1. ИНСТРУМЕНТАЛЬНАЯ СРЕДА РАЗРАБОТКИ

1.1. Понятие и структура современной системы программирования

Системой программирования будем называть весь комплекс программных средств, предназначенных для кодирования, тестирования и отладки программного обеспечения.

Системы программирования в современном мире доминируют на рынке средств разработки. Практически все фирмы-разработчики компиляторов поставляют свои продукты в составе соответствующей системы программирования в комплексе всех прочих технических средств. Отдельные компиляторы являются редкостью и, как правило, служат только узко специализированным целям.

На рис. 1.1 приведена общая структура современной системы программирования. На ней выделены все основные составляющие современной системы программирования и их взаимосвязь. Отдельные составляющие разбиты по группам в соответствии с этапами развития средств разработки. Эти группы отражают все этапы развития от отдельных программных компонентов до цельной системы программирования.

Рис. 1.1. Общая структура и этапы развития систем программирования

* Жирным шрифтом выделены новые понятия, которые необходимо усвоить. Знание этих понятий будет проверяться при тестировании.

Из рис. 1.1 видно, что современная система программирования – это достаточно сложный комплекс различных программно-технических средств. Все они служат цели создания прикладного и системного программного обеспечения.

Тенденция такова, что все развитие систем программирования идет в направлении неуклонного повышения их дружественности и сервисных возможностей. Это связано с тем, что на рынке в первую очередь лидируют те системы программирования, которые позволяют существенно снизить трудозатраты, потребные для создания программного обеспечения на этапах жизненного цикла, связанных с кодированием, тестированием и отладкой программ. Показатель снижения трудозатрат в настоящее время считается более существенным, чем показатели, определяющие эффективность результирующей программы, построенной с помощью системы программирования.

В качестве основных тенденций в развитии современных систем программирования следует указать внедрение в них средств разработки на основе так называемых "языков четвертого поколения" – 4GL (four generation languages), а также поддержка систем "быстрой разработки программного обеспечения" – RAD (rapid application development).

Языки четвертого поколения – 4GL – представляют собой широкий набор средств, ориентированных на проектирование и разработку программного обеспечения. Они строятся на основе оперирования не синтаксическими структурами языка и описаниями элементов интерфейса, а представляющими их графическими образами. На таком уровне проектировать и разрабатывать прикладное программное обеспечение может пользователь, не являющийся квалифицированным программистом, зато имеющий представление о предметной области, на работу в которой ориентирована прикладная программа. Языки четвертого поколения являются следующим (четвертым по счету) этапом в развитии систем программирования.

Описание программы, построенное на основе языков 4GL, транслируется затем в исходный текст и файл описания ресурсов интерфейса, представляющие собой обычный текст на соответствующем входном языке высокого уровня. С этим текстом уже может работать профессиональный программист-разработчик – он может корректировать и дополнять его необходимыми функциями. Дальнейший ход создания программного обеспечения идет уже традиционным путем, как это показано на рис. 1.1.

1.2. Функции текстовых редакторов в системах программирования

Текстовый редактор в системе программирования – это программа, позволяющая создавать, изменять и обрабатывать исходные тексты программ на языках высокого уровня.

В принципе текстовые редакторы появились вне какой-либо связи со средствами разработки. Они решали задачи создания, редактирования, обработки и хранения на внешнем носителе любых текстов, которые не обязательно должны были быть исходными текстами программ на языках высокого уровня. Эти функции многие текстовые редакторы выполняют и по сей день.

Возникновение интегрированных сред разработки на определенном этапе развития средств разработки программного обеспечения позволило непосредственно включить текстовые редакторы в состав этих средств. Первоначально такой подход привел к тому, что пользователь (разработчик исходной программы) работал только в среде текстового редактора, не отрываясь от нее для выполнения компиляции, компоновки, загрузки и запуска программы на выполнение. Для этого потребовалось создать средства, позволяющие отображать ход всего процесса разработки программы в среде текстового редактора, – такие, например, как метод отображения ошибок в исходной программе, обнаруженных на этапе компиляции, с позиционированием на место в тексте исходной программы, содержащее ошибку.

Можно сказать, что с появлением интегрированных сред разработки ушло в прошлое то время, когда разработчики исходных текстов вынуждены были первоначально готовить тексты программ на бумаге с последующим вводом их в компьютер. Процессы написания текстов и собственно создание программного обеспечения стали единым целым.

Интегрированные среды разработки оказались очень удобным средством. Они стали завоевывать рынок средств разработки программного обеспечения. А с их развитием расширялись и возможности, предоставляемые разработчику в среде текстового редактора. Со временем появились средства пошаговой отладки программ непосредственно по их исходному тексту, объединившие в себе возможности отладчика и редактора исходного текста. Другим примером может служить очень удобное средство, позволяющее графически выделить в исходном тексте программы все лексемы исходного языка по их типам, – оно сочетает в себе возможности редактора исходных текстов и лексического анализатора компилятора.

В итоге в современных системах программирования текстовый редактор стал важной составной частью, которая не только позволяет пользователю подготавливать исходные тексты программ, но и выполняет все интерфейсные и сервисные функции, предоставляемые пользователю системой программирования. И хотя современные разработчики по-прежнему могут использовать произвольные средства для подготовки исходных текстов программ, как правило, они все же предпочитают пользоваться именно тем текстовым редактором, который включен в состав данной системы программирования.

1.3. Функции компилятора, компоновщика, загрузчика

1.3.1. Компилятор как составная часть системы программирования

Компиляторы являются, безусловно, основными модулями в составе любой системы программирования. Без компилятора никакая система программирования не имеет смысла, а все остальные ее составляющие на самом деле служат лишь целям обеспечения работы компилятора и выполнения им своих функций.

От первых этапов развития систем программирования вплоть до появления интегрированных сред разработки пользователи (разработчики исходных программ) всегда так или иначе имели дело с компилятором. Они непосредственно взаимодействовали с ним как с отдельным программным модулем.

Сейчас, работая с системой программирования, пользователь, как правило, имеет дело только с ее интерфейсной частью, которую обычно представляет текстовый редактор с расширенными функциями. Запуск модуля компилятора и вся его работа происходят автоматически и скрытно от пользователя – разработчик видит только конечные результаты выполнения компилятора. Хотя многие современные системы программирования сохранили прежнюю возможность непосредственного взаимодействия разработчика с компилятором (это и Makefile, и так называемый "интерфейс командной строки"), но пользуется этими средствами только узкий круг профессионалов. Большинство пользователей систем программирования сейчас редко непосредственно сталкиваются с компиляторами.

На самом деле, кроме самого основного компилятора, выполняющего перевод исходного текста на входном языке в язык машинных команд, большинство систем программирования могут содержать в своем составе целый ряд других компиляторов и трансляторов. Так, большинство систем программирования содержат в своем составе и компилятор с языка ассемблера, и компилятор (транслятор) с входного языка описания ресурсов. Все они редко непосредственно взаимодействуют с пользователем.

Тем не менее, работая с любой системой программирования, следует помнить, что основным модулем ее всегда является компилятор. Именно технические характеристики компилятора, прежде всего, влияют на эффективность результирующих программ, порождаемых системой программирования.

1.3.2. Компоновщик. Назначение и функции компоновщика

Компоновщик (или редактор связей) предназначен для связывания между собой объектных файлов, порождаемых

компилятором, а также файлов библиотек, входящих в состав системы программирования.

Объектный файл (или набор объектных файлов) не может быть исполнен до тех пор, пока все модули и секции не будут в нем увязаны между собой. Это и делает редактор связей (компоновщик). Результатом его работы является единый файл (часто называемый "исполняемым файлом"), который содержит весь текст результирующей программы на языке машинных кодов. Компоновщик может порождать сообщение об ошибке, если при попытке собрать объектные файлы в единое целое он не смог обнаружить какой-либо необходимой составляющей.

Функция компоновщика достаточно проста. Он начинает свою работу с того, что выбирает из первого объектного модуля программную секцию и присваивает ей начальный адрес. Программные секции остальных объектных модулей получают адреса относительно начального адреса в порядке следования. При этом может выполняться также функция выравнивания начальных адресов программных секций. Одновременно с объединением текстов программных секций объединяются секции данных, таблицы идентификаторов и внешних имен. Разрешаются межсекционные ссылки.

Процедура разрешения ссылок сводится к вычислению значений адресных констант процедур, функций и переменных с учетом перемещений секций относительно начала собираемого программного модуля. Если при этом обнаруживаются ссылки к внешним переменным, отсутствующим в списке объектных модулей, редактор связей организует их поиск в библиотеках, доступных в системе программирования. Если же и в библиотеке необходимую составляющую найти не удается, формируется сообщение об ошибке.

Обычно компоновщик формирует простейший программный модуль, создаваемый как единое целое. Однако в более сложных случаях компоновщик может создавать и другие модули: программные модули с оверлейной структурой, объектные модули библиотек и модули динамически подключаемых библиотек.

Возможности системы программирования во многом определяет состав доступных библиотек подпрограмм. Объектный код библиотеки подключается компоновщиком к результирующей программе при создании исполняемого модуля. Принципиально новые возможности предоставили современные операционные системы (ОС), которые позволили подключать к результирующим программам не статические, а динамические библиотеки. Динамические библиотеки в отличие от традиционных (статических) библиотек подключаются к программе не в момент ее компоновки, а непосредственно в ходе выполнения, как только программа затребовала ту или иную функцию, находящуюся в библиотеке. Такие библиотеки не требуют включать в программу объектный код часто используемых функций, чем существенно сокращают объем кода. Широкий набор динамических библиотек

поддерживается всеми современными ОС. Как правило, они содержат системные функции ОС и общедоступные функции программного интерфейса (API).

1.3.3. Функции загрузчика

Большинство объектных модулей в современных системах программирования строятся на основе так называемых относительных адресов. Компилятор, порождающий объектные файлы, а затем и компоновщик, объединяющий их в единое целое, не могут знать точно, в какой реальной области памяти компьютера будет располагаться программа в момент ее выполнения. Поэтому они работают не с реальными адресами ячеек оперативных запоминающих устройств (ОЗУ), а с некоторыми относительными адресами. Такие адреса отсчитываются от некоторой условной точки, принятой за начало области памяти, занимаемой результирующей программой (обычно это точка начала первого модуля программы).

Конечно, ни одна программа не может быть исполнена в этих относительных адресах. Поэтому требуется модуль, который бы выполнял преобразование относительных адресов в реальные (абсолютные) адреса непосредственно в момент запуска программы на выполнение. Этот процесс называется трансляцией адресов и выполняет его специальный модуль, называемый загрузчиком.

Однако загрузчик не всегда является составной частью системы программирования, поскольку выполняемые им функции очень зависят от архитектуры целевой вычислительной системы, в которой выполняется результирующая программа, созданная системой программирования. На первых этапах развития ОС загрузчики существовали в виде отдельных модулей, которые выполняли трансляцию адресов и готовили программу к выполнению – создавали так называемый "образ задачи". Такая схема была характерна для многих ОС (например, для ОСРВ на ЭВМ типа СМ-1, ОС RSX/11 или RAFOS на ЭВМ типа СМ-4 и т. п.). Образ задачи можно было сохранить на внешнем носителе или же создавать его вновь всякий раз при подготовке программы к выполнению.

С развитием архитектуры вычислительных средств компьютера появилась возможность выполнять трансляцию адресов непосредственно в момент запуска программы на выполнение. Для этого потребовалось в состав исполняемого файла включить соответствующую таблицу, содержащую перечень ссылок на адреса, которые необходимо подвергнуть трансляции. В момент запуска исполняемого файла ОС обрабатывала эту таблицу и преобразовывала относительные адреса в абсолютные. Такая схема, например, характерна для ОС типа MS-DOS. В этой схеме модуль загрузчика как таковой отсутствует (фактически он входит в состав ОС), а система

программирования ответственна только за подготовку таблицы трансляции адресов – эту функцию выполняет компоновщик.

В современных ОС существуют сложные методы преобразования адресов, которые работают непосредственно уже во время выполнения программы. Эти методы основаны на возможностях, аппаратно заложенных в архитектуру вычислительных комплексов. Методы трансляции адресов могут быть основаны на сегментной, страничной и сегментно-страничной организации памяти. Тогда для выполнения трансляции адресов в момент запуска программы должны быть подготовлены соответствующие системные таблицы. Эти функции целиком ложатся на модули ОС, поэтому они не выполняются в системах программирования.

1.4. Отладчики и отладка программ

Еще одним модулем системы программирования, функции которого тесно связаны с выполнением программы, является отладчик.

Отладчик – это программный модуль, который позволяет выполнить основные задачи, связанные с мониторингом процесса выполнения результирующей прикладной программы. Этот процесс называется отладкой и включает в себя следующие основные возможности:

- последовательное пошаговое выполнение результирующей программы на основе шагов по машинным командам или по операторам входного языка;
- выполнение результирующей программы до достижения ею одной из заданных точек останова (адресов останова);
- выполнение результирующей программы до наступления некоторых заданных условий, связанных с данными и адресами, обрабатываемыми этой программой;
- просмотр содержимого областей памяти, занятых командами или данными результирующей программы.

Первоначально отладчики представляли собой отдельные программные модули, которые могли обрабатывать результирующую программу в терминах языка машинных команд. Их возможности в основном сводились к моделированию выполнения результирующих программ в архитектуре соответствующей вычислительной системы. Выполнение могло идти непрерывно либо по шагам.

Дальнейшее развитие отладчиков связано со следующими принципиальными моментами:

- появлением интегрированных сред разработки;
- появлением возможностей аппаратной поддержки средств отладки во многих вычислительных системах.

Первый из этих шагов дал возможность разработчикам программ работать не в терминах машинных команд, а в терминах исходного языка программирования, что значительно сократило трудозатраты на отладку

программного обеспечения. При этом отладчики перестали быть отдельными модулями и стали интегрированной частью систем программирования, поскольку они должны были теперь поддерживать работу с таблицами идентификаторов и выполнять задачу, обратную идентификации лексических единиц языка. Это связано с тем, что в такой среде отладка программы идет в терминах имен, данных пользователем, а не в терминах внутренних имен, присвоенных компилятором. Соответствующие изменения потребовались также в функциях компиляторов и компоновщиков, поскольку они должны были включать таблицу имен в состав объектных и исполняемых файлов для ее обработки отладчиком.

Второй шаг позволил значительно расширить возможности средств отладки. Теперь для них не требовалось моделировать работу и архитектуру соответствующей вычислительной системы. Выполнение результирующей программы в режиме отладки стало возможным в той же среде, что и в обычном режиме. В задачу отладчика входили только функции перевода вычислительной системы в соответствующий режим перед запуском результирующей программы на отладку. Во многом эти функции являются приоритетными, поскольку зачастую требуют установки системных таблиц и флагов процессора вычислительной системы.

Отладчики в современных системах программирования представляют собой модули с развитым интерфейсом пользователя, работающие непосредственно с текстом и модулями исходной программы. Многие их функции интегрированы с функциями текстовых редакторов исходных текстов, входящих в состав систем программирования.

Рассмотрим возможности отладчика системы программирования Delphi. Это мощный отладчик, встроенный непосредственно в интегрированную среду разработки. Набор поддерживаемых им функций включает все, что можно ожидать от отладчика: трассировку и пошаговое выполнение, установку точек останова, добавление и просмотр контролируемых значений, вычисление и модификацию данных, а также просмотр содержимого стека.

Точки останова (breakpoints) или просто остановы, позволяют при выполнении определенных условий приостановить работу программы. Чаще всего точки останова размещаются в определенной строке кода, при этом останов происходит в тот момент, когда данная строка должна начать выполняться. Такой останов можно установить, например, щелкнув слева от строки кода. Если необходимо проанализировать поведение программы внутри определенной процедуры или функции, достаточно просто установить точку останова в ее первой строке.

Условная точка останова. Можно задать дополнительное условие к точке останова, и тогда программа будет приостанавливаться по достижении определенной строки кода только при выполнении этого

условия. Типичным случаем применения такой точки останова будет проверка кода внутри цикла. Вероятно, вы не захотите останавливать и опять запускать программу всякий раз, когда выполняется цикл – а это происходит сотни, а то и тысячи раз. Вместо того, чтобы непрерывно производить повторный запуск программы, можно установить точку останова по достижении какой-то переменной определенного значения.

Точка останова по обращению к данным. Этот тип точки останова приостанавливает выполнение программы при модификации определенного участка памяти. Он применяется для низкоуровневой отладки, когда отслеживаются ошибки присвоения значений переменным. Можно также ввести начальный адрес области памяти, которую требуется контролировать, и ее размер (в байтах). Устанавливая необходимый размер, можно контролировать переменную любого типа – от Char (1 байт) и Integer (4 байта) до массива или записи произвольного размера. Как и в случае точки останова по условию, можно ввести выражение, которое будет критерием останова при изменении области памяти. Это позволяет выявлять ошибки, которые проявляются только после n-го обновления значения переменной.

Точка останова по адресу. Останов по адресу реализуется, когда выполняется инструкция, находящаяся в памяти по заданному адресу. Эти точки обычно устанавливаются, если невозможно установить обычную точку останова в определенной строке кода из-за отсутствия исходного кода для соответствующего модуля. Как и в случае других видов точек останова, можно задать условия приостановки выполнения программы.

Точка останова по загрузке модуля, как следует из названия останавливает выполнение отлаживаемой программы при загрузке указанного модуля.

Группы точек останова представляют собой одну из самых мощных и эффективных функций интегрированного отладчика. С помощью механизма групп, любая точка останова может быть активизирована или деактивизирована другой точкой останова, что позволяет строить весьма сложные алгоритмы функционирования групп точек останова, предназначенные для обнаружения самых трудноуловимых и специфических ошибок.

Пошаговое выполнение программы. Программу можно выполнять последовательно, строка за строкой, используя команды Step Over или Trace Into (по умолчанию клавиши <F8> и <F7> соответственно). Команда Trace Into при выполнении программы обеспечивает вход в вызываемые процедуры и функции, а команда Step Over немедленно их выполняет и представляет как одно действие. Этими опциями удобно пользоваться после останова программы в каком-то месте ее текста.

Можно также дать указание Delphi выполнять программу до того места, в котором в настоящий момент находится курсор, – с помощью команды Run to Cursor (клавиша <F4>). Эту возможность удобно

использовать для пропуска многократно выполняющегося цикла, чтобы непрерывно не нажимать клавиши <F8> и <F7>.

Использование окна инспектора данных (Watch). Окно Watch можно использовать для контроля значений переменных по ходу выполнения программы. Программа должна находиться в режиме просмотра кода программы (т.е. выполняться какая-либо из точек останова) – только в этом случае содержимое окна Watch будет корректным. В этом окне можно ввести некоторое выражение Object Pascal или указать определенное в программе имя.

Инспекторы отладки (Debug Inspector) представляют собой разновидность окна инспектора данных. Они обладают большими возможностями, и их легче использовать, чем окно Watch. В окне инспектора отладки можно просматривать содержимое данных, состоящих из множества индивидуальных элементов, например, таких как классы и записи.

Использование команд Evaluate и Modify. Команды Evaluate и Modify позволяют соответственно просматривать и изменять содержимое переменных, включая массивы и записи, "на лету", во время выполнения приложения, в интегрированном отладчике. (Однако они не предоставляют доступ к функциям и переменным вне области видимости.)

Доступ к стеку вызовов (Call Stack) позволяет просмотреть вызовы функций и процедур вместе с переданными им параметрами, в той последовательности, в которой они выполнялись до определенного момента выполнения программы.

Просмотр потоков (Thread Status). Если приложение использует множество потоков, встроенный отладчик позволяет получить информацию о каждом из них.

Журнал событий (Event Log) – это место, в которое отладчик записывает информацию о различных событиях. Типы событий, сведения о которых могут помещаться в журнал, включают запуск и останов процесса, загрузку модулей отладчиком.

Окно CPU содержит пять информационных панелей: CPU, Memory Dump, Register, Flags и Stack. С его помощью разработчик получает возможность узнать, что именно происходит в машине. Каждая из панелей позволяет во время отладки следить за важными аспектами функционирования процессора.

В панели CPU отображаются коды операций и мнемоники дизассемблированного кода, выполняемого в данный момент. Можно просмотреть код приложения по любому адресу и произвольно выбрать новую инструкцию для текущего выполнения. Это позволит изучить поведение ассемблерного кода программы. Опытным разработчикам прекрасно известно, что множество ошибок находится и устраняется при проверке сгенерированного ассемблерного кода.

Контекстное меню окна CPU позволяет настроить внешний вид окна, просмотреть различные адреса, перейти к инструкции, выполняемой в

данный момент, осуществить поиск, вернуться к просмотру исходного кода и т.п. Кроме того, можно выбрать контекст потока, в котором будет просматриваться информация CPU.

На панели Memory Dump можно просмотреть содержимое любой области памяти. Это содержимое может быть представлено по-разному: как Byte, Word, DWORD, QWORD, Single, Double или Extended. Можно выполнить поиск в памяти определенной последовательности байтов, модифицировать текущие данные и перейти к следующим, либо последовательно, либо используя текущие данные в качестве указателя.

Назначение панелей Register и Flags очевидно: в них отображаются и могут быть изменены все регистры и флаги процессора.

На панели Stack можно просмотреть память, используемую приложением в качестве стека. Можно также изменять значения и перемещаться по адресам.

1.5. Система подсказок и справок

Лексический анализ "на лету" – это функция текстового редактора в составе системы программирования. Она заключается в поиске и выделении лексем входного языка в тексте программы непосредственно в процессе ее создания разработчиком.

Реализуется это следующим образом: разработчик создает исходный текст программы (набирает его или получает из некоторого другого источника), и в то же время система программирования параллельно выполняет поиск лексем в этом тексте.

В простейшем случае обнаруженные лексемы просто выделяются в тексте с помощью графических средств интерфейса текстового редактора – цветом, шрифтом и т. п. Это облегчает труд разработчика программы, делает исходный текст более наглядным и способствует обнаружению ошибок на самом раннем этапе – на этапе подготовки исходного кода.

В более развитых системах программирования найденные лексемы не просто выделяются по ходу подготовки исходного текста, но и помещаются в таблицу идентификаторов компилятора, входящего в состав системы программирования. Такой подход позволяет экономить время на этапе компиляции, поскольку первая ее фаза – лексический анализ – уже выполнена на этапе подготовки исходного текста программы.

Следующей сервисной возможностью, предоставляемой разработчику системой программирования за счет лексического анализа "на лету", является возможность обращения разработчика к таблице идентификаторов в ходе подготовки исходного текста программы. Разработчик может дать компилятору команду найти нужную ему лексему в таблице. Поиск может выполняться по типу или по какой-то части информации лексемы (например, по нескольким первым

буквам). Причем поиск может быть контекстно-зависимым – система программирования предоставит разработчику возможность найти лексему именно того типа, который может быть использован в данном месте исходного текста. Кроме самой лексемы разработчику может быть предоставлена некоторая информация о ней – например, типы и состав формальных параметров для функции, перечень доступных методов для типа или экземпляра класса. Это опять же облегчает труд разработчика, поскольку избавляет его от необходимости помнить состав функций и типов многих модулей (прежде всего, библиотечных) или обращаться лишний раз к документации и справочной информации.

Лексический анализ "на лету" – мощная функция, значительно облегчающая труд, связанный с подготовкой исходного текста. Она входит не только в состав многих систем программирования, но также и в состав многих текстовых редакторов, поставляемых отдельно от систем программирования (в последнем случае она позволяет настроиться на лексику того или иного языка).

Другой удобной сервисной функцией в современных системах программирования является система подсказок и справок. Как правило, она содержит три основные части:

- справку по семантике и синтаксису используемого входного языка;
- подсказку по работе с самой системой программирования;
- справку о функциях библиотек, входящих в состав системы программирования.

Система подсказок и справок в настоящее время является составной частью многих прикладных и системных программ. Как правило, она поддерживается соответствующими утилитами ОС. Поэтому, кроме всего прочего, многие системы программирования включают в свой состав сервисные функции, позволяющие создавать и дополнять систему подсказок и справок. Это делается таким образом, чтобы разработчик мог создавать и распространять вместе со своими прикладными программами соответствующие им подсказки и справки.

1.6. Примеры современных систем программирования

1.6.1. Системы программирования компании Borland/Inprise

Системы программирования компании Borland достаточно широко известны разработчикам в России. Известность и распространенность этих систем программирования определила, прежде всего, простота их использования, поскольку именно в системах программирования этой компании были впервые реализованы на практике идеи интегрированной среды программирования.

Borland Delphi. Система программирования Borland Delphi явилась логическим продолжением и дальнейшим развитием идей, заложенных

компанией-разработчиком еще в системе программирования Turbo Pascal.

В качестве основных в новой системе программирования можно указать следующие принципиальные изменения:

- новый язык программирования – Object Pascal, явившийся серьезной переработкой прежней версии языка Borland Pascal;
- компонентная модель среды разработки, в первую очередь ориентированная на технологию разработки RAD (rapid application development).

Язык программирования Object Pascal создавался в то время, когда на рынке средств разработки уже существовало значительное количество объектно-ориентированных языков, включая такие известные, как C++ и Java. Компания Borland попыталась учесть все недостатки существующих языков объектно-ориентированного программирования, а также свой опыт создания языка Borland Pascal. Новый язык вышел довольно удачным как с точки зрения синтаксиса, так и с точки зрения предоставляемых возможностей. Этот язык поддерживает практически все основные механизмы объектно-ориентированного программирования.

Компонентная модель среды разработки предусматривает создание основной части программы в виде набора взаимосвязанных компонентов – классов объектно-ориентированного языка. Во время разработки исходной программы (design time) компоненты предстают в виде графических образов и обозначений, связанных между собой. Каждый компонент обладает определенным набором свойств (properties), событий (events) и методов. Каждому из них соответствует свой фрагмент исходного кода программы, отвечающий за обработку метода или реакции на какое-то событие. Разработчик может располагать на экране и связывать между собой компоненты, а также редактировать связанный с ними исходный код программы. Причем поведение компонентов во время выполнения программы (run time) полностью определяется их взаимосвязью, исходным кодом программы и объектным кодом самого компонента.

Система программирования Borland Delphi предназначена для создания результирующих программ, выполняющихся в среде ОС Windows различных типов и ОС Linux.

Основу системы программирования Borland Delphi и ее компонентной модели составляет библиотека визуальных компонентов VCL (visual component library). В этой библиотеке реализованы в виде компонентов все основные органы управления и интерфейса ОС. Также в ее состав входят классы, обеспечивающие разработку приложений для архитектуры "клиент–сервер" и трехуровневой архитектуры (в современных реализациях Borland Delphi). Разработчик имеет возможность не только использовать любые компоненты, входящие в состав библиотеки VCL, но также и разрабатывать свои собственные

компоненты, основанные на любом из классов данной библиотеки. Эти новые компоненты становятся частью системы программирования и затем могут быть использованы другими разработчиками.

Для поддержки разработки результирующих программ для архитектуры "клиент–сервер" в состав Borland Delphi входит средство BDE (Borland database engine). Оно обеспечивает результирующим программам возможность доступа к широкому диапазону серверов БД посредством классов библиотеки VCL. Посредством BDE результирующая программа может взаимодействовать с серверами БД типа Microsoft SQL Server, InterBase, Sybase, Oracle и т. п. Система программирования Borland Delphi поддерживает также создание результирующих программ, выполняющихся в архитектуре "клиент–сервер", на базе других технологий, например ADO (ActiveX Data Objects).

Система программирования Borland Delphi выдержала несколько реализаций. Последние реализации данной системы программирования включают широкий набор средств для поддержки разработки результирующих программ в трехуровневой архитектуре приложений. Система программирования Borland Delphi позволяет разрабатывать как серверную, так и клиентскую часть приложения в данной архитектуре. Возможно использование как технологий COM/DCOM (наиболее распространенных в среде ОС типа Microsoft Windows), так и технологии CORBA (но только при разработке клиентской части приложения), а также межплатформенная разработка приложений (Windpws, Linux), разработка Web-приложений.

В качестве недостатков данной системы программирования можно указать использование нестандартного формата объектных файлов (сохранился еще от системы Turbo Pascal, но в версии Borland Delphi 5 уже можно использовать стандартный формат), а также нестандартного формата для хранения ресурсов пользовательского интерфейса. Кроме того, сам язык Object Pascal не является признанным стандартом. Этот факт несколько затрудняет использование Borland Delphi в масштабных проектах в качестве основного средства разработки.

Тем не менее система программирования Borland Delphi получила широкое распространение среди разработчиков в России.

Borland C++ Builder. Система программирования Borland C++ Builder объединила в себе идеи интегрированной среды разработки, реализованные компанией в системах программирования Turbo Pascal и Borland Delphi с возможностями языка программирования C++. История этой системы программирования начинается с интегрированной среды разработки Borland Turbo C.

Современная реализация Borland C++ Builder ориентирована на разработку результирующих программ, выполняющихся под управлением ОС Microsoft Windows всех типов. Сама система программирования Borland C++ Builder, как и Borland Delphi, также функционирует под управлением ОС типа Microsoft Windows. Он

полностью поддерживает стандарт языка С, что делает возможным создание с помощью данной системы программирования модулей и библиотек, используемых в других средствах разработки (чего очень сложно достигнуть с помощью Borland Delphi).

По возможностям, внешнему виду и технологиям система программирования Borland C++ Builder схожа с системой программирования Borland Delphi. В ее основу положены те же основные идеи и технологии. Структура классов языка C++ в системе программирования Borland C++ Builder построена в той же библиотеке VCL (visual control library), в которой строится структура классов Object Pascal в системе программирования Borland Delphi. Правда, разработчик, создающий программы на C++, может не пользоваться классами VCL и взять за основу любую другую библиотеку, чего нельзя сказать о разработчике, использующем Object Pascal – набор доступных библиотек для последнего языка сильно ограничен.

Успешное распространение систем программирования Turbo Pascal и Borland Delphi способствовало и внедрению на рынок системы программирования Borland C++ Builder от той же компании-разработчика. Эта система программирования занимает прочную позицию на рынке средств разработки для языка C++, где существует довольно жесткая конкуренция.

1.6.2. Системы программирования фирмы Microsoft

Компания Microsoft в настоящее время как производитель операционных систем и программного обеспечения доминирует на рынке персональных компьютеров, построенных на базе процессоров фирмы Intel. Прежде всего, это относится ко всем вариантам ОС типа Microsoft Windows.

Этот факт явился одним из главных факторов, которые обусловили прочную позицию данной компании на рынке средств разработки программных продуктов для ОС типа Microsoft Windows. Все виды ОС типа Microsoft Windows создавались как закрытые системы. Поэтому безусловное знание компанией-разработчиком структуры и внутреннего устройства "своей" ОС зачастую являлось определяющим в ситуации, когда надо было создать средство разработки приложений для данной ОС. Хорошие финансовые ресурсы и положение компании на рынке позволили ей создать довольно удачные системы программирования, несмотря на то, что она начала их разработку довольно поздно и не являлась "законодателем мод" в данной области.

Microsoft Visual Basic. Это средство разработки прошло долгую историю под руководством компании Microsoft. История языка Basic на персональных компьютерах началась с примитивных интерпретаторов данного языка. Сам по себе язык Basic позволял легко организовать интерпретацию исходного кода программ, а его синтаксис и семантика

достаточно просты для понимания даже непрофессиональными разработчиками.

Система программирования Microsoft Visual Basic также первоначально была ориентирована на интерпретацию исходного кода. Однако требования и условия на рынке средств разработки толкнули компанию-производителя на создание компилятора, вошедшего в состав данной системы программирования. При этом основные функции библиотеки языка были вынесены в отдельную, динамически подключаемую библиотеку VBRun, которая должна присутствовать в ОС для выполнения результирующих программ, созданных с помощью данной системы программирования. Различные версии системы программирования Microsoft Visual Basic ориентированы на различные версии данной библиотеки. Интерпретатор языка был сохранен и внедрен компанией-разработчиком в состав модулей другого программного продукта – Microsoft Office.

Развитие системы программирования Visual Basic потребовало существенного изменения синтаксиса и семантики самого языка. Последняя версия данной системы программирования – Microsoft Visual Basic 6.0 – является одним из эффективных средств для создания результирующих программ, ориентированных на выполнение под управлением ОС типа Microsoft Windows. Эта система программирования ориентирована на технологию разработки RAD. Microsoft Visual Basic 6.0 содержит интегрированные средства визуальной работы с базами данных, поддерживающие проектирование и доступ к базам данных SQL Server, Oracle и т. п. К этим средствам относятся Visual Database Tools, ADO/OLE DB, Data Environment Designer, Report Designer и ряд других.

В данной системе программирования поддерживается также создание серверных Web-приложений, работающих с любым средством просмотра на базе новых Web-классов. В Microsoft Visual Basic 6.0 возможно создание интерактивных Web-страниц.

Microsoft Visual Basic 6.0 обеспечивает простое создание приложений, ориентированных на данные. Visual Basic 6.0 позволяет создавать результирующие программы, выполняемые в архитектуре "клиент–сервер", которые могут работать с любыми базами данных. Система программирования Microsoft Visual Basic ориентирована, прежде всего, на создание клиентской части приложений.

Теперь Visual Basic 6.0 поддерживает универсальный интерфейс доступа к данным Microsoft при помощи технологии ADO. Visual Basic 6.0 обеспечивает просмотр таблиц, изменение данных, создание запросов SQL из среды разработки для любой совместимой с ODBC или OLE DB базы данных. Так же как и в редакторе Visual Basic, синтаксис SQL выделяется цветом и незамедлительно проверяется на наличие ошибок. Это делает код SQL легче читаемым и менее подверженным случайным ошибкам.

Среда разработки обладает множеством новых возможностей, таких как выделение синтаксиса и автоматическое завершение ключевых слов. Система программирования Microsoft Visual Basic интегрируется с семейством программных продуктов Microsoft BackOffice, которое обеспечивает среду для выполнения и создания сложных приложений масштаба предприятия для работы в локальных сетях или в Интернете. Использование новых интегрированных визуальных средств работы с данными облегчает выполнение рутинных задач но обеспечению доступа к ним; эти средства доступны прямо из среды разработки Visual Basic.

Система программирования Visual Basic неплохо сочетает в себе простоту и эффективность разработки. Все недостатки, присущие данной системе, в большинстве своем проистекают из недостатков используемого исходного языка программирования. Средства языка Basic даже после значительной модификации ограничивают возможности его применения в современных архитектурах взаимодействия приложений, которые в значительной мере основаны на объектно-ориентированном подходе. Кроме того, язык программирования в системе Visual Basic не является признанным стандартом, а потому возникают трудности по использованию созданных на его основе модулей и компонентов в других средствах разработки.

Microsoft Visual C++. Система программирования Microsoft Visual C++ представляет собой реализацию среды разработки для распространенного языка системного программирования C++, выполненную компанией Microsoft. Эта система программирования в настоящее время построена в виде интегрированной среды разработки, включающей в себя все необходимые средства для разработки результирующих программ, ориентированных на выполнение под управлением ОС типа Microsoft Windows различных версий.

Основу системы программирования Microsoft Visual C++ составляет библиотека классов MFC (Microsoft foundation classes). В этой библиотеке реализованы в виде классов C++ все основные органы управления и интерфейса ОС. Также в ее состав входят классы, обеспечивающие разработку приложений для архитектуры "клиент–сервер" и трехуровневой архитектуры (в современных версиях библиотеки). Система программирования Microsoft Visual C++ позволяет разрабатывать любые приложения, выполняющиеся в среде ОС типа Microsoft Windows, в том числе серверные или клиентские результирующие программы, осуществляющие взаимодействие между собой по одной из указанных выше архитектур.

Классы библиотеки MFC ориентированы на использование технологий COM/DCOM, а также построенной на их основе технологии ActiveX для организации взаимодействия между клиентской и серверной частью разрабатываемых приложений. На основе классов библиотеки

пользователь может создавать свои собственные классы в языке C++, организовывать свои структуры данных.

В отличие от систем программирования компании Borland, система программирования Microsoft Visual C++ ориентирована на использование стандартных средств хранения и обработки ресурсов интерфейса пользователя в ОС Windows. Это не удивительно, поскольку все версии ОС типа Windows разрабатываются самой компанией Microsoft. Microsoft Visual C++ обеспечивает все необходимые средства для создания профессиональных Windows-приложений. От версии к версии продукт становится проще в использовании, расширяются возможности применения, повышается производительность.

Система программирования Microsoft Visual C++ выдержала несколько реализаций. В процессе выхода новых версий системы программирования было выпущено и несколько версий библиотеки MFC, на которой основана данная система. Visual C++ 6.0 полностью интегрируется с Visual Studio 6.0 и другими средствами разработки, входящими и состав данного пакета.

2. ТЕСТИРОВАНИЕ И ДОКУМЕНТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

2.1. Этапы и виды тестирования

Тестирование является основным методом отладки, измерения качества и определения реальных характеристик программ и информации баз данных на любых этапах их жизненного цикла. Результаты тестирования и измерения показателей качества должны сравниваться с требованиями технического задания, спецификаций или иных эталонов для определения степени соответствия предъявлявшимся требованиям, полученным разработчиком от заказчика. Наличие достаточно полных эталонов, таких как совокупность требований технического задания и поэтапная их декомпозиция в спецификациях, является необходимой базой тестирования программ.

Цель тестирования и отладки – обнаружение, локализация и устранение дефектов в программах и данных. Важной особенностью тестирования сложных программ является необходимость достаточно полной их проверки при ограниченных ресурсах и длительности испытаний. Это определяет целесообразность тщательного планирования и документирования тестирования с учетом всех результатов, полученных на предыдущих этапах разработки. При планировании основная задача состоит в достижении максимальной достоверности испытаний и определения качества программных компонентов при ограниченных затратах ресурсов всех видов на проведение тестирования.

Полное исчерпывающее тестирование, гарантирующее абсолютную проверку программ недостижимо при существующей сложности средних по объему и крупных программных компонентах и доступных ресурсах. Поэтому тестирование проводится в объемах минимально необходимых для проверки программ в ограниченных пределах изменения исходных данных и условий функционирования. Ограниченность ресурсов отладки приводит к целесообразности тщательного упорядочения и применения экономичных и эффективных методов и средств автоматизации тестирования с целью получения наибольшей глубины проверок программ.

Достаточный объем тестирования реально оценивается субъективно, и процессы отладки строятся с позиции необходимого минимума тестов, обеспечивающих корректность в предполагаемых наиболее активных режимах эксплуатации программ. Такой подход является конструктивным и реализуемым, позволяет сохранять затраты на тестирование в разумных пределах. Для сложных программ объем полного, исчерпывающего тестирования может на несколько порядков превышать необходимое и достаточное тестирование.

Процесс тестирования проходит следующие этапы:

- выбор метода тестирования адекватного объекту, а также основной цели его выполнения;
- планирование тестирования в соответствии с выбранным методом с учетом ограниченных ресурсов испытаний, имеющихся для достижения заданной достоверности проверки качества программы;
- разработку или моделирование наборов конкретных тестовых значений и соответствующих им эталонов;
- составление заданий на тестирование с указанием контролируемых параметров, исходных данных и эталонов;
- реализацию процесса тестирования и получение результатов функционирования объекта испытаний при подготовленных тестах и заданиях;
- сравнение результатов тестирования с эталонами и обнаружение отклонений для принятия решений о проведении дополнительного тестирования с целью диагностики и локализации дефектов;
- оценку полноты проведенного тестирования выбранным методом и необходимости применения других методов тестирования;
- оценку наличия ресурсов для продолжения испытаний и момента их завершения, а также определение достигнутого качества программ или выбор очередного метода тестирования;
- документирование результатов тестирования и отладки программ.

Следует выделять два крупных этапа и соответствующие им виды тестирования и комплексной отладки, а также состав сопровождающих документов:

- тестирование компонентов – модулей и функциональных групп программ;

• тестирование и отладку программных средств (ПС) в целом.

В сложных ПС каждому такому этапу соответствуют более мелкие – частные процессы и работы. На них преследуются различные цели и применяются иногда одни и те же методы и средства обеспечения тестирования в различных сочетаниях. Каждый из выделенных этапов тестирования может рассматриваться как обеспечивающий создание определенного промежуточного продукта – функциональной группы программ или программного средства с некоторыми ограниченными характеристиками качества. Эти характеристики выделяются и детализируются на основе первичного технического задания и спецификации требований на ПС. В процессе проектирования ПС они уточняются и углубляются в спецификациях требований на группы программ и их компоненты. В результате создается совокупность эталонов, имеющих последовательно расширяющуюся номенклатуру показателей качества, которым должны соответствовать отлаживаемые компоненты на каждом этапе тестирования.

Основные конструктивные элементы программы, подлежащие тестированию и отладке позволяют выделить тестирование потоков управления и тестирование потоков данных. Тестирование потоков управления (структуры программы) является начальным этапом, так как при некорректной структуре возможны наиболее грубые искажения выходных результатов и даже отсутствие некоторых из них. Оно состоит в проверке корректности последовательностей передач управления при обработке тестов при различных исходных данных. Для тестирования структуры программ в большинстве случаев требуются относительно меньшие затраты по сравнению с тестированием на потоках данных.

Тестирование потоков данных можно разделить на два этапа. Первый этап тестирования состоит в анализе обработки данных, определяющих значения предикатов в операторах выработки логических решений. Эти решения влияют на маршруты обработки информации, что сближает метод тестирования потоков данных с тестированием структуры программы. Второй этап тестирования обработки данных состоит в проверке вычислений по аналитическим формулам или численных значений результатов в зависимости от числовых значений исходных данных. В качестве эталонов используются результаты ручных или автоматизированных расчетов по тем же или близким по содержанию формулам.

Практические оценки полноты тестирования осуществляются преимущественно по степени выполнения основных функций в соответствии со спецификацией и по интегральным характеристикам надежности и безопасности функционирования ПС. Значительную помощь в повышении качества, надежности и безопасности применения сложных, критических ПС может оказать расширение и систематизация видов тестирования и упорядочение их проведения: по этапам разработки функциональных компонентов и ПС в целом или по

возможным типам дефектов в программах и данных. Для каждого такого вида тестирования целесообразно разрабатывать частную методику его выполнения с указанием контролируемых параметров и ожидаемых, эталонных результатов. Кроме того, при заключительных испытаниях или сертификации следует проводить интегральное тестирование при максимально широком варьировании типов тестов и сценариев в условиях, соответствующих эксплуатации в типовых и критических ситуациях.

2.2. Принципы и стандарты документирования прикладных программных средств

Создание и применение прикладных ПС сложных информационных систем (ИС) сопровождается документированием этих объектов и процессов их жизненного цикла для обеспечения интерфейса с пользователями, а также для обеспечения возможности освоения и развития функций программных средств и баз данных на любых стадиях проекта ПС. По своему назначению и ориентации на определенные задачи и группы пользователей, документацию ПС можно разделить на:

• **технологическую документацию процесса разработки**, включающую подробные технические описания, и подготавливаемую для специалистов, ведущих проектирование, разработку и сопровождение ПС, обеспечивающую возможность отчуждения, детального освоения, развития и корректировки ими программ и данных на всем жизненном цикле ПС;

• **эксплуатационную документацию продукта и результатов разработки**, создаваемую для конечных пользователей ПС и позволяющую им осваивать и квалифицированно применять эти средства для решения конкретных функциональных задач ИС.

Технологическая документация, непосредственно и в наибольшей степени должна отражать процессы жизненного цикла (ЖЦ) прикладных программ и данных и регламентировать требования к этим документам. Стандарты и нормативные документы, входящие в жизненный цикл ПС, должны выполнять следующие функции:

• регламентировать структуру и состав этапов, работ и документов ЖЦ ПС;

• обеспечивать их адаптацию к характеристикам объекта проектирования, внешней и операционной среды;

• поддерживать и регламентировать процессы организации и планирования реализации ЖЦ конкретных ПС;

• формализовать выполнение и документирование конкретных работ при проектировании, разработке и сопровождении ПС;

• регламентировать процессы обеспечения качества, показатели качества ПС и его компонентов, методы и средства их достижения, реальные значения достигнутых показателей качества.

По функциональному назначению технологическую документацию ПС целесообразно разделить на следующие группы исходных документов:

• базовые документы, определяющие цели и методологию применения конкретной версии ЖЦ ПС;

• ссылочные документы на руководства по организации подобных разработок, включая общесистемные стандарты и нормативные документы различных уровней;

• стандарты и нормативные документы, непосредственно регламентирующие работы и документы жизненного цикла программ, планирование и технологию разработки документации, состав и описание инструментальных средств автоматизации разработки;

• стандарты и нормативные документы, непосредственно используемые при разработке, испытаниях и сопровождении программ на различных этапах.

Технологические документы создаваемых и сопровождаемых прикладных ПС должны определять:

• структуру и содержание исходных и отчетных документов по этапам разработки, испытаний и сопровождения ПС;

• логическую структуру программных и информационных компонентов и базы данных проекта ПС;

• спецификации на внутренние межмодульные интерфейсы компонентов прикладных ПС и на интерфейсы с внешней средой ИС;

• язык и правила программирования, идентификации компонентов, комментирования текстов программ и описаний данных;

• методы тестирования, испытаний и аттестации программных компонентов и ПС в целом;

• оформление, форматы и обозначения отчетных и результирующих документов.

Эксплуатационная документация должна обеспечивать отчуждаемость ПС от их первичных разработчиков и возможность освоения и эффективного применения ПС достаточно квалифицированными специалистами. Состав этой документации формируется выборкой из технологических документов с учетом требований заказчиков или потенциальных пользователей ИС. Эксплуатационные документы должны исключать возможность некорректного использования ПС за пределами условий эксплуатации, при которых документами гарантируются определенные показатели качества функционирования ПС. При формировании эксплуатационных документов прикладных ПС, кроме базовых стандартов жизненного цикла должны использоваться ряд национальных стандартов, ведомственных нормативных документов и фирменных руководств.

Документация разработки образует основу документации сопровождения, которая может являться частью документации продукции, если разработчик допускает изменение программ и данных

пользователями. Эксплуатационная документация ПС включает:

•руководства администраторов и операторов, осуществляющих инсталляцию и непосредственное управление режимами решения функциональных задач, регламентированными в информационной системе;

•руководства операторов-пользователей, использующих ПС по прямому назначению;

•документацию сопровождения ПС, включая руководство по сопровождению и модификации программ и информации баз данных;

•справочные руководства по применению программных средств;

•учебные руководства по освоению программных средств и информационной системы.

Пользовательская эксплуатационная документация может поставляться на традиционных бумажных носителях или на стандартных магнитных носителях для ЭВМ.

Кроме представленных классов формализованных документов в ряде случаев целесообразно подготавливать исследовательскую документацию. Она имеет экспериментальный характер, зависящий от возможных целей исследований. Основная ее задача состоит в фиксировании и обобщении характеристик объектов и процессов всего жизненного цикла ПС и ИС. Этой документацией пользуются в основном руководители, разработчики и исследователи реализации проектов при анализе технологий, планировании новых разработок ПС или их переноса на иные платформы. Из-за разнообразия исследовательских задач этот тип документации практически всегда имеет оригинальный состав и содержание и пока не стандартизируется.

Организация работ по документированию ПС в значительной степени определяет достигаемое качество сложных комплексов программ, трудоемкость и длительность их создания. Она должна определять стратегию, стандарты, процедуры, распределение ресурсов и планы создания, изменения и применения документов на программы и данные информационных систем. Для этого в общем случае должны быть выделены коллективы специалистов и руководители, которые должны планировать, описывать, утверждать, выпускать, распространять и сопровождать комплекты документов.

В процессе установления стратегии, стандартов и руководств по документированию необходимо осуществить:

- выбор модели жизненного цикла ПС и его документов;
- определение типов, содержания и степени детализации каждого документа;
- определение необходимого качества оформления документов;
- определение форматов и системы обозначения документов;
- установление процедур документирования;
- распределение ресурсов для документирования: персонала;

технических средств; финансов, а также на планирование документирования.

Сложные комплексы программ и их компоненты почти непрерывно изменяются в процессе длительного развития проекта в зависимости от множества внешних и внутренних факторов. Соответственно должны изменяться документы, отражающие состояние процессов и компонентов проектов. Для этого организация документирования должна обеспечивать гибкое и точное изменение документов – сопровождение и конфигурационное управление версиями и редакциями каждого документа. Эти процессы и поддерживающие их средства автоматизации должны быть адекватными тем, которые применяются при сопровождении непосредственных объектов разработки – программ и данных. Они должны быть поддержаны организацией контроля, регистрации и утверждения версии каждого документа, в той степени и на том уровне, которые необходимы в данном проекте для определенного документа.

Для хранения, тиражирования и распространения документов сложных ПС высокого качества следует выделять группу специалистов, ответственных за обеспечение и гарантированное сохранение документации. Для критических, важных информационных систем документация на программы и данные должна храниться и дублироваться на различных типах магнитных носителей и эпизодически выводиться на бумажные носители. При определении схемы обеспечения сохранности документации разного содержания следует учитывать ее важность, трудоемкость хранения и возможность аварийного восстановления Кроме того, должна быть организована служба нормоконтроля, ответственная за соблюдение стандартов, нормативных и руководящих документов при подготовке документации всеми специалистами, участвующими в проекте. Эта служба обязана обеспечить унификацию и высокое качество, содержания, структуры и оформления документов.

Общие типовые требования к номенклатуре, структуре и содержанию документов на прикладные ПС представлены в ряде стандартов разного ранга и в фирменных описаниях технологий, поддерживающих жизненный цикл сложных комплексов программ. Рекомендуемый состав и содержание документов широко варьируется в зависимости от класса и характеристик ЖЦ объекта, а также в зависимости от используемой технологии. Наиболее сложному случаю разработки критических ПС высокого качества соответствует самая широкая номенклатура и детализация, применяемых документов. Такой перечень и структуру документов целесообразно использовать как типовые, базовые при адаптации и формировании состава и содержания документов в конкретных и/или более простых случаях.

На базе стандартов ISO 12207 и ISO 9000-3, с учетом нормативных документов создана структура комплекта документов полного жизненного

цикла типовой, базовой версии сложного программного средства ИС, создаваемого "с нуля". Детальную структуру каждого документа целесообразно разрабатывать или уточнять фирмой или ведомством на основе типовой в соответствии с их традициями, используемой технологией и особенностями проектируемого ПС.

Некоторые документы автоматизированно формируются инструментальными средствами, поддерживающими ЖЦ ПС, которые определяют их структуру и содержание. Ряд дополнительных документов при проектировании ПС формируется современными CASE-средствами. В результате номенклатура, структура и содержание документов может значительно варьироваться и расширяться. Формализация структуры и типового содержания каждого документа должна позволять контролировать результаты и качество выполненных работ. В общем случае каждый документ должен иметь:

- титульный лист и наименование документа;
- сформулированное назначение;
- область его действия;
- категории специалистов, для которых он предназначен и кем он разрабатывается;
- этапы работ, на которых следует его применять;
- функциональную, содержательную часть в соответствии с его назначением.

2.3. Типовая структура и содержание базовых комплектов эксплуатационных документов на программы и данные для пользователей программных средств информационных систем

2.3.1. Общие положения

Пользователи программных средств делятся на два крупных класса, каждый из которых должен быть обеспечен комплектной эксплуатационной документацией:

- администраторы, подготавливающие ПС к эксплуатации и обеспечивающие их функционирование и использование по прямому назначению,
- операторы, реализующие функционирование и применение программных средств в информационной системе.

Документация администрирования при эксплуатации информационной системы должна обеспечивать поддержку первичной инсталляции, функционирования и восстановления программ и данных после сбоев. Администратор системы должен быть информирован о всех изменениях функционирования устройств ИС, могущих привести к сбою или возникновению аварийной ситуации, и

предпринимать соответствующие действия. Для этого требуется полная информация о компонентах системы (компьютерах, сетевых устройствах), которые имеют свои особенности в управлении с помощью специального программного обеспечения, поддерживающего администрирование и управление системой.

Управляющая деятельность администратора состоит в манипулировании управляемыми объектами и должна описываться, анализироваться и регламентироваться совокупностью требований и документов в четырех направлениях: информационном; функциональном; коммуникационном и организационном.

С информационной позиции управление прикладными программами и процессами их функционирования должно содержать документирование и описание управляемых объектов и используемых ресурсов. Для каждого управляемого объекта и ресурса необходимо определить их свойства – атрибуты, а также их структуру, объем, сложность и т.п. Кроме того, должна регистрироваться и храниться информация о содержании операций управления и справках (извещениях) о результатах воздействия на них, механизм и форма передачи которых должны быть стандартизированы.

Функциональный аспект документирования административного управления ПС должен соответствовать требованиям конкретной информационной системы на реализацию функций управления и обработку управляющей информации при подготовке и исполнении заданий. Комбинации функций должны удовлетворять общим требованиям на сферу функционирования и обеспечивать корректную реализацию каждой функции.

Коммуникационный аспект определяет взаимодействие пользователей (управляющие объекты) и исполнителей заданий (управляемые объекты) посредством обмена управляющей информацией. Должна быть обеспечена и документирована транспортировка запросов на управляющие операции и извещений о их реализации, а также результатов контроля состояния объектов и процессов взаимодействия с внешней средой.

Организационный аспект состоит в документировании сфер административного управления и общего набора функций, подлежащих реализации. Кроме того, должны быть созданы механизмы контроля соблюдения стандартов и аттестационного тестирования реализации функций администрирования.

Каждый из документов управления должен не противоречить международным стандартам на коммуникацию, интерфейсы с пользователями и базами данных, защиту информации и т.п. Определяющую роль среди них играют стандарты на услуги и протоколы управления, которые регламентируют форму спецификаций выполняемых заданий, способы перемещения документов,

необходимых для их выполнения, и результатов исполнения заданий, а также данных для контроля процессов реализации заданий.

Основу современного *пользовательского интерфейса с ПС* составляют наборы графических элементов и действий над ними, представляемые как меню и системы окон для манипулирования с изображениями. Основные особенности современного интерфейса с пользователями состоят в следующем:

- наличие механизмов управления окнами;
- использование готовых графических символов (икон) для отображения управляемых объектов;
- непосредственное манипулирование графическими объектами и окнами посредством "мыши";
- объектно- и проблемно-ориентированное проектирование диалоговых систем.

В распределенных информационных системах с архитектурой клиент-сервер пользователи непосредственно взаимодействуют с клиентской частью системы, управляющей запуском и режимами работы прикладных программ. Серверные части прикладных программ обеспечивают доступ к данным и вычислительным ресурсам серверов. Система предоставляет пользователю формы документов и наборы процедур, которые он может выполнять. Для реализации такого взаимодействия компоненты клиентской части среды, относящиеся к группе функций пользовательского интерфейса, обеспечивают средства работы с документами, механизмы управления окнами, готовые примитивы символов для формирования нужных объектов, непосредственное манипулирование объектами на экране.

Типовые формы документов и процедуры работы с ними, рассматриваемые как объекты стандартизации, относятся к функциональному уровню взаимодействия пользователей с информационными системами. Объектами стандартизации, соответствующими процедурам из этого перечня являются компоненты интерфейса пользователя, определяющие возможность начала соответствующей операции, ее ход и результат. Должна быть предусмотрена идентификация ошибочных действий и стандартизирована форма сообщения об ошибках. В этих документах должны быть описаны:

- соответствие между элементами интерфейсов пользователя (экранными формами) и типовыми процедурами;
- последовательность допустимых операций и переходы между экранными формами;
- форма идентификации ошибочных действий и/или ситуаций;
- формы входных и выходных документов.

Обучение представляет собой процесс обеспечения и сопровождения обучаемого персонала. Приобретение, поставка, разработка, функционирование и сопровождение программных средств в большой степени зависит от квалификации персонала. Поэтому

обязательно должно быть запланировано и осуществлено обучение персонала с целью подготовки работы персонала при приобретении, поставке, разработке или сопровождения программного средства.

Наиболее полно общие требования к пользовательской документации на программные средства широкого применения представлены в стандарте IEEE 1063-1987 (подтвержден 1993) – "Пользовательская документация на программное обеспечение".

Стандарт определяет минимальные требования к структуре и содержанию комплекта документов для пользователей программных продуктов. Стандарт ориентирован на документы, применяемые при инсталляции, эксплуатации и поставке ПС любого размера и назначения, но без изменения и сопровождения программ. Он не применим для технологической документации, используемой при проектировании, разработке, тестировании, испытаниях и сопровождении ПС, а также для оформления коммерческих пакетов прикладных программ. Использование стандарта не должно препятствовать применению более строгих и широких требований к документам, а также собственных стандартов организаций по стилю изложения документов.

2.3.2. Особенности использования документов на импортные прикладные программные средства

Зарубежным специалистам свойственно ошибаться так же, как и отечественным, однако более высокое качество используемых технологий разработки и современная проектировочная культура позволяют значительно снижать уровень дефектов в изделиях, поступающих на рынок и в эксплуатацию. Тем не менее, любые сложные импортные ПС и их документы всегда не гарантированы от полного, абсолютного отсутствия случайных ошибок. Их применение в отечественных ИС ответственного народно-хозяйственного назначения требует соответствующего дополнительного контроля качества изделий и документов и специальных работ по повышению безопасности эксплуатации.

Разрыв в пространстве и времени при проектировании ИС между первичными создателями импортных ПС и потребителями, непосредственными разработчиками отечественных ИС, затрудняет взаимодействие по предотвращению ошибок за счет применения CASE-технологий. Отечественный покупатель импортных ПС обычно не знает, какая технология была применена при их разработке и какие классы ошибок могли быть предотвращены. Поэтому методы предотвращения ошибок в импортных программах почти всегда остаются недоступными и неизвестными отечественным специалистам. Это отражается на хроническом недоверии к качеству и безопасности применения зарубежных ПС или к слепой вере в их абсолютную безупречность.

Комплексирование готовых импортных прикладных ПС в конкретной ИС создает условия функционирования не всегда адекватные предусмотренным разработчиками в документации и проверенным при испытаниях, хотя возможно и не выходящие за пределы требований документации. Это способствует проявлению ранее скрытых ошибок проектирования и необходимости их устранения. Для этого ответственные и квалифицированные поставщики зарубежных ПС имеют службы сопровождения, регистрации и накопления претензий пользователей, и быстрого реагирования для устранения реальных дефектов функционирования. Легальная закупка и использование лицензионно чистых ПС, обеспеченных полной, комплектной документацией и сопровождением солидной фирмы-поставщика, позволяет в значительной степени снижать влияние на ИС дефектов, не предотвращенных в процессе проектирования.

При закупке зарубежных ПС следует требовать достаточно полный комплект документов, а также сертификат соответствия и сопроводительную документацию по методам и результатам испытаний. В ряде случаев может быть целесообразна дополнительная сертификация изделий отечественными сертификационными центрами. Четкое экономическое и юридическое взаимодействие с определенными фирмами-поставщиками ПС позволяет держать под контролем не только алгоритмическую и программно-технологическую безопасность ИС, но и значительно снижает вероятность злоумышленных дефектов в поставляемых ими изделиях

Порядок представления информации в пользовательской документации на коммерческие пакеты импортных программных средств должен соответствовать международному стандарту ISO 9127.

2.4. Типовая структура и содержание базового комплекта технологических документов разработчиков в жизненном цикле прикладных программных средств информационных систем

2.4.1. Документация при проектировании и разработке программных средств информационных систем

Общие положения. Для представления и детализации структуры и содержания технологической документации жизненный цикл ПС целесообразно разделить на этапы:

- системный анализ и проектирование программного средства, разработка и интегрирование программных компонентов;
- тестирование компонентов и комплексов программ;
- испытания программных средств;

• сопровождение и конфигурационное управление версиями программных средств.

На начальных фазах системного анализа и проектирования воздействия на жизненный цикл ПС заключаются в предварительном прогнозировании технико-экономических показателей проекта ПС и в выборе на их базе технологического процесса и комплекса средств автоматизации для создания программ. Для этого анализируются и выбираются прототипы комплексов программ, которые позволили получить необходимые характеристики. На их основе появляется возможность прогнозировать и документировать процессы разработки и достигаемые показатели качества вновь создаваемого ПС. Этим же целям способствует предварительное распределение ресурсов, доступных для создания проекта.

В процессе эскизного проектирования ПС уточняются характеристики объекта и среды разработки, вследствие чего появляются необходимость и возможность более полно и точно спланировать и документировать весь последующий жизненный цикл ПС. Одновременно уточняются содержание и перечни частных работ и документов, а также приближенные графики последовательности их выполнения. Эти данные позволяют принимать решения по корректировке требований к ПС, по изменению среды разработки или состава коллектива специалистов. Если необходимые требования к ПС не могут быть удовлетворены при доступных ресурсах, технологиях и специалистах, то возможны решения по прекращению дальнейшей разработки.

При техническом проектировании ПС происходит дальнейшее уточнение характеристик объекта и среды разработки, что позволяет перейти к подробному планированию графика выполнения частных работ. Выбор подходящего прототипа может происходить с использованием уточненных исходных данных об объекте и среде разработки. В сведениях о прототипе могут содержаться графики частных работ, которые выполнялись при его создании. Эти данные могут использоваться как первичный план. Для превращения первичного ТЗ и плана в реальный рабочий план обычно требуются их корректировка и уточнения. В результате такой доработки может быть создан детальный график частных работ и подготовки документов для всех последующих этапов. В них должны быть отражены содержание частных работ и ожидаемые результаты, а также ресурсы, необходимые для их реализации (время, число специалистов, трудоемкость и т.п.).

В процессе разработки компонентов, их сборки, комплексной отладки и испытаний версий ПС особенно возрастает роль оперативного управления выполнением и документированием частных работ. Значительный рост числа участников разработки, взаимозависимости частных работ и их исполнителей приводят к необходимости повышения контроля за всем процессом проектирования. Для этого целесообразно

формализовать и документировать технологический процесс и каждую частную работу сопровождать техническим заданием и требованиями на ее выполнение, а также содержанием типового отчета о результатах после ее завершения. Эти результаты должны гарантировать заданное качество отдельных компонентов и процессов выполнения частных работ, обеспечивающих требуемое качество ПС в целом.

Структура и содержание разделов документов. Отчет "Обследование объекта информатизации и формирование требований к программному средству" должен содержать разделы:

- характеристика объекта информатизации;
- описание существующей информационной системы;
- описание недостатков существующей информационной системы;
- обоснование необходимости совершенствования информационной системы;
- цели, критерии и ограничения создания ПС;
- функции и задачи создаваемого ПС;
- ожидаемые технико-экономические результаты создания ПС;
- выводы и предложения;
- рекомендации по созданию ПС.

Отчет "Разработка концепции проектирования ПС" должен содержать:

- описание результатов обследования и изучения объекта информатизации;
- описание и оценку преимуществ и недостатков разработанных альтернативных вариантов концепции создания ПС;
- сопоставительный анализ требований пользователя к ПС и вариантов концепции ПС на предмет удовлетворения требований заказчика и пользователей;
- обоснование выбора оптимального варианта концепции и описание постановки задач ПС;
- ожидаемые результаты и эффективность реализации выбранного варианта концепции ПС;
- ориентировочный план реализации выбранного варианта концепции ПС;
- необходимые затраты ресурсов на разработку, ввод в действие и обеспечение функционирования;
- требования, гарантирующие качество ПС;
- условия испытаний и приемки системы.

"Описание постановки комплекса задач для проектирования ПС" должно содержать разделы:

- характеристики комплекса задач:
  - назначение комплекса задач;
  - перечень объектов (технологических объектов управления, подразделений предприятия и т. п.), при управлении которыми решают комплекс задач;

– периодичность и продолжительность решения;

– условия, при которых прекращается решение комплекса задач автоматизированным способом;

– связи данного комплекса задач с другими комплексами ИС;

– распределение действий между персоналом и техническими средствами при различных ситуациях решения комплекса задач;

• входная информация:

– перечень и описание входных сообщений (идентификаторы, формы представления, сроки и частота поступления);

– перечень и описание структурных единиц информации входных сообщений или ссылка на документы, содержащие эти данные;

– источники информации и их идентификаторы;

• выходная информация:

– перечень и описание выходных сообщений, периодичность выдачи, допустимое время задержки решения;

– получатели и назначение выходной информации.

"Техническое задание на проектирование программного средства" должно содержать поэтапно уточняющиеся и детализирующиеся разделы:

• полное наименование проекта ПС;

• назначение и цель разработки (развития) ПС;

• общие технические требования и базовые нормативные документы для выполнения проекта ПС;

• характеристики объекта информатизации;

• общие требования к программному средству:

• специальные требования к аппаратной и операционной платформам для реализации ПС;

• требования к составу и содержанию работ по внедрению ПС в эксплуатацию;

• ожидаемые результаты и форма их представления,

• предложения по применению и развитию проекта ПС.

"План-график работ" устанавливает перечень работ, сроки выполнения и исполнителей работ, связанных с созданием ПС.

"Ведомость эскизного (технического) проекта" содержит перечень всех документов, разработанных на соответствующих стадиях создания ПС и применяемых из проектов других ИС. Ведомость заполняют по разделам – частям проекта ПС.

"Общее описание программного обеспечения" информационной системы должно содержать:

• основные сведения о техническом, информационном и других видах обеспечения, необходимые для разработки программного обеспечения или ссылку на соответствующие документы проекта ИС;

• структуру программного обеспечения – перечень частей программного обеспечения с указанием их взаимосвязей и обоснованием выделения каждой из них;

• функции частей программного обеспечения – назначение и описание основных функций для каждой части программного обеспечения;

• методы и инструментальные средства разработки программного обеспечения с указанием компонентов ПС, при разработке которых следует использовать соответствующие методы и средства;

• описание операционной системы;

• средства, расширяющие возможности операционной системы.

"Описание функционирования ПС" в техническом проекте и технологической документации должно содержать разделы:

• исходные данные;

• характеристика функциональной структуры;

• типовые решения с указанием функций и задач, для выполнения которых они применены.

Документ "Описание алгоритма" должен содержать разделы:

• назначение и характеристики:

– постановка задачи, для решения которой он предназначен;

– краткие сведения о процессе (объекте), при управлении которым используют алгоритм, а также воздействия на процесс с точки зрения пользователя, осуществляемые при функционировании алгоритма;

– ограничения на возможность и условия применения алгоритма;

– характеристики качества решения;

– общие требования к входным и выходным данным;

• используемая информация и ее характеристики:

– массивы информации, сформированные из входных сообщений;

– массивы информации, полученные в результате работы других алгоритмов и сохраняемые для реализации данного алгоритма;

• результаты решения:

– перечень и содержание массивов информации, формируемых в результате реализации алгоритма;

• математическое описание:

– математическая модель или экономико-математическое описание процесса (объекта);

– перечень принятых допущений и оценки соответствия принятой модели реальному процессу (объекту);

– сведения о результатах научно-исследовательских работ, если они использованы для разработки алгоритма;

– описание логики алгоритма и способа формирования результатов решения с указанием последовательности этапов счета, расчетных и логических формул, используемых в алгоритме;

• требования к разработке программы:

– диагностические сообщения при работе с программой;

– требования к контролю данных в процессе выполнения проектной операции;

– ограничения, связанные с машинной реализацией;

– требования к контрольной задаче.

"Описание информационного обеспечения ПС" информационной системы должно содержать разделы:

- состав информационного обеспечения с указанием наименования и назначения всех баз данных и наборов данных;
- организация информационного обеспечения;
- организация сбора и передачи информации;
- система классификации и кодирования информации;
- организация информационных баз данных;
- организация ведения информационной базы – последовательность процедур при создании и обслуживании базы с указанием регламента выполнения процедур и средств защиты базы от разрушения и несанкционированного доступа, а также с указанием связей между массивами баз данных и массивами входной информации.

"Описание организации информационной базы" должно содержать описание логической и физической структуры базы данных.

"Ведомость машинных носителей информации" должна содержать обозначения и наименования документов, выполненных на машинных носителях. Запись документов осуществляется в порядке возрастания присвоенных обозначений.

"Документация на разработанный функциональный программный компонент или модуль ПС" должна содержать:

- техническое задание (ТЗ) и/или спецификацию требований на разработку программы;
- описание программы в виде печатного документа;
- фрагмент руководства пользователя – описание применения программного компонента;
- исходный текст программы в виде печатного документа;
- исходный текст и объектный код программы на магнитных или на иных носителях.

"Описание технологии автоматизированного проектирования ПС" должно содержать разделы:

- общие положения;
- постановка задачи;
- методика проектирования;
- исходные данные;
- проектные процедуры;
- оценка результатов.

"Технологическая инструкция" разрабатывается на операцию или комплекс операций технологического процесса обработки данных. В документе указывают наименование технологической операции, на которую разработан документ, и приводят сведения о порядке и правилах выполнения операции технологического процесса обработки данных. В инструкции приводят перечень должностей персонала, на которые распространяется данная инструкция. Номенклатуру технологических

инструкций определяют, исходя из принятого процесса обработки данных. Структуру документа устанавливает разработчик в зависимости от содержания операций.

2.4.2. Документация испытаний комплексов программ информационных систем

Основная цель любых испытаний состоит в проверке и фиксировании реальных показателей качества программ, достигнутых в результате тестирования при разработке и отладке комплекса программ и его компонентов. Результаты тестирования и измерения этих показателей сравниваются с требованиями технического задания или спецификаций для определения степени соответствия предъявлявшимся требованиям, полученным разработчиком от заказчика. Наличие таких достаточно полных эталонов, как совокупность требований технического задания и содержание технической документации, являются важной особенностью тестирования при испытаниях. Если испытываемое ПС удовлетворяет всем требованиям технического задания, то считается что разработка завершена успешно, заказчик обязан принять ПС и закрыть контракт. Если имеются отступления от требований технического задания или документации, то возможны различные решения о продолжении или прекращении разработки.

Количество и глубина испытаний зависит от категории критичности каждого ПС. В практику вошло выделение трех уровней критичности ПС:

- критическое ПС – для него проявление дефекта проектирования или возникновение отказовой ситуации прекращают безопасное функционирование системы обработки информации и резко увеличивает риск для жизни и здоровья людей или риск аварии оборудования с большим ущербом;
- важное ПС – для которого проявление дефекта или возникновение отказовой ситуации может существенно снизить качество выпускаемой продукции и заметно увеличить риск и цену грубых ошибок при обработке информации;
- ординарное ПС – для которого ошибки или отказовые ситуации отражаются только на качестве выходных результатов, которые не могут привести к значительному ущербу в объектах внешней среды или к потерям у пользователей.

Испытания ПС могут иметь три статуса. Обычные испытания, которым подвергается широкий спектр ординарных ПС с относительно невысокими требованиями к качеству, которые предстоит эксплуатировать в не критических системах. Аттестационные испытания, которым подвергаются важные категории ПС, чьи ошибки могут нанести большой ущерб. Сертификационные испытания критических ПС, эксплуатация которых недопустима без высоких гарантий качества,

удостоверяемых уполномоченным государственным или ведомственным органом. На каждом этапе испытаний разработчики ПС и испытатели должны отчитываться перед заказчиком соответствующими документами, тестами и результатами проверок.

2.4.3. Документация сопровождения и конфигурационного управления версиями программ информационных систем

База данных программ и документов, находящихся на разных этапах разработки должна строиться как многоуровневая по степени доступности возможных изменений, хранимых описаний и версий компонентов и ПС в целом. Иерархическая структура размещения в базе данных версий программных компонентов и документов должна обеспечивать разделение их по степени доступности для использования и корректировки, по уровню детализации описания и качества отработки соответствующих компонентов, а также по полноте сопровождающей документации.

Особо должна выделяться и защищаться база данных отработанных, высококачественных версий документов повторно используемых компонентов, запись в которую следует санкционировать руководством проекта. Еще более жестко следует регламентировать и ограничивать доступ к базовым версиям ПС. Две последние базы данных подлежат тщательной защите от несанкционированного доступа и от случайного разрушения. Накопленные в них компоненты снабжаются необходимой документацией и тестами, обеспечивающими квалифицированную эксплуатацию и полную повторную проверку качества функционирования.

Средства накопления сообщений об ошибках, предложениях на изменение, выполненных корректировках и характеристиках версий являются основными при автоматизации конфигурационного управления и управления базой данных сопровождения. Изменения, отобранные группой конфигурационного управления, переводятся из журнала предварительных изменений во вторую часть базы данных. Здесь изменения конкретизируются вплоть до текстов корректировок программ или созданных новых компонентов. Все корректировки, утвержденные для введения в очередную версию, следует регистрировать в третьей части базы данных сопровождения. Для каждого изменения должны документироваться содержательная аннотация, а также общие характеристики и достигнутые на испытаниях показатели качества данной базовой версии ПС.

Учет тиражирования, адаптирования, переноса и распространения версий ПС ведется с фиксированием документов в базе данных журнала пользовательских версий. В этом журнале накапливаются сведения о платформах и параметрах среды применения и адаптации ПС у каждого пользователя, а также активность его работы с версиями программ.

Средства накопления, упорядочения, каталогизации и документирования тестовых наборов данных обеспечивают возможность многократного использования тестов в течение жизненного цикла ПС. Каждый создаваемый тест используется с определенной целью обнаружения, диагностики или локализации ошибки. Большинство тестов приходится использовать повторно, что определяет рентабельность их документирования и хранения.

3. ОЦЕНКА КАЧЕСТВА ПРОГРАММНЫХ ПРОДУКТОВ

3.1. Основные понятия и характеристики качества программных продуктов

Выбор и формирование требований к программному средству (ПС) состоит в анализе необходимых свойств, характеризующих качество его функционирования и применения с учетом технологических и ресурсных возможностей разработчиков. При этом под качеством функционирования ПС понимается множество свойств, обусловливающих его пригодность обеспечивать надежное и своевременное представление требуемой информации потребителю для ее дальнейшего использования по назначению. В соответствии с принципиальными особенностями, назначением и свойствами каждого ПС при проектировании должны выбираться номенклатура и значения характеристик качества, необходимых для эффективного применения пользователями, которые впоследствии отражаются в спецификациях требований и в технической документации на конечный продукт. Каждая характеристика качества может эффективно использоваться, если определена ее метрика, мера и шкала и может быть указан *способ ее измерения или оценивания, а также сопоставления с требующимся значением.* Они должны, прежде всего, отражать *функциональную пригодность* для применения с заданными целями.

Качество изменяется в течение жизненного цикла ПС, то есть его требуемое и реальное значение в начале жизненного цикла (ЖЦ) почти всегда отличается от фактически достигнутого при завершении проекта и качества поставляемой версии продукта. На практике важно оценивать качество программ не только в завершенном виде, но и в процессе их проектирования, разработки и сопровождения.

Требуемые характеристики качества ПС с различных позиций отражают их свойства и особенности, и в свою очередь зависят от ряда факторов и ограничений. При системном анализе и проектировании программных средств необходимо определять и учитывать связи, влияние и взаимодействие следующих основных факторов, которые отражаются на их качестве:

•назначение, содержание и описание функциональных и конструктивных характеристик, субхарактеристик и атрибутов,

определяющих специфические особенности свойств и качества конкретного программного средства;

•метрики, меры и шкалы, выбранных и пригодных для измерения и оценивания конкретных характеристик и атрибутов качества ПС;

•внешние и внутренние, негативные факторы, влияющие на достигаемое качество ПС;

•доступные ресурсы, ограничивающие возможные величины реальных характеристик качества ПС.

Множество характеристик качества программных средств можно разделить на две принципиально различающихся группы:

•*функциональные характеристики* (функциональность) – определяющие назначение, свойства и задачи, решаемые комплексом программ для основных пользователей, отличающиеся очень широким спектром и разнообразием, состав и специфику которых трудно унифицировать и можно категоризировать только по большому количеству классов и свойств ПС;

•*конструктивные характеристики* качества, номенклатура которых может быть унифицирована, адаптирована и использована для описания остальных, внутренних и внешних, стандартизируемых характеристик качества, поддерживающих реализацию основных, функциональных требований к качеству объектов и процессов ЖЦ программных средств.

Определение и сравнение *функционального качества программ* целесообразно рассматривать в пределах ограниченных классов ПС, выполняющих подобные функции. Такие классы функций могут выделяться в пределах крупных проблемно-ориентированных сфер применения (административные, банковские, медицинские, авиационные и т.п.), и для решения более мелких, специализированных, функциональных задач в этих областях.

Функциональная пригодность ПС непосредственно определяет основное его назначение и функции для пользователей. В контракте и техническом задании для каждого проекта, она должна быть выделена и формализована для однозначного понимания и оценивания всеми партнерами на каждом этапе ЖЦ и при значительных модификациях задач ПС. В силу своей специфичности, при последующем изложении функциональная пригодность обозначается как *основная цель и главная характеристика* для всего множества типов ПС.

Вторая группа характеристик – *конструктивных,* играет подчиненную роль и должна, в первую очередь, поддерживать и обеспечивать высокое качество реализации функций ПС и его применения по основному назначению. Номенклатура этих характеристик относительно не велика, и стандартами рекомендуется в составе: корректности, защищенности, надежности, ресурсной эффективности, практичности, сопровождаемости и мобильности. Их выбор и значения определяются требованиями к функциональной пригодности ПС.

3.2. Стандарты, регламентирующие характеристики качества программных средств

Основой формального регламентирования показателей качества ПС является небольшой международный стандарт "ISO 9126:1991 – Информационная технология. Оценка программного продукта. Характеристики качества и руководство по их применению". Развитие этого международного стандарта проводится в направлении уточнения, детализации и расширения описаний, характеристик качества комплексов программ. Для замены редакции 1991 года завершается разработка и формализован проект стандарта, состоящего из четырех частей ISO 9126-1-4.

Проект нового стандарта ISO 9126 состоит из следующих частей под общим заголовком: *Технический отчет – Информационная технология – Качество программных средств:*

Часть 1. Модель качества.

Часть 2. Внешние метрики качества.

Часть 3. Внутренние метрики качества.

Часть 4. Метрики качества в использовании.

Часть первая стандарта ISO 9126 определяет модель характеристик качества ПС (рис. 3.1) и ее связи с жизненным циклом комплексов программ, которая детализируется во второй и третьей частях стандарта.

Модель внутренних и внешних характеристик качества ПС состоит из шести групп базовых показателей, каждая из которых детализирована несколькими нормативными субхарактеристиками:

Функциональная пригодность детализируется:

- пригодностью для применения;
- корректностью (правильностью, точностью);
- способностью к взаимодействию;
- защищенностью.

Надежность характеризуется:

- уровнем завершенности (отсутствия ошибок);
- устойчивостью к дефектам;
- восстанавливаемостью;
- доступностью–готовностью.

Эффективность рекомендуется отражать:

- временной эффективностью;
- используемостью ресурсов.

Применимость (практичность) предлагается описывать:

- понятностью;
- простотой использования;
- изучаемостью;
- привлекательностью.

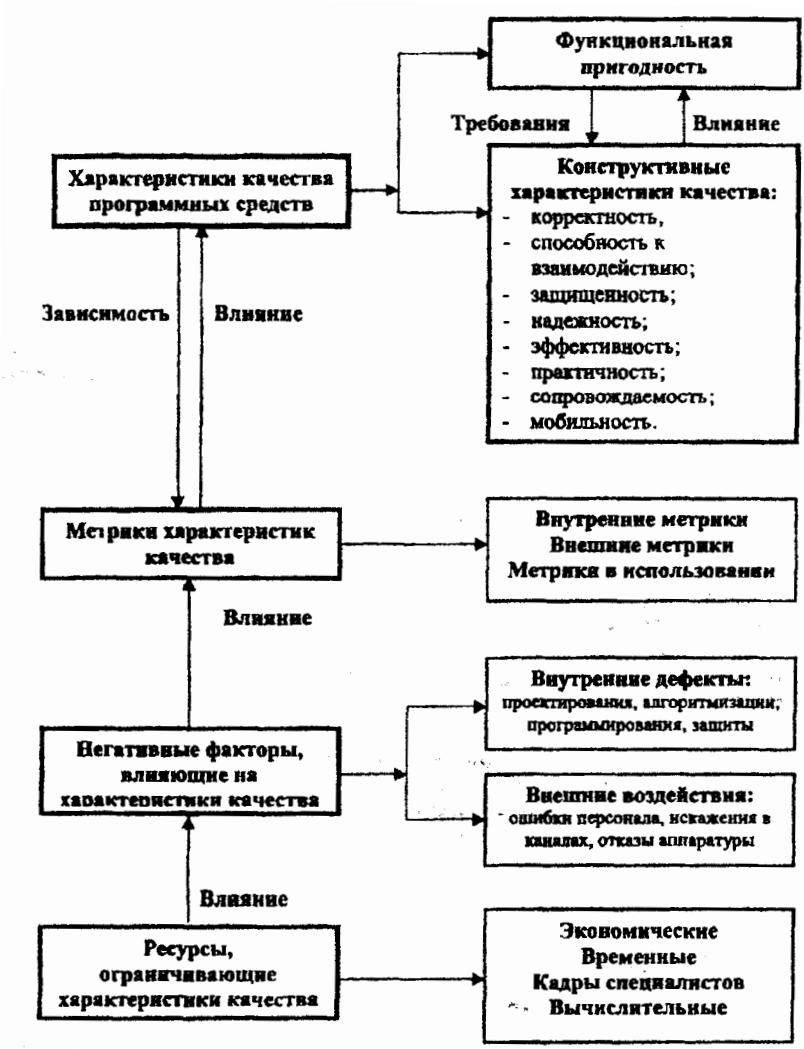

Рис. 3.1. Основные факторы, влияющие на качество программных средств

Сопровождаемость представляется:
- удобством для анализа;
- изменяемостью;
- стабильностью;
- тестируемостью.

Переносимость (мобильность) предлагается отражать:

- адаптируемостью;
- простотой установки – инсталляции;
- сосуществованием – соответствием;
- замещаемостью.

Дополнительно каждая характеристика сопровождается субхарактеристикой *согласованность*, которая должна отражать отсутствие противоречий с иными стандартами и нормативными документами, а также с другими показателями в данном стандарте.

В первой части стандарта также выделена модель характеристик качества в использовании. В этой модели используются иные базовые характеристики по сравнению с моделью внутреннего и внешнего качества. Основными характеристиками *качества ПС в использовании* являются:

- *системная эффективность* применения программного продукта по назначению;
- *продуктивность* – производительность при решении основных задач ПС, достигаемая при реально ограниченных ресурсах в конкретной вычислительной и внешней среде применения;
- *удовлетворенность требований* и затрат пользователей в соответствии с целями при применении ПС по основному назначению.
- **защищенность** – безопасность функционирования комплекса программ и возможный риск от его применения для людей, бизнеса и внешней среды.

Вторая и третья части стандарта ISO 9126 посвящены формализации соответственно внешних и внутренних метрик характеристик качества сложных программных средств. Эти части стандарта построены по одинаковой схеме и перечни их разделов практически не отличаются. Взаимосвязь метрик качества в этих частях стандарта отражена одинаковыми моделями, аналогичными базовой модели в первой части стандарта (см. рис. 3.2). Показано, что внутреннее и внешнее качества относятся непосредственно к самому программному продукту, а метрики качества в использовании проявляются в эффекте от его применения и зависят от внешней среды.

Общее представление о качестве ПС стандартом ISO 9126 рекомендуется отражать тремя взаимодействующими и взаимозависимыми *метриками характеристик качества*, отражающими:

- внутреннее качество, проявляющееся в процессе разработки и других промежуточных этапов жизненного цикла ПС;
- внешнее качество, заданное требованиями заказчика в спецификациях и отражающееся в характеристиках на конечный продукт;
- качество при использовании в процессе нормальной эксплуатации и результативность достижения потребностей пользователей с учетом затрат.

Рис. 3.2. Оценивание эффективности системы защиты ИС

Качество ПС можно измерять внутренне – статическим анализом мер программного кода или внешне – измерением поведения программного кода при его исполнении. Эти типы метрик применимы при определении целей проекта и требований к качеству ПС, включая промежуточные компоненты и продукты. Подходящие внутренние

атрибуты качества ПС являются предпосылкой достижения требуемого внешнего поведения, а приемлемое внешнее поведение – предпосылка достижения качества в использовании.

Четвертая часть стандарта ISO 9126 – метрики качества в использовании – предназначена для покупателей, поставщиков, разработчиков, сопровождающих, пользователей и менеджеров качества ПС. Здесь рассмотрена модель качества в использовании.

В отличие от характеристик, описанных в предыдущих частях стандарта, в этой части для оценивания качества в использовании рекомендуется четыре: эффективность; продуктивность; удовлетворение требований и защищенность. В данном разделе приводятся подробные рекомендации для описания и оценивания выделенных для проекта ПС характеристик качества в использовании: специфика определения целей и контекста среды пользователя; выбор, селекция и интерпретация каждой из выделенных метрик; выделение и утверждение критериев для реализации и оценивания качества в использовании; интерпретация результатов измерений и оценивания.

3.3. Особенности измерения и оценивания характеристик качества программных средств

Для выбора характеристик качества ПС и достоверного сравнения их с требованиями, а также для сопоставления их значений между различными программными продуктами необходимы измерения и использование определенных мер и шкал. Стандартами рекомендуется, чтобы было предусмотрено измерение каждой характеристики качества ПС (субхарактеристики или ее атрибута) с точностью и определенностью, достаточной для выполнения сравнений с требованиями, и чтобы эта точность обеспечивалась при измерении. Следует предусматривать нормы допустимых ошибок измерения, вызванных инструментами и ошибками человека – эксперта. Меры, используемые для сравнений, должны быть утверждены и иметь точность, достаточную для выполнения надежных сравнений. Для этого требуется, чтобы измерения были объективны и воспроизводимы.

Чтобы измерения были объективными, должна быть документирована и согласована процедура для присвоения числового значения, свойства или категории каждому атрибуту программного продукта. При эмпирических измерениях для получения данных должны использоваться наблюдения или одобренные вопросники с применением номинальной, интервальной или порядковой шкалы. Процедуры измерений должны давать в результате одинаковые меры с приемлемой устойчивостью, получаемые различными субъектами при выполнении одних и тех же измерений характеристик ПС в различных случаях.

Характеристики, субхарактеристики и атрибуты качества ПС с позиции возможности и точности их измерения можно разделить на *три группы показателей*, особенности которых следует уточнять при их выборе:

- категорийные – описательные, отражающие набор свойств и общие характеристики объекта – его функции, категории ответственности, защищенности и важности, которые могут быть представлены номинальной шкалой категорий-свойств;
- количественные – представляемые множеством упорядоченных числовых точек, отражающих непрерывные закономерности и описываемые интервальной или относительной шкалой, которые можно объективно измерить и численно сопоставить с требованиями;
- качественные – содержащие несколько упорядоченных или отдельных свойств – категорий, которые характеризуются порядковой или точечной шкалой набора категорий (есть – нет, хорошо – плохо), устанавливаются, выбираются и оцениваются в значительной степени субъективно и экспертно.

К *первой группе* относятся показатели качества, которые характеризуются наибольшим разнообразием значений – свойств программ и наборов данных и охватывают весь спектр классов, назначений и функций современных ПС. Эти свойства можно сравнивать только в пределах однотипных ПС и трудно упорядочивать по принципу предпочтительности. Среди стандартизированных показателей качества к этой группе, прежде всего, относится *функциональная пригодность*, являющаяся самой важной и доминирующей характеристикой любых ПС. Номенклатура и значения всех остальных показателей качества непосредственно определяются требуемыми функциями программного средства и, в той или иной степени, влияют на выполнение этих функций.

Ко *второй группе* стандартизированных показателей качества относятся достаточно достоверно и объективно измеряемые численные характеристики ПС. Значения этих характеристик обычно в наибольшей степени влияют на функциональную пригодность и метрики в использовании ПС. Их субхарактеристики могут быть описаны упорядоченными шкалами объективно измеряемых значений, требуемые численные величины которых могут быть установлены и выбраны заказчиками или пользователями ПС. Такими характеристиками являются *надежность* и *эффективность* комплексов программ. Надежность может отражаться временем наработки на отказ, средним временем восстановления, а также коэффициентом готовности – вероятностью застать ПС в работоспособном состоянии при нормальной эксплуатации.

Третью группу стандартизированных показателей качества ПС трудно полностью описать измеряемыми количественными значениями и их некоторые субхарактеристики имеют описательный, качественный вид. В зависимости от функционального назначения ПС по согласованию с заказчиком можно определять экспертно степень необходимости этих свойств и бальные значения уровня реализации их атрибутов в

жизненном цикле конкретного ПС. Например, не всегда может требоваться мобильность программ на иные операционные и аппаратные платформы, а также выбор и оценка соответствующих субхарактеристик, которые можно полностью исключать из метрик качества в использовании. В других случаях мобильность можно оценивать категориями: отличная, хорошая, удовлетворительная или неудовлетворительная. Такие оценки могут проводиться экспертно на основе анализа возможной трудоемкости и длительности, реализации процессов переноса комплекса программ на новую платформу.

Сопровождаемость может иметь ограниченный характер редкой полной замены программ на вновь разработанные версии и тем самым сливаться с процессами разработки или осуществляться как непрерывная поддержка множества пользователей консультациями, адаптациями и корректировками программ. Соответственно качественно могут быть установлены субхарактеристики сопровождаемости и описаны требуемые их свойства.

Практичность тесно связана с функциональной пригодностью. Обобщенно этот показатель можно отразить трудоемкостью и длительностью, которые необходимы для изучения и полного освоения функций и технологии применения соответствующего ПС. Каждая из субхарактеристик практичности имеет ряд качественных атрибутов, которые могут выбираться и оцениваться экспертно с учетом функционального назначения ПС, а также надежности и ресурсной эффективности комплекса программ.

3.4. Оценивание характеристик качества программных средств

3.4.1. Оценивание функциональных возможностей программных средств

Оценивание функциональной пригодности программных средств. Системный анализ и прослеживание взаимного соответствия базовых и конструктивных субхарактеристик является основой для оценивания функциональной пригодности при любых специфических функциях ПС. Меры этого соответствия иногда могут быть количественными, и почти всегда, остаются качественными описаниями различных свойств и возможных их дефектов.

Хотя исходные атрибуты качества функциональной пригодности конкретных проектов ПС, в той или иной степени, субъективны, их формализованные описания и согласование разработчиком и заказчиком способны предотвратить многие конфликты при оценивании комплексов программ соответствия требованиям. Для обеспечения достоверности оценивания функциональной пригодности особое

значение имеет формализация на основе стандартов программы, методик испытаний ПС и обработки результатов.

Основная задача оценивания корректности программных средств – формально и максимально достоверно установить степень соответствия комплекса реализованных программ исходным требованиям контракта, технического задания и спецификаций на ПС и его компоненты. Корректность ПС в информационных системах является важнейшей характеристикой для эффективного их применения и определения функциональной пригодности для пользователей.

В процессе разработки и на других этапах жизненного цикла программ невозможно полностью избежать дефектов и ошибок, которые негативно, а иногда катастрофически, отражаются на корректности ПС. Поэтому в стандартах, регламентирующих жизненный цикл ПС, значительное внимание уделяется процессам упорядоченного, иерархического анализа корректности, исходящего от требований к информационной системе и к ПС. Этот процесс включает анализ, просмотр (обзор) и тестирование корректности выполнения требований. Он проводится сверху вниз, начиная от общих требований в техническом задании и/или спецификации на всю информационную систему до детальных требований на отдельные модули программ и их взаимодействие. Тестирование снизу вверх должно обеспечивать проверку, степени корректности реализации всей совокупности требований.

Цели оценивания и обеспечения корректности ПС достигаются посредством последовательного выполнения комбинаций из просмотров, анализов, разработки тестовых сценариев и процедур, и последующего выполнения этих процедур. Просмотр может включать инспекцию и оценивание качества выходных результатов указанных процессов. Анализ должен детально исследовать функциональность, эффективность и прослеживаемость взаимодействия компонентов ПС, а также их связи с другими компонентами системы и с оборудованием ИС.

Просмотры и анализы требований высокого уровня должны обнаружить, зарегистрировать и устранить дефекты, которые внесены в процессе разработки и преобразования требований к ПС.

Просмотры и анализы требований низкого уровня предназначены для выявления дефектов и ошибок детализированных требований, которые могут быть внесены в процессе проектирования компонентов ПС.

Просмотры и анализы исходного текста программ должны подтверждать, что выходные результаты программирования являются точными и полными. Анализ обычно ограничивается исходным текстом программ, при котором проверяется, что он является корректным, полным и соответствует требованиям к компонентам низкого уровня, а также то, что исходный текст не содержит не описанных функций. Должно

быть проверено, что процесс разработки программ осуществлялся полностью в соответствии со стандартами на программирование и отклонения от этих стандартов обоснованы, особенно, в случаях ограничения сложности, использования типовых конструкций программ, вложенности управляющих структур, сложности логических или числовых выражений. Следует проверить и оценить корректность и непротиворечивость исходного текста, включая реализацию стеков, переполнение и разрешающую способность для арифметики с фиксированной точкой, конкуренцию ресурсов, обработку особых ситуаций, использование неинициализированных переменных или констант, а также возможность нарушения целостности данных из-за конфликтов прерываний.

Тестирование программ, основанное на требованиях в процессе всего жизненного цикла должно охватывать функционирование ПС во всей доступной области варьирования исходных данных и режимов применения. Тестирование ПС имеет две взаимодополняющие цели. Первая цель – показать, что ПС и его компоненты полностью и корректно удовлетворяют заданным требованиям к ним. Должно быть рассмотрено и подтверждено, что тестовые варианты правильно представлены в процедурах тестирования и ожидаемых результатах; гарантировать, что результаты тестирования являются корректными и что объяснимы несоответствия между фактическими и ожидаемыми результатами. Вторая цель – показать с высокой степенью доверия, что устранены дефекты и ошибки, которые могли бы привести к возникновению недопустимых отказных ситуаций, влияющих на корректность функционирования ПС. При этом номенклатура и диапазоны варьирования тестов ограничены либо допустимой длительностью подготовки и исполнения тестов, либо вычислительными ресурсами, доступными для использования с целью контроля и тестирования в режиме нормальной эксплуатации.

Оценивание защищенности программных средств состоит из определения полноты и эффективности использования доступных методов, средств и ресурсов для защиты ПС от различных, потенциальных угроз и достигнутой при этом безопасности функционирования информационной системы. Защищенность во многих информационных системах является ключевой характеристикой для эффективного их применения и определения функциональной пригодности пользователями. Из-за ограниченности ресурсов и других факторов априори невозможно обеспечить отсутствие дефектов проектирования и абсолютную защиту сложных ПС от негативных воздействий, вследствие чего безопасность их функционирования в ИС имеет всегда конечное, ограниченное значение.

Процессы проектирования и оценивания программ обеспечения безопасности ИС принципиально не отличается от проектирования любых других программных комплексов. Для этого, прежде всего,

1	2	3
16	**Оценивание мобильности ПС**	состоит в определении атрибутов субхарактеристик качества переноса программ и данных на иные аппаратные и операционные платформы
17	**Доступ к информации**	получение возможности использовать информацию, хранящуюся в ЭВМ (системе)
18	**Защита информации**	совокупность методов, позволяющих управлять доступом выполняемых в системе программ к хранящейся в ней информации
19	**Шифрование информации**	обработка информации путем замены и перемешивания букв, при котором объем данных не меняется
20	**Пакет ошибок длиной «n»**	любой ряд ошибок, для которого число знаков между первой и последней ошибками, включая эти ошибки, равно «n»
21	**Кодирование информации**	преобразование информации, в результате которого обеспечивается изменение объема памяти, занимаемой данными

ГЛОССАРИЙ

№ п/п	Новые понятия	Содержание
1	2	3
1	**Система программирования**	комплекс программных средств, предназначенных для кодирования, тестирования и отладки программного обеспечения
2	**Текстовый редактор в системе программирования**	программа, позволяющая создавать, изменять и обрабатывать исходные тексты программ на языках высокого уровня
3	**Компоновщик**	предназначен для связывания между собой объектных файлов, порождаемых компилятором, а также файлов библиотек, входящих в состав системы программирования
4	**Отладчик**	программный модуль, который позволяет выполнить основные задачи, связанные с мониторингом процесса выполнения результирующей прикладной программы
5	**Тестирование**	основной метод отладки, измерения качества и определения реальных характеристик программ и информации баз данных на любых этапах их жизненного цикла
6	**Технологическая документация процесса разработки**	включает подробные технические описания и подготавливается для специалистов, ведущих проектирование, разработку и сопровождение ПС, обеспечивает возможность отчуждения, детального освоения, развития и корректировки ими программ и данных на всем жизненном цикле ПС
7	**Эксплуатационная документация продукта и результатов разработки**	создается для конечных пользователей ПС и позволяет им осваивать и квалифицированно применять эти средства для решения конкретных функциональных задач ИС
8	**Документация администрирования при эксплуатации информационной системы**	должна обеспечивать поддержку первичной инсталляции, функционирования и восстановления программ и данных после сбоев
9	**Защищенность программ**	безопасность функционирования комплекса программ и возможный риск от его применения для людей, бизнеса и внешней среды
10	**Основная задача оценивания корректности программных средств**	формально и максимально достоверно установить степень соответствия комплекса реализованных программ исходным требованиям контракта, технического задания и спецификаций на ПС и его компоненты
11	**Оценивание защищенности программных средств**	состоит из определения полноты и эффективности использования доступных методов, средств и ресурсов для защиты ПС от различных, потенциальных угроз и достигнутой при этом безопасности функционирования информационной системы
12	**Оценнвание надежности ПС**	включает измерение количественных субхарактеристик в использовании: завершенности, устойчивости к дефектам, восстанавливаемости и доступности–готовности
13	**Оценивание ресурсной эффективности**	состоит в измерении количественных субхарактеристик и их атрибутов: временной эффективности (метрик поведения ПС во времени) и используемости ресурсов ЭВМ комплексом программ
14	**Оценивание практичности – применимости ПС**	включает определение атрибутов удобства и комфортности для пользователей при его освоении, подготовке комплекса программ к эксплуатации и при использовании по назначению
15	**Оценивание сопровождаемости**	заключается в определении мер и атрибутов процессов сопровождения и конфигурационного управления изменениями и версиями в жизненном цикле комплексов программ

необходимо анализировать, конкретизировать и оценивать задачи, а также исходные данные и факторы, определяющие безопасность функционирования программ:

• критерии и их значения, характеризующие необходимый и достаточный уровень безопасности ИС в целом и каждого из ее основных, функциональных компонентов, в соответствии с условиями среды применения ИС и требованиями спецификаций заказчика;

• состав и характеристики возможных внутренних и внешних дестабилизирующих факторов и угроз, способных влиять на безопасность функционирования программных средств и баз данных проектируемой ИС;

• состав и содержание подлежащих решению задач защиты, перекрывающих все потенциально возможные угрозы и оценки эффективности решения отдельных задач, необходимых для обеспечения равнопрочной защиты ИС с заданной эффективностью;

• оперативные методы и средства повышения безопасности функционирования программ в течение всего жизненного цикла ИС путем введения временной, программной и информационной избыточности для реализации системы защиты от актуальных видов угроз;

• ресурсы, необходимые и доступные для разработки и размещения программных компонентов системы обеспечения безопасности ИС (финансово-экономические, ограниченная квалификация специалистов и вычислительные ресурсы ЭВМ);

• стандарты, нормативные документы и методики воспроизводимых измерений и оценивания характеристик защиты, а также состав и значения исходных и результирующих данных, обязательных для проведения испытаний защиты.

Оценивание достигнутой эффективности реализованной системы защиты ИС начинается с анализа и регистрации: возможных угроз безопасности функционирования ПС; требуемых критериев и характеристик качества защиты; доступных и использованных ресурсов на реализацию защиты, а также характеристик и особенностей объектов, подлежащих защите (рис 3.2).

Далее должны быть оценены выбранные методы и инструментальные средства для реализации защиты, а также использованные способы их применения и интеграции для обеспечения равной прочности защиты информационных ресурсов и программ при различных угрозах. Реализацию такой защиты следует сопоставить с требованиями заказчика и обязательствами ее поставщика, а также оценить степень поддержки руководствами для пользователей и специалистов-администраторов по полному и корректному применению всего комплекса методов и средств защиты. Проведенные оценки и документы обобщаются в необходимых уточнениях и корректировках концепции построения и организации системы обеспечения

безопасности функционирования и применения ИС. Завершающий анализ и оценивание реализации концепции и всего комплекса методов, средств и их взаимодействия в системе защиты ИС приводит к возможности определения достигнутых характеристик и атрибутов качества комплексной защищенности ИС, а также к их сопоставлению с требованиями заказчика проекта. Для эффективного применения и совершенствования системы защиты необходимо также оценивать качество средств обнаружения и регистрации предумышленных нападений на ИС, проявлений и реализации угроз безопасности, а также мониторинга инцидентов, угрожавших нарушению и/или преодолению системы защиты.

3.4.2. Оценивание надежности функционирования программных средств

Оценивание надежности ПС включает измерение количественных субхарактеристик в использовании: завершенности, устойчивости к дефектам, восстанавливаемости и доступности–готовности. При этом предполагается, что в контракте, техническом задании или спецификации требований зафиксированы, и утверждены определенные значения атрибутов стандартизированных субхарактеристик. Прямые и непосредственные измерения этих показателей проводятся при функционировании готового программного продукта для сопоставления с заданными требованиями и для оценивания степени соответствия этим требованиям. Значения надежности коррелированы с субхарактеристикой корректность, однако можно достигать высокую надежность функционирования программ при относительно невысокой их корректности за счет сокращения времени восстановления при отказах.

Кроме того, надежность ПС можно оценивать косвенно в процессе разработки по прогнозируемой плотности обнаружения скрытых дефектов и ошибок, а также по плотности выявляемых и устраняемых ошибок выходных результатов и взаимодействия с операционной системой в динамике тестирования рабочего функционирования комплекса программ. Распределение реальных длительностей и эффективности восстановления при ограниченных ресурсах для функционирования программ может рассматриваться как дополнительная полезная составляющая при оценивании надежности.

Для прямых, количественных измерений атрибутов надежности необходимы инструментальные средства, встроенные в операционную систему или в соответствующие компоненты ПС. Эти средства должны в динамике реального функционирования ПС автоматически селектировать и регистрировать аномальные ситуации, дефекты и искажения вычислительного процесса, программ и данных, выявляемые аппаратным, программно-алгоритмическим контролем или

пользователями. Накопление и систематизация проявлений дефектов при исполнении программ позволяет оценивать основные показатели надежности, помогает определять причины сбоев и отказов и подготавливать данные для улучшения надежности ПС. Регулярная регистрация и обобщение таких данных позволяет устранять ситуации, негативно влияющие на функциональную пригодность и другие важные характеристики ПС.

Прямые экспериментальные методы оценивания интегральных характеристик надежности ПС – завершенности, устойчивости и готовности, в ряде случаев весьма трудно реализовать при нормальных условиях функционирования сложных комплексов программ, из-за больших значений требуемого времени наработки на отказ (сотни и тысячи часов), которые необходимо достигать при разработке и фиксировать при испытаниях. Сложность выявления и регистрации редких отказов, а также высокая стоимость экспериментов при длительном многосуточном функционировании сложных ПС приводят к тому, что на испытаниях получаются малые выборки зарегистрированных отказов и низка достоверность оценки показателей надежности. Кроме того, при таких экспериментах трудно гарантировать полную представительность выборки исходных данных, так как проверки определяются конкретными условиями применения данного ПС на испытаниях.

При испытаниях надежности ПС в первую очередь обнаруживаются отказы – потери работоспособности. Однако в большинстве случаев первоначально остается неизвестной причина происшедшего отказа. Для выявления фактора, вызвавшего отказ и устранения его причины, необходимо, прежде всего, определить, каким компонентом информационной системы стимулирован данный отказ. Наиболее крупными источниками отказов являются частичные физические неисправности или сбои аппаратуры ЭВМ, а также дефекты и ошибки программных средств. Стабильные неисправности аппаратуры диагностируются достаточно просто, соответствующими аппаратными тестами, после чего должен следовать ремонт или замена определенных блоков. Однако при возникновении случайного отказа, после которого происходит автоматически полное восстановление нормального функционирования, во многих случаях трудно или невозможно однозначно выявить его первичный источник, особенно при очень редких отказах.

Если интенсивное тестирование программ в течение достаточно длительного времени не приводит к обнаружению дефектов или ошибок, то у специалистов, ведущих испытания, создается ощущение бесполезности дальнейшего тестирования данной программы, и она передается на эксплуатацию. Экспериментальное исследование характеристик сложных ПС позволило оценить темп обнаружения дефектов, при котором сложные комплексы программ передаются на

регулярную эксплуатацию: 0,002–0,005 ошибок в день на человека, т.е. специалисты по испытаниям или пользователи в совокупности выявляют только около одной ошибки или дефекта каждые два – три месяца использования ПС. Интенсивность обнаружения ошибок ниже 0,001 ошибок в день на человека, т.е. меньше одной ошибки в год на трех–четырех специалистов, непосредственно выполняющих тестирование и эксплуатацию ПС, по-видимому, может служить эталоном высокой надежности для ПС обработки информации. Если функционирование программ происходит непрерывно, то эти показатели соответствуют очень высокой наработке на обнаружение дефекта или отказа порядка 5–10 тысяч часов и коэффициенту готовности выше 0,99. При использовании этого критерия обычно учитывается календарное время испытаний, включающее длительность непосредственного тестирования, как для обнаружения, так и для локализации дефектов, а также длительность корректировки программ и других вспомогательных работ для восстановления нормального функционирования ПС.

Форсированные испытания для оценивания надежности программных средств значительно отличаются от традиционных методов испытаний аппаратуры. Основными факторами, влияющими на надежность ПС, являются исходные данные и их взаимодействие с дефектами и ошибками программ и данных или сбоями в аппаратуре ЭВМ. Поэтому форсирование испытаний осуществляется повышением интенсивности искажений исходных данных и расширением варьирования их значений, а также специальным увеличением интенсивности потоков информации и загрузки ПС выше нормальной. На надежности функционирования сложных, критических ПС весьма существенно отражаются перегрузки по памяти и производительности при исполнении программ. Эти виды угроз стимулируют отказы, которые трудно выявляются при испытаниях в режимах нормальной загрузки и требуют особой организации тестирования.

Особым видом форсированных испытаний является целенаправленное тестирование эффективности средств оперативного контроля и восстановления программ, данных и вычислительного процесса для оценивания субхарактеристики восстанавливаемости. Для этого имитируются запланированные условия функционирования программ, при которых в наибольшей степени стимулируется срабатывание средств программного рестарта и оперативного, автоматического восстановления работоспособности. При таких испытаниях основная задача состоит в оценивании качества функционирования средств автоматического повышения надежности и измерение характеристик восстанавливаемости.

3.4.3. Оценивание эффективности использования ресурсов ЭВМ программным средством

Оценивание ресурсной эффективности состоит в измерении количественных субхарактеристик и их атрибутов: временной эффективности (метрик поведения ПС во времени) и используемости ресурсов ЭВМ комплексом программ.

Для измерения атрибутов временной эффективности необходимы инструментальные средства, встроенные в операционную систему или в соответствующее прикладное ПС. Эти средства должны в динамике реального функционирования программ регистрировать: загрузку вычислительной системы; значения интенсивности потоков данных от внешних абонентов; длительность исполнения заданий; характеристики функционирования устройств ввода/вывода; время ожидания результатов (отклика) на задания пользователей; заполнение памяти обмена с внешними абонентами в различных режимах применения комплекса программ и т.п. Значения этих характеристик зависят не только от свойств и функций ПС, но также от особенностей архитектуры и операционной системы ЭВМ, которые целесообразно выделять при анализе. Регулярная регистрация и обобщение таких данных позволяет выявлять ситуации, негативно влияющие на функциональную пригодность, надежность и другие важные характеристики ПС.

При использовании комплексом программ производительности и памяти реализующей ЭВМ менее чем на 50%, разработчик может практически не учитывать эти ограничения. Поэтому закономерным является стремление разработчиков программ применять особенно для систем реального времени объектные ЭВМ с предельным использованием технических характеристик. Опыт создания ПС реального времени позволяет утверждать, что практически невозможно использовать производительность объектной ЭВМ более чем на 95% и почти всегда целесообразно ограничиваться на уровне 90%, так как иначе, затраты на разработку ПС могут значительно увеличиться.

Для оценивания использования ресурсов производительности должны быть измерены:

- реальные значения интенсивностей поступающих исходных данных и заданий на вызов функциональных программ, а также распределения вероятностей этих интенсивностей для различных источников и типов сообщений;
- длительности автономного решения отдельно каждой из функциональных задач, обрабатывающей исходные данные или включаемой внешними заданиями;
- загрузка ЭВМ в нормальном режиме поступления сообщений и заданий, а также вероятность перегрузки заданиями различных типов и распределения длительностей перегрузки в реальных условиях;

• влияние пропуска в обработке заданий или сообщений каждого типа и снижения темпа решения определенных задач на функциональную пригодность и другие характеристики качества ПС.

Перечисленные задачи могут быть решены экспериментально в процессе тестирования завершенной разработкой ИС, однако при этом велик риск, что производительность ЭВМ окажется недостаточной для решения заданной совокупности задач в реальном времени, что отразится на качестве использования ПС. Кроме того, не всегда условия испытаний или опытной эксплуатации системы соответствуют режимам массового ее применения. Поэтому при оценивании требуется принимать специальные меры для создания реальных, а также контролируемых, наиболее тяжелых по загрузке условий функционирования ПС. Такие критические ситуации могут быть в значительной степени предотвращены в процессе разработки ПС путем использования результатов последовательного расчета длительностей исполнения модулей по тексту программ и объединения этих характеристик в соответствии со структурой программных компонентов и всего комплекса программ.

3.4.4. Оценивание практичности программных средств

Оценивание практичности – применимости ПС включает определение атрибутов удобства и комфортности для пользователей при его освоении, подготовке комплекса программ к эксплуатации и при использовании по назначению. Субхарактеристики практичности содержат понятность, простоту использования, изучаемость и привлекательность программных средств.

Оценки практичности зависят не только от собственных характеристик ПС, но также от организации и адекватности документирования процессов их эксплуатации. При этом предполагается, что в контракте, техническом задании или спецификации зафиксированы и утверждены требования к основным параметрам и качеству организационных методов и средств поддержки использования ПС. Эти требования могут влиять на функциональную пригодность и успех внедрения комплекса программ у пользователей, а также значительно различаться в зависимости от функционального назначения и сферы применения ПС. Для оценивания реализованных атрибутов качества практичности в сложных, массовых ПС необходима организация заказчиком группы экспертов, которая в процессе системного проектирования должна подготовить квалифицированные требования к этим атрибутам. При завершении разработки и испытаниях ПС эти эксперты должны оценить степень реализации требований, посредством соответствующего анализа документации и практического измерения характеристик процессов освоения комплекса программ, пользователями.

Атрибуты субхарактеристик понятность и привлекательность близки по содержанию и субъективному восприятию оценщиками-испытателями. Их можно оценивать только качественно, экспертно категориями хорошая – плохая или, более детально, еще одной – двумя категориями, включающими дополнительно меры: отличная и удовлетворительная. Дальнейшее увеличение числа градаций при оценивании таких атрибутов как демонстрационные возможности, четкость концепции и наглядность документации вряд ли целесообразно. Для обеспечения объективности оценивание целесообразно проводить группой экспертов-пользователей при типовом и экстремальном применении ПС.

Субхарактеристики простота использования и изучаемость также могут обобщенно оцениваться качественно теми же шкалами с двумя–четырьмя категориями. Такой же метод наиболее адекватен для оценивания комфортности эксплуатации и простоты управления функциями ПС. Однако некоторые атрибуты этих субхарактеристик доступны для более полной количественной оценки путем измерения трудоемкости и длительности соответствующих процессов подготовки и обучения пользователей к эффективной эксплуатации ПС. Наиболее удобно этими мерами определять качество изучаемости ПС. Этот показатель зависит как от внутренних свойств и сложности комплекса программ, так и от метрик качества в использовании, а также от субъективных характеристик квалификации конкретных пользователей. На значения изучаемости существенно влияют демонстрационные возможности справочных средств обучения, качество и объем эксплуатационной документации, а также электронных учебников, которые можно оценивать соответственно по числу сопровождающих страниц документов или занятых учебниками килобайтов памяти.

3.4.5. Оценивание сопровождаемости программных средств

Оценивание сопровождаемости заключается в определении мер и атрибутов процессов сопровождения и конфигурационного управления изменениями и версиями в жизненном цикле комплексов программ. Субхарактеристики сопровождаемости включают: анализируемость, изменяемость, стабильность и тестируемость программ, которые можно оценивать качественно и субъективно экспертами с использованием бальной шкалы. Подобный способ оценивания сопровождаемости с большим числом (около 15) качественных атрибутов рекомендуется в стандарте ISO 9126. Оценки этого показателя качества зависят не только от собственных характеристик ПС, но также от организации и документирования реализации процессов сопровождения. Обобщенно, эту характеристику и совокупность ее атрибутов можно оценивать количественно величиной затрат: стоимостью, трудоемкостью и длительностью реализации процедур сопровождения и модификации определенных программ и данных. При

этом предполагается, что в контракте, техническом задании или спецификации зафиксированы и утверждены требования к основным параметрам и качеству реализации процессов сопровождения ПС.

Оценивание полноты и эффективности реализации этих функций и их конкретизации в атрибутах сопровождаемости определяют качество ПС для специалистов, которые модифицируют и совершенствуют функции комплексов программ, а также их эксплуатируют. Для обеспечения возможности такого оценивания в процессе проектирования ПС должна быть формализована и документально зафиксирована концепция организации процесса сопровождения, параметры которого являются исходными для определения его качества:

- ожидаемую длительность поддержки сопровождением развития проекта ПС;
- объем, сложность и уровень предполагаемых изменений;
- возможное число и периодичность выпуска версий со значительными модификациями;
- организационные основы процессов сопровождения и конфигурационного управления;
- требования к документированию изменений и версий;
- кто будет осуществлять сопровождение – покупатель, разработчик первичной версии или специальный персонал поддержки развития и адаптации версий ПС.

Сопровождаемость следует также оценивать полнотой и достоверностью документации о состояниях ПС и его компонентов, всех предполагаемых и выполненных изменениях, позволяющей установить текущее состояние версий программ в любой момент времени и историю их развития. Она должна определять стратегию, стандарты, процедуры, распределение ресурсов и планы создания, изменения и применения документов на программы и данные информационных систем. Структура документации и формы отдельных документов, используемых для сопровождения программ, должны позволять:

- точно документально описывать и идентифицировать каждую оформленную версию программных компонентов и ПС в целом в любое время на всем протяжении их жизненного цикла;
- надежно учитывать и регистрировать все анализируемые, подготавливаемые и реализованные изменения в версиях ПС;
- снабжать руководителей проекта обобщенной и детальной информацией для принятия решений на изменения программ, а также для контроля выполнения принятых решений;
- обеспечивать заказчиков и пользователей сведениями о наиболее существенных корректировках программ и о новых версиях ПС.

По истечении некоторого времени, иногда через пять-десять лет, сопровождение конкретного ПС прекращается. Это может быть обусловлено разработкой более современных ПС или нерентабельным

возрастанием затрат на сопровождение. Часто сопровождение прекращается постепенно, в процессе последовательного снижения полноты и качества его поддержки. Для того чтобы со временем прийти к обоснованному решению о прекращении сопровождения необходимо периодически оценивать эффективность эксплуатации версий ПС у пользователей и возможный экономический ущерб у них от отмены сопровождения конкретной версии.

3.4.6. Оценивание мобильности программных средств

Оценивание мобильности ПС состоит в определении атрибутов субхарактеристик качества переноса программ и данных на иные аппаратные и операционные платформы. Эти субхарактеристики включают адаптируемость, простоту установки, сосуществование и замещаемость программ, которые можно оценивать экспертами качественно и субъективно с использованием бальной шкалы. Обобщенно, эту характеристику ПС и совокупность ее атрибутов можно и целесообразно оценивать экономически – количественно: совокупными затратами, стоимостью, трудоемкостью и длительностью на реализацию процедур переноса определенной совокупности программ и данных. Оценки мобильности зависят не только от внутренних субхарактеристик ПС, но также от организации, технологии и документирования реализации жизненного цикла и процессов переноса.

В зависимости от степени программной совместимости между исходной и новой платформами возможны варианты оценивания исходных условий мобильности:

- при полной несовместимости платформ, когда может потребоваться переписывание всего ПС или компонентов заново (возможно, с использованием имеющихся спецификаций требований);
- при несовместимости языков программирования или диалектов одного языка может потребоваться переписывание программ на том языке, который принят для проекта нового ПС (возможно, с использованием имеющихся проектных спецификаций повторно используемых компонентов);
- при несовместимости аппаратно-программных платформ, поддерживающих один и тот же язык программирования, требуется перекомпиляция программ на новой платформе (возможно, с автоматической оптимизацией, обеспечиваемой применяемой системой программирования).

Задачи оценивания мобильности программ и данных охватывают: встраивание готового программного комплекса в создаваемую ИС при условии, что поставщики этого ПС гарантируют его функционирование на выбранной платформе, а также перенос программ и данных с платформ, в среде которых они были ранее реализованы, на выбранную

новую платформу. Задачи и объекты, связанные с мобильностью ПС в информационных системах и подлежащие рассмотрению при оценивании адаптируемости и замещаемости, включают:

- унифицированные протоколы и интерфейсы взаимодействия функциональных программных компонентов между собой, со средой ИС, с пользователями, с внешней средой, включающие интерфейсы операционных систем, сетевые протоколы, спецификации служб организации процессов;
- языки программирования и инструментальные средства, поддерживающие создание переносимых ПС и средства программной инженерии;
- языки баз данных и системы управления базами данных;
- форматы переносимых электронных документов.

Значительная часть этих задач решается путем применения соответствующих стандартов информационных технологий, действующих как международные или национальные нормативные документы, или открытых спецификаций, отражающих сложившиеся промышленные стандарты "де-факто".

При переносе свойства ИС практически всегда изменяются, что следует учитывать при оценивании целесообразности и эффективности переноса, а также могут быть необходимы тестирование, испытания и сертификация получаемых ПС в новой среде.

3.5. Пример выбора и формирования требований к характеристикам качества программного средства

Разнообразие функций и потребителей характеристик качества программных средств делает невозможной унификацию всей совокупности требований к их составу и значениям для всех видов ПС. Поэтому целесообразно ограничиться анализом выбора и формирования требований на гипотетическом примере ПС для некоторого типа информационных систем. Приведенные ниже примеры обоснования и выбора требований к характеристикам качества ПС могут служить ориентирами для анализа и синтеза аналогичных данных в реальных проектах.

В качестве примера ниже за основу принята сложная административная ИС с определенными функциями и ограниченными условиями применения, с десятками действующих операторов-пользователей, работающих в реальном времени по схеме клиент-сервер. К такой системе предъявляются достаточно конкретные и высокие требования к разнообразию и качеству решения функциональных задач при относительно больших экономических и вычислительных ресурсах. Эти требования ниже формируются с позиции заказчиков и непосредственных пользователей ПС. Аналогами такой системы можно рассматривать банковские, налоговые, коммунальные,

таможенные и другие административные системы, имеющие наборы разнообразных функциональных задач и интенсивные потоки исходной и результирующей информации в реальном времени. Регламент реального времени в них более легкий, чем в системах управления быстрыми динамическими объектами, например, самолетами.

Характеристики качества ПС различаются по степени влияния на системную эффективность их применения по прямому назначению. Вследствие реальной ограниченности ресурсов, которые доступны в жизненном цикле ПС, их необходимо выделять с учетом влияния на эту эффективность. Для этого целесообразно присваивать и учитывать уровень приоритета требований к каждой субхарактеристике. В таблице 3.1 представлен пример возможного распределения приоритетов требований заказчика к субхарактеристикам ПС для принятого варианта административной ИС.

Таблица 3.1

Пример распределения приоритетов требований к характеристикам качества программного средства

Характеристика	Субхарактеристика	Приоритет требований
Функциональность	Функциональная пригодность	Высокий
	Корректность – правильность	Высокий
	Способность к взаимодействию	Средний
	Защищенность	Высокий
Надежность	Завершенность	Низкий
	Устойчивость к дефектам	Средний
	Восстанавливаемость	Высокий
	Доступность – готовность	Высокий
Эффективность	Временная эффективность	Высокий
	Используемость ресурсов	Средний
Практичность	Понятность	Средний
	Простота использования	Низкий
	Изучаемость	Средний
	Привлекательность	Низкий
Сопровождаемость	Анализируемость	Средний
	Изменяемость	Средний
	Стабильность	Низкий
	Тестируемость	Средний
Мобильность	Адаптируемость	Средний
	Простота установки	Средний
	Сосуществование–соответствие	Низкий
	Замещаемость	Низкий
Качество в использовании	Системная эффективность	Высокий
	Производительность	Высокий
	Защищенность	Высокий
	Соответствие требованиям	Средний

Высшие приоритеты естественно, выделены функциональной пригодности и в наибольшей степени поддерживающим ее характеристикам: корректности, защищенности, надежности, системной и временной эффективности. Эти приоритеты должны учитываться при выделении доступных ресурсов для достижения требуемых значений характеристик в жизненном цикле ПС. Остальным субхарактеристикам могут быть присвоены более низкие приоритеты.

Выбор требований к мерам характеристик качества должен начинаться с определения необходимых свойств и значений группы показателей, отражающих функциональные возможности ПС, куда входят субхарактеристики: функциональная пригодность, корректность, способность к взаимодействию и защищенность. Для конкретного проекта ПС перечень атрибутов качества следует адаптировать и уточнять с учетом его назначения и функций.

Выбор и формирование требований к функциональной пригодности ПС наиболее ответственная задача начальных этапов проектирования и всего последующего развития жизненного цикла. В данном примере эта задача не иллюстрируется, вследствие необходимости излишнего расширения и еще большей детализации примера.

Пример выбора требований к трем субхарактеристикам качества из группы функциональные возможности ПС сложной административной ИС кратко представлен в таблице 3.2. Требования к субхарактеристике корректность могут представляться в виде описания двух основных свойств, которым должны соответствовать все программные компоненты и ПС в целом. Первое требование состоит в выполнении полной прослеживаемости сверху вниз реализации требований технического задания и спецификации на ПС при последовательной детализации описаний программных компонентов вплоть до текстов и объектного кода программ. Второе требование заключается в выборе степени и стратегии покрытия тестами программных компонентов, достаточной для функционирования ПС с необходимым качеством и точностью результатов. Реализация этих требований должна обеспечивать требуемую функциональную пригодность.

Требования к характеристике способность к взаимодействию могут быть достаточно полно формализованы и утверждены в процессе системного проектирования, с некоторыми уточнениями на последующих этапах. Их основой являются нормативные документы на интерфейсы открытых систем или выбранные для конкретного проекта стандарты де-факто.

Выбор и формирование требований к характеристике защищенность должны осуществляться на основе потребностей эффективной реализации назначения и функций ПС. В процессе системного анализа и проектирования должны быть выявлены потенциальные предумышленные и случайные угрозы функционированию ПС и установлен необходимый уровень защиты данного комплекса программ.

Таблица 3.2

Пример требований к характеристикам качества функциональных возможностей программного средства

Характеристика качества	Содержание требований	Реализация требований (есть – нет)
Корректность	Требуемая прослеживаемость соответствия текстов программ требованиям к функциям программных компонентов и ПС в целом. Требуемая степень покрытия тестами структуры программных компонентов и ПС в целом.	
Способность к взаимодействию	Соответствие ПС утвержденным документам на интерфейсы: с аппаратной, операционной и внешней средой ИС. Соответствие ПС утвержденным документам на интерфейсы с пользователями.	
Защищенность	Соответствие выбранным, стандартизированным категориям защищенности ПС. Использование выбранных и согласованных методов и средств равнопрочной защиты ПС.	

Пример требований к основным количественным характеристикам качества ПС сложной административной ИС представлен в таблице 3.3. Требования к атрибутам характеристики надежность могут быть выбраны с учетом следующих факторов. При отсутствии автоматического рестарта, за счет отладки и при наличии администратора, контролирующего работоспособность ПС, можно считать допустимой наработку на отказ порядка 10 часов. За счет программно-аппаратных механизмов автоматического рестарта эта наработка при проявлении отказов, может быть повышена приблизительно в 5 раз, т.е. при 80% отказов, возможно их автоматическое обнаружение и оперативное восстановление, вследствие чего наработка на отказ возрастет до 50 часов. По опыту на это может потребоваться около 10% вычислительных ресурсов ИС. Предполагается, что для оперативной работы пользователей административной ИС допустимая длительность прерывания работы для полного восстановления нормального функционирования системы может составлять не более 5 минут.

В результате при таких значениях атрибутов надежности, коэффициент готовности – вероятность застать ПС в работоспособном состоянии, составит достаточно высокую величину 0,998.

Таблица 3.3

Пример требований к количественным характеристикам качества программного средства

Характеристики качества	Мера	Требуемое значение	Реальное значение
Надежность			
Завершенность: - наработка на отказ при отсутствии рестарта.	Часы	10	
Устойчивость: - наработка на отказ при наличии автоматического рестарта; - относительные ресурсы на обеспечение надежности и рестарта	Часы %	50 10	
Восстанавливаемость: - длительность восстановления.	Минуты	5	
Доступность-готовность: - относительное время работоспособного функционирования.	Вероятность	0,998	
Эффективность			
Временная эффективность: - время отклика – получения результатов на типовое задание; - пропускная способность – число типовых заданий, исполняемых в единицу времени.	Секунды Число в минуту	5 20	
Используемость ресурсов: - относительная величина использования ресурсов ЭВМ при нормальном функционировании программного средства.	Вероятность	0,8	

Также как при формировании требований к корректности для характеристики надежности большое значение имеет установление требований к степени покрытия тестами в процессе отладки структуры программных компонентов и ПС в целом. Формализацию этой характеристики целесообразно устанавливать отдельно от общих характеристик качества для проверки на технологических этапах тестирования и испытаний. Основные атрибуты надежности могут быть объективно измерены и сопоставлены с требованиями, для чего в таблице 3.3 предусмотрен столбец – реальное значение. При этом

следует учитывать, что в примере принято весьма малое время восстановления, обусловленное мелкими программными дефектами без учета физического разрушения компонентов, которое должно поддерживаться дублированием вычислительных средств и высокой автоматизацией процессов аппаратурного обеспечения надежности ПС.

Основные требования к атрибутам характеристики эффективность использования вычислительных ресурсов ИС сосредоточены на наиболее критичных показателях производительности и длительности решения функциональных задач. В отличии от объемов памяти, временные характеристики труднее устанавливать и измерять и их ограниченность сильнее влияет на функциональную пригодность ПС. Для оперативной работы пользователей важно иметь малое (несколько секунд) время отклика из ЭВМ после получения типового задания и начала решения требуемой функциональной задачи. Это время обычно желательно иметь в пределах нескольких (для примера принято пяти) секунд, хотя длительность полной реализации задания может быть значительно больше. Требуемая пропускная способность решения функциональных задач зависит от их содержания и числа действующих пользователей. В примере предполагается, что десять операторов, могут вводить в минуту по два задания каждый, которые должны исполняться в отведенное время без дополнительной задержки, что приводит к требованию пропускной способности данного ПС на выбранной вычислительной среде – 20 заданий в минуту. Требования к используемости ресурсов памяти и производительности вычислительных средств могут устанавливаться, исходя, с одной стороны, из экономической целесообразности применения наиболее дешевой, с минимальными ресурсами ЭВМ, загрузка которой будет в среднем не ниже 0,5. С другой стороны, высокая загрузка (выше 0,9) может приводить к нежелательной задержке или даже потере заданий при случайном, кратковременном повышении их интенсивностей, что может негативно отразиться на функциональной пригодности. Таким образом, в данном примере рациональная величина вероятности использования ресурсов ЭВМ в процессе нормального функционирования ПС должна находится в пределах 0,8.

Пример требований к качественным характеристикам ПС представлен в таблице 3.4. Основная часть атрибутов качества в таблице 3.4. сведена к экономическим показателям трудоемкости и длительности, требуемых для реализации соответствующих характеристик.

Требования к практичности и ее субхарактеристикам – понятности и простоты использования, зависят от назначения и функций ПС и могут качественно формализоваться заказчиками набором свойств, необходимых для удобной и комфортной эксплуатации программ. Количественно, простоту использования можно в некоторой степени характеризовать требованиями ограничения средней длительности ввода типовых заданий и времени отклика на них, которое должно быть

Таблица 3.4

Пример требований к качественным характеристикам программного средства

Характеристики качества	Мера	Требуемое значение	Реальное значение
Практичность			
Простота использования:			
- среднее время ввода заданий;	Секунды	10	
- среднее время отклика на задание.	Секунды	5	
Изучаемость:			
- трудоемкость изучения применения ПС;	Чел.-часы	200	
- продолжительность изучения;	Часы	50	
- объем эксплуатационной документации.	Страницы	1000	
Сопровождаемость			
Изменяемость:			
- трудоемкость подготовки изменений;	Чел.-часы	10	
- длительность подготовки изменений.	Часы	5	
Тестируемость:			
- трудоемкость тестирования изменений;	Чел.-часы	20	
- длительность тестирования изменений.	Часы	5	
Мобильность			
Адаптируемость:			
- трудоемкость адаптации;	Чел.-часы	50	
- длительность адаптации.	Часы	10	
Простота установки:			
- трудоемкость инсталляции;	Чел.-часы	10	
- длительность инсталляции.	Часы	5	
Замещаемость:			
- трудоемкость замены компонентов;	Чел.-часы	50	
- длительность замены компонентов.	Часы	10	

в несколько раз меньше. Требования к продолжительности изучения ПС, достаточной для эффективной эксплуатации сложной административной ИС квалифицированным специалистом, в данном примере могут составить около недели или порядка 50 часов. Для коллектива из четырех человек-эксплуатационников это потребует трудоемкости около 200 человеко-часов. Для обеспечения полноценного изучения процессов применения ПС этими

специалистами может быть необходима эксплуатационная документация объемом около 1000 страниц, а также желательны адекватные по содержанию электронные учебники. Малый объем эксплуатационной документации может снизить качество и полноту использования функций сложного ПС, а очень большой объем – также может ухудшить эксплуатацию из-за трудности выделения и освоения наиболее существенных свойств и особенностей применения ПС из множества второстепенных деталей.

Требования к компонентам сопровождаемости количественно можно установить для субхарактеристик изменяемости и тестируемости. Требуемые значения зависят от четкости концепции и архитектуры ПС, от качества технологической документации и еще от некоторых факторов. Обобщенно это отражается на длительности и трудоемкости подготовки и реализации типовых изменений, обусловленных необходимостью устранения дефектов и небольшими усовершенствованиями функций ПС. В рассматриваемом примере для подготовки и выполнения каждого изменения (без учета затрат времени на обнаружение и локализацию дефекта) можно принять среднюю продолжительность в 5 часов и суммарную трудоемкость двух специалистов – 10 человеко-часов. Требования к продолжительности тестирования таких изменений могут составить также до 5 часов, но трудоемкость может увеличиться до 20 человеко-часов, так как требуемый коллектив тестировщиков может возрасти до трех-четырех специалистов.

Выбор и установление требований к мобильности ПС в данном примере сведены к трудоемкости и длительности процессов: адаптации к характеристикам пользователей и внешней среды, инсталляции версий ПС в среде пользователей и замены крупных компонентов версий ПС по требованиям заказчиков или конкретных пользователей. Наиболее простым и легко формализуемым из перечисленных процессов является инсталляция готовой версии ПС с комплектом документации без дополнительных изменений на платформе пользователя, которая может требовать до 5 часов работы двух специалистов (10 человеко-часов). Более сложный процесс включает адаптацию ПС по формализованным инструкциям к специфической аппаратной и внешней среде конкретного пользователя, которая может потребовать вдвое большего времени и в несколько раз (в примере 5) большего числа специалистов. Еще более сложный и трудоемкий процесс замены крупных компонентов ПС и перенос их на иную аппаратурную и операционную платформу. Для этого процесса в примере требуется не менее 20 часов и коллектив около 5 человек (100 человеко-часов).

Рассмотренный пример выбора и формирования характеристик качества проекта ПС, может служить ориентиром подходов при анализе факторов и реализации процессов установления требований к ним в технических заданиях и спецификациях. Обсуждение и согласование

между заказчиком и разработчиком рациональных значений всего ансамбля характеристик и их атрибутов качества позволяет избегать как нецелесообразного завышения требований, так и снижения требований к отдельным характеристикам, которые могут негативно отразиться на функциональной пригодности ПС.

4. ЗАЩИТА ПРОГРАММ И ДАННЫХ

4.1. Особенности систем защиты информации

В связи с широким распространением персональных компьютеров не только как средств обработки информации, но также как оперативных средств коммуникации (электронная, телефаксная почта), возникают проблемы, связанные с обеспечением защиты информации от преднамеренных или случайных искажений.

Актуальность этих проблем подчеркивается также тем обстоятельством, что персональный компьютер или автоматизированное рабочее место (АРМ) является частью систем обработки информации, систем коллективного пользования, вычислительных сетей. В таких случаях предъявляются достаточно жесткие требования по надежности и достоверности передаваемой информации.

Любой канал связи характеризуется наличием в нем помех, приводящих к искажению информации, поступающей на обработку. С целью уменьшения вероятности ошибок принимается ряд мер, направленных на улучшение технических характеристик каналов, на использование различных видов модуляции, на расширение пропускной способности и т. п. При этом также должны приниматься меры по защите информации от ошибок или несанкционированного доступа.

Доступ к информации – это получение возможности использовать информацию, хранящуюся в ЭВМ (системе).

Всякая информация в машине или системе требует той или иной **защиты,** под которой понимается совокупность методов, позволяющих управлять доступом выполняемых в системе программ к хранящейся в ней информации.

Задача защиты информации в информационных вычислительных системах решается, как правило, достаточно просто: обеспечиваются средства контроля за выполнением программ, имеющих доступ к хранимой в системе информации. Для этих целей используются либо списки абонентов, которым разрешен доступ, либо *пароли,* что обеспечивает защиту информации при малом количестве пользователей. Однако при широком распространении вычислительных и информационных систем, особенно в таких сферах, как обслуживание населения, банковское дело, этих средств оказалось явно недостаточно. Система, обеспечивающая защиту информации, не должна позволять

доступа к данным пользователям, не имеющим такого права. Такая система защиты является неотъемлемой частью любой системы коллективного пользования средствами вычислительной техники, независимо от того, где они используются. Данные экспериментальных исследований различных систем коллективного пользования показали, что пользователь в состоянии написать программы, дающие ему доступ к любой информации, находящейся в системе. Как правило, это обусловлено наличием каких-то ошибок в программных средствах, что порождает неизвестные пути обхода установленных преград.

В процессе разработки систем защиты информации выработались некоторые общие правила, которые были сформулированы Ж.Солцером и М. Шредером (США):

1. *Простота механизма защиты.* Так как средства защиты усложняют и без того уже сложные программные и аппаратные средства, обеспечивающие обработку данных в ЭВМ, естественно стремление упростить эти дополнительные средства. Чем лучше совпадает представление пользователя о системе защиты с ее фактическими возможностями, тем меньше ошибок возникает в процессе работы.

2. *Разрешения должны преобладать над запретами.* Нормальным режимом работы считается отсутствие доступа, а механизм защиты должен быть основан на условиях, при которых доступ разрешается. Допуск дается лишь тем пользователям, которым он необходим.

3. *Проверка полномочий любого обращения к любому объекту информации.* Это означает, что защита выносится на общесистемный уровень и предполагает абсолютно надежное определение источника любого обращения.

4. *Разделение полномочий* заключается в определении для любой программы и любого пользователя в системе минимального круга полномочий. Это позволяет уменьшить ущерб от сбоев и случайных нарушений и сократить вероятность непреднамеренного или ошибочного применения полномочий.

5. *Трудоемкость проникновения в систему.* Фактор трудоемкости зависит от количества проб, которые нужно сделать для успешного проникновения. Метод прямого перебора вариантов может дать результат, если для анализа используется сама ЭВМ.

6. *Регистрация проникновений в систему.* Иногда считают, что выгоднее регистрировать случаи проникновения, чем строить сложные системы защиты.

4.2. Криптографические методы защиты информации

Обеспечение защиты информации от несанкционированного доступа – дело сложное, требующее широкого проведения теоретических и экспериментальных исследований по вопросам системного проектирования. Наряду с применением разных

приоритетных режимов и систем разграничения доступа разработчики информационных систем уделяют внимание различным криптографическим методам обработки информации.

Криптографические методы можно разбить на два класса:

1) обработка информации путем замены и перемешивания букв, при котором объем данных не меняется (**шифрование**);

2) сжатие информации с помощью замены отдельных сочетаний букв, слов или фраз (кодирование).

По способу реализации криптографические методы возможны в аппаратном и программном исполнении.

Для защиты текстовой информации при передачах на удаленные станции телекоммуникационной сети используются аппаратные способы шифрования и кодирования. Для обмена информацией между ЭВМ по телекоммуникационной сети, а также для работы с локальными абонентами возможны как аппаратные, так и программные способы. Для хранения информации на магнитных носителях применяются программные способы шифрования и кодирования.

В простейшем случае для построения правил шифрования текстовой информации используется некоторый смешанный алфавит, например перестановка обычного алфавита. В примере 4.1 показан исходный алфавит, смешанный алфавит и шифрование короткого сообщения, в котором каждая буква заменяется соответствующей буквой смешанного алфавита.

Пример 4.1. Зашифруйте с помощью представленного кода фразу "Попробуйте прочитать шифрованный текст".

Алфавит:	АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ
Код:	ЙЦУКЕНГШЩЗХЪФЫВАПРОЛДЖЭЯЧСМИТЬБЮ
Шифруемая информация:	ПОПРОБУЙТЕ ПРОЧИТАТЬ ШИФРОВАННЫЙ ТЕКСТ
Ответ:	
Код:	АВАПВЦЛЗОН АПВЯЩОЙОТ ЧШДПВУЙЫЫИЗ ОНХРО

Сообщения, зашифрованные с помощью простой подстановки, расшифровываются следующим образом. Определяется частота появления каждой буквы в шифрованном сообщении и сравнивается с частотами букв алфавита. Таблица частот букв русского алфавита (табл. 4.1) приведена ниже

Для сообщений длиной 40 символов и более по распределениям частот можно определить буквы исходного текста.

Более надежное шифрование текстов обеспечивается с помощью ключевых слов и нескольких алфавитов шифрования. Ниже (табл. 4.2) показан квадрат Виженера, построенный на основе смешанного алфавита, приведенного в примере 4.1. Каждая строка квадрата образуется из предыдущей с помощью циклического сдвига на одну

Таблица 4.1

Таблица частот букв русского алфавита

Буква	Частота	Буква	Частота
О	0,0940	Б	0,0197
А	0,0896	З	0,0193
Е	0,0856	У	0,0179
И	0,0739	Г	0,0153
Н	0,0662	Ь	0,0125
Т	0,0611	Ч	0,0118
Р	0,0561	Й	0,0094
С	0,0554	Х	0,0093
П	0,0421	Ц	0,0087
М	0,0417	Ж	0,0064
В	0,0400	Ю	0,0063
Л	0,0358	Щ	0,0048
К	0,0322	Ф	0,0034
Д	0,0280	Э	0,0033
Я	0,0243	Ш	0,0032
Ы	0,0225	Ъ	0,0002

позицию. Таким образом, квадрат состоит из 32 смешанных алфавитов, каждому из них соответствуют буквы исходного алфавита. Эти буквы записываются в квадрате слева перед каждым смешанным алфавитом.

Пример 4.2. Зашифруйте с помощью представленного кода следующую фразу:

Информация: ПОПРОБУЙТЕ ПРОЧИТАТЬ ШИФРОВАННЫЙ ТЕКСТ

Ключевое слово: КЛЮЧКЛЮЧКЛ ЮЧКЛЮЧКЛЮ ЧКЛЮЧКЛЮЧКЛ ЮЧКЛЮ

Ответ: ССЫШЧФРЙТП ЫШЧУГЗХЪМ АОЮВНФЬЬЕНД ПТАТП

или

ССЫШ ЧФРЙ ТПЫШ ЧУГЗ ХЪМА ОЮВН ФЬЬЕ НДПТ ДТП

На примере 4.2 показано шифрование фразы при помощи ключевого слова КЛЮЧ и данного квадрата. Ключевое слово многократно записывается под исходным текстом, и буквы исходного текста шифруются при помощи соответствующих смешанных алфавитов. Этот метод шифрования уже не поддается раскрытию с помощью простого подсчета частот букв.

Если использовать несколько ключевых слов различной длины, то можно еще более повысить защищенность информации, поскольку разным сообщениям будут соответствовать разные ключевые слова.

Таблица 4.2

Квадрат Виженера

	АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ
А	ЙЦУКЕНГШЩЗХЪФЫВАПРОЛДЖЭЯЧСМИТЬБЮ
Б	ЦУКЕНГШЩЗХЪФЫВАПРОЛДЖЭЯЧСМИТЬБЮЙ
В	УКЕНГШЩЗХЪФЫВАПРОЛДЖЭЯЧСМИТЬБЮЙЦ
Г	КЕНГШЩЗХЪФЫВАПРОЛДЖЭЯЧСМИТЬБЮЙЦУ
Д	ЕНГШЩЗХЪФЫВАПРОЛДЖЭЯЧСМИТЬБЮЙЦУК
Е	НГШЩЗХЪФЫВАПРОЛДЖЭЯЧСМИТЬБЮЙЦУКЕ
Ж	ГШЩЗХЪФЫВАПРОЛДЖЭЯЧСМИТЬБЮЙЦУКЕН
З	ШЩЗХЪФЫВАПРОЛДЖЭЯЧСМИТЬБЮЙЦУКЕНГ
И	ЩЗХЪФЫВАПРОЛДЖЭЯЧСМИТЬБЮЙЦУКЕНГШ
Й	ЗХЪФЫВАПРОЛДЖЭЯЧСМИТЬБЮЙЦУКЕНГШЩ
К	ХЪФЫВАПРОЛДЖЭЯЧСМИТЬБЮЙЦУКЕНГШЩЗ
Л	ЪФЫВАПРОЛДЖЭЯЧСМИТЬБЮЙЦУКЕНГШЩЗХ
М	ФЫВАПРОЛДЖЭЯЧСМИТЬБЮЙЦУКЕНГШЩЗХЪ
Н	ЫВАПРОЛДЖЭЯЧСМИТЬБЮЙЦУКЕНГШЩЗХЪФ
О	ВАПРОЛДЖЭЯЧСМИТЬБЮЙЦУКЕНГШЩЗХЪФЫ
П	АПРОЛДЖЭЯЧСМИТЬБЮЙЦУКЕНГШЩЗХЪФЫВ
Р	ПРОЛДЖЭЯЧСМИТЬБЮЙЦУКЕНГШЩЗХЪФЫВА
С	РОЛДЖЭЯЧСМИТЬБЮЙЦУКЕНГШЩЗХЪФЫВАП
Т	ОЛДЖЭЯЧСМИТЬБЮЙЦУКЕНГШЩЗХЪФЫВАПР
У	ЛДЖЭЯЧСМИТЬБЮЙЦУКЕНГШЩЗХЪФЫВАПРО
Ф	ДЖЭЯЧСМИТЬБЮЙЦУКЕНГШЩЗХЪФЫВАПРОЛ
Х	ЖЭЯЧСМИТЬБЮЙЦУКЕНГШЩЗХЪФЫВАПРОЛД
Ц	ЭЯЧСМИТЬБЮЙЦУКЕНГШЩЗХЪФЫВАПРОЛДЖ
Ч	ЯЧСМИТЬБЮЙЦУКЕНГШЩЗХЪФЫВАПРОЛДЖЭ
Ш	ЧСМИТЬБЮЙЦУКЕНГШЩЗХЪФЫВАПРОЛДЖЭЯ
Щ	СМИТЬБЮЙЦУКЕНГШЩЗХЪФЫВАПРОЛДЖЭЯЧ
Ъ	МИТЬБЮЙЦУКЕНГШЩЗХЪФЫВАПРОЛДЖЭЯЧС
Ы	ИТЬБЮЙЦУКЕНГШЩЗХЪФЫВАПРОЛДЖЭЯЧСМ
Ь	ТЬБЮЙЦУКЕНГШЩЗХЪФЫВАПРОЛДЖЭЯЧСМИ
Э	ЬБЮЙЦУКЕНГШЩЗХЪФЫВАПРОЛДЖЭЯЧСМИТ
Ю	БЮЙЦУКЕНГШЩЗХЪФЫВАПРОЛДЖЭЯЧСМИТЬ
Я	ЮЙЦУКЕНГШЩЗХЪФЫВАПРОЛДЖЭЯЧСМИТЬБ

Основным отличием двоичной информации при передаче данных от информации любых других видов (текстовой, графической) является ее полная формализованность, т. е. отсутствие каких-либо логических связей между отдельными элементами передаваемой информации. Это приводит к тому, что ошибка в таком сообщении не может быть обнаружена по смысловому содержанию принятой информации.

Основным методом повышения достоверности является помехозащищенное кодирование в сочетании с автоматическим запросом и повторением искаженных кодовых комбинаций. Кодирование заключается во введении определенным образом в информационную последовательность дополнительных проверочных разрядов таким образом, что между отдельными разрядами устанавливается определенная математическая связь. Тогда ошибка, возникающая при прохождении сообщения по каналу связи, вызовет нарушение этой закономерности, что будет обнаружено при декодировании.

Наиболее широкое применение получили в настоящее время *циклические* коды, которые обладают высокими корректирующими свойствами и относительно просто реализуются технически. Циклические коды обнаруживают как независимые, так и групповые ошибки. Обнаруживающие свойства циклических кодов определяются видом порождающего многочлена. Циклический код, образованный многочленом степени "К", обнаруживает все пакеты ошибок длины n≤K. Под **пакетом ошибок длиной "n"** подразумевается любой ряд ошибок, для которого число знаков между первой и последней ошибками, включая эти ошибки, равно "n".

Как показывает опыт, ошибки во всех реальных каналах связи не являются независимыми. Они обычно возникают группами. Закон распределения ошибок в пакетах и самих пакетов очень сложен. Он определяется огромным количеством факторов, учесть которые не просто.

4.3. Аппаратные средства защиты

Аппаратные способы шифрования информации применяются для передачи защищенных данных по телекоммуникационной сети. Для реализации шифрования с помощью смешанного алфавита используется перестановка отдельных разрядов в пределах одного или нескольких символов.

На рис. 4.1 показана схема аппаратного шифрователя, использующего операцию перестановки разрядов в пределе одного байта информации. Для расшифровки сообщений применяется симметричная перестановка. Блок перестановки может быть сменным или управляемым. Блок управления синхронизирует работу шифровального устройства. Возможное число перестановок для n-разрядных символов равно (n!-1). Если перестановка разрядов выполняется в пределах нескольких байт, то такая операция называется *запутыванием.*

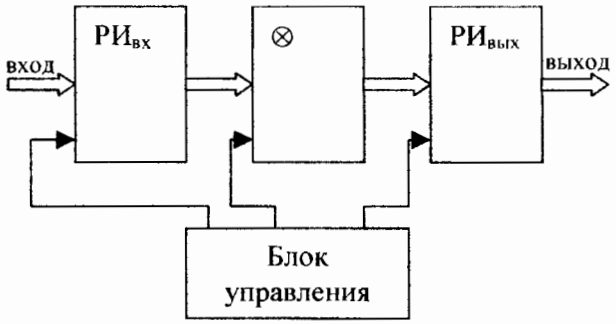

Рис. 4.1. Схема шифрователя с перестановкой разрядов: $РИ_{вх}$ – входной регистр, $РИ_{вых}$ –выходной регистр, ⊗ – блок перестановки

Для шифрования с помощью ключевых слов применяется операция сложения по модулю 2 (исключающее ИЛИ). Схема шифрования показана на рис. 4.2.

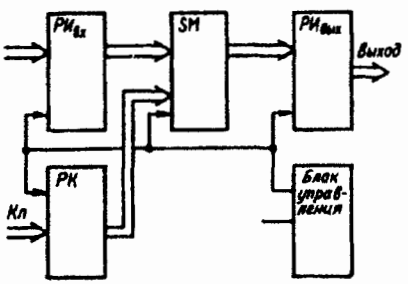

Рис. 4.2. Схема ключевого шифрователя: РК – регистр ключей, *SM*–сумматор

Ключевое слово хранится в 64-разрядном регистре ключа РК. Информация, подлежащая шифрованию, записывается в 64-разрядный регистр информации $РИ_{вх}$. После заполнения этого регистра выполняется операция сложения по модулю 2 с содержимым регистра ключа. Результат представляет собой зашифрованную информацию, которая поступает в выходной 64-разрядный регистр $РИ_{вых}$.

На практике используются несколько чередующихся операций запутывания и сложения по модулю 2 с различными ключами. Пример такой схемы приведен на рис. 4.3. Блок управления на рисунке не показан.

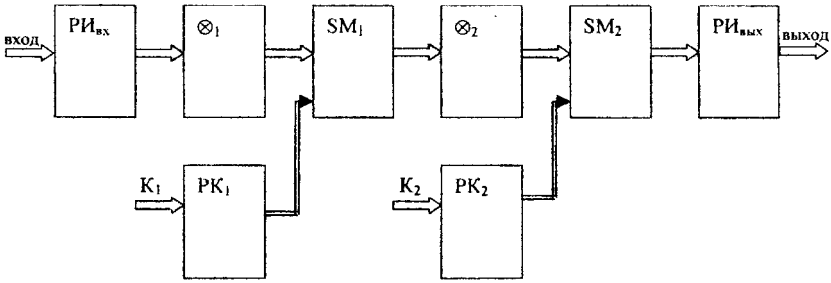

Рис. 4.3. Схема шифрователя с двойным ключом

При определении количества операций запутывания и сложения по модулю 2 приходится искать компромиссное решение между сложностью шифрования и временем преобразования.

Ключевые слова для работы схем шифрования выбираются с помощью специальных генераторов случайных чисел и передаются в приемное устройство в зашифрованном виде по предыдущим ключам. Дешифрование информации выполняется в обратной последовательности.

4.4. Программные средства защиты

Программные способы применяются для шифрования информации, хранящейся на магнитных носителях (дисках, лентах). Это могут быть данные различных информационно-справочных систем АСУ, АСОД и др. Программные способы шифрования сводятся к операциям перестановки, перекодирования и сложения по модулю 2 с ключевыми словами. При этом используется команда ассемблера XOR (исключающее ИЛИ).

Кодирование информации. Особое место в программах обработки информации занимают операции кодирования. Преобразование информации, в результате которого обеспечивается изменение объема памяти, занимаемой данными, называется **кодированием.** На практике кодирование всегда используется для уменьшения объема памяти, так как экономия памяти ЭВМ имеет большое значение в информационных системах. Кроме того, кодирование можно рассматривать как криптографический метод обработки информации.

Естественные языки обладают большой избыточностью. Не составит большого труда исправить все ошибки, которые есть в следующей фразе:

"Есл нсклко бкв удлть, эт прдлжн ещ м б прчтно". Для экономии памяти, объем которой ограничен, имеет смысл ликвидировать избыточность текста или уплотнить текст.

Существуют несколько способов уплотнения текста.

1. Переход от естественных обозначений к более компактным. Этот способ применяется для сжатия записи дат, номеров изделий, уличных адресов и т. п. Идея способа показана на примере сжатия записи даты. Обычно мы записываем дату в виде 22.09.83, что требует 8 байтов памяти ЭВМ. Однако ясно, что для представления дня достаточно 5 битов, месяца – 4, года – не более 7, т. е. вся дата может быть записана в 16 битах или в 2 байтах.

Другой распространенный способ представления дат был предложен Жозефом Скалигером в 1582 г. для астрономических целей. По этому способу дата записывается как общее число дней, прошедшее к данному дню, считая с 1 января 4713 г. до н. э. По этой схеме 1 января 1983 г. записывается как 2 444 323. Обычно для реальных расчетов ограничиваются четырьмя последними цифрами этого представления. 24 мая 1967г. записывается в виде 0000, и отсчет дней от этой даты требует 16 битов в упакованном десятичном формате.

2. **Подавление повторяющихся символов**. В различных информационных текстах часто встречаются цепочки повторяющихся символов, например пробелы или нули в числовых полях. Если имеется группа повторяющихся символов длиной более 3, то ее длину можно сократить до трех символов. Сжатая таким образом группа повторяющихся символов представляет собой триграф $S \cdot P \cdot N$, в котором S – символ повторения; P – признак повторения; N – количество символов повторения, закодированных в триграфе.

В других схемах подавления повторяющихся символов используют особенность кодов ДКОИ, КОИ-7, КОИ-8, заключающуюся в том, что большинство допустимых в них битовых комбинаций не используется для представления символьных данных. Обычно в коде ДКОИ не используются комбинации с нулем во второй слева позиции. Этот бит может являться признаком повторяемости последующего или предыдущего символа:

3. **Кодирование часто используемых элементов данных**. Этот способ уплотнения данных также основан на употреблении неиспользуемых комбинаций кода ДКОИ. Для кодирования, например, имен людей можно использовать комбинации из двух байтов дифраф $P \cdot N$, где P – признак кодирования имени, N – номер имени. Таким

образом может быть закодировано 256 имен людей, чего обычно бывает достаточно в информационных системах. Если в байте *N* старший байт использовать в качестве признака пола, то этим байтом можно закодировать 128 мужских и 128 женских имен.

Другой способ основан на отыскании в тексте наиболее часто встречающихся сочетаний букв и даже слов и замене их на неиспользуемые байты кода ДКОИ.

4. **Посимвольное кодирование**. Семибитовые и восьмибитовые коды не обеспечивают достаточно компактного кодирования символьной информации. Более пригодными для этой цели являются 5-битовые коды, например международный телеграфный код МГК-2. Перевод информации в код МГК-2 возможен с помощью программного перекодирования или с использованием специальных элементов на основе больших интегральных схем (БИС). Пропускная способность каналов связи при передаче алфавитно-цифровой информации в коде МГК-2 повышается по сравнению с использованием 8-битовых кодов почти на 40%.

5. **Коды переменной длины**. Коды с переменным числом битов на символ позволяют добиться еще более плотной упаковки данных. Метод заключается в том, что часто используемые символы кодируются короткими кодами, а символы с низкой частотой использования – длинными кодами. Идея такого кодирования была впервые высказана Хаффманом, и соответствующий код называется кодом Хаффмана. Использование кодов Хаффмана позволяет достичь сокращения исходного текста почти на 80 %.

Использование различных методов уплотнения текстов кроме своего основного назначения – уменьшения информационной избыточности – обеспечивает определенную криптографическую обработку информации. Однако наибольшего эффекта можно достичь при совместном использовании как методов шифрования, так и методов кодирования информации.

Способы контроля правильности передачи данных. Управление правильностью (помехозащищенностью) передачи информации по каналам связи выполняется с помощью помехоустойчивого кодирования. Простейшими способами обнаружения ошибок являются контрольное суммирование, проверка на нечетность. Однако они недостаточно надежны, особенно при появлении пачек ошибок. К числу эффективных кодов, обнаруживающих одиночные, кратные ошибки и пачки ошибок, относятся циклические коды (CRC – Cyclic Redundance Code).

Один из вариантов циклического кодирования заключается в умножении исходного кода на образующий полином g(x), а декодирование – в делении на g(x). Если остаток от деления не равен нулю, то произошла ошибка.

Образующий полином есть двоичное представление одного из простых множителей, на которые раскладывается число X^n-1, где X^n

обозначает единицу в n-м разряде, n равно числу разрядов кодовой группы. Так, если n=10 и X=2, то $X^n-1=1023=11\cdot93$, и если g(X)=11 или в двоичном коде 1011, то примеры циклических кодов $A_i \times g(X)$ чисел A_i в кодовой группе при этом образующем полиноме можно видеть в табл. 4.3.

Таблица 4.3

Примеры циклических кодов

Число	Циклический код
0	0000000000
1	0000001011
2	0000010110
3	0000100001
...	
13	0010001111
14	0010011010
15	0010100101
16	0011000110
...	

Основной вариант циклического кода, широко применяемый на практике, отличается от предыдущего тем, что операция деления на образующий полином заменяется следующим алгоритмом: 1) к исходному кодируемому числу A справа приписывается K нулей, где K– число битов в образующем полиноме, уменьшенное на единицу; 2) над полученным числом $A(2^k)$ выполняется операция O, отличающаяся от деления тем, что на каждом шаге операции вместо вычитания выполняется поразрядная операция "исключающее ИЛИ"; 3) полученный остаток B и есть CRC – избыточный K-разрядный код, который заменяет в закодированном числе C приписанные справа K нулей, т.е.

$$C = A(2^k) + B.$$

На приемном конце над кодом C выполняется операция O. Если остаток не равен нулю, то при передаче произошла ошибка и нужна повторная передача кода A.

Пример. Пусть A = 1001 1101, образующий полином 11001.

Так как K=4, то $A(2^k)=100111010000$. Выполнение операции O расчета циклического кода показано на рис. 4.4.

Рис. 4.4. Пример получения циклического кода

4.5. Защита программных продуктов

Основные понятия о защите программных продуктов. Программные продукты и компьютерные базы данных являются предметом интеллектуального труда специалистов высокой квалификации. Процесс проектирования и реализации программных продуктов характеризуется значительными материальными и трудовыми затратами, основан на использовании наукоемких технологий и инструментария, требует применения и соответствующего уровня дорогостоящей вычислительной техники. Это обусловливает необходимость принятия мер по защите интересов разработчика программ и создателей компьютерных баз данных от несанкционированного их использования.

Программное обеспечение является объектом защиты также и в связи со сложностью и трудоемкостью восстановления его работоспособности, значимостью программного обеспечения для работы информационной системы.

Защита программного обеспечения преследует цели:

- ограничение несанкционированного доступа к программам или их преднамеренное разрушение и хищение;
- исключение несанкционированного копирования (тиражирования) программ.

Программный продукт и базы данных должны быть защищены по нескольким направлениям от воздействия:

- человека – хищение машинных носителей и документации программного обеспечения; нарушение работоспособности программного продукта и др.;
- аппаратуры – подключение к компьютеру аппаратных средств для считывания программ и данных или их физического разрушения;
- специализированных программ – приведение программного продукта или базы данных в неработоспособное состояние (например,

вирусное заражение), несанкционированное копирование программ и базы данных и т.д.

Самый простой и доступный способ защиты программных продуктов и базы данных – *ограничение доступа.* Контроль доступа к программному продукту и базе данных строится путем:

- парольной защиты программ при их запуске;
- использования ключевой дискеты для запуска программ;
- ограничения программ или данных, функций обработки, доступных пользователям, и др.

Могут также использоваться и *криптографические методы* защиты информации базы данных или головных программных модулей.

Программные системы защиты от несанкционированного копирования. Данные системы предотвращают нелицензионное использование программных продуктов и баз данных. Программа выполняется только при опознании некоторого уникального некопируемого ключевого элемента. Таким ключевым элементом могут быть:

- дискета, на которой записан не подлежащий копированию ключ;
- определенные характеристики аппаратуры компьютера;
- специальное устройство (электронный ключ), подключаемое к компьютеру и предназначенное для выдачи опознавательного кода.

Программные системы защиты от копирования программных продуктов:

- идентифицируют среду, из которой будет запускаться программа;
- устанавливают соответствие среды, из которой запущена программа, той, для которой разрешен санкционированный запуск;
- вырабатывают реакцию на запуск из несанкционированной среды;
- регистрируют санкционированное копирование;
- противодействуют изучению алгоритмов и программ работы системы.

Для идентификации *запускающих дискет* применяются следующие методы:

- нанесение повреждений на поверхность дискеты ("лазерная дыра"), которая с трудом может быть воспроизведена в несанкционированной копии дискеты;
- нестандартное форматирование запускающей дискеты.

Идентификация среды компьютера обеспечивается за счет:

- закрепления месторасположения программ на жестком магнитном диске (так называемые *неперемещаемые программы);*
- привязки к номеру BIOS (расчет и запоминание с последующей проверкой при запуске контрольной суммы системы);
- привязки к аппаратному (электронному) ключу, вставляемому в порт ввода-вывода, и др.

На Западе наиболее популярны методы *правовой* защиты программных продуктов и баз данных.

Правовые методы защиты программных продуктов и баз данных. Правовые методы защиты программ включают:

- патентную защиту;
- закон о производственных секретах;
- лицензионные соглашения и контракты;
- закон об авторском праве.

Различают две категории прав:

- экономические права, дающие их обладателям право на получение экономических выгод от продажи или использования программных продуктов и баз данных;
- моральные права, обеспечивающие защиту личности автора в его произведении.

Во многих цивилизованных странах несанкционированное копирование программ в целях продажи или бесплатного распространения рассматривается как государственное преступление, карается штрафом или тюремным заключением. Но, к сожалению, само авторское право не обеспечивает защиту новой идеи, концепции, методологии и технологии разработки программ, поэтому требуются дополнительные меры их защиты.

Патентная защита устанавливает приоритет в разработке и использовании нового подхода или метода, примененного при разработке программ, удостоверяет их оригинальность.

Статус *производственного секрета* для программы ограничивает круг лиц, знакомых или допущенных к ее эксплуатации, а также определяет меру их ответственности за разглашение секретов. Например, используется парольный доступ к программному продукту или базе данных, вплоть до паролей на отдельные режимы (чтение, запись, корректировку и т.п.). Программы, как любой материальный объект большой стоимости, необходимо охранять от кражи и преднамеренных разрушений.

Лицензионные соглашения распространяются на все аспекты правовой охраны программных продуктов, включая авторское право, патентную защиту, производственные секреты. Наиболее часто используются лицензионные соглашения на передачу авторских прав.

Лицензия – договор на передачу одним лицом (лицензиаром) другому лицу (лицензиату) права на использование имени, продукции, технологии или услуги. *Лицензиар* увеличивает свои доходы сбором лицензионных платежей, расширяет область распространения программного продукта или базы данных; *лицензиат* извлекает доходы за счет их применения.

В лицензионном соглашении оговариваются все условия эксплуатации программ, в том числе создание копий. На каждой копии программы должны быть те же отметки, что и на оригинале:

- знак авторского права (обычно ©) и название разработчика, года выпуска программы, прочих ее атрибутов;

•знак патентной защиты или производственного секрета;

•торговые марки, соответствующие использованным в программе другим программным изделиям (обычно – ™ и название фирмы-разработчика программного продукта);

•символ зарегистрированного права на распространение программного продукта (обычно ®).

Существует несколько типов лицензий на программные продукты.

Исключительная лицензия – продажа всех имущественных прав на программный продукт или базу данных, покупателю лицензии предоставляется исключительное право на их использование, а автор или владелец патента отказывается от самостоятельного их применения или предоставления другим лицам.

Это самый дорогой вид лицензии, к нему прибегают для монопольного владения с целью извлечения дополнительной прибыли либо с целью прекращения существования на рынке программных средств программного продукта.

Простая лицензия – лицензиар предоставляет право лицензиату использовать программный продукт или базу данных, оставляя за собой право применять их и предоставлять на аналогичных условиях неограниченному числу лиц (лицензиат при этом не может сам выдавать сублицензии, может лишь продать копии приобретенного программного продукта или базы данных).

Такой вид лицензии приобретают дилер (торговец) либо фирмы-производители, использующие купленные лицензии как сопутствующий товар к основному виду деятельности. Например, многие производители и фирмы, торгующие компьютерной техникой, осуществляют продажу вычислительной техники с установленным лицензионным программным обеспечением (операционная система, текстовый редактор, электронная таблица, графические пакеты и т.д.).

Этикеточная лицензия – лицензия на одну копию программного продукта или базы данных. Данный тип лицензии применяется при розничной продаже. Каждый официальный покупатель заключает лицензионное соглашение с продавцом на их использование, но при этом сохраняется авторское право разработчика.

Экономические отношения между лицензиаром и лицензиатом могут строиться различным образом. За право пользования программным продуктом или базой данных выплачивается единовременное вознаграждение (паушальный платеж), которое и является фактической ценой лицензии. Возможны и периодические отчисления лицензиару за право пользования в виде *роялти* – фиксированная ставка в определенные интервалы времени в течение действия лицензионного соглашения, как правило, процент от стоимости программных продуктов или баз данных.

Закон об охране программных продуктов и компьютерных баз данных автором признает физическое лицо, в результате творческой

деятельности которого они созданы. Автору независимо от его имущественных прав принадлежат личные авторские права: авторство, имя, неприкосновенность (целостность) программ или баз данных.

Авторское право действует с момента создания программного продукта или базы данных в течение всей жизни автора и 50 лет после его смерти. Автор может:

- выпускать в свет;
- воспроизводить в любой форме, любыми способами;
- распространять;
- модифицировать;
- осуществлять любое иное использование программного продукта или базы данных.

Авторское право не связано с правом собственности на материальный носитель.

Имущественные права на программный продукт или базу данных могут быть переданы частично или полностью другим физическим или юридическим лицам по договору. Имущественные права относятся к категории наследуемых. Если программный продукт или база данных созданы в порядке выполнения служебных обязанностей, имущественные права принадлежат работодателю.

Программные продукты и базы данных могут использоваться третьими лицами – *пользователями* на основании договора с правообладателем.

Лицо, правомерно владеющее экземпляром программы или базы данных, вправе, без получения дополнительного разрешения правообладателя, осуществлять любые действия, связанные с функционированием программного продукта или базы данных в соответствии с ее назначением, в том числе:

- устанавливать один экземпляр, если не предусмотрено иное соглашение с правообладателем, программного продукта или базы данных на компьютер;
- исправлять явные ошибки;
- адаптировать программный продукт или базу данных;
- изготавливать страховые копии.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

1. Составьте логическую схему базы знаний по теме курса.

2. Нарисуйте обобщенную структуру системы программирования.

3. Перечислите известные вам системы программирования.

4. Нарисуйте схему основных факторов, влияющих на качество программных средств.

5. Изобразите модель характеристик качества ПС.

6. Нарисуйте схему оценивания эффективности системы защиты ИС.

7. Нарисуйте график изменения вероятности ошибок от длины пакета.

8. Нарисуйте схему шифрователя с перестановкой разрядов.

9. Нарисуйте схему ключевого шифрователя.

10. Нарисуйте схему шифрователя с двойным ключом.

11. Попробуйте раскодировать фразу:

01.02.03 04.05.06.07.08.05.09.10.11.03 04.08.12.13.11.02 14.11.15.08.05.13.12.10.11.03 11.06.04.05.02.16.17.18.09.07.06.03 06.19.09.14.12.10.10.20.21 12.02.15.12.13.11.07

ТРЕНИНГ УМЕНИЙ

1. Пример выполнения упражнения тренинга на умение №1

Задание

Зашифруйте фразу "Криптографические методы защиты информации" с помощью кода:

ЯФЙЧЫЦСВУМАКИПЕТРНЬОГБЛШЮДЩЖЗЭХЪ.

Решение

Предварительно заполните таблицу, подобрав к каждому алгоритму конкретное соответствие из данного задания.

№ п/п	Алгоритм	Конкретное действие, соответствующее предложенному алгоритму
1	Напротив каждой буквы алфавита записать символ кода.	АБВГ ДЕЖЗ ИЙКЛ МНОП РСТУ ФХЦЧ ШЩЪЫ ЬЭЮЯ ЯФЙЧ ЫЦСВ УМАК ИПЕТ РНЬО ГБЛШ ЮДЩЖ ЗЭХЪ.
2	Для каждой буквы фразы определить соответствующий символ кода.	АРУТЬЕЧРЯГУШЩНАУЦ ИЦЬЕЫЖ ВЯДУЬЖ УПГЕРИЯЛУУ.

Решите самостоятельно следующие задания:

Задание 1.1

Зашифруйте фразу "Компилятор как составная часть системы программирования" с помощью кода:
ЮБЬТИМСЧЯЭЖДЛОРПАВЫФЪХЗЩШГНЕКУЦЙ.

Задание 1.2

Зашифруйте фразу “Кодирование часто используемых элементов данных” с помощью кода: ЙУЕГЩХЦКНШЗЪФВПОДЭЫАРЛЖЯСИЬЮЧМТБ.

Задание 1.3

Зашифруйте фразу “Естественные языки обладают большой избыточностью” с помощью кода: ЯФЙЧЫЦСВУМАКИПЕТРНЬОГБЛШЮДЩЖЗЭХЪ.

Задание 1.4

Зашифруйте фразу “Существуют несколько способов уплотнения текста” с помощью кода: ФЭЫЖВДАЛПОРЯЮЧБСЬМТИЙЪЦХУЗКЩЕШНГ.

Задание 1.5

Зашифруйте фразу “Практичность тесно связана с функциональной пригодностью” с помощью кода:
ЯФЙЧЫЦСВУМАКИПЕТРНЬОГБЛШЮДЩЖЗЭХЪ.

2. Пример выполнения упражнения тренинга на умение №2

Задание

Зашифруйте фразу “Криптографические методы защиты информации” с помощью ключевого слова БУКВА и квадрата Виженера (таблица 4.2. в тексте юниты).

Решение

Предварительно заполните таблицу, подобрав к каждому алгоритму конкретное соответствие из данного задания.

№ п/п	Алгоритм	Конкретное действие, соответствующее предложенному алгоритму
1	Под заданной фразой написать ключевое слово, повторив его необходимое количество раз.	КРИПТОГРАФИЧЕСКИЕ МЕТОДЫ ЗАЩИТЫ БУКВАБУКВАБУКВАБУ КВАБУК ВАБУКВ ИНФОРМАЦИИ. АБУКВАБУКВ.
2	Для каждой буквы фразы определить соответствующий символ кода с помощью квадрата Виженера.	ЪКОРОАЭМУДЗХАЛХЗЧ ЭШОАЯН ЗЙМИТЬ ЩВШЧОФЦЗОХ.

Решите самостоятельно следующие задания:

Задание 2.1

Зашифруйте фразу "Компилятор как составная часть системы программирования" с помощью ключевого слова СЛОВО и квадрата Виженера (таблица 4.2. в тексте юниты).

Задание 2.2

Зашифруйте фразу "Кодирование часто используемых элементов данных" с помощью ключевого слова ЧИСЛО и квадрата Виженера (таблица 4.2. в тексте юниты).

Задание 2.3

Зашифруйте фразу "Естественные языки обладают большой избыточностью" с помощью ключевого слова ЮНИТА и квадрата Виженера (таблица 4.2. в тексте юниты).

Задание 2.4

Зашифруйте фразу "Существуют несколько способов уплотнения текста" с помощью ключевого слова ГЛАВА и квадрата Виженера (таблица 4.2. в тексте юниты).

Задание 2.5

Зашифруйте фразу "Практичность тесно связана с функциональной пригодностью" с помощью ключевого слова ПРОГРАММА и квадрата Виженера (таблица 4.2. в тексте юниты).

3. Пример выполнения упражнения тренинга на умение №3

Задание

Определить циклический код для числа 100101_2.

Решение

Предварительно заполните таблицу, подобрав к каждому алгоритму конкретное соответствие из данного задания.

№ п/п	Алгоритм	Конкретное действие, соответствующее предложенному алгоритму
1	Определить образующий полином g, как двоичное представление одного из простых множителей, на которые раскладывается число 2^n-1, где n равно числу разрядов кода.	n=6 $2^6-1=63=7\cdot9$ $g=7_{10}=111_2$
2	К кодируемому числу А справа приписывается К нулей, где К– число битов в образующем полиноме, уменьшенное на единицу.	A=100111 $A(2^k)$=10011100
3	Над полученным числом выполняется операция О, отличающаяся от деления тем, что на каждом шаге операции вместо вычитания выполняется поразрядная операция "исключающее ИЛИ".	10011100 |111 111 111 111 110 111 10 => CRC
4	Полученный остаток заменяет в закодированном числе приписанные справа К нулей.	Циклический код равен: 10011110

Решите самостоятельно следующие задания:

Задание 3.1

Определить циклический код для кода 111000_2.

Задание 3.2

Определить циклический код для кода 101010_2.

Задание 3.3

Определить циклический код для кода 110111_2.

Задание 3.4

Определить циклический код для кода 110010_2.

Задание 3.5

Определить циклический код для кода 100001_2.

4. Пример выполнения упражнения тренинга на умение №4

Задание

Выработать требования к практичности программного средства, автоматизирующего процесс приема коммунальных платежей у населения

Решение

Предварительно заполните таблицу, подобрав к каждому алгоритму конкретное соответствие из данного задания.

№ п/п	Алгоритм	Конкретное действие, соответствующее предложенному алгоритму
1	Определить приоритеты требований к субхарактеристикам	Простота использования – высокий. Изучаемость – средний. Привлекательность – низкий.
2	Определить требования к характеристикам	1. Простота использования: среднее время ввода заданий – 30 сек, среднее время отклика на задание – 5 сек. 2. Изучаемость: продолжительность изучения – 40 часов, объем эксплуатационной документации – 10 страниц. 3. Привлекательность (оценивается только качественно) – удовлетворительно.

Решите самостоятельно следующие задания:

Задание 4.1

Выработать требования к эффективности программного средства, автоматизирующего процесс приема коммунальных платежей у населения.

Задание 4.2

Выработать требования к надежности программного средства, автоматизирующего процесс приема коммунальных платежей у населения.

Задание 4.3

Выработать требования к практичности программного средства, предназначенного для обучения английскому языку детей среднего школьного возраста.

Задание 4.4

Выработать требования к надежности программного средства, предназначенного для обучения английскому языку детей среднего школьного возраста.

Задание 4.5

Выработать требования к мобильности программного средства, предназначенного для обучения английскому языку детей среднего школьного возраста.

5. Пример выполнения упражнения тренинга на умение №5

Задание

Сформируйте документ "Постановка комплекса задач для проектирования программного средства" для программы психодиагностического тестирования при приеме на работу, предназначенной для автоматизации работы менеджера по кадрам.

Решение

Предварительно заполните таблицу, подобрав к каждому алгоритму конкретное соответствие из данного задания.

№ п/п	Алгоритм	Конкретное действие, соответствующее предложенному алгоритму
1	Сформировать характеристики комплекса задач.	1. Назначение комплекса задач: проведение психологического тестирования с целью выявления профессионально важных качеств испытуемого. 2. Перечень объектов, при управлении которыми решают комплекс задач: отдел кадров. 3. Периодичность решения: по запросу менеджера по кадрам.

		4. Условия, при которых прекращается решение комплекса задач автоматизированным способом: при явном несоответствии диагностируемых показателей испытуемого предъявляемым требованиям, выявленном на начальном этапе тестирования. 5. Связи данного комплекса задач с другими комплексами информационных систем: результаты тестирования передаются в информационную систему, содержащую сведения о кадрах. 6. Распределение действий между персоналом и техническими средствами при различных ситуациях решения комплекса задач: менеджер по кадрам производит инструктирование испытуемого перед тестированием, запуск программы, печать результатов тестирования и их интерпретацию; технические средства осуществляют предъявление тестов, фиксирование ответов испытуемого, обработку и выдачу результатов тестирования.
2	Описать входную информацию.	1. Перечень и описание входных сообщений: входными сообщениями являются анкетные данные и ответы испытуемого. 2. Перечень и описание структурных единиц информации входных сообщений: анкетные данные включают: фамилию, имя, отчество испытуемого; возможные ответы испытуемого: 0–нет, 1–да, 2–не знаю. 3. Источники информации: испытуемый.
3	Описать выходную информацию.	1. Перечень и описание выходных сообщений: – степень коммуникабельности (в баллах по 10-ти балльной шкале); – коэффициент интеллекта (в %); – степень ответственности (в баллах по 10-ти балльной шкале). 2. Получатели и назначение выходной информации: получатель выходной информации – менеджер по кадрам; назначение – поддержка принятия решения о приеме на работу.

Решите самостоятельно следующие задания:

Задание 5.1

Сформируйте документ "Постановка комплекса задач для проектирования программного средства" для программы "Калькулятор".

Задание 5.2

Сформируйте документ "Постановка комплекса задач для проектирования программного средства" для игры в Тетрис.

Задание 5.3

Сформируйте документ "Постановка комплекса задач для проектирования программного средства" для программы, автоматизирующей продажу железнодорожных билетов.

Задание 5.4

Сформируйте документ "Постановка комплекса задач для проектирования программного средства" для программы ведения учета выдачи книг в библиотеке.

Задание 5.5

Сформируйте документ "Постановка комплекса задач для проектирования программного средства" для программы формирования ведомости на выдачу стипендии студентам по результатам сдачи сессии.

Рабочий учебник содержит:

понятий – 21
умений – 5
линков – 117

ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ (КУРС 1)

ЮНИТА 2

ТЕХНОЛОГИИ ПРОЕКТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Редактор В.П. Волкова
Оператор компьютерной верстки А.Б. Кондратьева

Изд. лиц. ЛР № 071765 от 07.12.1998 Сдано в печать
НОУ "Современный Гуманитарный Институт"
Уч.-изд. л. 6,56 Усл. печ. л. Тираж Заказ