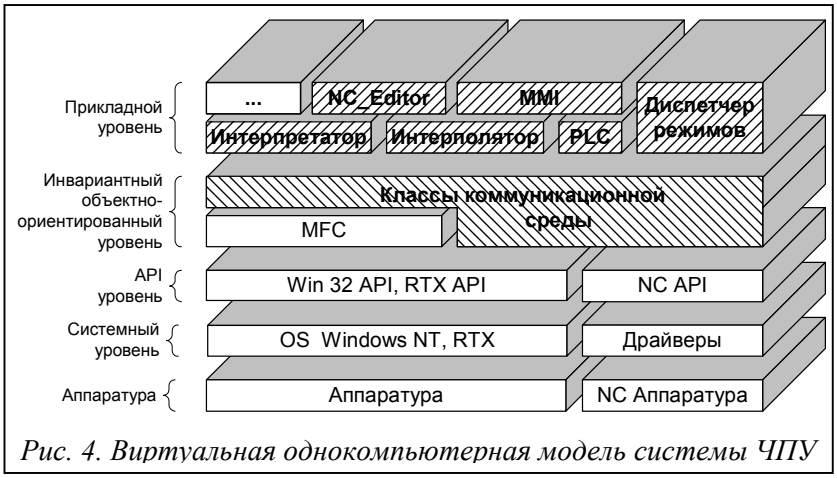

Рис. 4. Виртуальная однокомпьютерная модель системы ЧПУ

В «вертикальном сечении» система ЧПУ имеет многоуровневую структуру (см. Рис. 4) и соответствует модели виртуальной машины.

О каждом уровне виртуальной модели можно говорить как о независимом уровне абстракции, который может быть рассмотрен вне связи с другими уровнями.

На аппаратном уровне решают задачи выбора компонентов, их физической совместимости и т.д.

На системном уровне конфигурируют операционную систему Windows NT, встраивают расширитель реального времени RTX, инсталлируют (возможно, разрабатывают) драйверы для NC-оборудования.

На API уровне контролируют доступ к системному уровню и определяют спектр оказываемого сервиса.

На объектно-ориентированном уровне реализуют коммуникационные функции посредством виртуальной шины, а также функции глобального сервера системы.

На прикладном уровне решают конкретные прикладные задачи с использованием готовых решений и нижних уровней.

Выделение уровней виртуальной модели позволяет: переносить программное обеспечение на другие базовые платформы, снижать время разработки и сосредотачиваться на проблематике прикладных задач, обеспечивать независимость прикладного приложения от объекта управления. Неразумное увеличение числа уровней виртуальной модели создает опасность непроизводительного использования вычислительных ресурсов.

2.6 Потоковая модель данных системы ЧПУ (DFD-модель)

DFD-модель (Data Flow Diagram, диаграмма потоков данных) служит целям углубленной структуризации модулей системы PCNC, предшествующей объектно-ориентированному программированию этих модулей. DFD-модель является многоуровневой иерархией.

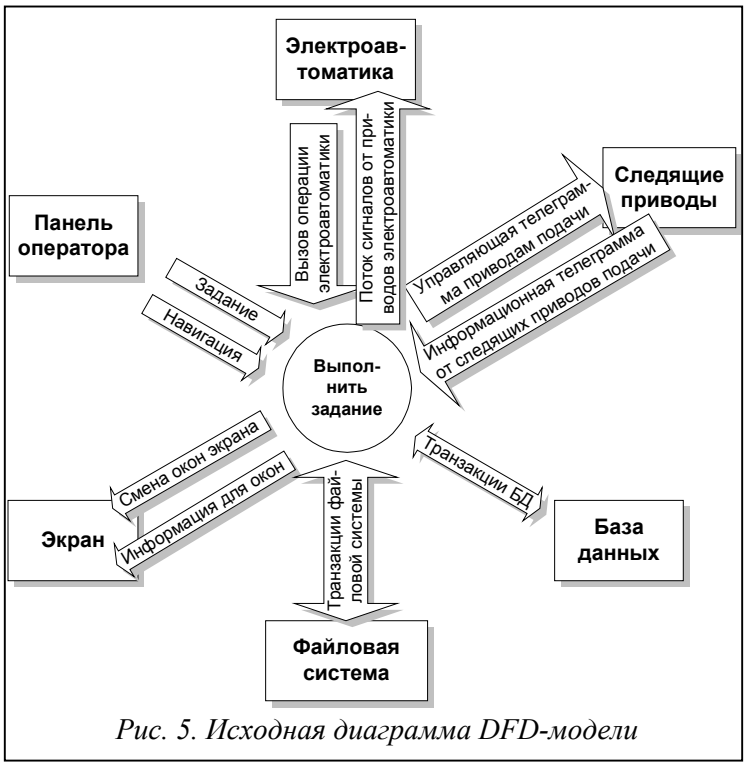

Рис. 5. Исходная диаграмма DFD-модели

На Рис. 5, приведена исходная диаграмма, в которой вся система PCNC представлена (в кружке) глобальным процессом “Выполнить задание”, а внешние источники и стоки (т.е. адресаты) данных (в прямоугольниках) составляют окружение системы PCNC. Потоки данных (обозначены стрелками с именами потоков) моделируют передачу информации, которая может быть одно- или двунаправленной. Назначение процесса (здесь процессом является глобальная функция системы PCNC) состоит в продуцировании из входных потоков выходных в соответствии с действием, заданным в имени про-

цесса.

Процесс «Выполнить задание» декомпозирован на два других, соответственно PC-подсистеме (процесс «Разработать задание») и NC-подсистеме (процесс «Управлять объектом»). В результате декомпозиции вскрылись новые потоки данных между двумя подсистемами (дальнейшая декомпозиция не отражена).

2.7 Объектно-ориентированная модель системы ЧПУ

Объектно-ориентированная модель использует объекты и отношения между ними, сохраняя семантические связи между функциями и соответствующими данными. Модель системы имеет иерархическую структуру, что позволяет на любом уровне иерархии сконцентрировать внимание на конкретных элементах модели.

Структура системы ЧПУ представлена на Рис. 6 как некоторая совокупность базовых модулей (обведены сплошными линиями) и дополнительных модулей (обведены пунктирными линиями). При этом каждый из модулей закреплен за определенной задачей управления. Модуль автономен и является вложенным объектом (embedded object), т.е. обладает алгоритмической структурой и структурой данных; а также и интерфейсной оболочкой, ориентированной на интерактивную работу в клиент-серверной среде.

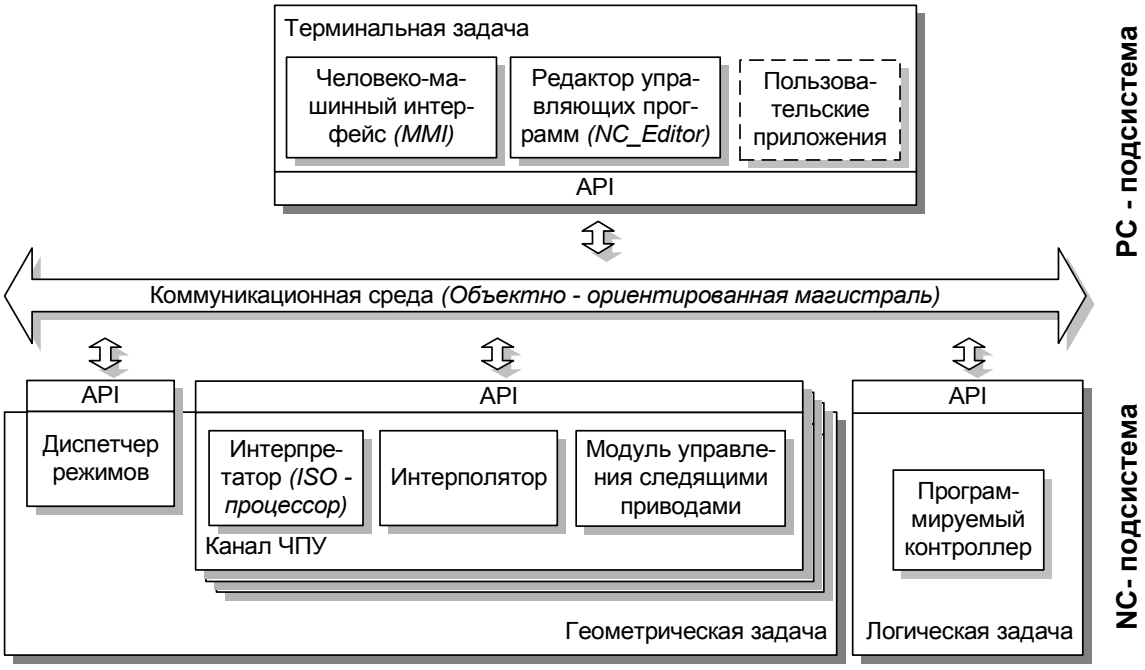

Рис. 6. Построение модели на базе общей объектно-ориентированной магистрали

Модель представлена двумя подсистемами - NC-подсистемой и PC-подсистемой. Первая является ведущей и формирует среду функционирования ЧПУ-ориентированных модулей в реальном времени. PC-подсистема образует среду Windows-образного интерфейса пользователя.

Взаимодействие модулей осуществляется посредством коммуникационной среды, реализованной по типу объектно-ориентированной магистрали, которая не только поддерживает программно-аппаратные коммуникационные протоколы, но и выполняет прикладные серверные функции, благодаря специальной объектной оболочке. Иерархию классов PCNC-системы определяют основные принципы объектно-ориентированного подхода: инкапсуляция, наследование, полиморфизм. В основе этой иерархии лежат классы, реализующие элементы абстрактной модели: данные, процессы, команды.

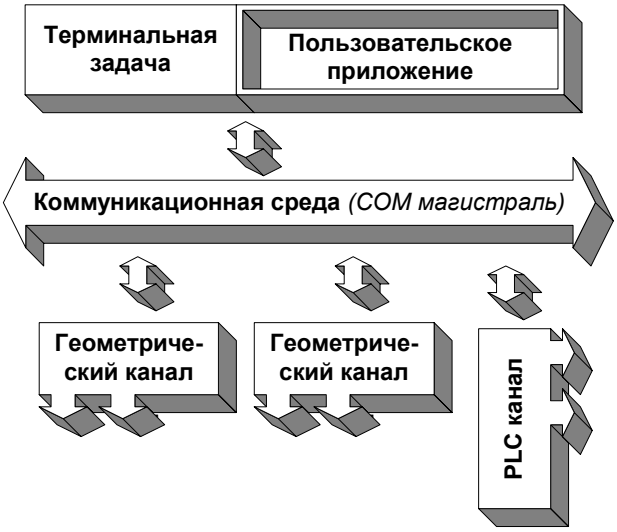

Рис. 7. Модель системы ЧПУ на базе общей COM магистрали

Объектно-ориентированный подход предоставляет ряд стандартных формализмов для описания модели. На базе стандартного формализма - диаграммы классов, спроектирована структура модуля PCNC в терминах объекта.

Для выделения данных в системе PCNC сформулирован принцип, согласно которому **основой выделения данных системы PCNC в классы служит неразрывное (единое) представление о данных и их функциональных характеристиках (способах запросов к ним).** В соответствии с принципом выделения данных, объект, представляющий данные PCNC-системы, имеет такие характеристики, как: тип, размер, текущее значение; а также функциональность, позволяющая запрашивать эти данные разными способами, форматировать данные, конвертировать, верифицировать и т.д.

Механизм отображения данных раскрыт с помощью стандартных формализмов: диаграммы взаимодействия и диаграммы объектов.

2.8 Компонентная модель системы ЧПУ

В рамках COM архитектуры (Component Object Model) выделены: раздел базовой платформы, представленный компонентами; и раздел прикладного математического обеспечения, использующий компоненты. Базовая платформа поддерживает выполнение прикладных программ, оказывая при этом свои услуги через COM-интерфейсы. На **Рис. 7** показана масштабируемая система PCNC, где количество геометрических каналов определяется текущей конфигурацией. Каждый элемент базовой платформы имеет сложную структуру и выполнен в виде компонента.

Геометрический канал осуществляет интерпретацию и интерполяцию кадра управляющей программы, а также управление приводами. PLC канал управляет электроавтоматикой станка и синхронизирует работу со вспомогательным оборудованием, таким, как робот, устройство смены палет и т.д. Терминальный модуль выполняет функции интерфейса с пользователем (MMI). Здесь реализованы рабочие экраны, а также машина состояния (State Machine), непосредственно реагирующая на действия оператора и изменения состояний системы PCNC.

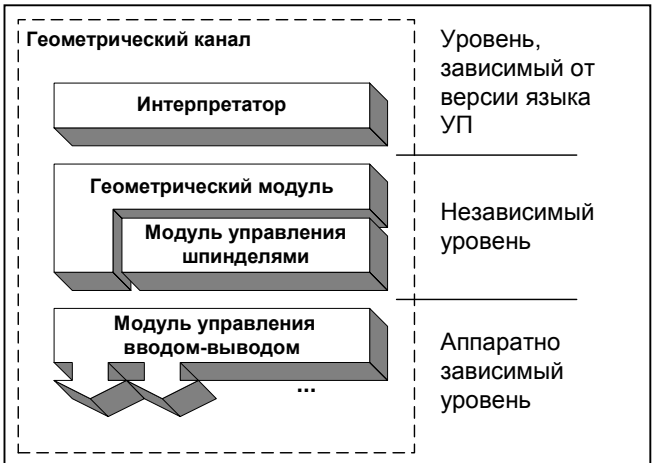

Рис. 8. Вертикальная структура геометрического канала

В обобщенной формулировке назначение канала (геометрического или PLC) заключается в трансформации данных и их передаче. Внутренние связи модулей канала обычно сильнее внешних. Канал представляет собой законченный программно-аппаратный узел, поставляемый в целом одним поставщиком для передачи данных в определенных направлениях (приводы подачи, приводы электроавтоматики и т.д.).

На примере геометрического канала показана его многоуровневая структура (см. Рис. 8). В ней выделен модуль «*управление вводом-выводом следящих приводов*», который маскирует от других уровней аппаратные особенности приводов. Адаптация геометрического канала к другому типу приводов состоит в замене модуля управление вводом-выводом.

Следующий уровень независим от типа приводов, а также независим от версии языка управляющей программ. Программное обеспечение этого уровня одинаково для всех систем PCNC.

Еще выше расположен «*интерпретатор*», который учитывает специфику разнообразных версий языка ISO-7bit. Замена интерпретатора позволяет адаптировать работу геометрического канала к другой версии языка УП.-

Модули геометрического канала, оформлены в виде отдельных компонентов, и каждый из них имеет некоторый набор системных и прикладных интерфейсов.

В работе создана клиент-серверная модель PCNC системы с выделенным глобальным сервером для решения проблемы системной интеграции модулей системы. Специфика модулей системы ЧПУ заключается в том, что должны быть одновременно и серверами и клиентами.

Выделение общего для системы сервера, позволило клиенту получать все необходимые ему данные и услуги. Выделенный сервер агрегирует в себе функциональные возможности всех компонентов системы PCNC и оказывает необходимый сервис любому модулю, выступающему в роли клиента. Анализ архитектурной, потоковой, объектно-ориентированной и компонентной моделей систем PCNC, выявил, что коммуникационная среда, через которую проходят практически все потоки данных, является наилучшим глобальным сервером.

При выделении общего PCNC COM-сервера системы была выдвинута и реализована идея динамической агрегации, когда один компонент расширяет свои функциональные возможности за счет возможностей второго во время работы (run time).

2.9 Модель распределенной системы ЧПУ

Распределенная архитектура системы ЧПУ предполагает работу отдельных ее модулей (компонентов) на разных машинах в сети. Отдельной машиной может быть удаленный терминал в локальной сети, с помощью которого можно управлять несколькими PCNC-системами. Другой пример, - диагностика системы PCNC, осуществляемая из диагностического центра по сети Internet. В обоих случаях PCNC служит сервером, поскольку предоставляет сервис другим системам в сети. Но возможен и другой случай, когда PCNC является клиентом как например, при выполнении сложных CAM-расчетов в вычислительном центре по корпоративной сети (см. Рис. 9).

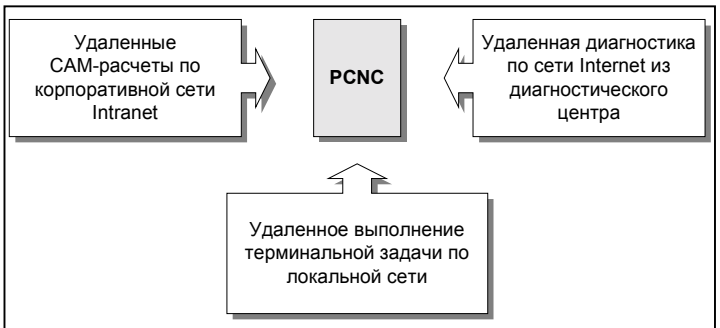

Рис. 9. Пример распределенного функционирования PCNC системы

Распределенность системы PCNC означает: обеспечение функционирования ее компонентов на удаленных машинах; возможность перенаправления компонентов к удаленной системе, т.е. запуск модуля на удаленной системе; возможность расширения системы и включения новых модулей только за счет конфигурирования.

Модель распределенной системы полностью соответствует компонентной модели системы ЧПУ. Различие состоит в необходимости реализации компонентов в виде исполняемых модулей (*.exe файлов) и необходимости создания регистрационной записи на этапе конфигурации системы. Проблемы межпроцессного обмена в распределенной системе решаются на уровне DCOM (Distributed Compomponet Object Model - распределенная COM-модель), т.е. на уровне операционной системы, в которой DCOM-модель реализована.

3. Глава 3. Методологические аспекты построения открытых систем ЧПУ

Теоретические результаты этой главы имеют своей целью сделать процесс разработки системы PCNC регулярным с учетом специфики каждого экземпляра системы управления.

3.1 Представление о системе PCNC как об открытой системе управления

В Табл. 1 систематизированы уровни открытости PCNC системы.

Табл. 1. Открытость системы PCNC на основных этапах жизненного цикла

№	Уровни открытости	Описание
1	Открытость на уровне производителей	Возможность встраивания готовых решений (аппаратных, программных, программно-аппаратных решений) в базовую платформу Определение набора API функций, который открыт и доступен для использования.
2	Открытость на уровне станкостроителей	Создание собственной версии языка управляющих программ; определение собственного набора стандартных циклов; реализация отдельных собственных алгоритмов интерполяции. Встраивание собственных или покупных диагностических комплексов. Реализация собственных интерфейсов оператора (MMI); включение коммерческих приложений; построение информационно-технологических подсистем.
3	Открытость на уровне конечных пользователей	Расширение интерфейса оператора, настройка на технологический процесс. Использование собственного технологического “ноу-хау”.

Реально, открытость систем PCNC находится в руках производителей, но не пользователей. Использование API-функций доступно или производителям систем ЧПУ, или квалифицированным коллективам технических университетов.

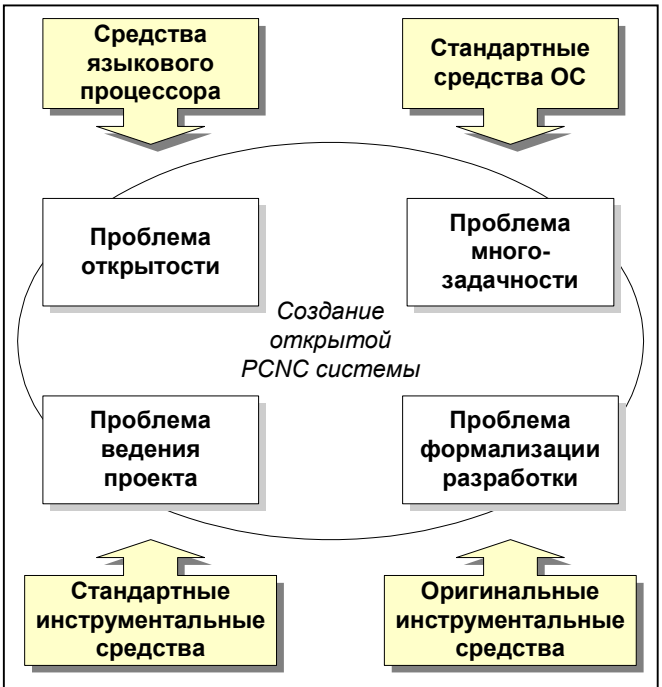

Рис. 10. Методологические проблемы создания открытых PCNC систем

Когда мы говорим об открытости на уровне пользователя, то, в первую очередь, подразумеваем: возможность использования API-функций, вынесенных во входной язык; возможность настройки с помощью конфигурационных файлов и файлов инициализации; возможность изменения значения реестра Windows NT; возможность добавления и синхронизации внешних приложений

на базе входных и выходных файлов. И лишь во вторую очередь рассматриваем возможность получения дополнительного сервиса от ядра системы PCNC через API-интерфейс.

При построении систем ЧПУ помимо проблемы открытости, нерешенными остаются проблемы реализации многозадачности прикладной компоненты, проблемы контроля над ведением проекта, проблемы формализации рутинных работ при разработке и настройке системы для конкретного заказчика.

При систематизации средств построения открытых PCNC систем выделены четыре основные группы средств (см. Рис. 10), соответствующие решению ключевых методологических проблем: средства языкового процессора, стандартные средства операционной системы, стандартные инструментальные средства и оригинальные инструментальные средства.

3.2 Опыт и методика построения систем ЧПУ по типу открытого языкового процессора

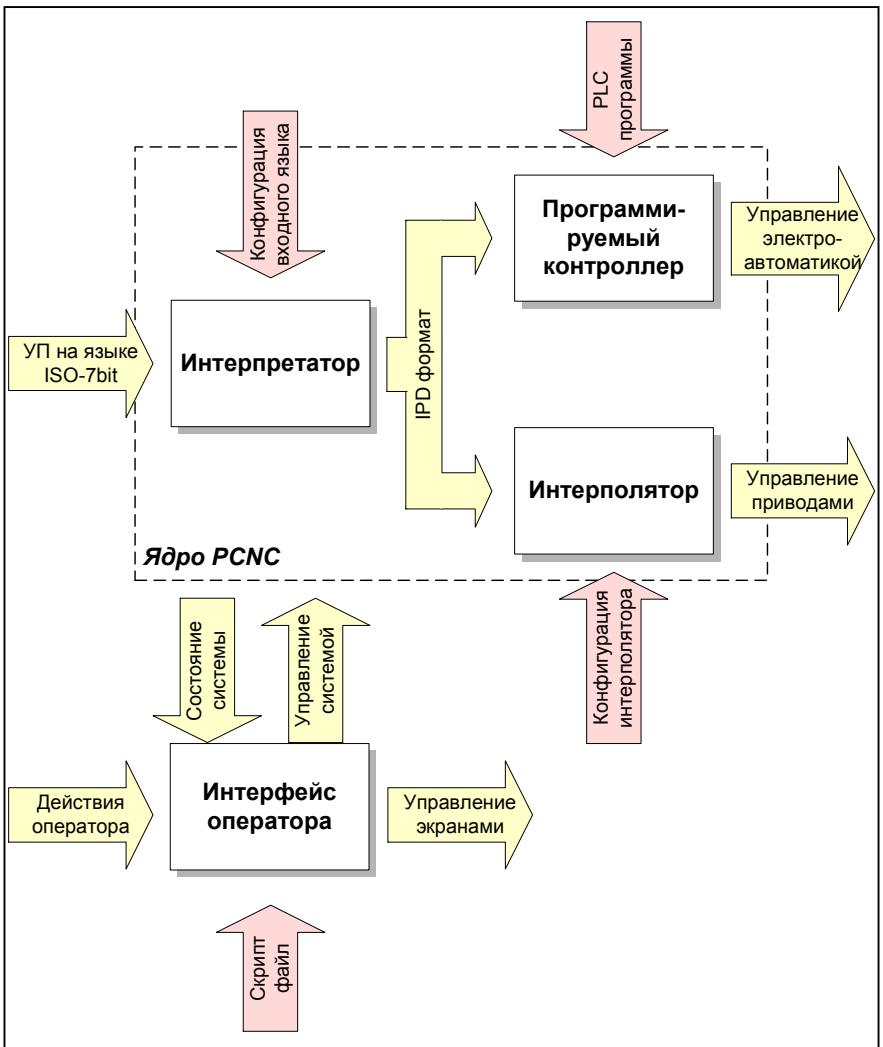

Рис. 11. PCNC-система как набор языковых процессоров

Под открытым языковым процессором будем понимать модуль, у которого хотя бы часть API-функций вынесена во входной язык или в файл конфигурации модуля, что позволяет использовать функциональные возможности ядра системы ЧПУ, не прибегая к прямому программированию. Примером служит реализация специфичных измерительных циклов G581-G589 в системе.

На Рис. 11 система PCNC представлена в виде упрощенного набора языковых процессоров. На вход интерпретатора поступают управляющие программы на языке ISO-7bit; настройка на конкретную версию

языка ISO-7bit осуществляется с помощью специального конфигурационного файла. Выходом интерпретатора является специальный IPD-файл, содержащий команды для интерполятора и программируемого контроллера. Интерполятор в соответствии с заданной конфигурацией обрабатывает IPD-команды и выдает приращения на приводы. Аналогично программируемый контроллер в соответствии с заложенными программами обрабатывает свои IPD-команды и управляет электроавтоматикой.

Экран, расположение управляющих элементов, размеры, шрифты, цвета, язык выдачи сообщения и соответствующая машина состояния задаются файлами описания (скрипт-файлами, script). Интерфейс оператора интерпретирует скрипт-файлы состояния системы в соответствии с действиями оператора и для отображения экранов и управления системой PCNC.

Функциональность системы PCNC определяется во многом геометрической задачей. Здесь увеличение резервов гибкости и ресурсов быстродействия немедленно сказывается на потребительских качествах системы управления в целом. Поэтому была разработана концепция интерпретатора управляющей программы в виде геометрического ISO-процессора. Суть концепции состоит в привлечении двух принципов организации программной среды для решения геометрической задачи.

Согласно первому принципу ISO-процессор должен быть построен таким образом, чтобы воспринимать операторы языка ISO-7bit управляющей программы так, как если бы они были машинными инструкциями. Второй принцип заключается в том, что реализация ISO-процессора осуществляется на основе объектно-ориентированного подхода.

Кадр управляющей программы на языке ISO-7bit несет информацию о заявленных алгоритмах и структурах данных (см. Табл. 2). Алгоритмы представлены подготовительными функциями (G-функциями). Структуру данных составляют функции размерных перемещений (X, Y, Z, I, J, K, R), функция подачи (F), функция скорости главного движения (S). Функции структур данных можно рассматривать как параметры G- функций, а сами G-функции как системы команд ISO-процессора.

Табл. 2. Структура кадра управляющей программы в терминах концепции ISO-процессора

Данные		Алгоритмы	
наименование	обозначение	наименование	обозначение
Размерные перемещения	X, Y, Z, I, J, K, R	Подготовительные функции	G
Подача	F		
Скорость главного движения	S	Вспомогательные функции	M
Управление последовательностью кадров	L		

Все команды разбиты на несколько групп, определяющих их функциональное назна-

чение. В каждой группе выделены одна или несколько подгрупп ортогональных (т.е. взаимоисключающих) G-функций. Интерпретация кадра для каждой группы осуществляется независимо.

Конфигурация интерпретатора формируется структурой системы команд. Число групп G-функций задает количество групповых интерпретаторов; а в каждом интерпретаторе предусмотрены подгруппы, с которыми он работает, но не определены G-функции, которые входят в каждую подгруппу. Групповой интерпретатор обращается к подготовительной функции как к соответствующей координате G-вектора и передает ей управление. Использование подобной схемы означает, что одно и то же устройство ЧПУ может использовать различные системы команд.

3.3 Опыт и методика использования стандартных средств для поддержания открытой архитектуры

Расширение RTX заменяет аппаратно зависимый уровень HAL (Hardware Abstraction Layer) ядра Windows NT для того, чтобы гарантировать необходимое время реакции на события. При этом появляются новые объекты ядра с идентичными для прежних объектов механизмами взаимодействия, что и позволяет рассматривать Windows NT и расширение RTX как одно целое, т.е. операционную систему реального времени.

В системе PCNC существуют два или более процессов, которые могут быть процессами реального времени и процессами машинного времени; при этом процессы обмениваются данными и сообщениями.

В процессах реального времени в виде отдельных потоков должны быть реализованы: интерполятор, модуль Look-a-head, система поддержки коммуникационной среды. В процессе машинного времени реализуют потоки поддержки коммуникационной среды, терминальную задачу (MMI); интерпретатор, запускающий свои групповые интерпретаторы как отдельные процессы.

Многопоточность позволяет добиться минимального простоя процессора, и, следовательно, более эффективной его работы. Например, прорисовка сложного экрана MMI может занять слишком много времени, что недопустимо для систем реального времени. Выделение MMI в отдельный поток с низким приоритетом снимает эту проблему. При возникновении любой паузы ОС передаст MMI время для прорисовки экрана. Срабатывание таймера запустит высокоприоритетный поток интерполятора, и MMI-поток будет вытеснен.

В отдельные потоки выделяют любые длинные операции с файлами. Например, в NC-редакторе в отдельные потоки выделены операции: 3D-моделирования управляющей программы, расчета G-вектора в текущем кадре управляющей программы и

т.д.

Выделение модулей системы PCNC в отдельные потоки обеспечивает их независимость друг от друга. Например, интерполятор, работающий в отдельном потоке, не зависит от фазы подготовки кадра интерпретатором, работающим в другом потоке. Непрерывность работы интерполятора обеспечивается буферизацией кадров между интерпретатором и интерполятором. Следует, однако, учитывать, что при выделении большого количества потоков возрастает время на их переключение. Особенно заметен этот эффект в быстрых процессах, таких как интерполяция.

Потоки являются мощным инструментом, который следует применять осознанно. Существуют ситуации, в которых следует избегать создания потоков. Нельзя, например, располагать управляющие элементы (control-элементы) пользовательского интерфейса в отдельных потоках, посколку выделение потока неизбежно породит Windows NT проблему синхронизации при обработке сообщений в других потоках.

Использование многопоточности в системе PCNC требует решения таких проблем, как работа с разделяемой памятью, синхронизация потоков, работа с разделяемыми ресурсами. Наши решения и предложения построены на стандартных объектах операционных систем (критические секции, мютексы, таймеры, семафоры, события), большая часть из которых являются объектами ядра.

Использование разделяемой памяти с помощью критической секции (critical section) рассмотрим на примере интерпретатора и интерполятора. Интерпретатор интерпретирует кадр и записывает его в формате IPD (Interpolator Data) в кольцевой буфер. Интерполятор считывает кадр из кольцевого буфера и интерполирует его. На этапе записи и считывания кольцевой буфер должен быть заблокирован, чтобы невозможно было прочитать неполный кадр или переписать не полностью прочитанный кадр. Для этой цели создается критическая секция и инициализируется, например, в главном потоке процесса. В обоих потоках код, который должен быть защищен, обрамляется вызовом функции EnterCriticalSection() в начале и функции LeaveCriticalSection() в конце. Если передача управления была осуществлена в интервале критической секции, то второй поток не сможет войти в эту же критическую секцию, и передаст управление первому потоку для выхода из критической секции.

Другим средством синхронизации потоков служит объект ядра - мютекс (mutex), позволяющий синхронизировать потоки разных процессов. Рассмотрим пример синхронизации данных коммуникационной среды между процессом реального времени и процессом машинного времени. Данные передаются через разделяемую память (Shared memory). В коммуникационном потоке процесса реального времени создает-

ся объект ”мютекс” с помощью функции CreateMutex(), которая сразу возвращает описатель (handler) объекта. Win32 поток получает описатель того же объекта ”мютекс” с помощью функции CreateMutex() или OpenMutex(). Код в обоих синхронизируемых потоках обрамляется функцией WaitForSingleObject() в начале и функцией ReleaseMutex() в конце. Таким образом, одновременно блокируются попытки записи и считывания из разделяемой памяти соответственно коммуникационных потоков реального времени и машинного времени.

Объект «семафор» (semaphore) используют для управления ресурсами. В PCNC системе – это порты ввода-вывода. Когда запрашивается ресурс, то операционная система уменьшает содержание счетчика ресурсов. Семафор создается с помощью функции CreateSemaphor(), в которой указано количество ресурсов, подлежащих мониторингу. Описатель семафора из другого потока создается с помощью той же функции CreateSemaphor() или функции OpenSemaphor(). Аналогично с объектом ”мютекс” синхронизация происходит в блоке, обрамленном функциями WaitForSingleObject() и ReleaseSemaphor(). Следует учитывать тот факт, что мютекс может освободить только поток, занявший мютекс, а семафор может быть освобожден любым потоком.

Существует и другой объект ядра для синхронизации потоков - событие. В PCNC системах событие можно использовать, когда код инициализации системы или код выхода из ошибки выводится в отдельном потоке. Инициализирующий поток переводит объект-событие в состояние «занято» и приступает к своим операциям. В этот момент рабочий поток приостанавливает свое исполнение и ждет. По окончании инициализации поток инициализации возвращает объект-событие в состояние «свободно». Рабочий поток активизируется и продолжает свою работу.

Использование объекта «событие» (event) позволяет упростить схему диспетчеризации процессов реального времени. Диспетчер процессов реального времени построен на базе таймера. По сути, это отдельный поток с наивысшим приоритетом. При каждом срабатывании таймера вызывается его call-back функция, в которой реализована схема диспетчеризации. Согласно этой схеме управление потоками осуществляется путем изменения их приоритетов. На этом работа call-back функции таймера завершается. Останов и запуск процессов осуществляет сама операционная система.

Предложены решения важных проблем при создании открытых систем PCNC, построены на использовании системных возможностей операционной системы Windows NT, на ее концепциях многозадачности, механизма сообщений и объектов син-

хронизации. Использование отлаженных решений и опыт, накопленный в этой области, позволяют существенно снизить трудоемкость разработок.

3.4 Опыт и методика использования стандартных инструментальных средств для поддержания открытой архитектуры

Опыт лаборатории кафедры компьютерных систем управления МГТУ «СТАНКИН» показывает, что интеграция инструментальных средств возможна на основе технологии Microsoft-OLE-Automation. В качестве базы был использован Visual C++. Сегодня Visual C++ входит в состав самого мощного пакета средств разработки программного обеспечения под Windows - MSDN (Microsoft Developer Network).

В проекте PCNC системы для Visual C++ каждый модуль оформлен в виде подпроекта. Подпроекты состоят в некоторой зависимости друг с другом; зависимость эта определяет очередность при компиляции. Суть состоит в том, что во время компиляции модуль PCNC должен иметь уже те скомпилированные модули, от которых он зависит. Существует и основной проект, компиляция которого обеспечивает перекомпиляцию всех остальных модулей. Каждый модуль проекта PCNC оформлен физически как динамическая библиотека и имеет обособленное логическое назначение.

Visual C++ обладает мощным механизмом для настройки средств разработки; утилитами, обслуживающими проект; механизмами встраивания внешних приложений. Система поддержки проекта Source Safe и система проектирования Rational Rose встраиваются в Visual C++ как внешние приложения.

В проект создания PCNC системы вовлечено несколько десятков человек; постоянно вносятся изменения; нужна поддержка релизов и дальнейшее развитие версий. Такой проект нельзя начинать без системы Source Safe (компании Microsoft) поддержки проекта. Source Safe реализует достаточно простую концепцию работы с исходными кодами, поскольку только один разработчик может взять файл из Source Safe с помощью функции CheckOut; по окончании работы разработчик возвращает файл обратно в Source Safe, используя функцию CheckIn.

Иерархия классов, которая была создана на промежуточном этапе разработки коммуникационной среды в виде объектно-ориентированной магистрали, содержит более 150 классов, поэтому архитектор проекта не в состоянии отследить взаимосвязь между ними и довести до программиста детали их реализации. Проблемы подобного рода решаются с помощью CASE-системы Rational Rose, поддерживающей практически все распространенные нотации представления моделей, в том числе и UML (Unified Modeling Language).

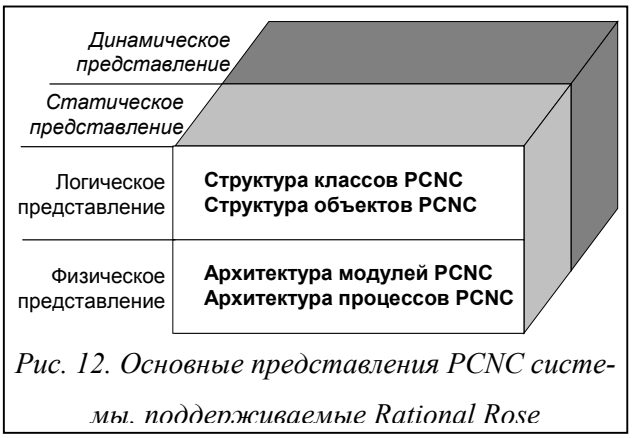

Рис. 12. Основные представления PCNC системы, поддерживаемые Rational Rose

Прикладные компоненты описываются системой Rational Rose с помощью четырех представлений объектно-ориентированного подхода (см. Рис. 12). Логическое представление определяет ключевые абстракции и механизмы, формирующие предметную область и архитектуру PCNC системы. Физическое представление определяет конкретную программно-аппаратную платформу, на которой реализована система. Все это описывается диаграммой классов, диаграммой объектов, диаграммой модулей, диаграммой процессов. Если рассматривать систему со стороны статики/динамики, то здесь интерес представляют: диаграммы переходов из одного состояния в другое, диаграмма взаимодействия и обмена сообщениями, определяющая порядок их передачи.

3.5 Опыт и методика использования оригинальных инструментальных средств для поддержания открытой архитектуры системы ЧПУ

Стандартные средства операционной системы и стандартные инструментальные средства недостаточны для разработки PCNC системы.

Опыт нашей лаборатории показывает, что ряд задач в PCNC-системах возможно формализовать. С точки зрения трудоемкости и требуемой квалификации программиста наиболее актуальны: создание каркаса модулей или внешних приложений, подключаемых к объектно-ориентированной магистрали; создание каркаса машины состояния любого модуля или внешнего приложения PCNC системы. Во всех случаях формализации части процесса разработки PCNC системы нужно создавать собственный инструментарий.

Под созданием каркаса программного модуля понимаем добавление в проект (или подпроект) тех классов и их объектов, которые реализуют базовые функциональные возможности, практически однотипные для подобных же программных модулей.

AppWizard представляет средство создания и конфигурирования нового проекта в среде Microsoft Visual C++. В частности, NCs AppWizard определяет в интерактивной форме свойства и функциональные возможности будущего модуля PCNC.

Скелет модуля создается на базе классов коммуникационной среды, что позволяет расширить возможности применения библиотеки классов OOCL путем генерации большей части необходимых программных кодов. Сведения о создаваемом приложении запрашиваются в форме диалога с пользователем, и генерируется исходный

код.

Установка Ncs AppWizard в систему разработки состоит в копировании файла с расширением «.AWX» в директорию Template Microsoft Visual C++.

Создание нового приложения при помощи Ncs AppWizard представляет собой обработку последовательно сменяющихся диалогов, предназначенных для ввода пользователем информации о новом приложении.

Под машиной состояния (State Machine) в системе управления понимают конечный автомат, реагирующий на управляющие воздействия и вызывающий необходимые действия. Управляющими воздействиями могут быть действия со стороны оператора (нажатие им кнопки), события (например, поступление сигнала от датчика), возникновение ошибки. В результате реакции на управляющее воздействие машина состояния переходит из одного состояния в другое, при этом выполняются функции, предусмотренные при переходе.

Машина состояния описывается с помощью графа. Это и было использовано при построении визуального инструментария разработки машины состояния State Machine Builder (генератора машины состояния). При этом были использованы дополнительные понятия, такие, как: сложные состояния, порты ввода/вывода.

Рис. 13. "Окружения разработки" PCNC системы

Эти понятия позволили выстраивать иерархические структуры графа; причем в начале иерархии находится единственное сложное состояние. На каждой следующей ступени иерархии сложные состояния распадаются на совокупности простых или простых и сложных состояний.

Инструмент разработки State Machine Builder позволяет визуально представить машину состояния в виде

иерархического графа, задать имена классов, реализующих состояния и переходы; определить файлы их расположения. В результате инструментарий генерирует C++ код для машины состояния и встраивает его в Visual C++ проект PCNC системы.

3.6 Формирование окружения разработки

Стадии разработки сопоставим системе решений, рассмотренных в разделах 3.3, 3.4, 3.5. Соответственно, Rational Rose и State Machine Builder закрепим за фазами анализа и проектирования; NCs AppWizard закрепим за фазами проектирования и разработки, а Visual C++ и Win32 API закрепим за фазой разработки. В результате получим целостную систему, наполняющую все фазы разработки и поддерживающую ведение проекта. Систему эту назовем “окружением разработки” открытой PCNC системы (см. Рис. 13). В окружение разработки, помимо рассмотренные механизмов и инструментов, можно ввести дополнительный инструментарий, например, систему ведения документации, что показано на рисунке серым цветом. Интеграция всего окружения осуществлена на базе MSDN.

В результате исследования сформирована методология решения проблемы создания открытой системы PCNC.

4. Глава 4. Теоретические аспекты построения диспетчера реального времени и коммуникационной среды

Декомпозируем проблему интеграции модулей в PCNC-систему на две части. Первая часть связана с встраиванием модулей в единую логическую структуру, в соответствии с работой системы. Вторая часть связана с встраиванием в единый механизм передачи данных и общения между ними. Если первая проблема решается на уровне планирования и диспетчирования задач, то вторая решается на уровне коммуникационной среды.

4.1 PCNC как система реального времени

На основе анализа систем реального времени, проведена обобщенная классификация и обозначено в ней место PCNC-системы. Современные системы числового программного управления используют операционную систему *Windows NT с расширением реального времени*.

Windows NT не является операционной системой реального времени (ОСРВ), поскольку: она не имеет достаточного диапазона приоритетов потоков; она не позволяет управлять наследованием приоритетов; механизм синхронизации потоков непредсказуем; время реакции на прерывание непредсказуемо. В силу растущей популярности операционной системы Windows NT управления проблема реального вре-

мени в системах должна быть решена.

Были выработаны основные правила реализации системы реального времени на базе Windows NT, которые сведены в Табл. 3.

Из всех существующих предложений по реализации ОСРВ на базе Windows NT практическое значение имеют всего два подхода. Первый подход состоит в запуске Windows NT в виде низко-приоритетной задачи операционной системы реального времени (супервизора); при этом предполагается применение ядра классической ОСРВ типа QNX или VxWorks.

Табл. 3. Правила реализации системы реального времени на базе Windows NT

Правила	Описание
1. Реализация разрешения реального времени.	Речь идет о времени реакции и времени переключение контекста, а также о требованиях, необходимых для обеспечения предсказуемости.
2. Обработка исключения сбоя Windows NT.	При “крахе” Windows NT генерируется исключение «blue screen», после чего возможна только перезагрузка системы.
3. Защита от некорректных Windows-драйверов и приложений.	Установка драйвера или приложение независимого поставщика может привести к «зависанию» системы, что абсолютно недопустимо.
4. Обработка сбоев жесткого диска	Программный механизм должен гарантировать архивизацию данных при сбое жесткого диска.

Второй подход заключается в расширении Windows NT для поддержки реального времени. Это может быть оригинальная разработка изготовителя системы управления или использование готового коммерческого решения, например RTX 4.1 фирмы VenturCom, на чем и базировалась работа.

Подход на базе расширения реального времени для Windows NT более перспективен. Во-первых, в расширении использованы те же типы объектов для управления задачами, как и у ядра Windows NT (мютексы, семафоры и т.д.). Во-вторых, нет необходимости во второй операционной системе, что сокращает расходы и снимает проблемы установки и стыковки обеих операционных систем на одном персональном компьютере.

Один из самых надежных и распространенных алгоритмов диспетчеризации в многозадачных ОС - это алгоритм циклической диспетчеризации, когда для выполнения конкретной задачи предоставляется некоторый квант времени. По истечении каждого кванта времени планировщик просматривает очередь активных задач и принимает решение - которой из них передать управление. Прежде, чем сформировать этот алгоритм, рассмотрим особенности режима времени, в котором работает система управления с операционной системой Windows NT и расширением реального времени.

Функциональные задачи систем управления соответственно были разделены на три группы. В режим *жесткого реального времени* попадают критические задачи (интерполяция кадров управляющей программы, ввод-вывод). В режиме *мягкого реального времени* решаются задачи, непосредственно связанные с задачами реального времени (например, интерпретация кадра управляющей программы); в отличие от жесткого времени здесь допустимы задержки потока из-за свопинга (подкачки) памяти, обращения к жесткому диску, прерывания и т.д. В режиме *машинного времени* работают остальные стандартные прикладные модули системы управления.

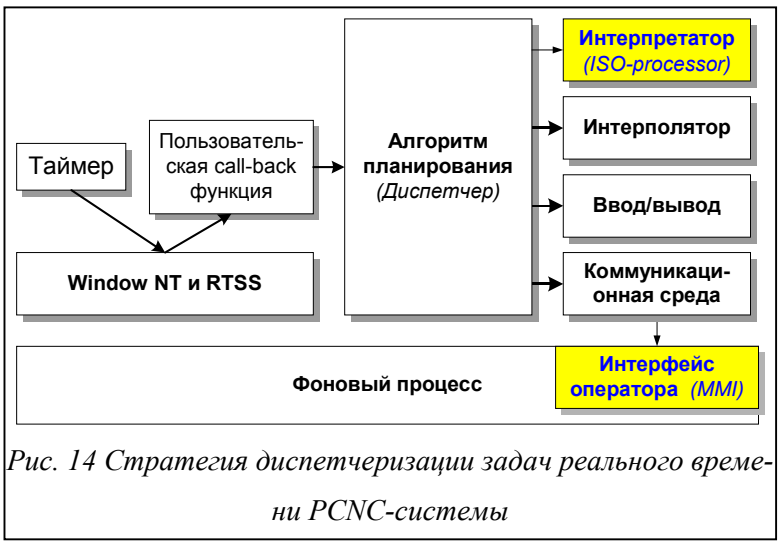

Рис. 14 Стратегия диспетчеризации задач реального времени PCNC-системы

Стратегия диспетчеризации задач в PCNC системе (см. Рис. 14) заключается в использовании таймера реального времени. По истечении кванта времени стандартный механизм генерирует прерывание, которое обрабатывается так называемой прикладной call-back функцией. Функция реализует алгоритм планирования (диспетчеризации) задач интерпретации, интерполяции, ввода/вывода, коммуникации и интерфейса оператора.

В соответствии с обозначенными для системы управления режимами времени в жестком реальном времени работают задачи диспетчеризации, интерполяции, ввода/вывода, коммуникации. В мягком реальном времени работают задачи интерпретации и обновления экранов интерфейса с оператором, а в фоновом процессе - задачи интерфейса с оператором.

Схема “один процесс - много потоков” имеет такие важные достоинства, как быстродействие и высокая реактивность. Высокая реактивность потоков объясняется меньшим временем переключения их контекстов по сравнению с процессами. Потоки используют общее адресное пространство процесса, а процессы нуждаются в разделяемой памяти.

Согласно принятым представлениям о режимах времени в системе управления, выделим три группы потоков: *потоки жесткого реального времени*; они работают в процессе реального времени; это так называемые RTSS-процессы; *потоки мягкого реального времени*; они функционируют в Win32 и RTAPI-процессах; *потоки машинного времени*; они работают в стандартных Win32-процессах.

Обмен данными и синхронизация процессов машинного времени и процесса мягкого реального времени традиционны, т.к. это осуществляется на базе общей платформы Win32.

Обмен данными между процессами мягкого реального времени и процессами жесткого реального времени осуществляется на базе разделяемой памяти (shared memory) – механизма, предоставляемого со стороны RTX.

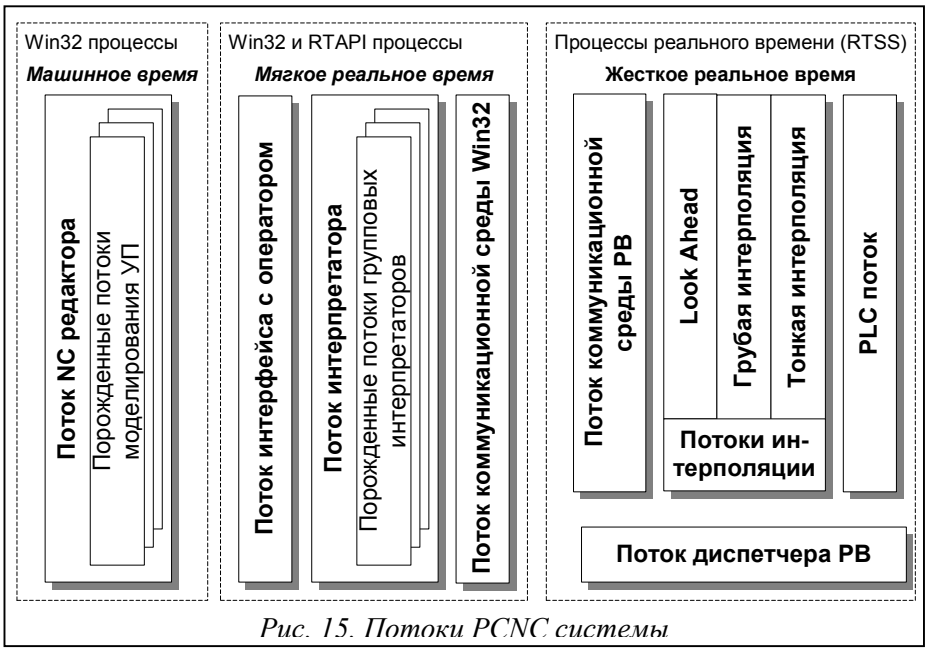

Рис. 15. Потоки PCNC системы

RTSS-процесс на Рис. 15 включает в себя набор потоков, решающих критические задачи в PCNC-системе. Поток диспетчера, по сути, является call-back функцией таймера, где реализован планировщик процессов. Планировщик с помощью мютексов или семафоров запускает или останавливает тот или иной поток.

Коммуникационную среду разделим на два потока, один из которых функционирует в жестком реальном времени (поток коммуникационной среды реального времени); а другой работает в мягком реальном времени (поток коммуникационной среды Win32). Задача состоит в передаче данных как между потоками внутри процесса, так и между процессами.

Задача интерполяции реализуется: потоком Look-Ahead, выполняющим опережающий просмотр и коррекцию кадров управляющей программы; потоком грубой интерполяции, вызываемым обычно с частотой 50Гц; потоком тонкой интерполяции, вызываемым обычно с частотой 1-2-мсек, осуществляющим сплайновую интерполяция между точками, рассчитанными в рамках грубой интерполяцией.

Частота вызова интерполяторов параметрически настраивается в планировщике (в потоке диспетчера реального времени). Поток программируемого контроллера решает задачу управления электроавтоматикой и задачу ввода-вывода.

В процессе мягкого реального времени, помимо потока коммуникационной среды Win32 и потока интерпретации кадров управляющей программы, работает поток интерфейса с оператором. Поток интерпретатора запускает групповые интерпретаторы

как порожденные потоки. В потоке интерфейса с оператором отображаются такие данные процесса реального времени, как текущие координаты, скорость подачи, состояние процесса, режимы системы управления. Отсюда же отправляются управляющие команды процессу реального времени: запуск строки ручного ввода, выбор управляющей программы, “стоп” и т.д.

Процессы машинного времени - это классические Windows-процессы. Примером может служить редактор управляющих программ, который запускает в качестве порожденных потоки моделирования управляющих программ.

Предложенное разбиение потоков в PCNC-системе позволяет выделить критические ко времени потоки, а их запуск в RTSS-процессе гарантирует нам жесткое реальное время в Windows NT. При этом оптимально используются вычислительные ресурсы процессора.

Часто при работе с определенными модулями системы ЧПУ, их отладке, написании управляющих программ, программ PLC и т.д. обходятся более дешевым вариантом и используют эмулятор операционной системы реального времени. Диспетчер реального времени и прикладного программного обеспечения функционирует в такой же логической последовательности, как и в реальном времени. Различие состоит в том, что управление процессами происходит в машинном масштабе времени.

Задача эмуляции в Windows NT актуальна только для решений на базе супервизора. Нужно разработать специальное программное обеспечение, которое решает проблемы: адресного пространства; приоритетов; соответствия объектов синхронизации; соответствия базовых функций.

Эмулятор транслирует типы, объекты синхронизации и функции ОС РВ соответственно в типы, объекты синхронизации и функции ОС Windows NT. При этом диспетчер реального времени и прикладные задачи имеют тот же самый исходный код для РВ, но они перекомпилированы и выполняются как Windows задачи.

Решение перечисленных проблем создания эмулятора супервизора определяет основные шаги методики.

1. Определение архитектурной реализации эмулятора, диспетчера реального времени и прикладных задачи в понятиях Windows приложения.
2. Создание таблицы соответствия приоритетов.
3. Реализация объектов синхронизации задач супервизора на базе объектов ядра Windows NT.
4. Реализация функции супервизора на базе API функции Windows NT.

Приведенная методика определяет основные шаги построения эмулятора суперви-

зора. Конечно, при этом прямое обращение из ОС Windows по адресу аппаратуры не обрабатываются как эмуляторов RTX, так и эмуляторов супервизора.

4.2 Построение коммуникационной среды по типу общей объектно-ориентированной магистрали

Традиционно коммуникационную среду системы ЧПУ трактуют как некоторый набор интерфейсных API-функций для обмена данными с ядром системы ЧПУ. При таком подходе, однако, любое изменение в архитектуре системы проблематично. API-функции задают некоторый общий интерфейс подключения модулей в системе ЧПУ, но не поддерживают их интеграцию.

Введем понятие объектно-ориентированной магистрали как средства межмодульной коммуникации и источника необходимых межмодульных услуг. Магистраль является программным (виртуальным) каналом для обмена данными между подключенными к каналу модулями. Объектно-ориентированная магистраль предполагает наличие в ее программном обеспечении набора объектов, решающих задачи подключения модулей, транспортировки данных и др. Выделим четыре типа функций объектно-ориентированной магистрали.

Функции запроса данных предполагают, что в PCNC-системе существуют данные разных типов и потребность в них различна. Например, данные о количестве и именах используемых на станке координатных осей требуются один раз в момент инициализации PCNC-системы. Данные о текущем состоянии выполняемого процесса нужны постоянно, чтобы принимать корректные решения. Существуют и другие варианты запросов на получение данных. Поэтому объектно-ориентированная магистраль предусматривает пять их видов: синхронный, асинхронный, синхронный по событию, асинхронный по событию, асинхронный циклический запрос.

Функции управления можно разбить на три группы: функции управления каналом, открывающие и закрывающие канал; функции процессов, контролирующие ход их выполнения, включая запуск и останов; функции управления состояниями (они будут рассмотрены ниже).

Если запрос связан с процессом получения данных из модуля-источника (сервера) через внутреннюю структуру коммуникационной среды, то процесс переноса и обработки этих данных в модуль-клиент относится к группе *функций отображения данных.*

В процессе обмена данными через магистраль выделены две фазы. В фазе запроса данных определяется сервер данных, устанавливается тип запроса и приемник данных (клиент). В фазе отображения определяется тип и формат отображения. Формат

отображения предполагает, что одни и те же данные могут быть отображены, например, в виде пиктограмм, в текстовом или числовом виде.

Объектно-ориентированная магистраль предусматривает три типа отображений, каждый из которых поддерживается собственным механизмом.

В зависимости от способа вывода данных, отображения разделены на несколько групп: визуализация в галерее управляющих элементов, которая строится на базе стандартных Windows-элементов; визуализация в галерее ActiveX-элементов, которая строится на базе OLE-элементов Windows; визуализация в среде «документа-представления» при создании пользовательских приложений на базе стандартного механизма MFC; визуализация с целью управления ходом процесса в PCNC-системе; статистическое накопление данных для сохранения, например, для их последующего анализа.

Четвертая группа функций объектно-ориентированной магистрали - это *вспомогательные функции*. Они составляют единый механизм конвертирования и форматирования данных, обработки ошибок, формирования исключений (exceptions) для всех модулей, подключенных к объектно-ориентированной магистрали.

Взаимодействие модулей PCNC-системы носит клиент-серверный характер. Транзакции (сессии) между модулем-клиентом, запрашивающим услугу, и модулем-сервером, оказывающим услугу, обобщены по их назначению. *Команда* направляется серверу для выполнения некоторого действия. Команда не предполагает ответа со стороны сервера. Другой вариант: *запрос*, направляется серверу с целью получения некоторых данных, например, значений текущих координат. Такой запрос предполагает ответ со стороны сервера.

В рамках *синхронной сессии* клиент направляет запрос серверу и приостанавливает работу в точке запроса. При готовности сервер отвечает, после чего клиент продолжает работу.

В рамках *асинхронной сессии* клиент направляет запрос серверу и продолжает свою работу. Ответ сервера обрабатывается специальной call-back-функцией (аналогичной функции обработки прерывания) клиента.

Событием в системе PCNC служит всякое изменение данных, например, изменение состояния процесса.

В рамках *асинхронной сессии по событию* клиент направляет запрос серверу и продолжает свою работу. Сервер отвечает только после того, как произойдет событие, т.е. изменятся запрашиваемые данные. Ответ обрабатывается call-back-функцией клиента.

Синхронную сессию по событию используют только для отладки PCNC-системы.

На основе базовых транзакций реализируется циклический опрос данных; например, постоянный опрос текущих значений координат для вывода этих значений на экран.

Организация транзакций в PCNC-системе на основе предложенных сессий позволяет свести трафик коммуникационной среды до минимума и экономично использовать ресурсы для выполнения других задач.

Другие схемы, как, например, “многие ко многим” представляют собой комбинацию указанных ранее.

Для реализации указанных транзакций и схемы отображения данных разработана объектно-ориентированная модель коммуникационной магистрали.

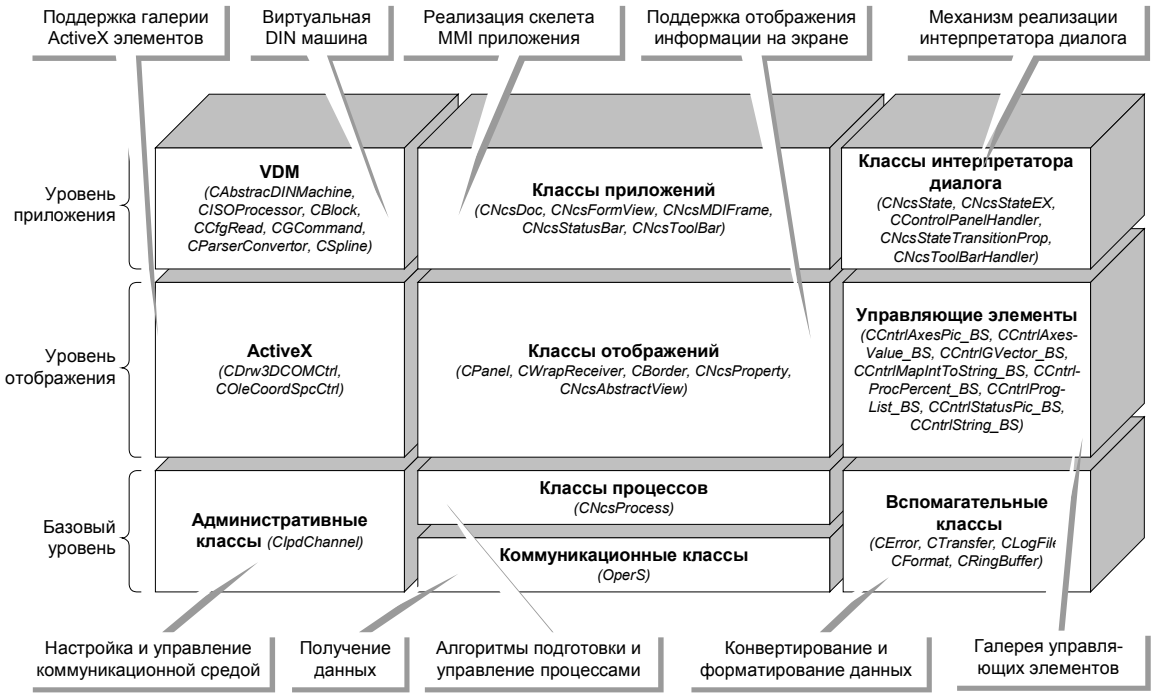

Рис. 16. Виртуальная структура коммуникационной среды в виде общей объектно-ориентированной магистрали (ООС)

Согласно выделенным функциям ООС виртуальная структура показана на Рис. 16 в виде трехуровневой модели классов. Базовый уровень реализует запрос данных. Второй уровень - уровень отображения, предназначен, в основном, для вывода информации на экран. Ведущую роль здесь играют классы отображения, устанавливающие связь между управляющими элементами экрана и коммуникационными классами. Классы управляющих элементов экрана и классы ActiveX-управляющих-элементов служат для приема конкретной информации в определенной форме; а галерея этих классов определяет дизайн экрана и меру богатства его изобразительных средств.

Прикладной уровень поставляет приложениям классы, работающие с коммуникаци-

онной средой, как «документ-представление».

Обозначенная подобным образом структура объектно-ориентированной магистрали с иерархическим построением уровней и блоками с четко установленной функциональностью упрощает переход к практическому построению коммуникационной среды.

Организация коммуникационной среды в виде открытой системы и модульной системы, предполагает выделение модулей и их реализацию в виде библиотек DLL.

5. Глава 5. Практические аспекты реализации модулей открытой системы ЧПУ

Накопленный практический опыт (“know-how”) является одним из основных критериев успехов или неудач на рынке производителей систем ЧПУ. В связи с этим рассмотрена технология применения изложенных ранее теоретических и методологических аспектов построения PCNC к конкретным задачам в системе управления.

5.1 Реализация геометрической задачи

Геометрическая задача состоит из трех крупных модулей: интерпретатора управляющих программ, интерполятора, модуля управления следящими приводами. Последний модуль сильно зависим от типа следящих приводов и способа замыкания позиционных контуров, поэтому его рассмотрение опущено.

Наилучший вариант реализации интерпретатора состоит в его построении по типу ISO-процессора, поскольку такое решение обеспечивает наибольшее быстродействие и гибкость PCNC системы в отношении системы команд, т.е. версии языка ISO-7bit. Выделение “независимого” и “зависимого” уровней в объектной реализации ISO-процессора позволило выделить программные компоненты, одинаковые для всех версий языка ISO-7bit.

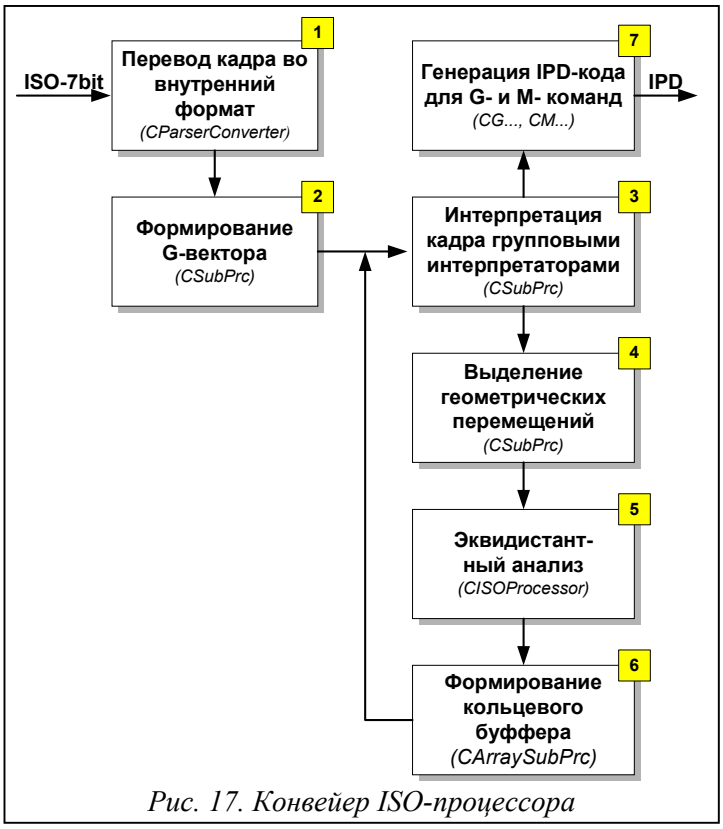

Рис. 17. Конвейер ISO-процессора

Интерпретация кадров управляющей программы построена на основе конвейера и выполняется за семь шагов (см. Рис. 17). На завершающей стадии данные поступают в кольцевой буфер (позволяющий анализировать

на совместимость группу соседних кадров с эквидистантной коррекцией), а окончательный результат интерпретации представлен в виде IPD-кода (InterPolator Data).

В работе приведены диаграмма классов и диаграмма взаимодействия конвейера.

Современные требования определяют новую (открытую) архитектуру интерполятора, в которой четко обозначены отдельные блоки. Открытый интерполятор допускает свободное наращивание алгоритмов интерполяции и произвольную их комбинацию при воспроизведении сложных траекторий в многокоординатном пространстве (в том числе и с использованием сплайнов). В работе приведен фрагмент формального грамматического описания одной из версий входного формата, в составе которого управляющие структуры (заголовки) и данные.

Модуль интерполяции подключен к общей объектно-ориентированной магистрали системы ЧПУ. Интерфейс интерполятора обеспечивает: прием параметров интерполяции и оперативных сигналов управления интерполятором, выдачу данных о состоянии модуля интерполяции.

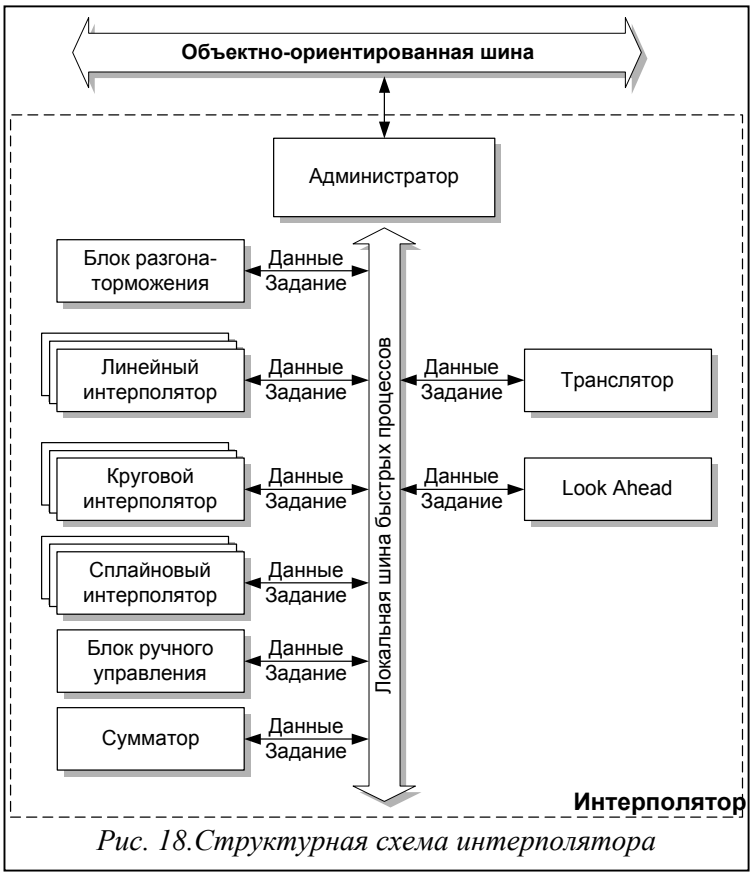

Рис. 18.Структурная схема интерполятора

Схема интерполятора показана на Рис. 18 в виде некоторого набора блоков, собственной внутренней шины и администратора. Кадры управляющей программы поступают на вход транслятора в IPD-формате, преобразуются во внутренний формат интерполятора, обрабатываются в блоке опережающего просмотра кадров Look Ahead (с целью сглаживания скорости подачи). Транслятор формирует сообщения, в которых упакованы параметры интерполяции. Сообщения адресуются к определенным блокам интерполятора и могут быть главными и дополнительными.

Внутренняя шина интерполятора является “шиной быстрых процессов” и связывает между собой все блоки. Она реализована на базе объектно-ориентированного подхода и соединена с основной объектно-ориентированной шиной системы ЧПУ с помощью администратора. Блоки, участвующие в отработке текущего кадра, назначаются с помощью специального кода. Этот код инициализируется в трансляторе и пе-

редается в администратор. Схема кодирования обеспечивает гибкость и открытость интерполятора. Возникает возможность построения администратора, который инвариантен к составу и количеству блоков интерполятора.

Общая схема работы интерполятора выглядит следующим образом. После предоставления кванта процессорного времени администратор (построенный по схеме микропрограммного автомата) посылает запрос транслятору на получение кодов блоков интерполятора, которые должны быть запущены.

5.2 Реализация логической задачи управления

Применение визуальных средств программирования, дают оператору инструмент для графической диалоговой разработки программы управления электроавтоматикой; но и порождают исполняемые C++ коды без компиляции в промежуточный язык. При этом изменяется (в сторону существенно большей эффективности) сама структура математического обеспечения системы PCC (Personal Computer-Controller, персональный программируемый контроллер).

Приведен формализм описания циклов электроавтоматики, использующий для описания электроавтоматики формализм иерархических графов.

Методика описания цикла электроавтоматики включает такие этапы: этап разработки “первичного автомата”, т.е. автомата верхнего уровня иерархии, являющегося по сути диспетчером режимов; этап разработки режима нерегулярных ситуаций (внутреннего режима), который сохраняет корректность состояния управляемого объекта при любых переключениях основных режимов, а также гарантирует неизменное состояние объекта, если цикл пассивен; этап выделения параллельно работающих автоматов, действующих в рамках цикла; этап разработки автоматов нижнего уровня иерархии. Методика описания цикла электроавтоматики применена на примере схемы управления револьверной головкой токарного станка.

Однотипность скелета исполняемого кода циклов позволила разработать инструментальную систему визуального проектирования, генерирующую исполняемые C++ исходные файлы. Конкретный граф вводят с панели интерфейса программиста, которая предлагает набор графических примитивов: простую вершину-состояние, сложную вершину-состояние, дугу, узел дуги. Свойства примитивов (имена, типы вершин-состояний и др.) задают в диалоговом режиме на “странице свойств” (property page). Функции визуального проектирования обеспечивают: многоуровневое вложение графов с работой на каждом уровне в отдельном окне; выполнение групповых операций (выделение фрагмента графа; удаление, копирование, перемещение фрагментов в разных позициях и на разных уровнях); сохранение-загрузку про-

екта или фрагмента; импорт одного проекта в другой; документирование проекта и генерацию отчетов; генерацию исходного кода для последующей компиляции; верификацию графа на уровне проектирования, моделирование и отладку циклов. Применение инструмента визуального проектирования многократно повышает производительность разработчика, позволяет создавать сложные циклы электроавтоматики, реализация которых без инструментальной поддержки проблематична.

Жизненный цикл логической задачи управления предполагает программирование, интерпретацию программы и ее исполнение. Современная тенденция состоит в упрощении первой фазы за счет визуального программирования, включая инструментальную поддержку; в объектно-ориентированной реализации второй фазы.

5.3 Реализация терминальной задачи

“Наполнение” терминальной задачи определяет привлекательность и конкурентоспособность системы ЧПУ на рынке. Свойства открытой системы ЧПУ развиты настолько, насколько терминальная задача поддается конфигурации и расширению. Наиболее важными разделами терминальной задачи служат: интерпретатор диалога оператора в Windows-интерфейсе, редактор управляющих программ в коде ISO-7bit, редактор-отладчик управляющих программ на языке высокого уровня.

Проектирование MMI-приложения включает: создание скелета приложения; реализацию экранов; разработку интерпретатора диалога; организацию информационных сессий с другими модулями системы управления.

В числе функций диалога можно обозначить: получение текущей информации о процессе управления; тестирование системы и объекта; редактирование и моделирование управляющей программы; ручной ввод и управление отработкой данных; ввод программы и автоматическое управление; управление наладочными операциями. Диалог устанавливает допустимые переходы между состояниями MMI-приложения, в рамках которых и воспроизводятся необходимые функции.

В качестве формальной модели описания диалога предложен иерархический граф состояний; вершины которого отражают устойчивые состояния MMI после нажатия оператором той или иной клавиши панели оператора, а дуги нагружены именами функциональной клавиатуры или других клавишей.

Иерархические графы удобны для описания многорежимных многоуровневых диалогов и позволяют проектировать диалог "шаг за шагом", от укрупненного выбора режимов к детальному определению поддерживаемых функций. При этом, возможно описывать процесс управления не только с использованием функциональной клавиатуры, но и с помощью меню, дерева навигации и т.д.

В рамках инструментальной системы визуального проектирования 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 концентрировать внимание на текущих фрагментах диалога.

Практика показывает, что сгенерированных CAM станции УП начинают работать на станке, как правило, после третьей редакции. При этом исправления, которые нужно внести в управляющую программу (УП), как правило, незначительные и их удобно выполнить, прежде всего, на самом станке.

К редактору управляющих программ предъявляют, во-первых, стандартные требования, характерные для текстового редактора. Во-вторых, существуют специфические требования: перенумерация после изъятия-включения кадров; изменение масштаба и размерности; вывод активных G-функций (G-вектора) на основе предыстории кадра; синтаксический и семантический контроль; диалоговый (графический) ввод кадра и параметров стандартных циклов (файлы графической помощи находятся в составе конфигурационного файла); создание управляющих программ в режиме обучения.

Средства отладки программ включают: пространственное графическое моделирование траектории инструмента с различением (по цвету, типу и толщине линий) быстрых и рабочих перемещений; активное использование точек останова, используемых, в том числе, для выделения фрагментов графического изображения; масштабирование графического изображения; поддержку различных режимов изображения; моделирование оставшейся части программы по отношению к текущей позиции станка. Подобные возможности требуют включения в состав редактора некоторого ядра и дополнительных подсистем: интерпретатора управляющих программ (для любых версий кода ISO-7bit), имитатора интерполятора для рисования траекторий и сервера обучения.

Конфигуратор формализует код ISO7-bit путем выделения в нем нескольких уровней абстракции. На первом уровне определяется система команд (G-функций) и параметры каждой команды. Следующий уровень разбивает систему команд на группы по функциональному назначению G-функций и формирует G-вектор активных команд. Последний уровень абстракции назначает списки разделителей, комментариев, имен осей и адресов, имен G-функций. Подобным способом удается формализовать любую версию кода ISO 7-bit и соответствующим способом сконфигурировать редактор.

Редактор управляющих программ имеет архитектуру, открытую для конечных пользователей, открытую для разработчиков самого редактора, открытую для внешних приложений. Для конечных пользователей это, прежде всего, означает возможность конфигурации на различные версии языка ISO-7bit с помощью конфигурационного файла, имеющего текстовый формат. Далее, существует возможность конфигурировать интерфейс пользователя, включая систему контекстных подсказок и систему помощи, используя текстовый файл инициализации и соответствующие динамические библиотеки ресурсов.

Разработчикам редактора предлагается архитектура, открытая для интеграции и компоновки. Так, редактор может быть интегрирован в существующий интерфейс системы ЧПУ или работать в качестве независимого приложения в технологическом отделе подготовки управляющих программ.

Архитектура, открытая для внешних приложений поддерживается интерфейсом OLE IDataObject, с помощью которого осуществляется передача данных через “clipboard”, - стандартный Windows-механизм для обмена данными между приложениями.

Независимо от версии, структуры всех языков управляющих программ высокого уровня однотипны: имеется основная программа и некоторый набор вызываемых подпрограмм. В теле программы представлен список переменных, которые по ходу выполнения программы меняют значения. Процесс выполнения сопровождается информационными сообщениями, предупреждениями, сообщениями об ошибках.

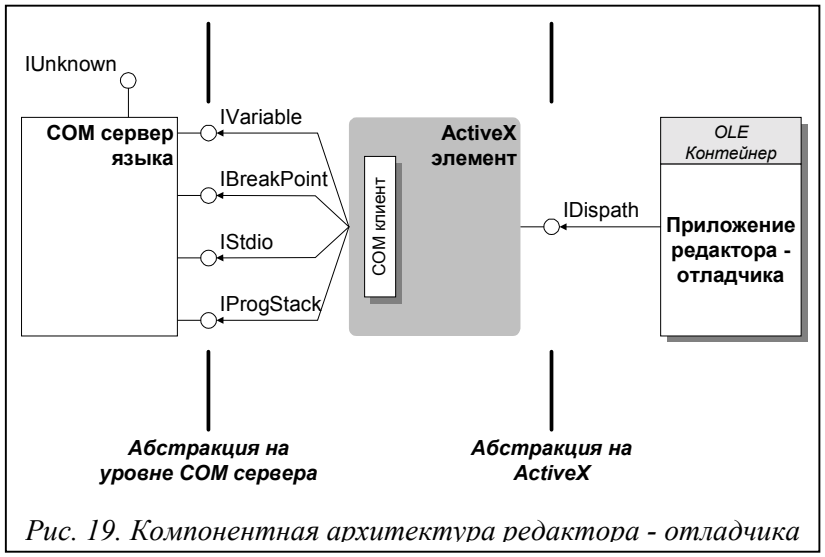

Рис. 19. Компонентная архитектура редактора - отладчика

Базовые окна постоянно присутствуют на экране, окна со вспомогательной информацией (например, списком точек останова) реализованы как всплывающие. Основные и вспомогательные окна редактора-отладчика образуют ActiveX управляющий элемент.

Архитектура редактора-отладчика включает COM сервер, ActiveX управляющий элемент, приложение (см. Рис. 19). В этой архитектуре выделены два абстрактных уровня. На первом уровне поддерживается работа с различными языками управляющих программ; причем для каждого языка необходимо разработать его собственный COM-сервер. Любой COM–сервер, однако, должен располагать неизменным

набором интерфейсов для работы с переменными, точками останова, буферизованными файлами и сообщениями. В этом случае, ActiveX управляющий элемент со всеми своими основными и вспомогательными окнами способен работать с любой версией языка высокого уровня управляющих программ.

Второй уровень абстракции развязывает ActiveX управляющий элемент и механизм управления им. Это позволяет использовать ActiveX или в составе терминальной задачи системы ЧПУ, или в отдельном приложении на персональном компьютере.

Терминальная задача относится к числу наиболее сложных и наиболее ответственных разделов системы ЧПУ при управлении мехатронными системами. Ее «скелетом» служит интерпретатор диалога оператора в Windows-интерфейсе, для разработки которого использована формальная методика, поддержанная оригинальной инструментальной системой. Для редактирования, отладки и моделирования управляющих программ предложены два конфигурируемых приложения, для управляющих программ низкого и высокого уровня соответственно.

5.4 Реализация диагностической задачи управления

Наиболее совершенные системы ЧПУ располагают отдельным режимом диагностики, который реализован в виде программно-аппаратного комплекса и ориентирован на тестирование и глубокое исследование логической и геометрической задач управления. Диагностика, как правило, выполняется “вне реального времени”; что означает: измерения сохраняются в памяти, а затем анализируются. Подсистема диагностики способна конфигурировать измерения, считывать измеряемые сигналы, запоминать результаты измерений вместе с результатами конфигурации измерений, распечатывать осциллограммы измерений, считывать файлы с результатами измерений и результатами конфигурации измерений, выполнять разнообразные операции над измеренными сигналами. Для диагностики логической задачи управления служит Логический анализатор, а для диагностики геометрической задачи предназначен Осциллограф.

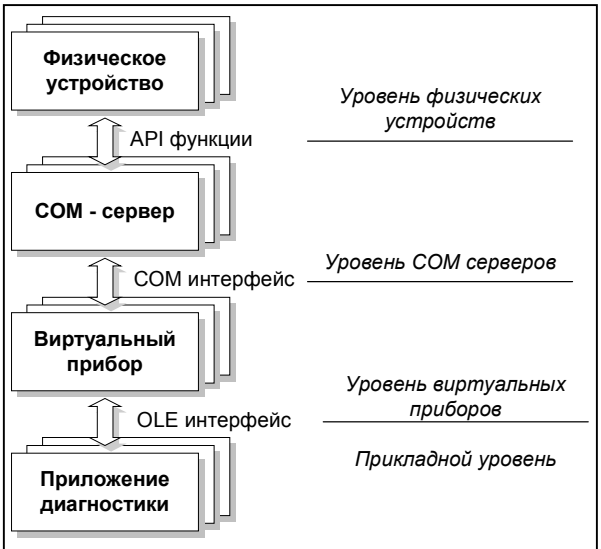

Рис. 20. Распределенная архитектура подсистемы диагностики

Под *виртуальным прибором* диагностики будем понимать ActiveX-элемент, предоставляющий результаты диагностических испытаний и создающий внешний образ измерительного устройства; например, Ло-

гического анализатора или Осциллографа.

В обобщенном виде распределенная архитектура подсистемы диагностики представлена на Рис. 20. Соединение с физическим устройством (контроллером ввода-вывода, контроллером приводов и т.д.) осуществляется с помощью интерфейсной функции этого устройства. COM-серверы маскируют особенности физических устройств, но организуют доступ к устройствам по общему COM-интерфейсу. Уровень виртуальных приборов предлагает средства интерактивного конфигурирования и визуализации измерений. На прикладном уровне эти приборы встроены в приложение с доступом к приборам через OLE-интерфейс.

В работе рассмотрены пример реализации диагностики программируемого контроллера (логический анализатор) и пример реализации диагностики следящего привода (осциллограф)

Электроавтоматика мехатронных систем достаточно сложна и требует высококвалифицированных специалистов при наладке и запуске оборудования в эксплуатацию. Подобные специалисты находятся обычно в удаленных сервисных бюро и занимаются дистанционным анализом входных и выходных сигналов программируемого контроллера с помощью все того же виртуального прибора, располагая конфигурацией и результатами измерений.

Оптимальная настройка регуляторов следящих приводов подачи невозможна без тщательного анализа их динамических характеристик с помощью Осциллографа подсистемы диагностики. Особенность распределенной архитектуры Осциллографа состоит в использовании “процесс-COM-сервера”, в котором собраны все операции над сигналами, независимо от устройства-источника этих сигналов. В числе возможных операций над сигналами: масштабирование, сдвиг, практически любые математические вычисления. Помимо стандартных процедур конфигурации и отображения измерения, Осциллограф позволяет строить с помощью процесс-сервера амплитудно-частотные и фазо-частотные характеристики следящих приводов.

Современные системы управления располагают свободными ресурсами вычислительной мощности, которые должны быть использованы наиболее эффективно. В этом смысле наибольший интерес представляет создание и развитие подсистемы диагностики. В первую очередь следует диагностировать логическую и геометрическую задачи управления. Концепция виртуальных приборов, построенных по типу ActiveX-элементов, позволяет использовать разработанные средства диагностики в самых различных приложениях, которые представляют наибольший интерес для конечных пользователей. Особенности COM-подхода и COM-технологии таковы, что

разработанные диагностические системы могут быть использованы в любых устройствах ЧПУ.

Заключение

В результате выполнения работы созданы научные и практические основы проектирования систем ЧПУ нового поколения, обладающих гибкостью на всех этапах жизненного цикла, обеспечивающих конкурентоспособность технологического оборудования и повышение эффективности производства. Эти результаты можно отнести к числу тех, которые поддерживают стратегическую безопасность страны, поскольку речь идет о восстановлении целостности ее технологической среды.

Эффективность автоматизации производства возрастает за счет повышения точности обработки при использовании многоканального управления и уменьшения при этом числа смен баз. Надежность систем ЧПУ увеличивается, благодаря модульной архитектуре и новой структуре программного обеспечения; благодаря удаленной диагностике приводов и электроавтоматики технологических машин. Возможности разработанной открытой архитектуры систем ЧПУ таковы, что позволяют своевременно реагировать на потребности рынка, повышая тем самым конкурентоспособность систем. Формализация интерфейса оператора и возможность достижения максимальной степени его эргономичности сокращают время обучения операторов. Использование гибких и мощных редакторов - отладчиков в составе систем ЧПУ снижает затраты на подготовку производства за счет снижения себестоимости управляющих программ ЧПУ.

Основные выводы

1. Решена актуальная научная проблема, имеющая важное народнохозяйственное значение, которая состоит в создании теоретического и практического базиса нового поколения систем ЧПУ широкого назначения; систем, обладающих модульным ядром, открытой архитектурой, гибкостью на всех этапах жизненного цикла, высокой конкурентоспособностью и низкими эксплуатационными затратами.
2. Предложенная концепция построения систем ЧПУ типа PCNC с открытой модульной архитектурой раскрывает дополнительно ресурсы для повышения мобильности, коммуникабельности, масштабируемости, экономичности и ремонтопригодности таких систем. Концепция впервые предлагает реализацию открытой архитектуры вплоть до уровня конечных пользователей.

3. Процесс проектирования модульной системы ЧПУ типа PCNC сведен к последовательной трансляции моделей разного уровня, когда на каждом уровне конкретизированы строго определенные аспекты системы. Выделение уровней абстракции в структуре системы ЧПУ типа PCNC создает независимость программного обеспечения от объекта управления и от конкретных прикладных задач. Общая магистраль в архитектуре системы ЧПУ типа PCNC обеспечивает стандартный доступ к ее внутренним модулям. Распределенная архитектура системы ЧПУ типа PCNC обеспечена технологией компонентного подхода.

4. Для комплексного решения ключевых проблем создания открытых систем ЧПУ (многозадачность, ведение проекта, формализация разработки) сформировано окружение процесса проектирования, использующее возможности языковых процессоров, системные возможности операционной системы Windows NT, возможности стандартных и оригинальных инструментальных средств. Проблема совместимости систем ЧПУ на уровне входного языка решается с помощью объектно-ориентированного подхода при построении систем ЧПУ типа PCNC по типу открытых языковых процессоров.

5. Выделение задач управления, работающих в “жестком” и “мягком” реальном времени и в фоновом процессе, позволяет оптимизировать работу программного обеспечения. Трафик в коммуникационной среде минимизирован посредством единообразных транзакций на базе синхронных, асинхронных сессий и сессий по событию.

6. Концепция геометрического канала ЧПУ включает выделение программных компонент, одинаковых для всех систем ЧПУ; свободное наращивание алгоритмов интерполяции и произвольной их комбинации в многокоординатном пространстве за счет предложенного набора входных форматов; исполнение и синхронизация управляющих программ в разных геометрических каналах.

7. Возможно увеличить эффективность разработки циклов электроавтоматики за счет их формализации и описания в виде иерархических графов, за счет применения средств визуального программирования и генерации инструментальной системой C++ кодов исполняемых файлов циклов электроавтоматики.

8. Формализация пользовательского интерфейса и использование оригинальных инструментальных средств разработки и отладки диалога с оператором позволяют преодолеть трудоемкость построения интерфейса оператора и его

адаптации к конкретному технологическому процессу пользователя.

9. Спецификация основных интерфейсов подсистемы диагностики, применение COM серверов диагностики, реализация виртуальных приборов в виде ActiveX элементов управления, - все это позволило решить задачу удаленной диагностики систем PCNC и компоновки программного обеспечения диагностики для разных устройств ЧПУ.

Работы, опубликованные по данной теме:

1. Мартинов Г.М. Виртуальные приборы диагностики в системе ЧПУ // Информатика-машиностроение. 1998, №4. С. 8-12.

2. Мартинов Г.М. Выделение геометрического канала как компоненты открытой PCNC-системы// Доклады международной конференции «Информатизационные средства и технологии» Том 3 - МФИ-99, 19-21 октября 1999, Москва - М.: МГТУ СТАНКИН. 1999. С.164-167.

3. Мартинов Г.М. Клиент-серверные отношения в системе ЧПУ типа PCNC // Тезисы Всероссийского электротехнического конгресса с международным участием. “На рубеже веков: итоги и перспективы” том 2 - ВЭЛК-99, Москва, 1999., С.73-74.

4. Мартинов Г.М. Компонентная модель системы ЧПУ типа PCNC// Доклады международной конференции «Информатизационные средства и технологии» том 2 - МФИ-98, 20-22 октября 1998, Москва - М.: МГТУ СТАНКИН. 1998. С.202-207.

5. Мартинов Г.М. Архитектура персональной системы ЧПУ (PCNC) широкого профиля // Доклады международной конференции «Информатизационные средства и технологии» том 2 - МФИ-97, 21-23 октября 1997, Москва - М.: МГТУ СТАНКИН. 1997. С.137-141.

6. Мартинов Г.М., Зоненштейн И.И. Современная трактовка языка управляющих программ ISO-7bit при построении виртуальной ISO машины // Доклады международной конференции «Информатизационные средства и технологии» том 2 - МФИ-97, 21-23 октября 1997, Москва. - М.: МГТУ СТАНКИН. 1997. С.142-147

7. Мартинов Г.М. Открытая система ЧПУ на базе общей магистрали // Автомобильная промышленность, 1997. №4, с.31-34

8. Мартинов Г. М. Реализация ISO-процессора в системы CNC Andronic фирмы ANDRON (ФРГ) //Сб. тезисов Международной научно-технической конференции ”Электротехнические системы транспортных средств и робототехнических производств”. - Суздаль 1995. С. 21-22.

9. Мартинов Г.М., Рыбкина Л.И, Старостин А.К. Использование объектно-ориентированного программирования для создания САПР ТП механообработки - «Технология» : Межотр. научно-технич сб. Сер. Гибкие производственные системы и робототехника /

ВИМИ, 1991 вып. 6, С.9-16

10. Мартинов Г.М., Сосонкин В.Л. Концепция числового программного управления мехатронными системами: проблема реального времени // Мехатроника, 2000, №3. С. 37-40.

11. Мартинов Г.М., Сосонкин В.Л. Концепция числового программного управления мехатронными системами: реализация геометрической задачи // Мехатроника, 2001, №1. С. 9-15.

12. Мартинова Л.И., Мартинов Г.М., Персональная система ЧПУ (PCNC) широкого профиля // Электрооборудование автомобилей и тракторов. Сборник научных трудов 74. М.: НИИАЭ. 1998. С.82-88.

13. Мартинов Г.М., Сосонкин В.Л. Концепция числового программного управления мехатронными системами: технология объектно-ориентированного программирования // Мехатроника, 2000, №3. С. 37-40.

14. Мартинов Г.М., Сосонкин В.Л. Концепция числового программного управления мехатронными системами: структура руководства по программированию// Мехатроника, 2000, №3. С. 37-40.

15. Соломенцев Ю.М., Сосонкин В. Л, Мартинов Г.М. Построение персональных систем ЧПУ (PCNC) по типу открытых систем управления // Информационные технологии и вычислительные системы. 1997.№3. С.68-76.

16. Сосонкин В.Л., Мартинов Г.М., Зоненштейн И.И. Новый подход к построению редакторов управляющих программ: Универсальная среда AdvancEd // Информационные технологии в проектировании и производстве. - М.: ВИМИ. - 1999. - №1. - с. 80-87.

17. Сосонкин В.Л., Мартинов Г.М., Зоненштейн И.И. Универсальная среда «AdvancEd» для редактирования, отладки и моделирования программ ЧПУ в коде ISO-7bit (любой версии) // Информатика технологии, 1998. №3 (21). С. 3-5.

18. Сосонкин В.Л., Мартинов Г.М. Концепция геометрического ISO-процессора для систем ЧПУ // СТИН, 1994, №7, с.17-20.

19. Сосонкин В.Л., Мартинов Г.М. Концепция системы ЧПУ типа PCNC с открытой архитектурой // СТИН, 1998, №5, с. 7-12.

20. Сосонкин В.Л., Мартинов Г.М. Концепция открытой архитектуры персональных систем числового программного управления. Сб. тезисов 3-го Международного конгресса «Конструкторско-технологическая информатика» - КТИ-96. МГТУ «СТАНКИН», Москва, 1996.

21. Сосонкин В.Л., Мартинов Г.М. Концепция числового программного управления мехатронными системами: архитектура систем типа PCNC // Мехатроника, 2000, №1. С. 26-29.

22. Сосонкин В.Л., Мартинов Г.М. Концепция числового программного управления мехатронными системами: построение межмодульной коммуникационной среды // Мехатроника, 2000, №6. С. 2-7.

23. Сосонкин В.Л., Мартинов Г.М. Концепция числового программного управления мехатронными системами: реализация логической задачи управления // Мехатроника, 2001, №2. С. 3-7.

24. Сосонкин В.Л., Мартинов Г.М. Концепция числового программного управления мехатронными системами: реализация терминальной задачи // Мехатроника, 2000, №4. С. 2-8.

25. Сосонкин В. Л., Мартинов Г. М. Концепция числового программного управления мехатронными системами: специфика объектно-ориентированного программирования // Мехатроника, 2000, №3. С. 37-40.

26. Сосонкин В. Л., Мартинов Г. М. Концепция числового программного управления мехатронными системами: конфигурация систем ЧПУ // Мехатроника, 2000, №3. С. 37-40.

27. Сосонкин В.Л., Мартинов Г.М. Концепция числового программного управления мехатронными системами: реализация диагностической задачи управления // Мехатроника, 2001, №3. С. 2-6.

28. Сосонкин В.Л., Мартинов Г.М. Методика разработки электроавтоматики для системы ЧПУ типа PCNC// Доклады международной конференции «Информатизационные средства и технологии» Том 2 - МФИ-98, 20-22 октября 1998, Москва - М.: МГТУ СТАНКИН. 1998. С.226-231.

29. Сосонкин В.Л., Мартинов Г.М. Концепция однокомпьютерной системы ЧПУ типа PCNC// Информатика-машиностроение, 1999, №4.

30. Сосонкин В.Л., Мартинов Г.М. Подход к построению однокомпьютерной системы ЧПУ типа PCNC// Доклады международной конференции «Информатизационные средства и технологии» Том 3 - МФИ-99, 19-21 октября 1999, Москва - М.: МГТУ СТАНКИН. 1999. С.196-201.

31. Сосонкин В.Л., Мартинов Г.М. Принципы построения систем ЧПУ с открытой архитектурой // Приборы и системы управления, 1996, №8, с.18-21.

32. Сосонкин В.Л., Мартинов Г.М Проблема реального времени при разработке систем числового программного управления (ЧПУ)// Доклады международной конференции «Информатизационные средства и технологии» Том 3, 17-19 октября 2000, Москва - М.: МГТУ СТАНКИН. 2000. С.164-168.

33. Сосонкин В. Л., Мартинов Г.М. Современное представление об архитектуре систем ЧПУ типа PCNC // Автоматизация проектирования, 1998, №3, С. 35-39.

34. Сосонкин В.Л., Мартинов Г.М. «AdvancEd» - универсальная среда для редактирования, отладки и моделирования программ ЧПУ в коде ISO-7bit (любой версии) // Автотракторное электрооборудование, 2001. №1-2, С. 41-42.