РАЗРАБОТКА И СТАНДАРТИЗАЦИЯ ПРОГРАММНЫХ СРЕДСТВ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

С. П. Сидоров

Оглавление

ВВЕДЕНИЕ

Увеличивающая в мировом масштабе конкуренция среди организаций разработчиков ПО, повышение требований конечного пользователя к качеству и надежности программных средств привело их разработчиков к пониманию важности вопросов стандартизации.

Для того чтобы поддерживать конкурентоспособность своей организации, разработчики ПО должны применять все более эффективные, рентабельные методы, технологии, инструментальные средства, способствующие постоянному повышению качества и более совершенному удовлетворению потребителей ПО. Требования потребителей часто включаются в технические условия или неформализованные требования, описанные на некотором вербальном языке. Однако технические условия и неформализованные требования сами по себе не гарантируют их удовлетворение в конечном продукте, так как в настоящее время существует проблема выработки приемлемых требований к программному продукту, а также ряд других проблем, возникающих в процессе разработки конечного продукта. Это соображение привело к разработке стандартов, руководств, руководящих документов, относящихся к системам качества и дополняющих релевантные требования к ПО, установленные в технических требованиях.

Курс «Разработка и стандартизация программных средств и информационных технологий» предназначен для студентов механико-математического факультета, обучающихся по специальности «Прикладная информатика (по областям)».

Цели курса:

- знакомство с организацией проектирования программного обеспечения,
- этапами процесса проектирования,
- стандартизацией информационных технологий,
- действующими стандартами,
- оценкой качественных и количественных характеристик программного обеспечения.

Данный курс описывает современные типовые подходы к разработке программного обеспечения и содержит теоретические основы и практические рекомендации по проектированию и разработке программных продуктов.

На практических занятиях планируется реализация процессов жизненного цикла программного изделия, для которого предполагается возможность его тиражирования и применения в виде пакета прикладных программ для решения определенного набора экономических задач конечного пользователя.

Особое внимание уделяется стандартизации и метрологии в разработке программного обеспечения.

Глава 1

Понятие стандартизации

1.1. Виды нормативных документов

Согласно определению стандарта ИСО-МЭК-2 стандартизация — это деятельность, направленная на достижение оптимальной степени упорядочения в определенной области посредством установления положений для всеобщего и многократного исследования в отношении реально существующих или потенциальных задач.

В соответствии с определением этого понятия формируются основные понятия стандартизации:

- *нормативный документ* — документ, устанавливающий правила, общие принципы или характеристики, касающиеся различных видов деятельности или их результатов;
- *стандарт* — нормативный документ по стандартизации, разработанный на основе согласия по существенным вопросам заинтересованных сторон и утвержденный признанным органом (предприятием);
- *правило (ПР)* — документ, устанавливающий обязательные для применения организационно-технические и общетехнические положения, порядки, методы выполнения работ;
- *рекомендации (Р)* — документ, содержащий добровольные для применения организационно-технические или общетехнические положения, порядки, методы выполнения работ;
- *норма* — положение, устанавливающее количественные или качественные критерии, которые должны быть удовлетворены;
- *регламент* — документ, содержащий правовые нормы и принятый органом власти;
- *общероссийский классификатор технико–экономической и социальной информации (ОКТЭСИ)* — официальный документ, представляющий со-

бой систематизированный свод наименований и кодов классификационных группировок и (или) объектов классификации в области технико–экономической и социальной информации.

Стандарты различают в зависимости от сферы действия:

- международный стандарт;
- региональный стандарт;
- государственный стандарт РФ (ГОСТ-Р);
- межгосударственный стандарт (ГОСТ);
- стандарт отрасли (ОСТ);
- стандарт научно-технического или инженерного общества (СТО);
- стандарт предприятия (СТП).

Термины «регламент» и «технический регламент» являются родовыми понятиями. К техническим регламентам относятся законодательные акты, постановления Правительства РФ, содержащие требования, нормы и технические характеристики; государственные стандарты в части устанавливаемых в них требований, нормы и правила федеральных органов исполнительной власти. Примером последнего могут служить Строительные нормы и правила, Санитарные нормы и правила, Правила по стандартизации, метрологии и сертификации Госстандарта РФ и пр.

Стандартизация как вид человеческой деятельности является результатом стремления отбирать наиболее ценные достижения и использовать их в дальнейшей работе. В измерительной технике это проявляется в создании систем мер и весов, в промышленности — в унификации изделий, в агрегатировании узлов и механизмов, в расчетах — в использовании определенных (стандартных) рядов чисел, в экономике — в единообразии представления информации и т. д.

Начиная с древних времен люди стремились отобрать наиболее удачно созданные орудия труда, самые удачные механизмы — от колеса до лазера, — самые ценные технологические процессы и т. д. В итоге были сформированы основные принципы стандартизации.

1.2. Основные принципы стандартизации

Можно выделить несколько основных принципов стандартизации.

1. *Сбалансированность интересов сторон, разрабатывающих, изготавливающих и потребляющих продукт деятельности человека.* Очевидно, что консенсус в такой цели — проблема сложная хотя бы потому, что разработчик, например, стремится создать наиболее совершенное средство измерения, изготовитель больше заботится о технологичности и стоимости, а потребитель - об удовлетворении потребности в измерениях. То же самое относится и к промышленной продукции и к сфере услуг.

2. *Системность и комплексность стандартизации.* Системность — это рассмотрение каждого объекта как части более сложной системы. Например, современный персональный компьютер состоит из узлов и стандартных программ, и при их разработке необходимо обращать внимание на определенные стандартные требования в комплексе.

3. *Динамичность и опережающее развитие стандартизации.* Этот принцип сформирован в современных основополагающих законах о стандартизации и состоит в том, что необходимо учитывать возможность появления новых изделий и новых технологических процессов. Новые изделия, не соответствующие действующим стандартам, не смогут эффективно использоваться, если в последних не предусмотрена возможность их появления.

4. *Эффективность стандартизации.* Применение стандартов должно давать экономический или социальный эффект. Это достигается экономией ресурсов, повышением надежности, повышением технической и информационной совместимости. Под социальным эффектом понимают факторы, влияющие на экологию, на обеспечение безопасности и здоровья людей.

5. *Приоритетность разработки стандартов, способствующих безопасности, совместимости и взаимозаменяемости продукции и услуг.*

6. *Принцип гармонизации.* Этот принцип предусматривает разработку гармонизированных стандартов. Это означает, что стандарты всех уровней от международных до стандартов отдельных предприятий должны быть составлены единообразно и без противоречий. Только такой подход обеспечивает беспрепятственное взаимодействие предприятий, министерств, партнеров в международной торговле.

1.3. Уровни стандартизации

Уровень стандартизации — форма участия в деятельности по стандартизации, учитывающая географические, политические и экономические признаки. Уровень стандартизации может быть международным, региональным, национальным, ведомственным.

Исторически истоки формирования международных органов по обеспечению единства измерений и по стандартизации имеет смысл проследить начиная с 1875 г., когда 17 государств мира, в том числе и Россия, приняли Метрическую конвенцию «для обеспечения единства и усовершенствования метрической системы». Для координации действий государств–членов метрической конвенции было учреждено Международное бюро по мерам и весам (МБМВ) с дислокацией в Севре — ранее в пригороде Парижа, а в настоящее время — одном из районов Парижа. Официально от каждого государства были названы национальные центры по координации взаимодействий метрологов стран–участниц конвенции с МБМВ. От России таким центром назван IMM (Institute Metrology Mendeleev) — Институт метрологии им. Д.И. Менделеева в Санкт-Петербурге.

МБМВ организует в среднем раз в четыре года конференции, на которых принимаются основополагающие решения в области метрологии. Например, на одной из Генеральных конференций по мерам и весам в 1954 г. была окончательно принята система СИ. В 1983 г. принято новое определение метра. На конференции в 1990 г. узаконена современная международная температурная шкала МПТШ–90 и т. д. На конференциях принимаются основополагающие решения, которые определяют направления наиболее важных для метрологии научных программ на следующие 4 года. Там же и распределяется финансирование этих программ. Из научных программ последних лет можно назвать исследования возможности использования "теплой"сверхпроводимости в создании эталонов, исследование средств измерения малых длин на тоннельном эффекте, совершенствование эталонной базы в области электрических и магнитных измерений, работы по уточнению постоянной Авогадро и т. д.

В перерывах между конференциями деятельность МБМВ проводится в рамках комитетов, которые в основном опираются на метрологические службы стран-учредителей Метрической конвенции. Так, например, работы по электричеству традиционно ведет Национальная физическая лаборатория Англии, температурными измерениями руководит Италия, по метрологии в физико-химических измерениях координатором является PTB - национальная физическая лаборатория Германии в г. Брауншвейге - и т. д. Наша страна

начиная с 1990 г. усилила свою активность в деятельности МБМВ.

Кроме обеспечения единства измерений по всем видам выдвинулись и были реализованы международные контакты в отдельных видах измерения. В первой главе указывалось, что к современному пониманию единства измерений физических величин специалисты пришли не сразу. Предлагалось, например, измерения электрических и магнитных величин строить по той же схеме, по которой построены механические измерения. Для этого специалисты считали необходимым принять несколько основных электрических единиц, например единицы силы тока, разности потенциалов и сопротивления. Далее следовало строить систему электрических единиц самостоятельно и независимо от единиц механических. Для выработки такого подхода и организации единства измерения в 1881 г. был созван Международный конгресс по электричеству, который созывался затем в 1889, 1893, 1900 и 1904 гг.

1.4. Международные организации по стандартизации

В 1904 г. на заседании Международного конгресса по электричеству было решено создать Комиссию по рассмотрению вопросов стандартизации терминологии и номинальных параметров электрических машин. В 1906 г. В Лондоне на конференции с участием представителей 13 стран была утверждена *Международная электротехническая комиссия (МЭК).* Первым его президентом был избран один из самых известных физиков лорд Кельвин, именем которого сейчас названа шкала термодинамической температуры. В 1908 г. был принят Устав МЭК. В состав комиссии вошли представители более 50 стран. Комиссия состояла из 100 комитетов и 500 подкомитетов, которые разрабатывали рекомендации, получившие в настоящее время права международных стандартов. Сегодня МЭК работает по следующим направлениям: унификация терминологии, обозначение маркировки, стандартизация материалов, применяемых в электротехнике, рекомендации по стандартизации электротехнического оборудования и еще по целому ряду направлений, связанных со стандартизацией различных узлов и электроизмерительных приборов.

Параллельно с деятельностью МЭК возникла необходимость стандартизации в общей и специальной технике и в машиностроении. В 1926 г. в Нью-Йорке обсуждался вопрос о создании международного органа, занимающегося этими вопросами. В этом же году в Праге была создана Международная федерация национальных ассоциаций по стандартизации (ИСА). В 1932 г.

в Берлине был проведен съезд, на котором было образовано 32 комитета, которые должны были заниматься разработкой основных принципов международной стандартизации. В период Второй мировой войны работа МЭК и ИСА была приостановлена, а по окончании войны в 1946 г. в Лондоне состоялась конференция, в которой приняли участие 64 делегата из 25 стран. В итоге работы этой конференции на базе объединения ИСА и МЭК была создана *Международная организация по стандартизации* — ИСО. 24 октября 1946 г. состоялось первое заседание Генеральной ассамблеи этой организации. Были утверждены Устав и Правила процедуры. Днем основания ИСО считается 23 февраля 1947 г. Сейчас в ИСО представлены 118 стран. Деятельность по международной стандартизации осуществляется более чем в 200 технических комитетах. Имеется более 2600 подкомитетов и рабочих групп, решающих более мелкие вопросы. С1972 г. ИСО издает международные стандарты, которые ранее назывались Рекомендациями по стандартизации.

МЭК присоединился к ИСО в 1947 г. на автономных правах, сохранив финансовую и организационную самостоятельность. МЭК, как и ИСО, пользуется консультативным статусом ООН. Для согласования работы МЭК и ИСО существует Координационный комитет ИСО/МЭК.

В марте 1987 г. ИСО приняла стандарты серии 9000 на системы качества, дополняющие требования к продукции или к услугам. Серия стандартов серии 9000 с 1988 г. принята в СССР и с 1991 г. в России в качестве национальных стандартов под номерами ГОСТ 40.9001-88 - ГОСТ 40.9003-88. Однако в связи с пересмотром в 1994 г. стандартов ИСО серии 9000 и их превращения в «семейство» из 30 наименований возникла необходимость пересмотра российских стандартов. Характерно, что более 90 стран мира имеют стандарты, эквивалентные ИСО-9000, а более 100 тысяч компаний сертифицировали свои системы качества на соответствие этим стандартам. В 70-х г. по инициативе комитета по надежности МЭК (ТК-56) и ИСО были начаты работы по созданию системы сертификации - документального подтверждения соответствия качества выпускаемой продукции стандартам ИСО.

Определенный интерес представляет деятельность Международной конференции по аккредитации испытательных лабораторий (ИЛАК), задачами которой являются обмен информацией по аккредитации, содействие взаимному признанию испытаний в разных странах, организация совместных исследований в лабораториях различных стран, аккредитованных по какому-либо определенному виду измерений.

Еще одна структура была создана в 1956 г. для решения международных

проблем в общих вопросах законодательного плана в обеспечении единства измерений. Эта организация известна как МОЗМ - Международная организация законодательной метрологии. В состав МОЗМ входят около 90 стран, в том числе и Россия. МОЗМ занимается вопросами метрологии общего плана: установлением классов точности средств измерения, обеспечением единообразия типов, образцов и систем измерительных приборов, разработкой рекомендаций по испытаниям, единообразием метрологических характеристик приборов, единообразием методик выполнения измерений, единообразием методов и средств поверки. Международная организация законодательной метрологии работает по комитетам, в которых руководящая роль (пилот комитета) поручается национальным органам какой-либо из стран-участниц. Другие страны участвуют в работе комитетов МОЗМ как докладчики, разрабатывая частные вопросы общих проблем.

В измерительной технике и в приборостроении в 1958 г. была образована научная консультативная организация ИМЕКО, в задачу которой входит проведение международных конгрессов и семинаров по актуальным проблемам развития измерительной техники.

Создавались и действуют региональные международные метрологические организации. Так, западноевропейские государства образовали Европейскую организацию по метрологии (ЕВРОМЕТ), страны Центральной и Восточной Европы - КООМЕТ. В последнюю входят Беларусь, Болгария, Германия, Польша, Россия, Румыния, Словакия, Украина, Республика Куба. КООМЕТ образовалась на этапе распада Совета Экономической Взаимопомощи и, соответственно, с 1991 г. унаследовала весь опыт и важнейшие документы по стандартизации СЭВ. Процесс окончательного оформления КООМЕТ еще не закончился, но уже приняты соглашения по взаимному признанию результатов испытаний в странах-участницах, учреждена Комиссия по метрологии КООМЕТ.

1.5. Национальные организации по стандартизации

В развитых странах метрологическая деятельность, вопросы обеспечения единства измерений, контроль за выполнением обязательных стандартов регулируются соответствующими законами, в которых обозначены государственные службы, ответственные за исполнение этих законов. Так, в Великобритании эта деятельность определена законом «О мерах и весах», в Германии — Конституцией (статья 73) и законами «Об измерительном деле и поверке» (1985 г.), «Об единицах измерений и измерительном деле» (1985

г.); в США — Конституцией (статья 1, раздел 8) и законами «О фасовке и хранении товара» (1966 г.), «О метрической системе» (1966 г.); во Франции — законом «О метрической системе и поверке средств измерений» (1985 г.); в Японии — законом «Об измерениях» (1992 г.).

В указанных странах вопросы исполнения законов об обеспечении единства измерений контролируются государственными метрологическими институтами и национальными лабораториями. В Великобритании во главе метрологических работ находится Национальная физическая лаборатория (NPL), подчиненная (до 1994 г.) государственному секретарю по торговле и промышленности; в Германии — Физико–технический институт (PTB), подчиненный Министерству экономики; в Италии имеется два государственных метрологических института; в США — Национальный институт стандартов и технологий NIST в г. Гейтесберге вблизи столицы США — Вашингтона.

Последний является признанным мировым центром метрологии и стандартизации и хорошо известен мировой общественности как NBS — Национальное бюро стандартов. Национальное бюро стандартов США было образовано как национальная метрологическая служба в 1901 г. В штате Мэриленд располагалась основная часть NBS с бюрократическими службами и научными подразделениями. Одновременно в Боулдери, штат Колорадо, был создан комплекс защищенных от помех лабораторий, в основном занимающихся электрическими и магнитными измерениями. Проводимая руководством NBS и Министерством торговли и промышленности США политика привлечения к работам ведущих специалистов разных стран привела к тому, что к настоящему моменту этот центр является наиболее авторитетным среди метрологических институтов в вопросах обеспечения единства измерений.

В 1988 г. МВБ был преобразован в Национальный институт стандартов и технологий (NIST) в связи с тем, что в стране и в мире появилась необходимость стандартизации не только измерительной техники, товаров и услуг, но и стандартизации технологий. На практике это означает, что высококвалифицированные специалисты NIST стали проводить исследования по заказам различных фирм по перепрофилированию бизнеса.

NIST выдает обоснованные рекомендации по внедрению новых технологий на предприятиях с учетом имеющегося у них задела. Естественно, такая услуга стоит достаточно дорого, что позволяет заработанные деньги использовать на разработку новой техники и новых технологий. Огромную работу по метрологии и стандартизации NIST США проводит в области создания и аттестации стандартных образцов. Используя эталонное хозяйство и научный потенциал сотрудников, NIST аттестует и продает большое количество

стандартных образцов. В других странах, в том числе и в нашей, для аттестации состава стандартных образцов проводят анализы в нескольких крупных аналитических центрах. Затем результаты сравниваются, анализируются и при отсутствии расхождений записываются в паспорт стандартного образца. Процедура получается длительная, и не всегда при наличии расхождений в измерениях удается выявить наиболее достоверные. В NIST США этой проблемы нет. Сторонние соисполнители привлекаются только в тех случаях, когда нет твердой уверенности в правильности анализов, но таких ситуаций в NIST практически не бывает.

Большая работа в NIST США проводится по сбору, анализу и публикации стандартных справочных данных. Сюда входят как фундаментальные физические константы, так и атомные и молекулярные константы, а также справочные данные по физическим и химическим свойствам веществ и материалов. Справочники NIST, например, по спектроскопии являются самыми полными, самыми надежными и самыми удобными в использовании среди всей справочной литературы в мире. Высокий научный авторитет национального центра США в области метрологии и стандартизации тем более удивителен, что в промышленности и в измерительной национальной технике США используют старую английскую систему мер. Длины принято выражать в футах и милях, вес в фунтах, объем в галлонах. Даже в измерениях температуры в повседневной жизни в США используется шкала Фаренгейта. Если сюда добавить, что стандарты параметров переменного тока в США (частота 60 Гц, напряжение 110 В) отличаются от европейских (частота 50 Гц, напряжение 220 В), то становится очевидным разница в основах стандартизации международной и национальной США.

Опыт NIST США по привлечению к работам самых квалифицированных ученых разных стран позволил добиться удивительных для метрологического центра научных результатов. Например, сотрудники этого института недавно получили Нобелевскую премию за разработку туннельного микроскопа — средства измерения сверхмалых длин. В мире действует семь международных региональных организаций по метрологии и стандартизации: в Скандинавии, в Латинской Америке, в арабском регионе, в Африке, в Европейском союзе.

Для нашей страны наиболее интересен опыт стандартизации в рамках Европейского союза (ЕС). Служба стандартизации ЕС обслуживает в общей сложности 400 млн. жителей 15 стран. В 1961 г. был учрежден Европейский комитет по стандартизации (СЕН). В 1972 г. был создан Европейский комитет по стандартизации в электротехнике (СЕНЭЛЕК). В рамках этих комитетов действует 239 технических комитетов (ТК). Основной задачей было

создать из сотен тысяч национальных стандартов несколько тысяч единых стандартов.

За прошедшие годы работы по созданию единых европейских стандартов касались в основном процессов конкурентной борьбы товаров на европейском рынке. К технике и технологиям разработки Комиссией ЕС в основном имели отношение в рамках программы «Зеленая книга» — создание евростандартов, отражающих новые достижения техники и технологии и директив, содержащих эффективные меры против проникновения в ЕС продукции, вредной для населения и небезопасной для окружающей среды.

Особенность большинства евростандартов состоит в том, что в их основу закладывают лучшие стандарты отдельных стран. Например, известные своим высоким качеством стандарты Швеции по электромагнитной безопасности персональных компьютеров положены в основу единого стандарта ЕС.

1.6. Структура и функции органов Госстандарта РФ

Согласно закону «Об обеспечении единства измерений», статьи 4-10, государственным управлением деятельности по обеспечению единства измерений в нашей стране занимается Комитет Российской Федерации по стандартизации, метрологии и сертификации (Госстандарт России). Законом регламентирована деятельность Госстандарта в координации всех работ в РФ по обеспечению единства измерений, в представлении Правительству РФ предложений по совершенствованию системы обеспечения единства измерений, руководство деятельностью институтов Госстандарта и центров стандартизации и метрологии (ЦСМ РФ), руководство международной деятельностью по метрологии и стандартизации.

Госстандарт утверждает все нормативные документы, регулирующие деятельность в стране по стандартизации, метрологии и сертификации. Госстандарт утверждает результаты испытаний средств измерений, организует работу по ведению государственных реестров средств измерений, реестров стандартных образцов, организует работы по сбору, анализу и публикации стандартных справочных данных. В области стандартизации и сертификации Госстандарт РФ осуществляет руководство Государственной службы времени, частоты и определения параметров вращения Земли (ГСВЧ) и службой стандартных образцов состава и свойств веществ и материалов.

В области стандартизации к функциям Госстандарта относятся:

- выполнение роли заказчика при разработке государственных стандартов, устанавливающих общетехнические требования;

- рассмотрение и принятие государственных стандартов;
- участие в работах по международному сотрудничеству.

Определенную законом деятельность Госстандарт РФ осуществляет при помощи около 20 научных центров и около 100 региональных центров стандартизации и метрологии. В функции научных центров (НИИ Госстандарта) входит хранение мер, разработка и совершенствование эталонной базы страны, разработка НТД по разделам метрологии, закрепленным за данным НИИ или центром, проведение испытаний по профилю института, разработка программ развития и совершенствования обеспечения единства измерений в какой-либо конкретной области. На институты Госстандарта как правило возлагается работа по руководству фундаментальными исследованиями в метрологии (программа ФИМ), выполнение международных работ по профилю института или центра.

Общее взаимодействие метрологических институтов РФ с Госстандартом РФ осуществляет Всероссийский институт метрологической службы (ВНИИМС, Москва). Здесь проводится координация всей конкретной метрологической деятельности в стране: распределяются обязанности между предприятиями Госстандарта, проводится экспертиза НТД, утверждаемой Госстандартом, ведется Госсреестр средств измерений. Кроме координирующих и распределяющих функций ВНИИМС занимается исследованиями в области метрологии геометрических величин, в измерениях давления, электрических величин, характеристик электромагнитной совместимости, в метрологических аспектах диагностики плазмы.

1.7. Направления работ по стандартизации в сфере информатизации

В России фонд действующих государственных стандартов в области информационных технологий включает более 300 стандартов, которые охватывают основные аспекты разработки информационных систем. Однако следует отметить, что по отдельным группам стандартов отсутствует комплексность охвата объектов стандартизации, а по ряду групп действующие стандарты требуют пересмотра с учетом современных требований.

Особого внимания требует расширение применения международных стандартов. Так, по состоянию на 1998 год по различным направлениям информационных технологий и смежных областей (вычислительная техника, системы связи и передачи данных, радиоэлектронные средства, открытые

системы, программная инженерия и др.) в России действовали свыше 700 государственных стандартов, обеспечивающих применение в стране около 400 международных стандартов. Это составляет всего 25% от общего числа международных стандартов ИСО, МЭК, МККТ, МККР, актуальных для применения в данной области.

Программой по стандартизации в сфере информатизаци предусматривается сотрудничество с международными организациями по стандартизации при проведении работ по трем приоритетным для Российской Федерации направлениям стандартизации информационных технологий, краткие сведения о которых мы приводим низке.

Направления 1-го приоритета

Языки программирования и программный интерфейс. Участие России в разработке международных стандартов по данной тематике позволяет поддерживать те направления российской математической школы, которые имеют традиционно устойчивую позицию, а также разрабатывать новые языки для перспективных направлений программирования.

Языки описания документов. Стандартизация в данной области позволяет обеспечить необходимую нормативную базу, поддерживающую создание, хранение и обращение документов в открытых системах, включая элементы доступа при поиске информации.

Программная инженерия. Данное направление стандартизации представляется особо важным для России в ближайшей перспективе. В сочетании с сертификацией и внедрением систем качества, соответствующих требованиям международных стандартов, участие в разработке и применении этой группы стандартов дает отечественным разработчикам, а также изготовителям и поставщикам программных средств возможность повысить качество и конкурентоспособность своей продукции как на внутреннем, так и на внешнем рынках.

Сервисы управления данными. Данное направление является перспективным в плане создания и развития отечественных систем распределенных баз данных и формирования национальных информационных ресурсов федерального, регионального и местного уровней в структуре Единого информационного пространства России.

Работа в сетях, и соответствующие соединения. Работы в данном направлении позволят стандартизовать функции, необходимые для установления и управления информационным обменом через сети и физические интерфейсы.

Безопасность информационных технологий. Работы в области безопасности информационных технологий позволяют создать комплект стандартов, поддерживающих методы и средства обеспечения безопасности, в первую очередь на уровне личности и различных общественных групп. Данное направление является одним из важнейших с учетом бурного роста информационного обмена между компонентами всех уровней и перспективы развития Российской и Глобальной информационной инфраструктуры, включая Интернет.

Терминология. Это направление предполагает разработку терминологии для информационных технологий и связанных областей.

Направления 2-го приоритета

Сбор данных и системы идентификации. Работы в данной области позволяют создать комплект стандартов, поддерживающих разработку идентификационных карт и соответствующих устройств для использования в межотраслевых приложениях и в международном обмене (например, как платежное средство в банковском деле), а также методы и средства для процесса автоматической идентификации и сбора данных, в частности с использованием штрихкодов.

Мультимедиа и представление информации. Стандартизация в данной области позволяет обеспечить необходимую нормативную базу, поддерживающую кодированное представление, обработку и обмен аудио, изображениями, мультимедиа и гипермедиа информацией для разнообразных приложений.

Пользовательский интерфейс. Работы в данной области позволяют создать комплект стандартов, поддерживающих пользовательский интерфейс для интерактивной деятельности в локальных и распределенных средах с использованием аудио, изображений, мультимедиа и гипермедиа информации, включая специальные интерфейсы для людей, имеющих физические недостатки или работающих в специфических условиях.

Офисное оборудование. Стандартизация в данной области позволяет обеспечить необходимую нормативную базу, поддерживающую адекватный уровень требований к эксплуатационным характеристикам и методам тестирования офисного оборудования (принтеры, копировальное оборудование, цифровые сканеры, факсимильное оборудование и их комбинации).

Кодированные наборы символов. Стандартизация в данной области позволяет обеспечить необходимую нормативную базу, поддерживающую множества графических символов и их кодированное представление для обеспе-

чения одно- и многоязыковых функций при работе с информацией (интернационализация).

Направления 3-го приоритета

Среды для информационного обмена. Работы в данной области включают стандарты, поддерживающие требования к оптическим и магнитным носителям данных и соответствующим устройствам на их основе, обеспечивающим хранение и обмен данными в системах обработки информации.

Геоинформационные технологии. Предусматривают развитие системы стандартов, направленных на повышение качества электронных карт и соответствие их требованиям международных стандартов, на сокращение трудоемкости и сроков изготовления электронных карт для создания предпосылок совместимости различных геоинформационных систем и в перспективе создания национальной базы геоинформационных данных.

Информационные технологии в охране здоровья. Для Российской Федерации это направление представляется одним из приоритетных, особенно если иметь в виду невысокий уровень здоровья и продолжительности жизни, а также проблемы, связанные с малой плотностью населения и низкой обеспеченностью врачами вне больших городов.

1.8. Классификация стандартов

В зависимости от возникновения

В зависимости от возникновения стандарты бывают двух видов:

- «де–факто»,
- «де-юре».

Стандарт «де–юре» создается формально признанной стандартизирующей организацией. Разрабатывается при соблюдении правил консенсуса в процессе открытой дискуссии.

Стандарты «де–факто» — термин, обозначающий продукт какого-либо поставщика, который захватил большую долю рынка и который другие поставщики стремяться эмулировать, копировать или использовать для того, чтобы захватить свою часть рынка. Недостаток — стандарты «де–факто» ставят пользователя в зависимое от производителей положение. Пример — стандарт SQL:

- 1970 — статья E. F. Codd. Relation Model of Data for Large Shared Data Banks. Communications of ACM, 1970
- 1970 – начало работ по созданию языка в лаболаториях IBM
- 1975–1980 — прототипы, основанные на SQL
- 1980 — появление первых коммерческих SQL–продуктов
- 1985 — работа комитета по стандартам над SQL
- 1986 — SQL-86
- 1989 — SQL-89
- 1990 — работа над SQL–2
- 1990 — работа над SQL–3
- 200? — ожидаемое принятие SQL–3

Стандарты на организацию ЖЦ

- Стандарты обеспечения качества
- Стандарты надежности
- Стандарты разработки ПО
- Стандарты тестирования
- Стандарты документирования
- Стандарты интерфейса
- Стандарты программирования
- Стандарты обмена данными

Стандарты и модели разработки

- RUP
- Tickit
- CMM
- Метод ORACLE
- IEEE Software Engineering Standarts
- ISO 12207

Глава 2

Стандарты на организацию жизненного цикла ПО

2.1. Понятие жизненного цикла ПО

В настоящее время просматривается тенденция в сторону увеличения объема работ, связанных с разработкой программного обеспечения. Таким образом, имеет смысл более подробно рассматривать технологический процесс разработки программных средств, жизненный цикл программного обеспечения.

В основе деятельности по созданию и использованию программного обеспечения лежит понятие жизненного цикла. В общем случае различают понятия жизненного цикла программного обеспечения и технологического процесса его разработки. Более четко различия между данными понятиями просматривается в отношении программных средств.

Жизненный цикл (ЖЦ) ПС — это период времени, который начинается с момента принятия решения о необходимости создания ПС и заканчивается в момент его полного изъятия из эксплуатации.

Существует несколько опубликованных моделей ЖЦ ПП. Но они в значительной степени подобны. Основные их отличия состоят в выделении наиболее важных с позиции авторов процессов, а так же способами их группирования. ЖЦ баз данных пока стандартами не регламентирован — существующие стандарты относятся только к ЖЦ программных средств.

Стандарты ЖЦ ПС могут использоваться как директивные, руководящие или рекомендательные документы. Наиболее полно ЖЦ, технология разработки и обеспечения качества ПС отражены в стандартах ISO. Стандарт ISO 12207:1995 — «Процессы жизненного цикла программных средств» — наиболее полно отражает архитектуру, работы, организацию и управление ЖЦ ПС.

Используемые реально в фирмах модели ЖЦ ПС в последнее время изменяются относительно приведенных в стандартах в связи с внедрением и развитием объектно–ориентированного анализа и методов быстрой разработки ПП, CASE–систем и языков четвертого поколения. В новых моделях сокращаются работы по непосредственному созданию программных компо-

нентов и детализируются работы по системному анализу и проектированию ПС и БД.

Вообще, при создании проектов ПС и обеспечении их ЖЦ целесообразно применять выборку из всей совокупности представленных стандартов (как международных, так и национальных), а имеющиеся пробелы в стандартизации заполнять стандартами де–факто и ведомственными нормативными документами.

Профиль — это совокупность нескольких базовых стандартов и других нормативных документов, предназначенная для реализации заданной функции или группы функций.

На базе одной и той же совокупности базовых стандартов могут формироваться различные профили для разных проектов. При сертификации информационных систем как специальный вид испытаний выделяют сертификацию на соответствие профилю. И здесь следует учитывать, что в международной стандартизации ИС принята жесткая трактовка понятия профиля — считается, что основой профиля могут быть только международные и национальные утвержденные стандарты (то есть не допускается использование стандартов де–факто и нормативных документов фирм).

Исходя уже из конкретного профиля планируется написание документации. Причем для каждого этапа жизненного цикла пишется своя документация, которая в свою очередь подразделяется на виды, в зависимости от того для каких специалистов она создается. ЖЦ ПО — это непрерывный процесс, который начинается с момента принятия решения о необходимости его создания и заканчивается в момент его полного изъятия из эксплуатации.

2.2. Стандарт ISO/IEC 12207

2.2.1. Структура стандарта

Основным нормативным документом, регламентирующим жизненный цикл (ЖЦ) ПО, является международный стандарт ISO/IEC 12207. Он определяет структуру ЖЦ, содержащую процессы, действия и задачи, которые должны быть выполнены во время создания ПО.

Основные термины:

- ПО или программный продукт — набор компьютерных программ, процедур и вязанной с ними документации и данных.
- Процесс — совокупность взаимосвязанных действий, преобразующих некоторые входные данные в выходные.

- Каждый процесс ЖЦ ПО имеет ответственного (заказчик, поставщик, разработчик и т.п.).
- Каждый из процессов состоит из ряда работ и решаемых при выполнении работ задач.
- Каждый процесс, действие или задача инициируется и выполняется другим процессом по мере необхдимости, причем нет заранее определенных последовательностей.

Структура ЖЦ ПО по стандарту ISO/IEC 12207 базируется на трех группах процессов:

1. основные процессы ЖЦ ПО:
   - приобретение,
   - поставка,
   - разработка,
   - эксплуатация,
   - сопровождение;
2. вспомогательные процессы, обеспечивающие выполнение основных процессов
   - документирование,
   - управление конфигурацией,
   - обеспечение качества,
   - верификация,
   - аттестация,
   - оценка,
   - аудит,
   - решение проблем;
3. организационные процессы
   - управление проектами,
   - создание инфраструктуры проекта,
   - определение,
   - оценка и улучшение самого ЖЦ,
   - обучение.

2.2.2. Основные процессы ЖЦ ПО

Приобретение

Действия заказчика, приобретающего ПС:

1. Инициирование приобретения
2. Подготовка заявочных предложений
3. Подготовка и корректировка договора
4. Надзор за деятельностью поставщика
5. Приемка и завершение работы

Поставка

Действия и задачи поставщика, который снабжает заказчика программным продуктом или услугой:

1. Инициирование поставки
2. Подготовка ответа на заявочное предложение
3. Подготовка договора
4. Планирование
5. Выполнение и контроль
6. Проверка и оценка
7. Поставка и завершение работы

Разработка

Действия и задачи, выполняемые разработчиком. Разработка включает в себя все работы по созданию ПО и его компонент в соответствии с заданными требованиями, включая оформление проектной и эксплуатационной документации, подготовку материалов, необходимых для проверки работоспособности и соответствующего качества программных продуктов, материалов, необходимых для организации обучения персонала и т.д.

Разработка ПО включает в себя три основные стадии: анализ; проектирование; реализацию (программирование).

Фаза разработки начинается с анализа осуществимости проекта, а далее происходит преобразование от требований пользователя в форму, доступную для реализации на ЭВМ. На эту фазу приходятся, как правило, 50% стоимости ПИ и 32% трудозатрат.

Разработка включает:

1. Подготовительную работу
2. Анализ требований к системе
3. Проектирование архитектуры системы
4. Анализ требований к ПС
5. Проектирование архитектуры ПС
6. Детальное проектирование ПС
7. Кодирование и тестирование ПС
8. Интеграцию ПС
9. Квалификационное тестирование
10. Интеграцию системы
11. Квалификационное тестирование системы
12. Установку ПС
13. Приемку ПС

Эксплуатация

Охватывает действия и задачи оператора организации, эксплуатирующей ПС. Процесс включает в себя следующие действия:

1. Подготовительная работа
2. Эксплуатационное тестирование
3. Эксплуатация системы
4. Поддержка пользователей

Эксплуатация начинается тогда, когда изделие передается пользователю, находится в действии и используется. Включает в себя работы по внедрению компонентов ПО в эксплуатацию, в том числе конфигурирование базы данных и рабочих мест пользователей, обеспечение эксплуатационной документацией, проведение обучения персонала и т.д., и непосредственно эксплуатацию, в том числе локализацию проблем и устранение причин их возникновения, модификацию ПО в рамках установленного регламента, подготовку предложений по совершенствованию, развитию и модернизации системы.

Сопровождение

Предусматривает действия, выполняемые сопровождающей организацией. Это внесение изменений в ПС в целях исправления ошибок, повышения производительности или адаптации к изменившимся условиям работы. Изменения, вносимые в ПС, не должны нарушать его целостность. Процесс сопровождения включает в себя следующие действия:

1. Подготовительная работа
2. Анализ проблем и запросов на модификацию ПС
3. Модификация ПС
4. Проверка и приемка
5. Перенос ПС в другую среду
6. Снятие ПС с эксплуатации

Фазу сопровождения также называют фазой продолжающейся разработки. Состоит из выявления и устранения ошибок в программах и изменения их функциональных возможностей. Практиками признано, что эта часть жизненного цикла (ЖЦ) должна приниматься во внимание с момента начала разработки с целью совершенствования ПИ в соответствии с потребностями пользователя. Процесс сопровождения, продолжатся собственно параллельно эксплуатации ПИ.

Структура трудозатрат на различные виды деятельности по сопровождению такова, что на изменение функциональных возможностей ПИ уходит около 78% времени, а на выявление ошибок — 17%.

2.2.3. Вспомогательные процессы ЖЦ ПО

Документирование

Процесс документирования представляет собой формализованное описание информации, созданной в течение ЖЦ ПО и состоит в разработке необходимых документов, предназначенных для руководителей, технических специалистов, пользователей и других заинтересованных лиц. Процесс документирования состоит из следующих работ:

- Подготовительная работа
- Проектирование и разработка
- Выпуск документации
- Сопровождение

Процесс управления конфигурацией

Управление конфигурацией является одним из вспомогательных процессов, поддерживающих основные процессы жизненного цикла ПО, прежде всего процессы разработки и сопровождения ПО. При создании проектов сложных ИС, состоящих из многих компонентов, каждый из которых может иметь разновидности или версии, возникает проблема учета их связей и функций, создания унифицированной структуры и обеспечения развития всей системы. Управление конфигурацией позволяет организовать, систематически учитывать и контролировать внесение изменений в ПО на всех стадиях ЖЦ.

Общие принципы и рекомендации конфигурационного учета, планирования и управления конфигурациями ПО отражены в проекте стандарта ISO 12207-2. Процесс управления конфигурацией состоит из следующих видов работ:

- Подготовительная работа
- Идентификация конфигурации
- Контроль конфигурации
- Учет состояния конфигурации
- Оценка конфигурации
- Управление выпуском и поставка

Процесс обеспечения качества

Под качеством ПО понимается совокупность свойств, которые характеризуют способность ПО удовлетворять заданным требованиям Процесс обеспечения качества позволяет обеспечить гарантии того, что ПО, процессы ЖЦ соответствуют заданным требованиям, утвержденным планам и состоит из следующих работ:

- Подготовительная работа
- Обеспечение качества продукта
- Обеспечение качества процесса
- Обеспечение прочих показателей качества системы

При этом могут использоваться стандарты качества серии ISO 9000.

Процесс верификации

Обеспечение качества проекта связано с проблемами верификации, проверки и тестирования ПО. Верификация — это процесс определения того, отвечает ли текущее состояние разработки, достигнутое на данном этапе, требованиям этого этапа. Проверка позволяет оценить соответствие параметров разработки с исходными требованиями. Проверка частично совпадает с тестированием, которое связано с идентификацией различий между действительными и ожидаемыми результатами и оценкой соответствия характеристик ПО исходным требованиям. В процессе реализации проекта важное место занимают вопросы идентификации, описания и контроля конфигурации отдельных компонентов и всей системы в целом.

Процесс аттестации

Этот процесс предусматривает определение полноты соответствия заданных требований и созданной системы или программного продукта их конкретному функциональному назначению. Под аттестацией подразумевают подтверждение и оценка достоверности проведенного тестирования. ПО должно соответствовать своим спецификациям, требованиям и документации. Аттестацию следует проводить с привлечением сторонних специалистов.

Процесс совместной оценки

Данный процесс состоит в оценке состояния работ по проекту и создаваемому ПО и заключается в контроле планирования и управления ресурсами, персоналом, инструментальными средствами проекта. Оценка может проводится в течение всего срока действия договора и может выполняться обеими сторонами, при этом одна сторона проверяет другую.

Процесс аудита

Аудит — это ревизия (проверка), проводимая компетентным органом (лицом) в целях обеспечения независимой оценки степени соответствия ПС или процессов установленным требованиям. Аудиторы не имеют прямую зависимость от разработчиков ПО. Они определяют состояние работ, использование ресурсов, соответствие документации спецификациям и стандартам, корректность тестирования и пр.

Процесс разрешения проблем

Данный процесс включает анализ и решение проблем (в том числе и обнаружение несоответствия) независимо от их происхождения или источника, которые обнаружены в ходе разработки, эксплуатации, сопровождения или других процессов. Каждая обнаруженная проблема должна быть определена, задокументирована, проанализирована и устранена.

2.2.4. Организационные процессы ЖЦ ПО

Процесс управления

состоит из действий и задач стороны, управляющей своими процессами. Менеджер отвечает за управление выпуском продукта, управление проектом, управление задачами соответствующих поцессов (таких, как приобретение, поставка и др.).

Управление проектом связано с вопросами определения области управления, планирования и организации работ, выполнения и контроля, создания коллективов разработчиков и контроля за сроками и качеством выполняемых работ.

Планирование, в частности, подразумевает составление графиков выполнения работ, оценку затрат, выделение требуемых ресурсов, распределение ответственности, оценку рисков, создание инфраструктуры управления.

Техническое и организационное обеспечение проекта включает выбор методов и инструментальных средств для реализации проекта, определение методов описания промежуточных состояний разработки, разработку методов и средств испытаний ПО, обучение персонала и т.п.

Каждый процесс характеризуется определенными задачами и методами их решения, исходными данными, полученными на предыдущем этапе, и результатами. Результатами анализа, в частности, являются функциональные модели, информационные модели и соответствующие им диаграммы.

ЖЦ ПО носит итерационный характер: результаты очередного этапа часто вызывают изменения в проектных решениях, выработанных на более ранних этапах.

2.3. Модели жизненного цикла программного обеспечения

Под моделью ЖЦ ПС понимается структура, определяющая последовательность выполнения и взаимосвязь процессов, действий и задач на протяжении ЖЦ. Модель ЖЦ зависит от: спецификации, масштабов, условий.

Модель ЖЦ определяет характер процессов его создания, который представляет собой совокупность упорядоченных во времени, взаимосвязанных и объединенных в стадии работ, выполнение которых необходимо и достаточно для создания ПС, соответствующего заданным требованиям. Стадия создания ПС — часть процесса создания ПС, ограниченного некоторыми временными рамками и заканчивающегося созданием конкретного продукта, определенного заданными для данной стадии требованиями.

Существует несколько моделей жизненного цикла. Традиционно выделяют следующие основные этапы жизненного цикла:

1. стратегическое планирование

2. анализ требований

3. проектирование ПО

4. программирование

5. тестирование и отладка

6. эксплуатация и сопровождение

Каждому этапу соответствуют определенный результат и набор документации, являющейся исходными данными для следующего этапа. В заключение каждого этапа производится верификация документов и решений с целью проверки их соответствия первоначальным требованиям заказчика.

Исторически, в ходе эволюционного развития теории проектирования программного обеспечения и по мере его усложнения, сложились три основные модели жизненного цикла. Эти модели выражают последовательность этапов ЖЦ ПО.

До 80-х годов имела место **каскадная модель ЖЦ**, подразумевавшая переход на последующие этапы ЖЦ только после полного окончания работ на предыдущих этапах. С развитием вычислительной техники, в середине 80-х, сложность и объемы программного обеспечения существенно возросли. В связи с этим возникли проблемы с разработкой и отладкой ПО. Продумать все шаги разработки нового ПО, наметить этапы проектирования и предусмотреть все варианты поведения при отладке программного обеспечения стало не под силу одному разработчику. Каскадная модель ЖЦ стала существенно сдерживать темпы создания сложных программных систем. Процесс отладки, при этом, затягивался и не давал гарантий безошибочной работы программ.

На смену каскадной модели, жестко регламентирующей последовательность этапов и критерии переходов между ними, пришла **поэтапная модель с промежуточным контролем**. Это итерационная модель разработки ПО с обратными связями между этапами. Проверки и корректировки разрабатываемой ИС проводятся на каждом из этапов, что позволяет существенно снизить трудоемкость отладки по сравнению с каскадной моделью. Итерационность модели проявляется в обработке ошибок выявленных промежуточным контролем. Если на каком-либо из этапов в ходе промежуточной проверки обнаружилась ошибка, допущенная на более ранней стадии разработки, работы этапа, повлекшего ошибку, необходимо провести повторно. При этом, анализируются причины ошибки и корректируются, по необходимости, исходные данные этапа или перечень проводимых работ. Аналогичная ситуация возникает если в ходе разработки ИС возникают новые требования заказчика или изменяются какие-либо условия функционирования ИС.

Спиральная модель поддерживает итерации поэтапной модели, но особое внимание уделяется начальным этапам проектирования: анализу требований, проектированию спецификаций, предварительному проектированию и детальному проектированию. Каждый виток спирали соответствует поэтапной модели создания фрагмента или версии ПО, уточняются цели и требования к ПО, оценивается качество разработанного фрагмента или версии

ПО и планируются работы следующего витка. Таким образом, углубляются и конкретизируются все детали проектируемого ПО, и в результате получается вариант, который удовлетворяет всем требованиям заказчика. Особенностью спирального подхода является то, что прикладное ПС создается не сразу, а по частям, с использованием метода прототипирования. Под прототипом понимается действующий программный компонент, реализующий отдельные функции и внешние интерфейсы разрабатываемого ПС. Создание прототипов осуществляется в несколько витков спирали. Каждый виток соответствует созданию фрагмента или версии ПС, где уточняются цели и характеристики проекта, оценивается качество и планируются работы следующей итерации.

Количество, состав и последовательность этапов ЖЦ для каждого конкретного ПО определяется на ранних стадиях планирования создания ПО. При этом учитываются особенности эксплуатации, наличие разного рода ограничений, численность и квалификация персонала разработчиков и эксплуатационников, а также множество других факторов.

Как было отмечено выше, жизненные циклы систем, процессов их разработки, эксплуатации и сопровождения регламентированы в стандартах. При этом стандарты, разрабатываемые международными организациями в той или иной области деятельности, носят рекомендательный характер и не возведены в ранг закона. Руководящие принципы, определенные в стандартах, имеют официальную силу тогда, когда приняты правительством той или иной страны.

2.4. Стандарт ГОСТ 34.601

Стандарт ГОСТ 34.601 распространяется на автоматизированные системы (АС), представляющие собой организационно-технические системы, обеспечивающую выработку решений на основе автоматизации информационных потоков в различных видах деятельности (исследование, проектирование, управление и т.п.), включая их сочетания, создаваемые в организациях, объединениях и на предприятиях.

Данный стандарт устанавливает стадии и этапы создания АС. В процессе разработки АС создают, в общем случае, следующие виды обеспечения: техническое, программное, информационное математическое и др.

Проектные решения по программному, техническому и информационному обеспечению реализуют как изделия в виде взаимоувязанной совокупности компонент и комплексов, входящей в состав АС с необходимой докумен-

тацией.

Справочное приложение к ГОСТ 34.601 устанавливает, что программное обеспечение АС — совокупность программ на носителях информации с программной документацией по ГОСТ 19.101. Таким образом, при разработке программного обеспечения можно пользоваться как стандартом ГОСТ 19.102, так и ГОСТ 34.601.

Процесс создания АС представляет собой совокупность упорядоченных во времени, взаимосвязанных, объединенных в стадии и этапы работ, выполнение которых необходимо и достаточно для создания АС, соответствующей заданным требованиям.

Стадии и этапы создания АС выделяются как части процесса создания по соображениям рационального планирования и организации работ, заканчивающих заданным результатом. Как и в стандарте ГОСТ 19.102, работы по развитию АС осуществляются по стадиям и этапам, применяемым для создания АС. Состав и правила выполнения работ на установленных настоящим стандартом стадиях и этапах определяют в соответствующей документации организаций, участвующих в создании конкретных видов АС.

Стадии и этапы создания АС в общем случае:

1. Формирование требований к АС
    - Обследование объекта и обоснование необходимости создания АС
    - Формирование требований пользователя к АС
    - Оформление отчета о выполненной работе и заявки на разработку АС (тактико–технического задания)
2. Разработка концепции АС
    - Изучение объекта
    - Проведение необходимых научно– исследовательских работ
    - Разработка вариантов концепции АС и выбор варианта концепции АС, удовлетворяющего требованиям пользователя
    - Оформление отчета о выполненной работе
3. Техническое задание
    - Разработка и утверждение технического задания на создание АС
4. Эскизный проект

- Разработка предварительных проектных решений по системе и ее частям
- Разработка документации на АС и ее части

5. Технический проект

- Разработка проектных решений по системе и ее частям
- Разработка документации на АС и ее части
- Разработка и оформление документации на поставку изделий для комплектования АС и/или технических требований (технических заданий) на их разработку
- Разработка заданий на проектирование в смежных частях проекта объекта автоматизации

6. Рабочая документация

- Разработка рабочей документации на систему и ее части
- Разработка или адаптация программ

7. Ввод в действие

- Подготовка объекта автоматизации к вводу АС в действие
- Подготовка персонала
- Комплектация АС поставляемыми изделиями (программными и техническими средствами, программно-техническими комплексами, информационными изделиями)
- Строительно–монтажные работы
- Пусконаладочные работы
- Проведение предварительных испытаний
- Проведение опытной эксплуатации
- Проведение приемочных испытаний

8. Сопровождение АС

- Выполнение работ в соответствии с гарантийными обязательствами
- Послегарантийное обслуживание

Стандартом ГОСТ 34.601 допускается исключать стадию «Эскизный проект» и отдельные этапы работ на всех стадиях, объединять стадии «Технический проект» и «Рабочая документация» в одну стадию «Технорабочий проект».

В зависимости от специфики создаваемых АС и условий их создания допускается выполнять отдельные этапы работ до завершения предшествующих стадий, параллельное во времени выполнения этапов работ, включение новых этапов работ.

Глава 3

Документирование в процессах жизненного цикла ПО

3.1. Документация и ее роль в обеспечении качества

Документирование ПО определяется стандартами ГОСТ серии 19, 34, ГОСТ Р ИСО/МЭК 8631, ГОСТ Р ИСО 9127, ГОСТ Р ИСО/МЭК ТО 9294.

3.1.1. ГОСТ Р ИСО/МЭК ТО 9294-93 ИТ.

Руководство по управлению документированием ПО является одним из основных стандартов в данной области и представляет собой руководство по документированию ПО для тех руководителей, которые отвечают за разработку программного обеспечения. Данное руководство предназначено для помощи руководителям в обеспечении эффективного проведения документирования в организациях. Данный стандарт направлен на определение стратегий, стандартов, процедур, ресурсов и планов, которыми должны заниматься сами руководители для того, чтобы эффективно управлять документированием ПО.

Руководство по управлению документированием ПО предназначено для применения ко всем типам программного обеспечения — от простейших программ до наиболее сложных систем программного обеспечения, охватывающих все типы программной документации, относящейся ко всем стадиям жизненного цикла ПО.

Принципы управления документированием программного обеспечения одинаковы для любого объема проекта. Для небольших проектов значительную часть положений, приведенных в данном стандарте, можно не применять, но принципы остаются теми же. Руководители могут адаптировать данные рекомендации для своих конкретных потребностей.

Эффективность выполнения руководящей роли можно рассматривать как основанную на следующих трех элементах.

1. Руководящая обязанность по документированию — требует признания того, что программная документация важна и что ее следует планиро-

вать, описывать, проверять, утверждать, выпускать, распространять и сопровождать.

2. Руководящая поддержка обязанностей персонала по документированию — требует руководство и стимулирование персонала при проведении требуемого документирования и обеспечении его ресурсами для содействия в данной работе.

3. Признаки руководящих обязанностей и поддержки — требует обеспечить:

   - опубликованные официальные отчеты о стратегии документирования;
   - стандарты и руководства, определяющие все аспекты документирования ПО;
   - опубликованные процедуры документирования;
   - выделение соответствующих ресурсов для документирования;
   - планирование документирования, осуществляемое как неотъемлемая часть процесса разработки ПО;
   - постоянную проверку, осуществляемую для обеспечения соответствия со стратегией, стандартами, процедурами и планами по документированию.

В ГОСТ Р ИСО/МЭК ТО 9294-93 программная документация рассматривается как имеющая следующие шесть основных функций.

1. *Информация для управления.* Во время разработки ПО администрации необходимо оценивать ход работы, возникающие проблемы и вероятности развития процесса разработки ПО. Периодические отчеты, согласно которым проверяют ход работ по графику и представляют планы на следующий период, обеспечивают контрольные механизмы и обзор проекта.

2. *Связь между задачами.* Большинство проектов разработки ПО разделяется на задачи, выполняемые различными группами: специалисты в предметной области, аналитики, проектировщики, специалисты по обеспечению качества и др. Этим группам, а также персоналу в пределах группы, необходимы средства общения друг с другом, обеспечивающие

информацию, которую можно воспроизводить, распространять и на которую можно ссылаться. Большинство методологий разработки ПО устанавливают официальные документы для связи между задачами. Например, аналитики представляют официальные спецификации требований проектировщикам, а они, в свою очередь, официальные проектные спецификации программистам.

3. *Обеспечение качества.* Требуется документация разработки и документация продукции для выполнения задач, связанных с обязанностями по обеспечению качества ПО.

4. *Инструкции и справки.* Документация, требующаяся операторам, пользователям, руководителям и другим заинтересованным лицам для того, чтобы понимать и использовать ПО.

5. *Сопровождение ПО.* Специалистам, сопровождающим ПО, требуется детальное описание ПО, такое, чтобы они могли локализовать и корректировать ошибки и модернизировать или изменять ПО соответствующим образом.

6. *Исторические справки.* Документация, требуемая в качестве исторической справки по проекту. Данная документация может помочь в переносе и переводе ПО в новое окружение.

Рассмотрим стратегии документирования. Стратегии документирования, подготовленные и контролируемые руководителем организации, обеспечивают руководящими положениями ответственных лиц, принимающих решения на всех нижних уровнях. Стратегия обеспечивает главное направление, но не дает рекомендаций, что делать и как это делать. Из–за существенной роли, которую играет документация на всех стадиях ЖЦ ПО, стратегия должна быть официально утверждена, доведена до каждого исполнителя проекта. Официальная стратегия устанавливает дисциплину, требуемую для эффективного документирования ПО.

Стратегия должна поддерживать основные элементы эффективного документирования:

- требования документации охватывают весь жизненный цикл ПО;
- документирование должно быть управляемым;
- документация должна соответствовать ее читательской аудитории;

- работы по документированию должны быть объединены в общий процесс разработки ПО;
- должны быть определены и использованы стандарты по документированию;
- должны быть определены средства поддержки.

Внутри организации, помимо стратегии, должны быть приняты стандарты и руководства для:

- модели жизненного цикла ПО;
- типов и взаимосвязей документов;
- содержания документа;
- качества документа;
- форматов документа;
- обозначение документа.

Данные стандарты и руководства будут определять, как следует выполнять задачи документирования, и будут обеспечивать критерии для оценки полноты, полезности и соответствия программной документации, создаваемой в организации.

Большинство стандартов и руководств выдают рекомендации, которые применимы на общем уровне. Зачастую будут требоваться управленческие решения для адаптации общих рекомендаций к конкретным проектам. Применение стандартов, распространяющихся на организацию документирования, облегчит руководителям проекта определение следующих вопросов: какие типы документов требуются? каков объем представляемой документации? что документы содержат? какой уровень качества будет достигнут? где документы будут созданы? как документы будут хранить, сопровождать и обращать?

В организации должны быть установлены процедуры для применяемых стратегий документирования. Процедуры определяют последовательность документирования: планирование, подготовка, конфигурационное управление, проверка, утверждение, производство, хранение, дублирование, распространение и модернизация, продажа. Кроме того, процедуры должны определять контрольные пункты и методы обеспечения качества.

Основными ресурсами, требуемыми для документирования, являются следующие: персонал, средства, финансирование.

Персонал. Для процесса разработки ПО необходимы люди со знанием программирования, сути предмета, документирования, проектировщики, специалисты в предметной области и другие. Важно, чтобы все специалисты, участвующие в разработке проекта были обучены методам документирования и полностью понимали и выполняли свою роль в документировании.

Средства. Важно предусмотреть обеспечение задач документирования соответствующими и подходящими средствами. Инструментальные средства полезны для подготовки и контроля документации. Они могут быть применены для повышения эффективности многих процессов документирования и использования стандартов данной организации, распространяющихся на документирование.

Финансирование. Важно, чтобы стоимость документирования определяли как отдельные статьи бюджета, так как она нередко составляет значительную часть стоимости разработки ПО.

Процесс документирования, как уже отмечалось выше, должен планироваться. План документирования определяет, что должно быть сделано, как это должно быть сделано, когда это должно быть сделано и кто это должен делать.

План документирования может быть как частью общего плана проекта, так и как отдельным документом. Он должен быть доведен до всех участников проекта. План документирования включает в себя изложение общей структуры документации, типов, содержания, качества, форматов, обозначения, комплектности и хранения документов, их обращения и графика документирования.

График документирования должен распределять время для:

- планирования документов;
- проверки плана и принципов документирования;
- подготовки проектов и проверки их на техническую точность, полноту и соответствие;
- редактирование при внесение комментариев, появившихся при проверке;
- проведения согласования;
- перевода;
- распространения.

Планирование документирования следует начинать заранее, и план необходимо проверять на всем протяжении проекта. Подобно любому плану, план документирования отражает намечаемые действия и является объектом для необходимых изменений. В проекте должны быть предусмотрены регулярные проверки результативности изменений в плане.

3.2. Определение типов и содержания документов

Программные документы можно представить разделенными на три категории:

- документация разработки;
- документация продукции;
- документация управления проектом.

Данная схема не является исчерпывающей и окончательной, но служит контрольной таблицей основных типов программных документов, которые руководители должны предусмотреть, когда определяют стандартные типы своих документов.

3.2.1. Документация разработки

Документы, описывающие процесс разработки ПО, определяют требования, которым должно удовлетворять ПО, определяют проект программного обеспечения, определяют, как его контролируют и как обеспечивают его качество. Документация разработки также включает в себя подробное описание ПО (программную логику, программные взаимосвязи, форматы и хранения данных и т.д.). Разработка документов преследует пять целей:

1. они являются средством связи между всеми участниками проекта и описывают подробности решений, принятых относительно требований к ПО, проекту, программированию и тестированию;
2. они описывают обязанности группы разработки и определяют, кто, что и когда делает, учитывая роль программного обеспечения, предмета работ, документации, персонала, обеспечивающего качество, и каждого вовлеченного в процесс разработки;

3. они выступают, как контрольные пункты, которые позволяют руководителям оценивать ход разработки (если документы разработки отсутствуют, неполны или устарели, то руководители проекта теряют важное средство для отслеживания и контроля проекта);

4. они образуют основу документации сопровождения ПО, требуемой лицам, сопровождающим ПО, как часть документации продукции;

5. они описывают историю разработки ПО.

Типовыми документами разработки являются: анализы осуществимости и исходные заявки; спецификации требований и функций; проектные спецификации, включая спецификации программ и данных; планы разработки, сборки и тестирования ПО; планы обеспечения качества, стандарты и графики; защитная и текстовая информация.

3.2.2. Документация продукции

Документация продукции обеспечивает информацию, необходимую для эксплуатации, сопровождения, модернизации, преобразования и передачи программной продукции.

Документация продукции преследует три цели:

1. обеспечение учебной и справочной информацией для любого, использующего или эксплуатирующего ПО;

2. облегчение сопровождения и модернизации ПО программистам, не разрабатывавших ПО;

3. помощь при продаже или приемке ПО.

Документация продукции должна включать в себя материалы для следующих типов читателей: пользователей, операторов, сопровождающих программистов. Кроме того, данная документация может включать в себя руководства и материалы для руководителей, вспомогательные материалы, общую информацию.

Типовые документы продукции включают в себя:

- учебные руководства;

- справочные руководства и руководства пользователя;

- руководства по сопровождению ПО;

- брошюры и информационные листки, посвященные продукции.

Рассмотрим стандарт ГОСТ Р ИСО 9127-94 Системы обработки информации. Документация пользователя и информация на упаковке для потребительских пакетов вводит определение документации пользователя, как документации, которая обеспечивает пользователей информацией, необходимой для установки и прогона ПО и является обязательной в поставке (документация пользователя выполняется в виде одного или нескольких руководств и вкладывается вместе с программным продуктом внутрь упаковки).

Данный стандарт определяет три категории информации:

1. Обязательная — информация, поставляемая с каждым пакетом;

2. Условная — информация, поставляемая с каждым пакетом, для которого она необходима;

3. Факультативная — информация, поставляемая с каждым пакетом, по усмотрению изготовителя или торгующей организации.

Назначением документации пользователя является обеспечение конечного пользователя достаточной информацией для ясного понимания:

- цели, функций и характеристик ПО;

- того, как ввести в действие и использовать ПО;

- договорных прав и обязанностей.

Документация пользователя должна включать в себя справочную документацию для повседневного использования программы. Дополнительно может быть выполнена учебная документация. В качестве справочной документации выступают обозначение пакета, компоненты пакета, функциональное описание ПО, ввод в действие ПО, использование ПО, техническая информация о ПО, тестирование, договорная информация, словарь, указатель, замечания конечных пользователей.

Таким образом, ГОСТ Р ИСО 9127, не имея формальных ссылок на ГОСТ Р ИСО/МЭК ТО 9294, фактически дополняет введенное в нем понятие документации продукции.

3.2.3. Документация управления проектом

Документы создаются на основе информации управления проектом, такой как:

- графики для каждой стадии процесса разработки и отчеты об изменениях графиков;
- отчеты о согласованных изменениях ПО;
- отчеты о решениях, связанных с разработкой;
- распределение обязанностей.

Данная документация обеспечивает информацию, относящуюся, с точки зрения руководства, к долговечности продукции.

3.3. Требования стандартов к программной документации

Как уже было отмечено, качество программного обеспечения, наряду с другими факторами, определяется полнотой и качеством пакета документов, сопровождающих ПО. К программным документам относятся документы, содержащие сведения, необходимые для разработки, изготовления, сопровождения программ и эксплуатации.

19-я система стандартов (Единая система программной документации - ЕСПД) устанавливает следующие виды программной документации.

1. *Спецификация.* Состав программы и документация на нее.
2. *Ведомость держателей подлинников.* Перечень предприятий, на которых хранят подлинники программных документов.
3. *Текст программы.* Запись программы с необходимыми комментариями.
4. *Описание программы.* Сведения о логической структуре и функционировании программ.
5. *Программа и методика испытаний.* Требования, подлежащие проверке при испытании программы, а также порядок и методы контроля.

6. *Техническое задание.* Назначение и область применения программы, технические, технико–экономические и специальные требования, предъявляемые к программе, необходимые стадии и сроки разработки, виды испытаний.

7. *Пояснительная записка.* Схема алгоритма, общее описание алгоритма и функционирования программы, а также обоснование принятых технических и технико–экономических решений.

8. *Эксплуатационные документы.* Сведения для обеспечения функционирования и эксплуатации программы.

3.3.1. Стандарт ГОСТ 19.201-78

Содержание технического задания на разработку программных продуктов должно соответствовать ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению. Помимо разработки технического задания на все ПО могут разрабатываться технические задания на этапы, например, техническое задание на выполнение НИР.

Согласно ГОСТ 19.201-78 техническое задание на разработку ПО должно включать следующие разделы:

1. введение;

2. основания для разработки;

3. назначение разработки;

4. требования к программе;

5. требования к программной документации;

6. технико–экономические показатели;

7. стадии и этапы разработки;

8. порядок контроля и приемки;

9. приложения.

В зависимости от особенностей разрабатываемого ПО стандарт допускает уточнение содержания разделов, введение новых разделов или их объединение.

В разделе *Введение* указывается наименование, краткая характеристика области применения ПО.

В разделе *Основания для разработки* указывается: документ (документы), на основание которых ведется разработка; организация, утвердившая документ, и дата утверждения; наименование (условное обозначение) темы разработки.

В разделе *Назначение разработки* должно быть указано функциональное и эксплуатационное назначение ПО.

В раздел *Требования к программе* включаются следующие подразделы.

1. Требования к функциональным характеристикам, в котором указываются требования к составу выполняемых функций, организации входных и выходных данных, временным характеристикам и т.д. В данном подразделе описывается поведение ПО с точки зрения соотношения входа и выхода, без конкретизации его внутренней структуры. Описание выполняемых функций делается либо в текстовом виде, либо на специальном графическом языке. Описание входа заключается в фиксации синтаксиса и семантики всех входных данных. Описание выхода должно содержать точное описание всех возможных выходных данных в тесной взаимосвязи с входными.

2. Требования к надежности, где указываются требования к обеспечению надежного функционирования ПО, его защите (контроль входной и выходной информации, описание последствий отказов ПО и т.д.).

3. Условия эксплуатации, в котором должны быть указаны характеристики операционной среды, вид обслуживания, необходимое количество и квалификация персонала и др., а также допустимые параметры окружающей среды (относительная влажность, температура и др.).

4. Требования к составу и параметрам технических средств - необходимый состав технических средств (конфигурация) с указанием их основных технических характеристик.

5. Требования к информационной и программной совместимости, в котором указываются требования к информационным структурам на входе и выходе и методам решения, исходным кодам, языкам программирования и программным средствам, используемым ПО. Кроме того, могут указываться протоколы межмашинного сетевого обмена данными, стандарты протоколов формализации данных и управления терминалами, стандарты и форматы сообщений, протоколы транзакций, протоколы запросов

данных, стандарты представления данных, требования к СУБД и операционным системам.

6. Требования к маркировке и упаковке.

7. Требования к транспортированию и хранению.

В разделе *Требования к программной документации* должен быть указан предварительный состав программной документации и, при необходимости, специальные требования к ней.

В разделе *Технико-экономические показатели* указывается: ориентировочная экономическая эффективность, предполагаемая годовая потребность, экономические преимущества разработки по сравнению с лучшими отечественными и зарубежными образцами или аналогами.

В разделе *Стадии и этапы разработки* устанавливают необходимые стадии разработки, этапы и содержание работ (перечень документов, которые должны быть разработаны, согласованы и утверждены), а также сроки разработки и исполнителей.

В разделе *Порядок контроля и приемки* должны быть указаны виды испытаний и общие требования к приемке ПО. Здесь фиксируют важнейшие характеристики ПО в некоторой количественной или иной достаточно простой форме, с тем, чтобы можно было установить степень соответствия готового ПО принятым техническим условиям.

В приложениях к техническому заданию при необходимости приводят: перечень научно–исследовательских и других работ, обосновывающих разработку; схемы алгоритмов, таблицы, описания, обоснования, расчеты и другие документы, которые могут быть использованы при разработке; другие источники разработки.

3.3.2. Стандарт ГОСТ 34.602-89

Техническое задание на создание АС разрабатывается в соответствии с ГОСТ 34.602-89. Данный стандарт устанавливает следующие разделы, включаемые в техническое задание.

1. Общие сведения, включающие полное наименование системы, условное обозначение системы, шифр темы (шифр (номер) договора), наименование предприятий разработчика и заказчика системы и их реквизиты, перечень документов, на основании которых создается система, плановые сроки начала и окончания работ по созданию АС, сведения об источниках и порядке финансирования работ.

2. Назначение и цели создания АС, в котором указывают назначение системы и цели ее создания..

3. Характеристика объекта автоматизации.

4. Требования к системе. Данный раздел состоит из следующих подразделов.

   - Требования к системе в целом. Здесь указывают перечень подсистем, их назначение и основные характеристики, требования к числу уровней иерархии и степени централизации системы, требования к способам и средствам связи для информационного обмена между компонентами системы, требования к характеристикам взаимосвязей АС со смежными системами, требования к ее совместимости, способы обмена информации. Кроме того, требования к численности и квалификации персонала и режиму его работы, к надежности, безопасности и т.д..
   - Требования к функциям.
   - Требования к видам обеспечения (математическому, информационному, лингвистическому программному, техническому организационному и т.д.).

5. Состав и содержание работ по созданию (развитию) АС.

6. Порядок контроля и приемки системы.

7. Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу в действие.

8. Требования к документированию.

9. Источники разработки.

Глава 4

Разработка требований, внешнее и внутренне проектирование программных средств

4.1. Определение требований к ПИ

В процессе разработки требований происходит следующее: проводятся поисковые, исследовательские работы; формируется комплекс требований, выражающий потребности пользователя в конкретном ПИ; будущий комплекс программ тщательно анализируется с учетом выполняемых им функций и основных свойств, обосновывается целесообразность их разработки; предварительно оцениваются трудовые и стоимостные затраты и сроки создания; вырабатываются рекомендации по выбору инструментальных средств и методов, которые предполагается использовать в процессе разработки программ; формирование требований к качеству программ в соответствии с условиями их функционирования и реализации конкретных функций.

Выполнение этих работ в процессе формирования требований позволят предотвратить дополнительные расходы, вызванные модификацией программ при их внедрении и сопровождении. Главной особенностью требований является то, что они отражают нужды организации пользователя. В связи с этим можно выделить несколько групп программных проектов:

- управляемые пользователем;
- утверждаемые пользователем;
- 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 пол-

ную ответственность за работу. При этом для более качественного выполнения работ на начальной стадии проектирования целесообразно привлекать к формулировке требований потенциальных пользователей. Целью сформулированных требований является отражение потребностей пользователей в конкретном программном изделии.

Наиболее полное отражение потребностей можно предположить в проекте, управляемом пользователем. Здесь лучше всего сформулированы требования к программному изделию. Однако если проектировщики не привлекаются к выработке требований, увеличивается вероятность неправильной интерпретации или ошибочного понимания уже утвержденных требований. Поэтому наиболее оптимальной является совместная работа проектировщиков и пользователей по выработке требований.

Требования к программному изделию должны разрабатываться небольшой группой лиц. Например, в группу по определению требований должны входить

1. от организации-пользователя:
   - представитель, наделенный правом принятия окончательных решений,
   - представитель, который будет принимать непосредственное участие в применении проектируемого программного изделия.
2. от организации-разработчика
   - специалист, который будет играть главную роль в процессе внешнего проектирования программного изделия,
   - специалист, принимающий затем непосредственное участие в одном из процессов внутреннего проектирования.

Многие проблемы разработки программного изделия возникают в результате того, что требования к ПИ были плохо поняты или неточно заданы. Это вызвано несовпадением уровней подготовки пользователя и разработчиков.

Иногда пользователи пытаются определить свои требования в таких формах, которые относятся к функциям разработчиков. Поэтому здесь требуется установка ограничений. Требования должны однозначно определять конечный продукт разработки и метод разработки, но не подход к проектированию.

Необходимо делать различие между жесткими требованиями и требованиями, которые не являются строго обязательными.

Требования анализируются на протяжении всего времени разработки проекта. Можно установить две фазы в выработке требований.

1. Первой является фаза планирования, на которой в процессе взаимодействия пользователей и разработчиков определяется

   - реализуемость,
   - устанавливаются цели,
   - оцениваются затраты,
   - обеспечивается ориентация для разработки проекта.

2. Второй фазой является фаза выработки требований пользователя. На этой фазе вырабатываются требования к:

   - входным данным,
   - информационным потокам,
   - выходным данным,
   - документации,
   - среде,
   - вычислительным ресурсам.

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

1. достаточным для идентификации целей ПИ, его среды, преимуществ и недостатков ПИ для пользователя, состава и конфигурации ресурсов для его работы;

2. достаточно полным, чтобы в последующем при разработке исключить серьезные модификации и пересмотр требований;

3. 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. Ответственным за полноту и точность сформулированных требований к ПИ является пользователь. Проектировщик отвечает за качество описания требований и их реализацию.

4.2. Определение целей создания ПИ

На этом этапе устанавливаются взаимосогласованные цели создания ПИ. Это связано с тем, что некоторые цели имеют противоречивый характер, и необходимо найти компромиссное решение: установить, какие из них более важны при разработке ПИ, а какими можно пренебречь для достижения более важных целей. При описании целей возможно возникновение следующих ошибок:

- противоречивость в описании сформулированных целей;
- наличие поверхностно выявленных целей, не отражающих специфических особенностей разрабатываемого ПИ;
- цели создания ПИ с точки зрения пользователя (цели продукта) и цели проекта с точки зрения проектировщика противоречивы.

В общем случае цели разработки ПИ могут быть сгруппированы в десять достаточно самостоятельных категорий, а именно:

1. Универсальность (общность),
2. Человеческие факторы,
3. Адаптируемость,
4. Сопровождаемость,
5. Безопасность,
6. Документация,
7. Стоимость,
8. Календарный план,
9. Производительность (эффективность),
10. Надежность.

Некоторые из перечисленных целей согласуются между собой, например человеческие факторы и надежность, адаптируемость и надежность.

Однако другие цели вступают в противоречие, например универсальность и стоимость, стоимость и надежность, и в процессе разработки приходится

принимать компромиссное решение для их согласования. 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;
- последствия сбоев системы и наиболее важных функций;
- допустимый объем данных, утрачиваемых во время сбоя, и уровень обеспечения безопасности;
- число ошибок ПИ, категории сложности ошибок и время их обнаружения;
- функции, необходимые для обнаружения и исправления ошибок, а также обеспечение устойчивости к ним;
- возможности обнаружения ошибок пользователя и аппаратуры, а также восстановления работоспособности.

Цели проекта — это цели, которые должны быть достигнуты в процессе проектирования. Они не проявляются явно в ПИ, но тем не менее должны быть официально установлены. В случаях, когда разработчики не имеют списка целей проекта, они получают противоречивые и неожиданные результаты.

Цели проекта должны содержать следующую информацию:

1. Стоимостные ограничения;
2. Календарный план выполнения работ;
3. Задачи каждого этапа тестирования;
4. Цели, указывающие степень адаптируемости или расширяемости, которая должна быть достигнута;
5. Цели в области сопровождаемости, которые необходимо учитывать при разработке;
6. Уровни надежности, которые должны быть достигнуты на каждом этапе разработки для получения заданной надежности ПИ;
7. Документирование в процессе разработки;
8. Критерии завершения разработки и начала эксплуатации.

Цели проекта должны быть ясными, обоснованными и измеримыми, а также известными как пользователям, так и разработчикам. Они должны быть реальными и по возможности выражаться в количественном измерении для последующей проверки достижения поставленных целей. Каждая цель должна быть сформулирована достаточно подробно, как того требуют процессы проектирования, но не должна предполагать конкретных проектных решений. Между целями необходимо определить зависимости, чтобы при изменении некоторой цели проектировщик мог определить, как это сказывается на других целях.

Следует также указывать важность целей, т.е. достижение каких целей в процессе разработки обязательно, а каких желательно, но не обязательно, и их приоритет. Это необходимо знать разработчику в моменты, когда приходится принимать компромиссные решения. После формулировки цели она должна быть сопоставлена с требованиями, чтобы убедиться, что все требования переведены в цели проекта.

Для проверки соответствия требований и целей привлекаются представители нескольких уровней руководства как в организации–пользователе, так и в организации-разработчике. Обязательно и участие в ней разработчиков требований и специалистов, выполняющих затем внешнее проектирование.

4.3. 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:

- доведение до минимума ошибок пользователя;
- обнаружение ошибок пользователя в случае их возникновения;
- 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;
- 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. Однако иногда бывает необходимо указать некоторые идеи относительно внутреннего проектирования функции, что и делается в этом разделе. Практиковать это следует как можно реже. 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.

Известен ряд методов проверки внешних спецификаций. Обычно эти методы дополняют друг друга и рекомендуются для последовательного использования.

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.

Следующим этапом в процессе создания программ является этап проектирования.

4.4. Внутреннее проектирование ПС

Внутреннее проектирование ПС начинается с изучения внешних спецификаций, разработанных на предыдущем этапе. Далее формируется структура ПС и общие правила взаимодействия компонентов ПС.

Процесс проектирования ПС включает: анализ и декомпозицию задач и данных в соответствии с принятым методом проектирования и завершается построением иерархической схемы, отражающей взаимосвязи между всеми модулями, описанием функций каждого модуля и интерфейса между модулями.

Правила формирования структуры и взаимодействия модулей ПС

1. Структура ПС и правила оформления каждого модуля должны быть унифицированы.

2. Каждый модуль должен характеризоваться функциональной законченностью, автономностью и независимостью в оформлении от модулей, которые его используют и которые он вызывает.

3. Применяются стандартные правила организации связей с другими модулями по управлению и информации.

4. Структура ПС должна быть представлена в виде совокупности небольших программных модулей, связанных иерархическим образом, что дает возможность полностью и относительно просто пояснить функцию и правила работы отдельных частей ПС в целом.

5. Должен отсутствовать эффект последействия очередного исполнения программного модуля на последующие исполнения.

Свойства модулей:

- *Связанность* — мера независимости частей модуля. Чем выше связность, тем лучше результат проектирования. Для обозначения связанности используется понятие силы связанности модуля.
- *Сцепление модуля* — мера взаимозависимости модулей по данным. Характеризуется как способом передачи данных, так и свойствами самих данных. Чем меньше сцепление, тем больше независимость модулей.

Критерии проектирования модулей:

- Сложность взаимодействия модуля с другими модулями должна быть меньше сложности его внутренней структуры.
- Хороший модуль снаружи проще, чем внутри.
- Хороший модуль проще использовать, чем построить.

Кроме внутренней связанности(по виду) и внешней связанности(по виду сцепления) степень независимости модуля определяется следующими факторами:

- Размер модуля — оказывает влияние на независимость, читаемость, сложность тестирования.
- Предсказуемые модули — модуль, работа которого не зависит от предыстории его использования.
- Структура принятия решения.
- Минимизация доступа к данным — объем данных, на который модуль может ссылаться, должен быть сведен к минимуму.

Свойства программного модуля (ПМ)

1. ПМ должен иметь один вход и один выход
2. ПМ должен решать самостоятельную задачу по принципу: один программный модуль — одна функция

3. Работа ПМ не должна зависеть от:
   - входных данных
   - того, какому ПМ предназначены выходные данные
   - предыстории вызовов ПМ
4. ПМ должен возвращать управление тому ПМ, который его вызвал
5. ПМ может вызывать любой ПМ
6. Размер ПМ желательно ограничивать одной–двумя страницами текста
7. ПМ должен иметь спецификацию

4.5. Проектирование и программирование модулей

В процессе внешнего проектирования модулей разрабатываются внешние взаимосвязи модулей, которые представляют собой внешнюю спецификацию каждого модуля.

Внешняя спецификация модуля не должна содержать никакой информации о внутреннем устройстве модуля, об особенностях реализованного в нем алгоритма. Кроме того, недопустимо, чтобы спецификация содержала какие–либо ссылки на вызывающие модули или контексты, в которых этот модуль используется.

Проектирование и кодирование модуля включает следующие шаги:

1. **Выбор языка программирования.**
2. **Проектирование внешних спецификаций модуля.** Это — процесс определения внешних характеристик каждого модуля. Внешние спецификации модуля должны содержать
   а) *Имя модуля.* Имя, с помощью которого можно обратится к модулю
   б) *Функция.* Что делает модуль, когда он вызван. Никаких сведений о том, как функция реализуется
   в) *Список параметров.* Описываются все входные параметры (указываются атрибуты, формат, размер, единицы измерения, допустимые диапазоны возможных значений всех входных параметров)

г) *Выходные параметры.* Описываются все данные, возвращаемые модулем. Определяется поведение модуля при любых входных условиях.

д) *Внешние эффекты.* Дается описание всех внешних по отношению к программе событий, происходящих при работе модуля, таких как прием запроса, выдача сообщений об ошибках и т.п.

3. **Проверка правильности внешних спецификаций модуля.** осуществляется путем сравнения их с полученной при проектировании информацией о взаимосвязях.

4. **Выбор алгоритма и структуры данных.**

5. **Оформление начала и конца будущего модуля** в соответствии с требованиями выбранного языка программирования.

6. **Объявление всех данных, используемых в качестве параметров**

7. **Объявление оставшихся данных.**

8. **Детализация логики программы.** Последовательная детализация логики модуля, начиная с достаточно высокого уровня абстракции и заканчивая готовым текстом программы. Для этого используются методы пошаговой детализации и структурного программирования.

   Пошаговая детализация — процесс разложения функции модуля на подфункции. Применяется при декомпозиции модуля.

   Структурное программирование — метод создания программ, основанный на

   а) проектировании «сверху – вниз»;

   б) модульном программировании;

   в) структурном кодировании.

   Структурное программирование предполагает создание улучшенных программ и служит для организации проектирования и кодирования программ таким образом, чтобы предотвратить большинство логических ошибок и обнаружить те, которые допущены.

   Метод проектирования программ «сверху – вниз» предусматривает вначале определение задачи в общих чертах, а затем постепенное уточнение структуры путем внесения более мелких деталей.

Модульное программирование — искусство разбиения задачи на некоторое количество модулей и использование стандартных модулей путем их параметрической настройки.

Структурное кодирование состоит в получении правильной программы из нескольких простых логических структур и базируется на теореме о структурировании: любая правильная программа (с одним входом, одним выходом, без зацикливания и недостижимых команд) может быть записаны с использованием только следующих основных логических структур:

- линейная ,
- нелинейная (развилка),
- циклическая (цикл).

Комбинация правильных программ, полученная с использованием этих трех структур, будет правильной.

9. **Окончательное оформление текста программы.** Проверяется текст, вставляются подробные комментарии.

10. **Проверка правильности программы.** Вручную проверяется правильность внутренней логики.

11. **Компиляция модуля.** Переход к этапу тестирования модуля.

Глава 5

Стандарты в области обеспечения качества программных систем

5.1. Стандарты серии ИСО 9000

ISO 9000 — серия международных стандартов ISO, регламентирующих управление качеством на предприятиях. Система стандартов разработана Международной Организацией по Стандартизации (ISO, International Organization for Standardization), которая основывалась на разработках Британского института стандартов BS 5750.

Стандарты ISO 9000, принятые более чем 90 странами мира, применимы к любым предприятиям, независимо от их размера и сферы деятельности. Сама ISO не производит сертификацию по ISO 9000, этим занимаются специально сформированные аудиторские организации в отдельных странах. Фактически сертификация производится не по ISO 9000, а по спецификации ISO 9001:2000. Сертификат ISO 9000 необходим предприятиям:

- работающим на международных рынках или с международными поставщиками, которые требуют наличия такого сертификата;
- работающим в секторах экономики, регулируемых правительством, или с правительственными организациями стран, в которых наличие сертификата ISO 9000 является обязательным.

В некоторых странах предприятия должны иметь сертификат ISO 9000 для того, чтобы предлагать свою продукцию не только правительственным организациям, но и потребителям определённых сегментов.

Стандарт не гарантирует качество продукции. Цель ISO 9000 — внести согласованность и объективность в действия системы контроля качества поставщика. Предполагается, что ISO 9000 будет использоваться в отношениях между компаниями, обычно в форме потребитель/поставщик. Стандарт помогает компаниям формализовать их систему управления процессом проверки качества и соответствия продукции.

В действительности ISO 9000 объединяет три стандарта:

- ISO 9000:2005 - Системы менеджмента качества. Основные положения и словарь
- ISO 9001:2000 - Системы менеджмента качества. Требования
- ISO 9004:2000 - Системы менеджмента качества. Рекомендации по улучшению деятельности

К стандартам этой серии также можно отнести ISO 19011:2003 - Рекомендации по аудиту систем менеджмента качества и/или охраны окружающей среды.

Идеология версии 2000 г. отличается от идеологии версии 1994 г. Она основана не на бизнес-функциях (элементах качества), а на бизнес-процессах предприятия. Международный стандарт ИСО 9001-2000 «поощряет применение процессуального подхода в управлении организацией и ее процессами, а также рассматривает его как способ быстрого выявления и реализации возможностей для улучшения». Выделяются 4 вида процессов:

1. Основной (производственный) процесс;
2. Процессы административного управления;
3. Процессы обеспечения ресурсами (включая трудовые);
4. Процессы корректировки (контроля, улучшения и т.д.).

В новой редакции стандарта используется единая модель системы качества (ИСО 9001-2000) для организаций всех видов (а не три модели - 9001, 9002, 9003 - как в версии 1994 г.). В соответствии с требованиями ИСО 9001-2000, организация: «должна установить и управлять процессами, необходимыми для обеспечения уверенности в том, что продукция и/или услуга соответствуют требованиям заказчика».

В качестве способа внедрения и демонстрации установленных процессов, организация должна создать систему менеджмента качества, основываясь на требованиях этого международного стандарта. Система менеджмента качества должна быть внедрена, поддерживаться в рабочем состоянии и подвергаться улучшениям со стороны организации".

Организация должна подготовить процедуры системы менеджмента качества, которые описывают процессы, необходимые для внедрения системы менеджмента качества. Масштаб и глубина процедур должна определяться такими факторами как размер и тип организации, сложность и взаимосвязь

процессов, применяемые методы, а также квалификация и степень подготовки персонала, участвующего в выполнении работ.

Процедуры системы качества должны включать

- общесистемные процедуры, которые описывают деятельность, необходимую для внедрения системы менеджмента качества;
- процедуры, описывающие последовательность и внутреннее содержание процессов, необходимых для обеспечения уверенности в соответствии продукции и/или услуги установленным требованиям;
- инструкции, описывающие операционную деятельность и управление процессами.

Такой подход:

1. более «гибок», чем подход ИСО 9000-1994;
2. гораздо более тесно увязан с совершенствованием системы управления;
3. устанавливает новую, более высокую планку в требованиях к системе управления организацией.

Семейство стандартов ИСО 9000 редакции 2000 года, разработано, чтобы помочь организациям всех типов и размеров внедрить и использовать эффективные системы менеджмента качества. Сообща они образуют комплект родственных стандартов системы менеджмента качества, перечисленных ниже: ИСО 9000:2000 обеспечивает введение в системы менеджмента качества и словарь менеджмента качества. ИСО 9001:2000 устанавливает детальные требования для систем менеджмента качества, в случае необходимости продемонстрировать способность организации, обеспечить соответствие продукции. ИСО 9004:2000 обеспечивает руководство по внедрению широко развитой системы менеджмента качества, чтобы достичь постоянного улучшения деловой деятельности. ИСО 19011:2002 обеспечивает руководство по управлению и проведению внутреннего и внешнего аудитов системы менеджмента качества.

Стандарт ИСО 9001:2000 предусматривает 4 группы процессов связанных с системой менеджмента качества:

1. Процессы управленческой деятельности руководства;
2. Процессы обеспечения ресурсами;

3. Процессы жизненного цикла продукции;

4. Процессы измерения, анализа и улучшения.

Первая группа — процессы управленческой деятельности руководства включают процессы из разделов 4 «Система менеджмента качества» и 5 «Ответственность руководства» стандарта ИСО 9001:2000. Эти процессы были включены в одну группу, основываясь на том, что они имеют одного «хозяина» — директора по качеству или представителя руководства, ответственного за систему менеджмента качества. В эту группу процессов входят:

1. Взаимоотношения с потребителем (определение и выполнение требований потребителей);

2. Формирование политики в области качества;

3. Планирование;

4. Распределение ответственности, полномочий и обмен информацией;

5. Анализ со стороны руководства;

6. Управление документацией;

7. Управление записями

Вторая группа — процессы обеспечения ресурсами состоит из процессов, описанных в разделе 6 «менеджмент ресурсов» и включает следующие процессы:

1. Менеджмент персонала;

2. Менеджмент инфраструктуры;

3. Управление производственной средой

Третья группа — процессы жизненного цикла продукции — составляют основные процессы организации по выпуску продукции или предоставлению услуги. Эти процессы представляют поток работ внутри организации, который имеет дело с товарами и услугами, предоставляемыми клиенту. Процессы жизненного цикла продукции:

1. Планирование процессов жизненного цикла продукции;

2. Процессы, связанные с анализом требований потребителя;

3. Проектирование и разработка;

4. Закупки;

5. Производство и обслуживание;

6. Управление устройствами для мониторинга и измерений.

Четвертую группу представляют процессы, завершающие цикл Деминга — процессы измерения, анализа и улучшения:

1. Мониторинг и измерение;

2. Управление несоответствующей продукцией;

3. Анализ данных;

4. Улучшение системы менеджмента качества;

   - Постоянное улучшение
   - Корректирующие действия
   - Предупреждающие действия

5.2. Менеджмент качества. Основные понятия

Все виды деятельности, встречающиеся в работе организации, рассматриваются как технологический процесс. В работе организации эти процессы взаимодействуют сложным образом, образуя систему или сеть процессов.

Международные стандарты семейства ИСО 9000 законодательно закрепили такой подход. Они основываются на понимании того, что всякая работа выполняется как процесс.

Каждый процесс, преобразуя некоторый объект труда, имеет вход и выход. Выход — это продукция, материальная и нематериальная, которая является результатом процесса. Выходом процесса может быть, например, документ, программный продукт, химическое вещество, банковская услуга, медицинское оборудование или промежуточная продукция (полуфабрикат) любой общей категории. В ИСО 9000 выделяется четыре общие категории продукции:

- оборудование (технические средства);
- интеллектуальная продукция (средства), под которым понимается продукт интеллектуальной деятельности, включающий в себя информацию, выраженную через средства поддержки; интеллектуальная продукция может быть как в форме программ для компьютера, так и в форме концепций, протоколов или методик;
- перерабатываемые материалы, под которыми понимается материальная продукция, получаемая путем переработки сырья в заданное состояние; перерабатываемые материалы могут представлять собой жидкость, газ, специфические материалы, слитки, прутки или листы; перерабатываемые материалы поставляются обычно в барабанах, мешках, цистернах, баллонах, канистрах, по трубопроводам и т.д.;
- услуги.

Входом процесса может являться материальная или нематериальная продукция или природное сырье.

Процесс, преобразуя объект труда, добавляет его стоимость. Каждый процесс включает определенным образом ресурсы, в том числе трудовые. На входе и выходе процесса, а также в различных фазах процесса могут проводиться измерения. Требования к системам качества в соответствии со стандартами ИСО 9000 могут быть применены ко всем четырем категорим продукции. Пожалуй, важнейшим моментом ИСО 9000 является то, что требования к системам качества по существу одни и те же для всех общих категорий продукции, различаться могут лишь детали административного построения и управления системами да терминология.

Общее руководство качеством достигается через управление процессами в организации.

Управление процессом включает:

1. определение целей и желаемых результатов процесса;
2. определение необходимых ресурсов, в том числе трудовых, для выполнения процесса;
3. определение методов и средств выполнения процесса;
4. управление использованием ресурсов, которые выделены для осуществления данного процесса, включая мотивацию персонала;

5. наблюдение за ходом процесса, анализ результатов его выполнения и коррекция хода процесса.

ИСО 9000 рекомендует строить управление процессами по двум направлениям:

- через структуру и работу самого процесса, внутри которого имеются потоки продукции и информации;
- через качество продукции и информации, протекающих внутри структуры.

В ИСО 9000 предполагается, что каждая организация существует для выполнения работы по добавлению стоимости продукции. Работа выполняется посредством сети процессов. Структура этой сети является достаточно сложной, поскольку большинство процессов взаимодействует между собой.

Концептуальной основой ИСО 9000 является то, что организация создает, обеспечивает и улучшает качество продукции при помощи сети процессов, которые должны подвергаться анализу и постоянному улучшению. Для обеспечения правильного управления процессами, организации взаимодействия между процессами в сети, ИСО 9000 предполагает, что у каждого процесса должен быть "владелец лицо, несущее ответственность за данный процесс. Этот "владелец"должен обеспечивать однозначное понимание всеми участниками процесса их ответственности и полномочий, должен организовывать взаимодействие при решении проблем, охватывающих несколько функциональных подразделений предприятия.

5.3. Показатели качества ПО в ГОСТ 28195 и ГОСТ Р ИСО/МЭК 9126

Показатели качества ПО устанавливают

- ГОСТ 28195 Оценка качества программных средств. Общие положения
- ГОСТ Р ИСО/МЭК 9126 Информационная технология. Оценка программной продукции. Характеристика качества и руководства по их применению.

Одновременное существование двух действующих стандартов, нормирующих одни и те же показатели, ставит вопрос об их гармонизации. Ниже кратко рассмотрим каждый из перечисленных стандартов.

ГОСТ Р ИСО/МЭК 9126 устанавливает шесть характеристик качества ПО. Под характеристикой качества ПО, согласно этому стандарту, понимается набор свойств (атрибутов) программной продукции, по которым ее качество оценивается или описывается. Определение качества и определенные в этом стандарте характеристики отражают представление пользователя о качестве ПО. Ниже приводятся существенное для оценки качества ПО обсуждение этих характеристик.

1. *Функциональные возможности.* Данная характеристика описывает свойства ПО в части полноты удовлетворения требований пользователя и в этом смысле является определяющей для потребительских свойств ПО, в то время как остальные характеристики носят более технический характер, что не уменьшает их значение при оценке качества ПО. Кроме того, эти характеристики (такие как надежность, эффективность и др.) могут входить в число требований пользователя.

   Требования пользователя четко обусловлены при наличие контракта и, соответственно, технического задания (технических требований). В других случаях речь идет о предполагаемых потребностях, которые должны быть установлены и формально определены какими-либо нормативными документами (стандартами, техническими условиями и пр.). Оценка качества ПС должна начинаться с точного и формального установления предъявляемых требований, которые могут различаться (и различаются) для различных ПО.

   Функциональные возможности — набор атрибутов, относящихся к сути набора функций и их конкретным свойствам. Функциями являются те, которые реализуют установленные или предполагаемые потребности. Данный набор атрибутов характеризует то, что ПО выполняет для удовлетворения потребностей, тогда как другие наборы, главным образом, характеризуют, когда и как это выполняется.

2. *Надежность.* Специфика ПО заключается в том, что оно не подвержено старению и износу, а отказы проявляются из-за ошибок в требованиях, проекте, реализации.

   Надежность — набор атрибутов, относящихся к способности ПО сохранять свой уровень качества функционирования в установленных условиях за определенный период времени.

3. *Практичность.* При оценке этой характеристики следует исходить из требований пользователя, так как пользователи разного уровня подго-

товленности предъявляют разные (часто взаимоисключающие) требования.

Практичность — набор атрибутов, относящихся к объему работ, требуемых для исполнения и индивидуальной оценки такого исполнения определенным или предполагаемым кругом пользователей.

4. *Эффективность.* Оценка данной характеристики также критически зависит от требований пользователя. ПО может выглядеть неэффективным не в силу плохого кодирования, а в силу противоречивости и нереальности исходных требований. Например, требования к ПО выполнять функции на технических средствах минимальной (по объему оперативной и дисковой памяти, тактовой частоте и пр.) конфигурации компьютера противоречит требованиям о высоком быстродействии. Вообще говоря, и теория, и практика свидетельствует, что быстродействие и объем используемой памяти является взаимодополняющими характеристиками в том смысле, что увеличение одного приводит к увеличению другого при прочих равных условиях.

   Эффективность — набор атрибутов, относящихся к соотношению между уровнем качества функционирования ПО и объемом используемых ресурсов при установленных условиях.

5. *Сопровождаемость. Мобильность.* Для этих двух характеристик следует учитывать, что в специфических российских условиях им часто не уделялось ранее и не уделяется сейчас достаточно внимания со стороны пользователя. Эти характеристики связаны с долгосрочным планированием развития ПО (и эксплуатирующей его организации). Улучшение сопровождаемости и мобильности может, вообще говоря, повысить стоимость ПО в настоящий момент времени, что (многократно) окупается лишь через несколько лет.

   Сопровождаемость — набор атрибутов, относящихся к объему работ, требуемых для проведения конкретных изменений (модификаций). Мобильность — набор атрибутов, относящихся к способности ПО быть перенесенным из одного окружения в другое.

Все приведенные характеристики являются наборами атрибутов и, следовательно, должны уточняться на множестве соответствующих подхарактеристик. ГОСТ Р ИСО/МЭК 9126 не устанавливает соответствующих показателей, но в рекомендуемом приложении А к этому стандарту дается пример

(качественная модель) таких подхарактеристик, называемых комплексными показателями.

В качестве примера приведем комплексные показатели для характеристики Функциональные возможности:

1. *Пригодность* — атрибут ПО, относящийся к наличию и соответствию набора функций конкретным задачам.

2. *Правильность* — атрибуты ПО, относящиеся к обеспечению правильности или соответствия результатов или эффектов.

3. *Способность к взаимодействию* — атрибуты ПО, относящиеся к способности его взаимодействовать с конкретными системами.

4. *Согласованность* — атрибуты ПО, которые заставляют программу придерживаться соответствующих стандартов или соглашений, или положений законов, или подобных рекомендаций.

5. *Защищенность* — атрибуты ПО, относящиеся к его способности предотвращать несанкционированный доступ, случайный или преднамеренный, к программам и данным.

В таблице показаны факторы и критерии качества ПО согласно ГОСТ 28195.

Характеристики двух нижних уровней (называемых метрика и оценочный элемент) устанавливаются в справочном приложении 2 к рассматриваемому стандарту. В том же приложении установлены методы проведения контроля за качеством ПО.

Методы определения качества ПО различаются:

- по способу получения информации о ПО — измерительный, регистрационный, органолептический, расчетный;

- по источникам получения информации — традиционный, экспертный, социологический.

Программно–аппаратные средства проведения контроля зависят от вида конкретного ПО и должны разрабатываться отдельно.

.5 cmc

Наименование факторов и критериев качества ПО и их обозначение	
1. Надежность ПО (Н)	Характериз
1.1. Устойчивость функционирования (Н1)	
1.2. Работоспособность (Н2)	
2. Сопровождаемость (С)	
2.1. Структурность (С1)	
2.2. Простота конструкции (С2)	
2.3. Наглядность (С3)	
2.4. Повторяемость (С4)	
3. Удобство применения (У)	
3.1. Легкость освоения (У1)	
3.2. Доступность эксплуатационных программных документов (У2)	
3.3. Удобство эксплуатации и обслуживания (У3)	

.5 cmc

Наименование факторов и критериев качества ПО и их обозначение	
4. Эффективность (Э)	
4.1. Уровень автоматизации (Э1)	Уровень авт
4.2. Временная эффективность (Э2)	
4.3. Ресурсоемкость (Э3)	
5. Универсальность (Г)	
5.1. Гибкость (Г1)	
5.2. Мобильность (Г2)	
5.3. Модифицируемость (Г3)	
6. Корректность (К)	
6.1. Полнота реализации (К1)	
6.2. Согласованность (К2)	Однознач
6.3. Логическая корректность (К3)	
6.3. Логическая корректность (К3)	
6.4. Проверенность (К4)	

Глава 6

Организация вычислений в программах сложной структуры

6.1. Модель предметной области пакета прикладных программ

Область науки или деятельности, к которой относятся задачи, решаемые с применением пакета прикладных программ (ППП), называется предметной областью пакета. Предметная область определяется совокупностью задач, решаемых пакетом. Разработчик ППП фактически имеет дело с некоторым упрощенным отображением предметной области, с некоторой моделью предметной области.

Модель предметной области можно представить совокупностью данных (переменных), используемых в пакете при решении задач, и связей между этими переменными.

Таким образом, *модель предметной области* есть объединение множества данных X, связей по определению R, множества функциональных связей F.

Каждое из этих множеств конечно, и значит, может быть отображено в памяти ЭВМ.

Если в процессе выполнения пакета множества X, R, F остаются неизменными (меняются только значения данных), то такую модель предметной области называют статической, а соответствующий ППП — пакетом со *статической* моделью предметной области.

Если же пользователь во время сеанса работы может изменить хотя бы одно из множеств X, R, F, то модель предметной области называется *динамической*.

Рассмотрим *множество данных* X. Данное характеризуется тремя показателями (имя, тип, значение). Имя — уникально. Каждое данное относится только к одному типу. По способу присваивания данные делятся на три группы:

- константы (в процессе работы пакета не меняются);

- данное имеет некоторое фиксированное значение при загрузке пакета (значение по умолчанию), но пользователь или обрабатывающий модуль могут изменить это значение;

- данное не имеет значение до тех пор, пока пользователь не определит его. Эти данные могут быть только входными.

Связи R между данными в информационной базе пакета устанавливаются при ее разработке. Типы таких связей:

- образование иерархических структур данных;

- связь подчинения по отношению к сохранению значений данных.

Связи типа подчинения легко представить в виде предикатов.

Таким образом, связи по определению отражают ограничения на совокупности возможных значений данных.

Функциональные связи F реализуются в пакете обрабатывающими модулями. Обрабатывающий модуль можно представить как функцию $y = f(x)$. Функциональная связь представляется

- набором входных данных,

- набором выходных данных,

- обрабатывающим модулем.

Функциональная связь *реализуема* (т.е. модуль выполним), если известны значения входных данных. Условие реализуемости функциональной связи можно описать как предикат $P_f(x)$, который принимает значение истина, если связь реализуема. В модели предметной области связи F не должны нарушать связи R.

Состояние модели предметной области можно охарактеризовать вектором $S = (s_1, s_2, \ldots, s_n)$, n — число данных в X, s_i равен 1, если значение i-го данного известно и 0, если неизвестно:

$$s_i = \begin{cases} 1, & \text{значение } i\text{-го данного известно,} \\ 0, & \text{значение } i\text{-го данного неизвестно.} \end{cases} \tag{6.1}$$

Функционирование пакета отображается на модели предметной области изменением состояния модели

$$S_0 \to S_1 \to S_2 \to \ldots \to S_k.$$

В модели предметной области, содержащей n данных, возможны 2^n состояний.

Возможные состояния модели и связи между ними можно представить графом переходов, где узлы — состояния модели, а дуги — модули пакета.

Текущее состояние модели предметной области пакета можно охарактеризовать вектором $S = (s_1, s_2, \ldots, s_n)$, n - число данных в X, s_i равен 1, если значение i-го данного известно и 0, если неизвестно.

$$s_i = \begin{cases} 1, & \text{значение i-го данного известно,} \\ 0, & \text{значение i-го данного неизвестно.} \end{cases}$$

Обрабатывающий модуль f_j называется *выполнимым*, если значения всех компонентов x известны. Выполнимый модуль можно вызвать (реализовать соответствующую этому модулю функциональную связь), и количество известных данных может увеличиться.

Выполнимый модуль называется *эффективным* в состоянии S, если его вызов переводит модель предметной области в новое состояние $S' \neq S$.

Совокупность функциональных связей в модели предметной области пакета представима матрицами T и R размерности $n \times n$, m – число функциональных связей, n – число данных,

$$t_{ij} = \begin{cases} 1, & \text{если j-ое данное является входным для i-го модуля,} \\ 0, & \text{в остальных случаях.} \end{cases}$$

$$r_{ij} = \begin{cases} 1, & \text{если j-ое данное является результатом i-го модуля,} \\ 0, & \text{в остальных случаях.} \end{cases}$$

Интерпретируем операции & (конъюнкцию), $\vee$ (дизъюнкцию) и $\neg$ (отрицание) как поэлементные булевские операции над битовыми строками.

Тогда условие выполнимости модуля f_i можно записать

$$T_i \& S = T_i,$$

где T_i — строка матрицы T, соответствующая модулю f_i.

Выполнимый модуль будет эффективным, если

$$R_i \& (\neg S) \neq 0.$$

6.2. ПЛАНИРОВАНИЕ ВЫЧИСЛИТЕЛЬНОГО ПРОЦЕССА В ППП

6.2.1. Постановка задачи

Под планированием вычислительного процесса в ППП будем понимать определение такой последовательности вызовов обрабатывающих модулей, которая обеспечит вычисление значений данных, указанных пользователем.

Текущее состояние модели предметной области пакета можно охарактеризовать вектором

$$S = (s_1, s_2, \ldots, s_n),$$

n - число данных в X, s_i равен 1, если значение i-го данного известно и 0, если неизвестно.

$$s_i = \begin{cases} 1, & \text{значение } i\text{-го данного известно,} \\ 0, & \text{значение } i\text{-го данного неизвестно.} \end{cases}$$

Возможные состояния модели и связи между ними можно представить графом переходов, где узлы — состояния модели, а дуги — модули пакета.

Указывая значения известных ему данных, пользователь определяет исходное состояние модели предметной области (т.е. один из узлов графа переходов).

Указывая список данных, значения которых требуется вычислить, пользователь определяет множество конечных состояний (т.е. узлов графа переходов, в которых известны как данные, заданные пользователем как исходные, так и данные, которые необходимо вычислить).

Если обозначить S_0 — вектор исходного состояния, Z — требования к конечному состоянию, тогда конечное состояние — это любое состояние S_k, удовлетворяющее условию $S_k \& Z = Z$.

Таким образом, задача планирования вычислительного процесса состоит в поиске пути на графе переходов из заданного узла в любой из узлов, соответствующих условию $S_k \& Z = Z$. Для решения этой задачи существует несколько алгоритмов. Опишем один из них.

6.2.2. Алгоритм планирования с прямым ходом.

Сущность метода прямой волны состоит в том, чтобы последовательно включать в управляющий вектор все модули, эффективные в каждом текущем состоянии МПО, и повторять этот процесс до тех пор, пока не будет получено одно из конечных состояний.

Управляющий вектор — это последовательность вызовов обрабатывающих модулей.

Обозначим Z — вектор искомых данных,

$$z_i = \begin{cases} 1, & \text{если i-е данное искомое,} \\ 0, & \text{в остальных случаях.} \end{cases}$$

Шаги алгоритма:

1. Подготовка. Очистить формируемый управляющий вектор U, установить длину вектора U: $K_U = 0$, установить текущее состояние МПО $S_T = S_0$.

2. Цикл (пока есть эффективные модули и не все искомые данные известны)

   - 2.1. Цикл по функциональным связям (по строкам матриц T и R).
     - 2.1.1. если модуль f_i – выполним и эффективен, т.е. $(T_i \& S_T = T_i) \& (R_i \& (\neg S_T) \neq 0)$, то
       * 2.1.1.1. включить модуль f_i в очередную порцию управляющего вектора U.
       * 2.1.1.2. преобразовать текущее состояние МПО $S_T = S_T \vee R_i$, и перейти к п. 2.2.
     - 2.1.2. Конец цикла 2.1.
   - 2.2. Конец цикла 2.

3. Конец. Если $S_T \& Z = Z$, то получено решение задачи, в противном случае задача неразрешима.

Пример 1. *Входные и выходные данные для модулей приведены в таблице.*

Таблица 6.1

Имя модуля	Входные данные	Результаты модуля
M1	a,b	c
M2	a,b	d
M3	c,b	e
M4	b,e	x
M5	a,d	x

Пусть исходное состояние МПО $S_0 = (110000)$. *Найти управляющий вектор для вычисления* x, *т.е.* $Z = (000001)$.

При первом прохождении цикла 2.1. в управляющий вектор включается модуль M1 и получается $S_1 = (111000)$, *затем добавляется модуль M2, и* $S_1 = (111100)$, *затем M3 и* $S_1 = (111110)$, *наконец M4,* $S_1 = (111111)$. *Прямой ход заканчивается.*

Глава 7

Модели надежности программного обеспечения

7.1. Понятие надежности программного обеспечения

Надежность программного обеспечения гораздо важнее других его характеристик, например, времени исполнения, и хотя абсолютная надежность современного программного обеспечения, по-видимому, недостижима, до сих пор не существует общепринятой меры надежности компьютерных программ.

Проблема надежности программного обеспечения относится, похоже, к категории «вечных». В посвященной ей монографии Г.Майерса, выпущенной в 1980 году (американское издание - в 1976), отмечается, что, хотя этот вопрос рассматривался еще на заре применения вычислительных машин, в 1952 году, он не потерял актуальности до настоящего времени. Очевидно, что надежность программы гораздо важнее таких традиционных ее характеристик, как время исполнения или требуемый объем оперативной памяти.

Термин модель надежности программного обеспечения, как правило, относится к математической модели, построенной для оценки зависимости надежности программного обеспечения от некоторых определенных параметров. Значения таких параметров либо предполагаются известными, либо могут быть измерены в ходе наблюдений или экспериментального исследования процесса функционирования программного обеспечения. Данный термин может быть использован также применительно к математической зависимости между определенными параметрами, которые хотя и имеют отношение к оценке надежности программного обеспечения, но тем не менее не содержат ее характеристик в явном виде. Например, поведение некоторой ветви программы на подмножестве наборов входных данных, с помощью которых эта ветвь контролируется, существенным образом связано с надежностью программы, однако характеристики этого поведения могут быть оценены независимо от оценки самой надежности. Другим таким параметром является частота ошибок, которая позволяет оценить именно качество систем реального времени, функционирующих в непрерывном режиме, и в то же время получать только косвенную информацию относительно надежности программного обеспечения (например, в предположении экспоненциального

распределения времени между отказами).

Одним из видов модели надежности программного обеспечения, которая заслуживает особого внимания, является так называемая феноменологическая, или эмпирическая, модель. При разработке моделей такого типа предполагается, что связь между надежностью и другими параметрами является статической. С помощью подобного подхода пытаются количественно оценить те характеристики программного обеспечения, которые свидетельствуют либо о высокой, либо о низкой его надежности. Так, например, параметр сложность программы характеризует степень уменьшения уровня ее надежности, поскольку усложнение программы всегда приводит к нежелательным последствиям, в том числе к неизбежным ошибкам программистов при составлении программ и трудности их обнаружения и устранения. Иначе говоря, при разработке феноменологической модели надежности программного обеспечения стремятся иметь дело с такими параметрами, соответствующее изменение значений которых должно приводить к повышению надежности программного обеспечения.

7.2. Классификация моделей надежности ПС

Рассмотрим классификацию моделей надежности ПС, приведенную на рис. 7.1. Модели надежности программных средств (МНПС) подразделяются на аналитические и эмпирические. Аналитические модели дают возможность рассчитать количественные показатели надежности, основываясь на данных о поведении программы в процессе тестирования (измеряющие и оценивающие модели). Эмпирические модели базируются на анализе структурных особенностей программ. Они рассматривают зависимость показателей надежности от числа межмодульных связей, количества циклов в модулях, отношения количества прямолинейных участков программы к количеству точек ветвления и т.д. Часто эмпирические модели не дают конечных результатов показателей надежности, однако они включены в классификационную схему, так как развитие этих моделей позволяет выявлять взаимосвязь между сложностью ПС и его надежностью. Эти модели можно использовать на этапе проектирования ПС, когда осуществлена разбивка на модули и известна его структура.

Аналитические модели представлены двумя группами: динамические модели и статические. В динамических МНПС поведение ПС (появление отказов) рассматривается во времени. В статических моделях появление отказов не связывают со временем, а учитывают только зависимость количества

ошибок от числа тестовых прогонов (по области ошибок) или зависимость количества ошибок от характеристики входных данных (по области данных).

Для использования динамических моделей необходимо иметь данные о появлении отказов во времени. Если фиксируются интервалы каждого отказа, то получается непрерывная картина появления отказов во времени (группа динамических моделей с непрерывным временем). Может фиксироваться только число отказов за произвольный интервал времени. В этом случае поведение ПС может быть представлено только в дискретных точках (группа динамических моделей с дискретным временем). Рассмотрим основные предпосылки, ограничения и математический аппарат моделей, представляющих каждую группу, выделенную по схеме.

7.3. Аналитические модели надежности

Аналитическое моделирование надежности ПС включает четыре шага:

1. определение предположений, связанных с процедурой тестирования ПС;
2. разработка или выбор аналитической модели, базирующейся на предположениях о процедуре тестирования;
3. выбор параметров моделей с использованием полученных данных;
4. применение модели – расчет количественных показателей надежности по модели.

7.3.1. Динамические модели надежности

Модель Шумана

Исходные данные для модели Шумана, которая относится к динамическим моделям дискретного времени, собираются в процессе тестирования ПС в течение фиксированных или случайных временных интервалов [**?**]. Каждый интервал – это стадия, на которой выполняется последовательность тестов и фиксируется некоторое число ошибок.

Модель Шумана может быть использована при определенным образом организованной процедуре тестирования. Использование модели Шумана предполагает, что тестирование проводится в несколько этапов. Каждый этап представляет собой выполнение программы на полном комплексе разработанных тестовых данных. Выявленные ошибки регистрируются (собирается

статистика об ошибках), но не исправляются. По завершении этапа на основе собранных данных о поведении ПС на очередном этапе тестирования может быть использована модель Шумана для расчета количественных показателей надежности. После этого исправляются ошибки, обнаруженные на предыдущем этапе, при необходимости корректируются тестовые наборы и проводится новый этап тестирования. При использовании модели Шумана предполагается, что исходное количество ошибок в программе постоянно и в процессе тестирования может уменьшаться по мере того, как ошибки выявляются и исправляются. Новые ошибки при корректировке не вносятся. Скорость обнаружения ошибок пропорциональна числу оставшихся ошибок. Общее число машинных инструкций в рамках одного этапа тестирования постоянно.

Предполагается, что до начала тестирования в ПС имеется E_T ошибок. В течение времени тестирования τ обнаруживается ϵ_c ошибок в расчете на команду в машинном языке.

Таким образом, удельное число ошибок на одну машинную команду, оставшихся в системе после τ времени тестирования, равно:

$$\epsilon_r(\tau) = \frac{E_T}{I_T} - \epsilon_c(\tau), \tag{7.1}$$

где I_T – общее число машинных команд, которое предполагается постоянным в рамках этапа тестирования.

Предполагается, что значение частоты отказов $Z(t)$ пропорционально числу ошибок, оставшихся в ПС после израсходованного на тестирование времени τ:

$$Z(t) = C\epsilon_r(\tau),$$

где C – некоторая константа; t – время работы ПС без отказа.

Тогда, если время работы ПС без отказа t отсчитывается от точки $t = 0$, а τ остается фиксированным; функция надежности, или вероятность безотказной работы на интервале времени от 0 до t, равна:

$$R(t, \tau) = \exp\{-C(E_T/I_T - \epsilon_c(\tau))t\}; \tag{7.2}$$

$$t_{cp} = \frac{1}{C(E_T/I_T - \epsilon_c(\tau))}. \tag{7.3}$$

Из величин, входящих в формулы (7.2) и (7.3), не известны начальное значение ошибок в ПС E_T и коэффициент пропорциональности C. Для их определения прибегают к следующим рассуждениям. В процессе тестирования собирается информация о времени и количестве ошибок на каждом

прогоне, т.е. общее время тестирования τ складывается из времени каждого прогона:

$$\tau = \tau_1 + \tau_2 + \ldots + \tau_n.$$

Предполагая, что интенсивность появления ошибок постоянна и равна λ, можно вычислить ее как число ошибок в единицу времени:

$$\lambda = \frac{\sum_{i=1}^{k} A_i}{\tau}, \tag{7.4}$$

где A_i – количество ошибок на i-ом прогоне;

$$t_{cp} = \frac{\tau}{\sum_{i=1}^{k} A_i}. \tag{7.5}$$

Имея данные для двух различных моментов тестирования τ_a и τ_b, которые выбираются произвольно с учетом требования, чтобы $\epsilon_c(\tau_a) > \epsilon_c(\tau_b)$, можно сопоставить уравнения (7.3) и (7.5) при τ_a и τ_b.

$$\frac{1}{\lambda_{\tau_a}} = \frac{1}{C(E_T/I_T - \epsilon_c(\tau_a))}, \tag{7.6}$$

$$\frac{1}{\lambda_{\tau_b}} = \frac{1}{C(E_T/I_T - \epsilon_c(\tau_b))}. \tag{7.7}$$

Вычисляя отношения (7.6) и (7.7), получим:

$$E_T = \frac{I_T(\lambda_{\tau_b}/\lambda_{\tau_a}\epsilon_c(\tau_a) - \epsilon_c(\tau_b))}{(\lambda_{\tau_b}/\lambda_{\tau_a}) - 1}. \tag{7.8}$$

Подставив полученную оценку параметров E_T в выражение (7.6), получим оценку для второго неизвестного параметра:

$$C = \frac{\lambda\tau_a}{E_T/I_T - \epsilon_c(\tau_a)}. \tag{7.9}$$

Получив неизвестные E_T и C, можно рассчитать надежность программы по формуле (7.2).

Модель La Padula

По этой модели выполнение последовательности тестов производится в m этапов. Каждый этап заканчивается внесением изменений (исправлений) в ПС. Возрастающая функция надежности базируется на числе ошибок, обнаруженных в ходе каждого тестового прогона.

Надежность ПС в течение i-го этапа:

$$R(t) = R(\infty) - A/i,\ i = 1, 2, \ldots,$$

где A – параметр роста; $R(\infty) = \lim_{i\to\infty} R(i)$ – предельная надежность ПС. Эти неизвестные величины можно вычислить, решив следующие уравнения:

$$\sum_{i=1}^{m} \left(\frac{S_i - m_i}{S_i} - R(\infty) + \frac{A}{i} \right) = 0,$$

$$\sum_{i=1}^{m} \left(\frac{S_i - m_i}{S_i} - R(\infty) + \frac{A}{i} \right) \frac{1}{i} = 0,$$

где S_i – число тестов; m_i – число отказов во время i-го этапа; m – число этапов.

Определяемый по этой модели показатель есть надежность ПС на i-ом этапе: $R(t) = R(\infty) - A/i$, $i = m+1, m+2, \ldots$.

Преимущество модели заключается в том, что она является прогнозной и, основываясь на данных, полученных в ходе тестирования, дает возможность предсказать вероятность безотказной работы программы на последующих этапах ее выполнения.

Модель Джелинского–Моранды

Модель Джелинского–Моранды относится к динамическим моделям непрерывного времени. Исходные данные для использования этой модели собираются в процессе тестирования ПС. При этом фиксируется время до очередного отказа. Основное положение, на котором базируется модель, заключается в том, что значение интервалов времени тестирования между обнаружением двух ошибок имеет экспоненциальное распределение с частотой ошибок (или интенсивностью отказов), пропорциональной числу еще не выявленных ошибок. Каждая обнаруженная ошибка устраняется, число оставшихся ошибок уменьшается на единицу.

Функция плотности распределения времени обнаружения 1-й ошибки, отсчитываемого от момента выявления $i-1$-й ошибки, имеет вид:

$$P(t_i) = \lambda_i e^{-\lambda_i t_i}, \tag{7.10}$$

где λ_i – частота отказов (интенсивность отказов), которая пропорциональна числу еще не выявленных ошибок в программе,

$$\lambda_i = C(N - i + 1), \tag{7.11}$$

где N – число ошибок, первоначально присутствующих в программе; C – коэффициент пропорциональности.

Наиболее вероятные значения величин N и C (оценка максимального правдоподобия) можно определить на основе данных, полученных при тестировании. Для этого фиксируют время выполнения программы до очередного отказа $t_1, t_2, \ldots, t_k$.

Значения N и C предлагается получить, решив систему уравнений:

$$\sum_{i=1}^{k}(\widehat{N}-i+1)^{-1}=\frac{K}{\widehat{N}+1-QK},\ \widehat{C}=\frac{K/A}{\widehat{N}+1-QK}, \tag{7.12}$$

где

$$Q=\frac{B}{AK},\ A=\sum_{i=1}^{k}T_i,\ B=\sum_{i=1}^{k}it_i.$$

Поскольку полученные значения N и C – вероятностные и точность их зависит от количества интервалов тестирования (или количества ошибок), найденных к моменту оценки надежности, асимптотические оценки дисперсий можно определить с помощью следующих формул [**?**]:

$$Var(\widehat{N})=\frac{K}{C^2D},\ Var(\widehat{C})=\frac{S}{D},$$

где $D=\frac{KS}{C^2}-A^2$, $S=\sum_{i=1}^{K}(N-i+1)^2$.

Чтобы получить числовые значения λ_i, нужно подставить вместо N и C их возможные значения $\widehat{N}$ и $\widehat{C}$. Рассчитав K значений по формуле (7.11) и подставив их в формулу (7.10), можно определить вероятность безотказной работы на различных временных интервалах. На основе полученных расчетных данных строится график зависимости вероятности безотказной работы от времени.

Модель Шика–Волвертона

Модификация модели Джелинского–Моранды для случая возникновения на рассматриваемом интервале более одной ошибки предложена Волвертоном и Шиком. При этом считается, что исправление ошибок производится лишь после истечения интервала времени, на котором они возникли. В основе модели Шика–Волвертона лежит предположение, согласно которому частота ошибок пропорциональна не только количеству ошибок в программах, но и времени тестирования, т.е. вероятность обнаружения ошибок с течением времени возрастает. Частота ошибок (интенсивность обнаружения ошибок) λ_i

предполагается постоянной в течение интервала времени t_i и пропорциональна числу ошибок, оставшихся в программе по истечении $i-1$-го интервала; но она пропорциональна также и суммарному времени, уже затраченному на тестирование (включая среднее время выполнения программы в текущем интервале):

$$\lambda_i = C(N - n_{i-1})(T_{i-1} + t_i/2). \tag{7.13}$$

В данной модели наблюдаемым событием является число ошибок, обнаруживаемых в заданном временном интервале, а не время ожидания каждой ошибки, как это было для модели Джелинского– Моранды. В связи с этим модель относят к группе дискретных динамических моделей.

Модель Муса

Модель Муса относят к динамическим моделям непрерывного времени. Это значит, что в процессе тестирования фиксируется время выполнения программы (тестового прогона) до очередного отказа. Но считается, что не всякая ошибка ПС может вызвать отказ, поэтому допускается обнаружение более одной ошибки при выполнении программы до возникновения очередного отказа.

Считается, что на протяжении всего жизненного цикла ПС может произойти M_0 отказов и при этом будут выявлены все N_0 ошибки, которые присутствовали в ПС до начала тестирования.

Общее число отказов M_0 связано с первоначальным числом ошибок N_0 соотношением

$$N_0 = BM_0, \tag{7.14}$$

где B – коэффициент уменьшения числа ошибок.

В момент, когда проводится оценка надежности, после тестирования, на которое потрачено определенное время t, зафиксировано m отказов и выявлено n ошибок.

Тогда из соотношения

$$n = Bm \tag{7.15}$$

можно определить коэффициент уменьшения числа ошибок B как число, характеризующее количество устраненных ошибок, приходящихся на один отказ.

В модели Муса различают два вида времени:

1. суммарное время функционирования τ, которое учитывает чистое время тестирования до контрольного момента, когда про водится оценка надежности;

2. оперативное время t - время выполнения программы, планируемое от контрольного момента и далее при условии, что дальнейшего устранения ошибок не будет (время безотказной работы в процессе эксплуатации).

Для суммарного времени функционирования τ предполагается:

- интенсивность отказов пропорциональна числу неустраненных ошибок;
- скорость изменения числа устраненных ошибок, измеряемая относительно суммарного времени функционирования, пропорциональна интенсивности отказов.

Один из основных показателей надежности, который рассчитывается по модели Муса, – средняя наработка на отказ. Этот показатель определяется как математическое ожидание временного интервала между последовательными отказами и связан с надежностью:

$$T = \int_0^\infty t f(t) dt = \int_0^\infty R(t) dt,$$

где t – время работы до отказа.

Если интенсивность отказов постоянна (т.е. когда длительность интервалов между последовательными отказами имеет экспоненциальное распределение), то средняя наработка на отказ обратно пропорциональна интенсивности отказов.

Модель переходных вероятностей

Эта модель основана на марковском процессе, протекающем в дискретной системе с непрерывным временем.

Процесс, протекающий в системе, называется марковским (или процессом без последствий), если для каждого момента времени вероятность любого состояния системы в будущем зависит только от состояния системы в настоящее время t_0 и не зависит от того, каким образом система пришла в это состояние. Процесс тестирования ПС рассматривается как марковский процесс.

В начальный момент тестирования $t = 0$ в ПС было n ошибок. Предполагается, что в процессе тестирования выявляется по одной ошибке. Тогда последовательность состояний системы ($n, n-1, n-2, n-3$ и т.д.) соответствует периодам времени, когда предыдущая ошибка уже исправлена, а новая еще не обнаружена. Например, в состоянии $n-5$ пятая ошибка уже исправлена, а шестая еще не обнаружена. Последовательность состояний $m, m-1, m-2, m-3$ и т.д. соответствует периодам времени, когда

ошибки исправляются. Например, в состоянии $m-1$ вторая ошибка уже обнаружена, но еще не исправлена. Ошибки обнаруживаются с интенсивностью χ, а исправляются с интенсивностью μ.

7.3.2. Статические модели надежности

Статические модели принципиально отличаются от динамических прежде всего тем, что в них не учитывается время появления ошибок в процессе тестирования и не используется никаких предположений о поведении функции риска $\lambda(t)$. Эти модели строятся на твердом статистическом фундаменте.

Модель Миллса

Использование этой модели предполагает необходимость перед началом тестирования искусственно вносить в программу («засорять») некоторое количество известных ошибок. Ошибки вносятся случайным образом и фиксируются в протоколе искусственных ошибок. Специалист, проводящий тестирование, не знает ни количества, ни характера внесенных ошибок до момента оценки показателей надежности по модели Миллса. Предполагается, что все ошибки (как естественные, так и искусственно внесенные) имеют равную вероятность быть найденными в процессе тестирования.

Тестируя программу в течение некоторого времени, собирают статистику об ошибках. В момент оценки надежности по протоколу искусственных ошибок все ошибки делятся на собственные и искусственные. Соотношение

$$N = \frac{Sn}{V} \tag{7.16}$$

дает возможность оценить N – первоначальное число ошибок в программе. В данном соотношении, которое называется формулой Миллса, S – количество искусственно внесенных ошибок, n – число найденных собственных ошибок, V - число обнаруженных к моменту оценки искусственных ошибок.

Вторая часть модели связана с проверкой гипотезы от N. Предположим, что в программе имеется K собственных ошибок, и внесем в нее еще S ошибок. В процессе тестирования были обнаружены все S внесенных ошибок и n собственных ошибок.

Тогда по формуле Миллса мы предполагаем, что первоначально в программе было $N = n$ ошибок. Вероятность, с которой можно высказать такое предположение, возможно рассчитать по следующему соотношению:

$$C = \begin{cases} 1, \ n > K; \\ \frac{S}{S+K+1}, \ n \leqslant K. \end{cases} \tag{7.17}$$

Таким образом, величина C является мерой доверия к модели и показывает вероятность того, насколько правильно найдено значение N. Эти два связанных между собой по смыслу соотношения образуют полезную модель ошибок: первое предсказывает возможное число первоначально имевшихся в программе ошибок, а второе используется для установления доверительного уровня прогноза.

Модель Липова

Липов модифицировал модель Миллса, рассмотрев вероятность обнаружения ошибки при использовании различного числа тестов. Если сделать то же предположение, что и в модели Миллса, т.е. что собственные и искусственные ошибки имеют равную вероятность быть найденными, то вероятность обнаружения n собственных и V внесенных ошибок равна:

$$Q(n,V) = \frac{m}{n+V} q^{n+V} (1-q)^{m-n-V} \frac{\frac{N}{n}\frac{S}{V}}{\frac{N+S}{n+V}},$$

где m – количество тестов, используемых при тестировании; q – вероятность обнаружения ошибки в каждом из m тестов, рассчитанная по формуле $q = \frac{n+V}{n}$, S – общее количество искусственно внесенных ошибок; N – количество собственных ошибок, имеющихся в ПС до начала тестирования.

Для использования модели Липова должны выполняться следующие условия: $N \geqslant n \geqslant 0$, $S \geqslant V \geqslant 0$, $m \geqslant n+V \geqslant 0$. Оценки максимального правдоподобия (наиболее вероятное значение для N) задаются соотношениями $N = \frac{Sn}{V}$ при $n > 1$, $V > 1$; nS при $V = 0$; 0 при $n = 0$.

Модель Липова дополняет модель Миллса, давая возможность оценить вероятность обнаружения определенного количества ошибок к моменту оценки.

Простая интуитивная модель

Использование этой модели предполагает проведение тестирования двумя группами программистов (или двумя программистами в зависимости от величины программы) независимо друг от друга, использующими независимые тестовые наборы. В процессе тестирования каждая из групп фиксирует все найденные ею ошибки. При оценке числа оставшихся в программе ошибок результаты тестирования обеих групп собираются и сравниваются. Получается, что первая группа обнаружила N_1 ошибок, вторая - N_2, а n_{12} – это ошибки, обнаруженные обеими группами.

Если обозначить через N неизвестное количество ошибок, присутствовавших в программе до начала тестирования, то можно эффективность те-

стирования каждой из групп определить как

$$E_1 = \frac{N_1}{N};\ E_2 = \frac{N_2}{N}. \tag{7.18}$$

Предполагая, что возможность обнаружения всех ошибок одинакова для обеих групп, можно допустить, что если первая группа обнаружила определенное количество всех ошибок, она могла бы определить то же количество любого случайным образом выбранного подмножества. В частности, можно допустить:

$$E_1 = \frac{N_1}{N} = \frac{N_{12}}{N_2}. \tag{7.19}$$

Из формулы (7.18) $N_2 = E_2 N$, подставив в (7.19), получим:

$$E_1 = \frac{N_{12}}{E_2 N} = \frac{N_1 N_2}{N_{12}}.$$

Модель Коркорэна

Модель Коркорэна относится к статическим моделям надежности ПС, так как в ней не используются параметры времени тестирования и учитывается только результат N испытаний, в которых выявлено N_1 ошибок i-го типа. Модель использует изменяющиеся вероятности отказов для различных типов ошибок.

В отличие от двух рассмотренных выше статических моделей, по модели Коркорэна оценивается вероятность безотказного выполнения программы на момент оценки:

$$R = \frac{N_0}{N} + \sum_{i=1}^{K} Y_i \frac{N_i - 1}{N},$$

где N_0 – число безотказных выполнений программы; N – общее число прогонов; K – априори известное число типов,

$$Y_i = \begin{cases} a, & N_i > 0; \\ 0, & N_i = 0, \end{cases}$$

a_i – вероятность выявления при тестировании ошибки i-го типа. В этой модели вероятность a_i, должна оцениваться на основе априорной информации или данных предшествующего периода функционирования однотипных программных средств.

Модель Нельсона

Данная модель при расчете надежности ПС учитывает вероятность выбора определенного тестового набора для очередного выполнения программы.

Предполагается, что область данных, необходимых для выполнения тестирования программного средства, разделяется на K взаимоисключающих подобластей Z_i, $i = 1, 2, \ldots, K$. Пусть P_i – вероятность того, что набор данных Z_i будет выбран для очередного выполнения программы. Предполагая, что к моменту оценки надежности было выполнено N_i прогонов программы на Z_i наборе данных и из них n_i количество прогонов закончилось отказом, надежность ПС в этом случае равна:

$$R = 1 - \sum_{i=1}^{K} P_i \frac{n_i}{N_i}, \tag{7.20}$$

На практике вероятность выбора очередного набора данных для прогона (P_i) определяется путем разбиения всего множества значений входных данных на подмножества и нахождения вероятностей того, что выбранный для очередного прогона набор данных будет принадлежать конкретному подмножеству. Определение этих вероятностей основано на эмпирической оценке вероятности появления тех или иных входов в реальных условиях функционирования.

7.4. Эмпирические модели надежности

Эмпирические модели в основном базируются на анализе структурных особенностей программного средства (или программы). Как указывалось ранее, эмпирические модели часто не дают конечных результатов показателей надежности, однако их использование на этапе проектирования ПС полезно для прогнозирования требующихся ресурсов тестирования, уточнения плановых сроков завершения проекта и т.д.

7.4.1. Модель сложности

В литературе неоднократно подчеркивается тесная взаимосвязь между сложностью и надежностью ПС. Если придерживаться упрощенного понимания сложности ПС, то она может быть описана такими характеристиками, как размер ПС (количество программных модулей), количество и сложность межмодульных интерфейсов.

Под программным модулем в данном случае следует понимать программную единицу, выполняющую определенную функцию (ввод, вывод, вычисле-

ние и т.д.) и взаимосвязанную с другими модулями ПС. Сложность модуля ПС может быть описана, если рассматривать структуру программы.

В качестве структурных характеристик модуля ПС используются:

1. отношение действительного числа дуг к максимально возможному числу дуг, получаемому искусственным соединением каждого узла с любым другим узлом дугой;

2. отношение числа узлов к числу дуг;

3. отношение числа петель к общему числу дуг.

Для сложных модулей и для больших многомодульных программ составляется имитационная модель, программа которой «засоряется» ошибками и тестируется по случайным входам. Оценка надежности осуществляется по модели Миллса.

При проведении тестирования известна структура программы, имитирующей действия основной, но не известен конкретный путь, который будет выполняться при вводе определенного тестового входа. Кроме того, выбор очередного тестового набора из множества тест-входов случаен, т.е. в процессе тестирования не обосновывается выбор очередного тестового входа. Эти условия вполне соответствуют реальным условиям тестирования больших программ.

Полученные данные анализируются, проводится расчет показателей надежности по модели Миллса (или любой другой из описанных выше), и считается, что реальное ПС, выполняющее аналогичные функции, с подобными характеристиками и в реальных условиях должно вести себя аналогичным или похожим образом.

Преимущества оценки показателей надежности по имитационной модели, создаваемой на основе анализа структуры будущего реального ПС, заключаются в следующем:

- модель позволяет на этапе проектирования ПС принимать оптимальные проектные решения, опираясь на характеристики ошибок, оцениваемые с помощью имитационной модели;

- модель позволяет прогнозировать требуемые ресурсы тестирования;

- модель дает возможность определить меру сложности программ и предсказать возможное число ошибок и т.д.

К недостаткам можно отнести высокую стоимость метода, так как он требует дополнительных затрат на составление имитационной модели, и приблизительный характер получаемых показателей.

7.4.2. Модель, определяющая время доводки программ

Эта модель используется для ПС, которые имеют иерархическую структуру, т.е. ПС как система может содержать подсистемы, которые состоят из компонентов, а те, в свою очередь, состоят из V модулей. Таким образом, ПС может иметь V различных уровней композиции. На любом уровне иерархии возможна взаимная зависимость между любыми парами объектов системы. Все взаимозависимости рассматриваются в терминах зависимости между парами модулей.

Анализ модульных связей строится на том, что каждая пара модулей имеет конечную (возможно, нулевую) вероятность, изменения в одном модуле вызовут изменения в другом модуле.

Данная модель позволяет на этапе тестирования, а точнее при тестовой сборке системы, определять возможное число необходимых исправлений и время, необходимое для доведения ПС до рабочего состояния.

Основываясь на описанной процедуре оценки общего числа изменений, требуемых для доводки ПС, можно построить две различные стратегии корректировки ошибок:

- фиксировать все ошибки в одном выбранном модуле и устранить все побочные эффекты, вызванные изменениями этого модуля, отрабатывая таким образом последовательно все модули;

- фиксировать все ошибки нулевого порядка в каждом модуле, затем фиксировать все ошибки первого порядка и т.д.

Исследование этих стратегий доказывает, что время корректировки ошибок на каждом шаге тестирования определяется максимальным числом изменений, вносимых в ПС на этом шаге, а общее время – суммой максимальных времен на каждом шаге.

Это подтверждает известный факт, что тестирование обычно является последовательным процессом и обладает значительными возможностями для параллельного исправления ошибок, что часто приводит к превышению затрачиваемых на него ресурсов над запланированными.

7.5. Сертификация средств информатизации в Российской Федерации

7.5.1. Основные понятия в области сертификации

Рынок средств и систем информатизации в России сейчас настолько разнообразен, что в подавляющем большинстве случаев потребитель не в состоянии самостоятельно убедиться в соответствии приобретаемой им продукции установленным на государственном уровне нормам и правилам.

На бытовом уровне логичным путем решения этой проблемы является обращение к некоторому третьему лицу, являющемуся специалистом в данной области и заведомо независимому от поставщика продукции, которое может дать заключение о соответствии продукции установленным требованиям. На государственном уровне аналогичная процедура называется сертификацией.

Сертификация — процедура, выполняемая третьей стороной, независимой от изготовителя (продавца) и потребителя продукции или услуг, по подтверждению соответствия этих продукции или услуг установленным требованиям.

Результатом выполнения процедуры сертификации является так называемый сертификат соответствия.

Сертификат соответствия — документ, выданный по правилам системы сертификации для подтверждения соответствия сертифицированной продукции установленным требованиям.

Общие правовые основы сертификации продукции и услуг в Российской Федерации установлены Законом «О сертификации продукции и услуг», где определены права и ответственность в области сертификации органов государственного управления, а также изготовителей (продавцов, исполнителей) и других участников сертификации.

В этом Законе, в частности, указано, что сертификация проводится в целях:

- создания условий для деятельности предприятий, учреждений, организаций и предпринимателей на едином товарном рынке Российской Федерации, а также для участия в международном экономическом, научно-техническом сотрудничестве и международной торговле;
- содействия потребителям в компетентном выборе продукции;
- защиты потребителя от недобросовестности изготовителя (продавца, исполнителя);

- контроля безопасности продукции для окружающей среды, жизни, здоровья и имущества;
- подтверждения показателей качества продукции, заявленных изготовителем.

Сертификация средств и систем информатизации является элементом общей системы сертификации продукции в Российской Федерации.

Основными целями сертификации средств информатизации, информационных технологий и услуг являются:

1. защита пользователей средств и систем информатизации от приобретения средств и систем, в том числе импортных, которые представляют опасность для жизни, здоровья, имущества, а также для окружающей среды;
2. обеспечение разработчиков систем, а также широкого круга пользователей этих систем достоверной информацией о состоянии отечественного и зарубежного рынков средств информатизации, телекоммуникаций, информационных технологий и услуг;
3. обеспечение информационного обмена между государственными системами информатизации (налоговая служба, правоохранительные органы, службы управления трудом и занятостью, образование, здравоохранение и др.);
4. обеспечение условий для информационного взаимодействия субъектов негосударственной принадлежности с субъектами государственной принадлежности;
5. содействие повышению научно-технического уровня и конкурентоспособности отечественных систем информатизации, информационных технологий и услуг;
6. содействие созданию условий для вхождения России в мировое информационное пространство.

Необходимо отметить, что сертификация средств информатизации не только обеспечивает удовлетворение интересов потребителя, но приносит определенные выгоды и изготовителю (поставщику) продукции. Так, в частности, сертификация способствует расширению рынка сбыта (распространению продукции в тех районах, где потребителю неизвестна репутация фир-

мы) и обеспечивает подтверждение качества продукции фирмы по сравнению с продукцией конкурентов. С точки зрения организации торговых взаимосвязей сертификация способствует созданию доверительных отношений между производителями (поставщиками) и потребителями продукции. Необходимо иметь в виду, что только имеющее место и объективно подтвержденное качество конкретных видов отечественной информационной продукции и средств информатизации может сделать их конкурентоспособными и реально обеспечить спрос на них.

Говоря о сертификации, нельзя не отметить ее тесную взаимосвязь со стандартизацией в сфере информатизации.

Во-первых, как уже говорилось выше, суть процедуры сертификации заключается в подтверждении соответствия средств информатизации установленным требованиям. Документами, содержащими эти требования, являются стандарты, разрабатываемые в процессе стандартизации.

Во-вторых, собственно процедура сертификации регламентируется действующими нормативными документами (стандартами).

Таким образом, основой сертификации являются результаты стандартизации. В нормативную базу сертификации средств и систем информатизации, информационных технологий и услуг включаются три группы документов:

- нормативные документы на объекты сертификации, где устанавливаются характеристики объектов, подтверждаемые при сертификации;
- нормативные документы на методы испытаний для оценки характеристик объектов сертификации;
- нормативные документы, регламентирующие процедуры сертификации.

В целом стандартизация вместе с сертификацией образуют единый процесс управления качеством средств, систем и технологий в области информатизации, одной из основных целей которого является защита интересов потребителя.

Ниже приводятся еще несколько терминов, знание которых необходимо для понимания сущности процедуры сертификации.

Система сертификации — система, располагающая собственными правилами процедуры и управления для проведения сертификации.

Орган по сертификации — орган, проводящий сертификацию соответствия. Орган по сертификации может сам проводить испытания или же осуществлять надзор за этой деятельностью, проводимой по его поручению другими органами.

Испытательная лаборатория — лаборатория (центр), который проводит испытания в процессе сертификации.

Аккредитация (испытательной лаборатории или органа по сертификации) — процедура, посредством которой уполномоченный в соответствии с законодательными актами Российской Федерации орган официально признает возможность выполнения испытательной лабораторией или органом по сертификации конкретных работ в заявленной области.

Знак соответствия (в области сертификации) — защищенный в установленном порядке знак, применяемый или выданный в соответствии с правилами системы сертификации, указывающий, что обеспечивается необходимая уверенность в том, что данная продукция, процесс или услуга соответствует конкретному стандарту или другому нормативному документу.

В Законе «О сертификации продукции и услуг» определены два вида сертификации: *обязательная* и *добровольная*. Обязательной сертификации подлежит продукция, включенная в перечни, определяемые соответствующими нормативными документами.

Организационная структура системы сертификации в России включает: государственный (национальный) орган по сертификации, ведомственные органы по управлению сертификацией продукции определенных классов, а также испытательные центры (лаборатории). Основными функциями государственного органа по сертификации являются организация, координация, научно-методическое, информационное и нормативно-техническое обеспечение работ по испытаниям и сертификации, а также аккредитация центров сертификационных испытаний в соответствии с полномочиями национального органа по сертификации. Ведомственные органы сертификации выполняют те же функции в ограниченном объеме для конкретных видов продукции.

Национальным органом по сертификации продукции в Российской Федерации является Госстандарт России, который осуществляет следующие функции:

1. организует ведение обязательной сертификации продукции по поручению органов законодательной или исполнительной власти;

2. организует и финансирует разработку, а также утверждает основополагающие нормативно-технические и методические документы системы сертификации;

3. утверждает документы, устанавливающие порядок сертификации конкретных видов продукции;

4. проводит аккредитацию испытательных центров (лабораторий) совместно с ведомственными органами по сертификации и выдает аттестат аккредитации;

5. признает иностранные сертификаты соответствия, осуществляет взаимодействие с соответствующими уполномоченными органами других стран и международных организаций по вопросам сертификации;

6. регистрирует и аннулирует сертификаты соответствия и сертификационные лицензии, рассматривает спорные вопросы, возникающие в процессе сертификации;

7. организует периодическую публикацию информации по сертификации.

Основой сертификации продукции в Российской Федерации является *Система сертификации ГОСТ Р Госстандарта России.* Этой системой, в частности, определяются правила создания и регистрации ведомственных систем сертификации для конкретных классов продукции.

7.5.2. Организация работ по сертификации средств информатизации в РФ

В соответствии с действующими законодательными и нормативными документами сертификация средств информатизации проводится в Российской Федерации в следующих основных направлениях:

- обязательная сертификация средств информатизации на соответствие требованиям электромагнитной совместимости, а также требованиям, обеспечивающим безопасность жизни, здоровья, имущества потребителей и охрану среды обитания;

- обязательная сертификация средств защиты информации;

- добровольная сертификация функциональных параметров средств и систем информатизации, по номенклатуре и характеристикам, устанавливаемым отраслевыми (фирменными) стандартами, и учитывающим различные аспекты применения аппаратуры и программного обеспечения.

Добровольная сертификация применяется для средств информатизации, не подлежащих в соответствии с законодательными актами Российской Федерации обязательной сертификации, и проводится по требованиям, на соответствие которым законодательными актами Российской Федерации не предусмотрено проведение обязательной сертификации.

Добровольная сертификация проводится для удостоверения качества средств и систем информатизации с целью повышения их конкурентоспособности, расширения сферы использования и получения дополнительных экономических преимуществ.

В общем случае упрощенную схему добровольной сертификации можно представить следующим образом. Необходимость добровольной сертификации обычно определяет разработчик или поставщик средств информатизации, руководствуясь при этом указанными выше соображениями. Разработчик или поставщик обращается в аккредитованный в установленном порядке сертификационный центр и финансирует проведение работ по сертификации. Совокупность и значения показателей качества, по которым проводится сертификация, формируются совместно заявителем и сертификационным центром. При положительных результатах испытаний средств информатизации, представленных для сертификации, заявитель получает сертификат соответствия, который используется, например, для рекламы при взаимодействии с потенциальным пользователем или потребителем. Последние не имеют непосредственных контактов с сертификационным центром. В случае выявления недостатков в сертифицированном изделии они обращаются непосредственно к поставщику, который обязан обеспечить доработку и повторные сертификационные испытания.

В соответствии с действующим законодательством добровольная сертификация средств информатизации может проводиться как в уже упоминавшейся нами Системе сертификации ГОСТ Р, так и в других системах сертификации, зарегистрированных Госстандартом России в установленном порядке.

Для удостоверения качества, надежности и безопасности применения сложных, критических ИС используемые в них ПС следует подвергать обязательной сертификации аттестованными, проблемно-ориентированными испытательными лабораториями. Такие испытания необходимо проводить, когда программы управляют сложными процессами или обрабатывают столь важную информацию, что дефекты в них или недостаточное качество могут нанести значительный ущерб.

Сертификационные испытания должны устанавливать соответствие комплексов программной документации и допускать их к эксплуатации в пределах изменения параметров внешней среды, исследованных при проведенных проверках. Эти виды испытаний характеризуются наибольшей строгостью и глубиной проверок и должны проводиться специалистами, не зависимыми от разработчиков и заказчиков (пользователей). Испытания ПС должны опираться на стандарты, формализованные методики и нормативные документы

разных уровней. Множество видов испытаний целесообразно упорядочивать и проводить поэтапно в процессе разработки для сокращения затрат на завершающихся сертификационных испытаниях.

Сертификация комплексов программ является их испытанием в наиболее жестких условиях тестирования особым третейским коллективом специалистов, имеющим право на официальный государственный или ведомственный контроль функций и качества ПС и гарантирующим их соответствие стандартам и другим нормативным документам, а также надежность и безопасность применения. Получение и обобщение результатов испытаний, а также принятие решения о выдаче сертификата являются прерогативой испытательных лабораторий. Они должны быть специализированными для проведения испытаний объектов определенных классов, целенаправленно и систематически работать по созданию и совершенствованию методик и средств автоматизации испытаний ПС конкретного функционального назначения.

Специалисты-сертификаторы имеют право на расширение условий испытаний и на создание различных критических и стрессовых ситуаций в пределах нормативной документации, при которых должны обеспечиваться заданное качество и надежность решения предписанных задач. Если все испытания проходят успешно, то на соответствующую версию ПС оформляется специальный документ - сертификат соответствия. Этот документ официально подтверждает соответствие стандартам, нормативным и эксплуатационным документам функций и характеристик испытанных средств, а также допустимость их применения в определенной области.

Методология принятия решений о допустимости выдачи сертификата на ПС определяется оценкой степени его соответствия действующим и/или специально разработанным документам. В исходных нормативных документах должны быть сосредоточены все функциональные и эксплуатационные характеристики ПС, обеспечивающие заказчику и пользователям возможность корректного применения сертифицированного объекта во всем многообразии его функций и показателей качества. Выбор и ранжирование показателей должны проводиться с учетом классов ПС, их функционального назначения, режимов эксплуатации, степени критичности и жесткости требований к результатам функционирования и проявлениям возможных дефектов и ошибок. При этом могут привлекаться документы предшествующих этапов испытаний и документы, подтверждающие соблюдение аттестованных технологий при разработке программ на всех этапах. Испытания ПС в конкретных проблемно-ориентированных системах проводятся по правилам и методикам, принятым для соответствующих классов критических информационных систем, например, авиационных или космических комплексов.

Работы по сертификации объединяются в технологический процесс, на каждом этапе которого регистрируются документы, отражающие состояние и качество результатов разработки ПС. В зависимости от характеристик объекта сертификации на ее выполнение выделяются ресурсы различных видов. В результате сложность программ, а также доступные для сертификации ресурсы становятся косвенными критериями или факторами, влияющими на выбор методов испытаний, а также на достигаемые качество и надежность ПС.

Сертификационные испытания удостоверяют качество и надежность ПС только в условиях, ограниченных конкретными стандартами и нормативными документами, с некоторой конечной вероятностью. В реальных условиях эксплуатации принципиально возможны отклонения от характеристик внешней среды функционирования ПС за пределы, ограниченные сертификатом, и ситуации, не проверенные при сертификационных испытаниях. Эти обстоятельства способны вызывать катастрофические последствия, угрожающие надежности функционирования и безопасности применения ПС. Наличие сертификата у ПС для критических систем является необходимым условием их допуска к эксплуатации. Однако любой сертификат на сложные системы не может гарантировать абсолютную их надежность применения, и всегда остается некоторый риск возникновения отказовых ситуаций.

Отсутствие гарантии достижения в процессе создания ПС абсолютной надежности их функционирования за счет использования высоких технологий, тестирования и сертификации заставляет искать дополнительные методы и средства повышения надежности ПС. Для этого разрабатываются и применяются методы оперативного обнаружения дефектов и искажений при исполнении программ путем введения в них временной, информационной и программной избыточности. Эти же виды избыточности используются для оперативного восстановления искаженных программ и данных и предотвращения возможности развития результатов реализации угроз до уровня, нарушающего надежность функционирования ПС. Основная задача ввода избыточности состоит в ограничении или исключении возможности аварийных последствий от возмущений, соответствующих отказу системы. Любые аномалии при исполнении программ необходимо блокировать и по возможным последствиям сводить до уровня сбоя путем быстрого восстановления.

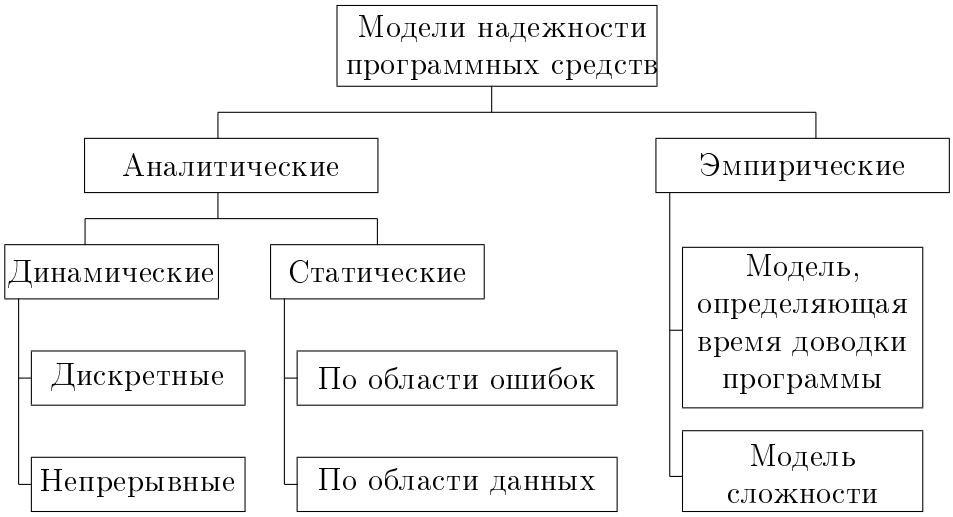

Рис. 7.1. Классификационная схема моделей надежности ПС

Глава 8

Тестирование ПО

8.1. Определение и принципы тестирования

Тестирование является одним из этапов жизненного цикла ПИ, направленным на повышение качественных характеристик. При создании типичного ПИ около 40% общего времени и более 40% общей стоимости расходуется на проверку (тестирование) разрабатываемой программы.

Программы как объекты тестирования имеют ряд особенностей, которые отличают процесс их тестирования от общепринятого, применяемого при разработке аппаратуры и других технических изделий. Особенностями тестирования ПИ являются:

- отсутствие эталона (программы), которому должна соответствовать тестируемая программа;
- высокая сложность программ и принципиальная невозможность исчерпывающего тестирования;
- практическая невозможность создания единой методики тестирования (формализации процесса тестирования) в силу большого разнообразия ПИ по их сложности, функциональному назначению, области использования и т.д.

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

Общепринятое мнение, что тестирование — это процесс, демонстрирующий отсутствие ошибок в программе или доказывающий корректность выполняемых программой функций — является не просто ошибочным, но и крайне вредным, так как это нечто противоположное тому, что следует понимать под тестированием.

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

Цель тестирования — выявление как можно большего числа ошибок.

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. 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:

1. описание входных данных
2. 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. Из рассмотренного принципа, который трудно реализуем, но которого следует придерживаться логически, вытекает следующий.

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.

8.2. 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.

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.

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

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

Большая трудоемкость тестирования и ограниченные ресурсы приводят к необходимости систематизации процесса и методов тестирования.

Рассмотрим подробнее последовательно применяемые методы тестирования:

1. статический,
2. детерминированный,
3. стохастический
4. в реальном масштабе времени.

Каждый метод тестирования предполагает использование конкретных процедур для реализации.

8.2.1. Статическое тестирование

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.

Статическое тестирование реализуется путем применения ручных методов тестирования программ. Они получили развитие с начала 70-х годов, когда в программировании были введены требования структурного и модульного написания программ. Это сделало программы удобочитаемыми и позволило проводить тестирование отдельных модулей вручную без использования ЭВМ.

Использование ручных методов тестирования достаточно эффективно. Для типичных программ, по данным фирмы IBM, можно находить от 30 до 80% ошибок логического проектирования и кодирования. Эти методы способствуют существенному увеличению производительности и повышению надежности программ, позволяют раньше обнаружить ошибки, а значит, уменьшить стоимость исправления. При ручных методах тестирования вероятность того, что при исправлении ошибок не вносятся новые ошибки, намного выше.

Основные методы ручного тестирования:

1. инспекции исходного текста

2. сквозные просмотры.

Они имеют общее: предполагают некоторую подготовительную работу и проведение собрания, состоящего из участников проверки, цель которого нахождение ошибок, но не их устранение (т.е. тестирование, а не отладка).

Рассмотрим каждую из предлагаемых процедур.

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

В инспектирующую группу входят обычно 4 человека:

1. председатель (не автор, но знакомый с деталями программы); В функции председателя входит:

   - подготовка материалов для заседания инспектирующей группы;

- составление графика проведения инспекций;
- ведение заседания;
- регистрация всех найденных ошибок;

2. автор программы;

3. проектировщик;

4. специалист по тестированию.

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. Предлагаемый список содержит около 70 вопросов, сгруппированных в 8 групп.

Председатель должен нести ответственность за результативность дискуссии, а участники должны сосредоточить свое внимание на поиске ошибок, а не на их корректировке. По окончании заседания автору программы передается список найденных ошибок. Если он очень велик и требует больших доработок, то может быть принято решение повторной инспекции. Оптимальная продолжительность заседания 90-120 мин. Скорость просмотра 150 операторов в час.

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).

Описанные методы рекомендуется использовать не только для тестирования новых программ, но и для модифицируемых программ, т.е. они эффективны как при разработке ПИ, так и на этапе эксплуатации при проведении работ по сопровождению.

8.2.2. Детерминированное тестирование

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

Детерминированное тестирование в силу трудоемкости возможно применять для отдельных модулей в процессе сборки программы или для небольших и несложных программных комплексов.

Детерминированное тестирование, или тестирование на определенных входных значениях, основывается на двух подходах:

1. структурное тестирование (СТ);
2. функциональное тестирование (ФТ).

Структурное тестирование, или тестирование программ как «белого ящика» (стратегия тестирования, управляемого логикой программы), предполагает детальное изучение текста (логики) программы и построение (подбор) таких входных наборов данных, которые позволили бы при многократном выполнении программы на ЭВМ обеспечить выполнение максимально возможного количества маршрутов, логических ветвлений, циклов и т.д.

Функциональное тестирование, или тестирование программ как «черного ящика» (тестирование по «входу-выходу»), полностью абстрагируется от логики программы, предполагается, что программа — «черный ящик», а тестовые наборы выбираются на основании анализа входных функциональных спецификаций. Т.е. при функциональном тестировании исходной информацией для построения тестовых наборов данных являются функциональные спецификации программы.

Для успешного и качественного проведения детерминированного тестирования необходимо разработать эффективные тестовые наборы данных.

Понятие «эффективного» тестового набора данных связано с невозможностью «полного» тестирования программы. Поскольку тестируемые программы состоят из множества сложных участков, то исчерпывающее тестирование маршрутов невыполнимо и невозможно.

Целью отбора тестовых наборов данных является попытка уменьшить эту «неполноту». Если ввести ограничения на время, стоимость, машинное время и т.п., то основным вопросом детерминированного тестирования становится вопрос о том, какое подмножество всех возможных тестов имеет наивысшую вероятность обнаружения большинства ошибок.

Подмножество всех возможных тестов, которое обладает этим свойством, т.е. имеет наивысшую вероятность обнаружения большинства ошибок, называется эффективным.

Чтобы разработать эффективный тестовый набор, необходимо знать ряд методов его построения и придерживаться определенных правил и рекомендаций. При построении тестовых наборов данных по принципу «белого ящика» руководствуются следующими критериями:

1. покрытие операторов;
2. покрытие узлов ветвления;
3. покрытие условий;
4. комбинаторное покрытие условий.

Рассмотрим подробнее каждый из перечисленных критериев.

Покрытие операторов. Этот критерий предполагает выбор такого тестового набора данных, который вызывает выполнение каждого оператора в программе хотя бы один раз. Критерий очень слабый, является не только необходимым, но и недостаточным условием тестирования.

Покрытие узлов ветвления (покрытие решений). Этот критерий предполагает разработку такого количества тестов, чтобы в каждом узле ветвления был обеспечен переход по веткам «истина» и «ложь» хотя бы один раз.

Покрытие решений обычно удовлетворяет критерию покрытия операторов, однако нельзя забывать о некоторых исключениях из этого правила. Например, если программа имеет несколько точек входа, то данный оператор выполняется только в том случае, если выполнение программы начинается с соответствующей точки входа.

Покрытие условий. Если узел ветвления содержит более одного условия, тогда нужно разработать число тестов, достаточное для того, чтобы возможные результаты каждого условия в решении выполнялись, по крайней мере,

один раз; каждой точке входа в программу, а также ON-единицам должно быть передано управление при вызове, по крайней мере, один раз. Этот критерий обычно называют покрытием условий.

Пример 1. Узел ветвления: $C > 2$ and $a = b$. Тесты, разработанные по критерию покрытия условий:

1. $C = 3$; $a = 0$; $b = 0$
2. $C = 2$; $a = 0$; $b = 1$

Однако рассмотренные критерии недостаточно чувствительны к ошибкам в логических выражениях.

Комбинаторное покрытие условий. Этот критерий требует создания такого числа тестов, чтобы все возможные комбинации результатов условия в каждом решении и все точки входа и ON-единицы выполнялись, по крайней мере, один раз.

Пример 2. Узел ветвления: $C > 2$ and $a = b$. Для условия по этому критерию надо покрыть тестами следующие 4 комбинации:

1. $C = 3$; $a = 0$; $b = 0$
2. $C = 3$; $a = 0$; $b = 1$
3. $C = 2$; $a = 1$; $b = 1$
4. $C = 2$; $a = 1$; $b = 0$

К стратегии «черного ящика» относятся методы:

1. эквивалентного разбиения;
2. анализ граничных значений;
3. функциональных диаграмм.

Метод эквивалентного разбиения. Построение тестов методов эквивалентного разбиения осуществляется в 2 этапа:

- выделение классов эквивалентности;
- построение тестов.

Классом эквивалентности называют множество входных значений, каждое из которых имеет одинаковую вероятность обнаружения конкретного типа ошибки.

Классы эквивалентности выделяются путем анализа входного условия и разбиением его на две или более групп: для любого условия существуют правильный (представляющий правильные входные данные программы) и неправильный, т.е. ошибочные входные значения, классы эквивалентности.

При выделении классов эквивалентности целесообразно придерживаться следующих правил.

1. Если входное условие описывает область значений (например, идентификатор метки может содержать от одного до восьми символов), то определяются один правильный класс эквивалентности ($1 \leq$ количество символов идентификатора метки ≤ 8) и два неправильных класса эквивалентности (ни одного и более 8).

2. Если входное условие описывает конечное число конкретных значений и есть основание полагать, что каждое значение программа трактует особо (например, способ передачи информации между городами допустим - почтовый, телеграфный, телетайпный), то определяются правильный класс эквивалентности для каждого значения и один неправильный класс эквивалентности (например, «курьерный»).

3. Если входное условие описывает ситуацию «должно быть» (например, «первым символом идентификатора должна быть буква»), то определяются один правильный класс эквивалентности и один неправильный.

На основе классов эквивалентности строятся тестовые наборы. Каждый тест должен покрывать по возможности большее число правильных классов эквивалентности. Для каждого неправильного класса эквивалентности строится хотя бы один тестовый набор.

Анализ граничных значений. Этот метод предполагает исследование ситуаций, возникающих на границах и вблизи границ эквивалентных разбиений. Например, если правильная область значений есть -1.0 до +1.0, то нужно предусмотреть тесты -1.0, 1.0,-1.001 и 1.001.

Метод функциональных диаграмм. Метод заключается в преобразовании входной спецификации программы в функциональную диаграмму (диаграмму причинно–следственных связей) с помощью простейших булевских отношений, построения таблицы решений (методом обратной трассировки),

которая является основой для написания эффективных тестовых наборов данных.

1. В спецификации программы выделяются причины и следствия. Причины — это отдельное входное условие, или класс эквивалентности входных условий. Следствие — выходное условие, или результат преобразования системы. Каждой причине и следствию приписывается уникальный номер.

2. Анализируется семантическое содержание спецификации, которая преобразуется в булевский граф, связывающий причины и следствия. Каждая вершина графа может находиться в состоянии «истина» (1) или «ложь» (0).

3. Диаграмма снабжается примечаниями, задающими ограничения и описывающими комбинации причин и (или) следствий, которые являются невозможными из-за синтаксических или внешних ограничений.

4. По полученной функциональной диаграмме строится таблица решений. Для этого поочередно для каждого следствия, значение которого условно устанавливается в 1, прослеживается обратный путь (по диаграмме) ко всем причинам, связанным с этим следствием, и фиксируется их состояние. Каждый столбец таблицы решений соответствует тесту.

5. Столбцы решений преобразуются в тесты.

8.2.3. Стохастическое тестирование и тестирование в реальном масштабе времени

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.

8.3. Сборка программ при тестировании

Разработка больших, многомодульных программных комплексов требует использования специальных способов сборки их при тестировании.

Прежде чем приступить к тестированию программного комплекса в целом, нужно, чтобы составляющие его части (отдельные модули или набор модулей) были тщательно оттестированы.

Сборка модулей в программный комплекс может осуществляться двумя методами:

1. Монолитным.

2. 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.

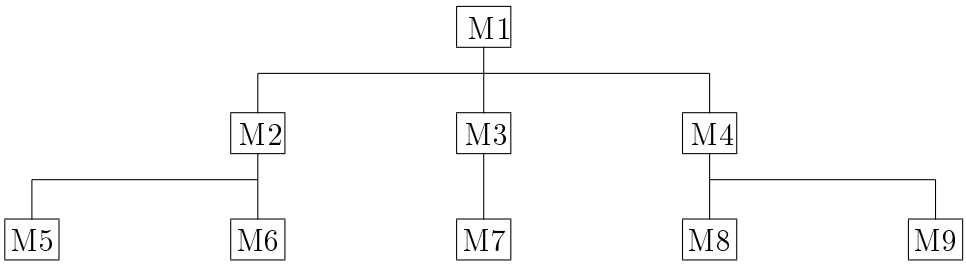

Рис. 8.1. Структура программы из девяти модулей

В качестве примера рассмотрим программу, состоящую из 9 модулей, структура которой представлена на рисунке 8.1.

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).

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

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

Для рассматриваемого примера модули-драйверы нужны для всех модулей, кроме модуля $M1$, а модули-заглушки нужны для всех модулей, кроме $M5$, $M6$, $M7$, $M8$, $M9$ (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.

Теперь будем использовать пошаговый метод сборки. Предположим, что тестируем сверху-вниз. Тогда для модуля $M1$ нужно разработать 3 заглушки. Далее подключается реальный модуль $M2$, для которого нужно предварительно разработать 2 заглушки, и тестируются $M1 - M2$. Затем заглушка $M5$ заменяется реальным модулем $M5$ и тестируется цепочка $M1 - M2 - M5$. Процесс продолжается до тех пор, пока не будет собран весь комплекс. Есть возможность некоторого распараллеливания работ и автономного тестирования цепочек (рисунок 8.2).

Ясно, что при пошаговой сборке сверху-вниз нужно разработать 9 заглушек, но не нужны драйверы.

При тестировании снизу-вверх процесс организуется следующим образом: тестируются модули низшего уровня — $M5$, $M6$, $M7$, $M8$, $M9$. Для каждого из них нужен драйвер. Далее параллельно можно проводить тестирование $M5 - M2$, $M6 - M2$, $M7 - M3$, $M8 - M4$, $M9 - M4$. Затем подключить $M1$

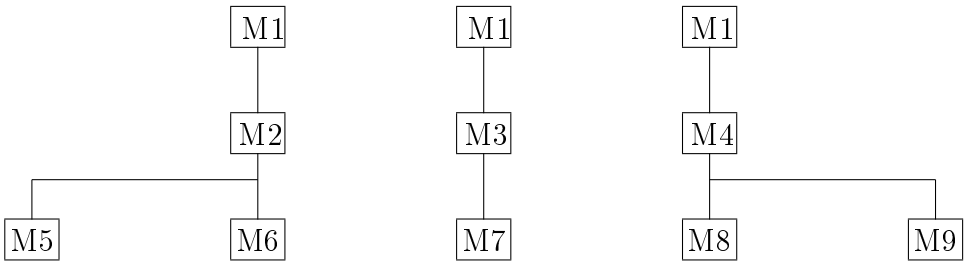

Рис. 8.2. Параллельное тестирование программы

и провести комплексное тестирование всей программы. Таким образом, при восходящем тестировании нужно будет разработать максимум 8 драйверов, но не нужны заглушки.

Сравнивая монолитную и пошаговую сборки программ, можно отметить ряд достоинств и недостатков каждого из них.

Монолитная сборка требует больших затрат, так как предполагает дополнительно разработку драйверов и заглушек, в то время как при пошаговой сборке разрабатываются либо только заглушки (сверху-вниз), либо только драйверы (снизу-вверх).

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.

Выбор способа сборки программ зависит от конкретных условий разработки и особенностей тестируемого программного комплекса.

8.4. Критерии завершенности тестирования

При проведении тестирования встает вопрос о том, когда завершить тестирование, когда разрабатываемое ПС достигло того уровня надежности, которое может удовлетворить будущих пользователей.

В основном на практике придерживаются следующих двух критериев:

- когда время, отведенное по графику на тестирование, истекло;
- когда все тесты выполняются без выявления ошибок (т.е. оказались неудачными).

Оба этих критерия недостаточно точны и логичны, так как первому можно удовлетворить, ничего не делая, а второй зависит от качества тестового набора данных.

Иногда используют критерий, основанный в значительной степени на здравом смысле и информации о количестве ошибок, полученных в процессе тестирования. Для этого строят график зависимости количества ошибок и времени их появления. По форме полученной кривой можно определить, стоит продолжать тестирование или нет:

Если с увеличением времени тестирования число ошибок растет, то естественно, что тестирование необходимо продолжать. Если в процессе тестирования в определенный момент наступило снижение числа выявленных ошибок, постепенно стремится к нулю или достигло нуля, то понятно, что процесс тестирования можно завершать.

Более прогрессивным может считаться подход, при котором для определения критерия завершенности тестирования используются количественные показатели надежности, рассчитываемые по моделям надежности.