ЛЕКЦИИ

Лекция 1. Программное обеспечение компьютерных систем

1. Программное обеспечение и его классификация

Под **программным обеспечением** (*Software*) понимается совокупность программ, выполняемых вычислительной системой. К программному обеспечению (ПО) относится также область деятельности по его проектированию и разработке:

- технология проектирования программ (например, нисходящее проектирование, структурное и объектно-ориентированное проектирование и др.);
- методы тестирования программ;
- методы доказательства правильности программ;
- анализ качества работы программ;
- документирование программ;
- разработка и использование программных средств, облегчающих процесс проектирования программного обеспечения, и многое другое.

Программное обеспечение – неотъемлемая часть компьютерной системы. Оно является логическим продолжением технических средств. Сфера применения конкретного компьютера определяется созданным для него ПО. Сам по себе компьютер не обладает знаниями ни в одной области применения. Все эти знания сосредоточены в выполняемых на компьютерах программах. Программное обеспечение современных компьютеров включает миллионы программ — от игровых до научных.

Программное обеспечение по назначению делится
на: **Базовое** (системное) ПО.
Рабочее (прикладное) ПО:
Инструментальное ПО: Средства разработки программного обеспечения; Системы управления базами данных (СУБД) — реляционные (например, DB2, Interbase, Microsoft SQL Server, MySQL, Oracle Database, PostgreSQL), обьектно-ориентированные, иерархические, сетевые.

Коммерческие программы. Большинство программ распространяется коммерческой основе. Такие программы обычно продаются в коробках, содержащих дискеты или компакт-диски, документацию, регистрационную карточку и т. д. Набор компакт-дисков, на котором распространяется программа, называется дистрибутивом.

Бесплатные программы. Существуют программы, распространяемые бесплатно (*freeware*). Типичные каналы их распространения — глобальная электронная сеть Internet, электронные доски объявлений (BBS) и т. д.

Условно -бесплатные программы. Промежуточное положение между бесплатными и коммерческими программами занимают условно-бесплатные программы (*shareware*). Их можно получить и опробовать бесплатно, но для систематического их использования необходимо уплатить разработчикам или распространителям программы определённую (чаще всего небольшую) сум-

му. Часто после этого разработчик высылает регистрационный ключ (комбинации символов), позволяющий задействовать дополнительные возможности программы.

Пиратские копии программ. Многие торговцы продают незаконно изготовленные копии программ. Часто такие программы называются пиратскими, хотя правильнее было бы их называть ворованными. Обычно они распространяются на компакт-дисках без всякой документации, причём на одном диске часто имеется множество (иногда десятки или даже сотни) программ. Часто пиратские компакт-диски содержат не последние, а предварительные выпуски программ.

2. Пакеты прикладных программ

Пакет прикладных программ - комплект программ, предназначенных для решения задач из определённой проблемной области . Обычно применение пакета прикладных программ предполагает наличие специальной документации: лицензионного свидетельства, паспорта, инструкции пользователя и т.п.

Выделяются следующие виды ППП:

1. проблемно-ориентированные. Используются для тех проблемных областей, в которых возможна типизация функций управления, структур данных и алгоритмов обработки. Например, это ППП автоматизации бухучёта, финансовой деятельности, управления персоналом и т.д.;
2. автоматизации проектирования (или САПР). Используются в работе конструкторов и технологов, связанных с разработкой чертежей, схем, диаграмм;
3. общего назначения. Поддерживают компьютерные технологии конечных пользователей и включают текстовые и табличные процессоры, графические редакторы, системы управления базами данных (СУБД);
4. офисные. Обеспечивают организационное управление деятельностью офиса. Включают органайзеры (записные и телефонные книжки, календари, презентации и т.д.), средства распознавания текста;
5. настольные издательские системы – более функционально мощные текстовые процессоры;
6. системы искусственного интеллекта. Используют в работе некоторые принципы обработки информации, свойственные человеку. Включают информационные системы, поддерживающие диалог на естественном языке; экспертные системы, позволяющие давать рекомендации пользователю в различных ситуациях; интеллектуальные пакеты прикладных программ, позволяющие решать прикладные задачи без программирования.

ППП состоит из нескольких программных единиц. Такие программные единицы обычно называют **программными модулями.** Пакет предназначен для решения задач определённого класса. Этот класс задач обычно называют **предметной областью пакета**.

Для реализации выбранных пользователем конкретных действий пакет должен воспринимать от пользователя управляющую информацию. Эта управляющая информация представляется на формальном языке — входном языке пакета. Описание конкретного задания пользователя на входном языке пакета называют **программой на входном языке** *(ПВЯ).*

Решение каждой задачи в пакете сводится к выполнению соответствующего алгоритма. Программные модули пакета, реализующие алгоритмы решения задач, предусмотренных в пакете, будем называть *обрабатывающими модулями.* Обрабатывающие модули выполняют преобразование данных, составляющих информационную базу пакета.

Для того чтобы преобразовать задание пользователя в последовательность вызовов обрабатывающих модулей, в пакет должны входить *управляющие модули***.**

Чтобы обеспечить взаимодействие пакета с пользователем и управляющих модулей пакета с информационной базой и обрабатывающими модулями, в состав пакета включаются *обслуживающие модули***.**

Таким образом, ППП можно рассматривать как объединение входного языка, информационной базы, управляющих, обслуживающих и обрабатывающих программных модулей. Совокупность обрабатывающих модулей часто называют *функциональным наполнением пакета* . Управляющие и обслуживающие модули называются *системной частью пакета*, или системным наполнением пакета.

3. Способы применения ППП

Под *способом применения ППП* будем понимать организацию взаимодействия пользователя с пакетом при решении задач.

Способы применения существующих в настоящее время ППП весьма разнообразны, но можно выделить некоторые типовые режимы , определяемые построением самого пакета и особенностями используемого компьютера и ОС.

Простейший режим с точки зрения построения ППП сводится к использованию отдельных программ пакета как подпрограмм некоторой главной программы, составляемой пользователем на каком-либо языке программирования. В этом случае ППП состоит только из обрабатывающих модулей и

может рассматриваться как расширение библиотеки подпрограмм используемого языка программирования.

Следующий по сложности реализации режим предполагает, что вся управляющая информация для конкретного выполнения пакета передаётся в виде законченной программы на входном языке при запуске пакета. Дальнейшая работа пакета проходит без участия пользователя. Такой режим по аналогии с соответствующим режимом ОС часто называют *пакетным* . Пакетный режим удобен, когда требуется решать много однотипных задач с использованием одной и той же программы на входном языке , когда время, затрачиваемое на решение каждой задачи, достаточно велико, когда программа на входном языке сложна и имеет значительный объём.

Большинство ППП, применяемых на ПК, ориентировано на *диалоговое взаимодействие* с пользователем в ходе решения задач.

Простейший диалоговый режим (вариант диалогового взаимодействия) состоит в том, что пользователь инициирует выполнение пакета, вводит задание в форме программы на входном языке и на этом заканчивает управление выполнением пакета. Фактически этот режим отличается от пакетного только возможностью исправления ошибок в ПВЯ, повторного запуска пакета при неудачах.

Более сложный вариант диалогового режима, называемый также *режимом сопровождения,* предусматривает возможность динамического управления выполнением пакета. Управляющая информация вводится по частям и формируется пользователем в процессе работы с пакетом на основе анализа промежуточных результатов . Такая работа в большинстве случаев более естественна для пользователя, в частности, при использовании пакетов редактирования текстов, при работе с электронными таблицами, при решении многих расчётных задач.

4. Программные средства и продукты

Программные средства – математические средства, с помощью которых решаются задачи автоматизированного получения, обработки, хранения и выдачи информации.

Программный продукт – это совокупность отдельных программных средств, их документации , гарантий качества, рекламных материалов, мер по обучению пользователей, распространению и сопровождению готового программного обеспечения.

Это понятие несколько шире, чем комплекс программ. Кроме собственно программ на носителях информации (например, компакт-дисках) оно включает упаковку, эксплуатационную документацию и лицензионное соглаше-

ние, когда речь идет о программном продукте, который тиражируется.

Программное изделие – программа или логически связанная совокупность программ:

- записанная на носителях данных;
- являющаяся продуктом промышленного производства;
- снабженная программной документацией;

- предназначенная для широкого распространения посредством продажи или методами freeware, shareware или OEM.

Путь от «программ для себя » до программных продуктов достаточно долгий, он связан с изменениями технической и программной среды разработки и эксплуатации программ, с появлением и развитием самостоятельной отрасли – информационного бизнеса, для которой характерны разделение труда фирм – разработчиков программ, их дальнейшая специализация, формирование рынка программных средств и информационных услуг.

Программные продукты могут создаваться как: индивидуальная разработка под заказ; разработка для массового распространения среди пользователей.

При *индивидуальной разработке* фирма-разработчик создаёт оригинальный программный продукт, учитывающий специфику обработки данных для конкретного заказчика. При разработке *для массового распространения* фирма-разработчик, с одной стороны, должна обеспечить универсальность выполняемых функций обработки данных , с другой стороны, гибкость и настраиваемость программного продукта на условия конкретного применения. Отличительной особенностью программных продуктов должна быть их *системность* – функциональная полнота и законченность реализуемых функций обработки, которые применяются в совокупности.

Программный продукт разрабатывается на основе промышленной технологии выполнения проектных работ с применением современных инструментальных средств программирования. Специфика заключается в уникальности процесса разработки алгоритмов и программ, зависящего от характера обработки информации и используемых инструментальных средств. На создание программных продуктов затрачиваются значительные ресурсы – трудовые, материальные, финансовые; требуется высокая квалификация разработчиков.

Как правило, программные продукты требуют сопровождения, которое осуществляется специализированными фирмами – распространителями программ (дистрибьюторами), реже – фирмами-разработчиками. Сопровождение

программ массового применения сопряжено с большими трудозатратами – исправление обнаруженных ошибок, создание новых версий программ и т.п.

Сопровождение программного продукта – поддержка работоспособности программного продукта, переход на его новые версии, внесение изменений, исправление обнаруженных ошибок и т.п.

Программные продукты в отличие от традиционных программных изделий не имеют строго регламентированного набора качественных характеристик, задаваемых при создании программ, либо эти характеристики невозможно заранее точно указать или оценить, т.к. одни и те же функции обработки, обеспечиваемые программным средством, могут иметь различную глубину проработки. Даже время и затраты на разработку программных продуктов не могут быть определены с большой степенью точности заранее.

5. Рынок программных продуктов

Рынок ПП — система экономических, правовых , организационных отношений по торговле программными продуктами на коммерческой основе. На этом рынке действуют:

- поставщики ПП;
- потребители ПП;
- посредники.

В структуре рынка ПП можно выделить следующие компоненты:

1. Технологическая составляющая.
2. Нормативно-правовая составляющая.
3. Информационная составляющая.
4. Организационная составляющая.

В результате развития рынка ИП и ИУ формируется инфраструктура рынка ПП — совокупность секторов, каждый из которых объединяет группы, предлагающие однородные продукты.

В условиях существования рынка программных продуктов важными характеристиками являются:
- стоимость,
- количество продаж;
- время нахождения на рынке (длительность продаж);
- известность фирмы-разработчика и программы;
- наличие программных продуктов аналогичного назначения.

Программные продукты массового распространения продаются по ценам, которые учитывают спрос и конъюнктуру рынка (наличие и цены про-

грамм-конкурентов). Большое значение имеет проводимый фирмой **маркетинг**, который включает:
- формирование политики цен для завоевания рынка;
- широкую рекламную кампанию программного продукта;
- создание торговой сети для реализации программного продукта (так называемые дилерские и дистрибьютерные центры);
- обеспечение сопровождения и гарантийного обслуживания пользователей программного продукта, создание горячей линии (оперативный ответ на возникающие в процессе эксплуатации программных продуктов вопросы);
- обучение пользователей программного продукта.

Спецификой программных продуктов (в отличие от большинства промышленных изделий) является также и то, что их эксплуатация должна выполняться на правовой основе – **лицензионные соглашения** между разработчиком и пользователями с соблюдением авторских прав разработчиков программных продуктов.

Приобретение программного продукта — это покупка ***лицензии*** – права на его использование. Условия использования любого программного продукта описаны в *лицензионном соглашении,* которое представляет собой договор между производителем программного продукта и пользователем программного обеспечения. Для разных пользователей (индивидуальных покупателей, организаций разного масштаба, учебных заведений и правительственных учреждений) могут быть установлены различные условия приобретения программного обеспечения.

Каждый пользователь программного продукта должен иметь лицензию на него. Лицензия должна быть закуплена для каждого компьютера, на котором установлен или используется загружаемый через сеть программный продукт. Договор между пользователем и производителем не подписывается: считается , что покупатель соглашается с условиями лицензионного соглашения, если он вскрывает *дистрибутив* — упаковку с дискетами или компакт-диском. Это так называемая *«оберточная» лицензия,* предусмотренная Законом «О правовой охране программ для ЭВМ и баз данных» от 23 сентября 1992 года.

Программное обеспечение на компьютере *находится «в пользовании»*, когда оно помещено в постоянную память (обычно на жёсткий диск, но возможно и на компакт-диск или другое устройство для хранения информации) или загружено в оперативную память. В компьютерной сети продукт может использоваться одним из двух способов: запуск программного обеспечения с локального жёсткого диска рабочей станции или установка продукта только на сервер сети и запуск программного обеспечения с сервера. Вне зависимости от того, как используется продукт в сети (с сервера или с локального рабочего места), каждый пользователь должен обладать лицензией на право

использования этого продукта. Только такой вариант использования программного продукта считается законным.

Существует несколько вариантов приобретения лицензии, т. е. права использовать программный продукт. Наиболее известный и распространённый путь — покупка коробки с программным продуктом. Коробка содержит лицензионное соглашение, регистрационную карточку, дистрибутив с программным продуктом и документацию. Это основные компоненты, которые входят в коробку с программным продуктом, предназначенную для новых пользователей, т. е. для тех, кто ранее не использовал данный программный продукт и приобрел его впервые. Если появляется необходимость в использовании этого программного продукта на других компьютерах , недостаточно приобрести одну коробку. В этом случае многие поставщики программного обеспечения предлагают приобрести только лицензию — конверт, содержащий лицензионное соглашение, цена которого ниже, чем цена коробки.

Лекция 2. Разработка ПС

1. Стадии разработки ПО, регламентированные ГОСТами.

В нашей стране жизненный цикл разработки ПО установлен стандартом ГОСТ 19.102-77 «Стадии разработки программ и программной документации» и содержит следующие стадии и этапы:

1. Техническое задание (ТЗ).
2. Эскизный проект (ЭП).
3. Технический проект (ТП).
4. Рабочий проект (РП).
5. Внедрение.

Техническое задание. На стадии Техническое задание выполняются следующие работы, входящие в состав соответствующих этапов.

1. Обоснование необходимости разработки программ: постановка задачи; сбор исходных материалов; выбор и обоснование критериев эффективности и качества; обоснование необходимости проведения НИР.

2. Выполнение научно-исследовательских работ: определение структуры входных и выходных данных; предварительный выбор методов решения задач; обоснование целесообразности применения ранее разработанных программ; определение требований к техническим средствам; обоснование принципиальной возможности решения поставленных задач.

3. Разработка и утверждение технического задания: определение требований к программе; разработка технико-экономического обоснования раз-

работки программы; определение стадий, этапов и сроков разработки программы и документации на нее; выбор языков программирования; определение необходимости проведения НИР на последующих стадиях; согласование и утверждение ТЗ.

Результатом выполнения данной стадии является **техническое задание**, оформленное в соответствии с ГОСТ 19.105- 78 (изм. 09.1981) «Общие требования к программным документам» и ГОСТ 19.106-78 «Общие требования к программным документам, выполненным печатным способом на листах формата 11 и 12».

Эскизный проект. Конкретное содержание работ стадии эскизного проекта и их объем определяет степень сложности разрабатываемого ПО. Результатом выполнения данной стадии является полное описание архитектуры ПО. Как правило, это описание делается на нескольких уровнях иерархии. На верхнем уровне детализации выделяются основные подсистемы, которым присваиваются имена , устанавливаются связи между подсистемами , их функции, получаемые путем декомпозиции предполагаемых функций ПО. Затем процедура декомпозиции выполняется для каждой подсистемы, выделяются модули, составляющие данную подсистему. В конечном итоге, получается иерархически организованная система, состоящая из уровней, каждый из которых представляет собой совокупность взаимосвязанных модулей. Единицы, выделяемые на различных иерархических уровнях функциональной архитектуры системы, определяются по усмотрению разработчика. Стандарты ЕСПД различают программные единицы только с точки зрения их документирования.

Результаты эскизного проекта отображаются в документе Пояснительная записка к эскизному проекту, оформленному в соответствии с ГОСТ 19.105-78 и ГОСТ 19.404-79. После утверждения пояснительной записки она становится программным документом, правила дублирования, учета, хранения которого определяется ГОСТ 19.601-78 «Общие правила дублирования, обращения, учета и хранения» и ГОСТ 19.602-78 «Правила дублирования, учета и хранения программных документов, выполненных печатным способом». Последующие стадии и этапы разработки ПО могут выявить необходимость внесения изменений в ЭП. Эти изменения должны быть отражены в пояснительной записке в соответствии с ГОСТ 19.603-78 «Общие правила внесения изменений в программные документы» и ГОСТ 19.602-78 «Правила внесения изменений в программные документы, выполненные печатным способом».

Технический проект. Содержанием работ на этой стадии является проектирование структуры ПО. Результатом – реализующий заданный и утвержденный в техническом задании комплекс программ как иерархическая структура программных модулей, заданных своими функциональными спецификациями. Форма представления результата – *пояснительная записка* к

техническому проекту согласно ГОСТ 19.105 -78, ГОСТ 19.404-79. Разработка структуры ПО заключается в выделении всех программных компонентов по функциональным признакам, определение функциональных спецификаций модулей и уточнение внешних функциональных спецификаций, структуры входных и выходных данных, определении операционной среды, языковых средств и конфигурации аппаратных средств. Спецификации модулей являются внешними характеристиками и содержат все сведения, необходимые вызывающим модулям. На последующих стадиях разработки спецификации оформляются в виде комментариев в начале текста исходной программы модуля. На данной стадии спецификации оформляются в виде комментария на принятом в организации, занимающейся разработкой ПО, языке спецификаций

Оформление пояснительной записки и ведомости технического проекта ПО осуществляется в соответствии с ГОСТ 19.105-78, ГОСТ 19.404-79 и ГОСТ 2.106-68 ЕСКД «Текстовые документы».

Рабочий проект. Содержанием работ на этой стадии является описание ПО на выбранном проблемно-ориентированном языке (кодирование), отладка, разработка, согласование и утверждение порядка и методики испытаний, разработка программных документов, проведение тестирования, корректировка программ и программной документации по результатам тестирования, проведение приемо-сдаточных испытаний. Результат – ПО в форме программной документации, в форме документации на ПО или в форме программного изделия.

2. Качество ПО

Качество программного обеспечения – способность программного продукта подтвердить свою спецификацию при условии, что спецификация ориентирована на характеристики, которые желает получить пользователь.

Одной из важнейших проблем обеспечения качества программных средств является формализация характеристик качества и методология их оценки. Основой регламентирования показателей качества программных средств ранее являлся международный стандарт ISO 9126:1991 (ГОСТ Р ИСО / МЭК 9126-93) «Информационная технология. Оценка программного продукта. Характеристики качества и руководство по их применению».

Методологии и стандартизации оценки характеристик качества готовых программных средств и их компонентов (программного продукта) на различных этапах жизненного цикла посвящен международный стандарт ISO 14598, состоящий из шести частей. Рекомендуется следующая общая схема процессов оценки характеристик качества программ:

- установка исходных требований для оценки – определение целей испытаний, идентификация типа метрик программного средства, выделение адекватных показателей и требуемых значений атрибутов качества;
- селекция метрик качества, установление рейтингов и уровней приоритета метрик субхарактеристик и атрибутов, выделение критериев для проведения экспертиз и измерений;
- планирование и проектирование процессов оценки характеристик и атрибутов качества в жизненном цикле программного средства;
- выполнение измерений для оценки, сравнение результатов с критериями и требованиями, обобщение и оценка результатов.

Функциональная пригодность – наиболее неопределенная и объективно трудно оцениваемая субхарактеристика программного средства. Области применения, номенклатура и функции комплексов программ охватывают столь разнообразные сферы деятельности человека , что невозможно выделить и унифицировать небольшое число атрибутов для оценки и сравнения этой субхарактеристики в различных комплексах программ.

Оценка корректности программных средств состоит в формальном определении степени соответствия комплекса реализованных программ исходным требованиям контракта, технического задания и спецификаций на программное средство и его компоненты. Путем верификации должно быть определено соответствие исходным требованиям всей совокупности к компонентов комплекса программ, вплоть до модулей и текстов программ и описаний данных.

Оценка способности к взаимодействию состоит в определении качества совместной работы компонентов программных средств и баз данных с другими прикладными системами и компонентами на различных вычислительных платформах, а также взаимодействия с пользователями в стиле, удобном для перехода от одной вычислительной системы к другой с подобными функциями.

Оценка защищенности программных средств включает определение полноты использования доступных методов и средств защиты программного средства от потенциальных угроз и достигнутой при этом безопасности функционирования информационной системы. Наиболее широко и детально методологические и системные задачи оценки комплексной защиты информационных систем изложены в трех частях стандарта ISO 15408:1999-1--3 «Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий».

Оценка надежности – измерение количественных метрик атрибутов субхарактеристик в использовании: завершенности, устойчивости к дефектам, восстанавливаемости и доступности/готовности.

Потребность в ресурсах памяти и производительности компьютера в процессе решения задач значительно изменяется в зависимости от состава и объема исходных данных. Для корректного определения предельной пропускной способности информационной системы с данным программным средством нужно измерить экстремальные и средние значения длительностей исполнения функциональных групп программ и маршруты, на которых они достигаются. Если предварительно в процессе проектирования производительность компьютера не оценивалась, то, скорее всего, понадобится большая доработка или даже замена компьютера на более быстродействующий.

Оценка практичности программных средств проводится экспертами и включает определение понятности, простоты использования, изучаемости и привлекательности программного средства. В основном это качественная (и субъективная) оценка в баллах, однако некоторые атрибуты можно оценить количественно по трудоемкости и длительности выполнения операций при использовании программного средства, а также по объему документации, необходимой для их изучения.

Сопровождаемость можно оценивать полнотой и достоверностью документации о состояниях программного средства 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.

Оценка мобильности – качественное определение экспертами адаптируемости, простоты установки, совместимости и замещаемости программ, выражаемое в баллах. Количественно эту характеристику программного средства и совокупность ее атрибутов можно (и целесообразно) оценить в экономических показателях: стоимости, трудоемкости и длительности реализации процедур переноса на иные платформы определенной совокупности программ и данных.

Выбор характеристик и оценка качества программных средств - лишь одна из задач в области обеспечения качества продукции, выпускаемой компаниями - разработчиками ПО. Комплексное решение задач обеспечения качества программных средств предполагает разработку и внедрение той или иной системы управления качеством. В мировой практике наибольшее распространение получила система, основанная на международных стандартах серии ISO 9000, включающей десяток с лишним документов, в том числе стандарт, регламентирующий обеспечение качества ПО (ISO 9000/3). Эти стандарты должны служить руководством для ведущих специалистов компаний, разрабатывающих ПО на заказ.

Определения характеристик и субхарактеристик качества (ISO 9126-1)

Функциональные возможности – способность программного средства обеспечивать решение задач, удовлетворяющих сформулированные потребности заказчиков и пользователей при применении комплекса программ в заданных условиях.

Функциональная пригодность – набор и описания субхарактеристики и ее атрибутов, определяющие назначение, номенклатуру, основные, необходимые и достаточные функции программного средства, соответствующие техническому заданию и спецификациям требований заказчика или потенциального пользователя.

Правильность (корректность) – способность программного средства обеспечивать правильные или приемлемые для пользователя результаты и внешние эффекты.

Способность к взаимодействию – свойство программных средств и их компонентов взаимодействовать с одной или большим числом компонентов внутренней и внешней среды.

Защищенность – способность компонентов программного средства защищать программы и информацию от любых негативных воздействий.

Надежность – обеспечение комплексом программ достаточно низкой вероятности отказа в процессе функционирования программного средства в реальном времени.

Эффективность – свойства программного средства, обеспечивающие требуемую производительность решения функциональных задач, с учетом количества используемых вычислительных ресурсов в установленных условиях.

Практичность (применимость) – свойства программного средства, обусловливающие сложность его понимания, изучения и использования, а также привлекательность для квалифицированных пользователей при применении в указанных условиях.

Сопровождаемость – приспособленность программного средства к модификации и изменению конфигурации и функций.

Мобильность – подготовленность программного средства к переносу из одной аппаратно-операционной среды в другую.

3. Надёжность программного обеспечения

Надежность программного обеспечения – способность программного продукта безотказно выполнять определенные функции при заданных условиях в течение заданного периода времени с достаточно большой **вероятностью**. *Степень надежности* характеризуется **вероятностью** работы программного продукта без отказа в течение определенного периода времени.

Случайное изменение исходных данных и накопленной при обработке информации, множество условных переходов в программе создают такое огромное количество различных маршрутов исполнения программы, что их нельзя протестировать полностью из-за ограниченного времени отладки и испытаний. Источниками ненадёжности являются непроверенные сочетания исходных данных, при которых отлаженные программы дают неверные результаты и отказы.

Отказ – это нарушение работоспособности программного изделия, являющееся следствием таких явлений, как нарушения кодов записи программ в памяти, стирания или искажения данных в оперативной или долговременной памяти, нарушения нормального хода вычислительного процесса.

Сбой – это самоустраняющийся отказ , не требующий внешнего вмешательства. Основное различие между сбоем и отказом – по временному показателю длительности восстановления. Если длительность восстановления больше какого-то порогового значения, то аномалию в работе программы относят к отказам, иначе – к сбоям.

Понятие правильной (корректной) программы рассматривается статически вне временного функционирования. Неправильность программы определяется вероятностью совмещения следующих событий: попадания исходных данных в область непроверенных при отладке и испытаниях; проявления ошибки в программе при обработке таких данных.

Правильность программы не зависит от динамики функционирования в реальном времени.

Надёжная программа должна обеспечивать низкую **вероятность** отказа в процессе функционирования. Количество ошибок в программе, как оказалось, не имеет никакого отношения к ее надежности:

1. Число ошибок в программе – величина «ненаблюдаемая», наблюдаются не сами ошибки, а результат их проявления.

2. Неверное срабатывание программы может быть следствием не одной, а сразу нескольких ошибок.

3. Ошибки могут компенсировать друг друга, так что после исправления какой-то одной ошибки программа может начать «работать хуже».

4. Надежность характеризует частоту проявления ошибок, но не их количество; в то же время хорошо известно, что ошибки проявляются с разной частотой: некоторые ошибки остаются не выявленными после многих месяцев и даже лет эксплуатации, но, с другой стороны одна единственная ошибка может привести к неверному срабатыванию программы при любых исходных данных, т.е. к нулевой надежности.

Тема 3. Разработка требований и внешнее проектирование ПО

1. Общая схема процесса создания ПО

Процесс создания программ можно представить как последовательность действий:

Постановка задачи *(problem definition)* – это точная формулировка решения задачи на компьютере с описанием входной и выходной информации.

Постановка задачи - обобщенный термин, который означает определенность содержательной стороны обработки данных. Постановка задачи связана с конкретизацией основ-параметров ее реализации, определением источников и структурой входной и выходной информации, востребуемой пользователем.

К основным характеристикам функциональных задач, уточняемым в процессе ее формализованной постановки, относятся:

-цель или назначение задачи, ее место и связи с другими задачами;
-условия решения задачи с использованием средств вычислительной техники;
-содержание функций обработки входной информации при решении задачи;
-требования к периодичности решения задачи;
-ограничения по срокам и точности выходной информации;
-состав и форма представления выходной информации ; -
источники входной информации для решения задачи;

-пользователи задачи (кто осуществляет ее решение и пользуется результатами решение и пользуется результатами решения).

Обычно постановка задач выполняется в едином комплексе работ по созданию структуры внутри машинной базы данных, проектированию форм и маршрутов движения документов, изменению организации управления в рамках предметной области.

Алгоритм – система точно сформулированных правил, определяющая процесс преобразования допустимых исходных данных (входной информации) в желаемый результат (выходную информацию) за конечное число шагов.

Алгоритм решения задачи имеет ряд обязательных свойств:

-*дискретность* – разбиение процесса обработки информации на более простые этапы (шаги), выполнение которых компьютером или человеком не вызывав затруднений;
-*определённость* алгоритма – однозначность выполнения каждого отдельного шага преобразования информации;
-*выполнимость* – конечность действий алгоритма решения задач, позволяющая получить желаемый результат при допустимых исходных данных за конечное число шагов;
-*массовость* – пригодность алгоритма для решения определенного класса задач.

В алгоритме отражаются логика и способ формирования результатов решения с указанием необходимых расчётных формул, логических условий, соотношений для контроля достоверности выходных результатов. В алгоритме обязательно должны быть предусмотрены все ситуации, которые могут возникнуть в процессе решения комплекса задач.

Алгоритм решения комплекса задач и его программная реализация тесно взаимосвязаны. Специфика применяемых методов проектирования алгоритмов и используемых при этом инструментальных средств разработки программ может повлиять на форму представления и содержание алгоритма обработки данных.

Примечание. Для решения задач могут использоваться алгоритмы, заложенные в готовых программных продуктах - пакетах прикладных программ (ППП) функционального назначения.

Программирование *(programming)* – теоретическая и практическая деятельность, связанная с созданием программ.

Программа – алгоритм решения задачи, формализованный на языке программирования.

В связи с ростом потребности в разнообразных программах обработки данных весьма актуален вопрос применения эффективных технологий программирования и их перевода на промышленную основу. Это означает:

- стандартизованность, тиражируемость и воспроизведение различными разработчиками методов программирования;
- внедрение прогрессивных инструментальных средств разработки программ;
- использование специальных методов и приемов организации работ по разработке программ.

Основная категория специалистов, занятых разработкой программ, - это ***программисты*** (programmer). Программисты неоднородны по уровню квалификации , а также по характеру своей деятельности. Наиболее часто программисты делятся на системных и прикладных.

- ***Системный программист*** *(system / software programmer, toolsmilh)* занимается разработкой, эксплуатацией и сопровождением *системного* программного обеспечения, поддерживающего работоспособность компьютера и создающего среду для выполнения программ, обеспечивающих реализацию функциональных задач.

- ***Прикладной программист*** *(application programmer)* осуществляет разработку и отладку программ для решения функциональных задач.

В условиях создания больших по масштабам и функциям обработки программ существует квалификация – ***программист-аналитик*** *(programmer-analyst),* который анализирует и проектирует комплекс взаимосвязанных программ для реализации функций предметной области.

В процессе создания программ на начальной стадии работ участвуют и специалисты - ***постановщики задач.***

Большинство информационных систем основано на работе с базами данных (БД). Если база данных является интегрированной, обеспечивающей работу с данными многих приложений, возникает проблема организационной поддержки базы данных, которая *выполняется* ***администратором базы данных.***

Основным потребителем программ служит ***конечный пользователь*** *(end user),* который, как правило, относится к категории пользователей-непрограммистов. Конечный пользователь не является специалистом в области программирования, т. е. не владеет методами и технологией проектирования и создания программ, но имеет элементарные знания и навыки рабо-

ты с вычислительной техникой. Такая квалификационная характеристика пользователя программного обеспечения в значительной степени влияет на спецификацию требований к создаваемым программам, интерфейсам , формам машинных документов, технологии решения задач на компьютере.

2. Разработка требований к ПО

Главной особенностью требований является то, что они отражают нужды организации-пользователя.

В связи с этим можно выделить три группы программных проектов: управляемые пользователем, утверждаемые пользователем и независимые от пользователя.

Для *проектов первой группы* требования формулируются организацией-пользователем. Для *проектов второй группы* совокупность требований разрабатывается проектировщиками или проектировщиками совместно с организацией-пользователем, которая утверждает эти требования, а иногда и внешние спецификации. Для *проектов третьей группы* требования полностью определяются и утверждаются организацией-разработчиком, которая несет полную ответственность за работу.

Целью сформулированных требований является отражение потребностей пользователей в конкретном программном изделии. Поэтому наиболее оптимальной является совместная работа проектировщиков и пользователей по выработке требований.

Можно установить *две фазы в выработке требований*. Первой является фаза планирования, на которой в процессе взаимодействия пользователей и разработчиков определяется реализуемость, устанавливаются цели, оцениваются затраты и обеспечивается ориентация для разработки проекта. В результате получают набор требований, основанный на определенных начальных условиях. Второй фазой является фаза выработки требований пользователя. На этой фазе вырабатываются требования к входным данным, информационным потокам, выходным данным, документации, среде, вычислительным ресурсам. Получаемая на этом этапе информация ориентирована на установление критериев для выходных результатов, задает функциональные возможности ПИ и ограничивает среду использования ПИ. Именно на этой фазе возникают детализированные требования к ПИ.

Результатом работы по выработке требований обычно является соответствующий документ, который должен быть:

- достаточным для идентификации целей ПИ, его среды, преимуществ и недостатков ПИ для пользователя, состава и конфигурации ресурсов для его работы;
- достаточно полным, чтобы в последующем при разработке исключить серьезные модификации и пересмотр требований;
- достаточным для просмотра и утверждения администрацией на основе его реализуемости в соответствии с выбранными критериями.

Требования являются определенными в том объеме, в котором они фиксируются в документации.

3. Цели разработки ПО

Цели ПИ, рассматриваемые с точки зрения пользователей, обычно включают следующую информацию.

1. *Краткое описание. В* нем кратко определяется общее назначение разрабатываемого ПИ и его функций.
2. *Определение пользователя.* Описывается круг возможных пользователей, характеризуются специфические особенности отдельных групп пользователей.
3. *Подробное описание функциональных задач.* Оно характеризует однозначное восприятие требований пользователей-разработчиков.
4. *Документация.* Определяются типы документации и предполагаемый круг читателей для каждого типа.
5. *Эффективность.* Описываются все цели, касающиеся производительности: временные характеристики, пропускная способность, использование ресурсов и т.д.
6. *Совместимость.* Указываются стандарты, которым необходимо следовать в процессе разработки, а также другие программные продукты, с которыми разрабатываемое ПИ должно быть совместимо.
7. *Конфигурация.* Определяются различные конфигурации технических и программных средств, в среде которых может работать проектируемое ПИ.
8. *Безопасность.* Формируются цели в отношении обеспечения безопасности ПИ.
9. *Обслуживание.* Описываются цели по затратам и времени исправления ошибок, а также функции для достижения этих целей.
10. *Установка.* Описываются методы и средства настройки ПИ на конкретные условия эксплуатации.
11. *Надежность.* Цели по достижению надежности в значительной мере зависят от конкретного типа разрабатываемого ПИ. Однако можно определить некоторые общие вопросы, которые должны быть рассмотрены:

 среднее время наработки на сбой для каждого вида сбоя (ПИ, пользователь, отдельная функция) и степень важности сбоя;

 среднее время восстановления ПИ после сбоя;

 цели по числу ошибок ПИ по категориям сложности и времени обнаружения;

 последствия сбоев системы и наиболее важных функций;

 допустимый объем данных, утрачиваемых во время сбоя, и уровень обеспечения безопасности;

 функции, необходимые для обнаружения и исправления ошибок, а также обеспечение устойчивости к ним;

 возможности обнаружения ошибок пользователя и аппаратуры, а также восстановления работоспособности.

4. Разработка внешних спецификаций проекта

Внешнее проектирование — это процесс описания планируемого поведения разрабатываемого ПИ с точки зрения потенциальных пользователей. Целью этого процесса является конкретизация внешних взаимодействий будущего ПИ без детализации внутреннего устройства. Внешний проект представляет собой внешние спецификации ПИ, предназначенные для различных групп специалистов — пользователей и разработчиков.

При разработке внешних спецификаций необходимо стремиться к концептуальной целостности проекта. Концептуальная целостность представляет собой меру единообразия способа взаимодействия с пользователем. Система, не обладающая концептуальной целостностью, имеет слишком сложное взаимодействие с пользователем и излишне сложную структуру. Система с концептуальной целостностью должна иметь следующую характеристику: все средства, *доступные одному пользователю*, должны быть *доступны и другим пользователям*, т.е. интерфейсы пользователей должны быть идентичны.

При разработке внешних интерфейсов пользователя проектировщик должен решить три проблемы:

1) доведение до минимума ошибок пользователя;
2) обнаружение ошибок пользователя в случае их возникновения;
3) доведение до минимума сложности разрабатываемого ПИ.

Обычно внешние спецификации представляют собой объемистый документ. Поэтому для упрощения процесса его разработки применяют иерархическую организацию. Разработка внешних спецификаций разбивается на две части: предварительный внешний проект и детальный внешний проект.

Предварительный внешний проект содержит описание основных компонентов, затем компонентов, из которых состоят эти основные компоненты, и далее — внешних функций (функций пользователя), составляющих отдельные компоненты проекта. Причем предварительный внешний проект содержит все функции пользователя, но их точный синтаксис, семантика, выходные результаты остаются неопределенными. При этом в продолжительном процессе внешнего проектирования становятся возможными проверка правильности промежуточного уровня проекта и сопоставление его с поставленными целями. Кроме того, появляется контрольная точка для руководства проектом.

Детальный внешний проект каждой функции пользователя должен включать следующую информацию:

1. *Описание входных данных*. Точное описание синтаксиса (формат, допустимые значения, области изменения) и семантики всех данных, вводимых пользователем.

2. *Описание выходных данных.* Точное описание всех результатов функции (ответы на запрос терминала, сообщения об ошибках, контрольные сигналы, отчеты). Описание функциональной связи между входными и выходными данными, чтобы специалист, читающий спецификации, мог представить себе выходные данные, получаемые из любой комбинации входных данных, как правильной, так и неверной.

3. *Преобразование* системы. Многие внешние функции не только порождают выходные данные, но и изменяют состояние системы. Так как рассматриваются внешние спецификации, эти изменения должны быть описаны с точки зрения пользователя.

4. *Характеристики надежности.* Здесь описывается влияние всех возможных отказов функций на систему, файлы пользователя.

5. *Эффективность*. Описание всех ограничений, которые накладываются на эффективность функции (например, затрачиваемое время, занимаемая память) .

6. *Замечания по программированию.* Внешние спецификации должны описывать функции с точки зрения пользователя и избегать ограничений на внутреннее устройство. Данный раздел является необязательным.

При завершении этапа внешнего проектирования, т.е. с момента получения детального внешнего проекта, заканчивается рассмотрение взаимодействия пользователя и ПИ. Дальнейшие действия по проектированию должны быть сосредоточены на внутренней структуре ПИ. В связи с этим возникает вопрос проверки внешних спецификаций на полноту и точность. На этом этапе очень важно обнаружить как можно больше ошибок, так как их обнаружение и внесение изменений наиболее легко выполнить, и это обходится значительно дешевле, чем на более поздних этапах разработки ПИ.

В процессе работы над проектом возникает необходимость внесения изменений, что связано с обнаружением ошибок и с изменением самих требований. Причем изменения требований — объективный фактор разработки ПИ, следствие технического прогресса, социальных перемен, изменений в психологии людей, предлагаемых улучшений и т.д.

Чтобы изменения в проекте не приводили к дополнительным ошибкам, следует внимательно относиться к работе с изменениями и соблюдать следующие правила.

1. Во время всего периода работы над проектом поддерживать документацию на уровне последних решений.

2. Фиксировать результаты каждого этапа процесса проектирования. Требования и цели фиксируются после их утверждения, внешние спецификации — после успешного завершения проверки их правильности. Изменения должны также проходить формальную процедуру утверждения.

3. Проверять каждое внесенное изменение до такой степени, до которой проверялось исходное решение.

4. Контролировать, чтобы нужные изменения были сделаны на всех уровнях разработки ПИ.

Лекция 4. Проектирование и разработка интерфейса ПО

1. Основы построения интерфейсов

Пользовательский интерфейс является своеобразным коммуникационным каналом, по которому осуществляется взаимодействие пользователя и компьютера.

Лучший пользовательский интерфейс — это такой интерфейс, которому пользователь не должен уделять много внимания, почти не замечать его. Пользователь просто работает, вместо того, чтобы размышлять, какую кнопку нажать или где щелкнуть мышью. Такой интерфейс называют *прозрачным* — пользователь как бы смотрит сквозь него на свою работу.

Чтобы создать эффективный интерфейс, который делал бы работу с программой приятной, нужно понимать, какие задачи будут решать пользователи с помощью данной программы и какие требования к интерфейсу могут возникнуть у пользователей.

Если говорить о самых **общих принципах проектирования пользовательских интерфейсов**, то можно назвать три основных положения:

1. Программа должна помогать выполнить задачу, а не становиться этой задачей.
2. При работе с программой пользователь не должен ощущать себя дураком.
3. Программа должна работать так, чтобы пользователь не считал компьютер дураком.

Первый принцип — это уже упоминавшаяся выше прозрачность интерфейса. Интерфейс должен быть легким для освоения и не создавать перед пользователем преграду, которую он должен будет преодолеть, чтобы приступить к работе.

Второй принцип часто нарушают те авторы программ, которые слишком недооценивают умственные способности пользователей.

И, наконец, **третий принцип** — несмотря на стремительное развитие информационных технологий, многие компьютерные программы все еще имеют примитивный искусственный интеллект. Они прерывают работу пользователя глупыми вопросами и выводят на экран бессмысленные сообщения, повергая его в недоумение в самых простых ситуациях. В результате многие люди, работающие с компьютерами, раздраженно восклицают: «Как мне надоела эта тупая машина!».

2. Графический интерфейс пользователя

Графический интерфейс пользователя (Graphics User Interface - GUI) - ГИЛ является обязательным компонентом большинства современных программных продуктов, ориентированных на работу конечного пользователя. К графическому интерфейсу пользователя предъявляются высокие требования как с чисто инженерной, так и с художественной стороны разработки, при его разработке ориентируются на возможности человека.

Наиболее часто графический интерфейс реализуется в интерактивном режиме работы пользователя для программных продуктов, функционирующих в среде Windows, и строится в виде системы спускающихся *меню* с использованием в качестве средства манипуляции мыши и клавиатуры. Работа пользователя осуществляется с *экранными формами,* содержащими *объекты управления, панели инструментов* с *пиктограммами* режимов и команд обработки. Стандартный графический интерфейс пользователя должен отвечать ряду требований:

- поддерживать информационную технологию работы пользователя с программным продуктом - содержать привычные и понятные пользователю пункты меню, соответствующие функциям обработки, расположенные в естественной последовательности использования;
- ориентироваться на конечного пользователя, который общается с программой на *внешнем* уровне взаимодействия;
- удовлетворять правилу "шести" - в одну линейку меню включать не более 6 понятий, каждое из которых содержит не более 6 опций;

Формы – это строительные блоки интерфейса пользователя . Хороший дизайн форм включает нечто большее, чем просто добавление элементов управления и программирование процедур обработки событии. Чтобы создать хорошо спроектированную форму, нужно уяснить ее назначение, способ и время использования, а также ее связи с другими элементами программы. Кроме того в приложении может находиться несколько форм, каждая из которых будет отображаться по мере необходимости.

Формы делятся на два вида: формы ввода и справочники. В формах ввода создаются и редактируются документы (приемные акты, накладные отгрузки, платежные поручения и т.д.). В справочниках вводится справочная информация.

Форма ввода состоит из одного или нескольких блоков, как правило, в виде таблиц. Если невозможно разместить все блоки в одном окне, они занимают несколько окон. Таблица (блок формы ввода) состоит из последовательности строк, расположенных друг под другом. В таблице может быть одна или несколько колонок. В каждой колонке располагаются однотипные по смыслу данные.

Пересечение одной колонки и одной строки называется *полем*. Поля, расположенные в одной строке, называются *записью*. Если полей в таблице много, то они могут быть расположены справа или снизу от нее. При перемещении с записи на запись эти поля обновляются. Над верхней строкой таблицы обычно находятся заголовки - названия каждой колонки. Текущая строка таблицы выделяется серым цветом.

Поля могут содержать: a) произвольную информацию предопределенного типа (текст, число, дата); b) предопределенные данные из списка значений. Списки значений формируются в соответствующих справочниках. Таким образом, справочник – это специальная форма ввода, предназначенная для создания списка значений какого-либо поля.

Если интерфейс пользователя должен содержать несколько форм, вам предстоит принять самое важное решение: какой использовать вид интерфейса – однодокументный (SDI) или многодокументный (MDI) . В SDl-приложениях окна форм появляются совершенно независимо друг от друга. Однако, не имеет значения какой тип интерфейса SDI или MD1 выбран; взаимодействие пользователя с формами происходит одинаково - посредством обработки событий, поступающих от элементов управления формы.

3. Диалоговый режим

Большинство программных продуктов, особенно прикладного характера, ориентированных на конечного пользователя, работают в *диалоговом режиме* взаимодействия с пользователем таким образом, что ведется обмен *сообщениями,* влияющими на обработку данных.

В диалоговом режиме под воздействием пользователя осуществляются запуск функций (методов) обработки, изменение свойств объектов, производится настройка параметров выдачи информации на печать и т.п.

Системы, поддерживающие диалоговые процессы, классифицируются на:

- системы с *жестким сценарием диалога* - стандартизированное представление информации обмена;
- *дескрипторные системы* - формат ключевых слов сообщений;
- *тезаурусные системы* - семантическая сеть дескрипторов, образующих словарь системы (аналог - гипертекстовые системы);
- системы с *языком деловой прозы* - представление сообщений на языке, естественном для профессионального пользования.

Наиболее просты для реализации и распространены диалоговые системы с жестким сценарием диалога, которые предоставлены в виде:

- меню- диалог инициируется программой; пользователю предлагается выбор альтернативы функций обработки из фиксированного перечня; предоставляемое меню может быть иерархическим и содержать вложенные подменю следующего уровня;

- действия запрос-ответ - фиксирован перечень возможных значений, выбираемых из списка, или ответы типа *Да/Нет;*

- запрос по формату - с помощью ключевых слов, фраз или путем заполнения экранной формы с регламентированным по составу и структуре набором реквизитов осуществляется подготовка сообщений.

Диалоговый процесс управляется согласно созданному сценарию, для которого определяются:

- точки (момент, условие) начала диалога;
- инициатор диалога - человек или программный продукт;
- параметры и содержание диалога - сообщения, состав и структура меню, экранные формы и т.п.;
- реакция программного продукта на завершение диалога.

Описание сценария диалога выполняют:

- блок-схема, в которой предусмотрены блоки выдачи сообщений и обработки полученных ответов;
- ориентированный граф, вершины которого - сообщения и выполняемые действия, дуги - связь сообщений; словесное описание;
- специализированные объектно-ориентированные языки построения сценариев.

Для создания диалоговых процессов и интерфейса конечного пользователя наиболее подходят объектно-ориентированные инструментальные средства разработки программ.

Лекция 5. Тестирование, отладка и сборка ПО

1. Стандартизация и сертификация ПО

Целью системы сертификации ПО является проведение единой политики в области эффективного применения качественных программных средств, основываясь на системном учете требований стандартов (международных, государственных и отраслевых) и проведении объективной и независимой *оценки потребительских свойств* программных продуктов.

Принципы сертификации. *В* соответствии с действующим законодательством РФ под **сертификацией** понимается действие третьей стороны, независимой от производителя и потребителя продукции, доказывающее, что *продукция соответствует конкретному стандарту* (международному, национальному, отраслевому) или другому нормативному документу.

В основе сертификации лежат *испытания* (тестирование), т.е. совокупность технических процедур, позволяющих адекватно определить заявленную характеристику объекта сертификации.

По результатам испытаний может быть оформлен *сертификат* соответствия или *заключение о* несоответствии сертифицируемой продукции установленным требованиям. **Сертификат** является документом, выданным в соответствии с правилами Системы сертификации и удостоверяющим, что программный продукт соответствует установленным требованиям. В ходе сертификации должен быть решен целый ряд наиболее общих задач оценки качества программного обеспечения. *Во-первых,* должна быть установлена полнота исходных данных и параметров, описываемых в технической и сопроводительной документации. *Во- вторых,* необходимо обеспечить применение стандартных (аттестованных) методик испытаний или разработку специальных методик, учитывающих особенности конкретного продукта. *В-третьих,* задачей является также проведение испытаний и оценки достоверности их результатов. *В-четвертых,* заключительной задачей является обобщение результатов испытаний и получение окончательной оценки показателей качества сертифицируемого программного продукта.

Необходимость сертификации программного обеспечения определяет заказчик и отражает это в договоре на поставку продукции. Рыночные отношения заставляют использовать механизм независимой оценки качества программного обеспечения . В этом случае добровольная сертификация может проводиться по заявке разработчика. Особенно эффективна сертификация для программ массового пользования, выпускаемых большими тиражами, т.к. в этом случае резко снижаются издержки, вызванные претензиями потребителей. Вместе с тем *результаты сертификации* должны окупать затраты на её проведение, поэтому перед проведением сертификации необходима *детальная экономическая оценка.*

Технологический процесс испытаний и сертификации поддерживается, как правило, достаточно эффективными инструментальными пакетами для автоматизации испытаний и методиками оценки качества. Испытания уникального программного обеспечения для целей сертификации требуют предварительных исследований как в системной, так и в предметной областях, поэтому в таких случаях требуется более высокий уровень формализации и документального оформления всех условий и результатов, чем при обычных испытаниях.

При создании *программного продукта массового применения,* особенно требующего длительного сопровождения, необходимо более тесное взаимодействие между разработчиком и органом по сертификации. Могут, в частности, заключаться долгосрочные соглашения, регламентирующие периодические испытания и предоставление оперативной информации об обнаруженных ошибках или дефектах с учетом претензий пользователей. В процессе сопровождения программного продукта, например, при его модернизации, разработчик может вносить изменения, которые могут повлечь за собой новые ошибки Поэтому необходимо *непрерывное отслеживание* органом по сертификации изменений в программном обеспечении информационных технологий. В России в качестве основы для *национальной системы сертификации* продукции принята Система ГОСТ Р, документы которой регламентируют лишь общие требования к продукции . С учётом специфики сертификации информационно-программных средств Роскоминформ в 1994 году разработал *систему сертификации "Росинфосерт",* которая была в то время зарегистрирована в Госстандарте России и осуществляет принципы добровольной сертификации средств программного обеспечения различного назначения в соответствии с российской законодательной базой.

2. Определение и принципы тестирования

ГОСТ Р ИСО.МЭК 12119-2000: работы по тестированию, протоколы тестирования, отчет о тестировании.

Применительно к ПИ *тестирование* — это процесс многократного выполнения программы с целью обнаружения ошибок.

Программа тестируется для того, чтобы повысить уровень ее надежности, т.е. выявить максимальное число ошибок.

Цель тестирования — выявление как можно большего числа ошибок. При организации любого процесса очень важен правильный выбор цели , потому что для человеческого сознания характерна целевая направленность.

Из правильного определения тестирования вытекает ряд принципов, которые интуитивно ясны, но именно поэтому на них не обращают должного внимания.

Принцип 1. Процесс тестирования более эффективен, если проводится не автором программы.

Из определения тестирования как процесса, направленного на выявление ошибок, ясно, что тестирование тем эффективней , чем больше ошибок выявлено. Тестовый прогон, в результате которого не выявлено ошибок, считается неудачным (неэффективным). Таким образом, тестирование — это процесс деструктивный (разрушительный). Именно этим и объясняется, почему многие считают его трудным. Особенно трудным и малоэффективным он является для самого автора программы, так как после выполнения конструктивной части при проектировании и написании программы ему трудно перестроиться на деструктивный образ мышления и, создав программу, тут же

приступить к пристрастному выявлению в ней ошибок. Очевидно, что обнаружение недостатков в своей деятельности противоречит человеческой психологии . Это не означает, что программист не может тестировать свою программу. Речь идет о повышении эффективности тестирования.

Все эти рассуждения не относятся к отладке, т.е. к исправлению уже известных ошибок. Она эффективнее выполняется самим автором программы.

Принцип 2. Описание предполагаемых значений результатов тестовых прогонов должно быть необходимой частью тестового набора данных.

Тестирование как процесс многократного выполнения программы проводится на многочисленных входных наборах данных (принципы выбора входных данных будут рассмотрены ниже). Чтобы определить правильность полученных в результате очередного тестового прогона данных, необходимо знать ожидаемый результат, иначе правдоподобные результаты тестового прогона могут быть признаны правильными.

Таким образом, тестовый набор данных должен включать два компонента: описание входных данных и описание точного и корректного результата, соответствующего набору входных данных.

Этот принцип довольно сложно, а в некоторых случаях даже невозможно реализовать на практике. Сложность его заключается в том, что при тестировании программы (модуля) необходимо для каждого входного набора данных рассчитать вручную ожидаемый результат или найти допустимый интервал изменения выходных данных. Процесс этот очень трудоемкий даже для небольших программ, так как он требует ручных расчетов, следуя логике алгоритма программы.

Принцип 3. Необходимо досконально изучать результаты применения каждого теста.

Из практики видно, что значительная часть всех обнаруженных в конечном итоге ошибок могла быть выявлена в результате самых первых тестовых прогонов, но они были пропущены вследствие недостаточно тщательного анализа результатов первых тестовых прогонов.

Принцип 4. Тесты для неправильных и непредусмотренных входных данных должны разрабатываться также тщательно, как для правильных, предусмотренных.

Согласно этому принципу при обработке данных, выходящих за область допустимых значений, в тестируемой программе должна быть предусмотрена диагностика в виде сообщений. Если сообщение о причине невозможности обработки по предложенному алгоритму отсутствует и программа завершается аварийно или ведет себя непредсказуемо, то такая программа не может считаться работоспособной и требует существенной доработки.

Тестовые наборы данных из области недопустимых входных значений обладают большей обнаруживающей способностью, чем тесты, соответствующие корректным входным данным.

Принцип 5. Необходимо проверять не только, делает ли программа то, для чего она предназначена, но и не делает ли она то, что не должна делать.

Это утверждение логически вытекает из предыдущего. Необходимо любую программу проверить на нежелательные побочные эффекты. Например, если программа обработки и печати какой-нибудь ведомости дублирует первую или последнюю строку, то она содержит ошибку.

Принцип 6. Вероятность наличия необнаруженных ошибок в части программы пропорциональна числу ошибок, уже обнаруженных в этой части.

Это свойство ошибок группироваться объясняется тем, что части программы, где при тестировании обнаружено большее число ошибок, либо были слабо проработаны идеологически, либо разрабатывались программистами более низкой квалификации. Из этого принципа можно сделать практический вывод: если в какой-нибудь части программы обнаружено больше ошибок, чем в других, то ее необходимо тестировать более тщательно.

Итак, нужно помнить и знать, что:

 тестирование — это процесс многократного выполнения программы с целью выявления ошибок;

 тестовый прогон считается удачным, если он позволяет выявить ошибки, а тот тестовый набор эффективен, который имеет высокую вероятность обнаружения большего числа ошибок;

 трудность тестирования в том, что это процесс творческий, плохо поддающийся формализации.

3. Методы тестирования программ

Тестирование программ является одной из составных частей более общего понятия — «отладка программ». Если тестирование — это процесс, направленный на выявление ошибок, то целью отладки являются локализация и исправление выявленных в процессе тестирования ошибок.

Под *отладкой* понимается процесс, позволяющий получить программу, функционирующую с требующимися характеристиками в заданной области изменения входных данных.

Процесс отладки включает:

 действия, направленные на выявление ошибок (тестирование);

 диагностику и локализацию ошибок (определение характера и местонахождения ошибок);

 внесение исправлений в программу с целью устранения ошибок.

Из трех перечисленных видов работ самым трудоемким и дорогим является тестирование , затраты на которое для типичных ПИ приближаются к 40% общих затрат на разработку.

Процесс отладки начинается с разработки тестовых наборов данных по определенной методике, придерживаясь ряда правил.

Применяемые методы тестирования: статический, детерминированный, стохастический и в реальном масштабе времени.

Статическое тестирование — наиболее формализованное, базируется на правилах структурного построения программ и обработки данных. Проверка степени выполнения этих правил проводится без изменения объектного

кода программы путем формального анализа текста программы на языке программирования. Операторы и операнды текста программы анализируются в символьном виде, поэтому этот метод тестирования иногда называют *символическим тестированием.*

Наиболее трудоемким и детализированным является *детерминированное тестирование ,* которое требует многократного выполнения программы на компьютере с использованием определенных, специальным образом подобранных тестовых наборов данных. При детерминированном тестировании контролируются каждая комбинация исходных данных и соответствующие результаты, а также каждое утверждение в спецификации тестируемой программы.

При тестировании ПИ невозможно перебрать все комбинации исходных данных и проконтролировать результаты функционирования на каждой из них, поэтому для комплексного тестирования ПИ применяется *стохастическое тестирование.* Стохастическое тестирование предполагает использование в качестве исходных данных множества случайных величин с соответствующими распределениями, а для сравнения полученных результатов используются также распределения случайных величин.

Стохастическое тестирование применяется в основном для обнаружения ошибок, а для диагностики и локализации ошибок приходится переходить к детерминированному тестированию с использованием конкретных значений исходных данных из области изменения ранее использовавшихся случайных величин. Стохастическое тестирование наилучшим образом подвергается автоматизации путем использования датчиков случайных значений (генераторов случайных величин).

ПИ, предназначенные для работы в системах реального времени, должны проходить *тестирование в реальном масштабе времени.* В процессе такого тестирования проверяются результаты обработки исходных данных с учетом времени их поступления, длительности и приоритетности обработки, динамики использования памяти и взаимодействия с другими программами. При обнаружении отклонений результатов выполнения программ от ожидаемых для локализации ошибок приходится фиксировать время и переходить к детерминированному тестированию.

Каждый из рассмотренных методов тестирования не исключает последовательного применения другого метода, скорее наоборот, требование к повышению качества ПИ предполагает необходимость подвергать их различным методам тестирования (и их сочетаниям) в зависимости от сложности и области применения.

Каждый метод тестирования предполагает использование конкретных процедур для реализации. Статическое тестирование реализуется путем применения *ручных методов* тестирования программ . При ручных методах тестирования вероятность того, что при исправлении ошибок не вносятся новые ошибки, намного выше.

Основные методы ручного тестирования: инспекции исходного текста и сквозные просмотры. Они имеют много общего: предполагают некоторую

подготовительную работу; собирается собрание, состоящее из участников проверки, цель которого *нахождение ошибок,* но не их устранение (т.е. тестирование, а не отладка).

Инспекции *исходного текста* — набор правил и приемов обнаружения ошибок при изучении текста программы группой специалистов.

В инспектирующую группу входят обычно 4 человека: председатель (не автор, но знакомый с деталями программы); автор программы; проектировщик и специалист по тестированию.

В функции председателя входят: подготовка материалов для заседания инспектирующей группы; составление графика проведения инспекций; ведение заседания и регистрация всех найденных ошибок.

Общая процедура инспекции заключается в том, что председатель заранее (за несколько дней) раздает листинг программы и проектную спецификацию остальным членам группы для ознакомления. Инспекционное заседание начинается с доклада автора о логике своей программы, о методах и приемах, использованных при ее написании. Далее программа анализируется по списку вопросов для выявления общих ошибок программирования.

В качестве положительных моментов при инспекции исходного текста можно отметить, что результаты инспекции позволяют программисту увидеть сделанные им ошибки и способствуют его обучению, а также позволяют оценить стиль программирования.

Сквозные просмотры представляют собой ряд процедур и способов обнаружения ошибок, осуществляемых группой специалистов.

Процедура подготовки и проведения заседания при сквозном просмотре отличается тем, что на подготовительном этапе специалист по тестированию готовит небольшое число тестов, во время заседания каждый тест мысленно выполняется, и состояние программы отслеживается на бумаге или доске. Количество тестов должно быть небольшим, и они должны быть простыми. Фактически оба рассмотренных метода — это развитие всем хорошо знакомой процедуры «проверки за столом», когда программист просматривает свою программу перед началом тестирования. Однако они более эффективны, так как осуществляется один из основных принципов тестирования (тестирование не должен проводить автор программы), и намного дешевле машинного тестирования (не используется дорогостоящее машинное время).

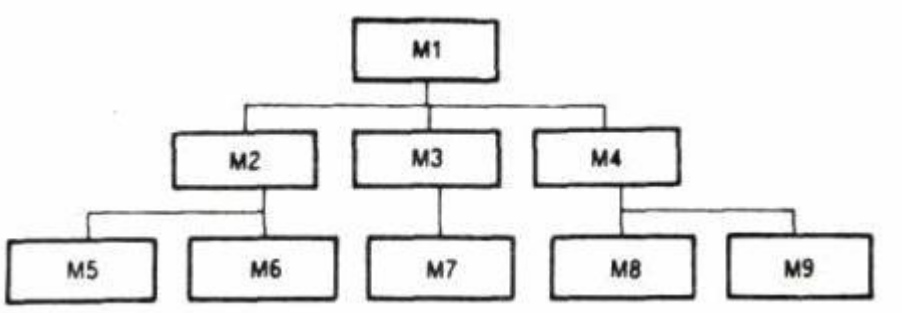
4. Сборка программ при тестировании

Сборка модулей в программный комплекс может осуществляться двумя методами: монолитным, пошаговым. Пошаговая сборка может, в свою очередь, быть восходящей (снизу-вверх) и нисходящей (сверху-вниз).

Монолитный метод сборки предполагает выполнение автономного тестирования каждого модуля, а затем их одновременную сборку и тестирование в комплексе.

Пошаговое тестирование предполагает последовательное подключение к набору уже оттестированных модулей очередного тестируемого модуля.

В качестве примера рассмотрим программу, состоящую из 9 модулей:



При монолитном тестировании все 9 модулей, входящих в программу, тестируются независимо друг от друга, последовательно или параллельно. Затем они собираются в одну программу. Для автономного тестирования любого модуля нужен модуль -драйвер (отлаживающий модуль) и один или несколько модулей-заглушек (имитаторы). Для рассматриваемого примера модули-драйверы нужны для всех модулей, кроме модуля М1, а модули-заглушки нужны для всех модулей, кроме М5, М6, М7, М8, М9 (т.е. модулей самого низшего уровня).

Таким образом, при монолитной сборке для автономного тестирования составляющих программный комплекс модулей дополнительно необходимо разработать 8 модулей-драйверов и минимум 9 модулей-заглушек.

Драйвер — это модуль, обеспечивающий вызов и передачу тестируемому модулю необходимых входных данных и обработку результатов.

Заглушка — это модуль, имитирующий функции модулей, вызываемых тестируемым.

Метод пошаговой сборки предполагает, что модули тестируются не автономно, а подключаются поочередно для выполнения теста к набору уже ранее оттестированных модулей.

При пошаговом тестировании раньше обнаруживаются ошибки в интерфейсах между модулями, поскольку раньше начинается сборка программы. При монолитном методе модули «не видят друг друга» до последней фазы.

Отладка программ при пошаговом тестировании легче, так как большинство ошибок интерфейса трудно локализовать при монолитном тестировании. Однако безусловным преимуществом монолитного метода сборки является большая возможность распараллеливания работ.

Лекция 6. Документация ПО

1. Нормативная база в области документирования ПС

Стандарты ЕСПД в основном охватывают ту часть документации, которая создается в процессе разработки ПС, и связаны , по большей части, с документированием функциональных характеристик ПС. Следует отметить, что стандарты ЕСПД (ГОСТ 19) носят рекомендательный характер. Дело в

том, что в соответствии с Законом РФ « О стандартизации» эти стандарты становятся обязательными на контрактной основе - то есть при ссылке на них в договоре на разработку (поставку) ПС.

Надо сказать, что наряду с комплексом ЕСПД официальная нормативная база РФ в области документирования ПС и в смежных областях включает ряд перспективных стандартов (отечественного, межгосударственного и международного уровней), например , Международный стандарт **ISO/IEC 12207: 1995-08-01** на организацию ЖЦ продуктов программного обеспечения (ПО).

Перечень документов ЕСПД.

1. ГОСТ 19.001-77 ЕСПД. Общие положения.
2. ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов.
3. ГОСТ 19.102-77 ЕСПД. Стадии разработки.
4. ГОСТ 19.103-77 ЕСПД. Обозначение программ и программных документов.
5. ГОСТ 19.104-78 ЕСПД. Основные надписи.
6. ГОСТ 19.105-78 ЕСПД. Общие требования к программным документам.
7. ГОСТ 19.106-78 ЕСПД. Требования к программным документам, выполненным печатным способом.
8. ГОСТ 19.201-78 ЕСПД. Техническое задание. Требования к содержанию и оформлению.
9. ГОСТ 19.202-78 ЕСПД. Спецификация. Требования к содержанию и оформлению.
10. ГОСТ 19.301-79 ЕСПД. Порядок и методика испытаний.
11. ГОСТ 19.401-78 ЕСПД. Текст программы. Требования к содержанию и оформлению.
12. ГОСТ 19.402-78 ЕСПД. Описание программы.
13. ГОСТ 19.404-79 ЕСПД. Пояснительная записка. Требования к содержанию и оформлению.
14. ГОСТ 19.501-78 ЕСПД. Формуляр. Требования к содержанию и оформлению.
15. ГОСТ 19.502-78 ЕСПД. Описание применения. Требования к содержанию и оформлению.
16. ГОСТ 19.503-79 ЕСПД. Руководство системного программиста. Требования к содержанию и оформлению.
17. ГОСТ 19.504-79 ЕСПД. Руководство программиста.
18. ГОСТ 19.505-79 ЕСПД. Руководство оператора.
19. ГОСТ 19.506-79 ЕСПД. Описание языка.
20. ГОСТ 19.508-79 ЕСПД. Руководство по техническому обслуживанию. Требования к содержанию и оформлению.
21. ГОСТ 19.604-78 ЕСПД. Правила внесения изменений в программные документы, выполняемые печатным способом.

22. ГОСТ 19.701-90 ЕСПД. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения.
23. ГОСТ 19.781-90. Обеспечение систем обработки информации программное.

2. Некоторые из стандартов документирования ПО

ГОСТ (СТ СЭВ) 19.101-77 (1626-79). ЕСПД. Виды программ и программных документов (Переиздан в ноябре 1987г с изм.1). Устанавливает виды программ и программных документов для вычислительных машин, комплексов и систем независимо от их назначения и области применения.

Виды программ

Вид программы	Определение
Компонент	Программа, рассматриваемая как единое целое, выполняющая законченную функцию и применяемая самостоятельно или в составе комплекса
Комплекс	Программа, состоящая из двух или более компонентов и (или) комплексов, выполняющих взаимосвязанные функции, и применяемая самостоятельно или в составе другого комплекса

Виды программных документов

Вид программного документа	Содержание программного документа
Спецификация	Состав программы и документации на нее
Ведомость держателей подлинников	Перечень предприятий, на которых хранят подлинники программных документов
Текст программы	Запись программы с необходимыми комментариями
Описание программы	Сведения о логической структуре и функционировании программы
Программа и методика испытаний	Требования, подлежащие проверке при испытании программы, а также порядок и методы их контроля
Техническое задание	Назначение и область применения программы, технические, технико-экономические и специальные требования, предъявляемые к программе, необходимые стадии и сроки разработки, виды испытаний
Пояснительная записка	Схема алгоритма, общее описание алгоритма и (или) функционирования программы, а также обоснование принятых технических и технико-экономических решений
Эксплуатационные документы	Сведения для обеспечения функционирования и эксплуатации программы

Виды эксплуатационных документов

Вид эксплуатационного документа	Содержание эксплуатационного документа
Ведомость эксплуатационных документов	Перечень эксплуатационных документов на программу
Формуляр	Основные характеристики программы, комплектность и сведения об эксплуатации программы
Описание применения	Сведения о назначении программы, области применения, применяемых мето-дах, классе решаемых задач, ограничениях для применения, минимальной конфигурации технических средств
Руководство системного	Сведения для проверки, обеспечения функционирования и настройки про-

программиста	граммы на условия конкретного применения
Руководство программиста	Сведения для эксплуатации программы
Руководство оператора	Сведения для обеспечения процедуры общения оператора с вычислительной системой в процессе выполнения программы
Описание языка	Описание синтаксиса и семантики языка
Руководство по техническому обслуживанию	Сведения для применения тестовых и диагностических программ при обслуживании технических средств

В зависимости от способа выполнения и характера применения программные документы подразделяются на подлинник, дубликат и копию (ГОСТ 2.102-68), предназначенные для разработки, сопровождения и эксплуатации программы.

ГОСТ 19.105-78 ЕСПД. Общие требования к программным документам.

Настоящий стандарт устанавливает общие требования к оформлению программных документов для вычислительных машин, комплексов и систем, независимо от их назначения и области применения и предусмотренных стандартами Единой системы программной документации (ЕСПД) для любого способа выполнения документов на различных носителях данных.

Программный документ может быть представлен на различных типах носителей данных и состоит из следующих условных частей: *титульной; информационной; основной.*

Правила оформления документа и его частей на каждом носителе данных устанавливаются стандартами ЕСПД на правила оформления документов на соответствующих носителях данных.

ГОСТ 19.106-78 ЕСПД. Требования к программным документам, выполненным печатным способом.

Программные документы оформляют:

- на листах формата А4 (ГОСТ 2.301-68) при изготовлении документа машинописным или рукописным способом;
- допускается оформление на листах формата А3;
- при машинном способе выполнения документа допускаются отклонения размеров листов, соответствующих форматам А4 и А3, определяемые возможностями применяемых технических средств; на листах форматов А4 и А3, предусматриваемых выходными характеристиками устройств вывода данных, при изготовлении документа машинным способом;
- на листах типографических форматов при изготовлении документа типографским способом.

Расположение материалов программного документа осуществляется в следующей последовательности:

титульная часть:

- лист утверждения (не входит в общее количество листов документа);
- титульный лист (первый лист документа);

информационная часть:

- аннотация;
- лист содержания;

основная часть:

- текст документа (с рисунками, таблицами и т.п.)
- перечень терминов и их определений;
- перечень сокращений;
- приложения;
- предметный указатель;
- перечень ссылочных документов;

часть регистрации изменений:

- лист регистрации изменений.

Перечень терминов и их определений, перечень сокращений, приложения, предметных указатель, перечень ссылочных документов выполняются при необходимости.

ГОСТ 19.402-78 ЕСПД. Описание программы.

Состав документа «Описание программы» в своей содержательной части может дополняться разделами и пунктами, почерпнутыми из стандартов для других описательных документов и руководств: *ГОСТ 19.404-79 ЕСПД. Пояснительная записка, ГОСТ 19.502-78 ЕСПД. Описание применения, ГОСТ 19.503- 79 ЕСПД. Руководство системного программиста, ГОСТ 19.504-79 ЕСПД. Руководство программиста, ГОСТ 19.505-79 ЕСПД. Руководство оператора.*

Есть также группа стандартов, определяющая требования к фиксации всего набора программ и ПД, которые оформляются для передачи ПС. Они порождают лаконичные документы учетного характера и могут быть полезны для упорядочения всего хозяйства программ и ПД (ведь очень часто требуется просто навести элементарный порядок!). Есть и стандарты, определяющие правила ведения документов ПС.

ГОСТ 19.301-79 ЕСПД. Программа и методика испытаний, который (в адап-тированном виде) может использоваться для разработки документов плани-рования и проведения испытательных работ по оценке готовности и качества ПС.

ГОСТ 19.701-90 ЕСПД. Схемы алгоритмов, программ, данных и систем. Обозначения условные графические и правила выполнения. Он устанавливает правила выполнения схем, используемых для отображения различных видов задач обработки данных и средств их решения и полностью соответствует стандарту ИСО 5807:1985.

Наряду с ЕСПД на межгосударственном уровне действуют еще два стандарта, также относящихся к документированию ПС и принятых не так давно, как большая часть ГОСТ ЕСПД.

ГОСТ 19781-90 Обеспечение систем обработки информации программное. Термины и определения. Разработан взамен ГОСТ 19.781-83 и ГОСТ 19.004-80 и устанавливает термины и определения понятий в области программного обеспечения (ПО) систем обработки данных (СОД), применяемые во всех видах документации и литературы, входящих в сферу работ по стандартизации или использующих результаты этих работ.

ГОСТ 28388-89 Системы обработки информации. Документы на магнитных носителях данных. Порядок выполнения и обращения. Распространяется не только на программные, но и на конструкторские, технологические и другие проектные документы, выполняемые на магнитных носителях.

ГОСТ Р ИСО/МЭК 9294-93. Информационная технология. Руководство по управлению документированием программного обеспечения.

ГОСТ Р ИСО/МЭК 9126-93. Информационная технология. Оценка программной продукции, характеристика качества и руководство по их приме-нению.

ГОСТ Р ИСО /МЭК 9127-94. Системы обработки информации. Документация пользователя и информация на упаковке для потребительских программных пакетов.

ГОСТ Р ИСО/ МЭК 8631-94. Информационная технология. Программные конструктивы и условные обозначения для их представления.

ГОСТ Р ИСО/МЭК 12119:1994. Информационная технология. Пакеты программных средств. Требования к качеству и испытания.

ГОСТ Р ИСО/МЭК 12207-99. Процессы жизненного цикла программных средств.

ISO 12207: 1995. (ГОСТ Р-1999). ИТ. Процессы жизненного цикла программных средств.

ISO 15271: 1998. (ГОСТ Р-2002). ИТ. Руководство по применению ISO 12207.

ISO 16326: 1999. (ГОСТ Р-2002). ИТ. Руководство по применению ISO 12207 при административном управлении проектами.

ISO 15504- 1-9: 1998 : Агапова А.С. Оценка и аттестация зрелости процессов создания и сопровождения программных средств. Изд. «Книга и бизнес», 2001

ISO 9126: 1991. (ГОСТ-1993). ИТ. Оценка программного продукта. Характеристики качества и руководство по их применению*

ISO 12119: 1994. (ГОСТ Р-2000). ИТ. Требования к качеству и тестирование. *

ISO 13210: 1994. ИТ. Методы тестирования для измерения соответствия стандартам POSIX.

Список литературы:

1. Автоматизированные информационные технологии в экономике: Учебник / Под ред. Г.А. Титоренко. – М.: ЮНИТИ-ДАНА, 2007. – 463 с.
2. Благодатских В.А., Волнин В.А., Поскалов К.Ф. Стандартизация разработки программных средств. – М.: Финансы и статистика, 2007. – 288 с.
3. Вендров А.М. Проектирование программного обеспечения экономических информационных систем – М: Финансы и статистика, 2006. – 544 с.
4. Глаголев В. Разработка технической документации. – СПб.: Питер, 2008. – 192 с.
5. Орлов С.А. Технологии разработки программного обеспечения: Разработка сложных программных систем: Учебное пособие. – 3-е изд. – СПб.: Питер, 2004. – 526 с.