Министерство образования Российской Федерации

Международный образовательный консорциум
«Открытое образование»

Московский государственный университет экономики,
статистики и информатики

АНО «Евразийский открытый институт»

А.В. Бойченко, В.К. Кондратьев,
Е.Н. Филинов

Основы открытых информационных систем

Рекомендовано Учебно-методическим объединением по образованию в области прикладной информатики в качестве учебного пособия для студентов высших учебных заведений, обучающихся по специальности «Прикладная информатика (по областям)» и другим специальностям в области прикладной информатики

Москва 2004

УДК 004
ББК 32.973.202
К 642

РЕЦЕНЗЕНТ:
Кафедра проектирования экономических информационных систем
Московского государственного университета экономики, статистики и информатики

Бойченко А.В., Кондратьев В.К., Филинов Е.Н. **Основы открытых информационных систем.** 2-е издание, переработанное и дополненное. Под ред. Кондратьева В.К. //Издательский центр АНО «ЕОАИ». – М.:, 2004. – 128 с.

Книга может служить основой для изучения современных подходов к созданию открытых информационных систем (ОИС).

В книге рассматриваются архитектура и структура ОИС, модель среды открытых систем, дается описание и содержание профилей ОИС. Определяются объекты стандартизации в профилях информационных систем и указываются источники базовых стандартов информационных технологий. Дается объяснение принципов и задач компонентной разработки приложений.

Предназначена для студентов старших курсов, обучающихся по специальностям «Прикладная информатика (по областям)». Книга может быть полезна для студентов, обучающихся по специальности «Математическое обеспечение и администрирование информационных систем».

ISBN 5-7764-0284-0

Содержание

Учебное пособие

Предисловие ко второму изданию

Время, прошедшее после первого издания книги (изданной небольшим тиражом), и опыт преподавания соответствующей дисциплины в Московском государственном университете экономики, статистики и информатики подтвердили дальновидность разработчиков Государственного образовательного стандарта высшего профессионального образования по специальности 351400 «Прикладная информатика (по областям)», которые включили в состав требований к обязательному минимуму содержания основной образовательной программы подготовки информатика дисциплину «Технологии открытых систем».

Сегодня область информационных технологий является основой индустрии обработки информации – важнейшего сектора общественного производства, оказывающего глубокое воздействие на темпы и характер развития современного общества. Уровень развития информационной индустрии и соответствующих технологий определяется уровнем развития научно-методических основ и, в частности, нормативной базы в области информационных технологий. Поэтому создание таких научно-методических основ является актуальной и стратегически важной задачей современного общества. Большую роль в решении данной задачи играет система международных организаций, ответственных за процесс стандартизации информационных технологий. Масштабность, систематичность, интенсивность, научная обоснованность разработок в области стандартизации информационных технологий позволили к настоящему времени развить систему стандартов до такого уровня, при котором она становится главным носителем научно-методических основ области информационных технологий. Создание таких основ явилось решающим фактором для становления области информационных технологий как самостоятельной научно-прикладной дисциплины, имеющей характерные для нее предмет, методы исследования, фундаментальный методологический базис.

Авторы представленного учебного пособия основное внимание, как и подобает подобным изданиям, уделили базовым концепциям построения открытых информационных систем. В связи с этим, изложение материала размещено в той методической последовательности, которая должна помочь студентам в изучении подхода к проектированию информационных систем с учетом сложившейся мировой практики. При этом не ставилась задача обучения проектированию информационных систем, предполагается, что это должно быть известно читателю. Главной задачей является обучение основным целям создания и применения концепции, методов и стандартов открытых систем как основе повышения общей экономической эффективности разработки и функционирования информационных систем, обеспечение логической и технической совместимости их компонентов, обеспечение мобильности программ и данных информационных систем.

Написание материала учебного пособия распределилось следующим образом: А.В. Бойченко написаны главы 4, 5 и приложения 4, 5; В.К. Кондратьевым – главы 1, 2, 7 и приложения 1, 2, 4; Е.Н. Филиновым – введение, главы 3, 6.

Авторы учебного пособия выражают благодарность рецензентам – кафедре проектирования экономических информационных систем, профессору, кандидату экономических наук Ю.Ф. Тельнову за ценные замечания, которые позволили улучшить качество учебного пособия.

Авторы благодарят профессора, доктора экономических наук В.П. Тихомирова, профессора, доктора технических наук С.Д. Кузнецова и профессора, доктора экономических наук А.А. Емельянова за активную поддержку в реализации формирования учебно-методического комплекса по дисциплине «Открытые информационные системы».

Введение

Представленная в книге информация адресована студентам, обучающимся по дисциплине «Открытые информационные системы», входящей в состав программы подготовки специалистов в области проектирования и разработки информационных систем (ИС) – системных аналитиков, системных проектантов и интеграторов, прикладных программистов. В качестве объектов проектирования предполагаются корпоративные ИС промышленных предприятий, торговых фирм, финансовых групп и холдингов, коммерческих банков и других организаций, составляющих основу экономики страны. К ним также относятся информационные системы государственных структур, такие, как таможенная и налоговая службы, правоохранительные органы и др.

Как известно, к этим системам предъявляются весьма высокие требования, связанные с ответственностью их функционирования, необходимостью адаптации к непрерывно изменяющимся условиям бизнеса в рыночной экономике, информационного взаимодействия со смежными ИС бизнес-партнеров и государственных органов.

Одной из наиболее важных особенности развития современных ИС следует считать тенденцию к интеграции. Речь идет о функциональной интеграции, интеграции неоднородных информационных ресурсов, интеграции разных способов представления информации для пользователей информационных систем, интеграции информационных и/или вычислительных систем с телекоммуникационными системами.

К настоящему времени эта тенденция проявилась буквально во всех составляющих архитектуры и структуры ИС.

Функциональная интеграция связана с объединение систем, ранее функционировавших автономно, независимо друг от друга (как «островки автоматизации», например, на производственных предприятиях в 70-х и 80-х годах), в единую интегрированную ИС, обслуживающую все функции, подразделения и службы объекта, предприятия или учреждения. Такой подход определил новый взгляд на архитектуру и структуру прикладных программных комплексов, реализующих взаимодействующие подсистемы, и на построение ИС в целом. Обеспечивая новые качества деятельности предприятия или организации (например, в плане быстрого реагирования на требования рынка и соответствующей перестройки производства), функциональная интеграция влечет за собой резкий рост сложности ИС. В свою очередь, возросшая сложность ИС выдвигает дополнительные требования к методам и средствам проектирования, разработки прикладных программ и баз данных.

С функциональной интеграцией связана вторая важная особенность современных ИС – распределенная обработка данных.

Функции и подсистемы, интегрируемые в единую ИС, могут быть реализованы на ***гетерогенных*** (от греч. *heterogenes* — разнородный) программно-аппаратных платформах, расположенных в подразделениях предприятия, возможно, территориально удаленных друг от друга. Различия в классах и типах компьютеров, операционных систем, систем управления базами данных (СУБД) и других системных программ диктуются либо требованиями соответствия этих платформ классам решаемых задач, либо необходимостью использования уже находящихся в эксплуатации технических и программных средств. Характерный пример таких гетерогенных платформ – это комплексы распределенной обработки информации с архитектурой «клиент-сервер», состоящие из совокупности серверов (серверов приложений, серверов баз данных, файловых серверов и т.д.) и автоматизированных рабочих мест пользователей (персональных компьютеров или рабочих станций), которые связаны через локальную или корпоративную сеть предприятия.

Проектируя такую распределенную интегрированную ИС, системный интегратор должен найти решение, которое позволяет обеспечить взаимодействие приложений, поддерживаемых разными операционными системами. При этом должно использоваться программное обеспечение промежуточного слоя (между приложениями и операционными системами), реализующее услуги среды распределенной обработки, и сетевое программное обеспечение. Прикладные программы, предназначенные для совместного функционирования в такой гетерогенной среде, должны иметь программные интерфейсы, соответствующие интерфейсам служб и услуг этой среды.

Важную роль в построении современных ИС играет ***интеграция информационных ресурсов***. Под информационными ресурсами имеются в виду базы данных, базы знаний, базы программ повторного использования. Затраты на создание этих ресурсов весьма значительны, а объем и ценность уже имеющихся ресурсов велики и быстро возрастают. Поэтому при создании современных ИС приходится решать две задачи:

– как обеспечить использование уже существующих информационных ресурсов при разработке новой или развитии действующей ИС;

– как обеспечить совместное использование общих (разделяемых) информационных ресурсов несколькими ИС различного назначения.

При этом следует учитывать, что информационные ресурсы, подлежащие использованию в создаваемой ИС, могут быть реализованы с помощью разных моделей представления данных, представления знаний, языков программирования и функционировать в разных операционных системах.

Следовательно, инструментальные средства, применяемые для создания ИС, а также среда создаваемой ИС, в которой будут функционировать приложения, должны включать в себя компоненты, обеспечивающие унифицированные представления информационных ресурсов и способы обращения к ним.

В современных ИС требуется сочетать несколько разных видов информационных технологий (ИТ), характер которых определяется ***способами представления информации***. К ним относятся технологии:

– решения вычислительных задач и/или задач управления;
– обработки данных;
– обработки текстов;
– машинной графики;
– обработки изображений (статических и видеоизображений);
– обработки речевых сообщений.

Для этих технологий требуются прикладные программы, обрабатывающие соответствующие типы данных, и системное программное обеспечение среды ИС (например, СУБД), поддерживающее соответствующие структуры данных. Эти способы представления информации должны быть поддержаны средствами пользовательского интерфейса.

В последние годы активно развивается процесс создания и использования ***интегрированных информационно-телекоммуникационных систем*** (ИИТС). В таких системах объединяются функции ИС и систем передачи данных, когда предъявляются повышенные требования, например, по защите данных при их хранении, обработке и передаче по каналам связи. Примерами таких ИИТС являются: системы электронных межбанковских расчетов и другие системы финансово-кредитной сферы, системы управления авиаперевозками, системы, обслуживающие правоохранительные органы.

Другой класс ИИТС – это глобальные информационные сети, в которых традиционные услуги связи и передачи данных дополняются услугами информационного обслуживания массовых пользователей. Бурное развитие сети Интернет, в которой владельцы информационных ресурсов предоставляют их пользователям через соответствующие сер-

веры, стимулировало развитие технологий представления информационных ресурсов и доступа к ним, называемых Web-технологиями.

Применение Web-технологий в ИИТС обеспечивает развитие таких областей, как электронный бизнес (оптовая, вплоть до виртуальных бирж, и розничная торговля – Интернет-супермаркеты; взаимодействие промышленных предприятий с потребителями и поставщиками – business-to-business), электронные библиотеки, дистанционное обучение (вплоть до виртуальных университетов).

ИИТС представляют собой наиболее сложный класс современных ИС с точки зрения методов и средств их создания, сопровождения и развития, средств проектирования и программирования приложений, средств представления информационных ресурсов и доступа к ним, средств обеспечения информационной безопасности.

Учитывая изложенные выше особенности современных ИС, следует отметить, что разработчики и пользователи ИС сталкиваются с объективным противоречием между возрастающей сложностью ИС и жесткими ограничениями на затраты и сроки их проектирования и внедрения.

Деятельность предприятий и организаций в условиях конкурентной борьбы на рынке, подвергается непрерывным изменениям.

Это требует соответствующих изменений и в ИС, составляющих теперь неотъемлемую часть предприятия и подвергающихся бизнес-реинжинирингу вместе с ним. Очевидно, что в этом случае ИС должна быть спроектирована как модульная с тем, чтобы изменения могли касаться только тех функциональных частей ИС, которые требуется изменить, и не затрагивали бы другие функциональные части. Это свойство ИС называют расширяемостью (extensibility). Изменениям подвергаются также и количественные характеристики ИС – число обслуживаемых пользователей, размерность решаемых задач. Отсюда – необходимость обеспечить масштабируемость ИС (scalability).

Продолжительность жизненного цикла ИС, в частности прикладного программного обеспечения, в несколько раз превышает сроки морального и физического старения технических и системных программных средств. Поэтому необходимо обеспечить переносимость прикладных программных средств между разными аппаратно-программными платформами (portability). Требование переносимости приложений вытекает и из применяемых гетерогенных платформ распределенной обработки данных.

Требование обеспечить взаимодействие ИС с другими системами как по обмену данными, так и по управлению процессами их обработки (например, при выполнении транзакций) определяет необходимость наделить ИС свойством интероперабельности (interoperability).

Наконец, требование сокращения затрат и времени на подготовку пользователей к работе с ИС определяет необходимость обеспечить стабильный и дружественный пользовательский интерфейс (friendly user interface).

Совокупность указанных выше свойств характеризует открытые ИС (ОИС). Взятые по отдельности, эти свойства в той или иной мере были реализованы и ранее в предыдущих поколениях ИС. Особенностью современного подхода к открытым ИС является то, что способы обеспечить свойства открытости теперь рассматриваются в комплексе, как взаимосвязанные.

Идеология и стандарты ОИС служат ответом на указанное выше противоречие между возрастающей сложностью ИС и ограничениями средств и времени на их создание.

Ясно, что преимущества открытых систем не даются даром. За них приходится платить некоторой избыточностью системных ресурсов (производительностью, объемом памяти) по сравнению с минимальными ресурсами закрытых, монолитных систем.

Однако прогресс, достигнутый к настоящему времени, в развитии аппаратуры и системного ПО, позволяет иметь необходимые избыточные ресурсы в открытых ИС за приемлемую цену. Этот избыток окупается экономией затрат на проектирование и программирование ИС, которые пришлось бы производить, если не придерживаться идеологии и стандартов открытых систем, а также сохранением инвестиций, вложенных в создание открытых ИС.

Как было сказано выше, концепция открытых систем не родилась вдруг. Она формировалась постепенно, по мере осознания пользователями ИС их потребностей и возможностей, желания их освободиться от зависимости от одного поставщика технических и программных средств. Разработчики и поставщики этих средств также приходили к выводу о том, что действовать на всем спектре средств не под силу даже крупнейшим фирмам и выгоднее рационально использовать лучшие достижения других фирм вместо замкнутой фирменной идеологии.

Поэтому история концепции открытых систем неразрывно связана с процессом стандартизации информационных технологий.

Можно сказать определенно, что история открытых систем – это история стандартов открытых систем.

Еще в 60-х годах забота о переносимости приложений между компьютерами с разной архитектурой привела к созданию языков программирования высокого уровня и принятию стандартов на эти языки. Ранние версии языков программирования, например, КОБОЛ, ФОРТРАН и др. давали решение проблемы переносимости программ (и прикладных программистов) при переходе от одной аппаратной платформы к другой, хотя и ограничивали функциональные возможности стандартными рамками. Эти стандарты были реализованы практически всеми производителями платформ.

Когда международные и национальные организации по стандартизации приняли де-юре стандарты на языки программирования, языки стали независимыми от конкретного производителя. То, что ФОРТРАН, например, был создан фирмой IBM, уже не имело значения. Существенным было то, что, соблюдая нормы синтаксиса и семантики языка, производители могли конкурировать за счет совершенствования компиляторов, а пользователи – создавать прикладные программы без жесткой привязки к какой-либо конкретной машине.

Достижение этого уровня переносимости было первым примером возможностей открытых систем.

Создание семейства компьютеров IBM/360, обладающих единым набором команд и способных работать под управлением одной и той же операционной системы, было шагом вперед в достижении переносимости программ на уровне исполняемого кода, когда для переноса не требовалось перекомпилировать программу, а эффективность обеспечивалась использованием всех возможностей аппаратуры. Переносимость, однако, обеспечивалась только в пределах данной архитектуры и была свойством ИС, построенных на компьютерах IBM/360.

Семейство компьютеров VAX фирмы Digital Equipment, работающих под управлением операционной системы VAX VMS, было следующей фазой развития концепции открытых систем в направлении переносимости приложений в операционной среде с виртуальным адресным пространством.

Одновременно происходило бурное развитие сетевых технологий, сначала в виде фирменных архитектур сетей ЭВМ – SNA фирмы IBM, DECnet фирмы Digital, а затем стандартных протоколов локальных и глобальных сетей, например, протоколов TCP/IP сети ARPA, затем Интернет.

Внимание организаций по стандартизации к сетевым технологиям привело к созданию эталонной модели взаимосвязи открытых систем (ВОС) – OSI/ISO и разработке стандартов на интерфейсы и протоколы всех семи уровней этой модели.

Исторически самым жизненным вариантом создания общей базы переносимости явилась операционная система Unix. Простота Unix, позволяющая на понятном и легко описываемом уровне выразить спецификации её интерфейсов. Это дало возможность разрабатывать легко переносимые прикладные (и даже системные, кроме ядра ОС) программы между различными аппаратными платформами, работающими как под управлением разных реализаций Unix, так и платформами с другими ОС, удовлетворяющими стандартам на интерфейсы POSIX (Portable Operating System Interfaces) и X/Open. Конечно, эта возможность непосредственно связана с наличием единого языка программирования Си. Кстати, именно язык Си обладает наиболее ясным и четким стандартом.

Сама ОС Unix была почти целиком написана на языке Си, является модульной и относительно гибкой. Она включает в себя: аппаратно-зависимое ядро программ, непосредственно взаимодействующих с аппаратурой, набор инструментальных утилит, выполняющих основные действия по обработке данных, и оболочку, представляющую пользовательский интерфейс.

Возможности простого перехода пользователя с одной платформы на другую в Unix были предусмотрены с самого начала. Единственный интерактивный интерфейс взаимодействия пользователя с системой – это оболочка (shell). Сколько бы ни было реализаций Unix и диалектов языка shell, принцип взаимодействия оставался неизменным. Но решающим шагом в стандартизации пользовательского интерфейса стало создание оконной системы (XWindows System), которая поддерживается на всех Unix-платформах и обеспечивает единообразные средства графических пользовательских интерфейсов взаимодействия с системой.

Наконец, важнейшим шагом в формировании концепции открытых систем стала реализация в Unix стека протоколов TCP/IP (третий и четвертый уровни модели OSI/ISO). Это обеспечило возможность реальной интероперабельности независимо разработанных программных компонентов, функционирующих на разных компьютерах в различных операционных средах.

Легко видеть, что с появлением Unix и стандартов POSIX в концепции открытых систем появилось понятие интерфейсов прикладного программирования между приложениями и средой, в которой они функционируют. Это понятие стало центральным для стандартизации открытых систем.

Потребности обеспечить электронный обмен данными (Electronic Data Interchange – EDI) между разными ИС привели к созданию стандартов взаимодействия на уровне приложений этих ИС. Для разных областей применения ИС были разработаны международные стандарты:

– архитектуры учрежденческих документов и форматов обмена этими документами – ISO Open Document Architecture (ODA) и Open Document Interchange Format (ODIF);

– форматы межбанковских платежных документов и протоколы обмена SWIFT, принятые мировым банковским сообществом – Society for World-Wide Interbank Financial Telecommunications;

– форматы электронного обмена данными для администрации, торговли и транспорта EDIFACT (Electronic Data Interchange For Administration, Commerce and Transport).

Дальнейшее развитие этого направления было связано с выработкой стандартов на форматы электронного обмена данными для различных областей, прежде всего науки, культуры и образования. Стандартизированные форматы обмена электронными представлениями научных публикаций предложены для математики, химии, кристаллографии, астрономии, метеорологии, биологии и медицины и других областей науки с учетом специфики описания объектов в этих областях. Имеются также стандартизированные форматы

для электронного представления музейных коллекций (архитектуры, живописи, скульптуры, музыкальных произведений) и документальных архивов.

Проблемы создания электронных библиотек решаются с помощью стандартов на представления электронных каталогов, библиографических описаний, полных текстов публикаций, на поиск необходимой информации в этих документах и обмен ими.

Все это позволило решить проблему формирования и развития ***информационной инфраструктуры общества*** (глобальной для всего мира, национальной или региональной), представляющей собой совокупность информационных ресурсов и средств доступа к этим ресурсам. Информационные ресурсы, предоставляемые для коллективного использования через глобальные или корпоративные сети, формируются и поддерживаются в ИС, построенных на основе идеологии и стандартов открытых систем. Поэтому проблема информационной инфраструктуры, в части ее реализации, сводится к распространению идеологии открытых систем не только на ***каждую*** ИС, но и на ***совокупность*** ИС, взаимодействующих в рамках инфраструктуры.

Сегодня внимание в концепции открытых систем обращено не только на стандарты операционных систем (ОС) и баз данных (БД), но и на стандартизованные интерфейсы объединения существующих систем, приложений и пользователей. Этот подход требует концентрации на выработке и применении признанных в мире промышленных стандартов на интерфейсы, протоколы, форматы обмена данными. В гетерогенной среде современных открытых ИС могут объединяться мэйнфреймы (универсальные ЭВМ) и суперкомпьютеры, серверы на платформах Unix и Windows NT, персональные компьютеры и рабочие станции, связанные через локальные и глобальные сети. Все это требует систематической методологической основы для выбора и реализации стандартов открытых систем в каждой системе, подключаемой к сети, и в самой сети.

Дальнейшее изложение посвящено систематическому изучению этой методологии и включает:

− основные определения и свойства открытых систем;

− концептуальную модель открытых систем (имеется в виду референсная модель в отличие от моделей поведения систем);

− интерфейсы эталонной модели OSE/RM (Open Systems Environment/Reference Model), принятой в качестве методологической основы курса;

− средства обеспечения основных свойств открытых систем;

− методологию формирования и применения профилей открытых систем;

− базовые стандарты открытых информационных систем и информационных технологий.

Предполагается, что студенты, изучающие этот курс, знакомы с методологиями структурного и объектно-ориентированного анализа и проектирования информационных систем, и программированием прикладных программных комплексов для этих систем. Изучение данного курса поможет им разобраться в безграничном море существующих стандартов и спецификаций, выбрать стандартизованную среду, необходимую и достаточную для конкретной прикладной области, и, в конечном счете, ответить на вопрос, каким стандартам должна удовлетворять создаваемая ИС.

Развернутый ответ на этот вопрос, полученный в ходе проектирования ИС, позволит обеспечить условия, при которых создаваемые приложения могут жить достаточно долго, быть повторно используемыми в новых проектах и при развитии данной ИС. Это необходимо, чтобы в будущем не возникала существующая сейчас проблема «унаследованных» от прошлого систем, без которых нельзя обойтись, но которые трудно (а иногда и невозможно) сопровождать и развивать.

1. Основные определения и свойства открытых систем

1.1. Основные определения

Говоря об открытых информационных системах во введении, мы не давали точного определения этого термина. Дальнейшее изложение материалов курса будет исходить из определения, принятого Комитетом IEEE POSIX 1003.0. В соответствии с этим определением, открытая система есть ***«система, которая реализует открытые спецификации на интерфейсы, сервисы (услуги среды) и поддерживаемые форматы данных, достаточные для того, чтобы дать возможность должным образом разработанному прикладному программному обеспечению быть переносимым в широком диапазоне систем с минимальными изменениями, взаимодействовать с другими приложениями на локальных и удаленных системах, и взаимодействовать с пользователями в стиле, который облегчает переход пользователей от системы к системе».***

Определение POSIX позволяет нам с течением времени развивать понимание и реализацию открытых систем. Не останавливаясь на конкретных продуктах или областях технологии, оно предоставляет основу, которая может быть расширена с изменением потребностей.

Принятое здесь понятие открытости трактует открытую систему как среду, в которой функционируют приложения (application environment), основанную на стандартах интерфейсов протоколов и форматов данных, обеспечивающих переносимость приложений (application portability), «переносимость» пользователей (user portability) и взаимодействие (interoperability).

Ключевым моментом определения POSIX является понятие «открытая спецификация», т.е. «общедоступная спецификация, которая поддерживается открытым, гласным согласительным процессом, направленным на приспособление новой технологии к ее применению, и которая согласуется со стандартами». В соответствии с этим определением открытая спецификация является технологически независимой, т.е. не зависит от специфического аппаратного или программного обеспечения или продуктов конкретного производителя. Она должна быть одинаково доступна любой заинтересованной стороне. Более того, открытые спецификации находятся под контролем общественности, поэтому все организации, которых они затрагивают, могут участвовать в разработке и согласовании открытых спецификаций.

Спецификации многих фирм производителей аппаратуры и программного обеспечения, консорциумов этих фирм, описывающие технологии и продукты, разработанные ими, не подпадают под определение открытых спецификаций. ***Важнейшей характеристикой открытой спецификации являются не ее источник, а общедоступность и поддержка.***

Для дальнейшего изложения материала потребуются определения нескольких понятий, связанных с приведенной выше трактовкой понятия «открытые системы».

1.1.1. Функциональная среда открытых систем

Функциональная среда открытых систем (***Open System Environment – OSE***) поддерживает переносимые, масштабируемые и взаимодействующие друг с другом прикладные программы (приложения), предоставляя им стандартные услуги, интерфейсы, протоколы и форматы данных. Другими словами, среда OSE обеспечивает исполнение прикладных программ, если при их разработке были применены стандартные интерфейсы прикладного программирования, специфицированные для этой среды.

Стандартные интерфейсы могут быть представлены в виде международных, национальных или других открытых (общедоступных) спецификаций, которыми могут пользоваться разработчики ИС и поставщики технических и программных средств.

1.1.2. Интерфейсы прикладного программирования

Интерфейсом прикладного программирования называют набор исполняемых (*run-time*) программ или системных вызовов, которые позволяют прикладной программе пользоваться определенной услугой, представляемой либо операционной системой, либо другой прикладной программой. Для функциональной среды открытых систем API (***Application Program Interface***) – это набор программных интерфейсов между прикладными программами (приложениями) и средой. Спецификации API группируются по основным группам функций (услуг), предоставляемых средой приложениям:

- поддержка пользовательского интерфейса;
- организация процессов обработки данных;
- представление данных для хранения и обмена;
- услуги телекоммуникаций.

В случаях, когда услуги предоставляются другой прикладной программой, спецификации API содержат описание программного интерфейса между этими двумя прикладными программами.

1.1.3. Прикладная программа (приложение)

Прикладная программа (приложение) – это часть ИС, включающая в себя логически связанную группу программных модулей, данные, средства обращения к информационным ресурсам ИС, которые необходимы для выполнения определенной прикладной функции ИС. Прикладная программа должна обеспечивать взаимодействие с конечным пользователем ИС при выполнении этой прикладной функции ИС или группы функций.

Прикладные программы включают в себя: данные, документацию (исходные тексты и описания), обучающие средства, а также собственно программы в исполняемом коде.

1.1.4. Прикладная платформа

Прикладная платформа (*Application Platform*) – это операционная система и оборудование компьютера, на котором осуществляется выполнение прикладных программ. При построении ИС с архитектурой распределенной обработки данных типа «клиент-сервер» прикладные платформы серверов исполняют серверные части приложений, а платформы пользователей – клиентские части приложений.

Совокупность нескольких разных по своей архитектуре прикладных платформ может образовывать гетерогенную функциональную среду открытых систем.

1.1.5. Программные средства промежуточного слоя

Ряд стандартных услуг функциональной среды открытых систем реализуются непосредственно операционными системами прикладных платформ. Однако значительная часть услуг среды требует включения дополнительных программных средств. К таким услугам относятся:

- функции управления базами данных;
- функции организации распределенной обработки данных, для которых требуются мониторы транзакций, брокеры объектных запросов;
- средства защиты информации (аутентификации пользователей и приложений, управления доступом к данным и приложениям);
- услуги телекоммуникаций (например, электронной почты, передачи файлов и т.д.).

Такие программные средства занимают в референсной модели (от английского *Reference Model*) открытых систем промежуточное положение между приложениями и операционными системами и поэтому называются средствами промежуточного слоя (***middleware***).

Открытые спецификации интерфейсов программных средств промежуточного слоя составляют в настоящее время важную часть общего описания среды открытых систем. Особое внимание при этом обращается на средства интеграции приложений в гетерогенных средах открытых систем.

1.1.6. Архитектура и структура информационных систем

При проектировании открытых информационных систем следует различать понятия «архитектура» и «структура» ИС и ее функциональных частей.

Под архитектурой ИС понимается описание ее функций с точки зрения ***конечных пользователей*** и интерфейсов взаимодействия с внешней средой. Понятие архитектуры обычно вводится как внешний взгляд на описываемый объект, безотносительно к его реализации, в частности к структуре.

Декомпозиция заданных функций ИС, осуществляемая в процессе проектирования, дает структуру ИС в виде взаимодействующих между собой подсистем. Каждая из этих подсистем, в свою очередь может быть подразделена на составные части вплоть до неделимых модулей прикладных программ. Таким образом, понятие структуры ИС является иерархическим, включающим в себя несколько уровней разбиения, число которых зависит от того, каким образом предполагается реализовать свойство расширяемости ИС по функциям.

Полученные структурные единицы на каждом уровне разбиения подлежат архитектурному описанию, опять-таки с точки зрения выполняемых ими функций и интерфейсов взаимодействия с другими составными частями ИС, но без деталей их реализации.

Архитектуру функциональной среды открытых систем представляет совокупность спецификаций интерфейсов прикладного программирования (API), о которых было сказано выше.

Архитектуру прикладной платформы представляет спецификация интерфейсов операционной системы (оболочки, утилит, ядра, файловой системы, поддерживаемых сетевых протоколов).

Наконец, архитектура аппаратуры компьютеров представляется системой команд, системой организации прерываний, ввода-вывода и памяти.

1.2. Свойства открытых систем

Рассмотрим теперь более подробно свойства открытых систем, на которые опирается понятие «открытые системы», введенное в п. 1.1.

1.2.1. Расширяемость

Свойство расширяемости (*extensibility*) означает возможность добавления новых прикладных функций ИС или изменения некоторых функций из числа уже реализованных, не изменяя при этом остальные функциональные части (подсистемы) ИС. Эта возможность обеспечивается декомпозицией заданного состава прикладных функций ИС на подсистемы и модульным построением приложений, выполняющих функции подсистем. В полученной таким образом функциональной структуре ИС (имеющей иерархический характер) выделяются программные интерфейсы (API), специфицирующие взаимодействие приложений, принадлежащих одной или разным подсистемам. При внесении измене-

ний в какую-либо программу требуется сохранить ее интерфейс API с тем, чтобы не изменять другие программы, с которыми она взаимодействует.

1.2.2. Масштабируемость

Свойство масштабируемости (*scalability*) означает применительно к прикладным программам и базам данных, которые могут исполняться на разных прикладных платформах, возможность изменения их количественных характеристик (размерности решаемых задач, числа обслуживаемых пользователей и т.д.) путем настройки параметров, а не путем перепроектирования и программирования заново. Применительно к прикладным платформам свойство масштабируемости означает предсказуемый рост их количественных системных характеристик при добавлении определенных вычислительных ресурсов (например, процессоров, модулей оперативной и дисковой памяти в конфигурациях серверов).

1.2.3. Переносимость приложений, данных и персонала

Свойство переносимости (*portability*) означает возможность перевода ИС на более совершенные аппаратно-программные платформы при их модернизации или замене с минимальными затратами, сохраняя инвестиции, вложенные в разработку приложений, формирование массивов данных и обучение пользователей. Применительно к переносимости приложений (*application portability*) и данных (*data portability*) такая возможность обеспечивается соблюдением принятых стандартизованных API между приложениями и функциональной средой открытых систем. Применительно к «переносимости» пользователей (*user portability*) эта возможность обеспечивается дружественным пользовательским интерфейсом. Стабильность этого интерфейса поддерживается стандартизованными API среды по функциям пользовательского интерфейса, и сохранением средств взаимодействия с пользователем, реализуемых приложениями (экранные формы, способы работы с каталогами файлов, способы задания запросов к базам данных, командные языки и т.д.). Это необходимо для того, чтобы не переучивать пользователей при внесении изменений в приложения и прикладные платформы

1.2.4. Интероперабельность приложений и систем

Свойство интероперабельности (*interoperability*) означает возможность взаимодействия данной ИС с другими системами при необходимости обращения к информационным ресурсам (базам данных, базам знаний) этих систем или решения определенных задач с использованием их вычислительных ресурсов, если собственные ресурсы недостаточны. Интероперабельность систем обеспечивается, прежде всего, форматами данных, принятыми в качестве стандартов электронного обмена данными (electronic data interchange - EDI) для разных прикладных областей. Интероперабельность систем при запуске на исполнение программ, располагающихся в других системах, обеспечивается стандартами удаленного вызова процедур (remote procedure call – RPC).

В пределах каждой системы свойство интероперабельности рассматривается на трех уровнях:

− взаимодействие программ (program interaction), определяемое процессами взаимопередачи управления и обмена данными между программами;

− межзадачное взаимодействие (intertask communication), определяемое средствами языка программирования и операционной системы, которые обеспечивают запуск и синхронизацию задач и обмен данными между ними;

− межсетевое взаимодействие (internetworking), определяемое стандартными протоколами вычислительных сетей.

1.2.5. Способность к интеграции

На уровне интеграции систем (*system integration*) это свойство означает возможность объединения нескольких ИС различного назначения в единую интегрированную многофункциональную ИС.

На уровне баз данных (*database integration*) под интеграцией понимается представление для прикладной программы или пользователя нескольких баз данных как одной логически единой базы данных. Интеграция обеспечивает обращение пользователей к любой из этих баз данных независимо от места ее размещения, коллективный доступ к данным, одновременную обработку нескольких баз данных каждой из прикладных программ ИС.

На уровне интеграции данных *(data integration*) предполагается возможность одновременного и совместного использования программой или запросом пользователя нескольких файлов данных как единого целого (один из способов интеграции данных – базы данных). Логическая интеграция предполагает объединение данных на логическом уровне, не затрагивая их физической организации. Физическая интеграция связана со слиянием данных в единый информационный массив.

Наконец, на уровне интеграции приложений (application integration) рассматриваются методы и средства объединения прикладных программ в многозвенных архитектурах «клиент-сервер» с выделением серверов приложений, ориентированных на определенные классы задач, объектные среды и оболочки, позволяющие объединять приложения на основе механизмов обмена сообщениями.

1.2.6. Высокая готовность

Под свойством высокая готовность (*high availability*) понимается требование отказоустойчивости системы (*fault tolerance*), в которой в случае отказа какого-либо компонента гарантируется автоматическое восстановление работоспособности и сохранение целостности баз данных. Характеристика готовности, как меры способности системы принимать и успешно выполнять задания за доступный интервал времени, относится не только к открытым ИС. Здесь это свойство указано в связи с тем, что его реализация при проектировании системы находится во взаимосвязи с обеспечением других, указанных выше свойств.

1.3. Преимущества открытых систем

Применение идеологии и стандартов открытых систем выгодно для всех участников процесса создания и развития современных ИС.

В таблице 1.1 представлены преимущества применения идеологии открытых систем с разных точек зрения (пользователей ИС, проектировщиков ИС и системных интеграторов, программистов, реализующих прикладные программы ИС, разработчиков и поставщиков технических и программных средств).

Таблица 1.1

Преимущества открытых систем

Преимущества открытых ИС	Свойства открытых ИС, за счет которых достигаются преимущества
1. Для пользователей (заказчиков) ИС	
1.1. Сохранение уже сделанных инвестиций при изменении требований и развитии ИС	расширяемость, возможность замены отдельных приложений без изменения остальных, масштабируемость
1.2. Использование информационных ресурсов, существующих в других системах	интероперабельность
1.3. Дружественность человеко-машинного интерфейса, сокращение затрат на обучение персонала при переходе на новые версии ИС	«переносимость» пользователей
1.4. Освобождение от зависимости от одного поставщика технических и программных средств	переносимость приложений
2. Для проектировщиков ИС и системных интеграторов	
2.1. Возможность использования разных прикладных платформ	переносимость приложений
2.2. Повторное использование готовых приложений	переносимость приложений
2.3. Возможность использования существующих информационных ресурсов	интероперабельность
2.4. Облегчение решения проблемы «унаследованных» систем	интероперабельность, способность к интеграции
2.5. Возможность применения современных технологий и инструментальных средств анализа и проектирования ИС	переносимость приложений между инструментальными и целевыми прикладными платформами
3. Для прикладных программистов	
3.1. Модульная организация прикладных программных комплексов	
3.2. Применение стандартизованных программных интерфейсов	
3.3 Возможности применения компонентных технологий разработки	
3.4. Новые возможности разделения труда с использованием средств коллективной разработки	
4. Для поставщиков технических и программных средств	
4.1. Сокращение затрат на портирование прикладных и системных программных средств на новые аппаратные платформы	переносимость программ
4.2. Возможности интеграции выпускаемых программных продуктов с продуктами других поставщиков	способность к интеграции
4.3. Возможности расширения областей применения и рынков сбыта выпускаемых и разрабатываемых аппаратно-программных платформ	

Вопросы для самопроверки

1. Дайте определение понятия «открытая информационная система».
2. Что такое функциональная среда открытой системы?
3. Какие функции выполняют программные средства промежуточного слоя среды открытых систем?
4. Что является прикладной платформой ИС?
5. Назовите основные свойства открытых систем.
6. Назовите основные преимущества идеологии открытых систем для всех участников процесса разработки, развития и использования ИС.

2. Модели среды открытых информационных систем

2.1. Структура открытой информационной системы

Обобщенная структура любой ИС представляется состоящей из двух взаимодействующих частей (на рис. 2.1 обведены пунктирной линией):

– функциональной части, включающей прикладные программы, которые реализуют функции прикладной области;

– среды или системной части, обеспечивающей исполнение прикладных программ.

Обобщенная модель открытой информационной системы позволяет определить интерфейсы и протоколы взаимодействия как между приложениями в пределах одной открытой системы, так и между приложениями двух или более взаимодействующих систем. Эта модель учитывает тот факт, что всякая ИС может вступать во взаимосвязь со следующими сущностями «внешнего мира»:

– с пользователем (*User* – U), причем пользователем может быть как человек, так и прикладная программа;

– с внешней средой (*External Environment* – EE).

Взаимосвязь ОИС с «внешним миром» реализуется соответствующими интерфейсами:

– интерфейсом взаимодействия ОИС с пользователем (*User Interface* – UI);

– интерфейсом с внешней средой (*External Environment Interface* –EEI)

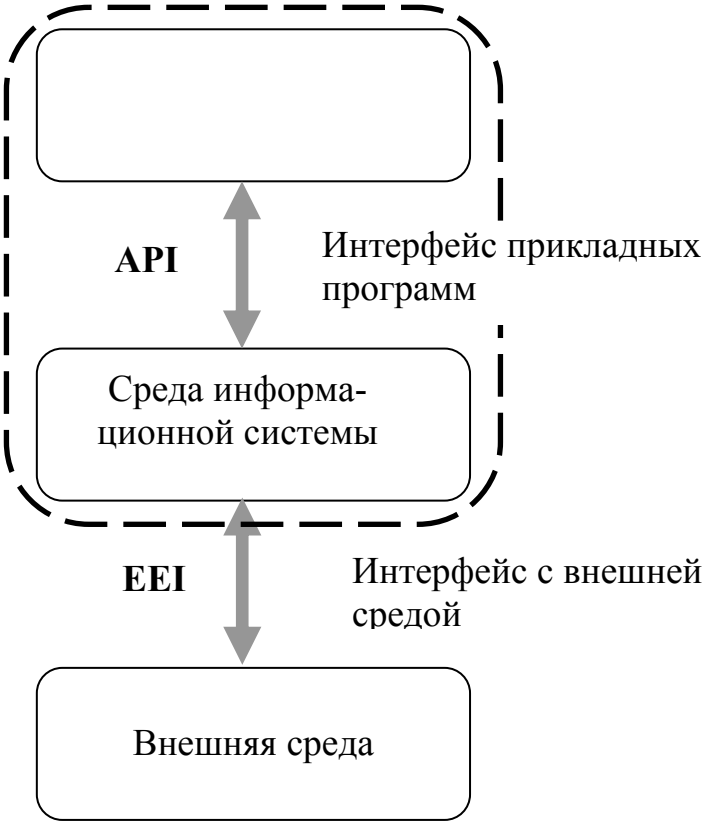

Рис. 2.1. Обобщенная структура информационной системы

Можно выделить тесно связанные две группы вопросов стандартизации:

– стандарты интерфейсов взаимодействия прикладных программ со средой ИС (*Application Program Interface* – API);

– стандарты интерфейсов взаимодействия самой ИС с внешней для нее средой (*External Environment Interface* – EEI).

Эти две группы интерфейсов определяют спецификации внешнего описания среды ИС – архитектуру с точки зрения конечного пользователя, проектировщика ИС, прикладного программиста, разрабатывающего функциональные части ИС.

Важным при рассмотрении интерфейса ОИС с внешней средой является то, что он определяет сопряжение ОИС и внешней среды при выполнении следующих ***групп функций***:

– взаимосвязь с пользователем (*User* – U);

– представление и хранение данных (*Information* – I);

– коммуникации, в том числе телекоммуникации (*Communication* – C).

Спецификации внешних интерфейсов среды ИС и, как будет видно далее, спецификации интерфейсов взаимодействия между компонентами самой среды, – это точные описания всех необходимых функций, служб и форматов определенного интерфейса. Совокупность таких описаний составляет модель открытых систем (Reference model).

2.2. Архитектура открытых систем

Как отмечалось выше, понятие архитектуры обычно вводится как внешний взгляд на описываемый объект, безотносительно к его реализации.

В связи с этим нужно уточнить представление об архитектуре систем и средств, как внешнем их описании (reference model) с точки зрения того, кто ими пользуется. Архитектура открытой системы, таким образом, оказывается иерархическим описанием ее внешнего облика и каждого компонента с точки зрения:

– пользователя (пользовательский интерфейс),

– проектировщика системы (среда проектирования),

– прикладного программиста (системы и инструментальные средства/среды программирования),

– системного программиста (архитектура ЭВМ),

– разработчика аппаратуры (интерфейсы оборудования).

Предлагаемый взгляд на архитектуру открытых систем вытекает из указанной выше необходимости комплексной реализации общих свойств открытости и является расширением принятого понятия об архитектуре (например, об архитектуре ЭВМ).

Для примера рассмотрим архитектурное представление системы обработки данных, состоящей из компонентов четырех областей: пользовательского интерфейса (соответственно точкам зрения всех указанных выше групп), средств обработки данных, средств представления и хранения данных, средств коммуникаций. Для этого представления требуется использовать три уровня описаний: среды, которая представляется системой, операционной среды (системы), на которую опираются прикладные компоненты, и оборудования. Каждый из этих уровней разделен для удобства на два подуровня (см. табл. П.4.2.1.).

Уровень среды для конечного пользователя (user environment) характеризуется входными и выходными описаниями (генераторы форм и отчетов), языками проектирования информационной модели предметной области (языки 4GL), функциями утилит и библиотечных программ и прикладным уровнем среды коммуникаций, когда требуются услуги дистанционного обмена информацией. На этом же уровне определена среда (инструментарий) прикладного программирования (*application environment*): языки и сис-

темы программирования, командные языки (оболочки операционных систем), языки запросов СУБД, уровни сессий и представительный среды коммуникаций.

На уровне операционной системы представлены компоненты операционной среды, реализующие функции организации процесса обработки, доступа к среде хранения данных, оконного интерфейса, а также транспортного уровня среды коммуникаций. Нижний подуровень операционной системы – это ее ядро, файловая система, драйверы управления оборудованием, сетевой уровень среды коммуникаций.

На уровне оборудования легко видеть привычные для разработчиков ЭВМ составляющие архитектуры аппаратных средств:

- система команд процессора (процессоров),
- организация памяти,
- организация ввода-вывода и т.д.,

а также физическую реализацию в виде:

- системных шин,
- шин массовой памяти,
- интерфейсов периферийных устройств,
- уровня передачи данных,
- физического уровня среды хранения.

Представленный взгляд на архитектуру открытой системы обработки данных относится к одномашинным реализациям, включенным в сеть передачи данных для обмена информацией. Понятно, что он может быть легко обобщен и на многопроцессорные системы с разделением функций, а также на системы распределенной обработки данных. Поскольку здесь явно выделены компоненты, составляющие систему, можно рассматривать как интерфейсы взаимодействия этих компонентов на каждом из указанных уровней, так и интерфейсы взаимодействия между уровнями.

Описания и реализации этих интерфейсов могут быть предметом рассмотрения только в пределах данной системы. Тогда свойства ее открытости проявляются только на внешнем уровне. Однако значение идеологии открытых систем состоит в том, что она открывает ***методологические*** пути к унификации интерфейсов в пределах родственных по функциям групп компонентов для всего класса систем данного назначения или всего множества открытых систем.

Стандарты интерфейсов этих компонент (де-факто или принятые официально) определяют лицо массовых продуктов на рынке. Область распространения этих стандартов являются предметом согласования интересов разных групп участников процесса информатизации – пользователей, проектировщиков систем, поставщиков программных продуктов и поставщиков оборудования.

Таблица 2.1

Детализация уровней модели среды открытых информационных систем

Иерархия представления архитектуры системы обработки данных	**Компоненты системы обработки данных**			
	Интерфейсы	Средства обработки данных	Представление и хранение данных	Коммуникации
Среда для конечного пользователя и инструментарий	Генераторы форм и отчетов	Утилиты и библиотеки	Языки программирования 4GL	Прикладной уровень OSI

Иерархия представления архитектуры системы обработки данных	**Компоненты системы обработки данных**			
	Интерфейсы	Средства обработки данных	Представление и хранение данных	Коммуникации
прикладного программиста	Языки программные и командные (оболочки)	Прикладные программы	Языки запросов СУБД	Уровни сессий и представительный OSI
Операционная система	Средства оконного интерфейса	Верхний уровень ОС (организация процесса обработки)	Средства доступа к среде хранения	Транспортный уровень OSI
	Драйверы	Ядро операционной системы	Файловая система	Сетевой уровень OSI
Оборудование	Системные интерфейсы (в т.ч. организация ввода-вывода)	Процессоры (система команд)	Организация памяти	Уровень передачи данных OSI
	Периферийные устройства	Системная шина	Шины (интерфейс) массовой памяти	Физический уровень OSI.

Выше был рассмотрен пример архитектуры открытых систем, реализующих технологию обработки данных. Можно было бы представить аналогичным образом открытые системы для всех классов информационных технологий: обработки текстов, изображений, речи, машинной графики. Особенно актуально проработать подходы открытых систем для мультимедиа-технологий, сочетающих несколько разных представлений информации.

Важным инструментом для выявления взаимосвязи различных функциональных компонент, используемых приложением в открытой среде, является модель такой среды.

Модель среды открытых систем должна отражать взаимодействие прикладных программ с другими компонентами среды, позволяя в каждом конкретном случае решать, какие стандарты необходимы для функционирования прикладной программы в выбранной среде. Известно несколько версий таких моделей, разработанных различными организациями. Основное отличие между ними заключается, как правило, в том, что внешняя по отношению к прикладной программе (или программной системе) среда подразделяется на различные элементы (службы) различным образом.

Общим для всех моделей является то, что с их помощью определяются место функциональных компонент и интерфейсов, обеспечивающих взаимодействие между прикладной программой и компонентами среды, которые обеспечивают те или иные виды обслуживания прикладных программ. Таким образом, модель позволяет структурировать и формально описать среду, в которой функционирует прикладная программа. С этой точки зрения модель может являться основой для применения точных методов для анализа характеристик системы и оптимизации последних с помощью различных методик и критериев.

В последние годы активно развивались два направления ***идеологии, концепции и системы стандартов открытых систем:***

– направление открытых вычислительных систем (*open computing systems* OCS), обеспечивающее возможность относительно простого и эффективного по трудоёмкости переноса апробированных программных средств и информации баз данных на различные

типы аппаратных платформ за счет стандартизации процессов и интерфейсов взаимодействия прикладных программ с операционными системами компьютеров;

– направление взаимосвязи открытых систем (*open system interconnection* – OSI), унифицирующее структуру, процессы и интерфейсы для обеспечения совместимости методов и средств обмена данными между разнотипными удаленными компьютерами, а также поддерживающее возможность предварительного выбора типов и ресурсов компьютеров в соответствии с потребностями ИС для решения конкретных прикладных задач.

В первом направлении основной задачей является перенос функций и процедур обработки данных, а также содержания баз данных между различными платформами. Подобные обмены функциями, процедурами и данными носят неоперативный характер и могут осуществляться вне реального масштаба времени. Проблемы состоят, преимущественно, в обеспечении сохранности апробированного функционального ядра программ и баз данных при их переносе на иные аппаратные платформы для снижения трудоемкости создания подобных систем.

Во втором направлении основную роль играет оперативная транспортировка данных между компонентами информационных систем в реальном масштабе времени. Проблема заключается в обеспечении совместимости различных систем передачи данных и эффективном использовании распределенных вычислительных ресурсов для обработки данных. Основной экономический эффект в этом случае достигается за счет сокращения дополнительных преобразований данных на стыках коммуникационных средств и повышения степени полезного использования вычислительных и коммуникационных ресурсов.

Важнейшими ***объединяющими*** целями развития обоих направлений открытых систем являются снижение трудоемкости и длительности создания, а также качества и функциональных возможностей современных ИС. В этих двух направлениях развиваются соответствующие концепции и методы, которые формализуются и детализируются комплексами стандартов. Для каждого направления характерно развитие методов и стандартов, ориентированных преимущественно на его поддержку и реализацию, а также некоторой общей части для обоих направлений. В результате выявились автономные части каждого направления и объединяющая их часть методов и стандартов открытых систем. В связи с этим, были созданы модели, призванные сгруппировать методы и стандарты открытых систем и представить их взаимодействие в наглядной форме.

Основой, обеспечивающей возможность реализации открытых систем, является совокупность стандартов, с помощью которых унифицируется взаимодействие аппаратуры и всех компонент программной среды: языки программирования, средства ввода/вывода, графические интерфейсы, системы управления базами данных, протоколы передачи данных в сетях и т.п. В результате сотрудничества многих национальных и международных организаций был определен набор стандартов, направленных на реализацию требований, обеспечивающих различные аспекты открытых систем.

2.3. Моделирование среды открытых систем

Важным инструментом для выявления взаимосвязи различных функциональных компонент, используемых прикладной системой в открытой среде, является модель такой среды. Модель отражает взаимодействие прикладных программ с системными программами и другими компонентами среды и позволяет в каждом конкретном случае решить, какие стандарты необходимы для функционирования прикладной программы в выбранной среде. На сегодняшний день не существует общепринятой и всеобъемлющей модели открытых систем. Различными организациями предложены свои версии моделей. Часть мо-

делей отражает отдельные аспекты взаимодействий в открытых системах, другие модели представляют обобщенный взгляд на системы в целом. Модели отличаются также и степенью проработанности, и набором используемых функциональных стандартов, обеспечивающих реализацию функций того или иного элемента модели.

Основное отличие между моделями заключается, как правило, в том, что внешняя по отношению к прикладной программе среда подразделяется на различные элементы (службы) различным образом. Общим для всех моделей является то, что с их помощью определяются положения и функциональных компонент и интерфейсов, обеспечивающих взаимодействие между прикладной программой и компонентами среды, которые обеспечивают те или иные виды обслуживания прикладным программам. Таким образом, модель позволяет структурировать и формально описать среду, в которой функционирует прикладная программа. С этой точки зрения модель может стать серьезной основой для применения точных методов для анализа характеристик системы и оптимизации последних с помощью различных методик и критериев, которые еще предстоит разработать.

2.3.1. Референсная модель (OSI/ISO)

Стандартизация функций информационного обмена между вычислительными системами имеет решающее значение для создания компьютерных сетей, интеграции предоставляемых ими ресурсов и услуг.

Когда речь заходит о моделях открытых систем, обычно упоминают известную референсную модель OSI/RM (Open System Interconnection Reference Model) или в русском варианте, «модель взаимосвязи открытых систем» ВОС. Эта модель берет свое начало из сетевой архитектуры SNA (System Network Architecture), предложенной IBM в 1974 году. Эта многоуровневая архитектура обеспечивала взаимодействие типа «терминал-терминал», «компьютер-компьютер» по глобальным связям. Нижние уровни архитектуры были реализованы специализированными аппаратными средствами, наиболее важным из которых являлся процессор телеобработки. Функции верхних уровней SNA выполнялись программными модулями. Один из них составлял основу программного обеспечения процессора телеобработки, другие входили в состав стандартной операционной системы.

Модель OSI/RM разработана международной организацией по стандартизации ISO. Ее описание приведено в документах, имеющих индекс ISO 7498, а также в рекомендации X.200 организации ITU-T (ранее, до 1994 г., называвшейся CCITT). Оба документа являются эквивалентными с технической точки зрения и имеют статус формального международного стандарта.

OSI/RM предназначена для определения общей основы процесса стандартизации в области взаимосвязи систем, обеспечивающей целостность и взаимную согласованность стандартов. Разработанные на этой основе стандарты позволяют реализовывать унифицированные средства обмена данными между системами, удовлетворяющие требованиям, определенным в модели OSI/RM. Системы, взаимодействующие посредством такого рода стандартных процедур обмена данными, называются «открытыми системами», а реализуемая ими взаимосвязь – «взаимосвязью открытых систем».

Модель описывает систему взаимодействий в процессах обмена сообщениями и данными между прикладными системами в вычислительных сетях. Она является наиболее проработанной с функциональной точки зрения, полноты набора стандартов и определения их совместимости друг с другом. При разработке модели использовался известный прием разбиения одной сложной задачи на несколько частных, более простых задач. При разработке модели было предложено разбиение среды на семь уровней (см. Приложение 1), взаимодействие между которыми описывается соответствующими правилами. Формали-

зованные правила, определяющие последовательность и формат сообщений, которыми обмениваются сетевые компоненты, лежащие на одном уровне, но в разных системах, называются протоколами. Модули, реализующие протоколы соседних уровней, взаимодействуют друг с другом также в соответствии четко определенными правилами, которые принято называть интерфейсами (см. 2.2.).

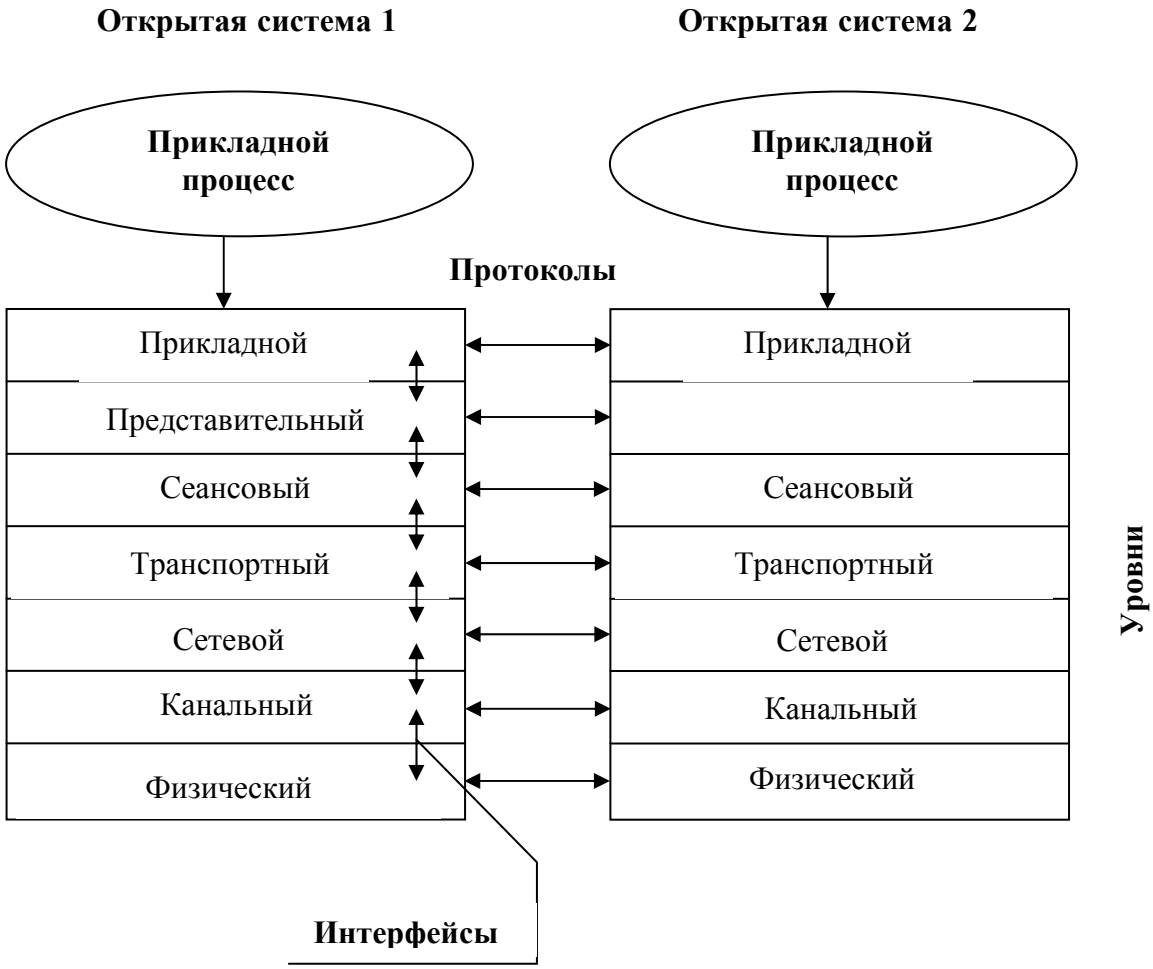

Рис. 2.2. Модель взаимосвязи открытых систем

Таким образом, моделью задается открытая коммуникационная среда, полностью независимая от того, как и на какой аппаратной и программной основе реализован каждый уровень. Вместе с тем, эта модель относится исключительно к области коммуникационных взаимодействий и не рассматривает взаимодействия составных элементов прикладных процессов в отдельной машине, на основе анализа которых возможно обеспечение мобильности прикладных программ. Это свойство модели легко объяснимо, так как в то время, когда формировалась основная концепция модели, мобильность программ основывалась, главным образом, на аппаратной совместимости платформ. Это, кстати, составляло основу технической политики ведущих фирм изготовителей ЭВМ и разработчиков программного обеспечения: IBM, DIGITAL EQUIPMENT, HP и др. В рамках данной модели отдельная машина рассматривается как единое целое. Подробнее на модели OSI, как составной части других моделей, мы остановимся ниже, при рассмотрении коммуникационного элемента моделей открытых систем.

2.3.2. Модель MUSIC

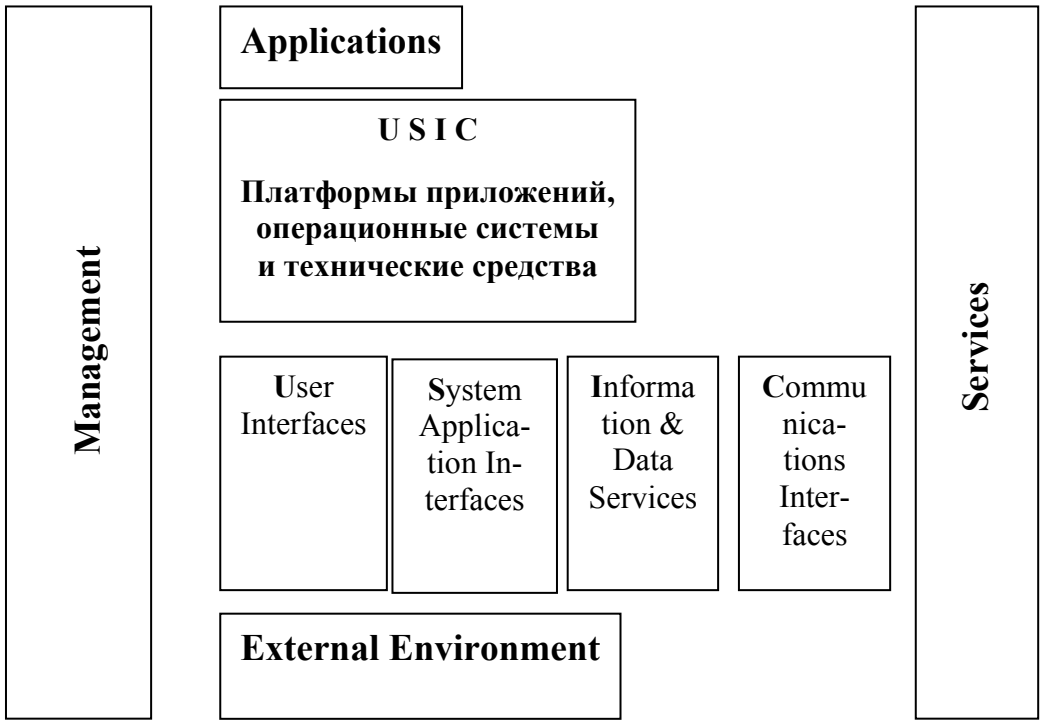

Рис 2.3. Модель MUSIC

Модель **MUSIC** была предложена Центральным Агентством по вычислительной технике и телекоммуникации (CCTA) Великобритании (рис.2.3).
MUSIC – это акроним от английских названий основных элементов модели:

M – **M**anagement;
U – **U**ser interface;
S – **S**ervice interface for programs;
I – **I**nformation and data formats;
C – **C**ommunications interfaces.

В модели MUSIC наибольшее внимание уделено тем аспектам взаимодействия и интерфейсам, которые могут оказаться критическими именно для прикладной системы, функционирующей в открытой среде. Несмотря на то, что модель не является ни всеобъемлющей, ни уникальной в смысле категорий используемых объектов, она обеспечивает ясность и четкое понимание связей между процессами, которые имеют место в открытых средах. Дальнейшее изложение строится в основном с использованием модели MUSIC. В следующих параграфах будут подробно рассмотрены функции и спецификации, определяемые для каждого из элементов модели.

2.3.3. Модель MIC

Модель открытой системы, разработанная AFUU (Французская Ассоциация пользователей UNIX и открытых систем) и AFNOR (Французская Ассоциация стандартизации), названа MIC (Model for Interactions between Components) – модель взаимодействия между компонентами, авторы также называют ее конвергентой моделью [23].

Эта модель представляет собой попытку объединить различные подходы к классификации компонент среды (рис. 2.4.)

	Область пользователя	**Область систем и процессов**	**Информационная область**	**Коммуникационная область**
Определения	Спецификация интерфейса пользователя	Спецификация процессов	Данные концептуального характера	Спецификации коммуникаций
Инструментарий высокого уровня	Объектное кодирование (символы)	Язык команд и программирования	Язык запросов	Сеансовый и представительский уровни модели OSI/RM
Системы высокого уровня	Многооконность	Система высокого уровня	Доступ к данным	Транспортный уровень модели OSI/RM
Система низкого уровня	Драйверы	Ядро системы	Файловые системы	Сетевой уровень модели OSI/RM
Исполнительные устройства высокого уровня	Интерфейсы	Центральный процессор	Память	Канальный уровень модели OSI/RM
Исполнительные устройства низкого уровня	Периферия	Шины	Шины и внешние накопители большой емкости	Физический уровень модели OSI/RM

Рис 2.4. Модель MIC

Модель строится в виде матрицы 7x4, столбцы которой соответствуют видам взаимодействия (обслуживания) в системе: взаимодействие с пользователем, системные средства, доступ к данным, коммуникационные средства. Легко видеть, что столбцы этой матрицы в точности соответствуют разбиению, предложенному в модели MUSIC (см. раздел 2.3.1.), за исключением отсутствующего элемента M (Management).

Строки матрицы соответствуют уровням обслуживания в рамках каждого типа взаимодействия от физического уровня до уровня связи с прикладной программой (или пользователем). Этот тип классификации соответствует принципу разбиения на уровни, принятому в коммуникационной модели OSI. Поэтому для варианта, использующего спецификации OSI для коммуникационных взаимодействий, столбец коммуникаций в точности соответствует модели OSI. Однако такое разбиение в настоящее время можно считать достаточно условным, поскольку на основе существующих стандартов далеко не все элементы допускают четкое разбиение на семь уровней. Так, даже коммуникационный элемент, реализованный на основе спецификаций TCP/IP, будет иметь другое разбиение.

Модель допускает использование различных стандартов для реализаций тех или иных функций, поэтому, в общем виде, модель представима в виде трехмерной матрицы, в которой третья координата используется для вариантов среды, которые строятся на основе различных стандартов, реализующих функциональные элементы модели.

2.3.4. Эталонная модель OSE/RM

Среда открытых систем OSE – это функциональная компьютерная среда, которая поддерживает переносимые, масштабируемые и взаимодействующие прикладные программы через стандартные услуги, интерфейсы, форматы и протоколы. Стандартами могут быть международные, национальные или другие открытые (общедоступные) спецификации. Открытые спецификации должны вырабатываться в ходе открытого процесса с участием всех заинтересованных сторон и быть доступны любому пользователю и поставщику для использования при построении систем и средств, удовлетворяющих критериям OSE.

Прикладные программы взаимодействуют, используя стандартные протоколы обмена данными, форматы обмена данными и интерфейсы систем распределенной обработки с целью передачи, приема, осмысленного восприятия и использования информации. Процесс перемещения информации из одной платформы через локальную вычислительную сеть, глобальную вычислительную сеть или комбинацию сетей к другой платформе должен быть прозрачен для прикладной программы и пользователя. Местоположение других платформ, пользователей, баз данных и программ также не должно иметь значения для данной программы. Короче говоря, OSE обеспечивает исполнение прикладных программ, используя хорошо определенные компоненты, методы сопряжения через соединители и модульный подход в разработке систем.

Комитетом IEEE POSIX 1003.0 была предложена эталонная модель среды открытых информационных систем OSE/RM (Open Systems Environment/Reference Model).

В целом функциональное обслуживание представлено следующими видами услуг среды ОИС:

- услуги, реализуемые операционной системой;
- услуги интерфейса «человек-машина»;
- услуги административного управления данными;
- услуги обмена данными;
- услуги программной инженерии;
- услуги компьютерной графики;
- сетевые услуги.

Кроме перечисленных выше основных видов услуг, существуют дополнительные, встроенные во все основные услуги: защиты информации, административного управления, а также набор инструментальных средств. Ниже приведено краткое описание каждого вида услуг.

– Услуги операционной системы являются корневыми в обеспечении функций прикладной платформы.

– Услуги интерфейса «человек-машина» определяют метод взаимодействия человека с прикладной программой.

– Услуги административного управления данными являются центральными для большинства систем относительно данных, которые могут быть определены независимо от процессов, создающих и коллективно использующих эти данные.

– Услуги обмена данными обеспечивают конкретную поддержку обмена информацией, включая формат и семантику элементов данных между прикладными программами одной и той же или различных платформ.

– Услуги программной инженерии охватывают стандартные языки программирования и инструменты программной инженерии.

– Услуги машинной графики обеспечивают функции, необходимые для создания выводимых на экран дисплея изображений и манипулирования этими изображениями.

– Сетевые услуги создают для распределенных прикладных программ возможности и механизмы доступа к данным и взаимодействия между ними в неоднородной сетевой среде.

– Услуги защиты информации предназначены для обеспечения защищенного распространения информации и защиты вычислительной инфраструктуры от несанкционированного доступа.

– Услуги административного управления – неотъемлемая часть любой операции, выполняемой в функциональной среде открытых систем. Они обеспечивают механизмы контроля и управления для операций, осуществляемых отдельными прикладными программами в базах данных, системах, сетях, а также средства взаимодействия пользователя с этими компонентами.

В простейшей форме эталонная модель OSE/RM иллюстрирует достаточно прямолинейные взаимоотношения пользователь-поставщик: прикладное программное обеспечение является пользователем предоставляемых услуг, а объекты прикладной платформы/внешней среды – поставщиком услуг. API и EEI определяют обеспечиваемые услуги.

Следует отметить, что помимо выше описанных, существуют и другие модели. Среди них можно отметить ряд специальных, т. е. проблемно ориентированных моделей. В частности, предлагаемая ISO модель ODP (*Open Distributed Processing*) – Открытая Распределенная Обработка – ориентирована, как следует из названия, на распределенную обработку в различных вычислительных сетях. Известны также модели CIM, EDI, Data Management DISC и др., описание которых можно найти в различных публикациях по проблемам открытых операционных систем.

2.3.5. Обобщенная модель среды открытых систем

Как уже отмечалось выше, OSE/RM – не единственная модель, используемая в качестве методологической основы стандартизации компонентов и интерфейсов среды открытых систем. На основе анализа и обобщения известных общих моделей (в том числе, MUSIC, MIC и OSI) модель среды ИС можно представить в виде матрицы типов компонентов этой среды, включающей три уровня, и четыре функциональные группы каждый (рис. 2.6).

Уровни описания в предлагаемой модели вместе с их подуровнями:

– компоненты служб и сервисов, предлагаемых средой для функционирования приложений, такие, например, как оконные оболочки, утилиты, системы программирования и системы управления базами данных;

– компоненты операционных систем;

– аппаратура: функциональные блоки и модули средств вычислительной техники и передачи данных (которые, например, видит системный интегратор при составлении спецификаций на оборудование ИС).

Функциональные группы компонентов в предлагаемой модели составляют:

– компоненты, обслуживающие интерфейс с пользователем (*User* – «U»);

– компоненты, обеспечивающие системные функции среды по организации процессов обработки данных (*System* – «S»);

– компоненты, обеспечивающие представление и хранение данных (*Information* – «I»);

– компоненты среды телекоммуникаций (*Communication* – «C»).

Модель предполагает, что взаимодействие между средой ОИС и внешней средой осуществляется с помощью ***трех*** типов интерфейсов (U, I и C).

Составные части ОИС разделены интерфейсом взаимодействия прикладных программ со средой ОИС, называемым интерфейсом прикладного программирования API. В отличие от интерфейса ОИС с внешней средой, этот (внутренний) интерфейс определяет сопряжение двух взаимодействующих объектов (функциональной части и среды ОИС) при выполнении функций не только групп U, I и C, но и функций среды по организации процессов обработки данных (*System* – S).

Рассмотрим сначала модель среды ИС, поддерживающей технологию обработки данных в одномашинной конфигурации (рис. 2.3). Это может быть, например, автоматизированное рабочее место или многотерминальная система какого-либо подразделения.

В соответствии с предложенным разделением компонентов на четыре функциональные группы на верхнем уровне, образующем собственно среду ИС, имеются:

– средства работы конечного пользователя с текстами и отчетными формами (текстовые редакторы, генераторы форм и отчетов, пакеты деловой графики и т.д.);

– языки и системы программирования, включаемые в целевые ИС для трансляции на месте новых версий фрагментов приложений;

– средства внесения новых или изменения существующих данных в информационной базе ИС;

– средства прикладного уровня эталонной модели взаимосвязи открытых систем (OSI/RM) для тех случаев, когда требуется дистанционный обмен информацией, (например, электронная почта или передача файлов).

Второй подуровень среды включает такие традиционные средства поддержки исполнения прикладных программ и работы пользователей, как:

– оболочки операционной системы, формирующие пользовательский интерфейс и командные языки;

– утилиты системных сервисов, библиотечные программы общего пользования; системы управления базами данных с характерными для них языками, в частности, со стандартным языком SQL;

– средства уровня представлений и уровня сессий эталонной модели OSI/RM.

Спецификации интерфейсов приведенных компонентов или систем представляют архитектурный облик среды ИС с точки зрения ее интерфейса с приложениями.

С другой стороны, эти компоненты или системы работают в среде операционной системы, составляющей второй уровень предлагаемой модели. На этом уровне присутствуют:

– средства оконного интерфейса, имеющиеся в составе операционной системы;

– средства организации процессов обработки данных;

– средства доступа к среде хранения данных;

– средства транспортного уровня эталонной модели ВОС.

Второй подуровень уровня операционной системы включает традиционные системные программы:

– драйверы ввода/вывода;

– ядро операционной системы;

– файловую систему;

– средства сетевого уровня эталонной модели ВОС.

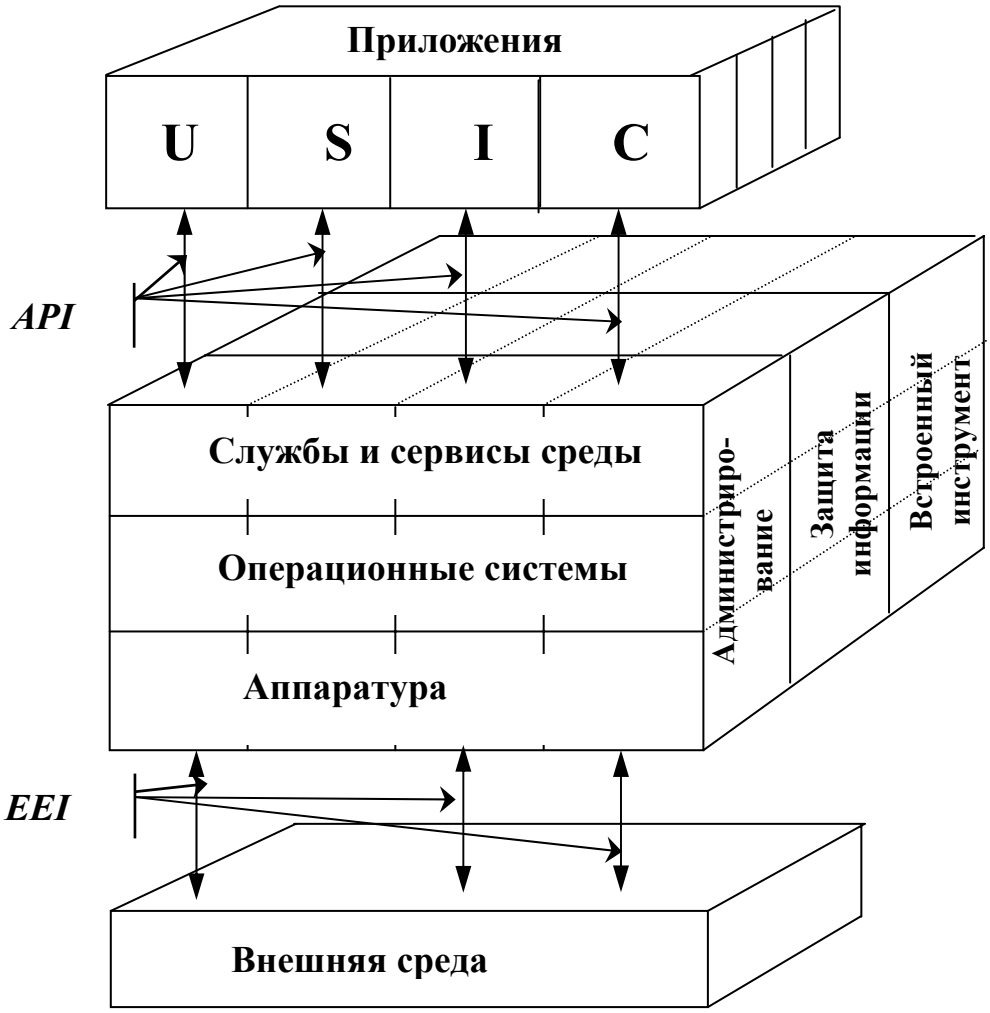

Рис. 2.5. Концептуальная модель OSE/RM

Выбор стандартов этого уровня определяется типами аппаратно-программных платформ: UNIX, Windows NT и т.д.

Нижний уровень предлагаемой модели среды ИС образуют спецификации аппаратуры. Здесь легко увидеть привычные для разработчиков ЭВМ и системных программистов характеристики архитектуры технических средств:

- организация ввода/вывода;
- система команд и управление прерываниями;
- организация памяти;
- канальный уровень (звено данных) эталонной модели ВОС.

Наконец, замыкают эту модель аппаратные интерфейсы:

- интерфейсы периферийных устройств;
- системная шина;
- интерфейс (шины) массовой памяти;
- физический уровень эталонной модели ВОС.

	U	S	I	C
Компоненты услуг среды	Текстовые процессоры, генераторы форм отчетов, и т.д.	Языки программирования	Средства проектирования и ведения баз данных	Прикладной уровень ВОС
	Оболочки ОС, командные языки	Утилиты, библиотеки программ	СУБД	Уровень представлений и сессий ВОС
Компоненты операционной системы	Оконный интерфейс	Организация процессов	Доступ к среде хранения	Транспортный уровень ВОС
	Драйверы ввода-вывода	Ядро ОС	Файловая система	Сетевой уровень ВОС
Аппаратура	Организация ввода-вывода	Система команд, организация прерываний, и т.д.	Организация памяти	Канальный уровень ВОС
	Интерфейс периферийных устройств	Системная шина	Шина массовой памяти	Физический уровень ВОС

Рис. 2.6. Модель среды информационной системы, поддерживающей технологии обработки данных в одномашинной конфигурации

Важно отметить, что современные тенденции в идеологии открытых систем направлены на освобождение потребителя от зависимости от одного определенного поставщика технических или программных средств. Прежде всего, это касается обеспечения переносимости (мобильности) программного обеспечения между разными аппаратными платформами. С этим связано постепенное смещение стандартизованных решений архитектуры от нижнего к верхним уровням модели среды ИС, хотя необходимость стандартизации ряда общих вопросов аппаратуры сохранится.

Для сложных и ответственных ИС важно предусмотреть наличие в модели следующих средств:

− защиты информации;

− встроенных инструментальных средств, предназначенных для развития и модернизации ИС силами пользователей;

− средств интернационализации/ локализации используемых программных продуктов, помогающих повторно использовать готовые программы.

Эти три группы функций могут относиться к разным функциональным группам компонентов, указанным выше, и поэтому они представляются третьим измерением предлагаемой модели.

Следует отметить, что характерной особенностью представленной модели является ее статичность, поскольку она остается неизменной в течение всего жизненного цикла ОИС. Она может быть успешно использована в качестве рабочего инструмента при формировании и применении стандартов ОИС, то есть на одном из начальных этапов создания ИС.

2.4. Цели создания эталонной модели OSE/RM

Пользователь рассматривает OSE/RM с той точки зрения, что она предоставляет ему все необходимое для доступа к технологии с целью получения желаемых результатов. Поставщик рассматривает OSE/RM с той точки зрения, что OSE/RM наиболее эффективно

и экономично обеспечивает разработчику все необходимое для поставки пользователям требуемой технологии. Стандарты OSE/RM должны быть, по возможности, изолированы от технологии. Тем не менее, очень многие изменения в технологии могут потребовать новых стандартов или новых версий существующих стандартов, и это должно учитываться при выборе необходимых стандартов.

Перечисленные ниже цели являются ключевыми в вопросе создания открытой системы. Описание этих целей вводит множество концепций, необходимых как для более четкого пояснения самих целей, так и для определения тех стандартов, которые требуются для достижения этих целей.

2.4.1. Переносимость прикладного программного обеспечения и повторное его использование

Исчерпывающий и согласующийся набор спецификаций OSE/RM на уровне исходного кода необходим для переносимости программного обеспечения между реализациями прикладной платформы. Это дает возможность организациям защитить свои вложения в существующее программное обеспечение, исключив затраты на его повторную разработку.

Переносимость прикладных программ часто связывается с одновременной переносимостью всей прикладной программы. Повторное использование программного обеспечения – это термин, используемый для описания переносимости только подмножества рабочих программ в новую прикладную программу. Новая прикладная программа может быть исполняема, но не обязательно на одной и той же прикладной платформе. Повторное использование программного обеспечения – это важный элемент в достижении преимуществ переносимости прикладных программ. Переносимость и повторное использование представлений, отличных от представлений в исходном коде, – это вторичная цель.

2.4.2. Переносимость данных

Стандарты OSE/RM должны обеспечить переносимость данных, хранимых во внешней памяти. Эта возможность должна позволить перемещать существующие данные на новую прикладную платформу и может использоваться также для обмена данными или для резервирования.

2.4.3. Взаимодействие приложений

Стандарты и OSE/RM должны определить спецификации по коммуникационным услугам и форматам, которые позволят двум логическим объектам программного обеспечения обмениваться данными и совместно их использовать. Эти спецификации должны быть предусмотрены для тех ситуаций, в которых взаимодействующие логические объекты функционируют на одной и той же или различных прикладных платформах.

2.4.4. Взаимодействие с точки зрения административного управления и защиты информации

Спецификации OSE/RM по прикладным платформам должны обеспечить возможность взаимодействия для целей административного управления и защиты информации между различными реализациями платформ.

2.4.5. Мобильность пользователей

Стандарты и OSE/RM должны позволить физическим лицам взаимодействовать с широким набором реализаций прикладных платформ без переобучения. Вариации в мето-

дах взаимодействия, которые не основаны на функциональных различиях или специальных требованиях, ухудшают производительность и должны исключаться общими спецификациями интерфейса пользователя.

2.4.6. Масштабируемость прикладной платформы

В тех случаях, когда требуются аналогичные услуги, и они обеспечиваются на различных типах прикладных платформ (например, на рабочих станциях и суперкомпьютерах), к каждой из этих платформ должны по возможности применяться одни и те же стандарты.

2.4.7. Масштабируемость распределенных систем

Стандарты OSE/RM должны определяться таким образом, чтобы скрыть механизм реализации услуг. Сложность реализации должна быть скрыта от пользователя услуг за интерфейсом услуг и, следовательно, должна быть прозрачна для пользователя. С точки зрения прикладного программного обеспечения это снижает объемы и стоимость прикладных программ и служит основой для миграций технологии.

Вопросы для самопроверки

1. Назовите составные части и интерфейсы открытых информационных систем.
2. Взаимодействие каких объектов обеспечивают интерфейсы API и EEI?
3. Назовите два основных направления, в которых развивались идеология, концепция и системы стандартов открытых систем.
4. Определите понятия «среда открытых систем» и «модель среды открытых систем»?
5. Что такое эталонная модель среды открытых систем?
6. Каковы цели создания эталонной модели OSE/RM?
7. Какова структура эталонной модели среды ОИС.
8. Назовите функциональные группы компонентов обобщенной модели среды ОИС.

3. Профили открытых информационных систем

3.1. Формирование и применение профилей открытых систем

Развитие и применение открытых информационных систем неразрывно связано с применением стандартов информационных технологий (ИТ). Основой применения этих стандартов в настоящее время стала методология функциональной стандартизации ИТ.

При создании и развитии сложных, распределенных, тиражируемых ИС требуется гибкое формирование и применение гармонизированных совокупностей базовых стандартов и нормативных документов разного уровня, выделение в них требований и рекомендаций, необходимых для реализации заданных функций ИС. Для унификации и регламентирования реализации заданных функций ИС такие совокупности базовых стандартов должны адаптироваться и конкретизироваться применительно к определенным классам проектов, функций, процессов и компонентов ИС. В связи с этой потребностью выделилось и сформировалось понятие «профилей» ИС, как основного инструмента функциональной стандартизации.

Профиль – это совокупность нескольких (или подмножество одного) базовых стандартов (и других нормативных документов) с четко определенными и гармонизированными подмножествами обязательных и факультативных возможностей, предназначенная для реализации заданной функции или группы функций.

Функциональная характеристика (заданный набор функций) объекта стандартизации является исходной для формирования и применения профиля этого объекта или процесса. В профиле выделяются и устанавливаются допустимые факультативные возможности и значения параметров каждого базового стандарта и/или нормативного документа, входящего в профиль. Профиль не может противоречить использованным в нем базовым стандартам и нормативным документам. Он должен использовать выбранные из альтернативных вариантов факультативные возможности и значения параметров в пределах допустимых.

На базе одной и той же совокупности базовых стандартов могут формироваться и утверждаться различные профили для разных проектов ИС и сфер применения. Эти ограничения базовых документов профиля и их гармонизация, проведенная разработчиками профиля, должны обеспечивать качество, совместимость и корректное взаимодействие компонентов системы, соответствующих профилю, в заданной области применения профиля.

Базовые стандарты ИТ и профили ИС в зависимости от проблемно-ориентированной области применения ИС могут использоваться как непосредственные директивные, руководящие или как рекомендательные документы, а также как нормативная база, используемая при выборе или разработке средств автоматизации технологических этапов или процессов создания, сопровождения и развития ИС.

Профили формируются в процессе проектирования ИС и составляют часть документации проекта системы. Следует рассматривать ***две группы*** профилей ИС:

1. ***Функциональные*** профили, регламентирующие архитектуру и структуру ИС и ее компонентов (функции, интерфейсы и протоколы взаимодействия, форматы данных и т.д.):

- профиль среды ИС (по основным функциям);
- профиль средств администрирования;
- профиль средств защиты информации;
- профиль инструментальных средств, встроенных в ИС.

2. ***Вспомогательные (технологические)*** профили, регламентирующие процессы жизненного цикла: проектирования, разработки, применения, сопровождения и развития ИС и их компонентов.

В зависимости от области распространения профилей они могут иметь разные категории и соответственно разные статусы утверждения:

– профили конкретной ИС, определяющие стандартизованные проектные решения в пределах данного проекта и являющиеся частью проектной документации;

– профили ИС, предназначенные для решения некоторого класса прикладных задач, которые распространяются на все ИС данного класса в пределах предприятия, отрасли или региона и утверждаются как стандарты предприятий, ведомственные или государственные стандарты.

Профили ИС унифицируют и регламентируют только часть требований, характеристик, показателей качества объектов и процессов, выделенных и формализованных на базе стандартов и нормативных документов. Другая часть функциональных и технических характеристик ИС определяется заказчиками и разработчиками творчески, без учета положений нормативных документов.

3.1.1. Назначение профилей

Профили ИС определяют комбинации базовых стандартов ИТ с целью:

– фиксирования конкретных классов, подмножеств, факультативных возможностей и параметров базовых стандартов для обеспечения взаимодействия программных и/или аппаратных компонентов в пределах одной системы или взаимодействия нескольких систем между собой;

– обеспечения совместимости компонентов, используемых в реальной прикладной системе, в соответствии с требованиями базовых стандартов;

– обеспечения вариантов использования в проектах ИС базовых стандартов, существенных как для пользователей систем, так и для поставщиков программных и технических средств;

– унификации при разработке тестов соответствия ИС или их компонентов требованиям профиля.

3.1.2. Категории и виды профилей

Как отмечалось выше, при определении номенклатуры разрабатываемых профилей ИС следует иметь в виду различные категории профилей. Эти категории вводятся в зависимости от сферы распространения профилей:

– профили конкретной ИС, определяющие стандартизованные проектные решения в пределах проекта данной системы;

– профили ИС, предназначенных для решения некоторого класса однотипных задач.

Виды профилей ИС и их компонентов вводятся в зависимости от набора функций, который охватывается профилем:

– базовые спецификации какого-либо компонента ИС;

– профили какой-либо группы функций, реализуемых несколькими компонентами (например, профиль протоколов транспортного уровня телекоммуникационной среды);

– профили аппаратно-программных платформ (например, платформы автоматизированного рабочего места);

– профили среды ИС, включая спецификации программных интерфейсов между приложениями и средой ИС;

– профили приложений;

– профиль интерфейсов между средой ИС и внешней средой;

– архитектурные спецификации – эталонные модели (например, эталонные модели OSE/RM и OSI/RM);

– полные профили ИС;

– стратегические профили – совокупности стандартов, определяющие ориентацию информатизации на длительное время (например, профиль переносимости приложений).

Предложения по разработке профилей должны учитывать состояние стандартизации ИТ, поскольку в настоящее время не все требуемые объекты стандартизации обеспечены международными стандартами и государственными стандартами России. Компоненты, применяемые при создании ИС, могут находиться на разных стадиях их эволюции. Могут встречаться прототипы новых технологий, полученные в результате исследований и разработок, продукты одного назначения, но несовместимые между собой, стандарты «де-факто» (общедоступные спецификации, предоставляемые поставщиками продуктов, соответствующих стандартам «де-факто»), международные и национальные базовые стандарты ИТ, профили и функциональные стандарты. При построении профилей ИС ***допустимо*** использовать стандарты или спецификации, относящиеся к применяемым в ИС компонентам, соответствующие той стадии эволюции, на которой эти компоненты находятся. ***Обязательным условием такого использования должно быть наличие общедоступных спецификаций, либо согласование использования спецификаций с их владельцем***.

3.1.3. Структура профилей

Структура и номенклатура профилей ИС тесно связаны с декомпозицией структуры самих ИС, которая позволяет представить разбиение системы на взаимодействующие подсистемы и компоненты. При этом для каждого компонента, выделяемого в структуре системы, конкретизируется состав выполняемых им функций и взаимосвязи с другими компонентами. Методологической базой при этом служит концептуальная модель ИС. Разбиение системы на взаимодействующие компоненты имеет иерархический характер в соответствии с декомпозицией по принципу «сверху вниз» и дает многоуровневую структуру построения системы. Этому же принципу должна соответствовать иерархия профилей, предусматривающая вложенность профилей, специфицирующих отдельные компоненты, в профили более крупных узлов.

В общем случае ИС, как открытая система, разбивается на прикладное ПО, реализующее функции основного назначения системы, и среду ИС, предоставляющую прикладным программам необходимые сервисы и услуги. Следовательно, в соответствии с принятой концептуальной моделью, для любой ИС должны быть определены:

– профили прикладного ПО (приложений);

– профили среды ИС, включающие в себя спецификации программных интерфейсов между приложениями и средой;

– профили интерфейсов между ИС и внешней для нее средой.

Профили прикладного ПО должны задавать архитектурное описание ИС с точки зрения состава прикладных функций ИС и регламентов их выполнения, а также общие требования к прикладным программам. Кроме того, профили прикладного ПО должны описывать его структуру, определяя состав компонентов и унифицированные интерфейсы взаимодействия между ними.

Общий подход к описанию структуры прикладного ПО и разделению функций между компонентами при построении профилей приложений целесообразно принять по тем же функциональным областям, которые установлены эталонной моделью среды OSE/RM. Это функции интерфейсов с пользователем, функции организации выполнения прикладных задач и обработки данных, функции представления и хранения данных (например, организация архивов), коммуникационные функции (например, представление форматов электронных сообщений).

Эти же четыре функциональные области представляют структуру профиля интерфейсов между приложениями и средой.

Профили среды ИС должны содержать:

– описание архитектуры среды с точки зрения наборов служб и сервисов, предоставляемых средой приложениям,

– описание структуры среды, содержащее ее разбиение на взаимодействующие компоненты программных и технических средств среды (в том числе на программные компоненты ПО промежуточного слоя),

– операционных систем,

– технические средства серверов и автоматизированных рабочих мест пользователей.

Правила построения профилей среды ИС базируются на выделении групп функций по четырем функциональным областям эталонной модели OSE/RM:

– функции поддержки человеко-машинного интерфейса;

– системные функции организации процессов обработки данных;

– функции поддержки представления и хранения данных;

– коммуникационные функции, в том числе функции телекоммуникационной среды.

Профиль среды ИС представляет собой набор выбранных базовых стандартов ИТ и других спецификаций, определяющих совокупность услуг, доступных прикладным программам. Примерами таких сред могут служить: среда рабочей станции (автоматизированного рабочего места), сосредоточенная среда управления выполнением прикладных программ, распределенная среда обработки данных, среда обработки прикладных транзакций. Эти среды могут иметь как различные, так и пересекающиеся наборы функций, которые могут определяться независимо друг от друга. Однако каждая функция должна быть представлена во всех средах ИС в соответствии с одним и тем же базовым стандартом.

Функции административного управления в ИС распределены между разными компонентами среды. Это функции управления в операционной системе, функции управления сетью или отдельными узлами сети, функции управления прикладными программами, функции управления базами данных, функции управления средствами пользовательского интерфейса, функции управления средствами защиты информации. Поэтому наборы функций административного управления выделяются в профиль администрирования ИС.

В составе профилей ИС выделяются также:

– профили защиты информации;

– профили инструментальных средств поддержки сопровождения и развития прикладных программных средств.

Эти группы функций могут быть реализованы как приложениями, так и средой ИС.

Здесь рассматриваются все функции среды открытых систем, которые определены концептуальной моделью ИС. Для построения профилей, соответствующих этим функциям, требуются базовые стандарты или общедоступные спецификации, которые относятся ко всему спектру стандартизованных элементов ИТ.

Профили интерфейсов между ИС и внешней для нее средой обеспечивают обмен информацией и состоят, главным образом, из протоколов, коммуникационных интерфейсов и поддерживаемых форматов данных, используемых при обмене информацией ИС с другими системами или внешними хранилищами данных.

Профили ИС, рассмотренные выше и связанные с описанием основных функций компонентов или ИС в целом, будем называть функциональными профилями.

Кроме функциональных профилей в составе профилей ИС должны рассматриваться ***вспомогательные профили***, регламентирующие процессы создания, сопровождения и развития ИС и средства поддержки этих процессов. Будем называть такие профили технологическими.

К технологическим профилям относятся:

– профили процессов жизненного цикла прикладных программных средств ИС;

– профили обеспечения качества прикладных программных средств ИС;

– профили инфраструктуры проекта ИС.

Среди них, в частности, должны быть предусмотрены:

– описание методологии и технологии создания, сопровождения и развития прикладных программных средств ИС;

– описание требований и регламентов тестирования прикладных программных средств;

– требования и регламенты документирования прикладного ПО;

– требования и регламенты управления конфигурацией прикладного ПО.

Набор функциональных и технологических профилей, названных выше, составляет полный профиль ИС.

Для каждой конкретной ИС состав ее полного профиля выбирается из указанного состава и конкретизируется в зависимости от требований, предъявляемых к ИС в техническом задании.

3.1.4. Цели и принципы формирования профилей информационных систем

Состояние и развитие стандартизации в области информационных технологий характеризуется следующими особенностями:

– существует несколько сотен разработанных международных и национальных стандартов, которые не полностью и неравномерно покрывают потребности в стандартизации объектов и процессов создания и применения сложных ИС;

– большая длительность разработки, согласования и утверждения международных и национальных стандартов (3-5 лет) приводит к их консерватизму и хроническому отставанию требований и рекомендаций этих документов от современного состояния техники и от текущих потребностей практики и технологии создания сложных ИС;

– стандарты современных ИС должны учитывать необходимость построения ИС как открытых систем, обеспечивать их расширяемость при наращивании или изменении выполняемых функций; переносимость прикладного программного обеспечения ИС между разными аппаратно-программными платформами; возможность взаимодействия с другими информационными системами той же проблемно-ориентированной сферы;

– в области ИС функциональными стандартами поддержаны и регламентированы только функционально наиболее простые объекты и рутинные, массовые процессы, такие, как телекоммуникация, программирование, документирование программ и данных и т.п.;

– наиболее сложные и творческие процессы создания и развития крупных распределенных ИС (системные анализ и проектирование, интеграция компонентов и систем,

испытания и сертификация ИС и т.п.) почти не поддержаны требованиями и рекомендациями стандартов, вследствие трудности их формализации, унификации и разнообразия содержания;

– чем сложнее объекты или процессы, подлежащие стандартизации, тем больше необходимо использовать и формулировать предварительных условий, учитываемых в требованиях и рекомендациях стандарта, которые следует адаптировать и конкретизировать для корректного их применения в определенном проекте;

– пробелы и задержки в подготовке и издании стандартов высокого ранга и текущая потребность унификации и регламентирования современных объектов и процессов в области ИС приводят к созданию и практическому применению многочисленных нормативных и методических документов отраслевого, ведомственного или фирменного уровня;

– последующие селекция, совершенствование и согласование нормативных и методических документов в ряде случаев позволяют создать на их основе национальные и международные стандарты.

В международной функциональной стандартизации ИТ принята жесткая трактовка понятия профиля. Считается, что основой профиля могут быть ***только международные и национальные утвержденные стандарты*** (не допускается использование стандартов де-факто и нормативных документов фирм). Подобное понятие профиля активно используется в гамме международных функциональных стандартов, конкретизирующих и регламентирующих основные процессы и объекты взаимосвязи открытых систем, в которых возможна и целесообразна жесткая формализация профилей (функциональные стандарты ИСО 10607 – 10613 и соответствующие им ГОСТ Р). Однако при таком подходе невозможны унификация, регламентирование и параметризация множества конкретных функций и характеристик сложных объектов архитектуры и структуры современных ИС.

Основными ***целями*** применения профилей при создании и использовании ИС являются:

– ***снижение трудоемкости, длительности разработки, стоимости, улучшение других технико-экономических показателей проектов ИС;***

– ***повышение качества разрабатываемых или применяемых покупных компонентов (и ИС в целом) при их разработке, приобретении, развитии и модернизации;***

– ***обеспечение расширяемости ИС по набору прикладных функций и масштабируемости*** в зависимости от размерности решаемых задач;

– ***обеспечение возможности функциональной интеграции*** в ИС задач, ранее решавшихся раздельно;

– ***обеспечение переносимости*** прикладного программного обеспечения (ПО) между разными аппаратно-программными платформами.

Выбор стандартов и документов для формирования профилей ИС зависит от того, какие из этих целей определены приоритетными. В ходе проектирования профиля цели уточняются.

Проектные решения, принятые на основании профилей, которые выбраны по целям с высшим приоритетом, фиксируются и определяют ограничения по выбору других составляющих профилей и их требований для достижения целей с более низкими приоритетами. Поставленные цели достигаются путем стандартизации и унификации построения и взаимодействия компонентов системы, обеспечения их совместимости, переносимости и качества.

Применение профилей при проектировании ИС позволяет ориентироваться на построение систем из крупных функциональных узлов, отвечающих требованиям стандартов профиля, применять хорошо отработанные и проверенные проектные решения. Профили определяют стандартизованные интерфейсы и протоколы взаимодействия компонентов системы таким образом, что разработчику системы, как правило, не требуется вдаваться в детали внутреннего устройства этих компонентов.

Таким образом, ***проектирование ИС в значительной степени может сводиться к ее компоновке из стандартизованных узлов. Этот подход позволяет осуществлять развитие и модернизацию ИС путем добавлений или замены отдельных узлов без изменения других частей системы.***

Применение стандартизованных профилей позволяет заказчику системы устранить зависимость от одного поставщика программных или аппаратных средств за счет выбора этих средств из числа доступных на рынке и соответствующих стандартам, нормативным требованиям и рекомендациям профиля.

Применение профилей, относящихся к прикладным программным комплексам (функциональным частям) ИС, облегчает ***повторное использование*** в проектируемой системе уже разработанных и проверенных прикладных программ.

В качестве методологической базы построения и применения профилей ИС необходимо использовать:

– ***ГОСТ Р ИСО/МЭК ТО 10000-1-99*** Информационная технология. Основы и таксономия международных функциональных стандартов. Часть 1. Общие положения и основы документирования.

– ***ГОСТ Р ИСО/МЭК ТО 10000-2-99*** Информационная технология. Основы и таксономия международных функциональных стандартов. Часть 2. Принципы и таксономия профилей ВОС.

– ***ГОСТ Р ИСО/МЭК ТО 10000-3-99*** Информационная технология. Основы и таксономия международных функциональных стандартов. Часть 3. Принципы и таксономия профилей среды открытых систем.

Эталонная модель среды открытых систем (OSE/RM) определяет разделение любой информационной системы на приложения (прикладные программы и программные комплексы) и среду, в которой эти приложения функционируют. Между приложениями и средой определяются стандартизованные интерфейсы. Эти стандартизованные интерфейсы являются необходимой частью профилей любой открытой системы. Кроме того, в профилях ИС могут быть определены унифицированные интерфейсы взаимодействия прикладных программ (функциональных частей) между собой и интерфейсы взаимодействия между компонентами среды ИС.

В соответствии с определениями профиля и базовых стандартов, входящих в профиль, в соответствии с ***ГОСТ Р ИСО/МЭК ТО 10000***, отдельные спецификации выполняемых функций и интерфейсов взаимодействия могут быть оформлены как профиль каждого компонента системы.

Таким образом, профили ИС, как сложной системы с иерархической структурой, могут включать в себя: *стандартизованные описания функций*, выполняемых данной системой, и взаимодействия с внешней для нее средой, *стандартизованные интерфейсы* между приложениями и средой ИС и *профили отдельных функциональных* компонентов, входящих в систему.

При практическом формировании и применении профилей ИС, в ряде случаев, можно использовать региональные, национальные стандарты, стандарты де-факто и ве-

домственные нормативные документы. Это может быть обусловлено отставанием в разработке некоторых задач в международных стандартах или необходимостью учета конкретных особенностей ИС. При применении стандартов и профилей могут быть выявлены пробелы в положениях некоторых стандартов и необходимость модификации или дополнения требований, определенных в них. Некоторые функции, не формализованные стандартами, но важные для унификации построения или взаимодействия компонентов ИС, могут определяться нормативными документами ведомства или фирмы, обязательными для конкретного профиля и проекта.

Особенности организационных структур, различия в размерах и сложности проектов ИС, требованиях к системам и применяемых методах их разработки, необходимость преемственности с системами, находящимися в эксплуатации, влияют на организацию разработки, приобретения, применения и сопровождения аппаратных и программных средств ИС.

Для эффективного применения конкретного профиля необходимо:

– ***выделить*** объединенные единой логической связью ***проблемно-ориентированные области функционирования***, где могут использоваться стандарты, общие для одной организации или группы организаций;

– ***идентифицировать стандарты*** и нормативные документы, варианты их применения и параметры, которые необходимо включить в профиль;

– ***документально зафиксировать*** участки конкретного профиля, где требуется создание новых стандартов или нормативных документов, и идентифицировать характеристики, которые могут оказаться важными для разработки недостающих стандартов и нормативных документов этого профиля;

– ***формализовать профиль*** в соответствии с его категорией, включая стандарты, различные варианты нормативных документов и дополнительные параметры, которые непосредственно связаны с профилем;

– ***опубликовать профиль*** и/или продвигать его по формальным инстанциям для дальнейшего распространения.

Каждый профиль ИС и его параметры для применения в конкретном проекте ИС необходимо поэтапно адаптировать и детализировать в соответствии со стадиями проекта ИС. Жизненный цикл конкретной ИС должен быть поддержан этапами развития и применения комплекта профилей в соответствии с основными процессами создания, сопровождения и развития ИС, включающими:

– ***системный анализ объекта*** информатизации и создание концепции ИС, когда производится первичный выбор исходного комплекта стандартов, которым должна соответствовать ИС, выявляется необходимость разработки и состав дополнительных нормативных документов; оформляется содержание и параметры комплектов документов предполагаемых профилей;

– ***проектирование ИС***, когда определяется ее архитектура и структура и, соответственно, уточняются положения, параметры и адаптируются стандарты комплекта профилей; они дополняются ведомственными нормативными документами; оформляются проекты документов и методических руководств по применению рабочей версии каждого профиля;

– ***разработка или приобретение готовых компонентов ИС***, при этом утверждаются и применяются все положения профиля; производится контроль, тестирование и испытания компонентов ИС на соответствие требованиям и документам конкретного профиля;

– ***сопровождение***, актуализация и развитие ИС, когда анализируются положения, параметры и результаты адаптации применяемой версии каждого профиля; выявляются и устраняются ее дефекты; производится модернизация профиля, с учетом появления более совершенных технических и программных средств и новых стандартов ИТ; при необходимости осуществляется формирование, документирование и внедрение новой уточненной версии соответствующего профиля.

При применении профилей ИС ***следует обеспечить проверку корректности*** их использования путем тестирования, испытаний и сертификации, для чего должна быть создана технология контроля и тестирования в процессе применения профиля. Она должна быть поддержана совокупностью методик, инструментальных средств, составом и содержанием оформляемых документов на каждом этапе обеспечения и контроля корректности применения соответствующей версии и положений профиля.

Применение профилей способствует унификации при разработке тестов, проверяющих качество и взаимодействие компонентов проектируемой ИС. Профили должны определяться таким образом, чтобы тестирование их реализации можно было осуществлять по возможности наиболее полно, по стандартизованной методике.

Ряд тестов проверки соответствия применяемых компонентов международным стандартам могут быть использованы готовыми, так как международные стандарты и профили являются основой при создании международно-признанных аттестационных тестов.

При ***сертификации*** информационных систем как специальный вид испытаний следует выделять сертификацию на соответствие профилям:

– ***процессов жизненного цикла ИС*** и ее компонентов;

– ***объектов ИС***, подготовленных и рекомендуемых для эксплуатации и сопровождения.

Следует учитывать, что общий объем испытаний при сертификации ИС и ее компонентов может быть значительно шире, чем проверка на соответствие профилям.

Для корректного применения профилей объектов и процессов ИС должна быть разработана совокупность ***методических руководств*** по использованию каждого профиля, в которых должно быть отражено:

– ***содержание и описание*** выбранных положений стандартов и нормативных документов профиля с позиции его пользователя;

– ***параметры адаптации*** стандартов профиля и содержание дополнительных нормативных документов;

– ***методика и сценарии*** корректного применения всех обязательных и рекомендуемых положений профиля;

– ***требования к содержанию*** отчетов о результатах контроля и тестирования компонентов ИС на соответствие обязательными положениям профиля в процессе их жизненного цикла.

3.1.5. Формирование содержания профилей информационных систем

Разработка и применение профилей являются ***органической*** частью процессов проектирования, разработки, сопровождения, модернизации и развития ИС. Профили характеризуют каждую конкретную ИС на всех стадиях ее жизненного цикла постольку, поскольку они задают гармонизированный набор базовых стандартов, которым должна соответствовать система и ее компоненты.

Проектированию системы предшествует стадия предпроектного обследования объекта автоматизации, результатом которой являются его функциональная и информацион-

ная модели, определение целей создания системы и состава ее функций. Стандарты, важные с точки зрения заказчика, должны задаваться в техническом задании (ТЗ) на проектирование системы и составлять ее первичный профиль. То, что не задано в ТЗ, остается первоначально на усмотрение разработчика системы, который, руководствуясь требованиями ТЗ, может дополнять и развивать профили ИС, которые впоследствии согласуются с заказчиком.

Таким образом, ***профиль*** конкретной системы ***не является статичным***, он развивается и конкретизируется (возможно, во взаимодействии с заказчиком) в процессе проектирования ИС и оформляется в составе документации проекта системы. В профиль конкретной системы включаются спецификации компонентов, разработанных в составе данного проекта, и спецификации использованных готовых программных и аппаратных средств, если эти средства не специфицированы соответствующими стандартами.

После завершения проектирования и испытаний системы, в ходе которых проверяется ее соответствие профилю, профиль применяется как основной инструмент сопровождения системы при эксплуатации, модернизации и развитии.

Каждый из выделенных профилей должен для последующего длительного использования пройти стадию формирования, адаптации и параметризации применительно к характеристикам стандартизируемых объектов или процессов создания ИС. Такая подготовка профилей должна проводиться с учетом применяемых методов и средств, текущего состояния и ведущихся работ на реальных компонентах ИС. Подготовка профилей к применению должна учитывать реальное состояние проекта ИС. При этом возможны следующие варианты:

– ***планируется*** создание новой ИС в условиях отсутствие задела по системе и компонентам данного проекта;

– ***имеется*** типовой проект ИС, и предстоит его адаптация и реализация;

– ***существует и эксплуатируется*** реальная ИС, для которой следует подготовить и адаптировать профили с учетом ее реального состояния и перспективы развития.

Как было сказано выше, при формировании и применении профилей конкретных ИС допустимо использовать как международные и национальные стандарты, так и ведомственные нормативные документы, а также стандарты де-факто при условии доступности соответствующих им спецификаций.

Для обеспечения корректного применения профилей их описания должны содержать:

– ***определение целей***, которые предполагается достичь применением данного профиля;

– ***точное перечисление функций*** объекта или процесса стандартизации, определяемого данным профилем;

– ***формализованные сценарии*** применения базовых стандартов и спецификаций, включенных в данный профиль;

– ***сводку требований к ИС*** или к ее компонентам, определяющих их соответствие профилю и требований к методам тестирования соответствия;

– ***нормативные ссылки*** на конкретный набор стандартов и других нормативных документов, составляющих профиль, с точным указанием используемых редакций и ограничений, способных оказать влияние на достижение корректного взаимодействия объектов стандартизации при использовании данного профиля;

– ***информационные ссылки*** на все исходные документы.

Процессы, выполняемые на протяжении жизненного цикла ИС, могут быть разбиты на три группы:

– процессы, непосредственно связанные с созданием, эксплуатацией и сопровождением ИС (прикладного программного обеспечения и среды ИС);

– процессы, обеспечивающие контроль и управление выполнением всех остальных процессов; организационные процессы, обеспечивающие организацию работ на протяжении жизненного цикла ИС;

– процессы поддержки, каждый из которых обеспечивает технологическую поддержку всех остальных процессов на протяжении жизненного цикла ИС (процессы поддержки разработки документации ИС, процессы обеспечения качества прикладного ПО, процессы тестирования прикладного ПО, процессы создания и поддержки инфраструктуры проекта – методологии и инструментальных средств, процессы обучения).

Практически все указанные процессы тесно связаны между собой либо по результатам, либо по выполняемым работам. Уровень стандартизации профилей, процессов и объектов их применения отражается не только на технико-экономических показателях ИС, но и, что особенно важно, на их качестве. ***Качество информационных*** систем тесно связано с методами и технологией их разработки, поэтому важной ***группой документов в профилях являются стандарты и их рекомендации по непосредственному обеспечению качества ИС.***

На стадиях жизненного цикла ИС выбираются и затем применяются основные ***функциональные профили***:

– профиль прикладного программного обеспечения;

– профиль среды ИС;

– профиль защиты информации в ИС;

– профиль инструментальных средств, встроенных в ИС.

Прикладное программное обеспечение является всегда проблемно-ориентированным и определяет основные функции информационной системы. Функциональные профили ИС должны включать в себя гармонизированные базовые стандарты. При применении функциональных профилей ИС следует также иметь в виду ***согласование*** (*гармонизацию*) этих профилей между собой. Необходимость такого согласования возникает, в частности, при применении стандартизованных API интерфейсов, в том числе интерфейсов приложений со средой их функционирования, интерфейсов приложений со средствами защиты информации. При согласовании функциональных профилей возможны также уточнения профиля среды ИС и профиля встраиваемых инструментальных средств создания, сопровождения и развития прикладного программного обеспечения.

Применение функциональных профилей поддерживают вспомогательные технологические профили:

– профиль жизненного цикла прикладных программных средств;

– профиль обеспечения качества прикладных программных средств;

– профили инфраструктуры обеспечения проекта ИС, в том числе профили методологий и технологий создания, сопровождения и развития ИС, тестирования прикладных программных средств, документирования прикладных программных средств.

Взаимосвязи функциональных профилей ИС и вспомогательных профилей, поддерживающих создание, сопровождение и развитие ИС, показаны на рис. 3.1.

Функциональные профили ИС состоят из профилей компонентов, реализующих те или иные прикладные функции или функции среды ИС. Детализация функциональных профилей производится по мере декомпозиции структуры ИС на составляющие ее компо-

ненты в ходе проектирования системы. Следовательно, выбор и применение функциональных профилей являются органической частью процессов проектирования, разработки, сопровождения и развития системы. Применение функциональных профилей ИС заключается в выполнении следующих работ:

– выбор готовых программных и аппаратных средств, соответствующих профилям;

– проектирование и разработка прикладного программного обеспечения (функциональных частей) ИС в соответствии с выбранными профилями, в частности, в соответствии со стандартизованными интерфейсами;

– разработка требований к методам тестирования компонентов ИС на соответствие функциональным профилям, выбор или разработка тестов соответствия;

– тестирование компонентов ИС на соответствие профилям или проверка сертификатов соответствия для применяемых готовых программных и аппаратных средств;

– комплексирование компонентов в создаваемой системе на основе последовательного применения функциональных профилей.

Нормативные документы, регламентирующие жизненный цикл ИС и ее профилей, либо задаются директивно в ТЗ на создание системы, либо выбираются разработчиком в зависимости от характеристик проекта. Эти нормативные документы, адаптированные и конкретизированные с учетом характеристик проекта и условий разработки, составляют профиль жизненного цикла конкретной системы. В этом профиле должен быть учтен набор этапов, частных работ и операций, связанных с разработкой и применением профилей ИС, специфицирующих ее проектные решения. При этом надо иметь в виду итерационный характер формирования и ведения профилей конкретной ИС, связанный с итерациями самих процессов проектирования и сопровождения системы. Профиль жизненного цикла должен определять стадии создания, сопровождения и развития ИС, а также все основные и поддерживающие процессы, выполняемые на протяжении жизненного цикла.

Международные стандарты, регламентирующие жизненный цикл сложных информационных систем, в настоящее время отсутствуют. Поэтому ниже представлены методические рекомендации по разработке и применению профиля жизненного цикла в проектах конкретных ИС, основанные на стандарте ИСО/МЭК 12207:1995 «Информационные технологии. Процессы жизненного цикла программного обеспечения».

Такой подход правомерен постольку, поскольку программное обеспечение составляет большую часть стоимости и трудозатрат на создание современных ИС, а продолжительность жизненного цикла программного обеспечения фактически определяет продолжительность жизненного цикла ИС. Кроме того, современные методы создания программного обеспечения, переносимого между разными аппаратно-программными платформами, позволяют уменьшить зависимость жизненного цикла ИС от жизненного цикла технических средств.

Наиболее актуальными в настоящее время представляются открытые распределенные ИС с архитектурой «клиент-сервер». Поэтому ниже рассматриваются подходы к построению функциональных профилей таких систем.

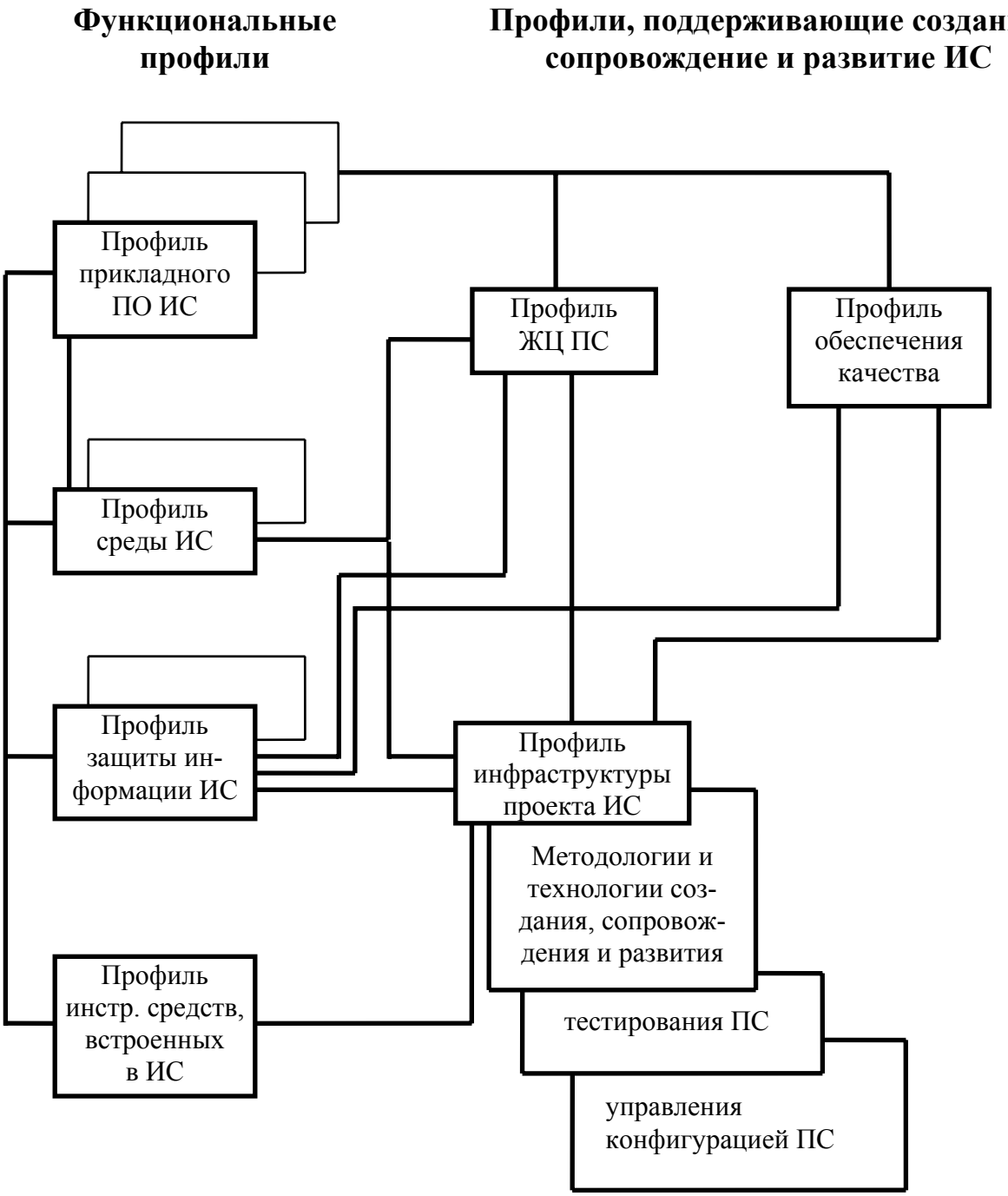

Рис. 3.1. Взаимосвязи функциональных профилей ИС и профилей, поддерживающих создание, сопровождение и развитие ИС

Профиль среды ИС должен определять ее архитектуру в соответствии с выбранной моделью распределенной обработки данных: моделью DCE (*Distributed Computing Environment*) или моделью CORBA (*Common Object Request Broker Architecture*), которые будут рассмотрены в главе 6. В первом случае модель определяется стандартами Консорциума OSF (см. приложение 2), в частности механизма удаленного вызова процедур RPC (*Remote Procedure Call*) с учетом стандартов де-факто, специфицирующих применяемые

мониторы транзакций. Во втором случае модель определяется стандартами консорциума OMG, в частности спецификацией брокера объектных запросов ORB (*Object Request Broker*). Стандарты интерфейсов приложений со средой ИС – API должны быть определены по функциональным областям профилей ИС. Декомпозиция структуры среды функционирования ИС на составные части, выполняемая на стадии эскизного проектирования, позволяет детализировать профиль среды ИС по функциональным областям эталонной модели OSE/RM:

– графического пользовательского интерфейса (например, стандарт *Motif* консорциума OSF или стандарт *X Window* IEEE);

– реляционных или объектно-ориентированных СУБД (например, стандарт языка SQL-92 и спецификации доступа к разным базам данных);

– операционных систем с учетом сетевых функций, выполняемых на уровне операционной системы (например, набора стандартов POSIX – ISO и IEEE);

– телекоммуникационной среды в части услуг и сервисов прикладного уровня: электронной почты (по рекомендациям ITU-T X.400, X.500), доступа к удаленным базам данных RDA (по стандарту ISO 9594-1.2), передачи файлов, доступа к файлам и управления файлами (по стандарту ISO 10607 – 1,2,3,4,5,6).

Профиль среды распределенной ИС должен включать стандарты протоколов транспортного уровня (по ISO OSI или стандарт «де-факто» протокола TCP/IP), стандарты локальных сетей (например, стандарт Ethernet IEEE 802.3 или стандарт Fast Ethernet IEEE 802.3 u), а также стандарты средств сопряжения проектируемой ИС с сетями передачи данных общего назначения (например, по рекомендациям ITU-T X.25, X.3, X.29 и др.)

Выбор аппаратных платформ ИС связан с определением требуемых их параметров: вычислительной мощности серверов и рабочих станций в соответствии с проектными решениями по разделению функций между клиентами и серверами; степени масштабируемости аппаратных платформ; надежности. Профиль среды ИС должен содержать стандарты, определяющие параметры технических средств и способы их измерения (например, стандартные тесты измерения производительности).

Профиль защиты информации в ИС должен обеспечивать реализацию политики информационной безопасности, разрабатываемой в соответствии с требуемой категорией безопасности и критериями безопасности, заданными в ТЗ на систему. Построение профиля защиты информации в распределенных системах «клиент-сервер» методически связано с точным определением компонентов системы, ответственных за те или иные функции, сервисы и услуги, и функций защиты информации, встроенных в эти компоненты. Функциональная область защиты информации включает в себя функции защиты, реализуемые разными компонентами ИС:

– функции защиты, реализуемые операционной системой;

– функции защиты от несанкционированного доступа, реализуемые на уровне программного обеспечения промежуточного слоя;

– функции управления данными, реализуемые СУБД;

– функции защиты программных средств, включая средства защиты от вирусов;

– функции защиты информации при обмене данными в распределенных системах, включая криптографические функции;

– функции администрирования средств безопасности.

Основополагающим документом в области защиты информации в распределенных системах являются рекомендации X.800, принятые МККТТ (сейчас ITU-T) в 1991 г. Под-

множество указанных рекомендаций должно составлять профиль защиты информации в ИС с учетом распределения функций защиты информации по уровням концептуальной модели ИС и взаимосвязи функций и применяемых механизмов защиты информации. При применении профиля защиты информации при проектировании, разработке и сопровождении ИС целесообразно использовать методические рекомендации, изложенные в интерпретации «Оранжевой книги» национального центра компьютерной безопасности США для сетевых конфигураций. Профиль защиты информации должен включать указания на методы и средства обнаружения в применяемых аппаратных и программных средствах, для борьбы с несанкционированным доступом и вирусами. Профиль должен также включать указания на методы и средства резервного копирования информации и восстановления информации при отказах и сбоях аппаратуры системы.

Профиль инструментальных средств, встроенных в ИС, должен отражать решения по выбору методологии и технологии создания, сопровождения и развития конкретной ИС. В этом профиле должна быть указана ссылка на описание выбранной методологии и технологии, выполненное на стадии эскизного проектирования ИС. Состав инструментальных средств, встроенных в ИС, определяется на основании решений и нормативных документов об организации сопровождения и развития ИС. При этом должны быть учтены правила и порядок, регламентирующие внесение изменений в действующие системы. Функциональная область профиля инструментальных средств, встроенных в ИС, охватывает функции централизованного управления и администрирования, связанные с:

- контролем производительности и корректности функционирования системы в целом;
- управлением конфигурацией прикладного программного обеспечения, тиражированием версий;
- управлением доступом пользователей к ресурсам системы и конфигурацией ресурсов;
- перенастройкой приложений в связи с изменениями прикладных функций ИС;
- настройкой пользовательских интерфейсов (генерация экранных форм и отчетов);
- ведением баз данных системы;
- восстановлением работоспособности системы после сбоев и аварий.

Дополнительные ресурсы, необходимые для функционирования встроенных инструментальных средств (минимальный и рекомендуемый объем оперативной памяти, размеры требуемого пространства на дисковых накопителях и т.д.), должны быть учтены в разделе проекта, относящемся к среде ИС. Выбор инструментальных средств, встроенных в ИС, должен производиться в соответствии с требованиями профиля среды ИС. Ссылки на соответствующие стандарты, входящие в профиль среды, должны быть указаны и в профиле инструментальных средств, встроенных в ИС. В этом профиле должны быть также предусмотрены ссылки на требования к средствам тестирования, которые необходимы для процессов сопровождения и развития системы и должны быть в нее встроены. В число встроенных в ИС средств тестирования должны входить средства, обеспечивающие:

- функциональное тестирование приложений;
- тестирование интерфейсов пользователя;
- системное тестирование;
- тестирование серверов и клиентов при максимальной нагрузке.

3.2. Процессы формирования, развития и применения профилей информационных систем

В общем случае созданию сложной информационной системы должна предшествовать стадия предпроектного обследования организации (объекта информатизации), для которого предполагается создавать систему. Результатами работ на этой стадии являются функциональная и информационная модели организации и спецификации требований к предполагаемой системе, которые служат в качестве исходных данных для проектирования системы. Желательно, чтобы функциональная и информационная модели и спецификации требований были выполнены с помощью формализованных методов их описания, например, с использованием средств описания моделей в известных методологиях структурного или объектно-ориентированного проектирования и языков спецификаций. В этом случае в ТЗ на создание конкретной ИС, разрабатываемое в результате стадии предпроектного обследования, должно быть указание на имеющиеся исходные данные и на средства описания исходных данных. Ссылки в ТЗ на документы, определяющие выбранные средства описания исходных данных, являются частью профиля инструментальной среды, поддерживающей основные процессы: проектирование, разработку, сопровождение и развитие прикладного программного обеспечения ИС.

В ТЗ должны быть определены требования к жизненному циклу ИС и даны ссылки на действующие нормативные документы по жизненному циклу, то есть определен профиль жизненного цикла ИС. Аналогично в ТЗ задаются требования к качеству прикладного программного обеспечения ИС и, соответственно, первичный профиль качества.

В ТЗ задаются функциональные требования к ИС (состав задач, решаемых ИС) и указываются ссылки на ведомственные нормативные документы, которые регламентируют правила и процедуры выполнения функций и операций.

При этом стадии разработки профилей, которые определяются разработчиком системы по его усмотрению, должны быть увязаны со стадиями жизненного цикла ИС, и выполняться во времени таким образом, чтобы эти разрабатываемые профили могли быть применены тогда, когда это требуется по логике детализации проекта. Исходя из выбранной модели жизненного цикла ИС и возможного влияния решений, принимаемых на какой-либо стадии проекта, на решения, принятые ранее, следует учитывать итерационный характер формирования функциональных профилей ИС и, при необходимости, корректировки ТЗ.

На стадии стратегического планирования и анализа требований уточняются исходные данные и разрабатываются спецификации требований к прикладному программному обеспечению и требований к среде. Эти спецификации должны позволять уточнить первичные функциональные профили ИС, заданные в ТЗ, дополняя их стандартами, применение которых потребуется на стадии проектирования. Такие дополнения, в частности, могут возникать в связи с принятием принципиальных решений по: структуре прикладного ПО, архитектуре среды распределенной обработки данных; распределению функций защиты информации между прикладным программным обеспечением и средой ИС для обеспечения заданной категории информационной безопасности; выбору инструментальных средств проектирования и программирования. Принимаемые на этой стадии решения исходят из альтернативного выбора методологии и принципов построения ИС между функционально-модульным подходом и объектным подходом. В плане создания ИС, разрабатываемом на этой стадии, должны быть учтены работы, связанные с построением и оформлением функциональных профилей ИС.

Стадия предварительного (эскизного) проектирования ИС связана с обоснованием и принятием принципиальных проектных решений, относящихся к каждому из четырех функциональных профилей ИС. Принятые проектные решения документируются в составе эскизного проекта ИС, в частности документируются разработанные на данной стадии проекта функциональные профили, дополняющие и конкретизирующие первичные профили, заданные в ТЗ.

Профиль прикладного ПО (функциональных частей ИС), формируемый на данной стадии, должен определять архитектуру прикладных программных комплексов (модели функций, логические модели данных, внешние интерфейсы) и их структуру (разбиение системы на подсистемы и подсистем на модули, определение унифицированных интерфейсов взаимодействия между прикладными программами). Профиль прикладного ПО конкретной ИС должен учитывать функциональную ориентацию приложений. При этом функции каждого прикладного объекта и задачи всего прикладного программного комплекса в целом, задаваемые на стадиях анализа и эскизного проектирования, не должны быть привязаны к организационной структуре подразделений или к каким-либо пользователям. Такая привязка должна выполняться динамически при задании прав доступа пользователей к ресурсам системы. Приложения, работа которых может быть связана с частыми изменениями нормативно-инструктивной базы функциональных операций, должны иметь встроенные автоматические средства перенастройки, позволяющие пользователям настраивать их без привлечения программистов. Описания блоков настроечной информации в этих случаях должны быть частью профиля прикладного ПО. Общие требования к прикладному ПО, заданные в ТЗ, должны быть конкретизированы в профиле на основе выбранной методологии и принципов построения системы (функционально-модульного или объектного подхода). Профиль прикладного ПО должен содержать ссылки на стандартизованные интерфейсы между приложениями и средой ИС, которые описываются в профилях среды ИС, защиты информации и встроенных инструментальных средств.

Стадия детального проектирования ИС связана с декомпозицией крупноблочной структуры системы на компоненты и выбором готовых компонентов (прикладных программ повторного использования, покупных программных и технических средств среды). При выборе и заказе готовых компонентов применяются функциональные профили ИС, полученные на предыдущих стадиях проекта. Применение функциональных профилей в этих случаях заключается в том, чтобы предъявить к используемым компонентам требования их соответствия стандартам применяемого профиля и сформировать требования к тестам, проверяющим это соответствие.

До начала разработки (программирования) приложений может производиться ***эталонное тестирование*** производительности серверов баз данных и серверов приложений, различных системных конфигураций операционных систем и аппаратуры с помощью имитационных программ клиентов и стандартных тестов измерения производительности. После выбора аппаратных платформ, типа СУБД и других компонентов среды ИС создаются прототипы приложений, рассчитанные на двухзвенную схему «клиент-сервер» или на трехзвенную схему с использованием мониторов транзакций. Выполняя одни и те же тесты на разных прототипах, проектировщик может уточнить и оптимизировать архитектуру проектируемой системы за счет рационального распределения функций между ее узлами. В результате окончательно определяется и оформляется профиль среды ИС, который в дальнейшем применяется при разработке приложений, комплексировании и испытаниях системы, а также при модернизации и развитии системы, связанными с заменой отдельных ее компонентов.

Применение профиля защиты информации на стадии детального проектирования ИС заключается в том, чтобы структурировать распределение функций защиты и реализующих их механизмов между компонентами системы, которые определяются при детализации ее структуры. Каждой группе функций профиля защиты информации должны отвечать конкретные компоненты системы, ответственные за выполнение этих функций. Различия в уязвимости разных компонентов по отношению к внешним и внутренним негативным воздействиям, влияющим на информационную безопасность, определяют различные требования к этим компонентам. Конкретизация требований к компонентам ИС в части защиты информации должна производиться на основе стандартов, включаемых в профиль защиты информации с их адаптацией к условиям конкретной ИС и принятой политике информационной безопасности. В части услуг и механизмов защиты при передаче информации следует применять стандарт ISO 7498-2, определяющий набор факультативных услуг и механизмов защиты по уровням эталонной модели ВОС. Конкретизацию требований к прикладным процессам по функциям аутентификации следует производить с учетом стандарта ISO 9594-8. Компоненты ИС, реализующие механизмы цифровой подписи, должны соответствовать требованиям *ГОСТ 28147-89. СОИ. Защита информации. Алгоритм криптографической подписи*.

Стадия разработки связана, прежде всего, с программированием и отладкой компонентов приложений, которые создаются заново для данной ИС. Одновременно создаются функциональные тесты для проверки выполнения приложениями заданных функций и тесты производительности приложений. Разработка приложений (прикладных программных средств) производится с помощью инструментальных средств, отвечающих требованиям выбранного ранее профиля методологии и технологии. Аппаратно-программные платформы, на которых выполняются клиентские и серверные части приложений, должны соответствовать требованиям профиля среды ИС. После детального проектирования версии прикладных программных средств, начиная со стадии разработки вплоть до стадии интеграции и тестирования комплекса прикладных программ в составе ИС, все работы должны проводиться в соответствии с требованиями функциональных профилей ИС.

На стадии интеграции и тестирования ИС применяется весь набор функциональных профилей, подготовленных на предшествующих стадиях проекта. На этой стадии производится комплексная проверка всех компонентов созданной системы: клиентских приложений, служб, выполняемых серверами, программных средств среды ИС, сетевой инфраструктуры (системное тестирование). Системное тестирование позволяет ответить на три главных вопроса: правильно ли взаимодействуют компоненты системы друг с другом, справляются ли серверы с обслуживанием заданного числа пользователей и получают ли конечные пользователи корректную информацию. Применение функциональных профилей ИС на данной стадии позволяет установить соответствие компонентов системы и всей ИС в целом требованиям этих профилей при помощи тестов соответствия.

На стадии внедрения производится перенос разработанного прикладного ПО с инструментальной платформы разработчика системы на реальную платформу ИС. При этом проверяется соответствие реальной платформы требованиям функциональных профилей ИС и функционирование прикладного ПО на реальной платформе. Стадия внедрения предполагает адаптацию и настройку ИС на реальные условия эксплуатации, для которых она создавалась. Применение функциональных профилей ИС в этих случаях

позволяет обусловить пределы изменений в системе, связанных с ее адаптацией, и границы значений параметров, в пределах которых может производиться настройка.

Приемочные испытания ИС проводятся в условиях реальной эксплуатации на соответствие требованиям ТЗ и требованиям полного профиля ИС, который был сформирован в процессе создания системы.

При сопровождении ИС важнейшее значение имеют регламенты процессов сопровождения и применение инструментальных средств, встроенных в ИС, в частности, средств управления конфигурацией. Эти регламенты рекомендуется устанавливать с использованием стандартов *ISO 687: 1983, ISO 12207:1995 и ANSI/IEEE 1042: 1987.*

Рассмотренные общие методические положения создания и применения комплекса профилей ИС следует детализировать для каждого профиля до уровня ***Руководящих указаний*** по адаптации и параметризации и ***Методик*** по применению для конкретных проблемно-ориентированных областей или конкретных проектов ИС. В этих документах должны быть представлены конкретные операции, структура и содержание документов, обеспечивающих регламентированное применение профиля и контроль соответствия процессов и объектов ИС требованиям и рекомендациям профиля. Положения каждой методики и сценарии их конкретного применения должны быть поддержаны технологией и средствами автоматизации их реализации, документирования и контроля соответствия утвержденному профилю. Для создания Руководящих указаний и Методик применения профилей при подготовке к проектированию ИС следует утвердить их номенклатуру и требования к содержанию. Дальнейшие работы следует организовать с целью создания конкретных методик применения каждого профиля с тесным взаимодействием с подразделениями, которым предстоит использовать соответствующие профили. Такие методики должны предусматривать длительное развитие, расширение и модернизацию функций компонентов и ИС в целом.

Вопросы для самопроверки

1. Дайте определение понятия «профиль информационной системы».
2. Какие виды профилей существуют?
3. Каковы цели и принципы формирования профилей информационных систем?
4. Как жизненный цикл конкретной ИС в соответствии с основными процессами создания, сопровождения и развития ИС должен быть поддержан этапами развития и применения комплекта профилей?
5. Каковы структура и содержание профилей информационных систем?
6. Назовите основные этапы процессы формирования, развития и применения профилей информационных систем.

4. Методология построения профилей информационных систем

В этой главе рассматриваются принципы, методы и стандарты, обеспечивающие регламентированное, упорядоченное создание и применение профилей сложных информационных систем (ИС). Объектами функциональной стандартизации являются архитектура и структура сложных ИС, а также процессы жизненного цикла ИС. Положения по применению профилей ИС ориентированы на достижение высокого качества, надежности и безопасности функциональных компонентов и ИС в целом.

При создании и развитии сложных, распределенных, тиражируемых ИС требуется гибкое формирование и применение гармонизированных совокупностей базовых стандартов и нормативных документов разного уровня, выделение в них требований и рекомендаций, необходимых для реализации заданных функций ИС. Для унификации и регламентирования реализации заданных функций ИС такие совокупности базовых стандартов должны адаптироваться и конкретизироваться применительно к определенным классам проектов, функций, процессов и компонентов ИС. В связи с этой потребностью выделилось и сформировалось понятие «профилей» ИС, как основного инструмента функциональной стандартизации.

В профиле выделяются и устанавливаются допустимые факультативные возможности и значения параметров каждого базового стандарта и/или нормативного документа, входящего в профиль. Профиль не может противоречить использованным в нем базовым стандартам и нормативным документам. Он должен использовать выбранные из альтернативных вариантов факультативные возможности и значения параметров в пределах допустимых. На базе одной и той же совокупности базовых стандартов могут формироваться и утверждаться различные профили для разных проектов ИС и сфер применения. Эти ограничения базовых документов профиля и их гармонизация, проведенная разработчиками профиля, должны обеспечивать качество, совместимость и корректное взаимодействие компонентов системы, соответствующих профилю, в заданной области применения профиля.

Базовые стандарты ИТ и профили ИС в зависимости от проблемно-ориентированной области применения ИС могут использоваться как непосредственные директивные, руководящие или как рекомендательные документы, а также как нормативная база, используемая при выборе или разработке средств автоматизации технологических этапов или процессов создания, сопровождения и развития ИС.

Следует рассматривать ***две группы*** профилей ИС:

– профили, регламентирующие архитектуру и структуру ИС и ее компонентов (функции, интерфейсы и протоколы взаимодействия, форматы данных и т.д.);

– профили, регламентирующие процессы проектирования, разработки, применения, сопровождения и развития ИС и их компонентов.

В зависимости от области распространения профилей они могут иметь разные категории и соответственно разные статусы утверждения:

– профили конкретной ИС, определяющие стандартизованные проектные решения в пределах данного проекта и являющиеся частью проектной документации;

– профили ИС, предназначенные для решения некоторого класса прикладных задач, которые распространяются на все ИС данного класса в пределах предприятия, отрасли или региона и утверждаются как стандарты предприятий, ведомственные или государственные стандарты.

Профили ИС унифицируют и регламентируют только часть требований, характеристик, показателей качества объектов и процессов, выделенных и формализованных на базе стандартов и нормативных документов. Другая часть функциональных и технических характеристик ИС определяется заказчиками и разработчиками творчески, без учета положений нормативных документов.

Методология построения открытых ИС, обеспечивающая возможности расширения функций систем, взаимодействия систем, мобильности программ, данных и персонала, неразрывно связана с применением стандартов информационных технологий (ИТ). Применение при создании открытых ИС стандартизованных проектных решений, обеспечивающих реализацию заданной функции или группы функций, позволяет получать лучшие технико-экономические показатели проектов ИС по сравнению с проектированием ИС «с нуля».

Общие положения функциональной стандартизации в области информационных технологий определены стандартом ГОСТ Р ИСО/МЭК ТО 10000-99 «Информационная технология. Основы и таксономия международных функциональных стандартов». В этом документе введено важнейшее понятие – «профили», которые определяются как подмножества или комбинации базовых стандартов ИТ, необходимые для реализации конкретных функций информационных систем или их компонентов. Профили, разработанные применительно к определенным областям использования ИТ и конкретным наборам функций ИС согласно ГОСТ Р ИСО/МЭК 10000-99, приобретают статус функциональных стандартов (стандартов предприятия, региональных, отраслевых/ведомственных, национальных или международных) после их утверждения в установленном порядке.

Технические и программные средства, соответствующие базовым стандартам ИТ, используются при создании информационных систем для различных областей применения. При этом требования прикладных задач каждой области применения диктуют необходимость конкретизации факультативных возможностей и параметров, предусмотренных в базовых стандартах, а также выбор совокупностей базовых стандартов, которым должны отвечать информационные системы данной области.

Методы создания и развитие типовых, унифицированных, тиражируемых ИС, направленные на снижение затрат и сокращение сроков создания ИС в условиях роста их сложности и наращивания функций, требуют нового подхода к стандартизации. Построение профилей ИС и принятие соответствующих им функциональных стандартов является одним из основных способов решения этой задачи.

Процессы создания и модернизации ИС и процессы функциональной стандартизации ИС развиваются параллельно. Они взаимосвязаны и взаимообусловлены.

Анализ архитектуры и структуры существующих ИС должен позволить определить первоочередную номенклатуру профилей ИС. Прежде всего, такой анализ, предшествующий разработке профилей, требуется для построения профилей типовых, унифицированных, тиражируемых ИС.

Разработанные профили ИС должны учитываться при формировании технических заданий на создание или модернизацию и развитие ИС, определяя нормативные требования к этим системам, в частности: функциональные требования, общие и частные технические требования, а также требования к сопровождению систем в эксплуатации. Профиль ИС должен составлять нормативную базу проектных решений на всех стадиях жизненного цикла конкретной системы. При принятии решений о расширении состава задач, решаемых ИС, модернизации аппаратно-программных платформ ИС, должна производиться проверка соответствия этих решений требованиям профиля ИС. Одновременно должна

производиться актуализация профиля, учитывающая принятие новых стандартов ИТ и вносимые в них изменения.

Планы разработки профилей ИС и сроки введения функциональных стандартов, относящихся к профилям ИС, должны учитываться при планировании создания, развития и модернизации ИС.

При построении профилей ИС необходимо учитывать определенную противоречивость условий, в которых происходят процессы стандартизации ИТ. С одной стороны, темпы улучшения характеристик и совершенствования функциональных возможностей средств вычислительной техники и телекоммуникаций таковы, что длительность жизненного цикла сложных информационных систем превышает сроки морального и физического старения аппаратных и программных средств, на базе которых строятся эти системы. С другой стороны, стандарты, регламентирующие требования к этим средствам, имеют естественную консервативность, закрепляя указанные требования на определенный период времени. Поэтому определение номенклатуры профилей ИС, подлежащих разработке, связано с оценками состояния стандартизации ИТ (наличия базовых стандартов для построения профилей) и с выбором таких решений построения ИС, которые обеспечивают возможность замены отдельных их компонентов, не затрагивая другие компоненты.

Профили ИС должны играть роль инструмента, позволяющего проектировать конкретные ИС на основе стандартизованных компонентов, составлять основу для разработки или выбора тестов соответствия ИС требованиям профиля, проводить сопровождение и модернизацию ИС, имея строго очерченные границы допустимых изменений их архитектуры и структуры. Здесь уместно вспомнить о назначении, категориях и видах профилей, а также об их структуре и содержании.

4.1. Порядок разработки профилей информационных систем

Разработке каждого профиля ИС должен предшествовать предварительный анализ, включающий оценку состояния стандартизации и сведений об архитектуре и структуре имеющихся ИС, на основании которых можно выбрать концептуальную модель ИС данного класса.

Выбранная концептуальная модель ИС должна использоваться при разработке профиля для определения места компонентов каждой функциональной области профиля на модели, их интерфейсов и протоколов взаимодействия.

Результаты предварительного анализа, номенклатура базовых стандартов ИТ и уже существующие профили ИС должны составлять основу для выбора базовых стандартов при формировании профилей ИС, подлежащих разработке.

В зависимости от категории разрабатываемого профиля этапы разработки профиля, рассматриваемые ниже, допускается объединять. При появлении новых базовых стандартов ИТ или внесении изменений в действующие базовые стандарты, необходимо проводить актуализацию разработанного профиля с обязательным выполнением стадий гармонизации базовых стандартов и формирования требований соответствия профилю.

Содержание и результаты работ по этапам разработки профиля ИС рассмотрены ниже.

4.1.1. Определение прикладных задач, решаемых информационной системой

На этом этапе производится выделение класса прикладных задач и определяется состав прикладных функций ИС, для которой строится профиль. Описание набора прикладных функций должно входить в описание профиля и служить основой для последую-

щего использования профиля при реализации выбранных проектных решений, тиражировании и модернизации систем. Кроме того, этот набор функций используется при параметризации компонентов среды ИС. Название рассматриваемого класса прикладных задач должно входить в название профиля.

Результатом работы на данном этапе должно быть перечисление прикладных задач (с указанием их характеристик), решаемых ИС, для которой строится профиль.

4.1.2. Выбор концептуальной модели среды информационной системы

Как указывалось выше, разработке профиля должен предшествовать анализ архитектуры и структуры действующих систем, т.е. в основу разработки профиля должны быть положены выбранные архитектурные решения для систем данного класса. На базе этих решений выбирается концептуальная модель ИС, предназначенной для решения заданного класса задач.

Результатом работ на данном этапе является структура среды ИС, включая определение компонентов среды, реализующих выбранные сервисы и услуги среды.

Кроме того, на этом же этапе в соответствии с ГОСТ Р ИСО/МЭК 10000-1 должен быть разработан «сценарий профиля», иллюстрирующий место разрабатываемого профиля в концептуальной модели.

Сценарий профиля должен показывать в упрощенном графическом виде компоненты ИС, охватываемые данным профилем, и их взаимодействие с другими компонентами в рамках выбранной концептуальной модели ИС.

4.1.3. Параметризация компонентов среды информационной системы

На данном этапе, с учетом номенклатуры базовых стандартов ИТ, а также положений, изложенных в п. 4.2.2, определяются перечни параметров для каждого компонента среды. Эти параметры включают в себя:

− функциональные параметры, определяющие состав сервисов и услуг, предоставляемых данным компонентом;

− интерфейсные параметры, определяющие характеристики взаимодействия данного компонента с другими компонентами среды и приложениями.

Для каждого параметра определяется его значение с учетом выбранной архитектуры среды и требований прикладных задач.

Результатом являются перечни параметров и их значения, которые будут использоваться на дальнейших этапах разработки профиля.

4.1.4. Наполнение профиля базовыми стандартами информационных технологий

На этом этапе осуществляется выбор базовых стандартов ИТ для построения профиля, который должен производиться из числа стандартов, включенных в номенклатуру базовых стандартов ИТ. Для каждого компонента (группы компонентов) из перечня выбираются стандарты де-юре и де-факто, определяющие номенклатуру функций и значения параметров компонентов. Далее, при необходимости, определяется номенклатура параметров, не регламентированных стандартами. Вместо отсутствующих стандартов готовятся спецификации на основе результатов по п. 4.2.1.

Для построения профиля ИС следует использовать стандарты со статусом международных или ГОСТ Р. Однако, для многих функциональных областей открытых систем такие стандарты отсутствуют. В таких случаях допускается включение в профиль специ-

фикаций, частично охватывающих заданный набор функций, при условии, что такие спецификации являются общедоступными.

При выборе стандартов и спецификаций для каждой функциональной области рекомендуется оценивать их с точки зрения:

– степени согласованности, оцениваемой числом организаций-пользователей и поставщиков средств, которые руководствуются данным международным стандартом или приняли согласованные спецификации стандарта «де-факто»;

– доступности изделий (аппаратных и программных средств), соответствующих требованиям данного стандарта или спецификации, на рынке России;

– полноты, т.е. по возможности полного охвата наборов функций в каждой функциональной области профиля ИС;

– зрелости, т.е. отсутствия ожидаемых изменений в стандарте или спецификации;

– фактического использования данного стандарта;

– проблем или ограничений при использовании спецификаций, например, совместимости с их предыдущими версиями, а также отсутствия лицензионных ограничений на их использование.

Для каждого базового стандарта ИТ должна быть разработана спецификация его применения, включающая рекомендации по выбору номенклатуры параметров и их значений.

Результатом данного этапа являются наборы базовых стандартов ИТ, составляющие функциональные и технологические профили ИС.

4.1.5. Уточнение концептуальной модели и параметров компонентов

На данном этапе производится уточнение концептуальной модели, а также номенклатуры и значений параметров компонентов с учетом результатов, полученных на предыдущих этапах.

Результатом этого этапа являются:

– изображенная в графическом виде уточненная концептуальная модель;

– таблицы с параметрами компонентов и их значениями;

– перечни базовых стандартов, структурированные по функциональным областям профиля с учетом уточнения концептуальной модели.

4.1.6. Гармонизация базовых стандартов

На данном этапе производятся:

– устранение противоречий и уточнение альтернативных возможностей в выбранном комплексе базовых стандартов ИТ;

– устранение избыточности требований базовых стандартов с точки зрения описания компонентов ИС, для которой разрабатывается профиль.

Результатом этого этапа являются ограничительные спецификации базовых стандартов ИТ, обеспечивающие совместимость компонентов ИС, отвечающих данному профилю.

4.1.7. Формирование требований соответствия информационной системы профилю

На этом этапе производится разработка требований, которым должны отвечать системы, подлежащие проверке на соответствие профилю, и ссылок на соответствующие методы тестирования и тесты (включая тесты для всего профиля и тесты соответствия базовым стандартам ИТ).

4.1.8. Оформление профилей информационной системы

Разработка профилей ИС завершается их оформлением в виде документов, соответствующих установленным требованиям.

4.2. Согласование и утверждение профилей информационной системы

При создании конкретных ИС рекомендуется сопровождать принятые проектные решения разработкой соответствующих профилей, оформляя их в составе документации проекта ИС.

Разработка функциональных профилей конкретной ИС должна планироваться в составе работ на стадиях жизненного цикла ИС в соответствии с правилами, которые регламентируются нормативными документами жизненного цикла ИС. Если на момент принятия решения о разработке профиля имеются стандарты или спецификации, относящиеся к функциям ИС или компонентов ИС, для выполнения которых разрабатывается профиль, то выбор базовых стандартов производится из имеющейся номенклатуры базовых стандартов (подлежит разработке на втором этапе данной темы). Если требуемые стандарты или спецификации отсутствуют, то необходимо принять решение о разработке в рамках данного проекта ИС спецификаций, заменяющих отсутствующие нормативные документы.

Решения о разработке и применении профилей конкретной ИС принимает лицо, отвечающее за проект ИС.

Разработка, согласование и утверждение профилей ИС, имеющих статус нормативных документов предприятия или администрации региона, проводятся на основании ГОСТ Р 1.4-93.

Разработка, согласование и утверждение профилей ИС, которые должны иметь статус ГОСТ Р, проводятся в соответствии с ГОСТ Р 1.2-92.

Вопросы для самопроверки

1. Каким документом регламентируются общие положения функциональной стандартизации в области информационных технологий?
2. Каков порядок разработки и согласования профилей ИС?
3. Что является результатом работы по созданию профиля на этапе определение прикладных задач, решаемых ИС?
4. Что является результатом работы по созданию профиля на этапе выбора концептуальной модели среды ИС?
5. Что является результатом работы по созданию профиля на этапе параметризации компонентов среды ИС?
6. Что является результатом работы по созданию профиля на этапе наполнения профиля базовыми стандартами ИТ?
7. Что является результатом работы по созданию профиля на этапе гармонизации базовых стандартов?
8. Каким документом регламентируются порядок утверждения профилей ИС?

5. Объекты стандартизации в функциональных профилях информационных систем и источники базовых стандартов информационных технологий

5.1. Исходные положения

Для построения функциональных профилей ИС необходимо задать набор функций, которые должны выполняться ИС или компонентами ИС.

Согласно стандарту ГОСТ Р ИСО/МЭК ТО 10000-3-99 «Информационная технология. Основы и таксономия международных стандартизованных профилей. Часть 3: Принципы и таксономия профилей среды открытых систем», опирающуюся на эталонную модель среды открытых систем OSE/RM, информационные системы разделяются на приложения (прикладные программы) и среду (платформы прикладных программ), в которой функционируют эти приложения. Между ними определяются стандартные программные интерфейсы – API. Кроме API определяются стандартизованные интерфейсы между ИС и внешней для нее средой – EEI.

Как было сказано выше, структура профилей ИС связана с декомпозицией ИС на составные части таким образом, что номенклатура профилей должна соответствовать иерархии структурного разбиения системы на крупные функциональные части (подсистемы). С этой точки зрения методология построения профилей должна учитывать два основных принципа:

– соответствие прикладных функций, реализуемых каждой подсистемой, разделению набора функций системы между подсистемами, которое определяется при декомпозиции;

– соответствие состава профилей принятой концептуальной модели ИС.

Для определенности ниже рассматриваются группы функций применительно к концептуальной модели систем распределенной обработки данных с архитектурой «клиент-сервер». Этот подход связан с тем, что такая архитектура представляется одним из основных направлений создания современных ИС. С другой стороны, методология построения профилей ИС, предложенная для систем с архитектурой «клиент-сервер», может быть применена и для более простых случаев, в частности, при более детальной проработке профилей компонентов, входящих в крупные функциональные узлы системы.

Концептуальная модель ИС с распределенной обработкой данных показана на рис. 5.1. в виде матрицы основных функций и структурных компонентов ИС, реализующих эти функции. Каждый элемент матрицы содержит программные и/или аппаратные средства, ответственные за выполнение названных для них функций.

Формирование профилей ИС на основании данной концептуальной модели заключается, прежде всего, в том, чтобы указать наборы необходимых функций для каждого из четырех горизонтальных уровней модели:

– функциональных частей ИС (приложений);

– среды распределенной обработки данных;

– операционных систем клиентов и серверов;

– технических средств, составляющих аппаратуру станций клиентов и серверов.

Функциональные области, представленных в четырех вертикальных столбцах матрицы концептуальной модели ИС составляют:

– функции человеко-машинного интерфейса;

– функции организации процессов обработки данных;

– функции управления данными и обмена данными;
– коммуникационные функции.

Таким образом, каждый элемент матрицы концептуальной модели определяет набор функций, необходимых для построения профиля (или группы профилей) программных или аппаратных компонентов системы. Кроме набора функций эти профили должны описывать интерфейсы взаимодействия соответствующих компонентов системы как с другими компонентами того же горизонтального уровня модели, так и с компонентами выше - и нижележащего уровней.

Такой же подход может быть применен при необходимости для дальнейшей детализации структуры компонентов ИС, относящихся к указанным выше элементам матрицы, на модули, имеющие стандартизованные описания.

После проведенной декомпозиции и определения профилей для компонентов системы необходимо выполнить обратный процесс, состоящий в объединении полученных профилей компонентов в профили функциональных узлов и в профили ИС в целом.

Функциональные части ИС (приложения)			
Регламенты действий пользователей	Наборы прикладных функций	Функции ведения архивов ИС Функции документооборота	Форматы электронных сообщений
Среда распределенной обработки данных			
Оболочки интерфейсов пользователя	Мониторы транзакций	Распределенные СУБД	Услуги телекоммуникационной среды прикладного уровня
Операционные системы клиентов и серверов			
Команды ОС и утилиты. Драйверы ввода/вывода	Ядра	Файловые системы	Сетевые протоколы транспортного уровня
Технические средства			
АРМ пользователя	Серверы приложений Серверы обработки транзакций	Серверы баз данных	Телекоммуникационные серверы Средства локальной сети

Рис. 5.1. Концептуальная модель ИС с распределенной обработкой данных

5.2. Объекты стандартизации в профилях приложений ИС

Задание наборов функций для профилей приложений ИС должно производиться на основе декомпозиции заданных прикладных функций ИС на крупные функциональные части ИС (подсистемы).

Профили приложений ИС должны содержать нормативные документы, регламентирующие правовые и организационные аспекты области деятельности организации, для которой создается ИС. Такими документами являются законодательные и распорядительные акты, распространяющиеся на эту область деятельности и подлежащие отражению в прикладном ПО ИС, классификаторы и справочники объектов.

Объектами стандартизации в профилях приложений являются программные интерфейсы между приложениями, принадлежащими разным подсистемам ИС, и протоколы взаимодействия подсистем.

Объекты стандартизации для реализации основных функций прикладного ПО:

- форматы электронных сообщений (электронного обмена данными);
- регламенты действий пользователей при решении прикладных задач;
- регламенты выполнения прикладных задач и управления этими процессами;
- организация хранения данных и документов, например, архивов ИС, документооборота.

5.3. Объекты стандартизации в профилях среды распределенной обработки данных

5.3.1. Объекты стандартизации в профилях компонентов сервисных служб среды ИС

Эти профили должны определять сервисы и услуги, предоставляемые приложениям со стороны среды. На этом уровне отражается основная доля стандартизованных интерфейсов между приложениями и средой (API).

Сервисы и услуги среды распределенной обработки данных реализуются с помощью программного обеспечения промежуточного слоя (middleware), размещенных в концептуальной модели ИС между уровнем приложений ИС и уровнем операционных систем (см. рис. 5.1.). Существуют три категории ПО промежуточного слоя:

- ПО, ориентированное на обеспечение конкретных приложений;
- ПО обмена информацией;
- ПО управления и поддержки.

Первую категорию составляют программные средства промежуточного слоя, обеспечивающие функции серверов приложений, серверов обработки транзакций, серверов баз данных, серверов обработки сообщений и серверов Web-технологий. В зависимости от того, какие проектные решения приняты для среды распределенной обработки данных, возможно применение ПО промежуточного слоя с архитектурой DCE (Distributed Computing Environment), ориентированной на вызов удаленных процедур RPC (Remote Procedure Call), архитектурой CORBA (Common Object Request Broker Architecture) и с архитектурой Web.

Объектами стандартизации в профилях среды распределенной обработки данных являются следующие.

- Среда распределенной обработки данных. Общие вопросы.
- Оболочки и утилиты пользовательского интерфейса.
- Архитектура DCE.
- Мониторы обработки транзакций.
- Архитектура CORBA.
- Брокеры объектных запросов.
- Архитектура распределенных хранилищ данных (Data Warehouse).
- Распределенные СУБД.
- Службы обмена сообщениями и обработки сообщений, в т.ч. услуги электронной почты, услуги передачи файлов.
- Услуги Web.

5.3.2. Объекты стандартизации в профилях операционных систем

Уровень операционных систем клиентов и серверов связан с поддержкой следующих групп функций.

Функции ядра операционной системы:

- создания процессов и управления процессами, исполнения программ;
- генерации и передачи сигналов операционной системы;
- генерации и обработки сигналов системного времени;
- управления файловой системой и каталогами;
- обработки запросов ввода-вывода и обслуживания внешних устройств.

Функции поддержки пользовательского интерфейса реализуются основными командами языка управления заданиями и сервисными программами (утилитами), которые обеспечивают:

- механизмы для исполнения функций уровня оператора, такие, как сравнение файлов, вывод на печатающее устройство и отображение содержимого файлов на экран, редактирование файлов;
- регистрацию сообщений;
- перемещение файлов из каталога в каталог;
- сортировку данных;
- исполнение командных строк;
- доступ к служебной информации ОС.

Данная область охватывает методы и средства, с помощью которых пользователи ИС могут взаимодействовать с прикладными программами (прикладными системами). В интерфейсах этого типа определяются следующие функции:

Операции типа «клиент-сервер». Они определяют взаимоотношения в сети между процессами «клиента» и «сервера», например, обслуживающего запрос пользователя.

Определение объектов и управление. Это спецификации задания характеристик отображаемых на экране элементов, например, цвета, формы, размеров, перемещения, графических характеристик изображения, взаимодействия между отдельными элементами изображения.

Параметры окон. Задают спецификации, которые позволяют определять, как окна создаются, передвигаются, сохраняются, восстанавливаются, удаляются и взаимодействуют друг с другом.

Поддержка диалога. Эти функции описываются спецификациями, с помощью которых устанавливаются взаимоотношения между тем, что отображено на экране (в т.ч. движение курсора, данные, введенные с клавиатуры и дополнительных устройств), и тем, как меняется изображение в зависимости от вводимых данных.

Спецификации мультимедиа. Они включают в себя:

- API-спецификации определения форматов данных и функций, которые поддерживают манипуляции информацией, представленной в различных формах (цифровой и аудиовизуальной), и объединяют текст, звук и видеоизображения в одном сеансе;
- функции аналого-дискретного преобразования, сжатия и запоминания крупных массивов данных;
- функции синхронизации представлений информации, зависящих от времени;
- функции многоканального ввода и вывода информации, представленной в различных формах.

Функции расширения реального времени. К ним относятся функции, реализующие прикладные и системные интерфейсы, которые используются приложениями, требующими детерминированного во времени исполнения, обработки и реакции.

Функции управления системой. Они включают в себя:

– функции создания и управления ресурсами, предоставляемыми пользователю (приложению);

– функции управления правами доступа к ресурсам;

– функции управления конфигурацией и производительностью;

– функции административного управления системой;

– функции авторизации доступа к ресурсам;

– функции поддержки живучести системы;

– функции учета выполняемых системой заданий и использования ресурсов.

Функции файловой системы ОС.

Функции поддержки сетевых протоколов транспортного уровня (например, протокола TCP/IP).

Объектами стандартизации на этом уровне являются:

– ОС типа Unix (стандарты POSIX);

– ОС типа Windows (спецификации фирмы Microsoft).

5.3.3. Объекты стандартизации в профилях технических средств ИС

Профили автоматизированных рабочих мест (АРМ) и профили серверов определяются, прежде всего, составом функций, которые задаются при декомпозиции ИС на функциональные узлы. Реализация этих заданных функций связана с требуемыми параметрами и конфигурациями аппаратуры. Они должны обеспечивать поддержку операционных систем, функции которых указаны выше.

При необходимости состав АРМ и серверов может быть детализирован до устройств, обладающих стандартизованными интерфейсами. Для определения этих интерфейсов может применяться та же концептуальная модель применительно к одномашинной конфигурации.

Объектами стандартизации в профилях технических средств ИС являются следующие:

– системные интерфейсы персональных компьютеров АРМ пользователей;

– системные интерфейсы аппаратуры серверов;

– протоколы и интерфейсы локальных сетей (уровней 1 и 2 эталонной модели взаимодействия открытых систем OSI/RM), в частности сетей типа Ethernet;

– требования к системам бесперебойного электропитания;

– эргономические и санитарно-гигиенические требования к устройствам отображения информации;

– требования электробезопасности;

– требования пожарной безопасности.

Кроме того, профили технических средств должны включать в себя стандартизованные общие технические требования.

5.3.4. Объекты стандартизации в профилях телекоммуникационной среды

Функции обработки транзакций данных (в соответствии с прикладным уровнем эталонной модели взаимосвязи открытых систем OSI/RM).

Функции доступа к файлам, расположенным в любом узле неоднородной сети (FTAM) и передачи файлов (FTP).

Функции вызова удаленных процедур, расположенных в узлах распределенной среды (RPC).

Функции передачи сообщений (электронная почта).

Функции службы каталогов.

Функции сетевой поддержки гетерогенных платформ, базирующихся на различных операционных системах.

Протоколы и интерфейсы сетей Интернет/Интранет.

5.3.5. Объекты стандартизации в профилях администрирования

Функциональная область профилей администрирования определяется в концептуальной модели проекцией плоскости основных функций на плоскость функций системного и сетевого администрирования. На этой плоскости выделяются следующие объекты стандартизации:

- Общие регламенты системного администрирования.
- Службы системных и сетевых каталогов.
- Протоколы доступа к каталогам.

5.3.6. Объекты стандартизации в профилях защиты информации

В связи с необходимостью комплексного подхода к обеспечению защиты информации и с разными вариантами распределения функций защиты информации между приложениями и средой, профиль защиты информации рассматривается как отдельный функциональный профиль ИС. Комплексный подход предполагает учет таких свойств информационной безопасности ИС, как:

- доступность информационных ресурсов;
- обеспечение целостности программ и данных;
- обеспечение защиты от несанкционированного доступа.

Показатели защищенности и правила проверки соответствия ИС требованиям безопасности должны определяться на основе соответствующих руководящих документов Гостехкомиссии России и ФАПСИ.

Для определения функциональной области защиты информации, необходимой при построении профиля, предлагается использовать ту же концептуальную модель ИС, имея в виду проекцию матрицы элементов основных функций на плоскость функций защиты информации. Этот подход позволяет определить распределение функций защиты информации между компонентами среды и приложений.

Можно выделить следующие объекты стандартизации в профилях защиты информации:

Функции защиты операционной системы. Сюда относятся функции управления доступом к системным данным, функциям, аппаратным и программным ресурсам со стороны пользователя и процессов пользователя.

Функции защиты пользовательского интерфейса. Включают в себя:

– задание и реализацию методов доступа к объектам в пределах функциональной области человеко-машинного интерфейса, например, доступа к окнам и меню;

– функции защиты административного управления пользовательским интерфейсом, например, управления правами доступа пользователей к ресурсам;

– защитная маркировка информации на устройствах ввода и вывода информации.

Функции защиты управления данными:

– средства контроля и управления доступом и целостностью данных в СУБД, например, привилегии доступа, представления о базе данных, доступные отдельному пользователю или прикладной программе, условия и операторы контроля, способы верификации содержимого базы данных.

Функции защиты целостности программных средств ИС:

– средства управления доступом и целостностью программных объектов типа библиотек, программных кодов и т.д.;

– средства защиты от вирусов;

– средства защиты встроенных в ИС компонентов, поддерживающих сопровождение программного обеспечения.

Функции защиты обмена данными в распределенных системах:

– функции аутентификации партнеров по обмену и аутентификации источника данных;

– функции управления доступом (защита от несанкционированного использования ресурсов, доступных в сети);

– функции обеспечения конфиденциальности данных, в том числе: защита пользовательской информации при обмене в режимах с установлением соединения и без установления соединения, конфиденциальность отдельных полей данных (избирательная конфиденциальность), конфиденциальность трафика в сети;

– функции обеспечения целостности при передаче данных;

– функции обеспечения неотказуемости (услуги, обеспечивающие невозможность отказа от совершенных действий): неотказуемости с подтверждением подлинности источника данных и неотказуемость с подтверждением доставки.

Для реализации функций защиты информации должны применяться стандарты, регламентирующие следующие механизмы защиты или их комбинации:

– механизмы шифрования, симметричного с секретным ключом или асимметричного с открытым ключом;

– механизмы электронной подписи;

– механизмы управления доступом;

– механизмы контроля целостности данных;

– механизмы аутентификации;

– механизмы дополнения трафика;

– механизмы управления маршрутизацией;

– механизмы нотаризации.

Администрирование средств защиты информации.

Функции администратора средств защиты информации включают в себя:

− администрирование системы в целом, поддержка принятой политики безопасности, взаимодействия с другими административными службами, аудит, безопасное восстановление системы;

− администрирование функций защиты;

− администрирование механизмов защиты: управление ключами, управление шифрованием, управление доступом (например, пароли, списки доступа), управление аутентификацией, управление дополнением трафика (например, правила, задающие характеристики дополняющих сообщений – частоту отправки, размер), управление маршрутизацией (выделение надежных путей), управление нотаризацией.

Основой администрирования средств защиты информации является база данных управления безопасностью.

5.3.7. Объекты стандартизации в профилях средств поддержки создания, сопровождения и развития программного обеспечения информационных систем

Состав встроенных в ИС инструментальных средств должен быть подмножеством средств инфраструктуры проекта ИС, применяемых при её создании.

Профили этих объектов описывают инструментальные средства, встраиваемые в ИС и работающие в среде ИС. Функциональная область этих профилей является проекцией матрицы основных функций на плоскость встроенных инструментальных средств.

К объектам стандартизации в профилях средств поддержки создания, сопровождения и развития программного обеспечения информационных систем относятся:

Стандартные языки программирования и среда поддержки прикладного ПО (отладчики, средства настройки и оптимизации программного кода, редакторы).

Генераторы интерфейсов, в том числе, средства генерации пользовательского интерфейса.

Средства поддержки проектирования и ведения баз данных, в состав которых входят***:***

− средства внесения изменений в схемы баз данных и изменения отдельных элементов баз данных;

− средства регистрации вносимых изменений.

Средства управления конфигурацией (например, при сопровождении ПО).

Средства верификации и валидации прикладного ПО (тестирования версий ПО в эксплуатируемой системе).

5.4. Источники базовых стандартов для функциональных профилей информационных систем

Для формирования профилей среды ИС следует использовать следующие источники базовых стандартов ИТ и спецификаций (организации по стандартизации приведены в Приложении 1.):

ISO/IEC JTC 1 – Объединенный технический комитет 1 «Информационная технология» ISO и IEC.

Результаты работы ISO/IEC JTC 1 находятся в каталоге стандартов ISO под рубриками:

- 35.100. Information technology. Open System Interconnection (OSI);
- 35.110. Information technology. Networking (LAN, MAN, WAN, ISDN, Private ISN;
- 35.140. Information technology. Computer Graphics;
- 35.200. Information technology. Interface and interconnection equipment.

ISO TC 46 – Технический комитет «Информация и документация»

ISO TC 154 – Технический комитет. «Процессы, элементы данных и документы в коммерции».

ITU-T. Рекомендации международного телекоммуникационного союза: G-series, H-series, I-series, Q-series, V-series, X-series, Z-series.

IEEE LMSC (LAN/MAN Standards Committee) – комитет IEEE по стандартизации локальных сетей с рабочими группами P802.1, P802.3, P802.4, P802.5, P802.6, P802.9, P802.11, P802.12, P802.14, P802.7, P802.8, P802.10.

IETF (Internet Engineering Task Force) – рабочая группа по протоколам сети Интернет.

W3C (World Wide Web Consortium) – консорциум по стандартам Web-технологий.

Open Group – консорциум по стандартам среды распределенной обработки данных с архитектурой DCE.

OMG (Object Management Group) – консорциум по стандартам среды распределенной обработки данных с архитектурой CORBA.

MDC (Meta Data Coalition) – консорциум по стандартам метаданных.

Для формирования профилей приложений ИС в части нормативно-правовых документов федерального и регионального уровней могут быть использованы информационно-справочные системы «Консультант-Плюс», «Гарант» и «Кодекс».

Вопросы для самопроверки

1. Что является объектом стандартизации в профилях приложений?
2. Назовите три категории программного обеспечения промежуточного слоя среды ОИС.
3. Назовите объекты стандартизации в профилях среды распределенной обработки данных.
4. Назовите объекты стандартизации в профилях компонентов сервисных служб среды ОИС.
5. Назовите объекты стандартизации в профилях операционных систем.
6. Назовите объекты стандартизации в профилях технических средств ИС.
7. Назовите объекты стандартизации в профилях телекоммуникационной среды.
8. Назовите объекты стандартизации в профилях администрирования.
9. Назовите объекты стандартизации в профилях защиты информации
10. Назовите объекты стандартизации в профилях средств поддержки создания, сопровождения и развития программного обеспечения ИС.

6. Компонентная разработка приложений

Модульная структура прикладных программных комплексов информационных систем является одним из основных способов обеспечения свойств открытых систем, рассмотренных в главе 1. В процессе проектирования системы заданный состав ее прикладных функций декомпозируется в виде функциональных подсистем, объединяющих родственные группы функций, подсистемы разбиваются на взаимодействующие между собой задачи и комплексы задач, программы, реализующие каждую из задач, разбиваются на программные модули вплоть до простейших неделимых элементов программной системы.

Результатом процесса проектирования ИС является ее иерархическая структура, представленная в виде программных модулей, которые подлежат программированию или выбору из состава уже существующих для включения их в создаваемую систему. Применяемые технологии и инструментальные средства анализа и проектирования, поддерживающие как структурную, так и объектную парадигмы, рассматриваются в других учебных курсах. В настоящем курсе мы рассматриваем их применение для компонентной разработки приложений, которая получила развитие в последние годы. С точки зрения реализации отличие компонентов от других типов программных модулей состоит в том, что их можно модифицировать в процессе разработки на уровне двоичных исполняемых кодов, тогда как библиотеки, подпрограммы и другие модули необходимо изменять на уровне исходных кодов (с соответствующей перекомпиляцией).

Первоначально компонентная разработка приложений была связана с использованием стандарта Open Doc, ориентированного на создание составных документов и их обработку с помощью настольных систем. Затем потребность в разработке корпоративных приложений, включающих распределенные серверные и клиентские компоненты, привела к созданию интегрированных сред разработки и исполнения распределенных компонентов (*distributed component platform* – DCP). Сейчас на рынке платформ DCP лидируют продукты, поддерживающие модель DCOM (*distributed component object model*) или ActiveX DCOM (см. ниже) фирмы Microsoft или спецификацию Java Beans (основного конкурента DCOM) фирмы Sun Microsystems.

Кроме того, ряд стандартов компонентной разработки приложений на основе архитектуры брокера объектных запросов CORBA предложен консорциумом Object Management Group (OMG). Стандарты компонентной разработки Web – приложений предложены Консорциумом World Wide Web Consortium в составе стандартов Интернет.

В настоящее время предлагаются также компонентные инфраструктуры (*component framework*), которые расширяют рамки платформ DCP и дают возможность создавать законченные решения для компонентной разработки приложений.

6.1. Основные концепции компонентной разработки приложений

Компоненты программного обеспечения – это простейшие структурные элементы, которые могут быть повторно использованы при построении программных систем. Они реализуют какие-либо прикладные функции ИС, представляя семантически значимые услуги прикладного или технического характера, и, как было сказано выше, могут быть модифицированы в процессе разработки на уровне двоичных исполняемых кодов.

6.1.1. Стандарты компонентов

Стандарты компонентов определяют методы построения программных компонентов и организации взаимодействия между ними. Эти стандарты указывают представление

компонента перед внешними для него объектами независимо от внутренней его реализации. Таким представлением компонента являются интерфейсы и протоколы взаимодействия. При соблюдении в процессе разработки приложений стандартных интерфейсов и протоколов компонентов гарантируется, что:

– компоненты со схожими спецификациями будут взаимозаменяемыми, и будут допускать возможность их независимой модернизации;

– внешний вид и поведение компонентов могут быть адаптированы разработчиками приложений применительно к заранее определенным прикладным функциям ИС;

– компоненты можно объединять друг с другом, формируя более крупные компоненты и законченные приложения.

Благодаря использованию стандартов при компонентной разработке приложений, становится возможным реализовать на практике преимущества повторного использования компонентов – повышение производительности труда при разработке, простоту применения, единообразие структуры приложений.

6.1.2. Интерфейсы компонентов

Под интерфейсом компонента, посредством которого внешние объекты могут подключаться к компоненту и поддерживать с ним взаимодействие, обычно понимают:

– дескриптор интерфейса;

– набор свойств компонента;

– набор событий, определяющих реакцию компонента на внешние воздействия или внутренние условия.

Свойства и методы компонента представляются каким-либо API-интерфейсом, который используют внешние объекты для доступа к сервисам, предоставляемым им данным компонентом. При этом свойства описывают значения общедоступных атрибутов компонента, а методы определяют его поведение. События определяют реакцию компонента на внешние воздействия или на внутренние условия (например, на изменение значения того или иного свойства). При этом интерфейс определяет, какое событие будет активизировано при возникновении некоторого условия. А внешние объекты, которые нуждаются в сервисах данного компонента, должны сами зарегистрироваться для получения события и предоставить свои методы для обработки этого события.

Такая модель взаимодействия объектов, основанная на механизме публикации и подписки, позволяет динамически устанавливать коммуникационные каналы между компонентами в распределенной среде выполнения.

6.1.3. Контейнеры

Компоненты существуют и функционируют внутри контейнеров. Контейнеры образуют общий контекст взаимодействия между компонентами приложений. Контейнеры предоставляют также компонентам, вложенным в другие компоненты (таким, например, как потоки процессов и ресурсы памяти), стандартный доступ к услугам среды выполнения (системного уровня).

Контейнеры обычно реализуются в виде компонентов и могут быть вложены в другие контейнеры. Для организации взаимосвязей между компонентом и вмещающим его контейнером обычно используются протоколы, основанные на механизме событий. Например, при буксировке пользователем компонента в контейнер с помощью мыши, работа протоколов динамических событий происходит следующим образом. В момент инициализации контейнер регистрируется в пункте фиксации (*drop site*) как объект, заинтересован-

ный в событиях, относящихся к буксировке. При этом пункты фиксации обычно реализуются в виде интерфейсов самого контейнера. Когда наступает событие завершения буксировки, пункт фиксации уведомляет об этом контейнер. В контейнере вызывается обработчик события (заранее зарегистрированный), и ему передается дескриптор перемещенного компонента. Обработчик передает перемещенному компоненту дескриптор контейнера, что открывает этому компоненту доступ к сервисам контейнера.

6.1.4. Метаданные

Стандарты компонентов определяют метаданные (т.е. данные о данных), которые компонент должен публиковать, чтобы иметь возможность взаимодействия с другими компонентами. Метаданные о свойствах данного компонента могут сообщаться либо статически на этапе проектирования приложения, либо динамически на этапе выполнения. О свойствах компонента через метаданные сообщается другим компонентам, контейнерам, сценариям и инструментам разработки приложений. В рамках DCP-платформ метаданные обычно оформляются в виде компонента особого типа, снабженного интерфейсами интроспекции (introspection) или рефлексии (reflection). Информация о компоненте описывает общие характеристики компонента, относящиеся к этапам компиляции и выполнения, в том числе она содержит указания о том, где можно найти компонент и как его активизировать (например, путь доступа и название процесса, которому принадлежит данный компонент, или процесса, который вызывает данный компонент).

Внешние ссылки указывают на метаданные, описывающие другие компоненты, с которыми данный компонент должен взаимодействовать.

Описатель типа и тип, – фундаментальные элементы метаданных, которые обеспечивают их идентификацию.

Интерфейсы определяют, как было сказано выше, общедоступные атрибуты (свойства), методы и события компонента. Атрибуты, методы и события характеризуют интерфейсы и классы. Все они сопровождаются своими метаданными.

Классы описывают реализации одного или нескольких интерфейсов.

Метаотношения между классами, типами и интерфейсами различны для разных DCP – платформ (см. ниже).

Типы возвращаемых значений и параметры задают входные и выходные данные для методов.

За счет использования метаданных разработчики компонентных приложений могут объединять компоненты и устанавливать динамические взаимоотношения между ними. С помощью метаданных компоненты могут обнаруживать интерфейсы других компонентов и взаимодействовать с ними на этапе выполнения.

Метаданные компонентов используются также инструментальными средствами, расширяющими среду разработки (программы просмотра объектов, отладчики, интеллектуальные редакторы программного кода).

6.1.5. Распределенные серверные компоненты

Системы распределенной обработки данных с архитектурой «клиент-сервер», составляющие основу построения современных ИС, являются наиболее сложным случаем для использования компонентных приложений. Однако именно в таких системах преимущества компонентной разработки приложений проявляются наиболее ярко, а использование готовых компонентов дает наибольший эффект.

В этих системах компоненты приложений на этапе выполнения должны использовать весь набор услуг, предоставляемых распределенными службами среды выполнения.

Для доступа к совместно используемым услугам среды, данным и вычислительным ресурсам компонентам требуются:

− протоколы удаленной связи, обеспечивающие взаимосвязь клиентских и серверных компонентов, распределенных по сети и выполняемых на серверах приложений, баз данных, системного администрирования и защиты информации (сетевые протоколы прикладного уровня эталонной модели ВОС). Эти протоколы могут быть синхронными, например, реализующими услугу удаленного вызова процедур, или асинхронными, например, реализующими услугу обмена сообщениями с промежуточным хранением без блокирования;

− службы каталогов, образующие глобальный механизм именования, организации и эксплуатации служб и ресурсов, совместно используемых приложениями;

− службы транзакций, объединяющие цепочки совместно работающих приложений и координирующие одновременные обновления корпоративных баз данных;

− службы системного администрирования, включающие в себя единый набор средств мониторинга и управления приложениями, сервисами и ресурсами среды выполнения;

− службы защиты информации, которые осуществляют аутентификацию пользователей и приложений, разграничение прав доступа и проверку полномочий при обращении к ресурсам системы, а также предохраняют сообщения от перехвата и вскрытия при несанкционированном доступе.

В предыдущих главах рассмотрены интерфейсы прикладного программирования (API) этих служб распределенной среды. Эти стандартные интерфейсы подлежат обязательному учету при проектировании и программировании приложений, в частности при компонентной разработке приложений.

Имеющиеся на рынке продукты, реализующие указанные выше службы распределенной среды, нередко предоставляют одни и те же услуги через разные интерфейсы. Номенклатура этих продуктов достаточно обширна и непрерывно обновляется, а сведения об их соответствии принятым («де-юре» и «де-факто») стандартам API не всегда приводятся их поставщиками. Поэтому задача выбора рационального состава продуктов ***системного ПО*** и ***ПО промежуточного слоя*** для распределенной среды является весьма сложной. А с учетом дополнительных условий конструирования серверных компонентов, опирающихся на услуги этой распределенной среды, задача тем более усложняется.

Например, серверный компонент приложения может получать от среды услуги промежуточного хранения данных в процессе обработки транзакций из своего контекста на этапе выполнения вместо того, чтобы реализовать их самостоятельно. И это должно быть предусмотрено разработчиком при компонентной разработке приложений.

Серверные компоненты должны обслуживать одновременно множество клиентов (тогда как компоненты настольных систем поддерживают только одиночных пользователей), что также должно предусматриваться разработчиком, конструирующим, например, серверы приложений. Кроме того, серверные компоненты в целях обеспечения масштабируемости и высокой готовности системы часто приходится делать многопоточными, дублировать их и объединять в пулы. Все эти особенности разработки серверных компонентов приводят к усложнению задачи организации этих компонентов в статические иерархии контейнеров, которая следует из основной концепции компонентной разработки приложений.

Разработчики платформ DCP работают над обобщением моделей, строящихся на базе контейнеров, стремясь к тому, чтобы они обеспечивали на этапе выполнения интерфейсы между серверными компонентами и платформами, на которых они должны функционировать. Развитие таких моделей на базе контейнеров является одним из важнейших направлений конкретизации общей эталонной модели среды открытых систем OSE / RM.

6.2. Интегрированные среды разработки приложений

В данном разделе приводится краткое описание наиболее распространенных сред, поддерживающих возможность интеграции компонентов приложений.

Встречающиеся в текущем разделе многочисленные названия моделей и интерфейсов следует рассматривать только как информацию, которую можно использовать при более детальном изучении того или иного объекта по опубликованным материалам.

Понятие ***компонент*** неразрывно связано с понятием ***среда разработки компонентов***, которая выступает в качестве конструктора компонентов. Ценность создаваемой в процессе проектирования ИС DCP-платформы во многом определяется эффективностью и практичностью применяемой инструментальной интегрированной среды разработки (*integrated development environment*, IDE).

Раньше среды IDE основывались, в основном, на программировании приложений в исходных кодах. Теперь они все в большей степени переходят на прямое манипулирование визуализированными компонентами. В качестве примеров наиболее популярных коммерческих сред IDE, доступных на рынке, можно привести Visual Studio фирмы Microsoft, Visual Age for Java фирмы IBM, Visual Cafe фирмы Symantec. Все они комплектуются языками сценариев, такими, как VB Script и Java Script.

В составе среды IDE обычно предусматриваются:

– одна или несколько палитр для отображения имеющихся компонентов (показываемых в виде пиктограмм) на экране, с которым работает разработчик;

– контейнер – «холст», где размещаются компоненты и устанавливаются взаимосвязи между ними (обычно с помощью мыши или всплывающих меню);

– редакторы свойств компонентов и сценариев, с помощью которых разработчики могут настраивать компоненты, включаемые в состав контейнеров;

– редакторы, программы просмотра, интерпретаторы, компиляторы и отладчики исходного кода, обеспечивающие разработку новых компонентов, тестирование и совершенствование приложений, состоящих из компонентов;

– архив компонентов и сопутствующие службы просмотра, позволяющие разработчику в процессе разработки отыскивать нужные компоненты либо по заданным критериям, либо путем изучения метаданных компонентов с помощью специальных программ – инспекторов;

– средства управления конфигурацией, которые структурируют и координируют процессы коллективной разработки приложений и выпуска версий программных продуктов, что весьма важно для крупных проектов.

Среда IDE используется, как правило, для построения приложений путем визуального объединения компонентов и создания сценариев взаимодействия между уже существующими (повторно используемыми) и новыми компонентами. Она должна не только поддерживать создание и соединение повторно используемых компонентов, но и выступать в роли базовой инфраструктуры разработки, в которую можно органично интегрировать новые компоненты. Среда IDE должна опираться на стандарты компонентов DCP-платформы, что гарантирует согласованное использование шаблонов иерархической композиции приложений и обмен метаданными компонентов, осуществляемое компонентами, контейнерами и самой средой IDE.

При использовании среды IDE разработчики могут настраивать уже существующие компоненты, устанавливая те или иные значения свойств компонентов с помощью редакторов свойств и задавая новые обработчики событий с помощью редакторов блоков событий, которые описывают дескрипторы блоков. Компоненты могут быть изменены разра-

ботчиками явно – с использованием меню и редакторов среды IDE, или неявно, – путем визуальных манипуляций, например, перемещения пиктограмм компонентов в контейнеры с помощью мыши. И в том, и в другом случае, среда IDE должна обеспечивать доступ к метаданным компонентов, которые хранятся в репозитарии, с тем, чтобы отобразить их на экране разработчика или передать в окно редактора, либо обработать событие буксировки компонента в контейнер.

Настраиваемые компоненты необходимо сопровождать сведениями о том, с каким этапом (выполнения или проектирования) связана работа, т.к. в каждом из этих случаев должны применяться разные программные интерфейсы и по-разному отображается поведение компонентов. На этапе проектирования компонент взаимодействует со средой IDE и инициирует необходимые редакторы свойств и событий, при этом среда обеспечивает доступ к метаданным компонента. Во время выполнения компонент функционирует так, как это определено на этапе проектирования, взаимодействуя с другими компонентами и средой выполнения через API этапа выполнения.

6.2.1. Модель DCOM

Платформа распределенных компонентов, предложенная фирмой Microsoft, базируется на модели DCOM (*Distributed Component Object Model*). Эта модель является развитием модели COM (*Component Object Model*).

Модель COM была реализована Microsoft в виде среды VBX применительно к среде разработки компонентов Visual Basic и в виде среды OCX применительно к элементам управления OLE (*Object Linking and Embedding*).

Модель DCOM задает тип и структуру интерфейсов, которые обеспечивают взаимодействие компонентов в распределенной среде. Каждый компонент в модели DCOM должен обладать, по крайней мере, одним интерфейсом IUnknown, который поддерживает основные механизмы интерфейсных ссылок. Механизм уведомления о событиях реализован в DCOM с помощью функции I Connection Point и некоторых других сопутствующих интерфейсов. Для доступа к метаданным, имеющимся в библиотеке типов Type Library, предусмотрены интерфейсы ItypeLib и ITypeInfo. Интерфейс IProvideClass для уже функционирующего экземпляра компонента предоставляет доступ к библиотеке типов, позволяя динамически обнаруживать интерфейсы и обеспечивать взаимодействие компонентов в процессе выполнения.

Интерфейсы компонентов DCOM описываются на языке *Interface Definition Language* (IDL), разработанном консорциумом *Open Software Foundation* (OSF).

Среда компонентной разработки, основанная на модели DCOM, поддерживает в настоящее время разработку компонентов на трех языках – Visual Basic, Visual C++ и J++ (язык Java, реализованный Microsoft). Визуальная разработка компонентов поддерживается интерфейсом, предусмотренным в модели DCOM, IProperty Page с помощью страниц свойств компонентов (property sheet) –встроенных редакторов свойств и их агрегированных наборов. Сама эта среда представляет собой приложение, построенное также на основе модели DCOM, благодаря чему ее можно расширять и настраивать с помощью стандартных механизмов DCOM.

Метаданные о свойствах компонентов содержатся в библиотеке типов Type Library, которая содержит сведения пяти основных категорий:

Co Class – метадескриптор объекта COM, описывающий все входные и выходные интерфейсы для данного класса COM, включая общедоступные свойства и методы;

Interface – сведения о схеме распределения памяти и описания общедоступных операций, например, имена и возвращаемые типы;

Module – описание модуля DLL – библиотеки, в т.ч. путь доступа, глобальные переменные и экспортируемые функции;

Type def – метадескриптор структур данных, определяемых пользователями;

Importlib – получение метадескриптора библиотеки типов Type Library по указанной ссылке.

В языках C++ и Visual Basic, в отличие от Java нет встроенной поддержки рефлексии. Поэтому в DCOM все метаданные (кроме относящихся к J++) выражаются в терминах компонентной модели, а не языка программирования.

Платформа распределенных компонентов на базе модели DCOM использует операционную систему Windows NT. В составе служб среды распределенной обработки данных на этой платформе предусмотрены:

– протокол удаленной связи, использующий механизм удаленного вызова процедур RPC, реализованный Microsoft на основе стандартной спецификации RPC OSF DCE *(Distributed Computing Environment*). Этот протокол обеспечивает связь распределенных компонентов через стандартный механизм посредников (proxy) и заглушек (stub). Планируется также обеспечить в DCOM поддержку асинхронного протокола путем интеграции в DCOM системы обмена сообщениями MS Message Quene;

– служба каталогов Microsoft Active Directory, которая объединяет систему именования DNS (*Domain Name System*) и протокол LDAP *(Lightweight Directory Access Protocol*), совместимый с протоколом X.500;

– служба безопасности, поддерживающая протокол SSL (*Secure Sockets Layer*), который обеспечивает защиту данных на базе механизма открытых ключей. Планируется также предусмотреть поддержку службы безопасности, совместимой с Kerberos 5.0;

– служба системного администрирования, поддерживающая механизмы мониторинга и контроля ресурсов и служб в распределенной среде;

– служба распределенной обработки транзакций в виде сервера Microsoft Transaction Server (MTS), который интегрирует службу транзакций в модель разработки компонентов и предоставляет серверным компонентам транзакционную среду исполнения. В сервере MTS определяется контекст объекта (*Object Context*), аналог контейнера для серверных компонентов. При этом серверный компонент обращается к собственному операционному контексту через свой интерфейс с Iobject Context.

Компоненты, созданные в рамках некоторого контекста или добавленные к нему, прозрачным образом участвуют в транзакциях этого контекста и пользуются его системой защиты.

6.2.2. Спецификация Java Beans

Платформа распределенных компонентов, основанная на спецификации Java Beans, предложена фирмой Sun Microsystems. Возможности компонентов Java Beans реализуются в виде набора языковых расширений стандартной библиотеки классов Java. То есть, спецификация Java Beans представляет собой совокупность специализированных интерфейсов языка программирования Java. Если модель DCOM нейтральна относительно языков программирования, но зависит от платформ (ориентирована, прежде всего, на Windows NT), то спецификация Java Beans, наоборот, нейтральна относительно платформ, но зависит от языка (ориентирована на язык Java). Мобильность создаваемых компонентов относительно платформ обеспечивается технологией виртуальной Java – машины.

Интерфейсы Java Beans, как и в DCOM, включают в себя свойства, методы и события. В модели свойств Java Beans, помимо обычных описаний свойств с одним или несколькими значениями, определены дополнительно так называемые ***связующие свойства***

(*bound properties*) и ***ограничительные свойства*** (*constrained properties*). Связующие свойства с помощью событий Java извещают об изменении значения некоторого свойства компонента другие компоненты. Ограничительные свойства дают возможность этим компонентам запретить изменение. Они обеспечивают единообразный языковый подход к правилам проверки корректности, что требуется для поддержки целостности обрабатываемых данных. Связующие и ограничительные свойства, предусмотренные в Java Beans, позволяют разработчикам компонентного ПО реализовать прикладную логику ИС на модульной основе, обеспечивая возможности сопровождения.

Механизм уведомления событийного типа, используемый в Java Beans, содержит три взаимосвязанных интерфейса классов Java: *Event*, *Event Source* и *Event Listener*. Источник события (*Event Source*) оповещает все зарегистрированные в системе приемники событий (*Event Listener*) о наступлении интересующего их события, передавая каждому из них объект Event. Источники событий по умолчанию считаются широковещательными, хотя их можно сделать при необходимости направленными, связанными с одним приемником. Кроме того, в модель можно включать адаптеры событий (*Event Adapter*), чтобы не писать заново программы обработки для всех событий, определенных в интерфейсе приемника.

Кроме конструкций интерфейсов компонентов в Java Beans предусмотрены механизмы формирования контейнеров и метаданных, аналогичные механизмам, используемым в модели DCOM.

Среда компонентной разработки, основанная на Java Beans, имеет API – интерфейс, который явным образом поддерживает визуальную разработку компонентов с помощью страниц свойств, аналогично интерфейсу I Property Page модели DCOM. В этом качестве выступают интегрированные среды разработки (Integrated Development Environment – IDE), доступные на рынке: Visual Cafe фирмы Symantec, Visual Age for Java фирмы IBM, J Builder фирмы Inprise (Borland), Java Workshop фирмы Sun Microsystems.

Они сопоставимы по функциональным возможностям со средой Visual Studio фирмы Microsoft, обеспечивая визуальную разработку компонентов с применением страниц свойств, палитр и заблаговременным определением правил буксировки готовых компонентов с помощью мыши.

Для формирования метаданных в платформе Java Beans используется API – интерфейс *Core Reflection*, унаследованный от языка Java. Он представляет собой специализированный набор классов Java. Каждому из методов, полей, конструкторов, интерфейсов и классов Java ставится в соответствие класс метаданных, поддерживающий динамический опрос, создание экземпляров и инициирование.

Помимо интерфейса *Core Reflection*, в составе Java Beans имеется интерфейс интроспекции *Introspection*, который предоставляет набор классов метаданных, адаптированных для поддержки компонентной разработки. Например, класс метаданных Bean Info определяет визуальные пиктограммы компонентов, отображаемые в палитре среды разработки. Другие классы метаданных Java Beans являются производными от общего базового класса Feature Descriptor.

Для систематического использования метаданных в Java Beans включен вспомогательный класс Inspector, который помогает разработчику ориентироваться в использовании API – интерфейсов Introspection и Core Reflection. Средства поддержки метаданных Java Beans в значительной степени опираются на соглашения об именах – конструктивные шаблоны (*design pattern*). Например, для правильной работы интерфейсов интроспекции методы аксессора и модификатора для некоторого свойства X должны иметь имена Get X и Set X. Аналогичные ограничения строчно-ориентированные шаблоны налагают на имена, используемые в интерфейсах Event Listener и Bean Info.

Платформа распределенных компонентов, основанная на Java Beans, описана спецификацией Enterprise Java Beans (EJB) фирмы Sun Microsystems. Она представляет собой распределенную масштабируемую серверную среду для Java Beans, аналогичную по функциональным возможностям среде для DCOM, построенной на базе ОС Windows NT.

Каждый компонент в среде EJB должен функционировать внутри контейнера, изолирующего его от рабочей операционной среды сервера.

Контейнер автоматически выделяет компоненту потока процессов и управляет от его имени службами поддержки параллелизма, защиты, долговременного хранения, транзакционной обработки и другими службами, предоставляемыми приложениям со стороны серверной среды.

В составе служб серверной среды EJB предусмотрены:

– Протокол удаленной связи. Java Beans имеет доступ к протоколу дистанционного вызова методов *(Remote Method Invocation* – RMI), входящему в состав Java и работающему непосредственно поверх сетевых протоколов TCP / IP. Однако для того, чтобы протокол RMI мог использоваться каким-либо классом Java, в определение этого класса необходимо явным образом внести некоторые изменения. Кроме того, экземпляры удаленных классов нельзя передавать по значению.

– Служба каталогов. Эта служба обеспечивается через API – интерфейс, независимый от реализации, *Java Naming & Directory Interface* (JNDI) фирмы Sun Microsystems. Этот интерфейс позволяет приложениям, написанным на языке Java, пользоваться существующими службами каталогов, такими, как DNS и LDAP. При этом интерфейс должен обеспечивать поиск контейнеров EJB.

– Служба безопасности. Платформа EJB может использовать все сервисы защиты, предоставляемые стандартным пакетом java-security фирмы Sun Microsystems. К ним относятся: аутентификация с применением открытого и секретного ключей, шифрование, управление цифровыми ключами и списками контроля доступа.

– Служба системного администрирования. Эта служба обеспечивается через API – интерфейс *Java Management* API (JMAPI) фирмы Sun Microsystems. Он предоставляет набор сервисов мониторинга, управления и администрирования, а также описание пользовательского интерфейса консоли системного администратора.

– Служба транзакций. В среде EJB определена плоская модель обработки транзакций, в основу которой положена спецификация Object Transaction Service, разработанная Object Management Group (OMG) (API – интерфейс *Java Transaction Service* – JTS в этой среде не используется). Служба делегирует права управления транзакциями контейнеру компонента Java Beans.

В Java Beans предусмотрены средства, позволяющие упаковать компоненты Java Beans для вложения в контейнеры, поддерживающие модель DCOM, в т.ч. в контейнеры Visual Basic, Internet Explorer, Office, Lotus Notes. Для этого служит специальный коммуникационный мост, технологической основой которого служит утилита упаковки. Эта утилита для выбранных компонентов Java Beans генерирует библиотеку OLE Type Library и реестровую информацию для интерфейса Win32. Полученные в результате данные позволяют контейнерам DCOM правильно анализировать, представлять и обрабатывать компоненты Java Beans, например, перехватывать события компонента, инициировать методы его служб, создавать страницы свойств для настройки компонентов.

6.2.3. Компонентная разработка WEB-приложений

Сеть Интернет и корпоративные сети (*intranet*) использовались до настоящего времени для представления информационных ресурсов и доступа к ним со стороны удаленных пользователей. Привычным представлением информационных ресурсов в Интернет стали Web-страницы. А Web-технологии, базирующиеся на стандартном языке гипертекстовой разметки документов HTML и стандартных протоколах обмена документами и файлами HTTP и FTP, реализованы в массовых продуктах, обеспечивающих доступ к распределенным в сетях документам. К сожалению, попытки использовать Web в качестве платформы для приложений с распределенными компонентами сдерживались до недавнего времени ограниченными возможностями языка HTML. Это связано с тем, что язык HTML позволяет представлять Web-документы в виде совокупности тегов только из строго определенного набора. Теги HTML задают содержимое и формат документа. В качестве временной меры консорциум World Wide Web Consortium (W3C), ведущий стандарты Интернет, включил в версию 4.0 языка HTML тег <*object*>.

Вместе с тем консорциум W3C организовал разработку ряда архитектурных решений с более развитой концептуальной основой, чем язык HTML. К ним относятся стандарты расширяемого языка разметки Extensible Markup Language (XML), описания метаданных ресурсов Web Resource Definition Framework (RDF) и модели документов Web Document Object Model (DOM). Указанные стандарты определяют семантику объектов для сетей распределенных документов. В эти же модели может быть вписана и семантика распределенных компонентных приложений. Они, так же, как и платформы распределенных компонентов, основаны на том, что метаданные обеспечивают развитие семантических возможностей взаимодействия между документами Web и между компонентами приложений.

Язык XML представляет собой метамодель для обмена структурированными документами. Он базируется на стандарте ISO *Standard General Markup Language* и поддерживает определение языков разметки, допускающих изменения для каждой конкретной области применения. В частности, можно определять теги XML, которые позволяли бы классифицировать апплеты компонентов по правилам, принятым в конкретной организации или отрасли.

Стандарт RDF представляет собой метамодель для описания и сбора метаданных о ресурсах Web, которая описана на языке XML. Метаданные, представленные в RDF, могут использоваться средствами поиска, интеллектуальными программными агентами и другими клиентскими и серверными Web-приложениями для поиска компонентов, их классификации, управления доступом, лицензирования и администрирования.

Модель DOM описывает интерфейсы, через которые сценарии или сами приложения могут взаимодействовать с документами Web. Используя подходящие теги языка XML, модель DOM дает возможность манипулировать Web-документами как распределенными компонентами или контейнерами компонентов.

Таким образом, создана возможность единообразного подхода к архитектуре Web-документов и компонентов Web-приложений.

Как было показано в предыдущих главах, услуги, предоставляемые приложениям со стороны среды открытых систем, реализуются с помощью программного обеспечения промежуточного слоя (*middleware*). Существенные требования к архитектуре ПО промежуточного слоя возникли на пересечении Web-технологий и объектной технологии как модели разработки и исполнения приложений, взаимодействующих с ресурсами Web. В случае реализации возможностей Web в многоуровневых системных архитектурах, ПО промежуточного слоя должно обеспечивать соединение унаследованных систем с клиен-

тами, находящимися в сети Интернет. Кроме того, это ПО должно вовлекать в обслуживание запросов все уровни, независимо от того, где сосредоточена бизнес-логика этого обслуживания – в унаследованной системе или в приложении, непосредственно взаимодействующим с ПО промежуточного слоя.

ПО промежуточного слоя систем, взаимодействующих в сети Интернет, должно поддерживать работу нескольких соединений в контексте запроса, используя данные и бизнес-логику систем, базирующихся на различных платформах и функционирующих в различных операционных средах. В общем случае это ПО должно обеспечивать такие возможности, как доступ к базам данных и управление ими, передачу сообщений и обработку транзакций, установление соединений между уровнями или серверами, а также интеграцию с Web-сервером.

По оценкам специалистов IBM, ее клиенты свыше половины средств, вкладываемых в проекты разработок сложных систем, расходуют на построение интерфейсов к тем или иным системным средствам. Это связано с тем, что существующие программные продукты промежуточного слоя поддерживают разные стандарты или не поддерживают никаких стандартов, не полностью совместимы с другими функциями среды и не могут тесно интегрироваться с ними, основываются на различных моделях организации вычислительной обработки. В связи с использованием возможностей Интернет для построения корпоративных ИС в центре внимания оказались следующие проблемы ПО промежуточного слоя:

– разработка стандартных, согласованных программных служб, таких, как безопасность и именование;

– предоставление возможностей Web унаследованным приложениям;

– создание прикладных инфраструктур промежуточного слоя, учитывающих конвергенцию технологий Web и объектных технологий;

– поддержка доступа к базам данных через Web;

– поддержка распределенной в Web логики приложений.

Эти проблемы требуется решать на основе достаточно стабильной и целостной архитектуры ПО. Ряд основных принципов такой архитектуры были предложены фирмой IBM на основе так называемой объектной модели Web и нового связующего ПО для компонентов и объектов – Component Broker.

Объектная модель Web представляет собой новую волну в создании информационных систем и способна оказать более ощутимое влияние, чем какая-либо другая из предшествующих парадигм.

Создание нового ПО промежуточного слоя для Web исходит из того обстоятельства, что Java – это не просто язык программирования для создания приложений Интернет.

Предполагается серьезная корректировка фундаментальных концепций структурирования и комплексирования приложений Web. Приложения и отдельные их компоненты должны «жить» в Web и использоваться либо независимо, либо как части единого целого. Они должны функционировать в рамках структурированной среды, основываться на стандартах и взаимодействовать со стандартными службами среды. Все это находит отражение в структурах языка Java и технологии EJB.

Точно так же, в целях оптимального использования новых возможностей должна быть структурирована среда исполнения Java-приложений.

Модель программирования Java – это модель компонентов.

Концепция разбиения прикладной программы на самодостаточные модули – компоненты, свойственная объектным технологиям, через средства Java переносится в Web, где на ее основе разработчики приложений могут строить структурированные, независи-

мые интеллектуальные элементы в виде компонентов Java Beans или апплетов, действующих в распределенной среде. Благодаря этому разработчикам не понадобится знать особенности реальных операционных сред и способов взаимодействия.

6.2.4. Спецификации компонентов в архитектуре CORBA

Открытые системы распределенной обработки объектов связаны с архитектурой брокера объектных запросов Common Object Request Broker Architecture (CORBA). Спецификации на эту архитектуру разрабатывает и поддерживает консорциум Object Management Group (OMG). До недавнего времени интересы OMG были сосредоточены на стандартах объектного уровня архитектуры CORBA. В связи со значительным расширением применений платформы распределенных компонентов Java Beans (см. раздел 6.2.2.) OMG определил четыре основные категории требований к спецификациям компонентного уровня:

– модель компонентов, которая определяет систему типов компонентов, интерфейсы для работы со свойствами компонентов и управления ими, механизмы инициирования и обработки событий, структуру жизненного цикла и порядок сериализации компонентов;

– средство описания компонентов, в состав которого входит рефлексивная информационная модель, поддерживаемая существующими или новыми – совместимыми, с моделью CORBA репозитариями;

– модель программирования, которая отображает описания компонентов в языки, поддерживающие язык определения интерфейсов CORBA IDL, и дает возможность манипулировать компонентами в запросах CORBA как параметрами, передаваемыми по значению;

– отображение в модель компонентов Java Beans, которое обеспечивает необходимые уровни взаимодействия, как на этапе проектирования, так и на этапе выполнения, в т.ч. инспекцию компонентов и автоматическую генерацию программного обеспечения, необходимого для интегрирования компонентов CORBA в инструменты, основанные на языке Java.

Известны три стандарта OMG, относящиеся к модели компонентов.

В *первом* из них рассматриваются проблемы компоновки объектов с помощью нескольких интерфейсов IDL (для непересекающихся служб) и разрешения конфликтов между интерфейсами в отношении одного и того же объекта.

Другой стандарт содержит предложения, направленные на то, чтобы интерфейсы в операциях с объектами CORBA могли передавать их по значению, поскольку передача параметров – объектов только по ссылке затрудняет перенос средств протокола дистанционного вызова методов RMI, базирующегося на Java, в протокол IIOP CORBA.

Третий стандарт касается языков сценариев, необходимых для автоматизации работы с компонентами CORBA.

Спецификациям OMG почти определенно (хотя это необходимо проверять каждый раз при построении профилей конкретных систем) соответствуют некоторые платформы распределенных компонентов, например, Java Beans. Другие платформы, например, DCOM (ActiveX) этим спецификациям не удовлетворяют. Для обеспечения при необходимости взаимодействия объектов CORBA и COM существуют специальные спецификации OMG. Аналогичные спецификации, определяющие взаимодействие компонентов DCOM и CORBA, также представляются актуальными и могут появиться под воздействием рынка в ближайшее время.

6.3. Перспективы развития методов и средств компонентной разработки приложений

Стандарты компонентного программного обеспечения развиваются в направлениях, заложенных конкурирующими между собой технологиями COM/DCOM фирмы Microsoft и Java Beans фирмы Sun Microsystems. Платформы распределенных компонентов (DCP), реализующие и расширяющие эти стандарты, быстро превращаются в зрелые коммерческие продукты. Потребности совместного использования разных платформ могут быть удовлетворены с помощью утилит, подобных мосту Java Beans/DCOM.

Простота применения новейших платформ интегрированной среды разработки приложений, сводящего разработку к сборке приложения из готовых компонентов, и возможности повторного использования компонентов позволяет прогнозировать рост рынка готовых компонентов. Если в настоящее время основную массу готовых компонентов, доступных на рынке, составляют элементы графического пользовательского интерфейса и компоненты общего характера, такие, как электронные таблицы, генераторы диаграмм и отчетов, то в ближайшем будущем следует ожидать появление на рынке готовых компонентов, представляющих объекты и процессы бизнеса в массовых областях применения, в частности электронной коммерции.

Весьма перспективным направлением является конвергенция Web-технологий и объектных технологий. Следует ожидать активности развития стандартов, которые обеспечат развертывание компонентного ПО в Интернет и интрасетях, подобно тому, как сейчас обеспечен доступ к Web-документам.

Ключевое значение для компонентной разработки приложений приобретает стандартизация.

Развитие компонентных инфраструктур, в которых DCP-платформы дополняются элементами, ориентированными на решение тех или иных прикладных задач, представляется новым поколением средств разработки приложений. Основная проблема здесь состоит в том, чтобы заменить палитры сегодняшних интегрированных сред разработки IDE, состоящие в основном из текстовых полей, таблиц, кнопок и других элементов пользовательского интерфейса, наборами объектов, служб и функциональных представлений бизнес-уровня.

В конечном итоге стоит ***задача*** дать разработчикам систем возможность строить ИС, как системы, опирающиеся ***на платформы бизнес-компонентов***, вместо программных систем, опирающихся ***на модульное их построение***.

Вопросы для самопроверки

1. Что заложено в основе концепции компонентной разработки приложений?
2. Дайте определение понятия «интерфейс компонента».
3. Что такое контейнер?
4. Что такое метаданные?
5. Что относится к распределенным серверным компонентам?
6. Что является интегрированной средой компонентной разработки приложений?

7. Термины и определения

В данном разделе приведены основные термины, используемые в тексте учебного пособия, и их трактовка применительно к контексту рассматриваемых вопросов. Поэтому приведенные толкования терминов не обязательно совпадают с толковыми словарями общего назначения.

API-профиль (*API-profile*). Профиль, определяющий конкретную комбинацию базовых спецификаций прикладного пользовательского интерфейса в соответствии с моделью OSE/RM, возможно дополненных базовыми стандартами и/или профилями для представления данных и их форматов.

OSI-профиль (*OSI-profile*). Профиль, составленный из базовых спецификаций, соответствующих модели OSE/RM, возможно дополненных базовыми стандартами и/или профилями для представления обмениваемых данных и их форматов.

Архитектура «клиент-сервер» (*client-server architecture*): модель выполнения прикладных программ (приложений) в распределенной функциональной среде, в которой выполнение программ, обеспечивающих взаимодействие с пользователем системы, производится на рабочих станциях – клиентах, а выполнение программ, реализующих выполнение серверных частей приложения, общих для нескольких клиентов – на компьютерах-серверах (серверах приложений, серверах баз данных и т.д.).

Архитектура аппаратуры компьютера (*computer architecture*) – описание системы команд, организации прерываний, организации памяти и ввода-вывода – с точки зрения разработчика операционной системы и системного администратора.

Архитектура брокера объектных запросов (*common object request broker architecture* – CORBA): архитектура функциональной среды открытых систем, в которой основным механизмом взаимодействия между приложениями (представляемыми в этом случае в виде программных объектов) является обмен сообщениями через брокер объектных запросов.

Архитектура вычислительной сети (*computer network architecture*): совокупность принципов логической и физической организации технических и программных средств, протоколов и интерфейсов вычислительной сети, например, локальной сети, объединяющей компьютеры клиентов и серверы информационной системы.

Архитектура информационной системы (*information system architecture*): описание прикладных функций системы и ее интерфейсов с внешней средой (пользователями, другими системами и т.д.) с точки зрения пользователя системы.

Архитектура прикладной платформы (*application platform architecture*): часть архитектуры функциональной среды, содержащая спецификации интерфейсов операционной системы (оболочек, утилит, ядра, файловой системы, сетевых протоколов), поддерживающей выполнение клиентских или серверных частей приложений – с точки зрения программиста приложений.

Архитектура среды распределенных вычислений (*distributed computing environment architecture*): архитектура функциональной среды открытых систем, в которой основным механизмом взаимодействия между приложениями является вызов удаленных процедур (remote procedure call – RPC).

Архитектура функциональной среды открытых систем (*open system environment architecture*): описание услуг, предоставляемых приложениям системы со стороны среды, в которой они функционируют, и интерфейсов прикладного программирования,

обеспечивающих взаимодействие приложений с функциональной средой, – с точки зрения проектирования системы и программиста приложений.

Базовый стандарт (*base standard*), также иногда используются термины формальный стандарт или стандарт de-ure. Международный стандарт, принятый ISO (международной организацией по стандартизации), или рекомендация организации ITU-T (до 1993 г. – CCITT) – международного союза по телекоммуникации.

Бизнес-процесс (*business process*): совокупность действий предприятия, получающая на входе определенные данные и продуцирующая результат, имеющий ценность для потребителя продукции этого предприятия. Например, процесс выполнения заказа является бизнес-процессом, на вход которого поступает заказ, а результат – заказанные товары, то есть доставка заказанных товаров потребителю и есть та ценность для потребителя, которую создает бизнес-процесс. Совокупность всех бизнес-процессов представляет собой бизнес-архитектуру предприятия. Бизнес-архитектура предприятия отображается на архитектуру информационной системы в виде состава прикладных функций ИС. Модель бизнес-процессов предприятия, полученная в результате предпроектного анализа его деятельности, является многоуровневой и обычно включает в себя три взаимосвязанные части: организационно-штатную структуру предприятия, собственно модель бизнес-процессов, пронизывающих предприятие «по горизонтали», и данные о ресурсах предприятия, необходимых для выполнения бизнес-процессов. Анализ требований к архитектуре ИС с точки зрения отображения бизнес-архитектуры предприятия показывает, как правило, необходимость информационного сопряжения подсистем, поддерживающих разные бизнес-процессы, на различных уровнях управления предприятием, и интерфейсного сопряжения функциональных подсистем. К ним можно отнести: системы управления рабочими потоками, системы планирования ресурсов предприятия, системы оперативного анализа данных, системы функционально-стоимостного анализа, системы имитационного моделирования и др. Детализация бизнес-процессов осуществляется посредством бизнес-функций, бизнес-операций и бизнес-правил, которые поддерживаются информационной системой, обслуживающей предприятие. Модель бизнес-процесса, с которой работает ИС, содержит набор информационных объектов (ИО), представляемых в виде кортежей D_i (a_i^1, a_i^2,....a_i^n), где D_i – идентификатор i-го ИО, а a_i^j – j-ый атрибут i-го ИО. Бизнес-операция представляется парой T_i D_j, где T_i – тип операции с j-ым ИО. Бизнес-функция представляется в виде кортежа бизнес-операций J_m ((T_{1m}, D_{11}), ..., (T_{km}, D_{k1})), где J_m – код должности исполнителя, а T_{1m}, ..., T_{km} – элементы множества бизнес-операций $\{T_i\}$. Модель бизнес-процесса представляет собой граф управления бизнес-функциями, состоящий из множества узлов, каждый из которых соответствует определенной бизнес-функции: множества управляющих ребер выполнения бизнес-функций, множества узлов, соответствующих структурным подразделениям предприятия и множества ребер подчиненности подразделений, множества ресурсов предприятия и множества взвешенных ребер использования ресурсов бизнес-функциями.

Бизнес-процесс-реинжиниринг (*business process reengineering*): фундаментальное переосмысление и радикальное перепланирование бизнес-процессов предприятия, имеющее целью резкое улучшение показателей деятельности предприятия, таких, как затраты, качество и скорость обслуживания потребителей.

Внешний интерфейс (*front-end interface*): средства и правила взаимодействия системы (подсистемы) с внешними для нее **объектами** (внешней средой) – пользователем, вычислительной сетью и т.д. – в отличие от ее взаимодействия с другими компонентами той же системы.

Внутренний интерфейс (*back-end interface*): интерфейс какого-либо компонента системы с другим компонентом той же системы.

Декомпозиция (*decomposition*) – разбиение объекта разработки (задачи, программы, данных, системы) на структурные единицы. Декомпозиция является одной из задач системного анализа и проектирования. Для программных средств ИС и программных изделий выделяют следующие уровни декомпозиции: версия, компонент, модуль, процедура, программа, макрокоманда. Декомпозиция заданных функций ИС – процесс детализации совокупности бизнес-процессов предприятия, определенных на стадии предпроектного обследования, до требуемого состава бизнес-функций, бизнес операций и бизнес-правил, выполняемая на стадии проектирования ИС.

Интеллектуальный интерфейс (*intelligent interface*): совокупность средств взаимодействия пользователя с ИС на ограниченном естественном языке, включающая: диалоговый процессор, планировщик, преобразующий описание задачи в программу ее решения на основе информации, хранящейся в базе знаний, и монитор, осуществляющий управление всеми компонентами интерфейса.

Интероперабельность (*interoperability*) – свойство открытой системы, означающее возможность взаимодействия данной ИС с другими системами при необходимости обращения к информационным ресурсам этих систем (массивам файлов, базам данных, базам знаний) или решения определенных задач с помощью вычислительных ресурсов этих систем. Интероперабельность обеспечивается стандартными форматами электронного обмена данными (electronic data interchange – EDI), принятыми для разных прикладных областей, стандартными протоколами удаленного вызова процедур (remote procedure call-RPC) и обмена сообщениями (message interchange).

Интерфейс (*interface*): совокупность средств и правил, обеспечивающих взаимодействие устройств вычислительной системы и/или программ; совокупность унифицированных технических и программных средств, используемых для сопряжения устройств в вычислительной системе или сопряжения между системами; граница раздела двух систем, устройств или программ. *Примечание*: в эталонной модели взаимосвязи открытых систем (OSI/RM) понятие «интерфейс» введено для обозначения границы между средствами двух соседних уровней модели, в отличие от понятия «протокол», которое обозначает средства и правила взаимодействия двух систем на одном и том же уровне модели, например, на транспортном уровне – протокол TCP.

Интерфейс пользователя (*user interface*): комплекс прикладных и системных программных средств, обеспечивающий взаимодействие пользователя с ИС.

Интерфейс прикладного программирования (*application program interface – API*): интерфейс взаимодействия между прикладными программами (приложениями) ИС и средой, в которой они функционируют; интерфейс взаимодействия между двумя прикладными программами, реализующими разные функции ИС, например, интерфейс между двумя подсистемами интегрированной ИС.

Компонент (*component*) – составная часть устройства, программы, системы, данных.

Компонент программного обеспечения (*software component*) – простейший структурный элемент, который может быть повторно использован при построении программ или программных систем. Программный компонент реализует какую-либо прикладную функцию ИС, представляя семантически значимые услуги прикладного или технического характера. Отличительной особенностью компонента (в отличие от модуля программы) является возможность его модификации в процессе разработки на уровне двоичного исполняемого кода. Представлением компонента перед внешними для него объектами (другими составными частями программы), независимым от его внутренней реализации, яв-

ляются интерфейсы и протоколы взаимодействия. Под интерфейсом компонента обычно понимаются: дескриптор интерфейса, набор свойств компонента, набор методов компонента, набор событий, определяющих реакцию компонента на внешние воздействия или внутренние условия.

Компонентная инфраструктура (*component framework*) – интегрированная среда компонентной разработки и исполнения приложений, содержащая: платформу, ориентированную на определенную модель компонентов (component object model – COM, distributed component object model – DCOM, Java Beans, CORBA или объектная модель Web), набор проектных шаблонов, адаптированных к приложениям некоторой области применения или технологии (например, технологии построения пользовательского интерфейса приложений), и набор готовых образцов компонентов.

Контейнер (*container*) – в терминологии объектно-ориентированного программирования – объект, файл или другой ресурс, используемый для хранения других объектов. При компонентной разработке приложений компоненты существуют и функционируют внутри контейнеров. Контейнеры образуют общий контекст взаимодействия между компонентами приложений. Контейнеры предоставляют также компонентам, вложенным в другие, более сложные компоненты, стандартный доступ к услугам среды выполнения. Контейнеры обычно реализуются в виде компонентов и могут быть вложены в другие контейнеры. Для организации взаимосвязей между компонентом и вмещающим его контейнером обычно используются протоколы, основанные на механизме событий.

Концептуальная модель (*conceptual model*) – система основных понятий и правил комбинирования классов понятий, независимых от языка их представления и являющихся смысловой структурой некоторой предметной области. Применительно к открытым системам концептуальная модель характеризует архитектуру системы в виде набора интерфейсов и протоколов взаимодействия между приложениями и средой, в которой они функционируют.

Масштабируемость (*scalability*): свойство открытой системы, означающее возможность изменения ее количественных характеристик, таких, как размерность решаемых задач, число одновременно обслуживаемых пользователей и т.д., путем настройки параметров приложений и баз данных, а не путем перепроектирования и программирования заново. Применительно к прикладным платформам ИС свойство масштабируемости означает предсказуемый рост их количественных системных характеристик при добавлении определенных вычислительных ресурсов, например, процессоров, модулей оперативной и дисковой памяти в конфигурациях серверов.

Метаданные (*metadata*): данные о данных. Вообще «мета» – это приставка, указывающая на то, что объект относится к более высокому уровню абстракции, чем уровень данных пользователя. В СУБД метаданные обозначают информацию о хранимых данных: таблицы описания данных и связей, адресные таблицы и другую информацию, используемую для просмотра данных и их трансформации. В компонентной разработке приложений метаданные компонента содержат сведения о компоненте, которые необходимы для обеспечения его взаимодействия с другими компонентами: описатель типа и тип, описание свойств компонента (классов и атрибутов), описания методов компонента и событий, определяющих реакцию компонента на внешние воздействия или внутренние условия. Метаданные о компоненте могут сообщаться либо статически (на этапе проектирования), либо динамически (на этапе выполнения).

Метамодель (*meta model*) – в СУБД – модель данных, определяемая на метаязыке и основанная на общих, независимых от конкретных моделей данных, концепциях, которые обеспечивают однозначное выражение семантических свойств разнообразных моде-

лей данных, определения их сходства и различий при использовании единого языка (представления и манипулирования данными).

Модель (*model*): материальный объект, система математических зависимостей, или программа, имитирующие структуру или функционирование какого-либо исследуемого объекта. Основное требование к модели – ее адекватность объекту. Например, модель бизнес-процесса представляет собой граф управления бизнес-функциями, состоящий из множества узлов, каждый из которых соответствует определенной бизнес-функции: множества управляющих ребер выполнения бизнес-функций, множества узлов, соответствующих структурным подразделениям предприятия и множества ребер подчиненности подразделений, множества ресурсов предприятия и множества взвешенных ребер использования ресурсов бизнес-функциями. Модель бизнес-процессов предприятия, полученная в результате предпроектного анализа его деятельности, является многоуровневой и обычно включает в себя три взаимосвязанных части: организационно-штатную структуру предприятия, собственно модель бизнес-процессов, пронизывающих предприятие «по горизонтали», и данные о ресурсах предприятия, необходимых для выполнения бизнес-процессов.

Обобщенная модель открытой системы (*generalized open systems model*) – представление структуры открытой системы в виде набора таблиц, где указываются ссылки на стандарты и спецификации интерфейсов и протоколов взаимодействия на всех уровнях структуры, включая взаимодействие на уровне «приложение-приложение» и на уровне «приложение-среда».

Открытая информационная система (*open information system*) – система, в которой реализованы открытые спецификации на интерфейсы, сервисы (услуги среды) и поддерживаемые форматы данных. Это дает возможность должным образом разработанному приложению быть переносимым в широком диапазоне систем с минимальными изменениями, взаимодействовать с другими приложениями на локальных и удаленных системах и взаимодействовать с пользователями в стиле, который облегчает переход пользователей от системы к системе.

Открытая спецификация (*open specification*) – общедоступная спецификация, которая поддерживается открытым, гласным согласительным процессом, направленным на приспособление новой технологии к ее применению, и которая согласуется со стандартами. Открытая спецификация является технологически независимой, т.е. не зависит от специфического аппаратного или программного обеспечения или продуктов конкретного изготовителя.

Переносимость (*portability*) –свойство открытой системы, означающее возможность переноса прикладных программ и данных на другие аппаратно-программные прикладные платформы при их модернизации или замене с минимальными затратами. Применительно к «переносимости» пользователей (user portability) это свойство обеспечивается дружественным пользовательским интерфейсом. Стабильность его поддерживается, чтобы не переучивать пользователей при внесении изменений в приложения и прикладные платформы, стандартизованными API по функциям пользовательского интерфейса и сохранением способов взаимодействия с пользователем, реализуемых приложениями (экранные формы, способы работы с каталогами файлов, способы задания запросов к базам данных, командные языки и т.д.).

Прикладная платформа (*application platform*) – операционная система и оборудование компьютера, на котором осуществляется выполнение прикладных программ (приложений). В ИС архитектурой распределенной обработки данных типа “клиент-сервер” прикладные платформы серверов (приложений, баз данных и т.д.) исполняют серверные

части приложений, а прикладные платформы АРМ пользователей – клиентские части приложений. Совокупность нескольких разных по своей архитектуре прикладных платформ может образовать гетерогенную функциональную среду открытых систем.

Прикладная программа (приложение) (*application program*) – функциональная часть ИС, включающая в себя логически связанную группу модулей или компонентов, данные, средства обращения к информационным ресурсам ИС, которые необходимы для выполнения определенной прикладной функции ИС (бизнес – функции).Приложения ИС включают в себя данные, документацию (исходные тексты и описания), обучающие средства для пользователей, а также собственно программы в исполняемом коде.

Прикладная программа (приложение) (*application program*) – функциональная часть ИС, включающая в себя логически связанную группу модулей или компонентов, данные, средства обращения к информационным ресурсам ИС, которые необходимы для выполнения определенной прикладной функции ИС (бизнес – функции).

Прикладная функция ИС – то же, что бизнес-функция.

Программные средства промежуточного слоя (*middleware*) – средства, реализующие стандартные услуги функциональной среды открытых систем, такие, как функции управления базами данных, функции организации распределенной обработки данных. Например, мониторы транзакций, брокеры объектных запросов, функции обмена сообщениями, функции защиты информации (аутентификации пользователей и приложений, управления доступом к данным и приложениям), услуги телекоммуникационной среды, например, электронной почты, передачи файлов и т.д. Эти средства занимают в эталонной модели среды открытых систем промежуточное положение между приложениями и операционными системами прикладных платформ и поэтому называются программными средствами промежуточного слоя.

Программный интерфейс (*program interface*): интерфейс взаимодействия между программами.

Профиль (*profile*) – взаимосвязанная упорядоченная совокупность базовых стандартов и спецификаций, ориентированная на выполнение определенной прикладной функции ИС (или группы функций), определенной функции телекоммуникационной среды или на построение конкретной ИС в целом.

Стандарт (по определению ISO). Технический стандарт или другой документ, доступный и опубликованный, коллективно разработанный или согласованный и общепринятый в интересах тех, кто им пользуется, основанный на интеграции результатов науки, технологии, опыта, способствующий повышению общественного блага и принятый организациями, признанными на национальном, региональном и международном уровнях.

Таксономия (*taxonomy*). Классификационная схема, применяемая для однозначной идентификации профилей или наборов профилей.

Транзакция (*transaction*): 1. В СУБД – входное сообщение, переводящее базу данных из одного непротиворечивого состояния в другое, запрос на изменение базы данных. 2. В приложениях обработки данных – групповая операция, реализующая набор связанных бизнес-операций. Например, банковской транзакцией является совокупность операций по проведению платежа.

Функциональная интеграция – объединение систем, ранее создававшихся и функционировавших на предприятии автономно, независимо друг от друга (как «островки автоматизации», например, на производственных предприятиях в 70-х и 80-х годах) в единую интегрированную информационную систему, которая поддерживает все основные бизнес-процессы предприятия. Обеспечивая новые качества деятельности предприятия, например, в плане быстрого реагирования на изменяющиеся требования рынка и соответ-

ствующей перестройки бизнес-процессов, функциональная интеграция влечет за собой резкий рост сложности ИС. В свою очередь, возрастающая сложность ИС выдвигает дополнительные требования к методологии и средствам их проектирования и разработки.

Функциональная среда открытой системы (*Open System Environment – OS*E) – среда, поддерживающая разработку и выполнение приложений в открытой системе, предоставляя им стандартные услуги. Среда OSE обеспечивает исполнение приложений, при разработке которых были применены стандартные интерфейсы прикладного программирования (API), специфицированные для этой среды.

Функциональный стандарт (*functional standard*) – стандарт, определяющий один или несколько взаимоувязанных профилей с выделением во втором случае общих базовых стандартов этих профилей в отдельные части стандарта.

Эталонная модель (*reference model*) в функциональной стандартизации – представление структуры открытой системы в виде набора таблиц, где указываются ссылки на стандарты и спецификации интерфейсов и протоколов взаимодействия между компонентами этой системы. Различают эталонные модели среды открытых систем (*open systems environment/reference model – OSE/RM*) и взаимосвязи открытых систем *(open systems interconnection/reference model – OSI/RM*).

Литература

Основная

1. Филинов Е.Н. Выбор и разработка концептуальной модели среды открытых систем. Открытые системы, 1995, вып. 6.

2. Козлов В.А. Открытые информационные системы. М.: Финансы и статистика. 1999.

3. Липаев В.В., Филинов Е.Н. Мобильность программ и данных в открытых информационных системах. М.: Научная книга. 1997.

4. Липаев В.В. Методы обеспечения качества крупномасштабных программных средств.- М.: СИНТЕГ, 2003

5. Лезер Н. Архитектура открытых распределенных систем: Модель OSF DCE// Открытые системы. №3. 1993.

6. Якубайтис Э.А. Открытые информационные сети. М.: Радио и связь. 1991.

Дополнительная

7. А. Бойченко, Г. Горелкин, В. Горшков, Е. Филинов. Обобщенная модель открытых информационных систем //Сетевой журнал Data Communications, 2000, №1.

8. Вендров А.М. Проектирование программного обеспечения экономических информационных систем. М.: Финансы и статистика, 2000.

9. Галатенко В.А. Информационная безопасность//Открытые системы 1995, №№ 4, 5, 6. и 1996 №№ 1, 2, 3, 4.

10. Д.Слама, Д.Гарбис, П.Рассел. Корпоративные системы на основе CORBA. Изд. дом «Вильямс», Москва, Санкт-Петербург, Киев, 2000.

11. Филинов Е.Н. Архитектура и структура среды распределенной обработки данных, методы и средства формального описания среды // Распределенная обработка информации. Труды Шестого международного семинара. Новосибирск. Сибирское отделение РАН. 1998.

12. Калянов Г.Н. Консалтиннг при автоматизации предприятий. М.: СИНТЕГ, 1997.

13. Ойхман Е.Г., Попов Э.В. Реинжиниринг бизнеса. М.: Финансы и статистика. 1997

14. Орфали Р., Харки Д. JAVA и CORBA в приложениях клиент-сервер. Изд. «Лори», М. 2000.

15. Орфали Р., Харки Д., Эдвардс Д. Основы CORBA (пер. с англ.) М: МАЛИП. 1999.

16. Смирнова Г.Н., Сорокин А.А., Тельнов Ю.Ф. Проектирование экономических информационных систем. М.: Финансы и статистика, 2001.

17. Смит Д.М., Маленовски М. Время пришло для профессионалов в области открытых систем. Открытые системы, №1, 1995.

18. Спецификации консорциума OMG http://www.omg.org

19. Сухомлин В.А. Методологический базис открытых систем //Открытые системы. № 4 (12).1996.

20. Щербо В.К. Международная стандартизация в области информационных технологий. Проблемы информатизации. М.: .№4 1992

21. Щербо В.К., В.А. Козлов. Функциональные стандарты в открытых системах. Часть 1, часть 2. Справочное пособие. М., МЦНТИ, 1997

22. ITU-T Rec. 902|ISO/IEC 10746-2:1995, Reference Model for Open Distributed Processing – Reference Model: Foundation. ITU-T Rec. 903|ISO/IEC 10746-3:1995, Reference Model for Open Distributed Processing – Reference Model: Architecture.

Приложение 1

Уровни модели взаимосвязи открытых систем

В модели OSI средства взаимодействия делятся на семь уровней (см. рис. П.1.1). Формализованные правила, определяющие последовательность и формат сообщений, которыми обмениваются сетевые компоненты, лежащие на одном уровне, но в разных системах, называются ***протоколом***.

Модули, реализующие протоколы соседних уровней, принадлежащие одной системе, также взаимодействуют друг с другом в соответствии с четко определенными правилами и с помощью стандартных форматов сообщений. Эти правила принято называть ***интерфейсом***. Интерфейс определяет услуги, предоставляемые данным уровнем соседнему уровню.

В сущности, протокол и интерфейс относятся к одному и тому же понятию, но традиционно в области вычислительных сетей за ними закрепили разные сферы действия: протоколы определяют правила взаимодействия модулей одного уровня в разных узлах сети, а интерфейсы – модулей соседних уровней в одном узле.

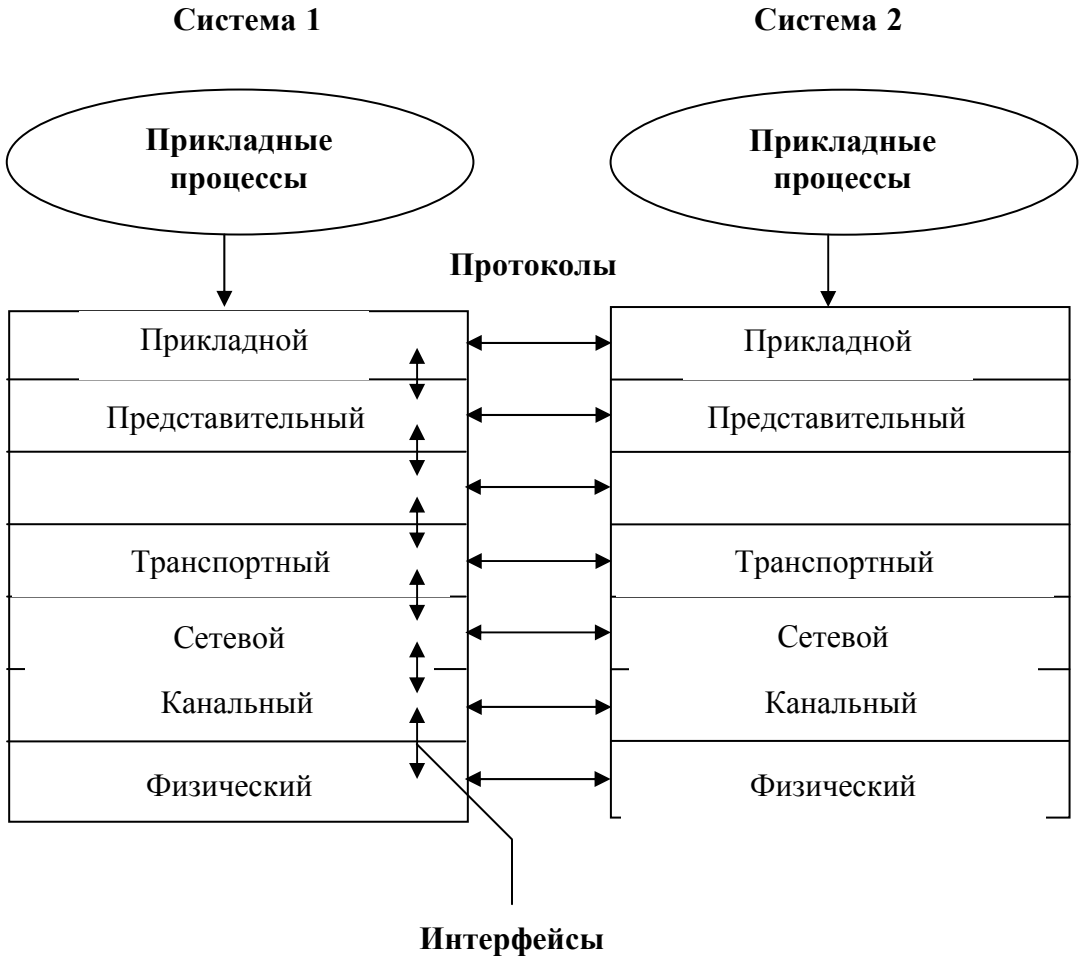

Рис. П.1.1. Модель ISO/OSI

Ниже представлено краткое описание уровней модели взаимосвязи открытых систем.

Прикладной уровень – это набор разнообразных протоколов, с помощью которых пользователи сети получают доступ к разделяемым ресурсам, таким как файлы, принтеры, Web-страницы, а также организуют свою совместную работу, например, с помощью протокола электронной почты. Единицей данных, которой оперирует прикладной уровень, обычно называется сообщением (message). Примерами протоколов прикладного уровня, могут быть протоколы NFS, FTP, TFTP, входящие в стек TCP/IP.

Представительный уровень (уровень представления данных) – имеет дело с формой представления передаваемых по сети данных, не меняя при этом их содержание (например, прием данных в коде ASCII и выдача их на экран дисплея в виде страницы текста с заданным числом и длиной строк). На этом уровне может выполняться шифрование и дешифрирование данных. Примером такого протокола является протокол Secure Socket Layer (SSL), который обеспечивает секретный обмен для протоколов прикладного уровня стека TCP/IP.

Сеансовый уровень – обеспечивает управление диалогом для того, чтобы фиксировать, какая из сторон является активной в настоящий момент, а также предоставляет средства синхронизации и выбора формы диалога пользователей (полудуплексная, дуплексная передача). На практике он редко реализуется отдельными протоколами, однако функции этого уровня часто объединяют с функциями прикладного уровня и реализуют в одном протоколе.

Транспортный уровень – обеспечивает связь между коммуникационной подсетью и верхними тремя уровнями и занимает центральное место в иерархии уровней. Главной его задачей является управление трафиком (данными пользователя) в сети. При этом выполняются такие функции, как деление длинных сообщений на пакеты данных (при передаче), формирование сообщений из набора пакетов (при приеме), обнаружение и исправление ошибок передачи (искажение, потеря или дублирование пакета). Транспортный уровень является границей, ниже которой объектом управления является пакет данных, а выше – сообщение. Примерами транспортных протоколов могут служить TCP и UDP стека TCP/IP и протокол SPX стека Novell.

Сетевой уровень – реализует функции буферизации и маршрутизации, т.е. прокладывает путь между отправителем и адресатом через всю сеть. Он служит для образования единой транспортной системы, объединяющей несколько сетей, причем эти сети могут использовать совершенно различные принципы передачи сообщений между конечными узлами. На сетевом уровне выполняются следующие функции: создание сетевых соединений, обнаружение и исправление ошибок, возникающих при передаче через коммуникационную сеть, управление потоками пакетов, организация последовательности пакетов, маршрутизация и коммутация, сегментирование и объединение пакетов. Примерами протоколов сетевого уровня являются протокол межсетевого взаимодействия IP стека TCP/IP и протокол IPX стека Novell.

Канальный уровень – осуществляет формирование и передачу блоков данных между системами. Главные его функции: управление передачей данных по информационному каналу (организация начала передачи, передача данных по каналу, проверка получаемых данных и исправление ошибок, отключение канала при его неисправности и восстановление после ремонта), и управление доступом к передающей среде. Физический и канальный уровни определяют характеристики физического канала и процедуру передачи по нему *кадров*, являющихся *контейнерами*, в которых транспортируются *пакеты*.

В локальных сетях протоколы канального уровня используются компьютерами мостами, коммутаторами и маршрутизаторами. В компьютерах функции канального уровня реализуются сетевыми адаптерами и их драйверами.

В глобальных сетях канальный уровень обеспечивает обмен сообщениями между двумя соседними компьютерами, соединенными линией связи. Примером может служить протокол PPP.

Физический уровень имеет дело с передачей битов по физическим каналам связи, таким, как коаксиальный кабель, оптоволоконный кабель, витая пара и т.д. Функции физического уровня реализуются во всех устройствах, подключенных к сети. Со стороны компьютера функции физического уровня выполняются сетевым адаптером или последовательным портом.

Примером протокола физического уровня может служить спецификация 10Base-T технологии Ethernet.

Приложение 2

Краткие сведения о международной системе стандартизации

В структуру международной системы стандартизации входят:
- официальные международные организации стандартизации;
- региональные организации стандартизации;
- национальные организации стандартизации;
- промышленные консорциумы и профессиональные организации.

Официальные международные организации стандартизации

ISO (International Organization for Standardization – Международная организация стандартизации, http://www.iso.ch/).

IEC (International Electrotechnical Commision – Международная электротехническая комиссия, http://www.iec.ch/).

ITU (International Telecommunication Union – Международный союз по телекоммуникации, http://www.itu.org/).

Перечисленные выше организации обладают признанными всеми странами полномочиями издавать международные стандарты, называемые также стандартами де-юре или формальными стандартами.

Формальными стандартами являются международные стандарты ISO, IEC и рекомендации ITU.

Деятельности этих организаций взаимосвязаны и согласованы. Они образуют основу системы международной стандартизации.

Европейские региональные организации стандартизации

По европейским законам в качестве официальных европейских организаций стандартизации ИТ признаются:

CEN (the European Committee for Standardization – www.cenorm.be) – европейский комитет стандартизации широкого спектра товаров, услуг и технологий, в том числе, связанных с областью ИТ – аналог ISO.

CENELEC (the European Committee for Electrotechnical Standardization – www.cenelec.be) – европейский комитет стандартизации решений в электротехнике, в частности, стандартизации коммуникационных кабелей, волоконной оптики и электронных приборов – аналог IEC.

ETSI (European Telecommunications Standards Institute – www.etsi.org) – европейский институт стандартизации в области сетевой инфраструктуры – аналог ITU-T.

Цель этих организаций – способствовать развитию процесса стандартизации в Европе, сотрудничеству с другими международными организациями стандартизации, обеспечению нормативной базы для создания (в 1992 г.) и эффективного функционирования общеевропейского рынка.

В 1998 г. организацией CEN создана организация ISSS (the Information Society Standartization System, www.cenorn.be/isss), целью которого является обеспечение участников рынка европейского информационного сообщества всеобъемлющей и целостной системой стандартов для продуктов и сервисов в области информационных и телекоммуникационных технологий.

Одной из задач ISSS является осуществление тесного взаимодействия с консорциумами для ускорения разработки стандартов, приближенных к коммерческому применению и занимающих промежуточное положение между формальными и неформальными аспектами стандартизации (Workshop-стандартизация).

Деятельность ISSS, направлена на разработку стандартов ИТ-приложений информационного общества, что дополняет деятельность организации ETSI по стандартизации технологий и сервисов информационной инфраструктуры.

Национальные организации стандартизации

В каждой индустриально развитой стране существует одна организация стандартизации, которая представляет данную страну в ISO в международном процессе стандартизации.

Примерами организаций национальных стандартов являются:

ANSI (American National Standards Institute, www.ansi.org) – американский институт национальных стандартов.

AFNOR (Association Francaise de Normalisation) – французская ассоциация по стандартизации, аналогичная по назначению ANSI.

BSI (British Standards Institute) – британский институт стандартов.

DIN (Deutsches Institute fur Normung e.v.) – германская организация национальных стандартов.

JISC (Japanese Industrial Standards Committee) – японский комитет промышленных стандартов.

Промышленные консорциумы и профессиональные организации

В последнее десятилетие быстрыми темпами развивалась стандартизация на уровне консорциумов (consortia standadization) и профессиональных организаций. Примерами представителей этой группы организаций-разработчиков стандартов являются:

ISOC (Internet Society – Общество Интернета, www.isoc.org/index.html) – ассоциация экспертов, отвечающая за разработку стандартов Интернет-технологий; IAB (Internet Architecture Board – Совет по архитектуре сети Интернет) – группа в составе ISOC, непосредственно отвечающая за развитие архитектуры Интернет, разработку и сопровождение стандартов протоколов и сервисов Интернет в виде RFC (Reference For Comments); два основных подразделения IAB:

IETF (Internet Engineering Task Force – Рабочая группа инженеров Интернета, www.ietf.org), решающая текущие задачи в области стандартизации и развития Интернет-технологий.

IRTF (Internet Research Task Force – Исследовательская группа Интернета, www.irtf.org), решающая проблемные задачи по развитию Интернет-технологий.

IEEE (Institute of Electrical and Electronic Engineers – Институт инженеров по электротехнике и электронике, www.ieee.org) – профессиональная международная организация-разработчик ряда важных международных стандартов ИТ.

OMG (Object Management Group – Группа управления объектами, www.omg.org) – международный консорциум, осуществляющий разработку стандартов для создания унифицированного распределённого объектного программного обеспечения.

ECMA (European Computer Manufactureres Association – Европейская ассоциация производителей вычислительных машин, www.ecma.ch.) – международная ассоциация, це-

лью которой служит промышленная стандартизация информационных и коммуникационных систем.

W3C (World Wide Web Consortium, www.w3.org) – консорциум, который специализируется на разработке и развитии стандартов WWW-технологий, таких, как, например, HTTP, HTML, URL, XML.

ATM Forum (ATM форум, www.atmforum.org) – консорциум, целями которого являются разработка и развитие стандартов широкополосных сетей асинхронного режима передачи данных (Asynchronous Transfere Mode, ATM).

DAVIC (Digital Audio-Visual Council – Совет по развитию цифровых аудио и видео мультимедиа систем, www.davic.org) – консорциум, осуществляющий разработку и развитие архитектурных, функциональных и информационных моделей и стандартов мультимедиа-сервисов Глобальной информационной инфраструктуры.

Open Group (www.opengroup.org) – организация, сформированная в 1996 году в результате объединения консорциумов X/Open и Open Software Foundation, исследует вопросы открытости и бесшовного введения информационных систем в интерсеть.

WFMC (Workflow Management Coalition – консорциум по управлению потоками работ, www.wfmc.org) – консорциум, занимающийся разработкой стандартов в области управления потоками работ и многие другие.

Международная организация ISO

ISO учреждена в 1947 г. как добровольная, неправительственная организации по соглашению между 25-тью индустриально развитыми странами о создании организации, обладающей полномочиями координировать на международном уровне разработку различных промышленных стандартов и осуществлять процедуру принятия их в качестве международных стандартов.

Деятельность ISO охватывает обширный спектр товаров, технологий и услуг в различных областях деятельности. Однако изначально полномочия ISO в международной стандартизации были ограничены двумя важными областями – телекоммуникационной, а также электротехнической и электронной отраслями, т.к. для них уже сложилась система международной стандартизации в лице организаций ITU и IEC.

Официальное название ISO это – International Organization for Standartization. ISO внесла большой вклад в становление международной системы стандартизации.

Общее число созданных и сопровождаемых ISO стандартов к 2001 г., составляло порядка 13000, из которых порядка 2000 стандартов относятся к области ИТ.

В ISO работает около 3 000 технических комитетов, подкомитетов и рабочих групп, в совещаниях которых ежегодно принимает участие более 30000 экспертов. ISO сотрудничает с более чем 500 международными организациями.

Стратегическим партнером ISO является Всемирная торговая организация (World Trade Organiztion, WTO).

Международная организация IEC

IEC образована в 1906 г. Также, как и ISO является добровольной неправительственной организацией.

Ее деятельность связана со стандартизацией физических характеристик электротехнического и электронного оборудования. Основное внимание IEC уделяет таким вопросам, как электроизмерения, тестирование, утилизация, безопасность электротехнического и электронного оборудования.

Членами IEC являются национальные организации (комитеты) стандартизации технологий в соответствующих отраслях. В настоящее время в состав IEC входит более 50 таких членов.

С организационной точки зрения IEC имеет устройство во многом аналогичное ISO. Процесс создания стандартов в IEC также аналогичен модели этого процесса, принятой в ISO.

Как и в ISO основную работу по разработке стандартов выполняют технические комитеты (TCs) и подкомитеты (SCs), общее число которых более 200.

Наряду со стандартами на электротехнические аппараты и машины IEC разрабатывает документы на электронные устройства, связанные с обработкой информации, в частности, шинные интерфейсы.

JTC1 (Joint Technical Committee 1 – Объединенный технический коммитет 1)

Имея совместные интересы в области стандартизации ИТ, ISO и IEC договорились объединить свои усилия, создав в 1987 г. единый орган **JTC1** (Joint Technical Committee 1 – Объединенный технический комитет 1), предназначенный для формирования всеобъемлющей системы базовых стандартов в области ИТ и их расширений для конкретных сфер деятельности.

Сфера деятельности – международная стандартизация в области ИТ, при этом понятие ИТ трактуется следующим образом: «Информационные технологии включают спецификацию, проектирование и разработку систем и средств, имеющих дело со сбором, представлением, обработкой, безопасностью, передачей, организацией, хранением и поиском информации, а также обменом и управлением информацией».

Основные цели JTC1: «разработка, поддержание, продвижение стандартов ИТ, являющихся необходимыми для глобального рынка, удовлетворяющих требованиям бизнеса и пользователей и имеющих отношение к следующему:

- проектированию и разработке систем и средств ИТ,
- производительности и качеству продуктов и систем ИТ,
- безопасности систем ИТ и информации,
- переносимости прикладных программ,
- интероперабельности продуктов и систем ИТ,
- унифицированным средствам и окружениям,
- гармонизированному словарю понятий области ИТ,
- дружеским и эргономичным пользовательским интерфейсам.

Работа над стандартами ИТ, осуществляемая в JTC1, тематически распределена по подкомитетам (Subcommittees – SC).

В 2001 г. состав подкомитетов JTC1 выглядел следующим образом:

- SC1 Vocabulary (словарь понятий).
- SC2 Corded character sets (Символьные наборы и кодирование информации).
- SC6 Telecommunication and information exchange between systems (Телекоммуникация и информационный обмен между системами).
- SC7 Software engineering (Программная инженерия).
- SC11 Flexible magnetic media for digital data interchange (Гибкая магнитная среда для обмена электронными данными).
- SC17 Identification cards and related devices (Идентификационные карты и связанные с ними устройства).
- SC22 Programming languages, their environments and system software interfaces (Языки программирования, их окружения и интерфейсы системного программного обеспечения).

SC24 Computer graphics and image processing (Компьютерная графика и обработка изображений).

SC25 Interconnection of information technology equipment (Взаимосвязь оборудования информационных технологий).

SC27 IT Securities techniques (Методы безопасности ИТ)

SC29 Coding of audio, picture, multimedia and hypermedia information (кодирование аудио, графической, мультимедийной и гипермедиа информации).

SC31 Automatic identification and data capture techniques (Автоматическая идентификация и методы считывания данных).

SC32 Data management and interchange (обмен и управление данными).

SC34 Document description and processing languages (языки описания и обработки документов).

SC35 Use interfaces (пользовательские интерфейсы).

SC36 Learning Technology (технологии обучения).

ITU (International Telecommunication Union – Международный Союз Электросвязи)

ITU – международная межправительственная организация, специализирующаяся в области стандартизации электросвязи.

Объединяет более 500 правительственных и неправительственных организаций. В ее состав входят телефонные, телекоммуникационные и почтовые министерства, ведомства и агентства разных стран, а также организации-поставщики оборудования для обеспечения телекоммуникационного сервиса.

Основная задача ITU состоит в координации разработки гармонизированных на международном уровне правил и рекомендаций, предназначенных для построения и использования глобальных телесетей и их сервисов.

В 1947 г. ITU получила статус специализированного агентства Организации Объединенных Наций (ООН). Центральный офис ITU расположен в Женеве (Швейцария).

ITU – основана в 1865 г. после подписания 20 европейскими государствами первой международной конвенции по телеграфии.

Первое название ITU расшифровывалось как Международный союз по телеграфии (International Telegraph Union).

Все время своего существования ITU несло ответственность за разработку правил и рекомендаций, регламентирующих развитие глобальных телекоммуникационных сетей и способствующих стандартизации телеуслуг, а также стандартизации операций по эксплуатации систем электросвязи.

ITU оперативно отслеживала новейшие достижения, включая изобретение телефона и радиотелеграфии, появление спутниковой связи и цифровых систем передачи данных, современных компьютерных сетей и систем мобильной связи, интегрируя эти достижения в глобальные телекоммуникационные услуги.

В 1932 г. было изменено название ITU. Организация стала называться Международным союзом по телекоммуникациям (International Telecommunication Union).

В 1956 г. в результате очередной реорганизации ITU был сформирован Международный консультативный комитет по телеграфии и телефонии (International Telephone and Telegraph Consultative Committee, – CCITT), в работах которого, в частности, были заложены основы стандартизации технологий компьютерных сетей.

В декабре 1992 г. на внеочередной женевской конференции была проведена структурная реформа ITU. Были созданы три сектора:

- Радиокоммуникации (Radiocommunication Sector или ITU-R) – сектор, включающий общие функции бывшего комитета по радиосвязи CCIR, а также задачи, выполнявшиеся советом по регистрации частот FRB.

- Стандартизации телекоммуникаций (Telecommunication Standardizaion Sector (TSS) или ITU-T), который принял на себя функции CCITT, а также функции комитета по радиосвязи CCIR, связанные с выходом средств радиосвязи на сети общего пользования.

- Развития телекоммуникаций (Telecommunication Development или ITU-D) – сектор, определяющий вопросы стратегии и политики развития систем электросвязи.

Принимаемые ITU-T стандарты имеют статус рекомендаций.

Основная работа по разработке стандартов выполняется исследовательскими группами (Study Groups – SGs).

В 2000 г. насчитывалось 14 таких групп. С точки зрения стандартизации ИТ наибольший интерес представляет деятельность таких групп, как, например:

- SG7 – Data and open communications systems (Данные и открытые информационные системы)

- SG8 – Multimedia Services (Мультимедийные сервисы)

- SG10 – Software languages (Языки для программного обеспечения) – стандарты языков программирования и языков формальной спецификации для разработки телекоммуникационных систем.

- SG13 – GII principles and structure (структура и принципы глобальной информационной инфраструктуры).

Все Рекомендации разбиты по сериям, которые идентифицируются буквами алфавита от A до Z. Примеры серий:

A: Organization of the work of the ITU-T (Организация работы ITU-T).

E: Overall network operation, telephone service and human factors (Общая работа сетей, телефонные услуги и человеческие факторы).

G: Transmission systems and media, digital systems and networks (Системы передачи и среды, цифровые системы и сети).

H: Audiovisual and multimedia systems (Аудиовизуальные и мультимедийные системы).

I: Integrated services digital network – ISDN (Цифровая Сеть с Интеграцией Служб, ЦСИС).

V: Data communication over the telephone network (Передача данных по телефонной сети).

X: Data networks and open system communications (Сети передачи данных и связь открытых систем).

Y: Global information infrastructure (Глобальная информационная инфраструктура).

Z: Programming languages (Языки программирования).

Примерами Рекомендаций этих серий могут служить следующие стандарты и их подсерии:

X.200-X.299 – Open Systems Interconnection /стандарты взаимосвязи открытых систем/

X.400-X.499 – Message Handling Systems / стандарты систем обработки сообщений/

X.500-X.599 – Directory /стандарты справочной сетевой службы/

X.700-X.799 – OSI Management /стандарты сетевого управления для модели OSI/

X.900-X.999 – ODP /стандарты ODP/

Y.100-Y.199 – Global information infrastructure. General /общие стандарты Глобальной информационной инфраструктуры/

Z.100 – LANGAGE DE DESCRIPTION ET DE SPECIFICATION /стандарт языка SDL/.

Так как интересы JTC1 и ITU-T в области стандартизации ИТ перекрываются, важную роль приобретает сотрудничество между JTC1 и ITU-T. Примерами сотрудничества являются:

- Соглашение об общем тексте для стандартов ISO/IEC (т.е. JTC1) и Рекомендаций ITU-T/CCITT, относящихся к одним и тем же аспектам в областях OSI и ODP.
- Принятие одной организацией текста стандарта, разработанного другой организацией. Пример – принятие предшественником ITU-T, организацией CCITT, эталонной модели OSI, разработанной в недрах ISO, а также принятие организациями ISO/IEC рекомендаций по технологии передачи сообщений (MHS), разработанных CCITT.
- Совместная разработка стандартов, как, например, разработка стандарта для службы Справочника (Directory) и стандарта для эталонной модели открытой распределенной обработки (ODP).

IEEE (Institute of Electrical and Electronic Engineers, www.ieee.org)

IEEE – институт инженеров по электротехнике и электронике, основанный в США, – одна из самых больших международных профессиональных организаций. Ее целью являются продвижение теоретических и прикладных достижений электротехнической и электронной индустрий, а также поддержка профессионального роста специалистов.

Организации, входящие в IEEE, образуют общества (Societies) различной профессиональной направленности, которые рассматриваются как структурно самостоятельные единицы IEEE.

Разработка стандартов ИТ осуществляется в рамках Компьютерного общества IEEE (IEEE Computer Society), самого большого среди обществ, входящих в состав IEEE.

IEEE является аккредитованной ANSI организацией стандартизации, она может направлять свои стандарты в Совет по рассмотрению стандартов ANSI для проведения их в качестве национальных стандартов США. Затем эти стандарты могут передаваться в JTC1 для принятия их в качестве международных.

Наиболее известными международными стандартами в области ИТ, разработанными IEEE, стали:

- стандарты для локальных компьютерных сетей, получивших название IEEE 802LAN, созданные комитетом Компьютерной связи (Computer Communication);
- на переносимые окружения операционных систем (1003 POSIX), созданные комитетом Приложений и окружений операционных систем (Operating Systems Applications and Environments);
- большой спектр стандартов в области программной инженерии, например, ISO/IEC 12207:1995 Information technology – Software life cycle processes (процессы жизненного цикла программного обеспечения).

Организации ISOC, IAB, IETF, IRTF, IESG и Internet-стандартизацияОрганизации **ISOC, IAB, IETF, IRTF, IESG** структурно взаимосвязаны.

Они несут ответственность за стандартизацию Интернет-технологий.

Интернет – глобальная международная сеть, выросшая из недр сети ARPANET и исследований по сетям с пакетной коммутацией, финансируемых Агентством перспективных научно-исследовательских проектов министерства обороны США (DARPA).

Структурно организации ISOC, IAB, IETF, IRTF, IESG взаимосвязаны следующим образом.

ISOC (Internet Society – Общество Интернета, www.isoc.org/index.html) – ассоциация экспертов, отвечающая за разработку стандартов технологий сети Интернет. В рас-

сматриваемой организационной структуре ISOC располагается на верхнем уровне иерархии. ISOC называют также организационным домом (organizational home) для организаций IAB, IETF, IRTF, IESG.

ISOC является некоммерческой неправительственной международной профессиональной организацией. Ее члены – 175 организаций и около 9000 физических лиц из более чем 170 стран мира.

Организация ISOC

Работа **ISOC** сфокусирована на решении следующих основных задач, включая:

- Организацию процесса стандартизации технологий сети Интернет.
- Осуществление публичной политики.
- Поддержку инфраструктуры (организационно- административное управление деятельностью, управление финансами, защиту прав интеллектуальной собственности и пр.).
- Образование и обучение, в том числе, организацию ежегодных семинаров по обучению Интернет-технологиям (Network Training Workshops – NTW), организацию системы учебных центров (Sustainable Internet Training Centers – SITCs) и пр.
- Поддержку членства в ISOC как для организаций, так и для персональных членов.***Организация IAB***

IAB (Internet Architecture Board – группа технических советников в составе ISOC, непосредственно отвечающая за развитие архитектуры Интернет, управление разработкой и сопровождением стандартов протоколов и сервисов Интернет в виде RFC (Request For Comments).

В частности, IAB несет ответственность за избрание председателя IETF и руководящего состава IESG, осуществляет надзор за развитием архитектуры протоколов и процедур сети Интернет, а также надзор за процессом создания системы стандартов сети Интернет, включая стек протоколов TCP/IP.

Кроме этого, IAB несет ответственность за управление редактированием и публикацией спецификаций RFC (Request for Comments), а также управление присваиванием (посредством механизма IANA – Internet Assigned Numbers Authorities) номеров для RFC.

IAB выполняет представительские функции ISOC при взаимодействии с другими организациями.

Организация IETF

IETF (Internet Engineering Task Force – Комиссия по проектированию Интернет-технологий, www.ietf.org) является международным открытым сообществом разработчиков, операторов, изготовителей, исследователей, сетевых технологий и сервисов на основе сети Интернет.

IETF открыта для всех, кто интересуется Интернет-технологиями.

Основная сфера деятельности IETF состоит собственно в разработке стандартов сети Интернет, их эффективной реализации и тестировании.

Организации IRTF, IESG

IRTF (Internet Research Task Force – Исследовательская группа Интернет-технологий, www.irtf.org) – подразделение IAB, которое выполняет долгосрочные исследовательские программы, связанные с вопросами развития архитектуры, базовых протоколов и сетевых приложений сети Интернет. Руководящие органы IRTF назначаются IAB.

IESG (Internet Engineering Steering Group – группа технического управления сети Интернет, www.ietf.org) – отвечает за техническое управление процессом стандартизацией Интернет-технологий, осуществляет экспертизу проектов спецификаций, разрабатываемых IETF, несет ответственность за принятие Интернет-стандартов и их дальнейшее продвижение.

OMG (Object Managemant Group, www.omg.org)

OMG – международный некоммерческий консорциум, осуществляющий разработку, распространение и сопровождение индустриальных спецификаций, предназначенных для создания интероперабельных бизнес-приложений.

Консорциум OMG был основан в 1989 г. с целью концентрации усилий для достижения предельно возможных результатов в переносимости, переиспользуемости и интероперабельности приложений на основе использования объектно-ориентированной технологии создания программного обеспечения и компонентно-базированных методах проектирования систем.

OMG объединяет более 900 членов, включая основных производителей вычислительной техники и программного обеспечения.

Разработанные OMG руководства и спецификации по управлению объектами определяют интегрированную среду распределенной обработки данных, которая служит общей инфраструктурой для объединения ресурсов и систем.

Примерами наиболее известных спецификаций, разработанных OMG, являются:

- CORBA как унифицированного программного обеспечения среднего уровня (Middleware);
- протокола взаимодействия через сеть Интернет объектных брокеров CORBA/IIOP;
- объектных служб (Facilities) и сервисов (Services);
- предметно-ориентированных интерфейсов (Domain Interfaces), в том числе, спецификаций для разработки приложений вертикального рынка, а именно, в таких областях как производство, финансы, телекоммуникация, электронная коммерция, системы реального времени, здравоохранение;
- языка UML, а также других спецификаций для анализа и проектирования систем на основе объектно-ориентированной парадигмы.

Приложение 3

Перечень основных стандартов в области обеспечения качества программных средств

1. **ISO 12207:1995.** (ГОСТ Р – 1999). ИТ. Процессы жизненного цикла программных средств.

2. **ISO 15271:1998.** (ГОСТ Р – 2002). ИТ. Руководство по применению ISO 12207.

3. **ISO 16326:1999.** (ГОСТ Р – 2002). ИТ. Руководство по применению ISO 12207 при административном управлении проектами.

4. **ISO 15504 – 1-9:1998.** ТО. Оценка и аттестация зрелости процессов жизненного цикла программных средств. Ч.1. Основные понятия и вводное руководство. Ч.2. Эталонная модель процессов и их зрелости. Ч.3. Проведение аттестации. Ч.4. Руководство по проведению аттестации. Ч.5. Модель аттестации и руководство по показателям. Ч.6. Руководство по компетентности аттестаторов. Ч.7. Руководство по применению при усовершенствовании процессов. Ч.8. Руководство по применению при определении зрелости процессов поставщика. Ч.9. Словарь.

5. **ГОСТ Р 51904 – 2002**. Программное обеспечение встроенных систем. Общие требования к разработке и документированию.

6. **ISO 9000-3:1997.** Стандарты в области административного управления качеством и обеспечения качества. Часть 3. Руководящие положения по применению стандарта ISO 9001 при разработке, поставке и обслуживании программного обеспечения.

7. **ISO 9000:2000.** (ГОСТ Р – 2001). Система менеджмента (административного управления) качества. Основы и словарь.

8. **ISO 9001:2000.** (ГОСТ Р – 2001). Система менеджмента (административного управления) качества. Требования.

9. **ISO 9004:2000.** (ГОСТ Р – 2001). Система менеджмента (административного управления) качества. Руководство по улучшению деятельности.

10. **ISO 10005: 1995** – Административное управление качеством. Руководящие указания по программам качества.

11. **ISO 10006: 1997** – Руководство по качеству при управлении проектом.

12. **ISO 10007: 1995** – Административное управление качеством. Руководящие указания при управлении конфигурацией.

13. **ISO 10013: 1995** – Руководящие указания по разработке руководств по качеству.

14. **ISO 10011-1-3: 1990.** Руководящие положения по проверке систем качества. Ч.1. Проверка. Ч.2. Квалификационные критерии для инспекторов-аудиторов систем качества. Ч.3. Управление программами проверок.

15. **ISO 12182:1998.** (ГОСТ Р– 2002). ИТ. Классификация программных средств.

16. **ISO 9126:1991.** (ГОСТ – 1993). ИТ. Оценка программного продукта. Характеристики качества и руководство по их применению.

17. **ISO 14598-1-6:1998-2000.** Оценивание программного продукта. Ч.1. Общий обзор. Ч. 2. Планирование и управление. Ч. 3. Процессы для разработчиков. Ч.4. Процессы для покупателей. Ч.5. Процессы для оценщиков. Ч. 6. Документирование и оценивание модулей.

18. **ISO 9126-1-4.** (проект). ИТ. Качество программных средств: Ч.1. Модель качества. Ч.2. Внешние метрики. Ч. 3. Внутренние метрики. Ч. 4. Метрики качества в использовании.

19. **ISO 14756: 1999.** ИТ. Измерение и оценивание производительности программных средств компьютерных вычислительных систем.

20. **ISO 12119:1994.** (ГОСТ Р – 2000 г). ИТ. Требования к качеству и тестирование.

21. **ISO 13210:1994.** ИТ. Методы тестирования для измерения соответствия стандартам POSIX.

22. **ANSI/IEEE 1008 – 1986**. Тестирование программных модулей и компонентов ПС.

23. **ANSI/IEEE 1012 – 1986**. Планирование верификации и подтверждения достоверности качества (валидации) программных средств.

24. **ISO 9945-1:1990** (IEEE 1003.1). ИТ. Интерфейсы переносимых операционных систем. Ч.1. Интерфейсы систем прикладных программ (язык Си).

25. **ISO 9945-2:1992** (IEEE 1003.2). ИТ. Интерфейсы переносимых операционных систем. Часть 2. Команды управления и сервисные программы.

26. **ISO 15846:1998.** ТО. Процессы жизненного цикла программных средств. Конфигурационное управление программными средствами.

27. **ISO 14764: 1999.** (ГОСТ Р – 2002). ИТ. Сопровождение программных средств.

28. **ISO 15408 -1-3. 1999.** (ГОСТ Р – 2002). Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Ч.1. Введение и общая модель. Ч. 2. Защита функциональных требований. Ч. 3. Защита требований к качеству.

29. **ISO 13335 – 1-5. 1996-1998.** ИТ. ТО. Руководство по управлению безопасностью. Ч. 1. Концепция и модели обеспечения безопасности информационных технологий. Ч.2. Планирование и управление безопасностью информационных технологий. Ч.3. Техника управления безопасностью ИТ. Ч.4. Селекция (выбор) средств обеспечения безопасности. Ч.5. Безопасность внешних связей.

30. **ISO 10181: 1-7. ВОС. 1996-1998.** Структура работ по безопасности в открытых системах. Ч.1. Обзор. Ч. 2. Структура работ по аутентификации. Ч.3. Структура работ по управлению доступом. Ч.4. Структура работ по безотказности. Ч.5. Структура работ по конфиденциальности. Ч.6. Структура работ по обеспечению целостности. Ч.7. Структура работ по проведению аудита на безопасность.

31. **IEC 61508:1-6: 1998-2000.** Функциональная безопасность электрических / электронных и программируемых электронных систем. Часть 3. Требования к программному обеспечению. Часть 6. Руководство по применению стандартов IEC 61508-2 и IEC 61508-3.

32. **ISO 17799:2000.** Управление информационной безопасностью. Практические правила.

33. **ISO 15910:1999.** (ГОСТ Р – 2002) ИТ. Пользовательская документация программных средств.

34. **ISO 6592:2000.** ОИ. Руководство по документации для вычислительных систем.

35. **ISO 9294:1990.** (ГОСТ–1993 г). ТО. ИТ. Руководство по управлению документированием программного обеспечения.

36. **ISO 14102:1995.** ИТ. Оценка и выбор CASE-средств.

37. **ISO 14471:1999.** ИТ. Руководство по адаптации CASE- средств.

38. **ГОСТ 34.602-89**. ИТ. Техническое задание на создание автоматизированных систем.

39. **ГОСТ 34.603-92**. ИТ. Виды испытаний автоматизированных систем.

40. **ГОСТ 34.201-89**. ИТ. Виды, комплектность и обозначение документов при создании автоматизированных систем.

41. **РД 50-34.698-90.** Методические указания. Информационная технология. Автоматизированные системы. Требования к содержанию документов.

42. **ГОСТ 28195-89.** Оценка качества программных средств. Общие положения.

43. **ГОСТ 28806-90.** Качество программных средств. Термины и определения.

Приложение 4

Состав услуг (сервисов), предоставляемых средой открытой системы приложениям, регламентированный стандартами POSIX OSE

В разделе 2.3.3. были определены основные принципы построения эталонной модели среды открытой системы OSE/RM. Ниже приводится состав услуг (сервисов) среды, регламентированный наиболее продвинутыми стандартами в этой области – POSIX OSE («интерфейс мобильной операционной системы, среда открытых систем»). Данный раздел базируется на следующем:

− руководстве по POSIX OSE (ISO/ IEC TR 14252:1996;

− стандартах POSIX (ISO 9945) на интерфейсы прикладного программирования (API) мобильных операционных систем;

− базовых стандартах IEEE (IEEE 1003.0, IEEE 1238, IEEE 13271).

– POSIX OSE предусматривает следующие категории услуг (сервисов) среды открытой системы, представленные в табл. П.4. 1:

Таблица П.4.1

Категории сервисов среды открытой системы

Категория услуг	Сервисы среды
Системные услуги	Языковые сервисы Сервисы ядра (операционной системы)
Услуги коммуникаций	Коммуникационные сервисы
Информационные услуги	Сервисы баз данных Сервисы обмена данными Сервисы обработки транзакций
Услуги человеко-машинного взаимодействия	Сервисы командного пользовательского интерфейса Сервисы символьного пользовательского интерфейса Сервисы оконного пользовательского интерфейса Сервисы графического пользовательского интерфейса Сервисы поддержки разработки прикладного программного обеспечения
Межкатегорийные услуги (Cross-Category Services)	Сервисы интернационализации Сервисы защиты информации (информационной безопасности) Сервисы управления системой (System Management Services)

Языковые сервисы

В целях обеспечения переносимости приложений информационных систем между разными аппаратно-программными платформами на уровне исходных кодов необходимо использовать стандартизованные языки программирования и инструментальные средства разработки прикладных программ.

Поэтому спецификации языков программирования и компиляторы рассматриваются как элементы среды открытых систем OSE, соответствующие международным стандартам. А соответствие используемых компиляторов международным стандартам должно быть подтверждено испытательными центрами, аккредитованными в установленном порядке. При этом, когда требование обеспечить переносимость приложений является обя-

зательным, необходимо избегать использования расширений языков программирования, не соответствующих общепринятым стандартам на эти языки. Эталонная модель языковых сервисов POSIX OSE включает в себя следующие языковые сервисы уровня API:

- сервисы арифметических операций;
- сервисы структуры кода программ;
- сервисы определения данных;
- сервисы представления данных;
- сервисы обработки ошибок;
- сервисы операций ввода-вывода;
- сервисы математических функций;
- сервисы логики управления программой;
- сервисы управления параллельным выполнением программ;
- сервисы низкоуровневого программирования;
- вспомогательные сервисы систем программирования.

POSIX OSE предусматривает применение следующих языков программирования для разработки приложений информационных систем:

- Ada ISO 8652 (ANSI/MIL 1815-1983);
- APL ISO 8485:1989;
- Full Basic ISO/IEC 10279:1991;
- C ISO/IEC 9899:1990;
- C++ Разрабатывает ISO/IEC JTC 1 / SC22 / WG 21;
- COBOL ISO 1989:1985 (ANSI X3.23-1985);
- Common Lisp Разрабатывает ISO/IEC JTC 1 / SC22 / WG 16 (на базе ANSI X3.226);
- Fortran ISO/IEC 1539:1991 (ANSI X.3.53-1976);
- Modula-2 ISO/IEC 10514-1;
- Pascal ISO/IEC 7185:1990; 10206:1991 Extended Pascal;
- PL/1 general purpose subset ISO 6522:1992 (ANSI X.3.74-1987);
- Prolog ISO/IEC 13211-1.1995. Part 1. General core.

Сервисы системного ядра

Сервисы системного ядра обычно являются частью операционной системы прикладной платформы, на которой выполняются приложения. Они обеспечивают переносимость и интероперабельность прикладного программного обеспечения за счет средств управления ресурсами платформы: процессами, памятью, файлами и вводом-выводом. В частности, такие сервисы относятся и к управлению выполнением приложений в распределенных системах.

Эталонная модель сервисов системного ядра POSIX OSE включает в себя следующие сервисы уровня API:

- сервисы планирования и управления выполнением процессов и потоков работ или задач;

- сервисы среды, обеспечивающие доступ приложений к информации, связанной с их выполнением (атрибуты идентификации, приоритетов, планирования, размещения в памяти, атрибуты идентификации пользователей);

- сервисы внутренней коммуникации узлов и синхронизации при одновременном выполнении нескольких приложений на одной платформе;

- обобщенные сервисы ввода-вывода (сервисы ввода-вывода часто обеспечиваются через языковые сервисы, однако язык СИ является исключением);

– сервисы хранения файлов, обеспечивающие доступ к выполняемым приложениям, иерархии директорий, субдиректорий и файлов;

– сервисы именования и директорий;

– сервисы управления событиями, обработкой ошибок и исключений;

– сервисы времени и таймеров;

– сервисы управления памятью, используемые прикладными процессами и потоками работ для управления доступным им адресным пространством;

– сервисы логического именования, позволяющие использовать системные ресурсы по их логическим именам, а не по реальным условиям именования аппаратных устройств.

– Стандарты сервисов системного ядра POSIX OSE приведены в табл. П.4.2.

Таблица П.4.2

Стандарты сервисов системного ядра

Наименование сервиса	Спецификация (стандарт) сервиса
Управление процессами	ISO/IEC 9945-1:1990 IEEE Std. 1003.2d: 1994
Управление потоками (thread management)	IEEE Std. 1003.1b: 1993 IEEE Std. 1003.1c: 1995
Среда	ISO/IEC 9945-1:1990
Внутренняя коммуникация узлов / синхронизация	ISO/IEC 9945-1:1990 IEEE Std. 1003.1b: 1993
Обобщенный ввод-вывод	ISO/IEC 9945-1:1990 IEEE Std. 1003.1b: 1993 IEEE Std. 1003.1c: 1995
Хранение файлов	ISO/IEC 9945-1:1990 IEEE Std. 1003.1b: 1993 IEEE Std. 1003.1c: 1995
Сервисы времени и таймера	ISO/IEC 9945-1:1990 IEEE Std. 1003.1b: 1993 IEEE Std. 1003.1c: 1995
Управление памятью	ISO/IEC 9945-1:1990 IEEE Std. 1003.1b: 1993 IEEE Std. 1003.1c: 1995
Логическое именование	ISO/IEC 9945-1:1990
Планирование (scheduling)	ISO/IEC 9945-1:1990 IEEE Std. 1003.1b: 1993
Асинхронный ввод-вывод	IEEE Std. 1003.1b: 1993
Синхронный ввод-вывод	IEEE Std. 1003.1b: 1993
Сигналы реального времени (или нотификация асинхронных событий)	IEEE Std. 1003.1b: 1993

Стандарт ISO/IEC 9945-1 определяет интерфейсы мобильной операционной системы, базирующейся на ОС типа Unix. Этот стандарт дополняется стандартом POSIX ISO/IEC 13210:1994 (идентичным IEEE Std. 1003.3:1991), который распространяется на методы тестирования и верификации соответствия программных средств стандартам POSIX.

Коммуникационные сервисы

Коммуникационные сервисы являются компонентами прикладной платформы. Прямой доступ к ним для прикладных программ и пользователей реализуется через стандартизованные API, хотя многие из этих сервисов выражаются в терминах «сетевых» возможностей и традиционно считаются сетевыми сервисами. Тем не менее использование коммуникационных сервисов приложениями может осуществляться без взаимодействия через физическую сеть. Коммуникационные сервисы поддерживают услуги распределенной среды, такие, как передача файлов, службы пространства имен и директорий, службы электронной почты, службы виртуальных терминалов, прозрачный доступ к файлам, коммуникации на уровне «процесс-процесс», удаленный вызов процедур, службы представления данных, службы распределенного управления временем. Прикладные программы используют коммуникационные сервисы через высокоуровневый контекстно-независимый интерфейс или через низкоуровневый контекстно-зависимый интерфейс. Сетевые протоколы, определяемые стандартами взаимосвязи открытых систем (ВОС – OSI) и стеком протоколов Интернет (т.е. TCP/IP), должны составлять базу для спецификаций интерфейсов коммуникационных сервисов. Эталонная модель коммуникационных сервисов POSIX OSE включает в себя на уровне API:

– прозрачные (для приложений) сетевые сервисы, аналогичные сервисам системного ядра, например, доступа к файлам и прозрачные в том смысле, что для их использования не требуется модифицировать приложения (например, для доступа к сетевой файловой системе;

– сервисы директорий, позволяющие приложениям находить доступные им имена и адреса объектов и служб;

– сервисы уровня «приложение-платформа», обеспечивающие относительно простые высокоуровневые API, такие, как передача файлов, электронная почта, виртуальный терминал;

– сервисы коммуникаций уровня «приложение-приложение» типа удаленного вызова процедур (RPC);

– сервисы распределенных систем, обеспечивающие возможности идентификации доступных приложениям ресурсов, динамического использования ресурсов, доступа к файлам безотносительно к их физическому размещению в системе, иметь надежные службы времени для всех ресурсов распределенной системы.

Коммуникационные сервисы определяются как на уровне API («приложение-среда»), так и на уровне интерфейсов с внешней средой (EEI).

Прикладные платформы на уровне интерфейсов с внешней средой (EEI) должны обеспечивать:

– сервисы виртуального терминала;

– сервисы удаленного выполнения команд;

– сервисы передачи файлов;

– удаленную аутентификацию (приложений и пользователей);

– удаленный доступ к данным;

– удаленный доступ к статусной информации;

– сервисы доставки электронной почты;

– сервисы директорий;

– POSIX OSE предусматривает следующие базовые стандарты коммуникационных сервисов (табл. П.4.3).

Таблица П.4.3

Стандарты коммуникационных сервисов

Категория сервиса	Стандарт (спецификация)
Прозрачные сетевые сервисы	IEEE P1003.1F (TFA API)
Сервисы доступа к директориям	ISO/IEC 9594-1:1990 (X.500) RFC 1034 (Domain naming) IEEE Std. 1224.2:1993 IEEE Std. 1327.2:1993
Сервисы «приложение-система»	IEEE Std. 1224.1:1993 IEEE Std. 1327.1:1993
Сервисы RPC	OSF DCE RPC RFC 1050
Сервисы передачи файлов	ISO 8571 ISO/IEC ISP 10607 RFC-959 (FTP) RFC-1094 (NFS) ISO/IEC ISP 9596-1:1991 (FTAM – File Transfer, Access and Management)
Сервисы простых коммуникаций	IEEE P1003.1g
Сервисы детализированных коммуникаций: слой прикладного уровня; слой транспортного уровня.	 IEEE Std. 1238.1-1994 (ACSE API) IEEE P1003.1g (Transport API)
Сервисы распределенных систем	ISO/IEC 10026-1:1992 IEEE P1003.1f (TFA API)
Сервисы X.400 API электронной почты	ITU-T X.400 IEEE Std. 1224:1993 IEEE Std. 1224.1:1993 IEEE Std. 1327:1993 IEEE Std. 1327.1:1993
Сервисы стека протоколов транспортного уровня TCP/IP	Internet Engineering Task Force RFC-793 (TCP) RFC-791 (IP)
Сервисы физических соединений	ITU-T X.25 ISO/IEC 8802.3:1993 (локальные сети Ethernet)

Сервисы баз данных

Сервисы баз данных реализуются штатными средствами СУБД, применяемых при создании информационных систем. Эталонная модель сервисов баз данных POSIX OSE на уровне API включает в себя:

− сервисы определения схемы базы данных и манипулирования схемой;

− сервисы доступа к данным и манипулирования данными;

− сервисы обеспечения целостности данных, защищающие базы данных от неправильного функционирования аппаратуры и программного обеспечения;

– смешанные сервисы, включающие администрирование привилегиями доступа, управление транзакциями, обработку исключений, управление соединениями для различных языков запросов к базам данных, формирование отчетов по обращениям к базам данных, динамические возможности интерактивного доступа конечных пользователей к манипулированию данными с последующим возвратом управления приложению, сервисы словарей данных (т.е. метаданных);

На уровне EEI POSIX OSE предусматривает протоколы удаленного доступа к данным (RDA – Remote Data Access services). Кроме того, на уровне EEI должны быть предусмотрены общие форматы обмена данными, которые требуются для обеспечения взаимодействия между двумя или более базами данных.

Сервисы управления ресурсами баз данных на уровне API для программиста прикладных программ охватывают следующие области:

– сервисы администрирования базами данных;

– сервисы извлечения данных;

– сервисы управления распределенными базами данных для их разделения на части и репликации по частям;

– сервисы поддержки гетерогенных сред баз данных;

– сервисы контроля извлечения данных для поддержки целостности;

– флаговые сервисы маркирования данных признаками наступления определенных событий;

В POSIX OSE предусмотрены следующие стандарты, распространяющиеся на область сервисов баз данных:

Структурированный язык запросов к реляционным базам данных SQL

– ISO/IEC 9075:1992 Database Language SQL, (независимый от языка программирования);

– ANSI X.3-168:1989 Embedded SQL, обеспечивающий как процедурное, так и встраиваемое связывание языка SQL с языками программирования COBOL, Fortran, Pascal, PL/1, C и Ada.

Сервисы систем словарей информационных ресурсов IRDS:

– ANSI X.3-138:1988 (требования к IRDS);

– ANSI X.3-195:1991 (IRDS export / import file format);

– ISO/IEC 10027:1990 (IRDS framework);

– ISO/IEC 10728, ANSI X.3-185:1992 (IRDS services interface, включая структуры данных, семантику сервисов, связывание с языком Pascal);

– ISO/IEC 10728:1993 (Services interface module for C language binding);

– рабочее предложение: IRDS design support for SQL applications.

Сервисы удаленного доступа к данным RDA

– ISO/IEC 9579.

Сервисы обмена данными

Сервисы обмена данными между приложениями или компонентами одного и того же приложения поддерживают обмен данными с использованием ряда различных механизмов: плоских файлов, сетевых сообщений, полей баз данных. При этом сервисы обмена данными обеспечивают:

– перемещение какой-либо одной прикладной программы и связанных с ней данных между разными операционными системами;

– перемещение данных между взаимодействующими прикладными программами в пределах одной и той же операционной среды;

– перемещение данных между взаимодействующими прикладными программами, исполняемыми в разных операционных средах;

– перемещение данных между родственными (имеющими установленные отношения), но не взаимодействующими прикладными программами, – в пределах одной операционной среды и через разные операционные среды.

В стандартах сервисов обмена данными необходимо определять наборы символов и представление данных, форматы данных и описания данных, которые совместимы для всех реализаций POSIX OSE.

POSIX OSE включает в себя стандартные сервисы обмена данными, протоколы и форматы обмена, которые распространяются на структурированные данные, включая обычные тексты, структурированные тексты, векторную и растровую графику.

Эталонная модель сервисов обмена данными POSIX OSE включает в себя на уровне API:

– форматы представления данных;

– сервисы преобразования (конвертирования) данных;

– сервисы передачи данных.

На уровне EEI эталонная модель сервисов обмена данными POSIX OSE включает в себя:

– наборы символов и представления данных;

– протоколы описания данных;

– протоколы форматов обмена данными.

Протоколы форматов данных необходимы для идентификации представления данных в форме, независимой от конкретной среды исполнения. Слой протоколов форматов данных дополняет атрибуты, которые описывают фундаментальные характеристики данных, которые должны быть известны для поиска значений данных и задаются в форматах хранения данных, «родных» для аппаратуры и программного обеспечения используемой среды. Протоколы описания данных обеспечивают возможность разделять данные между родственными приложениями, даже если эти приложения не были написаны в расчете на их взаимодействие. На уровне EEI протоколы описания данных обеспечивают средства согласования стандартным образом имен или других идентификаторов элементов данных, чтобы прикладные программы имели возможность корректно идентифицировать данные, которые были созданы «не родственными» приложениями.

В настоящее время большинство стандартов в этой области ограничены конкретными прикладными областями, и общие решения по сервисам обмена данными отсутствуют.

Для обмена документами предназначен стандарт ISO 8613, регламентирующий архитектуру офисного документа ODA, формат обмена этими документами ODIF и язык описания офисных документов ODL. Язык ODL представляет собой стандартизованный обобщенный язык разметки SGML – Standard Generalized Markup Language (стандарт ISO 8879:1986) с помощью которого кодируются документы ODA. А для обмена документами в этом случае применяется формат обмена SDIF (SGML Document Interchange Format) в соответствии со стандартом ISO 9069:1988.

Как уже было отмечено выше, в последние годы с развитием Web-технологий стал весьма популярным расширяемый язык разметки XML, стандартизацией которого занимается консорциум W3C (World-Wide Web Consortium).

В рамках POSIX OSE указываются следующие стандарты и спецификации сервисов обмена данными для различных прикладных областей (табл. П.4.4).

Таблица П.4.4

Стандарты и спецификации сервисов обмена данными для различных прикладных областей

Наименование сервиса	Стандарт / спецификация
I. Протоколы форматов данных	
Представление документов	ISO 8613 (ODA / ODIF / ODL) ISO 8879:1986 (SGML) ISO 9069:1988 (SGML / SDIF)
Представление «гипердокументов» (содержащих мультимедиа-документы и информационные объекты, объединяющие текст, речевые сообщения, статические и видеоизображения)	ISO/IEC 10744:1992 (HyTime)
Представление графики	ISO/IEC 8632:1992 (Computer Graphics Metafile – CGM) ANSI/ASME Y14.26M:1989 (Initial Graphics Exchange Specification – IGES)
Представление электронных данных для обмена EDI	ISO 9735:1988 (EDIFACT)
Представление информации о шрифтах (оформления документов)	ISO/IEC 9541 (Fonts)
Представление данных о продукции	ISO/IEC 10303 (STEP)
Представление данных об электронных схемах	IEEE Std. 1076:1993 (VHDL)
Форматы документов	ISO/IEC DIS 10179 (Document Style semantics and Specification Language – DSSSL) ISO/IEC DIS 10180 (SPDL)
Представление пространственных данных (географических карт)	FIPS Pub 173 (SDTS)
Представление данных в CASE-системах	EIA/IS 106, 107, 108, 109, 110, 111 (CASE Data Interchange Format – CDIF)
II. Протоколы описания данных	
Обмен документами	ISO 8613 (ODA / ODIF / ODL) ISO 8879:1986 (SGML) ISO 9069:1988 (SGML / SDIF) ISO/IEC DIS 10180 (SPDL)
Обмен графическими данными	ISO/IEC 8632:1992 (CGM)
Обмен электронными данными	ISO 9735:1988 (EDIFACT)
Обмен информацией о шрифтах	ISO/IEC 9541 (Fonts)
Обмен данными о продукции	ISO/IEC 10303 (STEP)
Обмен пространственными данными	FIPS Pub. 173 (STDS)
Обмен данными между CASE-системами	EIA/IS 106, 107, 108, 109, 110, 111

Сервисы обработки транзакций

Сервисы обработки транзакций (Transaction Processing – TP), рассматриваемые в данном подразделе отчета, отражают механизмы транзакций, характерные для работы с базами данных. Место этих сервисов в эталонной модели среды открытых систем долгое время было дискуссионным – их относили к областям управления базами данных, операционных систем, а в последнее время – к области коммуникаций.

Границы механизмов транзакций (как «единиц работ») определяются точками начала и окончания процесса, выполняемого прикладной программой, реализующей транзакцию. Прикладная программа может потребовать либо выполнения транзакции, либо «отката» от части выполненной работы, если она идентифицирует, что конечная точка не достигнута, и повторения процесса транзакции.

Система завершает заданную транзакцию, только в том случае, если все операции транзакции выполнены полностью и надлежащим образом. Иначе система прерывает выполнение транзакции, реализует «откат» от выполненной части работы, отмечает невыполненную транзакцию и повторяет ее с начала. Стандарт ISO/IEC 10026:1992 регламентирует механизмы обработки транзакций в распределенных системах (Distributed Transaction Processing – DTP) с учетом набора свойств, характеризующих эти механизмы – ACID (Atomicity, Consistency, Isolation, Durability). Атомарность означает, что все операции «единицы работы» либо выполнены, либо не выполнена ни одна из них. Совместимость (согласованность) означает, что операции «единицы работы», если они вообще выполняются, выполняются точно, корректно и обоснованно по отношению к прикладной семантике работы. Изоляция означает, что частные (промежуточные) результаты «единицы работы» являются недоступными, за исключением операций, которые составляют часть этой единицы. «Единицы работы», которые разделяют связанные данные, составляют серии операций. Прочность (стойкость) означает, что все эффекты завершенной «единицы работы» не подвержены влиянию ошибок любого сорта.

Сервисы обработки транзакций относятся к следующим типам:

− транзакционный доступ к одному менеджеру баз данных на одной машине;

− транзакционный доступ к менеджеру информационных ресурсов, не являющемуся средством управления базой данных (например, программного обеспечения ведения счетов на автоматических кассовых машинах);

− распределенные базы данных, т.е. базы данных, расположенные на нескольких машинах, но доступные прикладной программе так, как если бы это была одна база данных;

− онлайновая обработка транзакций (OLTP) – планирование транзакционных программ, основанных на терминальном вводе для консолидированного извлечения.

Эталонная модель сервисов обработки транзакций в POSIX OSE включает в себя набор API между транзакционной прикладной программой и менеджерами транзакций (TM), ресурсов обработки транзакций (TPRM). Между этими двумя менеджерами также определяются стандартизованные интерфейсы. Упрощенная схема обработки транзакций показана на рис. П.4.1.

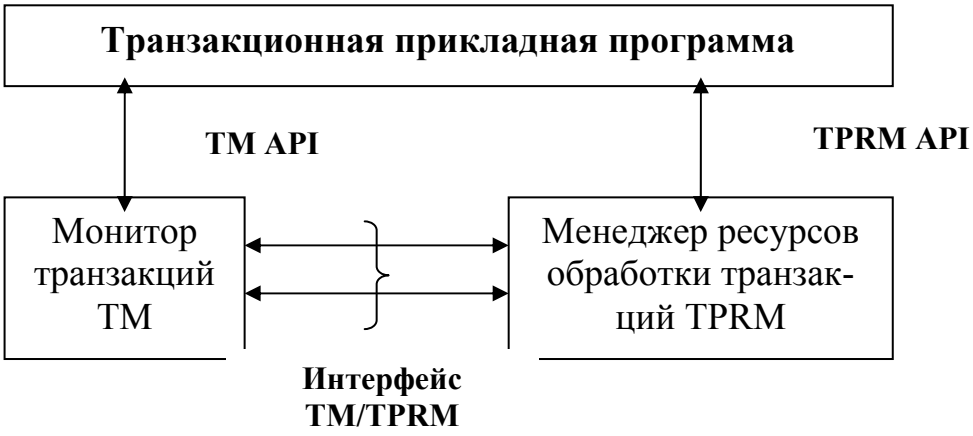

Рис. П.4.1. Упрощенная схема монитора транзакций

Работа по созданию эталонной модели сервисов обработки транзакций, дополняющей эталонную модель POSIX OSE, проводится в рамках Объединенного технического комитета по информационным технологиям ISO/IEC JTC1/SC21. До завершения этой работы в руководстве по POSIX OSE ISO/IEC 14252:1996 предложена эталонная модель сервисов обработки транзакций, включающая в себя на уровне API:

– сервисы разграничения транзакций (указатели точек начала и окончания транзакций, указатели приостановки и «отката» незавершенных транзакций;

– коммуникационные сервисы между прикладными программами обработки транзакций (вызовы других транзакционных программ, возможно удаленных, в контексте выполняемой транзакции, открытие диалогов, инициализация транзакций) отсылка и получение сообщений для взаимодействия с другими транзакционными прикладными программами, возможно, удаленными;

– презентационные сервисы (связи с терминалами) т.е. интерфейсы, обеспечивающие независимость транзакционных прикладных программ от конкретных устройств;

– сервисы планирования транзакционных программ:

– немедленного начала / окончания процессов транзакции;

– пакетного и ссылочного выполнения транзакций с организацией очередей запросов;

– сервисы платформно-независимого выполнения обработки транзакций;

– сервисы одновременной параллельной обработки;

– сервисы управления ресурсами рабочего пространства обработки транзакций.

На уровне EEI указываются интерфейсы взаимодействия двух или более платформ, прикладные программы которых участвуют в выполнении одной и той же транзакции.

Для онлайновой обработки транзакций (OLTP) должны быть специфицированы следующие виды сервисов:

– сервисы управления транзакциями, включая возможности назначения приоритетов разным транзакционным программам;

– сервисы мониторинга транзакций (сбор данных об использовании ресурсов с целью анализа производительности, организации процессов обработки транзакций, создания пулов ресурсов и т.д.);

– сервисы моделирования процессов обработки транзакций для предсказания требуемых ресурсов и ожидаемой производительности;

– сервисы директорий и пространств имен;

– тестовые сервисы, автоматически генерирующие тесты для моделирования рабочей нагрузки;

– конфигурационные сервисы, позволяющие заменять или добавлять в систему транзакционные прикладные программы без необходимости закрывать среду выполнения, добавлять сервисы для управления пулами ассоциаций (соединения) транзакций;

– классы проверки соответствия, определяющие формальные подмножества функций OLTP, такие, которые позволяли бы проверять соответствие реализаций спецификациям OLTP и для персональных компьютеров / рабочих станций, и для серверов / хост-машин.

В POSIX OSE предусмотрены ссылки на следующие стандарты и спецификации:

– ISO/IEC 10026, Parts 1-3 (OSI DTP);

– ISO/IEC 10026:1992 (Platform-independent exception handling);

– ISO/IEC 10026-2:1992 (Concurrent execution);

– X/Open TP, спецификация XA, описывающая интерфейс между менеджером транзакций и менеджером ресурсов.

Услуги человеко-машинного взаимодействия

Сервисы командного пользовательского интерфейса

Командные языки пользовательского интерфейса в POSIX OSE указаны на основе известных реализаций ОС типа Unix. На уровне API эталонная модель сервисов командного пользовательского интерфейса включает в себя:

– сервисы командных устройств ввода-вывода;

– сервисы обнаружения / коррекции ошибок;

– сервисы прекращения / приостановки (выданных) команд;

– сервисы оболочек (интерпретаторов команд) и утилит (shell and utility) командного языка.

На уровне EEI включаются:

– сервисы оболочек и утилит;

– сервисы утилит «переносимости» пользователей.

Единственный формальный стандарт, распространяющийся на область командного пользовательского интерфейса, – это ISO/IEC 9945-2:1993, идентичный по содержанию с IEEE Std. 1003.2-1992.

Стандарт IEEE Std. 1003.2d:1994, основанный на системе сетевых очередей NQS (Network Queuing System), расширяет ISO/IEC 9945-2 на область определения утилит, интерфейсов системного администрирования и протоколов прикладного уровня, адресованных следующим областям: утилиты для подписки и управления запросами, интерфейсов системного администрирования для создания, управления и авторизации сетевых очередей и систем пакетной обработки данных, сетевые протоколы прикладного уровня. При этом в руководстве даются ссылки на документацию по ОС Unix 4.3 BSD, в которой реализованы указанные сервисы.

Рабочая группа IEEE P1003.6 разрабатывает дополнение к ISO/IEC 9945-2 в виде проекта IEEE P 1003.2c, направленное на усиление требований к безопасности. Команды пользовательского интерфейса в традиционных системах типа Unix (uux, rsh, rcp, rlogin, rwho, ruptime и анонимный ftp). Они специализированы в документах OSF AES-OSC, X/Open XPG4. На приложения Интернет в данной области распространяются: RFC-959 (ftp), RFC-854 (Telnet), RFC-821 (SMTP).

Сервисы пользовательского интерфейса, основанного на символах

Определяют интерфейсы API с символьными терминалами и взгляд на взаимодействие между пользователем и терминальным оборудованием. Внимание к этой группе сервисов сохраняется, несмотря на рост популярности графических оконных интерфейсов, т.к. сервисы, основанные на символах, менее требовательны к ресурсам компьютеров (памяти, процессорам, разрешающей способности дисплеев). Они хорошо подходят для приложений, не требующих сложного взаимодействия с пользователем, и применяются тогда, когда суммарная стоимость дисплейных устройств в системе является главным требованием.

Символьные терминалы пока еще составляют значительную часть оборудования прикладных информационных систем, а гибкость пользовательских интерфейсов, основанных на заполнении форм (form based interface), позволяет пользователям применять структурированные методы для ввода данных и комбинировать их с форматированием документов при выводе. При этом сервисы удаленных прикладных платформ с услугами виртуального терминала в данной категории не рассматриваются, поскольку услуги виртуального терминала больше относятся к категории коммуникационных сервисов, а не к специфичным API приложений.

Эталонная модель сервисов символьного пользовательского интерфейса в POSIX OSE включает в себя:

- на уровне API:
- сервисы управления презентацией;
- сервисы управления экранными формами;
- сервисы управления экранами;
- на уровне EEI:
- сервисы описания (look and feel) символьного пользовательского интерфейса;
- сервисы описания (look and feel) блочного пользовательского интерфейса (block-mode);
- сервисы описания (look and feel) интерфейса, основанного на формах;
- сервисы управления устройствами ввода;
- интерфейсы сервисов управления, ориентированные на устройства типа «мыши»;
- протокольные сервисы терминалов и принтеров, скрытые от приложений.

В POSIX OSE предусмотрены ссылки на следующие стандарты символьного пользовательского интерфейса (табл. П.4.5.).

Таблица П.4.5

Стандарты символьного пользовательского интерфейса

Категория сервиса	Спецификация
Система управления интерфейсом форм FIMS (Forms Interface Management System)	ISO/IEC 11730:1994
Терминальные протоколы	ISO/IEC 11730:1994
Клавиатуры (расположение клавиш)	ISO/IEC 9995:1994
OCR	ANSI X3.62:1987
Windowing API (базирующееся на XVT Portability Toolkit, поддерживающее символьный пользовательский интерфейс, рассчитанный на работу с «мышью»	X/Open XPG4

Сервисы оконного пользовательского интерфейса

Сервисы оконных систем являются наиболее современными средствами поддержки прямого человеко-машинного взаимодействия. Пользовательские интерфейсы в этом случае базируются на наборах меню или указаниях, которые пользователь дает с помощью визуальных дисплеев, накладных шаблонов графических планшетов, светового пера. Доступность дешевых растровых рабочих станций и персональных компьютеров привела к широкому применению графических пользовательских интерфейсов (GUI), оконных технологий построения пользовательских интерфейсов, общих команд типа «Файл», «Сохранить», «Сохранить как», «Новое» и т.д. с использованием устройств типа «мышь», «трекболл», таблет (накладной шаблон графического планшета). Во многих случаях такие технологии стали уже стандартами «де-факто» и неформально приняты сообществом пользователей информационных систем.

Стандарты интерфейсов графических оконных систем распространяются на следующие области:

- протоколы управления окнами на локальных и удаленных дисплейных устройствах;
- API для этих протоколов;
- элементы управляемости графических оконных систем, определяющие общие подмножества описаний (look and feel), такие, как появление объекта, позиционирование его на экране, поведение объектов в окнах графического экрана;
- спецификации API для соответствующего связывания операций над объектами оконного интерфейса с языками программирования;
- интерфейсы командных языков, которые могут вводиться пользователем интерактивно из хранимых процедур;
- спецификации API и соответствующие языковые связывания, необходимые для поддержки символьных (не растровых) терминалов;
- спецификации API и соответствующие языковые связывания, необходимые для трансляции, манипулирования и управления командными утверждениями или сообщениями со стороны пользователя.

Эталонная модель сервисов оконного пользовательского интерфейса в POSIX OSE включает в себя:

- на уровне API:
- инструментарий оконных сервисов;
- базовые оконные сервисы;
- диалоговые сервисы;
- на уровне EEI:
- описание (look and feel) оконной системы и ее управляемости;
- интерфейсы к информации, представленной в окнах;
- протоколы сетевых интерфейсов.

Примечание. Для упрощения понимания термина «управляемость графического пользовательского интерфейса часто приводится аналогия со средствами управления автомобилем, ставшими привычными для водителей.

Сервисы оконных систем обеспечивают управляемый интерфейс между прикладными программами, специфичными для конкретных областей применения, и программным обеспечением, специфичным для выполнения функций пользовательского интерфейса. Пользователями этих сервисов являются все пользователи систем, удовлетворяющих

стандартам POSIX, а также персонал сопровождения процессоров и служб коммуникаций графических оконных систем.

В POSIX OSE предусмотрены стандарты сервисов графического оконного пользовательского интерфейса – IEEE Std. 1295:1993, который основан на широко известной для ОС типа Unix X Window System и библиотеке X lib. В нем определены слой инструментальных средств X Window System, основанных на спецификациях пользовательского интерфейса OSF Motif (табл. П.4.6).

Таблица П.4.6

Стандарты сервисов графического оконного пользовательского интерфейса

Категории сервисов	Спецификации
Базовые оконные сервисы	X Window System X lib
Инструментарий оконных сервисов	IEEE Std. 1295:1993 IEEE P 1201.01 X Window System X Toolkit
Сервисы диалога	IEEE P 1201.02
Интерфейсы «пользователь-система» и символы	ISO/IEC 9995:1994
Представление, диалог, основанный на формах, и взаимодействие с пользователем на базе окон	ISO/IEC 11730:1994
Сервисы человеко-машинного диалога в программной инженерии	ISO 9241
Единообразные API оконного интерфейса	IEEE P 1201.1
Рекомендации руководства управляемости графического пользовательского интерфейса	IEEE P 1201.2
Взаимодействие между приложениями, использующими X Window System	X Window System (X Protocol)

Спецификации языкового связывания стандартов оконного интерфейса PHIGS, GKS, GKS-3D, CGI, IEEE P 1201.1, IEEE Std. 1295 существуют для языка программирования C.

Рабочая группа IEEE P 1201.1 разрабатывает стандарт, определяющий единообразные API, которые могут быть применены при разработке прикладных программ, переносимых между различными оконными системами, такими, как OSF/Motif, Microsoft Windows, OS/2 Presentation Manager и Apple Macintosh.

Открытые публичные спецификации сервисов оконного пользовательского интерфейса не являются частями POSIX OSE. Однако они представляют значительный интерес для анализа опыта уже проведенных работ, чтобы заполнить пустоты в POSIX OSE, которые сейчас имеются. К ним относятся: разработанные в Массачусетском технологическом институте протокол X Window System и функциональный интерфейс X lib к этому протоколу.

Эта разработка продолжается под эгидой консорциума X Consortium, объединяющего заинтересованные организации компьютерной индустрии. Документами этого консорциума являются: X lib, спецификации X Protocol и X Toolkit.

X lib – это C-language X Interface. Он является общим компонентом для X Widow System и размещен как резидент на всех основанных на X системах.

Хотя X определен принципиально как сетевой протокол, разработчики прикладных программ не применяют прямых интерфейсов к X-протоколу. Вместо этого они обращаются к X-протоколу через X lib.

В X Window System используется клиент-сервисная модель. Клиентом является прикладная программа, выполняемая на любом узле сети и использующая дисплей. Для реализации клиентских функций используются вызовы X lib, чтобы генерировать протоколы. X-сервер является программным средством, которое принимает протоколы, посланные клиентом, и обрабатывает их для отображения на экране.

X/Open в сотрудничестве с X Consortium опубликовал спецификации X Window System Protocol (описание и определение X-протокола), X Window System File Format and Application Conventions, X Toolkit Intrinsics, X lib – C language binding.

Сервисы графического пользовательского интерфейса

Эти сервисы играют важную роль в POSIX OSE, поскольку они используются сегодня во многих областях применения в промышленности, бизнесе, правительственных учреждениях, образовании, предпринимательстве, а с недавних пор и в домашних условиях, число таких приложений очень быстро растет с увеличением их графических возможностей. К таким областям относятся собственно приложения, обеспечивающие графический пользовательский интерфейс, системы автоматизации проектирования, системы электронных публикаций, системы моделирования и визуализации результатов для научных исследований, искусство и управление технологическими процессами.

2D и 3D графика стала привычной технологией человеко-машинного взаимодействия. Стандартизация графических сервисов дает значительные выгоды разработчикам прикладного программного обеспечения и системным интеграторам Поэтому такие сервисы являются предметом для включения в POSIX OSE.

Эталонная модель сервисов графического пользовательского интерфейса в POSIX OSE сложилась благодаря тому, что за последние 10 лет были разработаны и реализованы многие стандарты машинной графики. Основным здесь является стандарт ISO/IEC 11072:1992, определяющий эталонную модель CG/RM, т.е. структуру, в которой могут быть размещены как существующие, так и будущие стандарты машинной графики. Она определяет 5 абстрактных уровней, называемых средами: – уровень конструкций, виртуальный уровень, уровень взгляда пользователя, логический уровень и уровень реализации. Операции над элементами данных для каждой среды должны быть выражены в терминах выходных примитивов, которые позволяют сделать из них композицию, представляемую оператору.

В эталонную модель графических сервисов включены API служба сервисов для их испытаний и опробований, API метафайла машинной графики, охватывающего пять уровней абстракции используемых данных, названных средами: среду конструкции, среду виртуализации, среду обзора, логическую среду и среду реализации. Между приложением и средой машинной графики единственным интерфейсом является интерфейс среды конструкции. Интерфейс метафайла аудита сервисов поддерживает записи, позволяющие осуществить импорт и экспорт всех или части используемых элементов данных. Интерфейс с оператором обеспечивается средой реализации.

Эталонная модель сервисов графического пользовательского интерфейса в POSIX OSE включает в себя:

- на уровне API:
- сервисы 2D графики;
- сервисы 3D графики;
- сервисы интерфейсов устройств;
- сервисы обработки изображений;
- на уровне EEI:

– форматы файлов;

– протоколы обмена.

Каждый стандарт API имеет присоединенный к нему стандарт языкового связывания, что позволяет обеспечить доступ к функциям приложений из разных языков программирования. Примерами API для графических сервисов являются GKS, PHIGS, PIK и т.д.

POSIX OSE предусматривает в своем составе следующие стандарты машинной графики:

– ISO/IEC 9592, Parts 2,3 (PHIGS – programmers Hierarchical Interactive Graphics Standard), определяет формат архивного файла для хранения и передачи структур PHIGS;

– ISO/IEC 9593 (language-binding standards), устанавливающий связывание с языками Ada, C, Fortran, Pascal, C-LISP;

– ISO 7942:1985 (GKS)

– ISO/IEC 9636:1991 (CGI – Computer Graphics Interface);

– ISO 8805:1988 (GKS-3D);

– ISO/IEC 12087, 12088 (Image Processing and Interchange);

– ISO/IEC 8632:1992 (CGM);

– ISO/IEC 10641:1993 (Conformance testing).

Расширение сетевого протокола для X Window System в случае 3D-графики применительно к PHIGS дано в публичной спецификации PEX (PHIGS Extensions tool). Эта спецификация опубликована Массачусетским технологическим институтом. Реализация PEX-SI сделана в 1991 г.

Сервисы поддержки разработки прикладного программного обеспечения

Сервисы, описываемые в данном разделе, обеспечивают разработку прикладного программного обеспечения и его выполнение на прикладной платформе. Традиционными средствами, реализующими эти сервисы, являются текстовые редакторы, компиляторы и компоновщики.

Эталонная модель определяет интерфейсы сервисов поддержки разработки приложений, которые влияют на переносимость приложений и системную интероперабельность.

Сервисы этих интерфейсов:

– сервисы для работы с исходными текстами (ISO/IEC 9945-2:1993);

– сервисы подготовки исполнения (ISO/IEC 9945-2:1993);

– сервисы исполнения приложений.

Стандарт ISO/IEC 9945-2:1993, являющийся частью POSIX OSE, описывает некоторые средства, поддерживающие разработку приложений, включая интерфейсы к некоторым компиляторам и утилитам разработки ПО. Кроме того, в стандарте рассмотрены вопросы связывания услуг поддержки разработки ПО с языками программирования.

Примерами сервисов, не охваченных стандартами, являются:

– синтаксическое редактирование;

– поддержка стиля кодирования;

– конфигурационное управление;

– сопровождение исходного кода;

– сервисы исполнения приложений;

– взаимодействие сред разработки приложений.

Область разработки прикладного ПО, которая также не рассматривается в POSIX OSE, – это т.н. среда разработки приложений (Software Development Environment – SDE). Такие среды обладают возможностями спецификации, планирования, разработки, тести-

рования и сопровождения сложных программных комплексов на всех этапах жизненного цикла этих программных комплексов. Стандартизацией этого направления занимается Институт стандартов США (NIST) – форум Integrated Software Engineering Environments (ISEE). Кроме того, вопросы стандартизации SDE рассмотрены в ISO/IEC 13719-1: 1995 – Portable Common Tool Environment (PSTE).

Межкатегорийные услуги

Эталонная модель POSIX OSE определяет набор концептуальных блоков, из которых строится система и которые описываются в POSIX OSE. Каждый блок обеспечивает специфический набор интерфейсов для доступа к соответствующим сервисам. Однако, существует другой класс сервисов и требований, которые могут влиять на базовые архитектурные блоки – они определяются как межкатегорийные услуги POSIX OSE.

Межкатегорийные услуги являются набором средств и/или особенностей, которые в случае их применения могут оказывать непосредственное влияние на функционирование одного или более системных компонент, но которые сами являются самостоятельными компонентами среды открытых систем. Примерами межкатегорийных сервисов могут быть интернационализация, защита информации (обеспечение информационной безопасности), администрирование (управление и контроль функционирования) и др.

Интернационализация

Интернационализация заключается в способности среды поддерживать естественные языки (кодовые наборы) и культурные соглашения (форматы дат, числовых величин, обозначения валют и др.).

В соответствии с этим сервисы интернационализации осуществляют поддержку и управление:

- кодовыми наборами и представлением данных;
- культурных соглашений;
- естественных языков.

Кодовый набор для английского языка определен ISO 646:1991, в котором предусмотрена 7-битная схема кодирования каждого из 95 знаков. 8-битная схема кодирования английского языка для стран западной Европы, Америки, Австралии и других англоговорящих стран определена в ISO 8859-1:1987. 8-битная схема кодирования используется для кодирования языков Восточной Европы, Греции, России, арабских языков (в соответствующих национальных стандартах). Япония, Китай, Корея и Тайвань используют 16 или более бит для идентификации знаков.

В связи с использованием различных схем кодирования, прикладная платформа должна иметь возможность поддерживать их все (для корректного отображения данных на дисплее, для печати и запоминания на внешних носителях).

Прикладная платформа (среда) должна обладать свойством независимости от кодовых наборов, т.е. способность ввода, манипулирования, извлечения, передачи и представления данных независимо от используемой схемы кодирования. Это включает 7,8,16-битные и мультиоктетные кодовые наборы.

Прикладная платформа должна иметь возможности поддержки и доступа к центральному репозитарию кодовых наборов. Этот репозитарий содержит все кодовые наборы, используемые платформой, а также релевантную информацию о них.

Основными сервисами, обеспечивающими культурные соглашения, являются:

– поддержка репозитария культурных соглашений (запоминание и доступ к правилам и объектам культурных соглашений), который должен содержать спецификации и правила представления для форматов даты и времени, перечисления недель и дней, числовых полей, денежных символов, размеров бумаги для печати;

– селекция культурных репозитариев – эти репозитарии должны быть доступны всем приложениям с возможностью выбора репозитария из прикладной платформы и перехода к другим репозитариям по запросам пользователей или приложений;

– обеспечение правил сортировки – кроме общих правил двоичной и знаковой сортировки, прикладная платформа должна иметь возможность сортировки данных в соответствии с соглашениями, хранимыми в культурном репозитарии.

POSIX OSE должна давать возможность выбора пользователем естественного языка для диалога с системой и приложениями. Это подразумевает поддержку для разных естественных языков меню выбора, сообщений об ошибках, документации и др. Кроме того, должна быть обеспечена поддержка разных выбранных языков для взаимодействия с прикладной платформой и языком конкретного приложения. Для обработки текстов и слов эти сервисы включают расстановку переносов и орфографический контроль в разных языках.

Все сервисы, упомянутые в данном разделе, должны быть доступны приложениям по запросам через API. Эти API могут быть структурированы также как соответствующие сервисы, описанные в настоящем разделе.

Некоторые стандарты, поддерживающие интернационализацию, приведены в табл. П.4.7.

Таблица П.4.7

Стандарты, поддерживающие интернационализацию

Сервисы	Стандарты
1. Наборы знаков / представление данных	ISO 646:1991, ISO/IEC 2022:1994, ISO 4031:1987, ISO 4217:1990, ISO 4873:1991, ISO 6093:1985, ISO/IEC 6429:1992, ISO 6936:1988, ISO/IEC 6937:1994, ISO/IEC 7350:1991, ISO 8601:1988, ISO 8859, ISO/IEC 10367:1991, ISO/IEC 10646-1:1993, ITU-T T.61
2. Культурные соглашения	ISO 2014:1976, ISO 3307:1975
3. Поддержка естественных языков	ISO/IEC 9995:1994

Защита информации

Обеспечение информационной безопасности является одной из важнейших особенностей информационной системы. Это достигается реализацией четырех основных свойств информационной безопасности:

– конфиденциальность – защищенность системы от неавторизованного доступа к данным;

– целостность – защита от несанкционированного изменения и удаления данных;

– доступность – система должна гарантировать авторизованным пользователям доступность данных и их обработку;

– неотказуемость – система должна обеспечивать ответственность пользователей за свои действия.

Эталонная модель POSIX OSE содержит коллекции сервисов со стороны API и EEI, каждый из которых может быть реализован соответствующей компонентой безопасности.

Эти компоненты могут функционировать самостоятельно или в совместно с другими компонентами защиты. Например, идентификация и аутентификация пользователя может включать в себя некоторые сервисы пользовательского интерфейса, коммуникационные сервисы и сервисы баз данных.

Свойства информационной безопасности поддерживаются сервисами реализующих их механизмов безопасности. Эти сервисы могут включать:

- идентификацию и аутентификацию;
- управление доступом;
- неотказуемость и аудит;
- доступность;
- защиту при обмене данными.

Со стороны API интерфейсы к сервисам безопасности для POSIX OSE определены в ISO/IEC 9945-1:1990. Здесь не рассматриваются аспекты безопасности, реализуемые архитектурными особенностями, и безопасность распределенных и сетевых систем. Они включают в себя интерфейсы аудита и привилегий, дискретного контроля доступа (discretionary access control – DAC), мандатного контроля доступа (mandatory access control – MAC), и интерфейсы присвоения информационных меток (information labels – IL).

EEI-сервисы не входят в число интерфейсов POSIX, определенных в IEEE P1003.1e. Однако, внешние сервисы защиты являются предметом рассмотрения рабочей группы в IEEE P1003.22.

Сюда входят:

- пользовательский интерфейс (human-computer interface – HCI):
- имя пользователя;
- надежные маршруты;
- изменение пользовательских паролей;
- администрирование;
- коммуникации/обмен данными:
- протоколы для удаленной аутентификации и авторизации, включая синтаксис для атрибутов безопасности объектов и субъектов;
- администрирование безопасности;
- защищенные коммуникации (целостность/защита диалогов);
- защищенные коммуникации с не открытыми системами;
- UDB и ACL форматы для взаимодействия.

Стандарты, поддерживающие сервисы безопасности для POSIX OSE, приведены в табл. П.4.8.

Таблица П.4.8

Стандарты, поддерживающие сервисы безопасности

Сервисы	Спецификации
Конфиденциальность и целостность	IEEE P1003.1e (устаревший стандарт)
Управление доступом	ISO 8613 Draft Addendum (устаревший стандарт)
Идентификация и аутентификация	ISO/IEC 9594-8:1990
Роли защиты и ответственность	ECMA 138
Сетевая защита	ISO/IEC 7498-2:1989
Обмен данными	отсутствуют

Администрирование

Информационные системы состоят из широкого спектра разнородных ресурсов, которыми нужно эффективно управлять, чтобы система успешно функционировала. Несмотря на то, что конкретные ресурсы могут сильно отличаться друг от друга, могут быть использованы унифицированным способом базовые концепции администрирования. Адаптация общей модели администрирования дает возможность администраторам управлять ресурсами без необходимости различных технологий каждого ресурса.

Целью стандартов системного и сетевого администрирования является обеспечение переносимости и интероперабильности технологий администрирования. Важной особенностью стандартов в этой области является то, что они не диктуют конкретную политику администрирования, а дают возможность использования разнообразных политик, выбранных на основе специфических потребностей в данной области конкретных систем.

Структуризация функциональности администрирования может быть сделана различными способами.

Одним из способов может быть разделение по видам администрируемых ресурсов, например:

- администрирование печати;
- администрирование программного обеспечения;
- управление пользователями;
- сетевое администрирование;
- управление хостами;
- управление процессором;
- управление файловой системой;
- управление устройствами.

Можно разделять функциональность администрирования в соответствии с некоторыми общностями для перечисленных выше ресурсов:

- конфигурационное управление;
- управление производительностью;
- управление сбоями;
- управление учетом;
- управление безопасностью.

Эталонная модель POSIX OSE для системного администрирования приведена на рис. П.4.2.

Рис П.4.2. Эталонная модель администрирования

Специфические задачи администрирования решаются с помощью соответствующего программного обеспечения, которое, в свою очередь, использует сервисы прикладной платформы. Кроме того, сервисы прикладной платформы также содержат внутри себя некоторые компоненты, ориентированные на управление.

Прикладная платформа включает также объекты управления. Объекты управления – это абстрактные представления ресурсов, которыми надо управлять. Эти абстракции поддерживаются методологией доступа к тем свойствам ресурсов, которыми надо управлять. Использование этих абстракций позволяет управлять широким диапазоном ресурсов унифицированным способом.

Внешняя среда включает администраторов, которые используют программное обеспечение администрирования через интерфейс с командной строкой. Этот интерфейс (и графический пользовательский интерфейс) обеспечивает переносимость администраторов. Внешняя среда включает также средства для обмена информацией вне модели. Такой информационный обмен может иметь место через внешние сети или переносимые носители.

В то время как пользователями сервисов администрирования являются администраторы, имеющие доступ к этим сервисам через EEI, существует ряд сервисов управления, которые поддерживаются компонентами платформы. Эти сервисы вместе с ассоцииро-

ванными с ними API рассматривались в предыдущем разделе, где описывались основные сервисы POSIX OSE.

Ниже перечислены сервисы, реализуемые соответствующими компонентами программного администрирования, и доступные для администраторов через EEI (классификация, как указывалась выше не единственная, а предлагаемая для представления этих сервисов POSIX OSE):

– конфигурационное управление, которое включает четыре базовых функции – идентификацию (ресурсов системы, которыми надо управлять), управление (возможность согласования и «замораживания» элементов конфигурации (configuration items – CI) и затем выполнение изменений только в соответствии с поименованными полномочиями), статусный учет (хранение и журналирование всей текущей и исторической информации о каждом CI) и верификация (проверка соответствия между всеми CI и их авторизованными состояниями, как записано в базе данных конфигурационного управления (configuration management database – CMDB));

– управление программным обеспечением (инсталляцией, распределением, оценкой конфигурации, верификацией конфигурации);

– инициализация системы, реинициализация (рестарт) системы, выключение системы;

– управление лицензиями;

– сервисы печати;

– управление носителями, резервное копирование, архивирование и восстановление;

– он-лайновое управление дисками;

– планирование заданий;

– управление пользователями;

– журналирование ресурсов и их использования;

– управление производительностью;

– управление загрузкой системы (по производительности, работами и т.д.);

– управление неисправностями;

– управление безопасностью.

Стандарты системного администрирования, которые являются частью POSIX OSE, представлены в табл. П.4.9.

Таблица П.4.9

Стандарты системного администрирования

Сервисы	Спецификации
Управление конфигурацией	ECMA TR-47
Управление ПО	IEEE Std. 1387.2-1995
Системная инициализация / реинициализация / выключение	ISO/IEC 9945-1:1990, XPG4, OSF AES-OSC
Печать	IEEE P1387.4
Администрирование пользователей	IEEE P1387.3
Журналирование	ISO/IEC DIS 10164-10
Управление производительностью	ANSI X3.141-1987
Управление неисправностями	ISO/IEC 10164-4:1992, ISO/IEC 10164-5:1993, ISO/IEC 10164-6:1993, ISO/IEC 10164-7:1992

Приложение 5

Фрагменты профилей автоматизированной банковской информационной системы (примеры)

Профиль приложения

Таблица П.5.1

Фрагмент профиля приложений

Функции	Задачи	Базовые нормативные документы
Ведение статистической отчетности	Сбор, систематизация, обработка и хранение статистической информации по РКЦ по части форм статистической отчетности ф.700д, ф.721, ф.722, ф.746, ф.707.	1. Письмо ЦБ РФ № 168 от 12.05.95 «О статистической отчетности по валютному регулированию и валютному контролю» (с изм. от 03.04.96) 2. Письмо ЦБ РФ № 148 от 09.03.95 «Об оперативном отчете об остатках средств на счетах местных органов власти и государственных внебюджетных фондов РФ по форме 746Д» (с изм. от 21.08.96.)

Требования к программным средствам промежуточного слоя
Требования к среде распределенных вычислений

В профиль среды АИБС входят спецификации программных средств промежуточного слоя, предназначенных для обеспечения взаимодействия распределенных клиент-серверных приложений.

Среда распределенных вычислений АИБС должна соответствовать спецификациям DCE 1.1, разработанным консорциумом OSF (Open Software Foundation) и поддерживаемым консорциумом Open Group (таблица П.5.2.).

Таблица П.5.2

Фрагмент профиля среды распределенных вычислений

Функции базовых служб среды распределенных вычислений	Нормативные документы
1. Служба вызова удаленных процедур	Open Group. C 706 (1997) DCE 1.1. Remote Procedure Call
2. Функции поддержки потоков, обеспечивающие создание, управление и синхронизацию множественных параллельно выполняемых путей доступа внутри одного прикладного процесса	IEEE POSIX 1003.4
3. Служба каталогов ячейки DCE, реализующая систему имен доменов Domain Name System (DNS)	DCE 1.1. Directory Services
4. Служба времени	DCE 1.1. Time Services Specification
5. Распределенная файловая система	DCE 1.1. Distributed File Service Specification
6. Служба безопасности, реализующая функции мандаторной защиты сообщений между клиентом и сервером и списки контроля доступа	DCE 1.1. Authentication and Security Services

Требования к среде распределенной обработки транзакций

Программные средства промежуточного слоя, предназначенные для распределенной обработки транзакций (мониторы транзакций), должны соответствовать спецификациям среды распределенной обработки транзакций DTP консорциума X/Open. Мониторы транзакций, применяемые в АБИС, должны работать в среде распределенных вычислений, соответствующей спецификациям DCE.

Средства распределенной обработки транзакций, применяемые в составе среды АБИС, должны соответствовать спецификациям консорциума Open Group, входящим в набор спецификаций Open Group CAE Specification (таблица П.5.3.)

Таблица П.5.3

Фрагмент профиля среды распределенной обработки транзакций

Функции базовых служб среды обработки распределенных транзакций	Нормативные документы
1. Эталонная модель обработки распределенных транзакций	Open Group. G 504. Distributed Transaction Processing. Reference Model. Version 3
2. Интерфейс между монитором транзакций и менеджером ресурсов	Open Group. C 423. Distributed TP: The XA+ Specification, Version 2
3. Интерфейс прикладного программирования к услугам обработки удаленных транзакций через протокол уровня представления ВОС	Open Group. C 409. X/Open ACSE/Presentation: Transaction Processing API (XAP-TP)
4. Интерфейс приложений с менеджером коммуникационных ресурсов при обработке распределенных транзакций	Open Group. C 419. Distributed TP:The XCPI-C Specification, Version 2
5. Интерфейс обращения приложений к монитору транзакций для разграничения глобальных транзакций и управления их завершением	Open Group. C 504. Distributed TP:The TX (Transaction Demarcation) Specification
6. Интерфейс между приложением и менеджером коммуникационных ресурсов для транзакционного вызова удаленных процедур	Open Group. C 505. TX RPC
7. Интерфейс между приложением и менеджером коммуникационных ресурсов в клиент-серверной архитектуре	Open Group. C 506. Distributed TP: The XATMI Specification
8. Структурированный язык определения транзакций	Open Group. C 611. Structured Transaction Definition Language (STDL)

Требования к операционной среде АБИС
Операционные системы серверов

1. Серверы приложений, серверы баз данных, серверы обмена сообщениями электронной почты, применяемые в АБИС, должны функционировать под управлением операционной системы UNIX, соответствующей стандартам ISO/IEC POSIX.

2. Операционные системы серверов должны поддерживать сетевые протоколы транспортного и сетевого уровней TCP/IP.

3. Операционные системы серверов должны соответствовать единой спецификации Unix (профилю Unix 95) консорциума X/Open (таблица П.5.4.).

Таблица П.5.4

Фрагмент профиля операционной среды

Функции	Нормативные документы
Системные вызовы и библиотеки	XPG4 Internationalised System Calls and Libraries Extended
Команды и утилиты	XPG4 Commands and Utilities, v.2
Язык С	XPG4 C language
Службы транспорта	XPG4 Transport Service (XTI)
Сокеты	XPG4 Sockets
Интерфейсы терминалов	XPG4 Terminal Interfaces

Спецификации профиля Unix 95 связаны со следующими документами консорциума X/Open:

- X/Open C410. X/Open Transport Innterface (XTI). Version 2;
- X/Open C 610. X/Open Curses, Issue 4, Version 2;
- X/Open C 434. System Interfaces and Hedders, Issue 4, Version 2;
- X/Open C 204. System Interface Definitions, Issue 4, Version 2;
- X/Open C 214. Programming Languages. Issue 3;
- X/Open X 951. Profile Definition (Part 3 of How to Brand – What to Buy);
- X/Open C 438. Networking Services, Issue 4;
- X/Open C 436. Commands and Utilities, Issue, Version 2;

Применяемые в АБИС операционные системы серверов должны быть аттестованы на соответствие профилю Unix 95. Консорциум Open Group, в который входит в настоящее время X/Open, производит сертификацию операционных систем типа Unix с помощью тестовых наборов XPG4 Test Suites.

Для проверки соответствия спецификациям профиля Unix 95 применяются следующие тестовые наборы:

- VSX 4;
- VSU 4;
- VSC 4;
- Plum Hall v.7.00 или Perennial ACVS 4.3;
- VST 4.

Операционные системы клиентов

АРМ пользователей в АБИС могут работать в операционной системе Windows 98. Операционные системы клиентов в среде распределенной обработки АБИС должны соответствовать подмножеству спецификаций Win 32 фирмы Microsoft, соответствующему интерфейсам Windows 98.

Требования к телекоммуникационной среде АБИС

Требования к интерфейсам и протоколам телекоммуникационной системы АБИС

Телекоммуникационная система (ТС) АБИС должна представлять собой ведомственную региональную защищенную сеть передачи банковской информации. ТС АБИС должна обеспечивать обмен информацией кредитных организаций с РКЦ (ГРКЦ) региона,

удаленных РКЦ с ГРКЦ (ГУ ЦБ РФ), ГРКЦ (ГУ ЦБ РФ) с ГЦИ ЦБ РФ и Департаментами ЦБ РФ, а также ГУ ЦБ РФ с администрацией региона, налоговой службой, органами госстатистики, таможней.

При разработке конкретного проекта АБИС должны быть предусмотрены варианты построения ТС, рассчитанные на различные первичные каналы связи, доступные в каждом регионе, где внедряется АБИС, в настоящее время и в перспективе.

Нормативные требования к программным и техническим средствам ТС АБИС при внедрении в конкретных регионах России должны учитывать решения, принятые ЦБ РФ при построении Единой телекоммуникационной банковской системы (ЕТКБС).

ТС АБИС должна обеспечивать следующие функции, необходимые для работы функциональных систем АБИС:

– передачу сообщений (электронная почта);

– передачу файлов;

– доступ к внешним информационным сетям и сетям передачи данных общего пользования;

– телекоммуникационное взаимодействие «клиент-сервер».

Функции электронной почты в ТС АБИС должны соответствовать требованиям, представленным в таблице П.5.5.

Таблица П.5.5

Фрагмент профиля телекоммуникационной среды

Функции	**Нормативные документы**
Электронная почта: - по протоколу X.400;	 ИСО/МЭК 10021 -8,9. ИТ. Передача текста. Системы обработки сообщений. (ITU-T X.400)
- по протоколам Internet;	IETF RFC 821, 822. SMTP RFC 2045 MIME RFC 1734 POP3 RFC 1730 IMAP4
- по протоколу системы РЕМАРТ	MHS Novell

Авторы

Бойченко Александр Викторович – ст. научный сотрудник Института системного программирования РАН

Кондратьев Вячеслав Константинович – заведующий Кафедрой открытых информационных систем Московского государственного университета экономики, статистики и информатики

Филинов Евгений Николаевич – главный научный сотрудник Института системного программирования РАН

Руководство по изучению дисциплины

Предисловие

Данное руководство содержит методические указания студентам, обучающимся по специальностям 351400 «Прикладная информатика по областям» и 351500 «Математическое обеспечение и администрирование информационных систем».

Основной задачей дисциплины является изучение концепции и принципов построения открытых информационных систем.

Эти знания особенно важны для специалистов в области проектирования и эксплуатации информационных систем. Основными целями создания и применения концепции, методов и стандартов открытых систем является повышение общей экономической эффективности разработки и функционирования информационных систем, а также логической и технической совместимости их компонентов, обеспечение мобильности программ и данных информационных систем.

Для усвоения курса дисциплины студентами необходимы знания и навыки, приобретенные при изучении дисциплин «Вычислительные машины и системы», «Операционные системы и оболочки», «Проектирование информационных систем».

1. Основные понятия и свойства открытых систем

1.1. Содержание темы

Основные определения открытых систем:
- функциональная среда открытых систем
- интерфейсы прикладного программирования
- прикладная программа (приложение)
- прикладная платформа
- программные средства промежуточного слоя
- архитектура и структура информационных систем

Свойства открытых систем:
- расширяемость
- масштабируемость
- переносимость приложений, данных и персонала
- интероперабельность приложений и систем
- способность к интеграции
- высокая готовность

Преимущества открытых систем.

1.2. Цели изучения темы

1. Знакомство с используемыми понятиями в области открытых информационных систем.
2. Знакомство с основными свойствами открытых информационных систем.

1.3. Задачи изучения темы

1. Формирование представления об открытых информационных системах.
2. Формирование представления о свойствах открытых информационных систем.

Изучив тему, студент должен:

знать:

1. Основные понятия области открытых систем:
- функциональная среда открытых систем
- интерфейсы прикладного программирования
- прикладная программа (приложение)
- прикладная платформа
- программные средства промежуточного слоя

2. Преимущества открытых систем.

уметь:

дать определение свойствам открытых систем:
- расширяемость
- масштабируемость
- переносимость приложений, данных и персонала
- интероперабельность приложений и систем
- способность к интеграции
- высокая готовность

Необходимо акцентировать внимание на следующих понятиях:

- информационная система
- архитектура информационной системы
- структура информационной системы
- открытая информационная система
- свойства открытой информационной системы

1.4. Порядок изучения темы

Для изучения темы выделяется 6 лекционных часов и 2 часа самостоятельной работы.

Предусмотрены:

1. Лекции на тему «Обзор дисциплины» и «Основные определения и свойства открытых систем».
2. Самостоятельная работа студента в формах:

- подготовки к лекции,
- изучения дополнительной литературы.

1.5. Методические указания

Вопросы темы:

1. Информационные системы:
   - назначение,
   - понятия архитектура и структура,
   - модель информационной системы.
2. Открытые информационные системы:
   - концепция открытых систем,
   - основные определения,
   - свойства.

При изучении первого вопроса:

Готовясь к лекции.

Прочтите материалы учебника «Основы открытых информационных систем.» [1] (с.8 – 18.). На лекции будут подробно рассмотрены концепция создания, основные понятия и свойства открытых систем.

Следует обратить внимание на различия в понятиях «архитектура» и «структура» открытых систем, модель информационной системы. Акцентируйте внимание на преимуществах открытых систем перед закрытыми информационными системами для различных категорий пользователей. Для формирования представления об основных целях и задачах концепции открытых систем обратитесь к дополнительной литературе: [6], [7], [13], [16].

При изучении второго вопроса:

Готовясь к лекции.

Прочтите материалы учебника «Основы открытых информационных систем»[1] (с.18 – 24.). На лекции будут подробно рассмотрены свойства открытых систем.

Ознакомившись с темой, разберитесь основные свойства открытых систем. Акцентируйте внимание на преимущества открытых систем перед закрытыми информационными системами для различных категорий пользователей. Ознакомьтесь с содержанием дополнительной литературы по теме [13], [16] и с архивом электронного издательства «Открытые системы» [24].

1.6. Контрольные вопросы

Ответьте – да или нет:

1. Архитектура информационной системы с точки зрения пользователя – это описание системы команд, организации прерываний, организации памяти и ввода-вывода.

2. Масштабируемость – это свойство открытой системы, означающее возможность изменения ее количественных характеристик (размерность решаемых задач, число одновременно обслуживаемых пользователей и т.д.) путем настройки параметров приложений и баз данных, а не путем перепроектирования и программирования заново.

3. В пределах каждой системы свойство интероперабельности рассматривается на уровнях межсетевого взаимодействия, определяемое стандартными протоколами вычислительных сетей.

4. Под свойством высокой готовности понимается возможность объединения нескольких ИС различного назначения в единую интегрированную многофункциональную ИС.

5. На уровне интеграции систем способность к интеграции означает возможность объединения нескольких ИС различного назначения в единую интегрированную многофункциональную ИС.

6. На уровне интеграции данных способность к интеграции означает представление для прикладной программы или пользователя нескольких БД как одной логически единой БД.

7. На уровне интеграции приложений способность к интеграции определяется методами и средствами объединения прикладных программ в многозвенных архитектурах «клиент-сервер» с выделением серверов приложений, ориентированных на определенные классы задач, объектные среды и оболочки, позволяющие объединять приложения на основе механизмов обмена сообщениями.

8. Интероперабельность систем обеспечивается, прежде всего, представлением для прикладной программы или пользователя нескольких баз данных как одной логически единой базы данных.

9. Термин «Функциональная среда открытой системы» является переводом на русский язык выражения open system environment.

2. Среда открытых систем

2.1. Содержание темы

Определение понятия «среда открытой информационной системы». Моделирование среды открытых систем. Модели MIC, MUSIC, OSE/RM. Службы среды открытых систем. Состав компонентов среды. Обобщенная модель среды открытых систем.

2.2. Цели изучения темы

1. Знакомство с основными направлениями развития идеологии открытых систем. Определение понятия «среды открытых систем».

2. Место среды открытых систем в общей структуре информационной системы. Компоненты среды открытых систем.

2.3. Задачи изучения темы

1. Изучение терминологии, используемой в публикациях по открытым системам.
2. Формирование представления о целях создания эталонных моделей среды открытых систем. Изучение функций компонентов среды.

Изучив тему, студент должен:

знать:

- место среды открытых систем в структуре ИС,
- принципы создания и назначение моделей среды,
- назначение основных функциональных компонентов среды,
- знать структуру концептуальной модели среды открытых систем;

уметь:

- дать определение терминов – концептуальная модель, услуги среды открытых систем,
- разбираться в структуре модели среды открытых систем.

Необходимо акцентировать внимание на следующих понятиях:

- среда открытой информационной системы,
- модель среды открытой системы,
- эталонная модель,
- эталонная модель среды открытой системы,
- эталонная модель взаимосвязи среды открытых систем,
- концептуальная модель среды открытой системы.

2.4. Порядок изучения темы

Для изучения темы выделяется 8 лекционных часов, 8 часов семинарских занятий и 2 часа самостоятельной работы.

Предусмотрены:

1. Лекция на тему «Среда открытой информационной системы».
2. Лекция на тему « Назначение и функции компонентов среды открытых систем».
3. Семинар 1.
4. Самостоятельная работа студента в формах:

- подготовки к лекции,
- разработка реферата по одной из тем семинара 1,
- изучение дополнительной литературы.

2.5. Методические указания

Вопросы темы:

1. Знакомство с основными направлениями развития идеологии открытых систем. Определение понятия «среды открытых систем».

2. Место среды открытых систем в общей структуре информационной системы. Компоненты среды открытых систем.

В этой части курса вводятся основные понятия, на которые опирается концепция открытых систем, рассматривается общая структура среды открытых систем, обсуждаются ее основные компоненты.

Готовясь к лекциям.

Прочтите материалы учебника «Основы открытых информационных систем»[1] (с.25 – 42.). На лекциях будут подробно рассмотрены основные понятия, место среды открытых систем в структуре ИС, состав и назначение компонентов среды. Акцентируйте внимание на специфичной терминологии, используемой в литературе по Открытым системам. Обратитесь к глоссарию. При необходимости, обращайтесь к дополнительным источникам [2], [6], [14], [16].

Готовясь к семинару.

Выберите тему из списка тем семинара 1. Подготовьте краткий реферат по теме в пределах 10 стр. Подготовьтесь к докладу по теме реферата.

2.6. Контрольные вопросы

Ответьте – да или нет:

1. Интерфейс – это составная часть устройства, программы, системы, данных.
2. Прикладная платформа – это…функциональная часть ИС, включающая в себя логически связанную группу модулей или компонентов, данные, средства обращения к информационным ресурсам ИС, которые необходимы для выполнения определенной прикладной функции ИС.
3. концептуальная модель – это система основных понятий и правил комбинирования классов понятий, независимых от языка их представления и являющихся смысловой структурой некоторой предметной области.
4. Спецификации API (интерфейсы прикладного программирования) группируются по основным группам функций (услуг), предоставляемых средой приложениям: поддержка пользовательского интерфейса; организация процессов обработки данных; – это представление данных для хранения и обмена; услуги телекоммуникаций.
5. На втором уровне обобщенной модели среды открытых систем присутствуют: средства оконного интерфейса, имеющиеся в составе операционной системы; средства организации процессов обработки данных; средства доступа к среде хранения данных; средства транспортного уровня эталонной модели OSI/RM.
6. Эталонная модель в функциональной стандартизации – это представление структуры открытой системы в виде набора таблиц, где указываются ссылки на стандарты и спецификации интерфейсов и протоколов взаимодействия между компонентами этой системы.
7. Метамодель (в СУБД) – это модель данных, определяемая на метаязыке и основанная на общих, независимых от конкретных моделей данных, концепциях, которые обеспечивают однозначное выражение семантических свойств разнообразных моделей данных, определяя их сходства и различия при использовании единого языка.
8. Декомпозиция – это разбиение объекта разработки (задачи, программы, данных, системы) на структурные единицы.
9. В системах распределенной обработки данных с архитектурой «клиент-сервер» службы системного администрирования включают в себя единый набор средств мониторинга и управления приложениями, сервисами и ресурсами среды выполнения.

3. Профили открытых информационных систем

3.1. Содержание темы

Цели и принципы формирования профилей информационных систем. Структура и содержание профилей ИС. Процессы формирования, развития и применения профилей ИС.

3.2. Цель изучения темы

Определение понятия «профиль информационной системы». Изучение принципов формирования профилей открытых систем. Изучение процессов формирования, развития и применения профилей информационных систем.

3.3. Задача изучения темы

Формирование представления о назначении профилей, категориях и видах профилей. Изучение структуры и содержания профилей информационных систем.

Изучив тему, студент должен:

знать:

- определение понятия «профиль информационной системы»,
- что такое функциональный профиль,
- различия между функциональными и вспомогательными профилями,
- назначение профилей,
- номенклатуру и структуру профилей,
- содержание профилей среды информационной системы;
- основные цели создания профилей информационных систем и современное состояние, и развитие стандартизации в области информационных технологий.

уметь:

- использовать принципы построения профилей и формирования их на различных этапах жизненного цикла открытых информационных систем.

Необходимо акцентировать внимание на следующих понятиях:

- профиль информационной системы,
- функциональные и вспомогательные (технологические) профили,
- назначение профилей,
- структура профилей,
- гармонизация профилей,
- применение профилей.

3.4. Порядок изучения темы

Для изучения темы выделяется 6 лекционных часа, 2 часа семинарских занятий и 2 часа самостоятельной работы.

Предусмотрены:

1. Лекция на тему «Назначение, структура и содержание профилей информационных систем»;

2. Лекция на тему « Процессы формирования, развития и применения профилей ИС»;
3. Семинар 2, объединенный по темам 3 и 4 (см. следующую главу).
4. Самостоятельная работа студента в формах:

− подготовки к лекции,
− изучения дополнительной литературы.

3.5. Методические указания

Вопросы темы:

1. Определение понятия «профиль открытой системы», цели создания, назначение и виды профилей.
2. Процессы формирования профилей.

При изучении первого вопроса:
Готовясь к лекциям.

Прочтите материалы учебника «Основы открытых информационных систем»[1] (с.46-64.). Акцентируйте внимание на определение понятия «профиль информационной системы». Внимательно изучите структуру и содержание профилей. Постарайтесь разобраться в основах стандартизации информационных технологий. Обратитесь к дополнительным источникам [4], [5], [10], [11].

При изучении второго вопроса:
Готовясь к лекциям.

Прочтите материалы учебника «Основы открытых информационных систем»[1] (с.65-73.). Разберитесь в процессах формирования профилей информационных систем. Научитесь применению профилей на различных стадиях жизненного цикла информационных систем. Ознакомьтесь с приложениями 3 и 4 учебника «Основы открытых информационных систем»[1] Обратитесь к дополнительным источникам [3], [5], [10], [11].

3.6. Контрольные вопросы

Ответьте – да или нет:

1. Базовый стандарт – это любой стандарт (или его подмножество), используемый при построении профиля.
2. Одной из основных целей применения профилей при создании и использовании ИС является использование информационных ресурсов, существующих в других системах.
3. Профиль конкретной системы в процессе проектирования является статичным.
4. Профиль – это общедоступная спецификация, которая поддерживается открытым, гласным процессом, направленным на приспособление новой технологии к ее применению, которая согласуется со стандартами.
5. При разработке тестов соответствия информационных систем или их компонентов требованиям профиля, с целью унификации, определяется концептуальная модель.
6. Для любой ИС в соответствии с принятой концептуальной моделью должны быть определены профили среды ИС, включающие в себя спецификации программных интерфейсов между приложениями и средой.
7. Устранение противоречий и уточнение альтернативных возможностей в выбранном комплексе базовых стандартов ИС производится на этапе параметризации компонентов среды ИС.

8. Функциональные параметры, определяющие состав сервисов и услуг, предоставляемых данным компонентом ИС, определяются на этапе параметризации компонентов среды ИС.

9. Устранение избыточности требований базовых стандартов с точки зрения описания компонентов ИС, для которой разрабатывается профиль, производится на этапе гармонизации базовых стандартов.

10. Таблицы с параметрами компонентов и их значениями являются результатом этапа гармонизации базовых стандартов.

11. Изображенная в графическом виде уточненная концептуальная модель является результатом этапа уточнения концептуальной модели и параметров компонентов.

4. Методология построения профилей ИС

4.1. Содержание темы

Категории, виды и структура профилей ИС. Порядок разработки, согласования и утверждения профилей ИС. Выбор модели среды ИС. Параметризация компонентов среды ИС. Наполнение профилей базовыми стандартами информационных технологий. Оформление профилей.

4.2. Цели изучения темы

Формирование представления о методологии разработки профилей информационных систем.

Последовательность согласование и утверждение профилей информационной системы.

4.3. Задачи изучения темы

Последовательность разработки профилей ИС:

- определение прикладных задач,
- выбор концептуальной модели среды информационной системы,
- параметризация компонентов среды информационной системы,
- наполнение профиля базовыми стандартами информационных технологий,
- уточнение концептуальной модели и параметров компонентов.
- гармонизация базовых стандартов,
- формирование требований соответствия информационной системы профилю,
- оформление профилей информационной системы.

Изучив тему, студент должен:

знать:

- какие существуют группы профилей,
- чем регламентируются общие положения функциональной стандартизации в области информационных технологий,
- что профили ИС должны учитываться при формировании технических заданий на создание или модернизацию и развитие ИС,
- порядок разработки профилей информационных систем.
- порядок согласования и утверждения профилей информационной системы.

Необходимо акцентировать внимание на следующих понятиях:

- категория профиля,
- статус утверждения профиля,
- сценарий профиля,
- параметризация компонентов,
- гармонизация базовых стандартов,
- наполнение профилей,
- оформление профилей,
- согласование и утверждение профилей.

4.4. Порядок изучения темы

Для изучения темы выделяется 4 лекционных часа, 2 часа семинарских занятий и 2 часа самостоятельной работы.

Предусмотрены:

1. Лекция на тему « Методология построения профилей ИС»;
2. Практические занятия в форме семинаров и лабораторных работ по темам Семинара 2.
3. Самостоятельная работа студента в формах:

- подготовки к лекции,
- изучения дополнительной литературы.

4.5. Методические указания

Вопрос темы:

Методология построения профилей ИС.

При изучении вопроса:

Готовясь к лекции

Прочтите материалы учебника «Основы открытых информационных систем»[1] (с.71-78.). Разберитесь в последовательности формирования профилей информационных систем. Ознакомьтесь с приложениями 2 и 3 учебника «Основы открытых информационных систем»[1]. Обратитесь к дополнительным источникам, например, [6], [24].

Готовясь к семинару:

Выберите тему из списка тем семинара 2. Подготовьте краткий реферат по теме в пределах 10 стр. Подготовьтесь к докладу по теме реферата**.**

4.6. Контрольные вопросы

Ответьте – да или нет:

1. Профиль конкретной системы в процессе проектирования развивается и конкретизируется.
2. Устранение противоречий и уточнение альтернативных возможностей в выбранном комплексе базовых стандартов ИС производится на этапе уточнения концептуальной модели и параметров компонентов.
3. Функциональные параметры, определяющие состав сервисов и услуг, предоставляемых данным компонентом ИС, определяются на этапе параметризации компонентов среды ИС.

4. Интерфейсные параметры, определяющие характеристики взаимодействия данного компонента с другими компонентами среды и приложениями, определяются на этапе гармонизации базовых стандартов.

5. Методология построения профилей должна учитывать принципы соответствия состава профилей.

6. Профили средств поддержки, создания, сопровождения и развития программного обеспечения ИС описывают инструментальные средства, встраиваемые в ИС и работающие в среде ИС.

5. Объекты стандартизации в функциональных профилях ИС

5.1. Содержание темы

Объекты стандартизации в профилях приложений ИС. Объекты стандартизации в профилях среды распределенной обработки данных. Объекты стандартизации в профилях компонентов сервисных служб среды ИС. Объекты стандартизации в профилях операционных систем. Объекты стандартизации в профилях технических средств ИС. Объекты стандартизации в профилях телекоммуникационной среды. Интерфейсы и протоколы сетей передачи данных. Объекты стандартизации в профилях администрирования. Объекты стандартизации в профилях защиты информации. Объекты стандартизации в профилях средств поддержки создания, сопровождения и развития программного обеспечения ИС. Источники базовых стандартов для функциональных профилей ИС.

5.2. Цели изучения темы

Знакомство с объектами стандартизации информационной системы и источниками базовых стандартов.

5.3. Задачи изучения темы

1. Знакомство с объектами стандартизации в профилях приложений открытой системы и в профилях среды открытой информационной системы.

2. Знакомство с источниками базовых стандартов для функциональных профилей информационных систем.

Изучив тему, студент должен:

знать:

− структуру концептуальной модели ИС,

− источники базовых стандартов,

− объекты стандартизации в приложениях и среде открытых систем.

уметь:

− правильно использовать функциональные стандарты при разработке профилей,

Необходимо акцентировать внимание на следующих понятиях:

− источники международных, государственных и т.д. стандартов и нормативных документов,

− объекты стандартизации в локальных и распределенных системах обработки данных,

− функции компонентов среды открытых систем.

5.4. Порядок изучения темы

Для изучения темы выделяется 4 лекционных часа, 2 часа семинаров практических занятий и 2 часа самостоятельной работы.

Предусмотрены:

1. Лекция на тему «Объекты стандартизации в функциональных профилях ИС»;
2. Практические занятия в виде семинара по темам «Объекты стандартизации в функциональных профилях ИС»« и «Компонентная разработка приложений» (См. темы семинара 3).
3. Самостоятельная работа студента в формах:

− подготовки к лекции,

− изучение дополнительной литературы.

5.5. Методические указания

Вопрос темы:

Объекты стандартизации в функциональных профилях ИС.

При изучении темы:

Готовясь к лекции

Прочтите материалы учебника «Основы открытых информационных систем» [1] (с.79-90). Обратитесь к моделям информационной системы и среды открытых систем с целью определения объектов стандартизации в их компонентах. Ознакомьтесь с приложениями 2 и 5 учебника «Основы открытых информационных систем»[1]. Обратитесь к дополнительным источникам [4], [7], [12].

Готовясь к практическим занятиям

Выберите тему из списка тем семинара 3. Подготовьте краткий реферат по теме в пределах 10 страниц. Подготовьтесь к докладу по теме реферата**.**

5.6. Контрольные вопросы

Ответьте – да или нет:

1. Информационные системы разделяются на приложения (прикладные программы) и среду (платформы прикладных программ), в которой функционируют эти приложения, в соответствии со стандартом ГОСТ Р ИСО/МЭК ТО 10000-3-99 «Информационная технология. Основы и таксономия международных стандартизованных профилей. Часть 1».
2. Объектами стандартизации в профилях среды распределенной обработки данных являются следующие. Мониторы обработки транзакций.
3. Объекты стандартизации в профилях операционных систем являются функции перемещение файлов из каталога в каталог.
4. Объектами стандартизации в профилях технических средств ИС являются протоколы и интерфейсы локальных сетей (уровней 3 и 4 эталонной модели взаимодействия открытых систем OSI/RM).

5. Форматы электронных сообщений (электронного обмена данными) являются объектами стандартизации в профилях приложений ИС.

6. Обеспечение защиты от несанкционированного доступа, доступности информационных ресурсов, целостности программ и данных является объектом стандартизации профилей операционных систем.

7. Общие регламенты системного администрирования и службы системных и сетевых каталогов являются объектом стандартизации в профилях администрирования.

8. Функции защиты управления данными включают в себя средства контроля и управления доступом и целостностью данных в СУБД.

9. Объектами стандартизации в профилях телекоммуникационной среды являются функции обработки транзакций (в соответствии с прикладным уровнем эталонной модели взаимосвязи открытых систем OSI/RM).

10. Общие регламенты системного администрирования и службы системных и сетевых каталогов являются объектом стандартизации в профилях… телекоммуникационной среды.

11. К объектам стандартизации в профилях средств создания, сопровождения и развития программного обеспечения информационных систем относятся стандартные языки программирования и среда поддержки прикладного ПО (отладчики, средства настройки и оптимизации программного кода, редакторы).

6. Компонентная разработка приложений

6.1. Содержание темы

Основные концепции компонентной разработки приложений. Стандарты компонентов. Интерфейсы компонентов. Контейнеры. Метаданные. Распределенные серверные компоненты.

Интегрированные среды разработки приложений. Модель DCOM. Спецификация Java Beans. Компонентная разработка WEB-приложений. Спецификации компонентов в архитектуре CORBA.

Перспективы развития методов и средств компонентной разработки приложений.

6.2. Цели изучения темы

1. Формирование представления студента об основных концепциях компонентной разработки приложений.
2. Дать представление об интегрированных средах разработки приложений.
3. Обзор перспективы развития методов и средств компонентной разработки приложений.

6.3. Задачи изучения темы

1. Определение понятия «компонента открытой информационной системы».
2. Определение цели создания стандартов компонентов.
3. Определение понятий: интерфейсы компонентов, контейнеры, метаданные.
4. Раскрытие содержания спецификаций: модели DCOM, компонентов Java Beans, компонентов WEB-приложений, компонентов в архитектуре CORBA.

Изучив тему, студент должен:

знать:

- что такое компонент информационной системы,
- основные цели компонентной разработки приложений,
- основные виды спецификаций компонентов различных видов моделей.

иметь представление

- о перспективах развития методов и средств компонентной разработки приложений

Необходимо акцентировать внимание на следующих понятиях:

- компонент,
- контейнер,
- метаданные,
- интегрированная среда разработки компонентов,
- интерфейс компонента,
- DCOM, Java Beans, WEB-приложения, CORBA.

6.4. Порядок изучения темы:

Для изучения темы выделяется 4 лекционных часа, 2 часа практических занятий и 2 часа самостоятельной работы.

Предусмотрены:

1. Лекция на тему « Компонентная разработка приложений».
2. Семинар 3.
3. Самостоятельная работа студента в формах:

- подготовки к лекции,
- подготовки к практическим занятиям,
- изучения дополнительной литературы.

6.5. Методические указания

Вопрос темы:

Цели и задачи методологии разработки компонентов приложений.

При изучении темы:

Готовясь к лекции.

Прочтите материалы учебника «Основы открытых информационных систем» [1] (с.91-107). Обратитесь к дополнительным источникам [10], [11], [22], [23], [25], [27].

Готовясь к практическим занятиям.

Выберите тему из списка тем семинара 3. Подготовьте краткий реферат по теме в пределах 10 страниц. Подготовьтесь к докладу по выбранной теме реферата**.**

6.6. Контрольные вопросы

Ответьте – да или нет:

1. COM, DCOM, Java Beans, CORBA – это модели метаданных.
2. Простейшие структурные элементы программного обеспечения, которые могут быть повторно использованы при построении программных средств, – это компоненты.

3. Компоненты программного обеспечения реализуют… открытые спецификации на интерфейсы, сервисы (услуги среды) и поддерживаемые форматы данных.

4. Свойства компонента описывают значения общедоступных атрибутов компонента.

5. Реакцию компонента на внешние воздействия или на внутренние условия определяют события.

6. Компоненты существуют и функционируют внутри процедур.

7. В составе служб среды распределенной обработки данных (на базе модели DCOM) служба распределенной обработки транзакций в виде сервера MS Transaction Server интегрирует службу транзакций в модель разработки компонентов и предоставляет серверным компонентам транзакционную среду исполнения.

8. Платформа распределенных компонентов, основанная на спецификации Java Beans, предложена фирмой Microsoft.

9. Платформа распределенных компонентов, основанная на модели DCOM, предложена фирмой Microsoft.

10. Каждый компонент в среде EJB должен функционировать внутри контейнера, изолирующего его от рабочей операционной среды сервера.

11. Возможности доступа к базам данных, передачи сообщений, обработки транзакций, интеграции с Web-сервером для систем, взаимодействующих в сети Интернет, должно обеспечивать программное обеспечение промежуточного слоя модели OSE/RM.

12. Такие платформы DCP, как DCOM или Java Beans, позволяют пользователям конструировать компоненты и связывать их между собой.

7. Рекомендуемая литература

Основная

1. Бойченко А.В., Кондратьев В.К., Филинов Е.Н. Основы открытых информационных систем. М.: МЭСИ, 2004. 173 с.

Дополнительная

2. А. Бойченко, Г. Горелкин, В. Горшков, Е. Филинов. Обобщенная модель открытых информационных систем //Сетевой журнал Data Communications, 2000, №1.

3. Вендров А.М. Проектирование программного обеспечения экономических информационных систем. М.: Финансы и статистика, 2000.

4. Галатенко В.А. Информационная безопасность//Открытые системы 1995, №№ 4, 5, 6. и 1996 №№ 1, 2, 3, 4.

5. Д.Слама, Д.Гарбис, П.Рассел. Корпоративные системы на основе CORBA. Изд. дом «Вильямс», Москва, Санкт-Петербург, Киев, 2000.

6. Козлов В.А. Открытые информационные системы. М.: Финансы и статистика. 1999.

7. Лезер Н. Архитектура открытых распределенных систем: Модель OSF DCE// Открытые системы. №3. 1993.

8. Липаев В.В. Методы обеспечения качества крупномасштабных программных средств. М.: СИНТЕГ, 2003

9. Липаев В.В., Филинов Е.Н. Мобильность программ и данных в открытых информационных системах. М.: Научная книга. 1997.

10. Орфали Р., Харки Д. JAVA и CORBA в приложениях клиент-сервер. Изд. «Лори», М. 2000.

11. Орфали Р., Харки Д., Эдвардс Д. Основы CORBA (пер. с англ.) М: МАЛИП. 1999.

12. Смирнова Г.Н., Сорокин А.А., Тельнов Ю.Ф. Проектирование экономических информационных систем. М.: Финансы и статистика, 2001.

13. Смит Д.М., Маленовски М. Время пришло для профессионалов в области открытых систем. Открытые системы, №1, 1995.

14. Сухомлин В.А. Методологический базис открытых систем //Открытые системы. № 4 (12).1996.

15. Тельнов Ю.Ф. Интеллектуальные системы в экономике. М: Изд. СИНТЕГ, 1999.

16. Филинов Е.Н. Архитектура и структура среды распределенной обработки данных, методы и средства формального описания среды // Распределенная обработка информации. Труды Шестого международного семинара. Новосибирск. Сибирское отделение РАН. 1998.

17. Филинов Е.Н. Выбор и разработка концептуальной модели среды открытых систем. Открытые системы, 1995, вып. 6.

18. Функциональные стандарты в открытых системах. В 2-х т. М.: Изд. МЦНТИ.1997.

19. Щербо В.К. Международная стандартизация в области информационных технологий. Проблемы информатизации. М.: 1992.№4

20. Щербо В.К., В.А. Козлов. Функциональные стандарты в открытых системах. Часть 1, часть 2. Справочное пособие. М., МЦНТИ, 1997

21. Якубайтис Э.А. Открытые информационные сети. М.: Радио и связь. 1991.

Ссылки в Интернет

1. http://java.sun.com/prodacts/JavaManagement/document.html
2. http://www.w3.org/TR/REC-html40
3. http://www.osp.ru
4. http://www.w3.org/TR/WD-rdf-Syntax
5. http://www.w3.org/TR/WD-DOM
6. http://www.omg.org

8. Ответы на вопросы

(да:+; нет: -)

В случае, если ваши ответы не совпадают с приведенными в таблице, еще раз внимательно изучите соответствующую тему и повторите попытку ответа на контрольные вопросы.

Номер вопроса	Номер темы					
	1	2	3	4	5	6
1	-	-	+	+	-	-
2	+	-	-	-	+	+
3	+	+	-	+	+	-
4	-	+	-	-	-	+
5	+	+	-	+	+	+
6	-	+	+	+	-	-
7	+	+	-		+	+
8	-	+	+		+	-
9	+	+	+		+	+
10			-		-	+
11			+		+	+

9. Семинары

Внимание!

Каждый из студентов должен разработать и написать три реферата (по одной теме для каждого семинара).

Выбор темы зависит от номера зачетной книжки:

– если последняя цифра номера четная, студент делает выбор из четных номеров тем;

– если последняя цифра номера нечетная, студент делает выбор из нечетных номеров тем.

9.1. Семинар 1

Темы рефератов

1. Обозначьте и прокомментируйте смысловую разницу между понятиями «информационная система» и «открытая информационная система».
2. Раскройте цели и задачи функциональной среды открытой системы.

3. Назначение и функции программных средств промежуточного слоя среды открытых систем.

4. Опишите основные характеристики и дайте оценку разных (2-3) прикладных платформ информационных систем.

5. Прокомментируйте преемственность и связь свойств открытых информационных систем.

6. Проиллюстрируйте основные преимущества идеологии открытых систем.

9.2. Семинар 2

Темы рефератов

1. Опишите, каким образом связаны друг с другом основные функциональные части и интерфейсы информационных систем.

2. Прокомментируйте роли интерфейсов API и EEI в информационной системе как необходимых составляющих Reference Model.

3. Выделите составляющие архитектуры открытых систем и функциональные связи между ними.

4. Определите цели и задачи создания моделей среды открытых систем.

5. Приведите примеры двух направлений развития идеологии, концепции и системы стандартов открытых систем.

6. Назначение эталонной модели открытых систем (OSE/RM), её структура.

7. Свяжите уровни описания и функциональные группы компонентов модели среды открытых систем, проследите логические связи между ними.

9.3. Семинар 3

Темы рефератов

1. Что явилось причиной создания интегрированных сред разработки распределенных компонентов (*distributed component platform* – DCP).

2. Раскрыть назначение и принципы работы интегрированных систем разработки приложений.

3. Опишите различия моделей COM и DCOM.

4. Опишите различия архитектур DCOM и CORBA.

5. Опишите спецификации Enterprise Java Beans (EJB) фирмы Sun Microsystems.

Практикум по дисциплине

Выбор темы и оформление реферата

Каждый из студентов должен разработать и написать три реферата (по одной теме для каждого семинара).

Выбор темы зависит от номера зачетной книжки:

− если последняя цифра номера **четная**, студент делает выбор **из четных номеров тем**;

− если последняя цифра номера **нечетная**, студент делает выбор **из нечетных номеров тем**.

Объём реферата зависит от выбранной темы, но должен полностью раскрывать содержание темы (ориентировочно должен занимать 10-15 страниц текста формата А4) **с обязательным указанием ссылок на используемую литературу.**

Все рефераты должны быть защищены или на семинаре, где будет сделан доклад по теме реферата, или отдельно в установленное преподавателем для защиты время **до начала зачетной сессии**.

Титульный лист **должен быть оформлен** в соответствии с **Приложением 3**. Структура реферата произвольная.

При разработке реферата используйте различные информационные источники, включая литературу, список которой приведен в **Приложении 1.** Обратите внимание на глоссарий, приведенный в **Приложении 2**.

Обратите внимание на выделенные выше жирным шрифтом слова!!!

Темы рефератов

Семинар 1

1. Обозначьте и прокомментируйте смысловую разницу между понятиями «информационная система» и «открытая информационная система».
2. Раскройте цели и задачи функциональной среды открытой системы.
3. Назначение и функции программных средств промежуточного слоя среды открытых систем.
4. Опишите основные характеристики и дайте оценку разных (2-3) прикладных платформ информационных систем.
5. Прокомментируйте преемственность и связь свойств открытых информационных систем.
6. Проиллюстрируйте основные преимущества идеологии открытых систем.
7. Опишите, каким образом связаны друг с другом основные функциональные части и интерфейсы информационных систем.

Семинар 2

8. Прокомментируйте роли интерфейсов API и EEI в информационной системе как необходимых составляющих Reference Model.
9. Выделите составляющие архитектуры открытых систем и функциональные связи между ними.
10. Определите цели и задачи создания моделей среды открытых систем.

11. Приведите примеры двух направлений развития идеологии, концепции и системы стандартов открытых систем.

12. Назначение эталонной модели открытых систем (OSE/RM), её структура.

13. Свяжите уровни описания и функциональные группы компонентов модели среды открытых систем, проследите логические связи между ними.

Семинар 3

14. Что явилось причиной создания интегрированных сред разработки распределенных компонентов (distributed component platform – DCP).

15. Раскрыть назначение и принципы работы интегрированных систем разработки приложений.

16. Опишите различия моделей COM и DCOM.

17. Опишите различия архитектур DCOM и CORBA.

18. Опишите спецификации Enterprise Java Beans (EJB) фирмы Sun Microsystems.

Приложение 1

Рекомендуемая литература

1. Бойченко А.В., Кондратьев В.К., Филинов Е.Н. Основы открытых информационных систем. М.: МЭСИ, 2004.
2. А. Бойченко, Г. Горелкин, В. Горшков, Е. Филинов. Обобщенная модель открытых информационных систем //Сетевой журнал Data Communications, 2000, №1.
3. Вендров А.М. Проектирование программного обеспечения экономических информационных систем. М.: Финансы и статистика, 2000.
4. Галатенко В.А. Информационная безопасность//Открытые системы 1995, №№ 4, 5, 6. и 1996 №№ 1, 2, 3, 4.
5. Д.Слама, Д.Гарбис, П.Рассел. Корпоративные системы на основе CORBA. Изд. дом «Вильямс», Москва, Санкт-Петербург, Киев, 2000.
6. Козлов В.А. Открытые информационные системы. М.: Финансы и статистика. 1999.
7. Лезер Н. Архитектура открытых распределенных систем: Модель OSF DCE// Открытые системы. №3. 1993.
8. Липаев В.В. Методы обеспечения качества крупномасштабных программных средств. М.: СИНТЕГ, 2003
9. Липаев В.В., Филинов Е.Н. Мобильность программ и данных в открытых информационных системах. М.: Научная книга. 1997.
10. Орфали Р., Харки Д. JAVA и CORBA в приложениях клиент-сервер. Изд. «Лори», М. 2000.
11. Орфали Р., Харки Д., Эдвардс Д. Основы CORBA (пер. с англ.) М: МАЛИП. 1999.
12. Смирнова Г.Н., Сорокин А.А., Тельнов Ю.Ф. Проектирование экономических информационных систем. М.: Финансы и статистика, 2001.
13. Смит Д.М., Маленовски М. Время пришло для профессионалов в области открытых систем. Открытые системы, №1, 1995.
14. Сухомлин В.А. Методологический базис открытых систем //Открытые системы. № 4 (12).1996.
15. Тельнов Ю.Ф. Интеллектуальные системы в экономике. М: Изд. СИНТЕГ, 1999.
16. Филинов Е.Н. Архитектура и структура среды распределенной обработки данных, методы и средства формального описания среды // Распределенная обработка информации. Труды Шестого международного семинара. Новосибирск. Сибирское отделение РАН. 1998.
17. Филинов Е.Н. Выбор и разработка концептуальной модели среды открытых систем. Открытые системы, 1995, вып. 6.
18. Функциональные стандарты в открытых системах. В 2-х т. М.: Изд. МЦНТИ.1997.
19. Щербо В.К. Международная стандартизация в области информационных технологий. Проблемы информатизации. М.: 1992.№4
20. Щербо В.К., В.А. Козлов. Функциональные стандарты в открытых системах. Часть 1, часть 2. Справочное пособие. М., МЦНТИ, 1997
21. Якубайтис Э.А. Открытые информационные сети. М.: Радио и связь. 1991.

Ссылки в Интернет

1. http://java.sun.com/prodacts/JavaManagement/document.html
2. http://www.w3.org/TR/REC-html40
3. http://www.osp.ru
4. http://www.w3.org/TR/WD-rdf-Syntax
5. http://www.w3.org/TR/WD-DOM
6. http://www.omg.org

Приложение 2

Глоссарий

В данном разделе приведены основные термины, используемые в тексте учебного пособия, и их трактовка применительно к контексту рассматриваемых вопросов. Поэтому приведенные толкования терминов не обязательно совпадают с толковыми словарями общего назначения.

API-профиль (API-profile). Профиль, определяющий конкретную комбинацию базовых спецификаций прикладного пользовательского интерфейса в соответствии с моделью OSE/RM, возможно дополненных базовыми стандартами и/или профилями для представления данных и их форматов.

OSI-профиль (OSI-profile). Профиль, составленный из базовых спецификаций, соответствующих модели OSE/RM, возможно дополненных базовыми стандартами и/или профилями для представления обмениваемых данных и их форматов/

Архитектура «клиент-сервер» (client-server architecture): модель выполнения прикладных программ (приложений) в распределенной функциональной среде, в которой выполнение программ, обеспечивающих взаимодействие с пользователем системы, производится на рабочих станциях – клиентах, а выполнение программ, реализующих выполнение серверных частей приложения, общих для нескольких клиентов – на компьютерах-серверах (серверах приложений, серверах баз данных и т.д.).

Архитектура аппаратуры компьютера (computer architecture) – описание системы команд, организации прерываний, организации памяти и ввода-вывода – с точки зрения разработчика операционной системы и системного администратора.

Архитектура брокера объектных запросов (*common object request broker architecture* – CORBA): архитектура функциональной среды открытых систем, в которой основным механизмом взаимодействия между приложениями (представляемыми в этом случае в виде программных объектов) является обмен сообщениями через брокер объектных запросов.

Архитектура вычислительной сети (*computer network architecture*): совокупность принципов логической и физической организации технических и программных средств, протоколов и интерфейсов вычислительной сети, например, локальной сети, объединяющей компьютеры клиентов и серверы информационной системы.

Архитектура информационной системы (*information system architecture*): описание прикладных функций системы и ее интерфейсов с внешней средой (пользователями, другими системами и т.д.) с точки зрения пользователя системы.

Архитектура прикладной платформы (*application platform architecture*): часть архитектуры функциональной среды, содержащая спецификации интерфейсов операционной системы (оболочек, утилит, ядра, файловой системы, сетевых протоколов), поддерживающей выполнение клиентских или серверных частей приложений – с точки зрения программиста приложений.

Архитектура среды распределенных вычислений (*distributed computing environment architecture*): архитектура функциональной среды открытых систем, в которой основным механизмом взаимодействия между приложениями является вызов удаленных процедур (remote procedure call – RPC).

Архитектура функциональной среды открытых систем (*open system environment architecture*): описание услуг, предоставляемых приложениям системы со стороны среды, в которой они функционируют, и интерфейсов прикладного программирования, обеспечивающих взаимодействие приложений с функциональной средой, – с точки зрения проектирования системы и программиста приложений.

Базовый стандарт (*base standard*), также иногда используются термины формальный стандарт или стандарт de-ure. Международный стандарт, принятый ISO (международной организацией по стандартизации), или рекомендация организации ITU-T (до 1993 г. – CCITT) – международного союза по телекоммуникации.

Бизнес-процесс (*business process*): совокупность действий предприятия, получающая на входе определенные данные и продуцирующая результат, имеющий ценность для потребителя продукции этого предприятия. Например, процесс выполнения заказа является бизнес-процессом, на вход которого поступает заказ, а результат – заказанные товары, то есть доставка заказанных товаров потребителю и есть та ценность для потребителя, которую создает бизнес-процесс. Совокупность всех бизнес-процессов представляет собой бизнес-архитектуру предприятия. Бизнес-архитектура предприятия отображается на архитектуру информационной системы в виде состава прикладных функций ИС. Модель бизнес-процессов предприятия, полученная в результате предпроектного анализа его деятельности, является многоуровневой и обычно включает в себя три взаимосвязанных части: организационно-штатную структуру предприятия, собственно модель

бизнес-процессов, пронизывающих предприятие «по горизонтали», и данные о ресурсах предприятия, необходимых для выполнения бизнес-процессов. Анализ требований к архитектуре ИС с точки зрения отображения бизнес-архитектуры предприятия показывает, как правило, необходимость информационного сопряжения подсистем, поддерживающих разные бизнес-процессы, на различных уровнях управления предприятием, и интерфейсного сопряжения функциональных подсистем. К ним можно отнести: системы управления рабочими потоками, системы планирования ресурсов предприятия, системы оперативного анализа данных, системы функционально-стоимостного анализа, системы имитационного моделирования и др. Детализация бизнес-процессов осуществляется посредством бизнес-функций, бизнес-операций и бизнес-правил, которые поддерживаются информационной системой, обслуживающей предприятие. Модель бизнес-процесса, с которой работает ИС, содержит набор информационных объектов (ИО), представляемых в виде кортежей D_i (a_i^1, a_i^2,....a_i^n), где D_i – идентификатор i-го ИО, а a_i^j – j-ый атрибут i-го ИО. Бизнес-операция – представляется парой T_i D_j, где T_i – тип операции с j-ым ИО. Бизнес-функция – представляется в виде кортежа бизнес-операций J_m ((T_{1m}, D_{11}), ..., (T_{km}, D_{k1})), где J_m – код должности исполнителя, а T_{1m}, ..., T_{km} – элементы множества бизнес-операций $\{T_i\}$. Модель бизнес-процесса представляет собой граф управления бизнес-функциями, состоящий из множества узлов, каждый из которых соответствует определенной бизнес-функции: множества управляющих ребер выполнения бизнес-функций, множества узлов, соответствующих структурным подразделениям предприятия и множества ребер подчиненности подразделений, множества ресурсов предприятия и множества взвешенных ребер использования ресурсов бизнес-функциями.

Бизнес-процесс-реинжиниринг (*business process reengineering*): фундаментальное переосмысление и радикальное перепланирование бизнес-процессов предприятия, имеющее целью резкое улучшение показателей деятельности предприятия, таких, как затраты, качество и скорость обслуживания потребителей.

Внешний интерфейс (*front-end interface*): средства и правила взаимодействия системы (подсистемы) с внешними для нее ***объектами*** (внешней средой) – пользователем, вычислительной сетью и т.д. – в отличие от ее взаимодействия с другими компонентами той же системы.

Внутренний интерфейс (*back-end interface*): интерфейс какого-либо компонента системы с другим компонентом той же системы.

Декомпозиция (*decomposition*) – разбиение объекта разработки (задачи, программы, данных, системы) на структурные единицы. Декомпозиция является одной из задач системного анализа и проектирования. Для программных средств ИС и программных изделий выделяют следующие уровни декомпозиции: версия, компонент, модуль, процедура, программа, макрокоманда. Декомпозиция заданных функций ИС – процесс детализации совокупности бизнес-процессов предприятия, определенных на стадии предпроектного обследования, до требуемого состава бизнес-функций, бизнес операций и бизнес-правил, выполняемая на стадии проектирования ИС.

Интеллектуальный интерфейс (*intelligent interface*): совокупность средств взаимодействия пользователя с ИС на ограниченном естественном языке, включающая: диалоговый процессор, планировщик, преобразующий описание задачи в программу ее решения на основе информации, хранящейся в базе знаний, и монитор, осуществляющий управление всеми компонентами интерфейса.

Интероперабельность (*interoperability*) – свойство открытой системы, означающее возможность взаимодействия данной ИС с другими системами при необходимости обращения к информационным ресурсам этих систем (массивам файлов, базам данных, базам знаний) или решения определенных задач с помощью вычислительных ресурсов этих систем. Интероперабельность обеспечивается стандартными форматами электронного обмена данными (electronic data interchange – EDI), принятыми для разных прикладных областей, стандартными протоколами удаленного вызова процедур (remote procedure call-RPC) и обмена сообщениями (message interchange).

Интерфейс (*interface*): совокупность средств и правил, обеспечивающих взаимодействие устройств вычислительной системы и/или программ; совокупность унифицированных технических и программных средств, используемых для сопряжения устройств в вычислительной системе или сопряжения между системами; граница раздела двух систем, устройств или программ. *Примечание*: в эталонной модели взаимосвязи открытых систем (OSI/RM) понятие «интерфейс» введено для обозначения границы между средствами двух соседних уровней модели, в отличие от понятия «протокол», которое обозначает средства и правила взаимодействия двух систем на одном и том же уровне модели, например, на транспортном уровне – протокол TCP.

Интерфейс пользователя (*user interface*): комплекс прикладных и системных программных средств, обеспечивающий взаимодействие пользователя с ИС.

Интерфейс прикладного программирования (*application program interface – API*): интерфейс взаимодействия между прикладными программами (приложениями) ИС и средой, в которой они функционируют; интерфейс взаимодействия между двумя прикладными программами, реализующими разные функции ИС, например, интерфейс между двумя подсистемами интегрированной ИС.

Компонент (*component*) – составная часть устройства, программы, системы, данных.

Компонент программного обеспечения (*software component*) – простейший структурный элемент, который может быть повторно использован при построении программ или программных систем. Программный компонент реализует какую-либо прикладную функцию ИС, представляя семантически значимые услуги прикладного или технического характера. Отличительной особенностью компонента (в отличие от модуля программы) является возможность его модификации в процессе разработки на уровне двоичного исполняемого кода. Представлением компонента перед внешними для него объектами (другими составными частями программы), независимым от его внутренней реализации, являются интерфейсы и протоколы взаимодействия. Под интерфейсом компонента обычно понимаются: дескриптор интерфейса, набор свойств компонента, набор методов компонента, набор событий, определяющих реакцию компонента на внешние воздействия или внутренние условия.

Компонентная инфраструктура (*component framework*) – интегрированная среда компонентной разработки и исполнения приложений, содержащая: платформу, ориентированную на определенную модель компонентов (component object model – COM, distributed component object model – DCOM, Java Beans, CORBA или объектная модель Web), набор проектных шаблонов, адаптированных к приложениям некоторой области применения или технологии (например, технологии построения пользовательского интерфейса приложений), и набор готовых образцов компонентов.

Контейнер (*container*) – в терминологии объектно-ориентированного программирования – объект, файл или другой ресурс, используемый для хранения других объектов. При компонентной разработке приложений компоненты существуют и функционируют внутри контейнеров. Контейнеры образуют общий контекст взаимодействия между компонентами приложений. Контейнеры предоставляют также компонентам, вложенным в другие, более сложные компоненты, стандартный доступ к услугам среды выполнения. Контейнеры обычно реализуются в виде компонентов и могут быть вложены в другие контейнеры. Для организации взаимосвязей между компонентом и вмещающим его контейнером обычно используются протоколы, основанные на механизме событий.

Концептуальная модель (*conceptual model*) – система основных понятий и правил комбинирования классов понятий, независимых от языка их представления и являющихся смысловой структурой некоторой предметной области. Применительно к открытым системам концептуальная модель характеризует архитектуру системы в виде набора интерфейсов и протоколов взаимодействия между приложениями и средой, в которой они функционируют.

Масштабируемость (*scalability*): свойство открытой системы, означающее возможность изменения ее количественных характеристик, таких, как размерность решаемых задач, число одновременно обслуживаемых пользователей и т.д., путем настройки параметров приложений и баз данных, а не путем перепроектирования и программирования заново. Применительно к прикладным платформам ИС свойство масштабируемости означает предсказуемый рост их количественных системных характеристик при добавлении определенных вычислительных ресурсов, например, процессоров, модулей оперативной и дисковой памяти в конфигурациях серверов.

Метаданные (*metadata*): данные о данных. Вообще «мета» – это приставка, указывающая на то, что объект относится к более высокому уровню абстракции, чем уровень данных пользователя. В СУБД метаданные обозначают информацию о хранимых данных: таблицы описания данных и связей, адресные таблицы и другую информацию, используемую для просмотра данных и их трансформации. В компонентной разработке приложений метаданные компонента содержат сведения о компоненте, которые необходимы для обеспечения его взаимодействия с другими компонентами: описатель типа и тип, описание свойств компонента (классов и атрибутов), описания методов компонента и событий, определяющих реакцию компонента на внешние воздействия или внутренние условия. Метаданные о компоненте могут сообщаться либо статически (на этапе проектирования), либо динамически (на этапе выполнения).

Метамодель (*meta model*) – в СУБД – модель данных, определяемая на метаязыке и основанная на общих, независимых от конкретных моделей данных, концепциях, которые обеспечивают однозначное выражение семантических свойств разнообразных моделей данных, определения их сходства и различий при использовании единого языка (представления и манипулирования данными).

Модель (*model*): материальный объект, система математических зависимостей, или программа, имитирующие структуру или функционирование какого-либо исследуемого объекта. Основное требование к модели – ее адекватность объекту. Например, модель бизнес-процесса представляет собой граф управления бизнес-функциями, состоящий из множества узлов, каждый из которых соответствует определенной бизнес-функции: множества управляющих ребер выполнения бизнес-функций, множества узлов, соответствующих структурным подразделениям предприятия и множества ребер подчиненности подразделений, множества ресурсов предприятия и множества взвешенных ребер использования ресурсов бизнес-функциями. Модель бизнес-процессов предприятия, полученная в результате предпроектного анализа его деятельности, является многоуровневой и обычно включает в себя три взаимосвязанных части: организационно-штатную структуру пред-

приятия, собственно модель бизнес-процессов, пронизывающих предприятие «по горизонтали», и данные о ресурсах предприятия, необходимых для выполнения бизнес-процессов.

Обобщенная модель открытой системы (*generalized open systems model*) – представление структуры открытой системы в виде набора таблиц, где указываются ссылки на стандарты и спецификации интерфейсов и протоколов взаимодействия на всех уровнях структуры, включая взаимодействие на уровне «приложение-приложение» и на уровне «приложение-среда».

Открытая информационная система (*open information system*) – система, в которой реализованы открытые спецификации на интерфейсы, сервисы (услуги среды) и поддерживаемые форматы данных. Это дает возможность должным образом разработанному приложению быть переносимым в широком диапазоне систем с минимальными изменениями, взаимодействовать с другими приложениями на локальных и удаленных системах и взаимодействовать с пользователями в стиле, который облегчает переход пользователей от системы к системе.

Открытая спецификация (*open specification*) – общедоступная спецификация, которая поддерживается открытым, гласным согласительным процессом, направленным на приспособление новой технологии к ее применению, и которая согласуется со стандартами. Открытая спецификация является технологически независимой, т.е. не зависит от специфического аппаратного или программного обеспечения или продуктов конкретного изготовителя.

Переносимость (*portability*) –свойство открытой системы, означающее возможность переноса прикладных программ и данных на другие аппаратно-программные прикладные платформы при их модернизации или замене с минимальными затратами. Применительно к «переносимости» пользователей (user portability) это свойство обеспечивается дружественным пользовательским интерфейсом. Стабильность его поддерживается, чтобы не переучивать пользователей при внесении изменений в приложения и прикладные платформы, стандартизованными API по функциям пользовательского интерфейса и сохранением способов взаимодействия с пользователем, реализуемых приложениями (экранные формы, способы работы с каталогами файлов, способы задания запросов к базам данных, командные языки и т.д.)

Прикладная платформа (*application platform*) – операционная система и оборудование компьютера, на котором осуществляется выполнение прикладных программ (приложений). В ИС архитектурой распределенной обработки данных типа “клиент-сервер” прикладные платформы серверов (приложений, баз данных и т.д.) исполняют серверные части приложений, а прикладные платформы АРМ пользователей – клиентские части приложений. Совокупность нескольких разных по своей архитектуре прикладных платформ может образовать гетерогенную функциональную среду открытых систем.

Прикладная программа (приложение) (*application program*) – функциональная часть ИС, включающая в себя логически связанную группу модулей или компонентов, данные, средства обращения к информационным ресурсам ИС, которые необходимы для выполнения определенной прикладной функции ИС (бизнес – функции). Приложения ИС включают в себя данные, документацию (исходные тексты и описания), обучающие средства для пользователей, а также собственно программы в исполняемом коде.

Прикладная программа (приложение) (*application program*) – функциональная часть ИС, включающая в себя логически связанную группу модулей или компонентов, данные, средства обращения к информационным ресурсам ИС, которые необходимы для выполнения определенной прикладной функции ИС (бизнес – функции).

Прикладная функция ИС – то же, что бизнес-функция.

Программные средства промежуточного слоя (*middleware*) – средства, реализующие стандартные услуги функциональной среды открытых систем, такие, как функции управления базами данных, функции организации распределенной обработки данных. Например, мониторы транзакций, брокеры объектных запросов, функции обмена сообщениями, функции защиты информации (аутентификации пользователей и приложений, управления доступом к данным и приложениям), услуги телекоммуникационной среды, например, электронной почты, передачи файлов и т.д. Эти средства занимают в эталонной модели среды открытых систем промежуточное положение между приложениями и операционными системами прикладных платформ и поэтому называются программными средствами промежуточного слоя.

Программный интерфейс (*program interface*): интерфейс взаимодействия между программами.

Профиль (*profile*) – взаимосвязанная упорядоченная совокупность базовых стандартов и спецификаций, ориентированная на выполнение определенной прикладной функции ИС (или группы функций), определенной функции телекоммуникационной среды или на построение конкретной ИС в целом.

Стандарт (по определению ISO). Технический стандарт или другой документ, доступный и опубликованный, коллективно разработанный или согласованный и общепринятый в интересах тех, кто им пользуется, основанный на интеграции результатов науки, технологии, опыта, способствующий повышению общественного блага и принятый организациями, признанными на национальном, региональном и международном уровнях.

Таксономия (*taxonomy*). Классификационная схема, применяемая для однозначной идентификации профилей или наборов профилей.

Транзакция (*transaction*): 1. В СУБД – входное сообщение, переводящее базу данных из одного непротиворечивого состояния в другое, запрос на изменение базы данных. 2. В приложениях обработки данных – групповая операция, реализующая набор связанных бизнес-операций. Например, банковской транзакцией является совокупность операций по проведению платежа.

Функциональная интеграция – объединение систем, ранее создававшихся и функционировавших на предприятии автономно, независимо друг от друга (как «островки автоматизации», например, на производственных предприятиях в 70-х и 80-х годах) в единую интегрированную информационную систему, которая поддерживает все основные бизнес-процессы предприятия. Обеспечивая новые качества деятельности предприятия, например, в плане быстрого реагирования на изменяющиеся требования рынка и соответствующей перестройки бизнес-процессов, функциональная интеграция влечет за собой резкий рост сложности ИС. В свою очередь, возрастающая сложность ИС выдвигает дополнительные требования к методологии и средствам их проектирования и разработки.

Функциональная среда открытой системы (*Open System Environment – OS*E) – среда, поддерживающая разработку и выполнение приложений в открытой системе, предоставляя им стандартные услуги. Среда OSE обеспечивает исполнение приложений, при разработке которых были применены стандартные интерфейсы прикладного программирования (API), специфицированные для этой среды.

Функциональный стандарт (*functional standard*) – стандарт, определяющий один или несколько взаимоувязанных профилей с выделением во втором случае общих базовых стандартов этих профилей в отдельные части стандарта.

Эталонная модель (*reference model*) в функциональной стандартизации – представление структуры открытой системы в виде набора таблиц, где указываются ссылки на стандарты и спецификации интерфейсов и протоколов взаимодействия между компонентами этой системы. Различают эталонные модели среды открытых систем (*open systems environment/reference model – OSE/RM*) и взаимосвязи открытых систем *(open systems interconnection/reference model – OSI/RM*).