

Тема 3. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ САПР

Содержание:

3.1. Свойства программного обеспечения САПР	1
3.2. Структура программного обеспечения	2
3.2.1. Пакеты прикладных программ	4
3.3. Принципы и этапы разработки ПО	12
3.4. Основные функции и состав операционных систем	14
3.5. Архитектура специального программного обеспечения	17
3.5.1. Функции основных компонентов общесистемного программного обеспечения САПР	14
3.6. Жизненный цикл программных комплексов	15
3.7. Способы описания структур и функций ПО	17
3.8. Методы разработки программного обеспечения	18
3.9. Показатели качества прикладных программ	26

Разработка программного обеспечения (ПО) является наиболее длительной и дорогостоящей частью проектирования САПР. От свойств ПО в значительной мере зависят возможности и показатели эффективности САПР.

3.1. Свойства программного обеспечения САПР

К ПО САПР предъявляются требования экономичности, удобства использования, надежности, правильности, универсальности, открытости, сопровождаемости и мобильности.

Экономичность ПО оценивается затратами вычислительных ресурсов – машинного времени и оперативной памяти. Характер их зависимости от размерности задачи определяется в первую очередь свойствами математического обеспечения. Однако неудачная программная реализация может существенно увеличить требуемое машинное время и объем оперативной памяти. Недостаточная экономичность ПО обычно оказывается основным фактором, ограничивающим возможности исчерпывающего анализа, оптимизации и структурного синтеза проектируемых объектов.

Удобство использования ПО определяется его надежностью, наличием проблемно–ориентированных входных языков и средств диагностики ошибок пользователя.

Надежность ПО – свойство выполнять заданные функции в заданных условиях. Функции и условия формулируются в терминах той предметной области, к которой относятся проектируемые объекты. Основным показателем надежности – вероятность получения правильного результата при использовании программы в сформулированных условиях.

Правильность ПО – свойство, характеризующее соответствие ПО спецификациям математического характера, т. е. правильность реализации в ПО выбранного МО. Несоответствие требований пользователей и выбранного математического обеспечения снижает надежность, но не влияет на правильность. Например, если в программе анализа статических состояний некоторого класса объектов безошибочно реализован метод Ньютона, то программа будет правильной, но, по–видимому, ненадежной, так как условия сходимости метода Ньютона будут выполняться не для любого объекта из заданного класса при произвольном задании начального приближения.

Универсальность ПО характеризуется ограничениями на применение ПО. Эти ограничения могут относиться к типам и элементному составу анализируемых или синтезируемых структур, диапазонам числовых значений внутренних и внешних параметров,

перечню выполняемых проектных операций и процедур. Универсальность связана с надежностью ПО – чем тщательнее и полнее выявлены и оговорены ограничения, чем ниже степень универсальности программы, но выше ее надежность. В САПР необходимо стремиться к достижению высокой надежности ПО. Поэтому эксплуатация нескольких узкоспециализированных, но надежных программ предпочтительнее применения одной универсальной программы, если за повышение степени универсальности приходится платить снижением надежности.

Открытость ПО характеризуется возможностями внесения в него изменений в процессе эксплуатации. Понятие открытости близко к понятию адаптируемости, под которым подразумевается возможность модификации ПО для поддержания его работоспособности и эффективности в изменяющихся условиях применения.

Сопровождаемость ПО – свойство, близкое свойству открытости, характеризует удобство поддержания ПО в работоспособном состоянии и обеспечивается структурированностью ПО и наличием необходимой эксплуатационной документации.

Мобильность ПО, называемая также переносимостью, определяется легкостью перестройки ПО, эксплуатировавшегося на ЭВМ с одной системой команд на ЭВМ с другой системой команд. Программы, записанные на машинно–ориентированных языках, непереносимы. Использование языков высокого уровня создает предпосылки для создания мобильных программ. Однако для повышения мобильности необходимы дополнительные меры по фиксации и выделению в сменяемые блоки элементов ПО, отражающих специфику архитектуры ЭВМ и связанных с ними ОС.

3.2. Структура программного обеспечения

Программой называют законченную совокупность команд, необходимых для выполнения определенной задачи, а *программированием* – процесс составления такой программы.

Программное обеспечение включает компоненты *общего* и *специального назначения*. Общее ПО не отражает специфики конкретной САПР и включает в себя компоненты, обеспечивающие организацию и контроль вычислительного процесса, автоматизацию трудоемких этапов подготовки и отладки прикладных программ. Компоненты общего ПО не разрабатываются при создании конкретной САПР.

Общее ПО САПР в свою очередь делится на *общесистемное* и *базовое*. *Общесистемное ПО* представлено операционными системами и характеризуется тем, что его компоненты не отражают специфики конкретной САПР и являются промышленно сопровождаемыми программно–методическими комплексами. Оно также может включать средства телекоммуникационного доступа

Базовое ПО характеризуется тем, что его компоненты отражают специфику САПР данного класса, однако не разрабатываются при их создании, а приобретаются как комплектующие изделия наравне с компонентами общесистемного ПО и предназначены для использования многими проектными организациями. Типичными примерами базового ПО является ПО обслуживающих подсистем САПР – графических редакторов, СУБД, диалоговых мониторов и т.п.

Специальное (прикладное) ПО отражает специфику конкретной САПР и содержит совокупность целевых программных средств, реализующих математическое обеспечение данной САПР для непосредственного выполнения проектных процедур. Прикладное ПО разрабатывается при создании конкретной САПР и имеет форму либо автономных ППП, каждый из которых обеспечивает отдельный этап автоматизированного проектирования, либо функционально законченных программных модулей (комплексов), работающих под управлением единой мониторинной системы. Иногда ППП типовых проектных процедур, поставляемые в централизованном порядке, относят к базовому ПО.

Укрупненная структура ПО, относящегося к одному из уровней САПР, представлена на рис. 3.1, а.

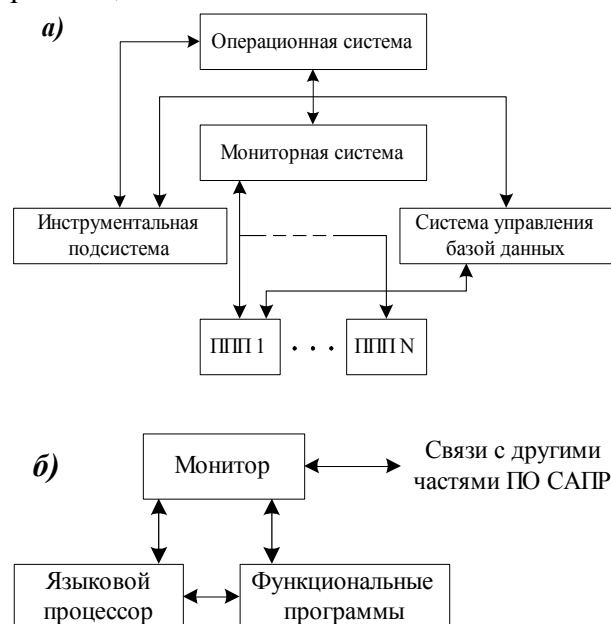


Рис. 3.1. Структура программного обеспечения САПР (а), программной системы (б)

При структурировании ПО используют понятия ППП, программных систем, комплексов и компонентов.

Пакет прикладных программ – совокупность программ, объединенных общностью применения, т. е. возможностью совместного исполнения или ориентацией на определенный класс задач.

Комплекс по определению в Единой системе программной документации (ЕСПД) – сложная программа, которую можно разделить на составные части.

Компоненты – составные части программ, имеющие свое функциональное назначение.

Понятия "комплекс – компонент" аналогичны понятиям "система – элемент" в блочно–иерархическом проектировании сложных систем, следовательно, на каждом иерархическом уровне проектирования ПО эти понятия наполняются своим конкретным содержанием. Так, операционная система MS DOS – комплекс, а служебные утилиты – его компоненты.

Различают несколько типов ППП в зависимости от состава пакета. *Пакеты прикладных программ простой структуры* характеризуются наличием только обрабатывающей части – набора функциональных программ, каждая из которых предназначена для выполнения некоторой проектной операции или процедуры. Объединение нужных программ для реализации маршрутов проектирования происходит средствами операционной системы ЭВМ или мониторинной системы САПР на основе соответствующего языка управления заданиями. Совместное исполнение этих программ определяется возможностями организации их информационного интерфейса. В ППП простой структуры организация информационного интерфейса в значительной мере возлагается на пользователя. Такие пакеты просты в разработке, но неразвитость средств управления, информационного интерфейса и отсутствие удобного лингвистического обеспечения ограничивают их возможности. Они применяются лишь на маршрутах проектирования, на

которых последовательно выполняемые операции достаточно автономны, информационные связи программ между собой и с пользователем достаточно слабые.

Пакеты прикладных программ сложной структуры и программные системы появились в результате развития прикладного ПО. В первых из них имеется собственная управляющая часть, называемая *монитором*, а во вторых кроме этого – языковой процессор с проблемно-ориентированным входным языком. Программные системы (рис.3.1, б) вместе с соответствующим лингвистическим и информационным обеспечением, предназначенные для реализации функций подсистемы САПР, называются *программно-методическими комплексами* (ПМК) САПР.

Непосредственно на маршруте проектирования выполняется рабочая программа, составляемая системой программирования из функциональных программ и модуля, являющегося результатом трансляции или интерпретации информации, заданной на входном языке пакета. Функциональные программы чаще всего организуются по библиотечному принципу. Примеры библиотек функциональных программ: математических моделей типовых элементов, типовых численных методов решения различных групп задач, элементарных математических функций и функционалов, операций статистического анализа и обработки результатов экспериментов, элементарных графических операций и т. п.

Управляющая часть ПО имеет иерархическую организацию и в общем случае в ней можно выделить уровни операционной системы вычислительной сети, операционных систем отдельных ЭВМ, мониторных систем САПР и мониторов отдельных ППП. Основные функции управляющей части:

- * связь с пользователем в режиме диалога, иногда эта функция реализуется самостоятельной диалоговой подсистемой САПР;
- * планирование вычислительного процесса путем выбора нужных функциональных программ, настройка их связей по информации и управлению;
- * распределение вычислительных ресурсов, например, динамическое распределение оперативной памяти, распределение задач по уровням технических средств и т. п.;
- * управление исполнением рабочей программы.

Инструментальная подсистема предназначена для облегчения и ускорения процесса разработки и сопровождения ПО САПР. В ее состав могут входить средства отладки программ, оформления и редактирования программной документации и др.

3.2.1. Пакеты прикладных программ

Одним из условий эффективного внедрения вычислительной техники в практику является создание специализированных пакетов прикладных программ (ППП). Доступность и простота использования их создает предпосылки более широкого внедрения ЭВМ в инженерный труд, решение конкретных задач научной области, экономики, культуры, образования.

Пакеты прикладных программ обычно строятся на базе специальных систем и являются дальнейшим их развитием в конкретном направлении. Они поставляются отдельно от программного обеспечения вычислительных средств, имеют свою документацию и не входят в состав операционных систем. Многие пакеты имеют собственные средства генерации. Разработка

пакета не должна требовать модификации операционных систем. Это относится к пакетам, влияющим на работу управляющих программ. Если пакет требует внесения изменений в управляющую программу, то это выполняется в процессе загрузки и инициализации пакета.

Все ППП могут быть разбиты на три группы: пакеты, расширяющие возможности операционных систем; пакеты общего назначения; пакеты, ориентированные на работу в АСУ.

Пакеты прикладных программ, реализующие возможности операционных систем, обеспечивают функционирование ЭВМ различных конфигураций. К ним относятся пакеты, обеспечивающие работу многомашинных комплексов типовых конфигураций, диалоговые системы, системы для работы в реальном масштабе времени, удаленную пакетную обработку.

Пакеты прикладных программ общего назначения включают в себя набор программ для широкого круга применений: для алфавитно-цифровых и графических дисплеев, графопостроителей, систем программирования для языков программирования, систем программирования для специальных языков, а также для научно-технических расчетов, математического программирования, обработки матриц, различного вида моделирования, решения задач теории массового обслуживания и т.д.

Пакеты, ориентированные на работу в АСУ, включают в себя набор программ для общецелевых систем обработки банков данных; информационно-поисковых систем общего назначения, систем обработки документов.

Пакеты прикладных программ являются наиболее динамично развивающейся частью программного обеспечения: круг решаемых с помощью ППП задач постоянно расширяется. Во многом внедрение компьютеров практически во все сферы деятельности стало возможным благодаря появлению новых и совершенствованию существующих ППП.

Достижения в области микроэлектроники, приводящие к появлению более мощных по своим функциональным возможностям компьютеров, также являются причиной создания новых ППП. В свою очередь, необходимость улучшения характеристик использования пакета при решении конкретных задач пользователя стимулирует совершенствование архитектуры и элементной базы компьютеров и периферийных устройств.

Структура и принципы построения ППП зависят от класса ЭВМ и операционной системы, в рамках которой этот пакет будет функционировать. Наибольшее количество разнообразных ППП создано для IBM PC-совместимых компьютеров с операционными системами MS DOS и Windows. Классификация этих пакетов программ по функционально-организационному признаку представлена на рис. 3.2.

Каждая группа пакетов имеет свои проблемы организации, трудности разработки и создания. Каждый пакет в зависимости от ЭВМ и его назначения реализуется на конкретном языке программирования в соответствии с требованиями, предъявленными к пакету, и возможностями языка.

В приведенной классификации не указаны игровые программы — они не являются инструментом для автоматизации, профессиональной деятельности и предназначены для доступа. Отсутствие программ-переводчиков, орфографии, электронных словарей связано с тем, что эти программы являются функциональным дополнением ППП типа редактора текста,

презентации и т.п. Наблюдается тенденция включения этих программ в состав пакетов прикладных программ.

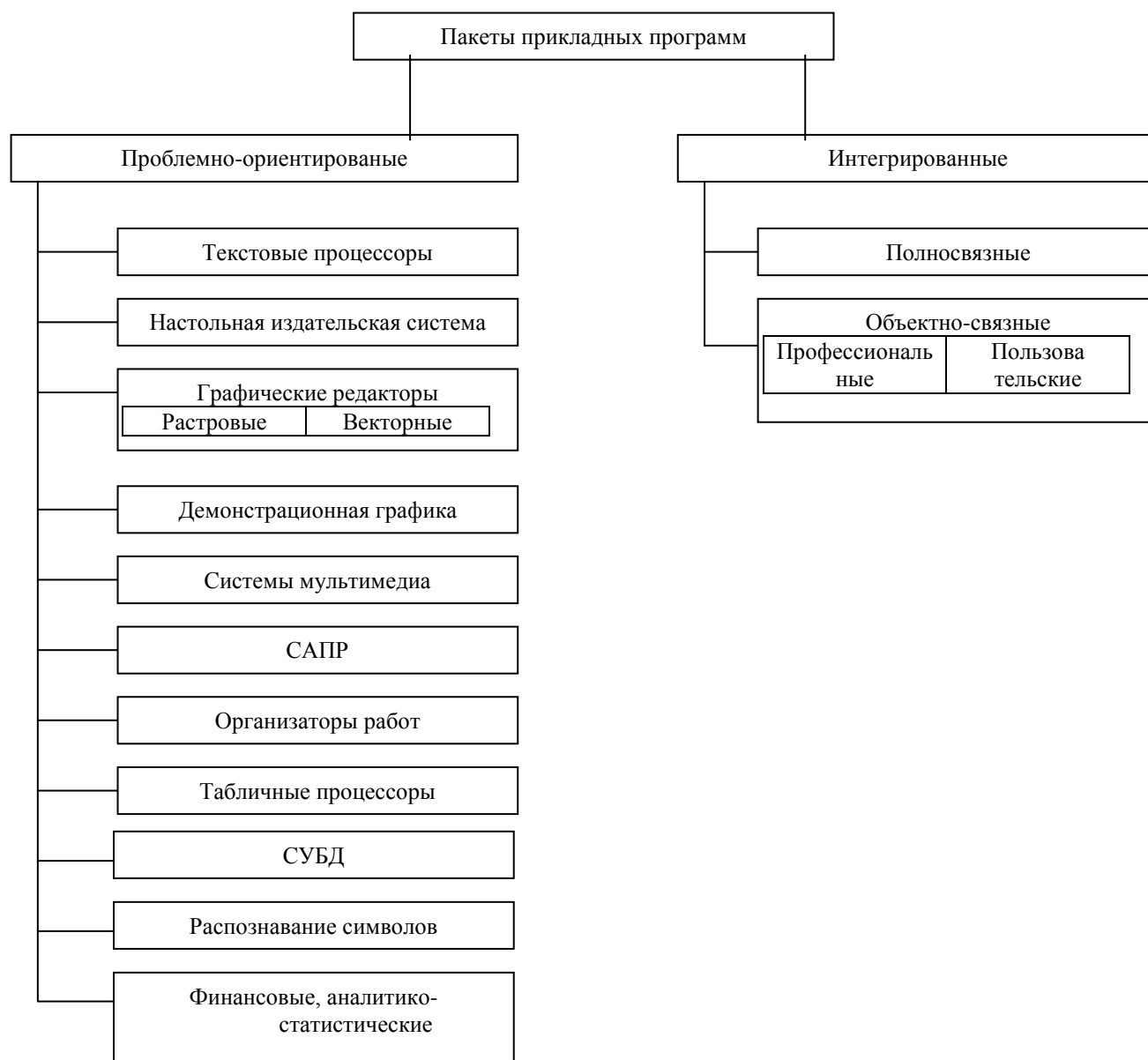


Рис.3.2. Классификация ППП

Существующие ППП охватывают почти все сферы человеческой деятельности, связанной с обработкой информации. Развитие и совершенствование ППП — поступательный процесс, поэтому следует ожидать появления новых ППП, возможности которых превзойдут достижения настоящих пакетов.

Проблемно-ориентированные ППП — наиболее развитая в плане реализуемых функций и многочисленная по количеству созданных пакетов часть ППП. Она включает следующие проблемно-ориентированные программные продукты: текстовые процессоры, издательские системы, графические редакторы, демонстрационную графику, системы мультимедиа, ПО-САПР, организаторы работ, электронные таблицы (табличные процессоры), системы управления базами

данных, программы распознавания символов, финансовые и аналитико-статистические программы.

Текстовые процессоры — специальные программы, предназначенные для работы с документами (текстами), позволяющие компоновать, форматировать, редактировать тексты при создании пользователем документа. Обычно они включают в себя дополнительные функции по работе с блоками текста и объектами. Признанными лидерами в части текстовых процессоров для ПЭВМ являются MS WORD, WordPerfect, AmiPro.

Настольные издательские системы (НИС) — программы, предназначенные для профессиональной издательской деятельности и позволяющие осуществлять электронную верстку широкого спектра основных типов документов, типа информационного бюллетеня, краткой цветной брошюры и объемного каталога или торговой заявки, справочника. Предусмотренные в пакетах данного типа средства позволяют:

- компоновать (верстать) текст;
- использовать всевозможные шрифты и осуществлять полиграфическое изображения;
- осуществлять редактирование текста на уровне лучших текстовых процессоров;
- обрабатывать графические изображения;
- обеспечивать вывод документов полиграфического качества;
- работать в сетях и на разных платформах.

Наилучшими пакетами в этой области для ПЭВМ являются: Corel Ventura, PageMaker, QuarkXPress, FrameMaker, Microsoft Publisher, PagePlus, CompuWork Publisher.

Графические редакторы — пакеты, предназначенные для обработки графической информации. Они делятся на ППП обработки растровой графики и изображений и векторной графики.

ППП первого типа предназначены для работы с фотографиями и включают в себя набор средств по кодированию фотоизображений в цифровую форму. Признанный лидер среди пакетов данного класса — Adobe Photoshop. Известны также пакеты Aldus Photo Styler, Picture Publisher, Photo Works Plus. Все программы ориентированы на работу в среде Windows.

Пакеты, для работы с векторной графикой предназначены для профессиональной работы, связанной с художественной и технической иллюстрацией с последующей цветной печатью (на рабочем месте дизайнеров, например), занимают промежуточное положение между пакетами для систем автоматизированного проектирования (САПР) и настольными издательскими системами.

Пакеты данного класса в настоящее время обладают достаточно широким набором функциональных средств для осуществления сложной точной обработки графических изображений и включают в себя:

- инструментарий для создания графических изображений;
- средства выравнивания (по базовой линии и странице, по сетке, пересечению, ближайшей точке и т.п.);
- средства манипулирования объектами;
- средства обработки текста в части оформления и модификации параграфов, работы с различными шрифтами;

- средства импорта (экспорта) графических объектов (файлов) различных форматов;
- средства вывода на печать с соответствующей настройкой экранного образа на полиграфическое исполнение;
- средства настройки цвета.

Своеобразным стандартом в этом классе является пакет CorelDraw. Можно также отметить такие пакеты, как Adobe Illustrator, Aldus Free Hand, Professional Draw.

Электронные таблицы (табличные процессоры) — пакеты программ, предназначенные для обработки табличным образом организованных данных. Пользователь имеет возможность с помощью средств пакета осуществлять разнообразные вычисления, строить графики, управлять форматом ввода-вывода данных, компоновать данные, проводить аналитические исследования и т.п.

В настоящее время наиболее популярными и эффективными пакетами данного класса являются Excel, Improv Pro, 1-2-3.

Организаторы работ — это пакеты программ, предназначенные для автоматизации процедур планирования использования различных ресурсов (времени, денег, материалов) как отдельного человека, так и всей фирмы или ее структурных подразделений. Целесообразно выделить две разновидности пакетов данного класса: управления проектами и организации деятельности отдельного человека.

Пакеты первого типа предназначены для сетевого планирования и управления проектами. Достаточно простые и удобные в использовании, эти программные средства позволят быстро спланировать проект любой величины и сложности, эффективно распределить людские, финансовые и материальные ресурсы, составить оптимальный график работ и проконтролировать его исполнение.

К пакетам данного типа относятся: Time Line, MS Project, CA-Super Project.

Пакеты второго типа представляют собой своего рода электронных помощников делового человека. Такие пакеты, как Lotus Organizer, АСТІ, выполняют функции электронных секретарей и предназначены для эффективного управления деловыми контактами.

Системы управления базами данных (СУБД) предназначены для автоматизации процедур создания, хранения и извлечения электронных данных. Многие существующие экономические, информационно-справочные, банковские, программные комплексы реализованы с использованием инструментальных средств СУБД.

Для различных классов компьютеров и операционных средств разработано множество СУБД, отличающихся по способу организации данных, формату данных, языку формирования запросов. Наиболее распространенными пакетами для ПЭВМ типа IBM PC являются dBase, Paradox, Microsoft Access, Oracle.

Пакеты демонстрационной графики являются конструкторами графических образов деловой информации, призванные в наглядной и динамической форме представлять результаты некоторого аналитического исследования.

Работа с пакетами этого типа строится по следующей схеме: разработка общего плана представления, выбор шаблона для оформления элементов, формирование и импорт элементов, таких, как текст, графики, таблицы, диаграммы, звуковые эффекты и видеоклипы. Программы

просты в работе и снабжены интерфейсом, почти не требующим дополнительного изучения. К наиболее популярным пакетам данного типа относятся Power Point, Harvard Graphics, WordPerfect Presentations, Freelance Graphics.

Пакеты программ мультимедиа предназначены для использования ПЭВМ для отображения и обработки аудио- и видеоинформации. Помимо программных средств компьютер при этом должен быть оборудован дополнительными платами, позволяющими осуществлять ввод-вывод аналоговой информации, ее преобразование в цифровую форму.

Программы мультимедиа для ПЭВМ появились сравнительно недавно благодаря значительному росту вычислительных возможностей ПК и большим достижениям в области производства оптических дисков. Дело в том, что при представлении аналоговой информации в цифровом виде требуются огромные объемы памяти: несколько минут видеофильма занимают десятки мегабайт памяти. Естественно, что работа с таким большим файлом возможна лишь при наличии быстродействующего процессора (желательно использовать ПК с М8С-процессором и быстродействующей шиной данных). Кроме того, распространение таких мультимедиа-приложений невозможно на традиционных магнитных дискетах, для этого необходимо использовать оптические компакт-диски (CD-ROM).

Среди мультимедиа-программ можно выделить две небольшие группы. Первая включает пакеты для обучения и досуга. Поставляемые на CD-ROMах емкостью от 200 до 500 Мбайт каждый, они содержат аудиовизуальную информацию по определенной тематике. Разнообразие их огромно, и рынок этих программ постоянно расширяется при одновременном улучшении качества видеоматериалов. Так, созданы и продаются электронные энциклопедии по отраслям знаний; электронные учителя в области иностранных языков, бизнеса, политики; деловые и авантюрные игры.

Вторая группа включает программы для подготовки видеоматериалов для создания мультимедиа представлений, демонстрационных дисков и стендовых материалов.

К пакетам данного вида относятся Director for Windows, Multimedia Viewer Kit, NEC MultiSpin.

Системы автоматизации проектирования — другая разновидность пакетов программ, связанная с обработкой графических изображений. Они предназначены для автоматизации проектно-конструкторских работ в машиностроении, автомобилестроении, промышленном строительстве и т.п. Пакеты САПР обладают набором инструментальных средств, обеспечивающих реализацию следующих основных функций:

- коллективная работа в сети пользователей с пакетом;
- экспорт-импорт файлов всевозможных форматов;
- масштабирование объектов;
- управление объектами в части их группировки, передвижения с растяжкой, поворота, разрезание, изменение размеров, работа со слоями;
- перерисовка (фоновая, ручная, прерываемая); »
- управление файлами в части библиотек и каталогов чертежей;

- использование разнообразных чертежных инструментов, позволяющих рисовать кривые, эллипсы, произвольной формы линии, многоугольники и т.п., использование библиотеки символов, выполнение надписей и т.д.;

- работа с цветом;

- автоматизация отдельных процедур с использованием встроенного макроязыка.

Своеобразным стандартом среди программ данного класса являются пакеты AutoCad фирмы Autodesk.

Программы распознавания символов предназначены для перевода графического изображения букв и цифр в ASCII- коды этих символов. Используются, как правило, совместно со сканерами.

Пакеты данного типа обычно включают разнообразные средства, облегчающие работу пользователя и повышающие вероятность правильного распознавания.

Скорость сканирования современных ППП составляет примерно 1,5 мин на страницу. К пакетам данного типа относятся Fine Reader, CunieForm, OmniPage.

Разнообразными пакетами представлена группа *финансовых программ*: для ведения деловых записей типа записной книжки и расчета финансовых операций (баланс денежных средств, определение процентных выплат по займам и кредитам, временная структура денежных вложений и т.п.).

Для расчета величины налогов можно использовать программы TurboTax for Windows, Personal Tax Edge.

С помощью программ Quicken, DacEasy Accounting, Peachtree for Windows можно автоматизировать бухгалтерский учет. Эту же функцию выполняет ряд отечественных программ: «Турбобухгалтер», «1С: Бухгалтерия», «Бухгалтер» фирмы «Атлант-Информ» и др.

Для аналитических исследований используются хорошо зарекомендовавшие себя зарубежные статистические пакеты, такие, как StatGraphics или Systas, или отечественная разработка «Статистик-Консультант». Однако в коммерческих фирмах внедрение этих пакетов сдерживается отсутствием соответствующим образом подготовленных специалистов, высокой ценой пакетов и широким внедрением табличных процессов, с помощью которых можно провести простейшие, но наиболее часто используемые статистические расчеты.

Интегрированные пакеты программ — по количеству наименований продуктов немногочисленная, но в вычислительном плане довольно мощная и активно развивающаяся часть ПО.

Идея создания интегрированных программных комплексов не нова и в той или иной мере была реализована во всех поколениях ЭВМ.

Внимание к этой проблеме объясняется как расширением сферы применения вычислительной техники, так и стремлением фирм-разработчиков программного обеспечения не «потерять» своих клиентов с переходом на более совершенные системы обработки данных.

Традиционные, или *полносвязанные*, интегрированные комплексы представляют собой многофункциональный автономный пакет, в котором в одно целое соединены функции и возможности различных специализированных (проблемно-ориентированных) пакетов, родственных в смысле технологии обработки данных на отдельном рабочем месте. Типичными

представителями таких программ являются пакеты Framework, Symphony, а также пакеты нового поколения Microsoft Word, Lotus Works.

В этих программах происходит интеграция функций редактора текстов, системы управления базами данных и табличного процессора. В целом стоимость такого пакета гораздо ниже суммарной стоимости аналогичных специализированных пакетов.

В рамках интегрированного пакета обеспечивается связь между данными, однако при этом сужаются возможности каждой компоненты по сравнению с аналогичным специализированным пакетом. Интерфейс более ранних программ был перегружен различными средствами обмена данными и описаниями среды работы, что требовало от пользователя определенных навыков и знаний в части переключения режимов пакета, форматов данных, принципов хранения и манипулирования различными типами данных, что в конечном счете снижало привлекательность пакетов. В современных пакетах (например, Microsoft Works) этот недостаток изжит: простота интерфейса позволяет применять его без предварительного обучения персонала.

В настоящее время активно реализуется другой подход интеграции программных средств: объединение специализированных пакетов в рамках единой ресурсной базы, обеспечение взаимодействия приложений (программ пакета) на уровне объектов и единого упрощенного центра-переключения между приложениями. Интеграция в этом случае носит *объектно-связанный характер*.

Типичные и наиболее мощные пакеты данного типа: Borland Office for Windows, Lotus SmartSuite for Windows, Microsoft Office. В профессиональной редакции этих пакетов присутствуют четыре приложения: текстовый редактор, СУБД, табличный процессор, программы демонстрационной графики. Целесообразность создания таких пакетов, очевидно, связана с желанием получить дополнительный эффект от интеграции по отношению к простой сумме составляющих его компонент. Этот эффект должен достигаться за счет согласованного взаимодействия компонент в процессе работы пользователя. При традиционном подходе к интеграции программ этот выигрыш может быть легко сведен на нет отсутствием нужной пользователю функции, присутствующей в специализированном пакете, и необходимостью в пусть небольшом, но дополнительном обучении.

Особенностью нового типа интеграции пакетов является использование общих ресурсов. Здесь можно выделить четыре основных вида совместного доступа к ресурсам:

- использования утилит, общих для всех программ комплекса. Так, например, утилита проверки орфографии доступна из всех программ пакета;
- применение объектов, которые могут находиться в совместном использовании нескольких программ;
- реализация простого метода перехода (или запуска) из одного приложения к другому;
- реализация построенных на единых принципах средств автоматизации работы с приложением (макроязыка), что позволяет организовать комплексную обработку информации при минимальных затратах на программирование и обучение программированию на языке макроопределении.

Совместное использование объектов с несколькими приложениями — краеугольный камень современной технологии интеграции программ и манипулирования данными. Разработаны два основных стандарта в этой области:

- динамической компоновки и встраивания объектов Object Linking and Embedding OLE 2.0 фирмы Microsoft;

- OpenDoc (открытый документ) фирм Apple, Borland, IBM, Novell и WordPerfect.

Механизм динамической компоновки объектов дает возможность пользователю помещать информацию, созданную одной прикладной программой, в документ, формируемый в другой. При этом пользователь может редактировать информацию в новом документе средствами того продукта, с помощью которого этот объект был создан (при редактировании автоматически запускается соответствующее приложение). Запущенное приложение и программа обработки документа-контейнера выводит на экран «согласованные» меню, часть пунктов которого принадлежит одной программе, а другая часть — другой.

Кроме того, данный механизм позволяет переносить OLE-объекты из окна одной прикладной программы в окно другой.

В этой технологии предусмотрена также возможность общего использования функциональных ресурсов программ: например, модуль построения графиков табличного процессора может быть использован в текстовом редакторе.

Недостатком данной технологии является ограничение на размер объекта размером одной страницы.

OpenDoc представляет собой объектно-ориентированную систему, базирующуюся на открытых стандартах фирм-участников разработки. В качестве модели объекта используется распределенная модель системных объектов (DSOM-Distributed System Object Model), разработанная фирмой IBM для OS/2. Предполагается совместимость между OLE и OpenDoc.

3.3. Принципы и этапы разработки ПО

Программное обеспечение САПР разрабатывается в соответствии с основными принципами блочно-иерархического проектирования сложных систем — модульности (блочности) и иерархичности.

Модуль — структурная составляющая ПО, рассматриваемая как единое целое на определенных стадиях разработки или в процессе эксплуатации.

Принципы модульности и иерархичности позволяют организовывать коллективную параллельную разработку различных частей ПО, создавать открытые программные системы, облегчают их комплексную отладку и информационное согласование.

Выделяют следующие иерархические уровни представления и соответственно нисходящего проектирования ПО:

- * системный,
- * прикладных программ,
- * подпрограмм.

Системный уровень — на нем конкретизируются функции ПМК, планируется его структура и состав, выбираются или разрабатываются языки проектирования, устанавливается степень

использования доступного для приобретения готового общесистемного и базового ПО, разрабатываются спецификации на отдельные программы пакета.

Уровень прикладных программ – на нем выбирается математическое обеспечение, разрабатываются специфические алгоритмы, устанавливается модульная структура программ, выбираются структуры данных, способы информационного интерфейса и язык программирования, разрабатываются спецификации на отдельные программные модули.

Уровень подпрограмм (модулей) – на нем производится конкретизация типов и структур данных, осуществляется кодировка алгоритмов – их запись на выбранном языке программирования.

При разработке ПО крупных САПР возможно выделение дополнительных промежуточных уровней.

Процесс проектирования ПО состоит из нескольких этапов (рис.3.3). Этапы 1...4 относятся к синтезу ПО и выполняются в нисходящей последовательности, этапы 3...7 относятся к отладке и выполняются в восходящей последовательности.

На этапе 1 анализа требований выявляются функции будущего ПМК и формулируется ТЗ на систему, учитывающее требования пользователей и понятное разработчикам ПО.

На этапе 2 разрабатываются спецификации на отдельные программы ПМК. Различают несколько разновидностей спецификаций.

Функциональные спецификации содержат описания функций по переработке информации, которые должна выполнять программа.

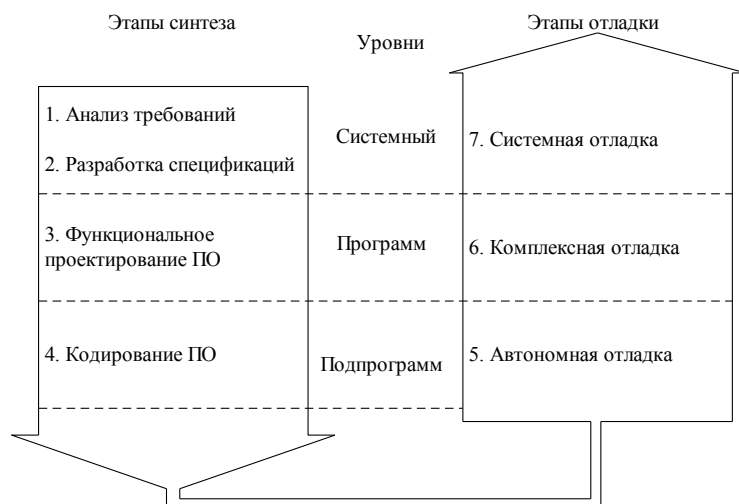


Рис. 3.3. Этапы проектирования программного обеспечения

прикладных программ и подпрограмм.

На этапах 5, 6, 7 осуществляется отладка, цель которой – обнаружение и устранение ошибок, допущенных на этапах синтеза ПО. Отладка выполняется с помощью процедур выбора тестов и верификации. Тесты представляют собой контрольные задачи с известными правильными результатами решения. Поэтому верификация программ путем исполнения тестов на отлаживаемой программе позволяет при несовпадении полученных результатов с правильными констатировать наличие ошибок в программе. Совпадение результатов является необходимым, а не достаточным условием правильности ПО. Создание полных тестов, т. е.

Эксплуатационные спецификации включают требования к быстродействию, надежности ПО, ограничения на используемые вычислительные ресурсы.

Спецификации по ЕСПД содержат описание состава ПО и перечень требуемой программной документации. На этапе 2 разрабатываются прежде всего функциональные спецификации, хотя в них могут фигурировать элементы эксплуатационных спецификаций.

На этапах 3 и 4 решаются описанные выше задачи уровней

таких, правильное исполнение которых гарантирует отсутствие ошибок в тестируемом ПО, существующими методами не обеспечивается, что обуславливает сложность отладки ПО.

Частичная верификация разрабатываемых алгоритмов и структур ПО возможна в рамках этапа 3, но основные процедуры отладки выполняются после кодирования и составляют самостоятельные этапы. Поскольку при разработке ПО не обеспечивается гарантия его правильности и надежности, процесс выявления и устранения ошибок продолжается и в процессе сопровождения ПО. Статистические данные показывают, что в ряде случаев затраты на сопровождение ПО превышают затраты на остальные этапы разработки ПО.

3.4. Основные функции и состав операционных систем

Операционная система — комплекс системных управляющих и обрабатывающих программ, предназначенных для наиболее эффективного использования всех ресурсов вычислительной системы (ВС) и удобства работы с ней.

В настоящее время только с помощью ОС можно полностью загружать высокопроизводительные ВС с их быстродействием в несколько миллионов операций в секунду. В программном обеспечении ВС операционная система занимает основное положение, поскольку осуществляет планирование и контроль всего вычислительного процесса. Любая из компонент программного обеспечения обязательно работает под управлением ОС. Современный пользователь не представляет себе возможности общения с ВС без посредства ОС, поскольку последняя предоставляет ему множество сервисных услуг для редактирования текстов, отладки программ, организации диалога, работы с файлами и других вычислительных процедур.

Первоначально прототипы современных ОС создавались как средство, освобождающее операторов ЭВМ второго поколения от рутинных работ по установке лент и колод перфокарт на соответствующие внешние устройства, загрузке программ для исполнения, обработке ошибок при чтении данных и сбоях процессора, составлению очередности прохождения отдельных заданий, перемотке лент и т.д.

Качественный скачок от сравнительно простых управляющих программ к современным сложным ОС произошел с появлением *режима мультипрограммной обработки задач*. Реализация этого режима оказалась возможной благодаря совмещению операций счета и обмена информацией.

Идея мультипрограммирования заключается в том, что в оперативной памяти современной ЭВМ находится сразу несколько задач, обслуживаемых центральным процессором по очереди. На время, необходимое в данной задаче для обмена информацией между оперативной памятью и внешним устройством, процессор переключается на обслуживание других задач.

Для правильного планирования и организации вычислительного процесса проектировщикам ОС приходится писать многочисленные и сложные модули обработки всевозможных прерываний, создавать дисциплину обслуживания задач в соответствии с их приоритетами, постоянно контролировать занятые и свободные области оперативной памяти, рационально распределять ее между конкурентными задачами, защищать наборы данных на внешних носителях от несанкционированного доступа, распределять между задачами

ограниченное число внешних устройств и т. д. Естественно, что в результате получается очень сложная и громоздкая ОС, что порождает негативные стороны: трудность освоения и эксплуатации, значительные затраты вычислительных ресурсов расходятся не на решение пользовательских задач, а на удовлетворение потребностей ОС. Но тем не менее без ОС невозможно эффективное функционирование современной вычислительной системы.

Кроме рационального распределения всех ресурсов и увеличения пропускной способности вычислительной системы ОС предоставляет пользователю различные сервисные услуги: стандартные методы доступа, утилиты, средства отладки, теледоступа и подробной диагностики всех этапов прохождения задачи, возможности получения аварийных дампов и пр.

Ознакомление с любой ОС следует начинать с изучения архитектуры соответствующей ЭВМ, т. е. получения представления о функционировании ЭВМ с точки зрения программиста.

Понятие архитектуры включает в себя: представление об оперативной памяти ЭВМ; форматы представления команд и данных; систему адресации, принятую в ЭВМ; способы отражения текущего состояния вычислительной системы; механизм обработки прерываний; организацию обменов информацией между оперативной памятью и внешними устройствами.

Все перечисленные характеристики ЭВМ, входящие в понятие ее архитектуры, определяются ее аппаратным исполнением и обуславливают определенные ограничения, в рамках которых должна функционировать ОС.

По назначению различают ОС **общего и специального назначения**. К ОС специального назначения относят ОС:

- * предназначенные для решения задач реального времени;
- * ориентированные на организацию и ведение баз данных;
- * предназначенные для поддержки однородных вычислительных структур и сетей.

По режиму обработки задач различают

- * ОС, обеспечивающие однопрограммный режим обработки задач,
- * и ОС, обеспечивающие мультипрограммный режим обработки задач.

Разновидностью мультипрограммного режима обработки задач является режим разделения времени.

По способу взаимодействия с пользователем можно выделить ОС:

- * взаимодействующие с пользователем в режиме пакетной обработки задач;
- * взаимодействующие с пользователем в режиме диалога.

При этом различают частичный диалог на этапе подготовки задач к решению и сквозной диалог на всех этапах обработки задачи. В свою очередь ОС, взаимодействующие с пользователем в режиме диалога, подразделяются на однопользовательские и многопользовательские.

Руководствуясь перечисленными выше основными характеристиками, существующие ОС можно объединить в несколько групп.

Операционные системы общего назначения, обеспечивающие однопрограммный режим обработки задач и диалоговый способ общения. Эти ОС включают в себя средства, обеспечивающие ввод и вывод информации, управляют работой системных обрабатываемых программ, трансляторов, редакторов, предоставляют пользователю сведения о ходе выполнения

задач, обеспечивают работу с библиотеками. Обычно такие операционные системы называют *мониторными*. Они не повышают производительности ЭВМ, но позволяют программисту вмешиваться в ход выполнения задания, что резко повышает производительность его работы, особенно на этапе отладки программ. Используются только в персональных, микро– и мини–ЭВМ.

Операционные системы общего назначения, обеспечивающие пакетную обработку задач в режиме мультипрограммирования. Их применяют в вычислительных системах средней и большой производительности. В оперативной памяти ЭВМ одновременно находится несколько системных и пользовательских задач, и когда одна из них обрабатывается центральным процессором, то для остальных осуществляются необходимые обмены информацией с внешними устройствами.

Операционные системы разделения времени. Их относят к ОС общего назначения, обеспечивающих мультипрограммный режим обработки задач и многопользовательский интерактивный способ общения. Термин "*разделение времени*" обусловливается особым методом реализации мультипрограммирования и коллективного диалогового доступа пользователей к системе и своим задачам.

Операционные системы реального времени. Эти ОС отличаются от ОС общего назначения в первую очередь тем, что поступающая в систему информация обязательно должна быть обработана в течение заданных интервалов времени (эти интервалы времени нельзя превышать). Еще одно отличие работы ОС общего назначения от работы ОС реального времени заключается в том, что в первой из них поток пользовательских задач планомерный и регулируется оператором ЭВМ, а во второй запросы на обработку могут поступать в непредсказуемые моменты времени. Поэтому ОС реального времени должна обеспечить некоторые дополнительные возможности, например создание постоянных задач.

К вычислительным системам, работающим в режиме реального времени, предъявляются высокие требования по надежности. Соответственно ОС должна располагать средствами, обеспечивающими быстрое обнаружение сбоев или аварийных ситуаций и выход из них, иметь возможность выключать неисправные устройства и включать резервные, сообщая об этом оператору ЭВМ.

Операционные системы, предназначенные для организации работы вычислительных сетей. Работа ОС в вычислительной сети характеризуется определенными особенностями. Главной из них является необходимость организации передачи данных внутри вычислительной сети. Любая информация внутри вычислительной сети передается отдельными порциями – блоками данных. Основные требования, предъявляемые к ОС по передаче блоков данных, можно сформулировать следующим образом:

- * блоки данных должны циркулировать в сети асинхронно и независимо в обоих направлениях между источником сообщения и его адресатом;
- * ОС должны осуществлять контроль за прохождением блока данных в течение всего периода его пребывания в сети;
- * необходимы программные и аппаратные средства, предотвращающие потерю или искажение блоков данных при одновременном нахождении их в вычислительной сети;

* ОС должны включать в себя механизм обнаружения повторных, потерянных или ошибочных блоков данных в вычислительной сети.

Блок данных должен состоять из заголовка, содержащего служебную информацию, и собственно текста. Служебная информация включает в себя идентификатор задачи, идентификатор пользователя, приоритет, определяющий прохождение блока данных по сети, адрес конечной ЭВМ, которой предназначается блок данных, и т. п.

Все процедуры ОС, направленные на создание из разнородных машин и терминалов единой вычислительной сети, осуществляются с помощью протоколов.

Операционная система обычно предоставляет широкие возможности по программированию и отладке прикладных программ САПР. Эти функции осуществляются системой программирования, включающей, как правило, следующие компоненты:

- * компиляторы с основных языков программирования (Ассемблер, Паскаль, Си и др.);
- * редактор связей и загрузчики программ, которые объединяют основную программу с подпрограммами и подготавливают их к выполнению;
- * средства отладки (например, программы трассировки и распечатки памяти);
- * программы управления данными и обработки.

Важной частью операционной системы является также набор сервисных программ (сортировки, объединения, копирования и т. д.), а также ППП, расширяющие возможности ОС (организация банков данных, программное обеспечение диалоговых систем и машинной графики и др.).

3.5. Архитектура специального программного обеспечения

ПО САПР разрабатывается после создания математического, лингвистического и информационного обеспечения. При этом ПО включает в себя последовательное решение следующих задач:

Выбор или разработка входных языков для всех подсистем САПР.

Проектирование схем алгоритмов с уточнением методов, алгоритмов, моделей, входящих в математическое обеспечение, построение иерархической структуры ПО САПР с разделением ПО на модули.

Выбор типа ППП и языков программирования для модулей всех уровней; разделение функций управления между операционной системой и управляющими программами пакетов.

Уточнение содержимого централизованной базы данных, составление списка массивов переменной части базы данных, выбор структуры этих массивов; составление технических заданий на разработку модулей.

Программирование модулей, выявление необходимости введения частных баз данных и при их наличии разработка информационного интерфейса.

Разработка тестовых задач макетов нижестоящих модулей и проведение тестирования модулей.

Рассмотрим один из возможных вариантов структуры ПО САПР. Программно—

технический комплекс такой САПР имеет многотерминальную радиальную структуру, реализован на базе ЭВМ высокой производительности (рис.3.4).

К *общесистемному ПО* САПР относятся следующие компоненты (на рис.3.4 показаны с тенью):

- * операционная система универсальной или специализированной ЭВМ;

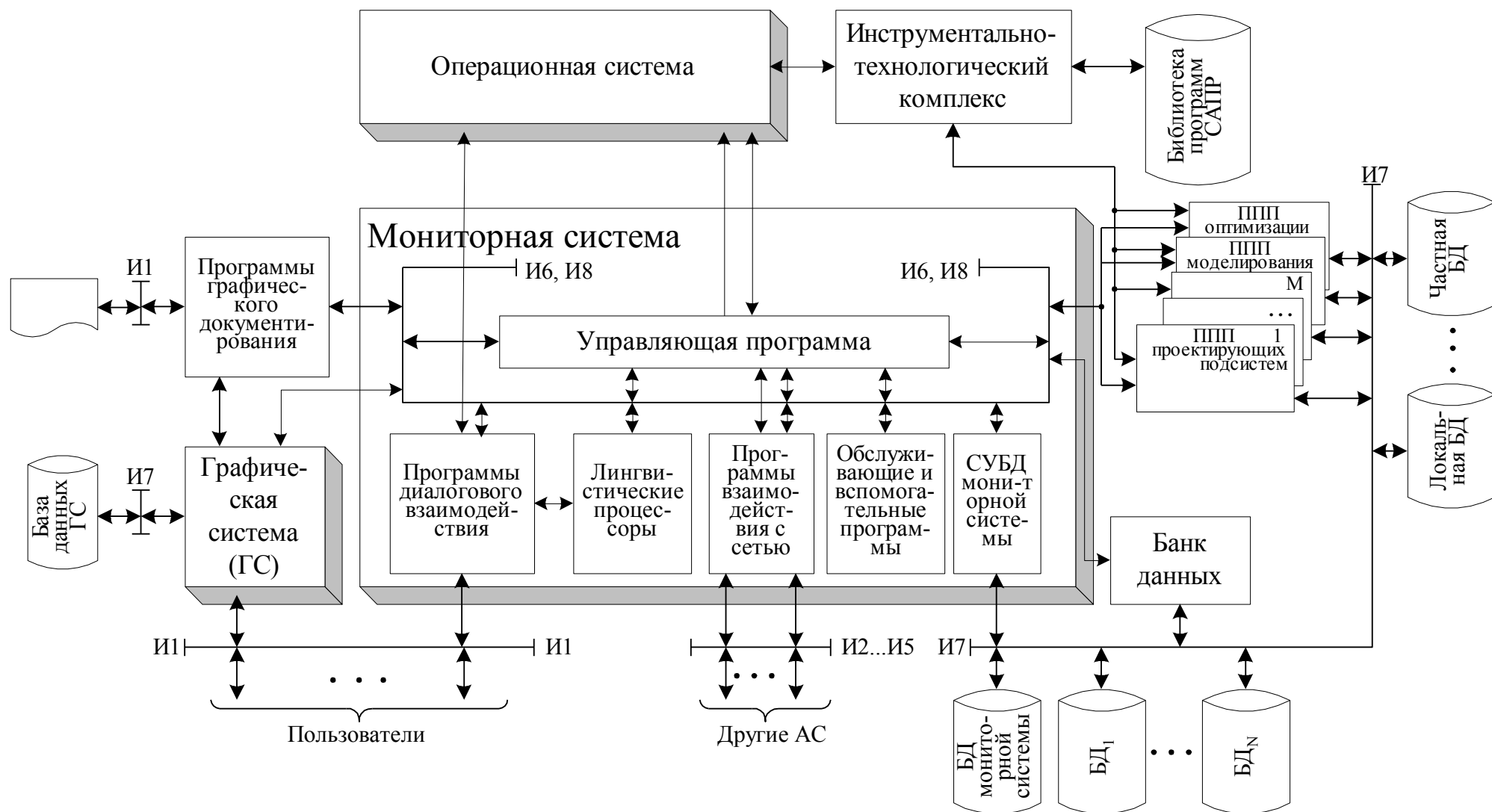


Рис. 3.4. Функциональная структура ПО САПР

- * мониторная система (МС);
- * графическая система (ГС) (графический и геометрический процессоры);
- * СУБД, поддерживающие базы данных (БД) нормативно–справочной информации (НСИ) и архива проектных решений, а также, возможно, СУБД некоторых ППП, реализующих проектирующие подсистемы;
- * телемонитор и (или) ПО сетей ЭВМ;
- * программы формирования графической документации;
- * инструментальный технологический комплекс для разработки программного обеспечения.

Кроме того, к числу общесистемных компонентов ПО САПР часто относят программные средства моделирования и оптимизации, общетехнических расчетов (библиотеки подпрограмм для научно–технических расчетов).

На рис.3.4 введены следующие обозначения:

- И1 – интерфейс связи с проектировщиками и обслуживающим персоналом САПР;
- И2 – интерфейс связи с автоматизированной системой научных исследований (АСНИ);
- И3 – интерфейс связи с гибким автоматизированным производством (ГАП);
- И4 – интерфейс связи с автоматизированным экспериментальным комплексом (АЭК);
- И5 – интерфейс связи с автоматизированной системой управления предприятием (АСУП);
- И6 – интерфейсы связи с проектирующими подсистемами;
- И7 – интерфейсы связи с универсальными и специализированными СУБД;
- И8 – интерфейсы по управлению.

Для ППП, решающих задачи проектирования, которые требуют высокой реактивности, могут применяться универсальные СУБД.

Функции взаимодействия с пользователями и управление реализуются в основном компонентами общесистемного ПО САПР, поэтому подробнее остановимся на них.

Функции взаимодействия с пользователем реализуют такие компоненты, как

- * графическая система;
- * программы диалогового взаимодействия и лингвистический процессор, обеспечивающие алфавитно–цифровой диалог и интерпретацию проблемно–ориентированного языка проектирования;
- * программы документирования проектных решений.

Таким образом, *интерфейс* И1 включает ряд разнородных интерфейсов, реализуемых диалоговыми средствами различных ПМК.

Функции взаимодействия САПР с другими автоматизированными системами, а также с другими вычислительными системами реализуют сетевые программы МС. В других вариантах реализации ПО САПР эту функцию могут выполнять либо ОС, либо специальный телемонитор (сетевое ПО), имеющий форму ППП, расширяющего функции операционной системы.

Обслуживающие и вспомогательные программы МС обеспечивают сбор и обработку измерительной информации с целью учета и планирования, выдачу диагностики, сопровождение информационного (ИО) и программного (ПО) обеспечения САПР и другие возможности.

Функцию управления процессом автоматизированного проектирования (АП) реализуют

мониторная система, частично графическая система и управляющие программы ППП. Возможны варианты построения ПО САПР, в которых часть функций взаимодействия с пользователем и управления разделяется между ОС и ППП.

В перспективном случае так называемых однородных (комплексных) САПР все функции управления и взаимодействия с пользователем и другими автоматизированными системами реализует единая МС. В зависимости от категории пользователя взаимодействие с системой осуществляется на одном проблемно– или процедурно–ориентированных языков проектирования, который интерпретируется МС.

При создании конкретной САПР перечисленные компоненты должны рассматриваться как типовые проектные решения и приобретаться на договорных условиях как продукция производственно–технического назначения. Однако при создании конкретной САПР не всегда можно выбрать подходящий компонент каждого типа, поэтому приходится разрабатывать оригинальные мониторную систему, программы документирования, специализированную СУБД. Наибольшая свобода выбора компонентов имеется для ОС, СУБД и телемониторов, наименьшая – для графических систем, инструментально–технологических комплексов, средств моделирования и оптимизации.

Компоненты общесистемного ПО САПР являются промышленно тиражируемыми изделиями и представляют собой ПМК. **Программно–методический комплекс** – взаимосвязанная совокупность компонентов программного, информационного, методического, математического и лингвистического обеспечений, необходимая для выполнения законченной совокупности проектных или обслуживающих операций.

Требования, предъявляемые к общесистемным ПМК САПР, сформулированы в ГОСТ 23501.201–83. Общесистемные ПМК предназначены для обеспечения работоспособности САПР на системном уровне и выполнения унифицированных обслуживающих процедур. В сочетании с ОС общесистемные ПМК являются операционной средой, в которой функционируют прикладные программы. Общесистемные ПМК должны быть инвариантны к объектам проектирования и защищены от пользователей САПР. Функционирование общесистемных ПМК должно обеспечиваться специальными подразделениями (специалистами) в составе службы САПР.

3.5.1. Функции основных компонентов общесистемного программного обеспечения САПР

Операционная система является основой общесистемного ПО САПР и обеспечивает автоматизированное функционирование комплекса технических средств (КТС) САПР и выполнение им компонентов ПО, а также, выполняя функции интерфейса между ПО и ТО, обеспечивает эффективное использование разнообразных функциональных возможностей отдельных компонентов ТО.

Мониторная система (подсистема управления) – обслуживающая подсистема САПР, предназначенная для организации и оптимизации управления процессом проектирования при выполнении проектных процедур и взаимодействии подсистем САПР. Мониторная система в общем случае включает компоненты программного, информационного, методического и

лингвистического обеспечения САПР. Требования к МС определены в ГОСТ 23501.13–81.

Мониторная система включает следующие компоненты (рис. 3.3):

- * управляющую программу;
- * программы диалогового взаимодействия с пользователем;
- * лингвистические процессоры проблемно–ориентированных языков (метаязыков) проектирования и управления процессом проектирования, а также языка описания функций МС (распределение ресурсов САПР, формирование перечня плановых заданий на проектирование и др.);

- * программы взаимодействия с сетью;
- * обслуживающие и вспомогательные программы;
- * СУБД мониторинг системы;
- * базу данных МС.

*Мониторная система выполняет следующие **основные функции**:*

- * управления процессом реализации проектных процедур и операций;
- * организации взаимодействия подсистем САПР;
- * интерпретации языковых форм заданий на выполнение проектных процедур и операций;
- * распределения ресурсов САПР в процессе проектирования с учетом заранее заданных приоритетов подсистем САПР и проектных процедур по запросам пользователей;
- * защиты ресурсов системы и баз данных САПР от несанкционированного доступа;
- * обеспечения диалоговых и интерактивных режимов работы при проектировании в условиях параллельной работы нескольких подсистем САПР;
- * сбора статистики для целей учета и планирования использования ресурсов различными пользователями и подсистемами САПР;
- * контроля и восстановления процесса функционирования программ;
- * выдачи диагностической и справочной информации о состоянии подсистем САПР.

Мониторная система не должна дублировать функций операционной системы, а будучи специализированным дополнением ОС, должна обеспечивать максимальную эффективность функционирования ПТК САПР за счет генерации и настройки программ.

3.6. Жизненный цикл программных комплексов

Значительное внимание в публикациях уделяется моделям жизненного цикла информационных систем обработки больших потоков данных в сфере организации и управления производством. Для программных комплексов этого класса характерны использование типовых систем управления базами данных, активный интерфейс со многими пользователями, относительно небольшой объем оригинальных программных разработок и широкое применение повторно используемых компонент. В жизненном цикле программных комплексов этого класса преобладают затраты на проектирование и могут значительно сокращаться затраты на разработку оригинальных программных компонент, их комплексирование и отладку в системе.

В исследованиях жизненного цикла слабо отражены разработки пакетов прикладных программ для инженерных расчетов, моделирования, научных экспериментов и т.п.,

осуществляемые индивидуально или малыми группами специалистов. Тем не менее, подобные проекты имеют значительные особенности жизненного цикла, которые следует учитывать при планировании и осуществлении разработки. Они допускают значительные упрощения моделей жизненного цикла и планов, создаваемых на их базе. Регламентирование таких планов на базе моделей жизненного цикла позволяет ускорять разработки и повышать их качество.

Модели жизненного цикла программных комплексов несколько отличаются терминологией и графическим представлением этапов их взаимодействия. Почти во всех моделях отражен ряд базовых этапов, к которым присоединяются или из которых выделяются другие этапы, характеризующиеся менее четкими целями. Такими *базовыми этапами–процессами* являются (рис. 3.5):

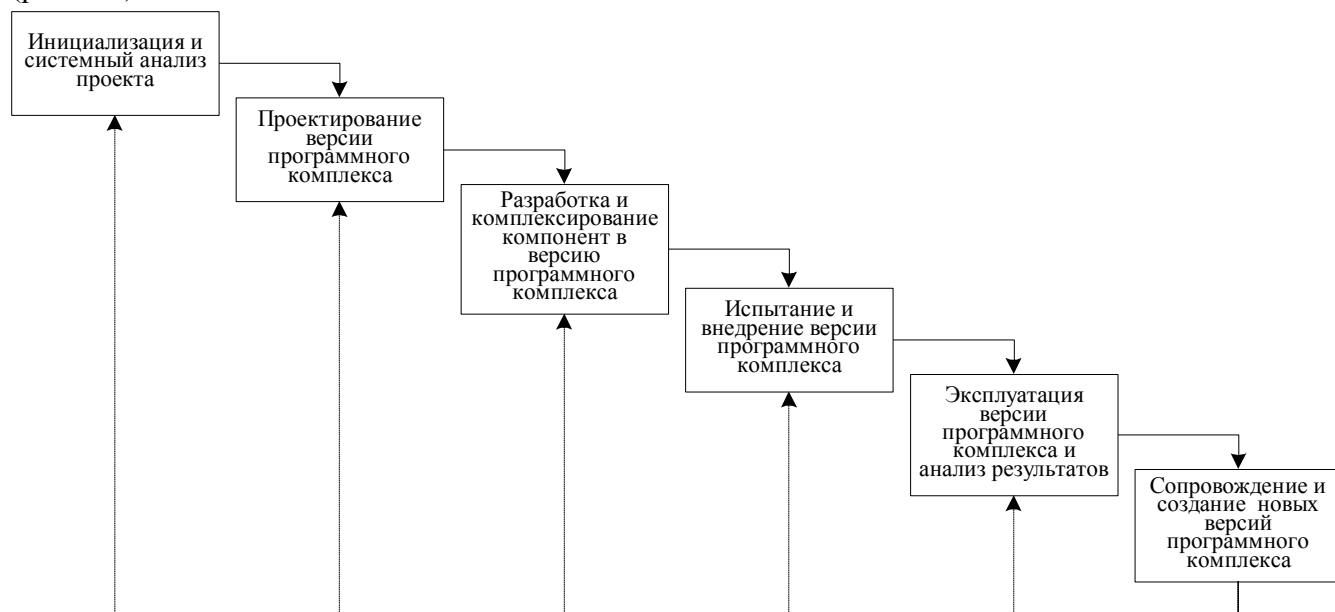


Рис. 3.5

- * инициализация, системный анализ и первоначальная разработка спецификации требований на программный комплекс;
- * предварительное (эскизное) и детальное (техническое) проектирование программного комплекса;
- * разработка программных компонент, их комплексирование и отладка программного комплекса в целом;
- * испытания, опытная эксплуатация и распространение программного комплекса;
- * регулярная эксплуатация ПС, ее поддержка и анализ результатов;
- * сопровождение программного комплекса, его модификация и совершенствование, создание новых версий.

При применении детальных моделей жизненного цикла для конкретных проектов программного комплекса их необходимо *адаптировать* в соответствии с характеристиками объектов, среды разработки и использования программного комплекса. Адаптация обычно сводится к дополнению или исключению некоторых работ, а также к корректировке их характеристик. Детализация и адаптация позволяют отработать рабочие планы создания и поддержки применения программного комплекса определенного назначения в течение всего жизненного цикла. Кроме того, подобные планы являются методической базой для выбора и

применения современных CASE (Computer–Aided Software Engineering) – средств автоматизации всего технологического процесса и отдельных работ жизненного цикла программного комплекса.

3.7. Способы описания структур и функций ПО

На каждом из иерархических уровней разработки ПО имеются свои способы представления проектных решений. Если после этапа кодировки получаются полные тексты программ на принятых языках программирования, то на предыдущих этапах необходимо иметь средства более лаконичного и укрупненного представления структур данных, вычислительных процессов и описания спецификаций. Такими средствами являются граф–схемы, диаграммы HIPO, функциональные схемы и псевдоязыки.

Граф–схема представляет собой граф, вершины которого изображают блоки обработки информации, а дуги (ребра) – связи по информации или по управлению между блоками. Блокам могут соответствовать различные по объему части ПО (от отдельного оператора алгоритмического языка до крупной программы), поэтому граф–схемы можно использовать на любом иерархическом уровне проектирования ПО, в частности для изображения различных маршрутов автоматизированного проектирования.

Применяется несколько разновидностей граф–схем. Во–первых, можно поменять местами роли вершин и дуг, поставив в соответствие первым структуры данных, а вторым – операции обработки. Во–вторых, можно ввести вершины двух типов – для отображения как операций обработки, так и структур данных. В–третьих, возможны граф–схемы, в которых выделяют дуги для отображения как связей по информации, так и связей по управлению.

Диаграммы HIPO (аббревиатура из начальных букв английских слов "иерархия – вход – обработка – выход") служат для представления спецификаций модулей в виде перечисления выполняемых функций и описания данных, являющихся входными и выходными для модуля.

Функциональные схемы используют для представления структур программных комплексов в виде последовательности блоков обработки, приемников и источников информации. Функциональные схемы используют на верхних иерархических уровнях проектирования ПО.

Разработка любой сложной системы должна начинаться с функционального анализа и моделирования системы в целом и всех ее подсистем. Для этой цели разработана методология IDEF0, представляющая собой совокупность методов, правил и процедур, предназначенных для построения функциональной структуры сложных иерархических систем. Эта методология может использоваться как для определения требований к функции на начальных этапах проектирования САПР, так и при разработке рабочих проектов систем, специфицированных с помощью IDEF0.

Граф–схемы и функциональные схемы наглядны, но мало удобны в процессе постепенной детализации описаний сложных программных комплексов, особенно при попытках формализации этого процесса. Поэтому наиболее гибким и универсальным средством описания ПО в процессе его проектирования являются псевдоязыки, называемые также псевдокодами или языками спецификаций.

Псевдоязык представляет собой объединение естественного языка с одним из языков программирования. Обычно средства языка программирования используются для описания структур программ, а средства естественного языка – для укрупненного описания процедур

обработки данных. Структуры программ описываются с помощью операторов перехода, цикла, варианта, обращения к подпрограммам.

Псевдоязыки – удобное средство не только представления промежуточных результатов разработки ПО, но и описания алгоритмов автоматизированного проектирования при их публикациях.

3.8. Методы разработки программного обеспечения

Современная технология создания ПО САПР – совокупность эффективных средств и методов проектирования, позволяющих упростить данный процесс, уменьшить стоимостные затраты, сократить календарные сроки проектирования системы и, в конечном итоге, за счет возможности более широкого выбора проверенных прогрессивных проектных решений, повысить качество разработки.

Основные средства проектирования:

- * *стандартные средства операционных систем и универсальных языков программирования*, обеспечивающих автоматическое прохождение на ЭВМ определенного класса задач;

- * *процедуры, реализующие типовые процессы обработки данных*, например контроль выходной информации и ее сортировку;

- * *инструментальные средства*, к которым относится совокупность взаимосвязанных специальных программных средств, предназначенных для инструментальной поддержки отдельных элементов процесса проектирования САПР. Это создание и актуализация словаря данных, документирование проекта, автоматизация контроля проектирования и др.;

- * *типовые компоненты*, представленные в виде типовых проектных решений (ТПР) и пакетов прикладных программ (ППП). *ТПР* – совокупность алгоритмических, программных, инструктивно–методических элементов, обеспечивающих машинную реализацию задач или комплекса с помощью соответствующих технических средств. ТПР – основа создания ППП, к которым относятся комплексы программ, обеспечивающих работу типовых конфигураций вычислительной техники, диалоговых систем при решении типовых функциональных задач;

- * *инструментальные системы автоматизированного проектирования (ИСАПР)*, предполагающие использование ЭВМ на всех этапах создания САПР и занимающие высшую ступень в эволюции средств проектирования системы.

В методах проектирования различают *классы* и *подклассы*.

Классы проектирования:

1) *оригинальное проектирование*. Средства, используемые при этом методе:

- * стандартные средства операционных систем;
- * процедуры, реализующие типовые процессы обработки данных.

2) *типовое проектирование*. Подклассы: элементы, подсистемы, объектное, групповое.

Средства:

- * стандартные средства операционных систем;
- * типовые компоненты (ТПР, ППП);
- * некоторые инструментальные средства.

3) *автоматизированное проектирование*. Подклассы: модульное; др. Средства:

- * стандартные средства операционных систем;
- * ИСАПР;
- * взаимосвязанный комплекс инструментальных средств.

Средства проектирования подразделяются на:

- * *комплексные* – это ТПР, ППП, типовые проекты автоматизированных систем, САПР.
- * *локальные* – большое разнообразие, в их состав входят системы управления базами данных, телеобработки, инструментальные средства и др.

Общие требования к средствам проектирования:

- * полный охват всего процесса создания ПО САПР;
- * совместимость, требующая согласованных решений как в процессе создания системы и ее обеспечивающих подсистем, так и в процессе их функционирования;
- * универсальность в своем классе, допускающем возможность применения одних и тех же средств для различных объектов;
- * легко доступными, не требующими особых усилий в освоении и просты в реализации;
- * возможность организации процесса проектирования в режиме интерактивного взаимодействия разработчика системы, проектировщика и ЭВМ;
- * адаптированными и экономически эффективными.

Методы оригинального проектирования являются традиционными и ориентированы на одно предприятие. Характерная черта – разработка оригинальных методик обследования объекта, его внедрения, создания необходимой проектной документации в виде индивидуального проекта. Достоинство – отражение в проекте ПО САПР специфических особенностей объекта автоматизации. Недостатки: сравнительно высокая трудоемкость и большие сроки разработки, низкий показатель функциональной надежности и адаптируемости к изменяющимся условиям. Проекты, созданные оригинальным методом, поддаются модернизации, однако в чистом виде этот метод используется редко. При его реализации используются в настоящее время различные средства проектирования и лишь для отдельных частей проекта требуются оригинальные проектные решения. Так, общесистемные проектные решения по разработке информационного обеспечения включают методы сбора, контроля и передачи данных, создание нормативно-справочных массивов информации, определяют версию операционной системы, типовые процедуры обработки информации и т.д. Это несколько сглаживает его недостатки. Этот метод особенно актуален при автоматизации проектирования сложных, неординарных объектов.

Типовое проектирование – индустриальный метод создания ПО САПР, использующий ТПР и ППП, характеризуется наличием апробированных, типовых организационно-экономических, технических, информационных, математических и программных средств автоматизации проектирования. Достоинства: уменьшает трудоемкость, снижает стоимость и сокращает сроки проектирования, повышая его качество путем более полного охвата задач функциональных подсистем, строгого соблюдения требований нормативных документов, применения передовых технических решений. Типовое проектирование призвано устранить

дублирование проектов, создать основу для расширения обмена готовыми типовыми компонентами, облегчить разработку рекомендаций по изменению организационной структуры и методов управления с учетом отраслевых и внутрихозяйственных особенностей. Процесс типового проектирования заключается в выборе и привязке указанных средств в соответствии с требованиями конкретной системы. Типовая часть САПР представляет собой комплекс информационного, программного и технического обеспечения. Типовой характер первого достигается путем строгого соблюдения единства структуры информационной базы, состава массивов, форм входных и выходных документов; второго – на использовании ППП, и последнего в результате применения ЭВМ одного или совместных типов.

ТПР создаются по модульному принципу, когда каждое проектное решение расчленяется на отдельные составные части – модули, которые реализуют определенную часть ТПР. Это позволяет создать проект новой автоматизированной системы путем сочетания отдельных типовых модулей.

При использовании *подсистемного метода* проектирования предполагается более высокая степень интеграции типовых элементов системы, когда для каждой подсистемы создаются проекты решений и пакеты прикладных программ. Выделение подсистем осуществляется в зависимости от объекта автоматизированного проектирования. Для каждой из подсистем разрабатывается свое автоматизированное проектное решение и ППП, которые могут быть общесистемного или функционального назначения. К первой группе относятся ППП управления данными, типовых процедур их обработки, методов математической статистики и дискретного программирования, решения непрерывных задач, например дифференциальных уравнений. Во вторую группу входят пакеты, ориентированные на промышленные предприятия с дискретным или непрерывным характером производства, на непромышленную сферу, отраслевое управление.

Важное требование, предъявляемое к ППП, – совместимость, т.к. при проектировании ПО САПР целесообразно использовать сразу несколько пакетов. Проектирование систем с применением ППП фактически сводится к привязке выбранных по определенным параметрам пакетов к конкретным условиям объекта автоматизации. Достоинства: менее трудоемкий процесс, занимает меньше времени по сравнению с оригинальным проектированием, реализует прогрессивные методы обработки данных, упрощает документирование проекта, т.к. используется документация пакетов, повышается надежность проектируемых систем.

Метод *объектного проектирования* базируется на применении типовых проектов САПР. Применяется недостаточно широко, т.к. слишком много разнообразных объектов, а модификация типового проекта системы в соответствии с конкретными условиями объекта автоматизации требует больших трудовых и материальных затрат.

Среди автоматизированных методов особое место занимают *методы модульного проектирования*. Создание и использование ИСАПР обеспечивает достаточно высокий уровень функциональной надежности, комплексный охват всех технологических процессов, снижение трудоемкости проектных работ с максимальным учетом интересов объекта автоматизации. Однако этот метод достаточно дорог и требует высококвалифицированных разработчиков. Ключевое требование, предъявляемое к ИСАПР, – возможность построения и поддержания в

системе проектирования в адекватном состоянии некоторой глобальной экономической информационной модели объекта автоматизации. *Модель* – отображение информационных компонентов объекта автоматизации и отношение между ними, заданные в явном виде. Основная цель построения модели – создание соответствующего этой модели проекта ПО САПР, учитывающего и активно использующего все характеристики объекта. Такая модель должна содержать в формализованном виде описание совокупностей программных компонентов и отношения между ними, включая информационные связи и алгоритмическое взаимодействие. С помощью модульного метода проектирования применяется системный подход, обуславливающий использование ЭВМ не только на всех стадиях создания системы, но и в процессе анализа результатов ее промышленной эксплуатации. Развитие и применение САПР предопределило переход к созданию индивидуальных проектов, но на значительно более высоком уровне, по сравнению с оригинальным методом проектирования.

Создание специального программного обеспечения (СПО) является одним из главных этапов построения САПР. Программные средства САПР образуются из различных видов программной продукции, таких, как программа, программный модуль, пакет прикладных программ (ППП), программная система (ПС).

Под *программой* понимают последовательность команд, написанных на каком-либо языке, которые должны быть выполнены ЭВМ для реализации некоторой прикладной задачи или функции.

Программный модуль описывает некоторую элементарную функцию и обычно используется для конструирования ППП.

Пакет прикладных программ или просто пакет программ представляет совокупность программ и модулей, снабженных системными и языковыми средствами и используемых главным образом как инструмент для создания на его основе программных систем.

Программной системой принято называть совокупность взаимосвязанных системных и прикладных программ, программных модулей и ППП, готовую к непосредственному выполнению и обеспечивающую режим крупных технико-организационных функций, например, подсистемы САПР.

В условиях увеличения затрат на создание сложных программных средств и повышения трудоемкости этих работ разработчику следует рассматривать создание СПО как комплекс научно-исследовательских и опытно-конструкторских работ, основной частью которого является этап программирования.

При создании СПО современных САПР большое внимание уделяется *технологии программирования*.

Наиболее широкое распространение при создании СПО получили *модульное и структурное программирование*.

Идея *модульного программирования* состоит в разделении сложной программы на простые объекты – программные модули и последующем синтезе программных средств из необходимых модулей. *Модуль* рассматривается как отдельная функционально законченная программная единица, которая структурно оформляется специальным образом, чтобы облегчить

объединение с другими модулями в рамках того же языка программирования или в многоязыковой системе.

Модульное программирование, возникшее в начале 60-х годов, характеризуется следующими *преимуществами*:

- * большую программу могут разрабатывать одновременно несколько исполнителей, причем каждый исполнитель может использовать свой язык программирования;
- * можно создавать и использовать библиотеки наиболее употребительных модулей;
- * становится проще проблема сегментации большой программы и процедуры загрузки в память;
- * облегчается тестирование программы;
- * проще проектирование и последующие изменения программы, т.к. изменение и совершенствование отдельных модулей не влияют на работоспособность ППП.

В модульном программировании наиболее нечеткие задачи – это разбиение на модули и определение их сопряжений. Задача разбиения на модули сложна уже тем, что в больших программах она имеет неоднозначное решение. В принципе модульное разбиение должно удовлетворять концепции "один модуль – одна функция", т. е. **модуль** – это элемент программы, выполняющий самостоятельную задачу.

Другая проблема модульного программирования – сопряжение программных модулей для быстрой сборки программы из необходимых модулей. Сопряжения модулей, написанных на одном языке, обычно не вызывает трудностей. Во всех языках программирования обычно имеются средства для автоматического объединения модулей на уровне исходного представления (в языковой форме).

Разработка больших программ, состоящих из модулей, написанных на разных языках программирования, представляет достаточно трудоемкий процесс. Их механическое объединение в большинстве случаев невозможно из-за существенных различий в языковых средствах и особенностей компиляторов. Для объединения модулей, записанных на разных языках программирования, используются специальные модули либо возможности операционных систем.

Первые работы в области **структурного программирования** опубликованы Дейкстрой в 1968г. Технология структурного программирования представляет собой форму структурирования программ и данных с целью создания ясных для понимания программ.

Основное требование к ПО – выбор такой структуры программ и способов их реализации, которые способствовали бы уменьшению затрат времени и средств на разработку и сопровождение программ. При этом под *сопровождением программы* понимается любая деятельность, направленная на исправление недостатков и улучшение программы в процессе ее эксплуатации. Улучшению организации разработки ПО САПР способствует применение структурного программирования.

Структурное программирование – это технология программирования, в которой используется совокупность определенных принципов, обеспечивающих:

- 1) повышение производительности труда программистов при написании и контроле программ;
- 2) получение конкретных программ с ясной и легко понимаемой структурой для удобства

их сопровождения.

Структурное программирование включает последовательное применение и конкретизацию блочно–иерархического подхода к проектированию программных систем. К *основным принципам структурного программирования* относятся: модульность. структуры; иерархия модулей; нисходящее проектирование.

Модульность структуры. Разбиение программы на модули в САПР целесообразно осуществлять по функциональному признаку. В этом случае реализация маршрутов проектирования выливается в комбинирование имеющихся модулей, облегчается построение межмодульного интерфейса, смена некоторого метода алгоритма или модели сводится к замене модуля, т. е. легче реализуется открытость САПР в отношении математического обеспечения. Модули должны оформляться в таком виде, чтобы каждый из них имел только один вход и один выход, а возврат из модуля должен происходить только в вызвавший его модуль. При большой сложности модуля, реализующего некоторую функцию, целесообразно производить его иерархическое разбиение на более мелкие модули.

Иерархия модулей. Разделение модулей на иерархические уровни в структурном программировании производится по принципу вложенности. При этом используется вертикальное управление, для которого характерно то, что обращение к любому модулю может происходить только из какого–либо модуля более высокого уровня. Следовательно, на верхнем уровне должен быть единственный модуль (ведущая программа), управление которым происходит из операционной системы. Взаимодействие равноуровневых программ при вертикальном управлении происходит только через программу более высокого уровня.

Нисходящее проектирование программ заключается в том, что планирование, реализация и контроль программной системы ведутся сверху вниз, т. е. сначала производится проектирование модулей высших, а затем низших уровней.

Модули высших уровней могут быть выражены на принятом языке программирования и предварительно отлажены до разработки модулей низших уровней. При этом в процессе отладки отсутствующие модули заменяются макетами (заглушками). Окончательная проверка ПО САПР производится с реальными модулями после завершения их разработки с помощью специально подбираемых тестовых задач.

Структурное программирование наряду с *иерархическим подходом* к программированию допускает и *операционный подход*, при котором модули разрабатываются в порядке их выполнения в маршрутах проектирования.

В структурном программировании также используется разбиения на модули, но приняты формализованные принципы разбиения и написания программ в соответствии с жесткими правилами.

Логическая структура программы может быть выражена как комбинация трех основных структур – следование, разветвление, цикл. Данные структуры являются элементарными блоками, из которых можно построить логику любой программы. Эти структуры считаются допустимыми, и любой алгоритм на любом уровне должен быть представлен только с помощью допустимых структур. Все допустимые структуры имеют один вход и один выход, причем допускаются вложения структур друг в друга и рекурсивность.

Для построения структурных программ из допустимых структур запрещается применение оператора безусловного перехода (типа "*GO TO*"), поэтому структурное программирование часто называют "программированием без *GO TO*".

Структурные программы в отличие от обычных имеют простую древовидную структуру без переходов вперед–назад, без "заплаток"; они легко читаются и модифицируются. Указанные свойства отчасти достигаются тем, что на этапе проектирования программ создаются определенные ограничения; для конструирования программ можно использовать лишь допустимые структуры данного языка.

Объектно–ориентированное программирование. Рассмотрим одну из наиболее интересных технологий программирования, которая получила конкретную практическую реализацию совсем недавно, но тем не менее ознаменовала совершенно новый подход в технологии программирования. Речь пойдет об *объектно–ориентированном программировании* (ООП) как о новой методологии, основанной на более высоком уровне абстрагирования и модульности. Если совершить небольшой экскурс в историю развития ООП, то можно отметить двух явных первопроходцев в этом направлении – объектно–ориентированные языки SmallTalk и Си++. Благодаря им идеи ООП получили конкретную реализацию, а языки программирования обогатились новыми синтаксическими конструкциями. И хотя рано говорить о создании определенного языкового стандарта для ООП, все–таки все последующие системы исходят из уже имеющегося опыта. Последние материалы периодических изданий в области программирования позволяют сделать вывод, что идет массированное наступление новой методологии почти на всех языковых направлениях.

Понятие "объектно–ориентированный" возникло в программировании сравнительно недавно. Когда вычислительная мощность машин была невысока, о создании объектно–ориентированных систем не могло быть и речи. Основой всего был программный код. Программисты записывали последовательности команд для выполнения тех или иных действий над данными, которые оформлялись в модули и процедуры. Для работы с каждым объектом создавалась своя процедура.

Постепенно с увеличением производительности вычислительных систем процедурный подход начал заменяться объектным. На первое место выдвинулся объект, а не код, который его обрабатывает. На уровне пользователя объектный подход выражается в том, что интерфейс представляет собой подобие реального мира, а работа с машиной сводится к действиям с привычными объектами. Так, папки можно открыть, убрать в портфель, документы – просмотреть, исправить, переложить с одного места на другое, выбросить в корзину, факс или письмо – отправить адресату и т. д. Понятие объекта оказалось настолько широким, что до сих пор не получило строгого определения.

Объектом может быть и константа, и переменная, и процедура, и процесс. Объекты в программах "рождаются" и "умирают", меняют свое состояние, запускают и останавливают процессы, "убивают" и "возрождают" другие объекты, т. е. воспроизводят все оттенки явлений реального мира. Под *объектом* можно подразумевать некоторое абстрактное понятие, (например, "уравнение" или "график функции"), имитирующее реальную систему или процесс (например,

"теплообменник", "станок", "автомобиль"). В этом плане *объект* – это сущность процесса или явления, которую способны выделить наш опыт, знания и интуиция.

Объект содержит данные различных типов. Но в отличие, например, от записи он может еще содержать методы работы над объектом – процедуры либо функции языка программирования (например, Паскаля), которые оперируют данными объекта. Если провести аналогию с физическим объектом, то очевидно, что запись по своему определению не идет дальше описания его конкретных физических свойств. В то же самое время любой физический объект, как правило, характеризуется своим поведением в окружающем мире. Поведение, заданное процедурами и функциями объекта, и свойства, определяемые данными различных типов, формируют конечную абстракцию объекта. Такое представление больше соответствует обычной человеческой логике и заставляет отбросить стандартные способы мышления, на которых основано программирование.

В любом программном объекте могут развиваться динамические процессы, определяющие изменение состояния объекта во времени. Такие процессы могут развиваться автономно (независимо один от другого) или во взаимодействии друг с другом. Концепция взаимодействия основывается на одновременном развитии нескольких процессов, при этом такая одновременность трактуется в программировании как логический параллелизм – одновременное выполнение нескольких действий (активностей), обусловленное логикой развития моделируемой системы. Реализация концепции логического параллелизма требует в общем случае наличия нескольких процессоров, что связано с использованием нового класса вычислительных систем – систем с мультипроцессорной архитектурой, – либо с имитацией логического параллелизма на обычной однопроцессорной ЭВМ.

Инкапсуляция – одна из специфических особенностей программирования, ориентированного на объекты. Эта особенность предполагает не только возможности "разложения целого на части" (принципа, определяющего основы любого программирования), но и умения "скрывать" частности от общего (целого). Такой подход позволяет программисту не знать частных деталей реализации программной системы, осуществлять конструирование из элементов, реализация которых скрыта от него "под оболочкой" модуля. Модуль в этом подходе приобретает роль основного конструктивного элемента, используемого для синтеза и разработки новых систем.

Специфические особенности модуля заключаются в следующем:

- 1) модуль – это автономно компилируемая программная единица;
- 2) информационные и управляющие связи между модулями требуют использования в его описании деклараций, которые в совокупности определяют оболочку модуля, регламентирующую такие связи;
- 3) сборка программной системы из модулей связана с отдельным технологическим этапом – компоновкой программы. Правила такой компоновки полностью определяются системой модульных оболочек.

Концепция оболочки реализуется декларациями импорта/экспорта, регламентирующими, какие объекты, определенные внутри модуля, можно использовать "за его пределами".

3.9. Показатели качества прикладных программ

Качество изделий, проектирования, производства и услуг является одной из узловых проблем в различных областях экономического и социального развития общества, определяющей уровень жизни человека и состояние народного хозяйства. Это полностью относится и к области программного обеспечения САПР. Возрастание роли программных средств в народном хозяйстве, широты их применения и ответственности решаемых задач способствует резкому повышению требований к качеству программ. Это обусловило появление, развитие и активное применение методов и средств, обеспечивающих создание программных средств с заданными высокими показателями качества при ограничениях, установленных на использование ресурсов разработки программных средств.

Для достижения высокого качества изделий применяются в основном *два принципиально различающихся метода*. При *первом методе* основное внимание сосредоточено на испытаниях и отбраковке изделий на конечном этапе разработки или производства. При этом значительная доля изделий может идти в брак, что нерентабельно, особенно для сложных и дорогих изделий. *Второй метод* акцентирует внимание и усилия на обеспечении высокого качества всего технологического процесса разработки или производства, гарантирующего необходимое качество конечного продукта. Контроль качества и испытания изделия на ряде промежуточных этапов его создания в значительной степени предотвращают возможность обнаружения брака при завершающих испытаниях и передаче его в эксплуатацию. Для разработки сложных программных средств этот метод является основным, гарантирующим их высокое качество.

При ограниченных ресурсах на разработку программных средств для достижения заданных требований необходимо планирование и управление обеспечением качества в течение всего цикла создания программ. Управление обеспечением качества программ предполагает формализацию технологии их разработки, а также выделение в специальный процесс поэтапное измерение и анализ текущего качества программных компонент.

В общем случае *в процесс управления обеспечением качества программных средств* входят:

- * анализ системных требований к программным средствам, выделение и ранжирование обобщенных показателей качества конечного продукта;
- * декомпозиция обобщенных показателей качества по контролируемым этапам и объектам разработки и создание разделов по качеству в спецификациях требований на программные компоненты;
- * выбор или создание методов, технологии и средств автоматизации разработки, обеспечивающих создание программных средств с заданными показателями качества;
- * создание методов и средств объективного измерения достигнутого качества программных компонент и средств в целом на фиксированных этапах их создания;
- * разработка методик и стандартов контроля за соблюдением правил и технологии проектирования и обеспечением всего жизненного цикла программных средств;
- * организация, обучение и стимулирование коллективов специалистов на создание компонент и средств в целом, в максимальной степени удовлетворяющих требования заказчиков и пользователей.

Качество сложных программных средств описывается совокупностью показателей–критериев, для каждого из которых должны быть определены метрики и методы их измерения. Критерии необходимо классифицировать в зависимости от классов программ, а также связать их с методами и этапами разработки. Адекватный набор показателей качества программных средств зависит от функционального назначения и свойств каждого программного средства. В соответствии с принципиальными особенностями программных средств выбираются номенклатура и значения показателей качества, которые отражаются в техническом задании и в спецификации требований на конечный продукт.

За ограниченный, относительно короткий период приемо–сдаточных испытаний трудно провести достаточно обширное тестирование, достоверно демонстрирующее достигнутые показатели качества, и гарантировать выполнение всех технических требований к сложному программному комплексу. Поэтому для обеспечения гарантий высокого качества программ проводятся испытания не только завершеного программного комплекса, но на ряде промежуточных этапов разработки проверяются состояние и характеристики компонент проекта.

В результате появляется необходимость выделения *показателей качества процесса разработки программ*. Качество этого процесса в значительной степени характеризуется совокупными затратами, необходимыми для достижения заданного качества программного комплекса. В результате достижимое качество программ определяется качеством методов и средств, применяемых при их разработке, а также затратами на их использование.

В реальных условиях при ограниченных ресурсах на проектирование программных средств критерием завершения разработки может быть *ограничение допустимых затрат*. Сложность программ и затраты становятся косвенными критериями или факторами, влияющими на качество программ.

В стандартах описываются до 20–30 показателей – *основных критериев качества* и приводятся определения для их вычисления. Однако многие показатели имеют иллюстративный характер, и их значения определяются экспертно.

В зависимости от класса и особенностей программных средств целесообразно выбирать различные наборы критериев, адекватные свойствам конкретного программного средства. Эти критерии должны наиболее полно отражать назначение и функциональные характеристики программного средства при его применении. При выборе критериев целесообразно учитывать возможность объективного и достоверного определения значений показателей. Поэтому ниже представлено только небольшое число критериев (рис.3.6), которые наиболее часто применяются и могут достаточно объективно измеряться.

Среди показателей качества можно выделить две крупные группы и соответствующие им наборы критериев:

* *функциональные критерии* отражают специфику областей применения и степень соответствия программ их основному целевому назначению;

* *конструктивные критерии* более инвариантны к целевому назначению программ и отражают эффективность использования программами ресурсов вычислительных средств, а также надежность функционирования программных средств.

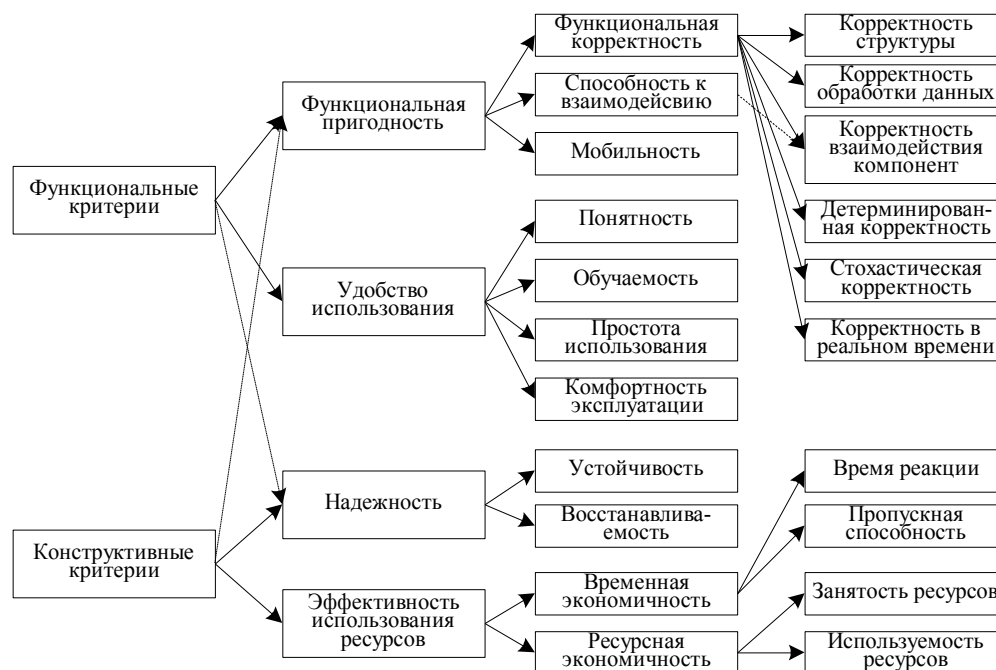


Рис.3.6

Деление критериев на эти две группы в значительной степени условно, что отражено на рис.3.6 наличием связей, обозначенных штриховыми линиями. На рис. 3.6 выделены четыре основные группы критериев, в составе каждой из которых детализируются еще несколько показателей и параметры, от которых они зависят. Возможна более глубокая детализация некоторых из них, которая будет представлена при описании основных критериев.

Функциональная пригодность – это набор атрибутов, определяющий назначение, номенклатуру, основные необходимые и достаточные функции программных средств, заданные техническими требованиями заказчика или потенциального пользователя. Эти атрибуты можно численно представить точностью вычислений, относительным числом поэтапно изменяемых функций, коэффициентом изменения спецификаций требований разработчиками и заказчиками и т.д. В наибольшей степени функциональная пригодность проявляется в корректности и надежности программных средств.

В зависимости от назначения программных средств почти каждый из показателей этой группы может стать в значительной степени доминирующим и почти полностью определяющим функциональную пригодность. Однако практически во всех случаях важнейшим критерием является *функциональная корректность ПС*. Значения этого показателя зависят от функциональной корректности применяемых компонент, от методов их достижения и оценивания: детерминированно, стохастически и в реальном времени.

Корректность структуры программ определяется корректностью структуры модулей и корректностью структуры групп программ, построенных из модулей. Для оценки корректности структуры программ используется несколько частных показателей, различающихся степенью охвата тестами структурных компонент программы при отладке. Проверка корректности

структурных компонент производится статистически по исходным текстам программ или динамически при исполнении программы в объектном коде.

Корректность обработки данных определяется степенью отладки процесса обработки представительной выборки значений переменных в диапазонах их изменения. Для сложных программ принципиально невозможно провести исчерпывающее тестирование при всех значениях переменных, которые могут появиться при реальном функционировании программ. Поэтому устанавливается иерархия критериев и соответствующих им корректностей обработки данных в программе. Выбор критерия зависит от ответственности задач, решаемых программой, и от ресурсов, которые могут быть выделены на тестирование.

Корректность межмодульных связей и взаимодействия компонент. Взаимодействие программ определяется двумя видами связей между модулями: по управлению и по информации. Связи по управлению составляют вызовы программных модулей и возвраты в вызывавшие. Каждая связь модуля по управлению может содержать ошибку и являться причиной одного из видов некорректностей:

- * отсутствие вызова необходимого взаимодействующего модуля;
- * вызов модуля, не подлежащего исполнению при данном вызове;
- * возврат управления от вызванного к вызывающему модулю в точку, не предназначенную для возврата управления.

Взаимодействие модулей по информации может происходить через обменные переменные, непосредственно подготавливаемые и используемые соседними модулями, или через глобальные переменные. Некорректности взаимодействия модулей по информации очень разнообразны, трудно контролируемы и носят стохастический характер. Ряд искажений переменных при взаимодействии модулей не вызывает ситуаций отказа и отражается только в искажении результирующих данных на выходе модуля. Многообразие и сложность информационных связей в больших программных комплексах значительно затрудняют формализацию достигнутой корректности программ.

Детерминированная корректность программ определяется по частоте отклонения конкретных вычисляемых результатов от эталонных значений, заданных в техническом задании или в иных исходных документах. Точность оценки детерминированной корректности зависит от представительности тестовых наборов, широты варьирования переменных, точности сравнения результатов тестирования и т.д.

Стохастическая корректность характеризуется величиной статистического отклонения распределений и их параметров (средних значений, среднеквадратических отклонений) от заданных эталонов. При этом не оценивается каждый результат тестирования, а они обобщаются и оцениваются интегрально по некоторой достаточно представительной выборке.

Корректность в реальном времени определяется величиной максимального или усредненного отклонения траектории выходных параметров от заданной эталонной траектории обработки тестовых исходных данных. Результаты тестирования оцениваются интегрально на некотором временном интервале или по наибольшему отклонению от эталонной траектории.

Трудности *оценки объема тестирования*, достаточного для установления корректности программ, быстро возрастают по мере усложнения программных средств. Для подтверждения

абсолютной корректности программ достаточным в пределе является полный перебор всех входных и выходных тестовых значений переменных во всех их сочетаниях. Достаточный объем тестирования реально никогда не оценивается, и процессы отладки строятся с позиции необходимого минимума тестов, обеспечивающих корректность в предполагаемых наиболее активных режимах эксплуатации программ. Такой подход является конструктивным и реализуемым, позволяет сохранять затраты на отладку в разумных пределах. Для сложных программных средств объем достаточного тестирования может на несколько порядков превышать необходимое тестирование.

Функциональная пригодность определяется качеством системного и структурного проектирования программных средств. Критериями, отражающими эти свойства, могут быть: способность компонент к взаимодействию и степень стандартизации интерфейсов, мобильность программ и их защищенность от внешних воздействий.

Способность программных и информационных компонент к взаимодействию можно оценивать объемом изменений в программных средствах, которые необходимо выполнить при дополнении или исключении некоторой функции, когда отсутствуют изменения операционной среды или ЭВМ.

Для современного использования программных средств важным показателем функциональной пригодности стала *мобильность*, или *переносимость* программ в иную операционную среду или в иную по архитектуре ЭВМ. Это свойство может оцениваться объемом необходимых доработок программных средств, которые следует выполнить для обеспечения полноценного функционирования программных средств после переноса. Мобильность может оцениваться на уровне исходных текстов программ или на уровне объектного кода.

Удобство использования программных средств – понятие достаточно абстрактное и трудно формализуемое, однако в итоге определяющее функциональную пригодность и полезность применения программных средств. В эту группу показателей входят критерии, с различных сторон отражающие понятность, обучаемость и простоту использования.

Понятность программных средств можно описать четкостью концепции, широтой демонстрационных возможностей и наглядностью представления возможных функций. Кроме того, этот показатель характеризуется распознаваемостью модифицируемых параметров и адаптируемостью программных средств к конкретной среде и условиям применения.

Обучаемость можно оценить длительностью подготовки пользователя к полноценной эксплуатации программных средств. Это время зависит от возможности предварительного обучения и совершенствования в процессе эксплуатации, от возможностей оперативной помощи и подсказки (Help) при использовании программных средств, а также от доступности и удобства использования руководств и инструкций по эксплуатации.

Комфортность эксплуатации программных средств отражается простотой и удобством его использования, степенью учета физических и психологических характеристик пользователей. Они характеризуют:

- * легкость управления программных средств и объем параметров управления, реализуемых по умолчанию;
- * информативность сообщений пользователю и унифицированность управления

экраном;

- * степень доступности изменения функций в соответствии с квалификацией пользователя;
- * число операций, необходимых для запуска определенного задания;
- * временем ввода и отклика на задание;
- * длительностью решения типовых задач.

Надежность программ. Оценка качества программ по показателям надежности является более обобщенной, чем по показателям корректности. В этом случае регистрируются только такие искажения в процессе динамического исполнения программ, которые приводят к потере работоспособности программных средств или их крупных компонент. В теории надежности *работоспособным* называется такое состояние объекта, при котором он способен выполнять заданные функции с параметрами, установленными требованиями технической документации. Надежность является внутренним свойством систем, проявляющимся только во времени.

Причиной нарушения работоспособности программ при безотказности аппаратуры всегда является конфликт между реальными исходными данными, подлежащими обработке, и программой, осуществляющей эту обработку. Работоспособность программных средств можно гарантировать при исходных данных, которые использовались при отладке и испытаниях. Реальные исходные данные могут иметь значения, отличающиеся от заданных техническим заданием и от использованных при тестировании программ.

Устойчивость наиболее широко характеризует способность к безотказному функционированию после произошедших сбоев. Она зависит от уровня неустранимых ошибок и способности программных средств реагировать на проявления ошибок так, чтобы это не отражалось на показателях надежности. Последнее определяется эффективностью контроля за доступом к данным, степенью обеспечения их секретности и сохранности, а также селекцией достоверных данных, поступающих из внешней среды (живучесть), и средствами обнаружения аномалий функционирования программных средств.

Восстанавливаемость характеризуется полнотой восстановления функционирования программ после перезапуска–рестарта. Перезапуск должен обеспечивать возобновление нормального функционирования программных средств, на что требуются ресурсы ЭВМ и время. Поэтому полнота и длительность восстановления функционирования после сбоев отражают качество программных средств и возможность его использования по прямому назначению.

Показатели надежности программных средств в значительной степени адекватны аналогичным характеристикам, принятым для других промышленных изделий. Наиболее широко используется критерий *длительности наработки на отказ*. Для определения этой величины измеряется время работоспособного состояния системы между последовательными отказами или началом нормального функционирования системы после них. Вероятностные характеристики этой величины в нескольких формах используются как разновидности критериев надежности.

Основным показателем процесса восстановления являются *длительность восстановления* и ее вероятностные характеристики. Этот критерий учитывает возможность многократных отказов и восстановлений. Обобщение характеристик отказов и восстановлений производится в критерии *коэффициент готовности*. Этот показатель отражает вероятность иметь

восстанавливаемую систему в работоспособном состоянии в произвольный момент времени.

Эффективность использования ресурсов. Несмотря на быстрый рост ресурсов памяти и производительности современных ЭВМ, очень часто потребности в них для решения конкретных задач обгоняют их увеличение, и остается актуальной задача экономного использования ресурсов. Поэтому среди конструктивных критериев качества программных средств заметную, а иногда и главную роль играют критерии эффективности использования ресурсов памяти и производительности ЭВМ при реализации определенного программного средства.

Временная экономичность программного средства определяется длительностью выполнения заданных функций. Она зависит от скорости обработки данных, влияющей непосредственно на интервал времени завершения конкретного вычислительного процесса, и от пропускной способности, т.е. от количества заданий, которые можно реализовать на данной ЭВМ в заданном интервале времени. Эти показатели качества тесно связаны со *временем реакции* (отклика) программного средства на запросы для полного решения основных функциональных задач. Величина этого времени зависит от длительности решения задачи центральным процессором ЭВМ, от затрат времени на ввод и вывод данных и от длительности ожидания в очереди до начала решения задачи.

Пропускная способность системы программных средств и ЭВМ отражает число сообщений или запросов на решение определенных задач, обрабатываемых в единицу времени, зависящую от некоторого показателя внешней среды. Пропускная способность зависит от функционального содержания программного средства и конструктивной его реализации.

Ресурсная экономия отражает количество и степень занятости ресурсов центрального процессора, оперативной и внешней памяти, каналов ввода–вывода, терминалов и каналов локальной сети. Этот критерий определяется структурой и функциями программного средства, а также архитектурными особенностями и доступными ресурсами реализующей ЭВМ.